Mini Research Project: Rural street networks

Felix J. Meigel

March 6, 2023

1 Project summary

Transportation on networks is limited by physical constraints set by the network morphology. Both for optimising transport and for the repairing of corrupted networks, insight into the fundamental characteristics of transportation networks is demanded. Here, we analyse rural street networks, where small villages are connected by few large roads, and identify crucial links for transportation inside and between villages. Specifically, we analyse three village regions in Germany and identify roads with a high impact on transportation when blocked. We generalise these findings using a simple random model for rural street networks. Furthermore, we identify which properties of real rural street networks are effectively captured by the random network model and the limitations of the random network model. Having identified general building principles in rural street networks, we speculate about implications for other transportation networks.

General remark about the format of this report

This report is written in a descriptive style. As such, the figures of this report are extensively showcasing the different network statistics. This results in a large degree of redundancy in this report. While the degree of redundancy could be reduced using summary statistics, in this report, it was decided against this approach, as subtle differences cannot be explored. Furthermore, the similarities of the different networks are more prominently showcased for the style chosen in this report.

2 Motivation

Transport processes on networks are ubiquitous in our daily live. Examples range from technical applications like street networks [10; 1; 8] and sewerage networks [9; 8] to biological transportation networks like the blood vasculature [7; 4] and leaf venation patterns[5; 3; 6]. In all applications, network morphology sets fundamental physical limits on transportation capacities. Moreover, corruption of the network may hinder transport drastically with sometimes severe consequences, like strokes or delayed delivery of goods. To optimise transport and to respond quickly to network corruptions, the characteristics of transport networks need to be understood.

In this project, we focus on a specific type of street network: Rural street networks connect small and medium-sized villages. In contrast to other transportation networks, rural street networks inherently have two distinct spatial scales: the spatial scale of the village and the average distance between villages. In villages, the street network is typically dense and consists of many small streets, while few long roads connect villages.

Here, we analyse three real world examples from rural regions in Germany and contrast the results with a random network model. The regions in Germany differ in spatial size and the number of villages included in the analysis. Also, for the random network model, different village sizes and the number of village centers are investigated. In contrast to the real world villages, which are structures grown over time, the random network model is created *ad hoc*, as randomly distributed junction points are connected by a Gabriel graph.

On both the real world networks and the random network model, we compute distance statistics, that reflect the heterogeneity of the networks. Building up on this insight, we turn to quantify the betweenness centrality in the networks and identify roads connecting villages as streets with high betweenness centrality. Moreover, we find a heterogeneity of betweenness centrality inside villages, where paths of high centrality cross the villages. We employ two different versions of the betweenness centrality, one disregarding edge weights and one includes the street length as edge weight. We contrast the results stemming from the two versions of betweenness centrality. In the last step, we identify streets that have a high impact on the average betweenness centrality in the network if blocked. Quantifying the relative change in betweenness centrality, we identify two categories of streets: Streets

Fig. 1. Annotated satellite images of the regions used for rural street network extraction Satellite images are obtained from `google maps` showing the regions used to extract the rural street networks. The region in Baden-Wuerttemberg is around the village Epfenbach, the region in Lower Saxony around the village Rethem, and the region in Saxony is around the villages Nossen and Rosswein. The included scalebar is in the units of kilometers. The extracted the rural street networks are displayed in the figures Fig. 7,8,9.

that increase and streets that decrease the average betweenness centrality when blocked. The results differ depending on whether a weighted or an unweighted version of the betweenness centrality is used. The categories are interpreted in terms of their importance for transportation in these networks. Finally, we speculate how insight into rural street networks can help to increase the understanding of other types of transportation networks like leaf venation patterns or the blood vasculature.

3 Data acquisition and random network model

3.1 Data acquisition for real world networks

For the real world rural street networks, network structures must be extracted from maps. To this end, in first step `geojson polygons` are created using `geojson.io` and then translated into a network using `OSMnx` in `python`. A `networkX` graph is created, where nodes are junctions and edges are streets. Nodes are indexed by the attribute `'pos'` denoting latitude ϕ and longitude θ of the junction nodes. The positions are then projected into the euclidean xy-plane using units of kilometers via

$$x = 40\,075\,\text{km} \cdot \cos(\phi_0 \cdot \pi/180)/360 \cdot \theta, \tag{1}$$

$$y = 113.2\,\text{km} \cdot \phi, \tag{2}$$

where $\phi_0 = (\max(\phi_j) + \min(\phi_j))/2$ is the average latitude and (ϕ_j, θ_j) denotes the junction set. The node positions are used to compute the length of the individual streets, where the length is computed as the euclidean distance between two junction nodes. Here, we approximate all streets as straight lines. The length is set as an edge attribute in the `networkX` graph. Street width is neglected in this analysis, i.e. this analysis doesn't distinguish dirt roads from highways but instead treats them equally. Also, all roads are assumed to be traversable in both directions, as the existence of one-way roads is neglected. This results in an undirected graph. Finally, self-loops, that connect individual junctions with themselves, are deleted. Here, three different rural regions in Germany are analysed, which vary in the number of villages considered and the size of the villages. The regions analysed in this report are

1. the villages Epfenbach, Spechbach, Lobenfeld, Waldwimmersbach, Michelbach, Reichartshausen in Baden-Wuerttemberg,
2. the region around the villages Rethem und Huelsen in Lower Saxony, and
3. the villages Rosswein and Nossen in Saxony.

Figure Fig. 1 show `google maps` satellite image excerpts of these regions. The figures Fig. 7,8,9 show the extracted rural street networks. The region in Baden-Wuerttemberg includes most villages, while the region in Lower Saxony is the smallest region both in size and in the number of streets. While both the regions in Baden-Wuerttemberg and Lower Saxony only include streets of approximately homogeneous width, the region in Saxony

includes streets of vastly different width, ranging from dirt roads to highways. Even though street width is neglected in this analysis, this results in street networks that are potentially not planar, as small roads cross under the highway without having a junction with the highway. To avoid this, only a subsection of the highway was taken into account for network extraction. As a result, all three extracted rural street networks are assumed to be presentable as planar networks.

3.2 Random network model

The random network model aims to mimic real world rural street networks. To this end, the random network model creates villages with a *dense*[1] street network, which are connected by few large roads. A crucial criterion for the random network models is the creation of planar networks, as no hierarchy between streets is assumed and no bridges over other streets are assumed to be built. Furthermore, in this model, we assume that villages in one network have all the same size.

In contrast to real world networks, which start from historical roads and grow over the years to include more and more streets [10], the random networks are built *ad hoc* following a simple algorithm:

1. C village centers are randomly distributed in circle with radius $R = \sqrt{C}$ on the xy-plane centered around the origin.
2. n junction points are distributed uniformly in circles of radius $r_0 = \sqrt{r}$ around the village centers. This results in $N = nC$ junction nodes for the full network.
3. The junction nodes are connected to a planar graph using a Delaunay triangulation.
4. From the Delaunay triangulation, the Grabriel graph is created by deleting edges.

The Delaunay triangulation connects node points such that the area enclosed by edges always has a triangular shape. The created triangles fulfill the condition that are no nodes inside the circle defined by the three nodes of the triangle. Here, the Delaunay triangulation is created to conclude on the Gabriel graph, which is a subgraph of the Delaunay triangulation. Hence the Gabriel graph can be created by deleting edges in the Delaunay triangulation. In the Gabriel graph, two nodes i, j are connected, if these nodes are the two closest nodes to the midpoint of the two respective nodes i, j. Having created the Delaunay triangulation, it's sufficient to check whether neighbours of nodes i, j in the Delaunay triangulation are closer to the midpoint i, j for each edge in the triangulation. By avoiding checking for all possible edges with all nodes, the algorithm speeds up tremendously. Finally, each edge is assigned its euclidean length as an attribute. The resulting undirected, fully connected graph is used as the random network model.

4 Definition of measures used for network analysis

To asses the importance and function of individual streets in the rural street networks, different measures are employed in this report. In this section, the different measures are defined for readability purposes.

4.1 Distance distributions

Distance distributions are statistics that allow the assessment of spatial extension and the degree of connectivity in networks. To this end, the distance between every two nodes in the network is computed. Distances from one node to itself are discarded from the statistics. For the rural street networks, the distance between two nodes can be computed using three definitions.

The first straightforward choice is the euclidean distance between every pair of nodes in the network, completely ignoring the network structure. While this statistic completely disregards information about how streets are connected, it's still capable of resolving the two intrinsic spatial scales of the rural street networks, namely the average size of the villages and the average spacing between villages. As we haven't assigned physical distances to the random network model, we measure the euclidean distance in arbitrary units. Note, that by linking the spreading of villages centers $\sqrt{C}$ to a physical distance, we can also convert this measure to physical distances.

[1] Here, we use the term *dense* loosely with no detailed mathematical description. *Density* could be interpreted as a measure for how spatially close junction nodes are to each other. Yet for this report, density is only estimated using visual impression.

A slightly more elaborate choice is measuring the distance between junctions in terms of the minimum number of streets needed to be traversed to go from one junction to the other. Via this choice, the shortest path in the unweighted network is computed. While the former statistic completely ignored the network structure, this statistic ignores the spatial information provided to the network. As such, this statistic is incapable of visualising the two intrinsic spatial scales of this network. We denote this distance measure in this report as the street distance measured in the number of streets.

A measure combining the ideas of both former measures is computing the weighted shortest path between two junctions, where the length of streets is used as weight. In this measure, the shortest paths don't necessarily have a minimal number of edges, but the cumulative weight of the path is minimised. As the first measure, this measure is capable of resolving the two intrinsic scales of the rural street networks. The distance statistics of this model are expected to yield higher values compared to the euclidean distance. We denote this distance measure in this report as the street distance measured in the same arbitrary units as the euclidean distance. Algorithms that compute the weighted and the unweighted shortest path are implemented in the `python` package `networkX` in the function `all_pairs_dijkstra_path_length`.

4.2 Centrality measures

While distance distributions give information about the network as a whole, no information about individual edges can be obtained from these measures. This can be achieved by focusing on centrality measures. While centrality measures aim to measure how "*central*" individual edges are in the network, the meaning and mathematical definition of "central" depends on the context. The centrality of nodes could be determined by their degree and the centrality of edges by the degree of the nodes connected by the edge. Another definition could be measuring how spatial close chosen edges and nodes are to the "*middle point*" of the spatially embedded network. Indeed, many more definitions could be employed.

Here, we use the edge betweenness centrality as a measure for edge centrality. The betweenness centrality measures how many shortest paths include a chosen edge in the network, normalised by the total number of possible edges in the network. Formally, the edge betweenness centrality is defined as

$$g(e;G) = \frac{1}{N(N-1)} \sum_{i,j} \frac{\sigma_{i,j}(e)}{\sigma_{i,j}}, \tag{3}$$

where the sum runs over all pairs of nodes, N is the total number of nodes in the network G, and $\sigma_{i,j}$ is the total number of shortest paths from i to j. Importantly, the definition $0/0 = 0$ is used when dealing with not fully connected graphs [2]. The edge betweenness centrality is bounded in the interval $g(e;G) \in [0,1]$, where the extremes coincide to the case that are no shortest path including e and that all shortest paths include e. The concept of edge centrality can easily be adjusted to measure the node betweenness centrality.

For computing the edge betweenness centrality, a weighted version or an unweighted version can be employed. While for the unweighted version, the shortest path is computed using the minimum number of edges between a pair of nodes, the weighted version is computes the shortest path which minimises the cumulative weight along the path. We find that the weighted and the unweighted version of the edge betweenness centrality give similar, but not identical results. Both versions are implemented in the `python` package `networkX` in the function `edge_centrality`.

In the context of street networks, the edge betweenness centrality renders more useful than the concepts of degree centrality or eigenvector centrality. The edge centrality comes with a simple interpretation in terms of shortest paths: Assuming that efficient transport relies on shortest paths and detours are avoided, more central edges will be more frequently traversed by agents transporting goods. By considering cars driving from one junction to another, both the weighted and the unweighted version of the edge betweenness centrality can be motivated. While the reduction of the total distance travelled (weighted version) is easily motivated, also the reduction in the number of junctions crossed can be motivated. Understanding that cars need to slow down at every junction to secure traffic safety, reducing the number of junctions crossed for a path could both decrease travel time and fuel consumed even if this coincides with a distance-wise longer path.

4.3 Impact measures

The edge betweenness centrality can be interpreted as how frequently roads are used. While this measure gives first information on how important certain roads might be, the betweenness centrality gives no information about how severe blockages of central roads are. To illustrate why the impact of road blockage might not be proportional

to edge centrality, imagine the following two scenarios: Parallel to a busy road is a marginally longer parallel road. Blocking the busy road results in a rerouting of the traffic on the slightly longer parallel road, with only little effects on traffic. In another case, a small village in the mountain is only connected via one road. While in a bigger context this road might not be especially central in the network, blocking this road could have severe consequences for the inhabitants of this mountain village, as they would be isolated from the rest of the world.

To estimate the effect of road blockages, we define impact measures. To this end, we delete a road from the rural street network and recalculate the edge betweenness centrality, which results in an altered centrality for every edge. Measuring this effect for every edge in the network creates a high-dimensional space, in which systematic effects are hard to recognise. Here, we reduce the complexity of this problem by focusing only on the average edge centrality. We define the impact of a blocked road as the relative change of the average betweenness centrality:

$$\delta(e) = \frac{\langle g(i; G_{\setminus e})\rangle_i - \langle g(i; G)\rangle_i}{\langle g(i; G)\rangle_i}. \tag{4}$$

Note, that in the literature alternative variants are used, where $\delta(e)$ denotes the negative relative change [10]. From this definition, a priori two different types of edges can be distinguished: edges that reduce the average centrality, and edges that increase the average centrality. Like for the centrality measure, also for the impact measure the shortest path could be computed using a weighted version or an unweighted version.

5 Evaluation of the random network model

The random network model aims to visually mimic real world rural street networks. In this section, we will describe the statistics of the random network model and check for the plausibility of the statistics, before we in detail compare the random network model with real world data in the next section. The random network model has two prominent parameters, which are the number of village centers and the size of the villages. For the random network model, the size of the village can be altered by changing the density of road - i.e. changing the number of junction nodes per village n - or changing the spatial size of the village - i.e. the sprawling radius r_0. In the subsections of this chapter, we will systematically vary these three parameters and infer their relevance for the different measures defined in section 4.

5.1 Single village network structure

Before we turn to mimic rural street networks, we focus on the village structure created by the random network model. In Figure Fig. 2 a single village with 150 junction nodes is simulated. Shown are the edge betweenness centrality for both the weighted and the unweighted case as well as the impact measure, as defined in section 4. The weight chosen is the euclidean distance between junction nodes, as described in section 3.2.

Before interpreting the network measure, we perform a visual analysis first checking for the plausibility of the created random village street structure. Notably, the village seems to have no clear structure but appears - not surprisingly - random. This contradicts our everyday experience of villages, where usually straight main roads through a village can be identified, from which side roads branch off with an angle of 90°. Adding to this, the degree distribution of the junction nodes appears to be broadly distributed. Many junctions are having a degree distribution of five or larger. This also contradicts our everyday experience, where junction degrees of three and four are predominant and junctions with a higher degree are rare. Furthermore, the random network model hardly contains any dead ends, which especially in small rural villages are frequent.

Just from visual inspection of individual villages, we conclude at this point that the random network model chosen in this report is incapable of effectively simulating village structure. While this fact should indicate a point of caution, we don't discard the model at this point without a more in-depth analysis of the model also including versions consisting of actually more than one village.

Comparing the edge betweenness centrality between the weighted and the unweighted version in figure Fig. 2, we observer a general theme that will persist for all further networks, in this report. While both the weighted and unweighted versions yield on the first glance visually similar results, comparing edges between both measures reveals partly strong deviations. Roads with a high unweighted betweenness centrality have a low weighted betweenness centrality and vice versa. Yet, roads that differ strongly between both versions are usually in close spatial proximity to other roads with a high betweenness centrality. This strengthens the importance of correctly identifying the weight used for the analysis when estimated the importance of roads in real world applications.

Focusing on the impact measure defined in equation Eq. (4), we find a qualitative difference between the weighted and the unweighted version. Notably, while the only road having a negative impact on the average

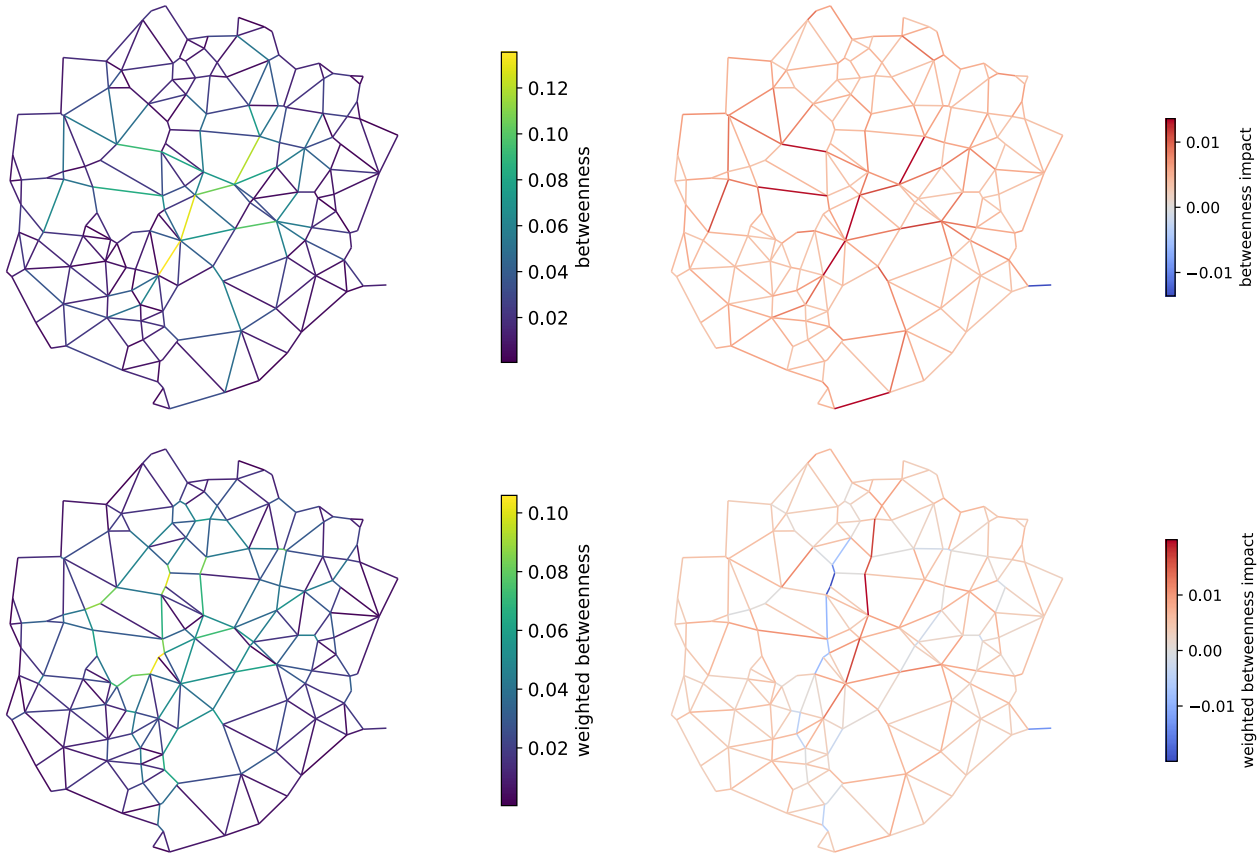


Fig. 2. Simulation of a single village using the random network model A single village is created using the random network model described in section 3.2. The network is analysed for its edge betweenness centrality, see Eq. (3), and impact, see Eq. (4) for both a weighted and unweighted version. Comparing individual edges, differences between the weighted and the unweighted version are observable. Roads with maximal betweenness centrality are not identical for both versions. While the unweighted impact measure shows only one road that decrease the average betweenness centrality, roads decreasing the average betweenness centrality are more frequent for the weighted betweenness centrality.

betweenness centrality when blocked is a dead end in the unweighted version, in the weighted version roads with a negative impact are more frequent. Inspecting the impact of road blockage in the weighted version, the roads with the strongest negative and strongest positive impact on the average centrality seem to be spatially correlated. Both the streets with the strongest negative and positive impact build paths connecting south and north of the village, which run parallel to each other. While the blocking of streets in one path decreases the average betweenness centrality, the blockage of streets in the other path increases the average betweenness centrality. We refrain from interpreting this observation at this moment and will instead analyse more networks, before coming up with interpretations in section 6.3.

5.2 Effect of the sprawling radius r on distance measures

Including more than one village in the random network model, we expect to include the two intrinsic spatial scales or rural street networks. In this section, we check how these two spatial scales are reflected in the distance measures defined in section 4. To this end, we consider the same distribution of five village centers but change the sprawling radii r.

In figure Fig. 3, six networks with different sprawling radii r are displayed, ranging from clearly separated villages to villages with overlapping village clusters. The number of junction nodes per village is held constant between villages and between different sprawling radii.

Focusing on the rural street network with the smallest sprawling radius in the top row of figure Fig. 3, we find that the euclidean distance measure shows eleven separated peaks. The largest peak is close to zero. The remaining ten peaks have approximately the same size. This multi-peaked distribution reflects the two intrinsic spatial scales of the rural street network. The largest peak reflects the euclidean distance in a single village. The ten smaller peaks reflect the distance between individual village centers. Having five villages centers, we expect $4+3+2+1 = n(n+1)/2 = 10$ different distances between villages; which perfectly coincides with the number of peaks observed for this network. Comparing with the street distance using length-weighted shortest paths, we find the same structure of a highly peaked distribution with $10+1$ peaks. Yet, the two distributions are not identical. Notably, the average distance between two junction nodes is increased using this measure in contrast to the euclidean distance. This reflects that detours have to be taken when travelling is only allowed on roads. Considering the statistics of the unweighted shortest path, we also observe a distribution with multiple peaks, yet only four overlapping peaks are identifiable. This reflects the structure of the villages, as two villages are connected to a ring of three villages via two roads. So between two villages one, two, or three long roads between villages must be traversed. This reflects in $3+1 = 4$ peaks. As the sprawling radius is increased, the peaks in all distance distributions spread out. Though peaks are still identifiable, the number of peaks differs from $10+1$ and $3+1$ peaks for larger sprawling radii r. Indeed, even for medium sprawling radii, where village centers are clearly distinguishable by eye in the mapped networks, the number of village centers cannot be reconstructed by analysing the distance distributions.

While the average euclidean distance is steadily increasing with increasing sprawling radius, the average distance is decreasing with increasing sprawling radii for the unweighted street distance and shows a non-monotonic dependency for the weighted street distance. For the last statistic, notably, the average statistic is decreased between the networks with $r = 0.2$ and $r = 0.5$. This can be understood, by observing that an additional road between the two previously unconnected village centers in the south of the region is established for $r = 0.5$.

5.3 Effect of the sprawling radius r on edge centrality and road blockage impact measure

Having identified a connection between the sprawling radius and distance measures in the previous section, in this section we investigate how the sprawling radius affects the edge betweenness centrality Eq. (3) and the impact of road blockage Eq. (4) in the random network model. Here, we focus on three different networks displaying small, intermediate, and large sprawling radii, as shown on the left, the middle, and the right column of figure Fig. 4 respectively. We check again for the weighted and the unweighted version of both the edge betweenness centrality and the impact measure. For the impact measure, we plot the results using two different colour scales. Using a multi-level colour scale, we can distinguish the strength of the impact of the blockage of individual roads. Using a binary colour scale, we distinguish whether a road blockage on average increases or decreased the betweenness centrality in the network under consideration.

As for the single village, the edge betweenness centrality shows similar results when comparing the weighted and the unweighted version. While differences are subtle for the small sprawling radius, differences can be distinguished for medium and large sprawling radii. Note, that the colour scale is not identical between different plots.

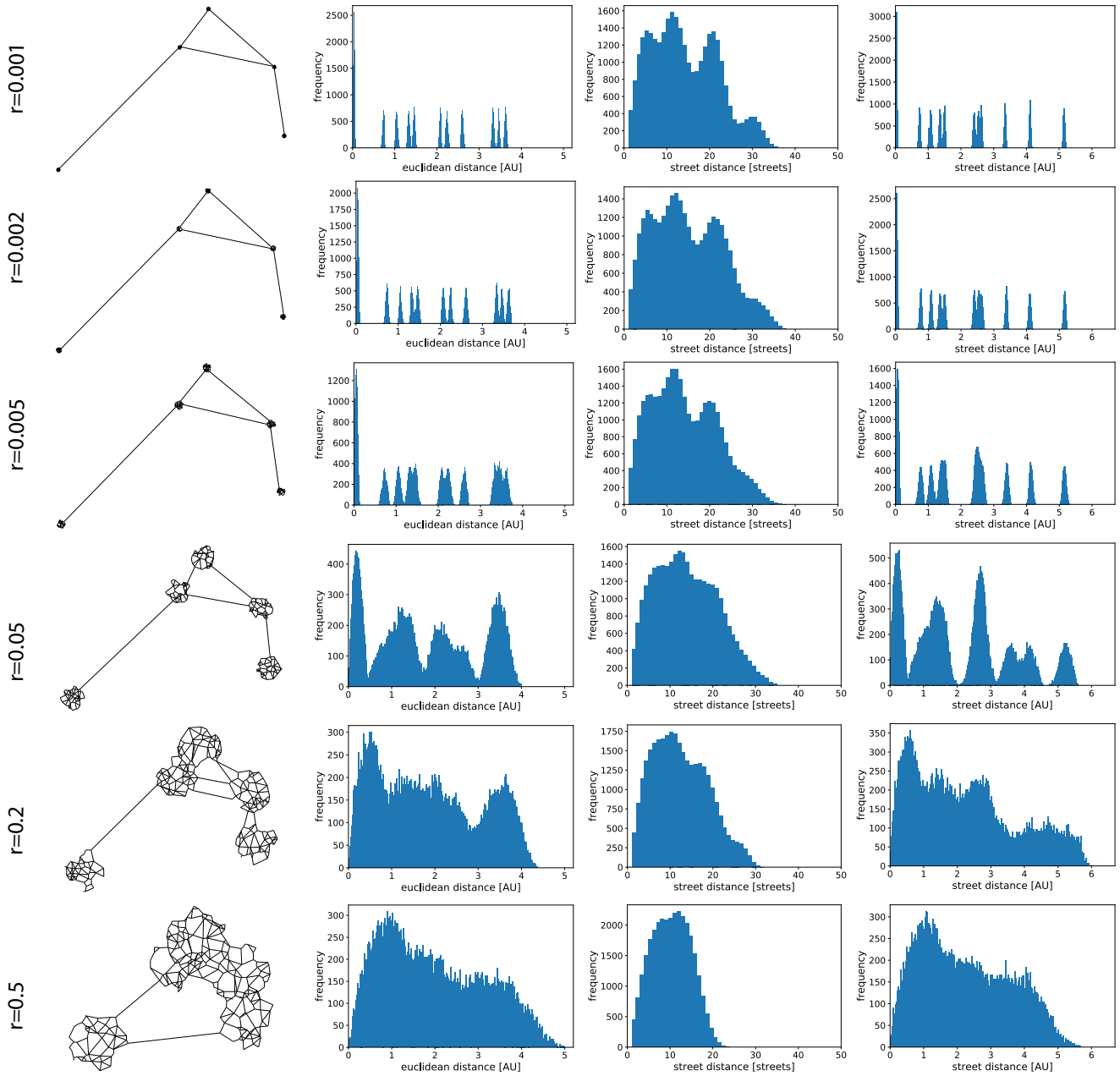


Fig. 3. Effect of the sprawling radius r on distance measures Different random street networks are created using the random network model described in section 3.2. The sprawling radius r is systematically increased, while the position of five village centers and the number of junctions nodes per village center is held constant. Shown are the spatially mapped rural street network, the euclidean distance measure, the weighted, and the unweighted street distance measure, as defined in section 4. While clearly separated peaked distribution is observable for small sprawling radii r, the peaks overlap for large sprawling radii. The distributions are interpreted in section 5.2.

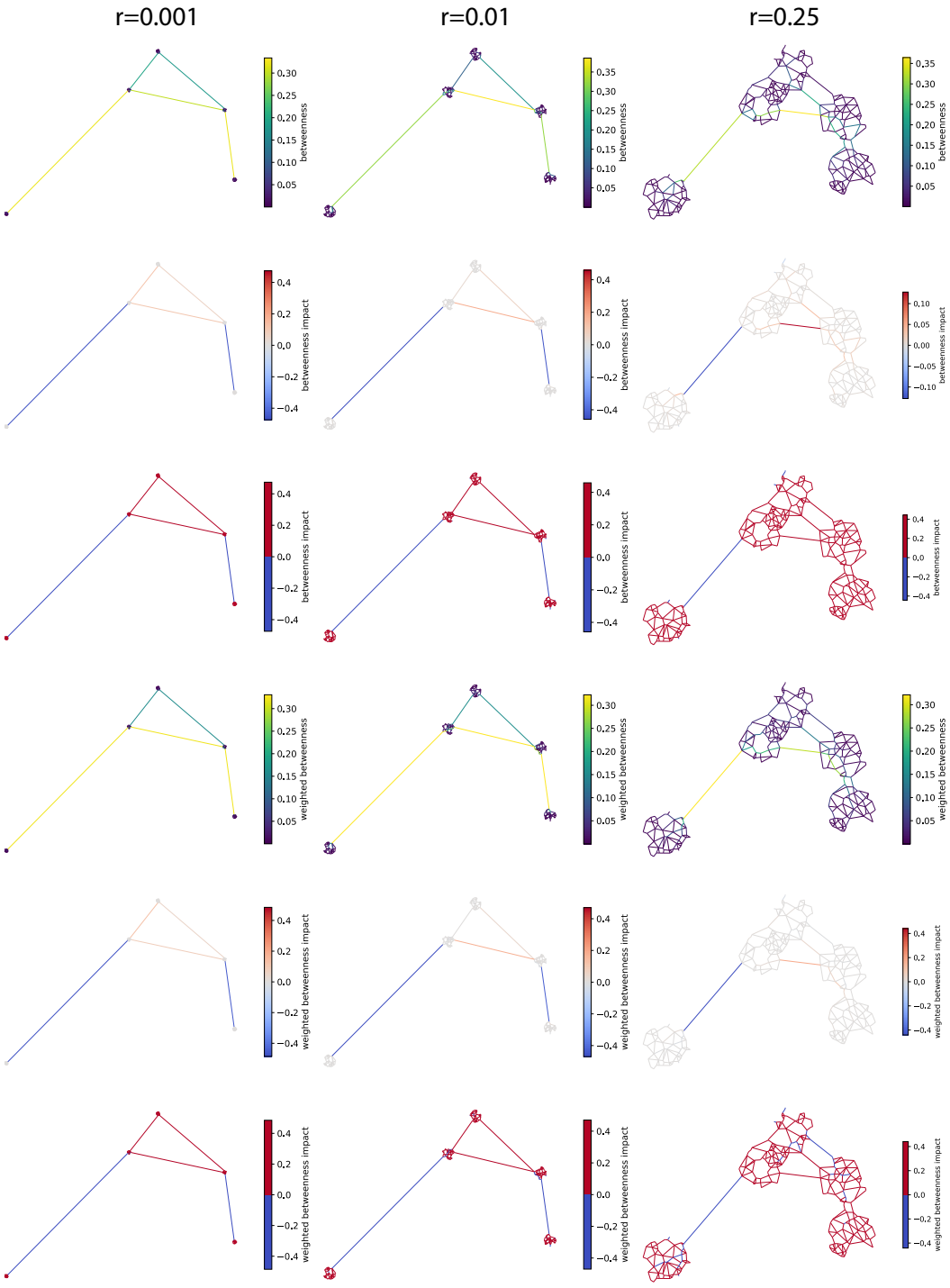


Fig. 4. Effect of the sprawling radius r on centrality measures. Different random street networks are created using the random network model described in section 3.2. The sprawling radius is systematically increased, while the position of five village centers and the number of junctions nodes per village center is held constant. The different columns exemplify small, medium, and large sprawling radii. Plotted are the edge betweenness centrality, see Eq. (3), and the impact measure for road blockage, see Eq. (4). For both measures, the weighted and the unweighted version are displayed. The spatial position of central edges is approximately conserved as the sprawling radius is increased.

As the structure of the village centers is fixed, the approximate spatial position of roads with high betweenness centrality is only slightly altered as the sprawling radius is increased.

Focusing on the impact measure, the results are qualitatively similar to the single village. While for the unweighted version roads that reduce average betweenness centrality when blocked are rare, for the weighted version these roads are more abundant. For both versions roads with a high impact connect village centers, while the impact of road blockage inside villages is small. The roads with the strongest negative impact on the average betweenness centrality are roads that cut off complete villages from the remaining network. This observation holds for both the weighted and the unweighted version of the impact measure. Also blocking dead ends has a negative effect on the average betweenness centrality for both the weighted and the unweighted impact measure.

For the large sprawling radius, two parallel roads connect the most northern village and middle village in the east. While for the unweighted version of the impact measure blocking either road increases the average betweenness centrality, for the weighted version we find a different picture. Here, blocking of one of the roads increases the average betweenness centrality, while blocking the other road decreases the average betweenness centrality. Also, some roads inside the villages decrease the averaged betweenness centrality in the weighted version but increase the averaged betweenness centrality in the unweighted version.

5.4 Effect of village size on edge centrality and road blockage impact measure

In contrast to changing the village size by changing the sprawling radius, as discussed in the previous section, in this section we change the village size by altering the number of junction nodes per village. In figure Fig. 5 three different networks mimic small villages (20 nodes), medium-sized villages (60 nodes), and large villages (100 nodes). The position of the village centers and the sprawling radius are kept constant for the different networks.

The results reproduce qualitatively the observation made in the previous section. No direct effect of the village size on the centrality measures or the impact measures are observable. Instead, we explore in this section the differences between the weighted and unweighted versions of the measures. Specifically, we focus on the structure of central edges inside the villages. Comparing with the single simulated village in section 5.1, where no paths of high centrality span the complete village, for villages connected to other villages central paths through villages appear. This can especially well be observed in the network with the medium-sized villages in figure Fig. 5. Here, the second village from the top serves as a clear example of how the weighted and unweighted versions of the edge between centrality show different edges as central. While a central path moves along the southern border of the village for the weighted version, for the unweighted version a central path runs through the village. We conclude that for distinguishing central edges inside villages, villages should be considered in a bigger context including also other villages. Notably, in the same network, parallel roads are connecting two cities. Yet, blocking of any of these two roads increases the average betweenness centrality for both roads in the unweighted as well as in the weighted version of the impact measure. This stands in contrast to the theme observed in the previous sections, where the parallel paths tend to have a qualitatively different impact on the average betweenness centrality when considering the weighted version.

5.5 Effect of village centers on edge centrality and road blockage impact measure

Finally, we explore the effect of varying the number of village centers on the edge betweenness centrality and the impact measure. Changing the number of village centers effectively changes the size of the region under consideration. Here, we explore three different networks with seven, five, and three village centers as shown in figure Fig. 6. The sprawling radius was fixed to $r = 0.1$ and the number of junction nodes per village was fixed to $n = 50$ for all three networks displayed.

Qualitative results of the previous two sections are reproduced. Notably, the network with seven village centers contains no street that could cut off a single village from the rest of the network when blocked, while such roads exist for the two smaller regions. These roads also have the highest impact in the latter two networks, and blocking those roads decreases the average betweenness centrality tremendously. As identified in the previous sections, long streets connecting villages have on averages a higher centrality in contrast to streets inside villages. This is intuitive, as the roads between cities are bottleneck when connecting junctions in different villages by their shortest path. Especially the network with seven village centers exemplifies how also the centrality of these long roads is affected by choosing either the weighted or the unweighted shortest paths. This shows how traversing different villages is preferred depending on whether the distance travelled or the number of junctions crossed is penalised.

In addition, roads connecting different village centers have a higher impact when blocked. While also streets inside villages have high betweenness centrality scores, blocking roads inside villages seem to have little impact.

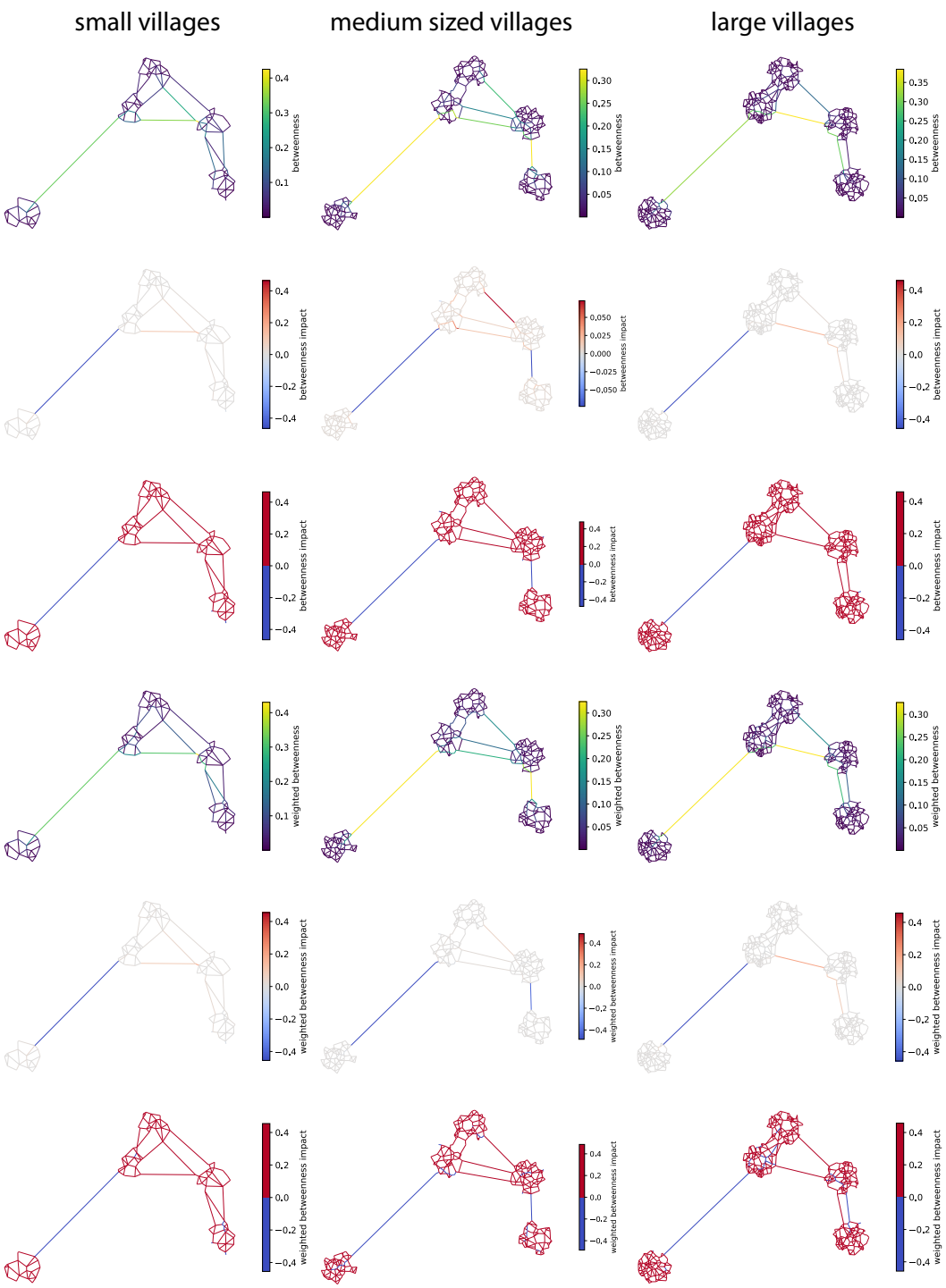


Fig. 5. Effect of village size on centrality measures. Different random street networks are created using the random network model described in section 3.2. Villages of different sizes are created as the number of junction nodes per village center is increased from small villages (20 nodes) to medium sized villages (60 nodes) to large villages (100 nodes). The position of five village centers and the sprawling radius r is held fixed. Plotted are the edge betweenness centrality, see Eq. (3), and the impact measure for road blockage, see Eq. (4). For both measures, the weighted and the unweighted version are displayed. No qualitative difference between different village sizes is observable.

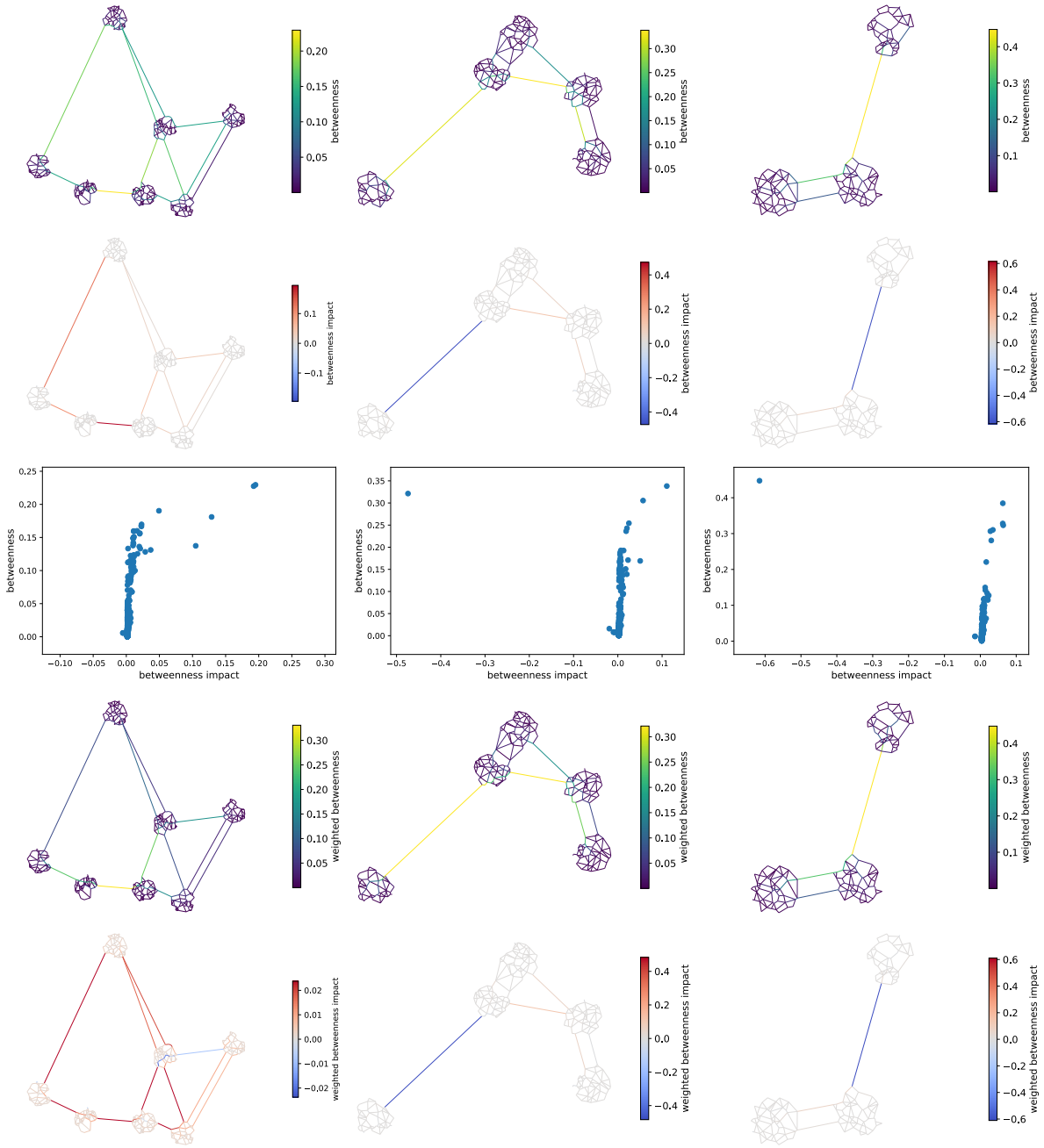


Fig. 6. Effect of changing the number of village centers on centrality measures. Different random street networks are created using the random network model described in section 3.2. Villages with a different number of village centers are created, effectively changing the size of the region under consideration. The number of junction nodes per village and the sprawling radius r are held constant. Plotted are the edge betweenness centrality, see Eq. (3), and the impact measure for road blockage, see Eq. (4). For both measures, the weighted and the unweighted version are displayed. Additionally, scatter plots are shown setting the edge betweenness centrality and the impact measure into relation. There is no proportionality between edge betweenness centrality and impact measure. The number of roads with capability to isolate complete villages when blocked is reduced as the number of village centers is increased.

To quantify this, the unweighted edge centrality is plotted against the unweighted impact measure for each road in a scatter plot, see figure Fig. 6. Notably, there is no proportionality between betweenness centrality and impact measure. While only roads with a large betweenness centrality seem to be capable of yielding large impact effects, the vice versa is not true. Many streets with large betweenness centrality scores have only a marginal effect on the average betweenness centrality when blocked. Comparing with the plotted networks, we identify that roads with high centrality but little blockage impact are located inside villages. Indeed, inside villages, frequent bypass possibilities exist that only marginally increase the length of the shortest path.

Summarising, we find that there are small effects of changing the number of village centers considered in the network. Larger regions tend to better connect villages between each other, such the number of roads that isolate complete villages when blocked reduces as the number of village centers is increased. Also changing the size of the region considered has qualitative effects on the centrality of roads both for roads connecting villages and central paths inside villages.

We refrain from concluding on the quality of the random network model at this point and instead focus next on the real world rural street networks. Only after comparing the results of both types of networks, we will discuss the advantages and shortcomings of the random network model.

6 Real world rural street networks

6.1 Descriptive analysis of the real world network

In this section, we apply the same measure established in section 5 on real world rural street networks and compare the results with the random network model. In section 3.1 we described how the real world rural street networks are extracted from maps, in section 4 we defined the quantities we use in this section to analyse the networks. We analyse three regions in different parts of Germany differing in region and village size. Figure Fig. 1 shows satellite images of the regions under consideration. The results are presented in three separate figures, where Fig. 7 displays results for the region around Epfenbach in Baden-Wuerrtemberg, Fig. 8 displays results for the region around Huelsen in Lower Saxony, and Fig. 9 displays results for the region around Rosswein in Saxony. As the results are to a large extend similar, the regions aren't discussed separately in this report. Differences are mentioned explicitly.

In the figures Fig. 7,8,9 the betweenness centrality for the edges and the nodes are displayed for both the weighted and the unweighted versions of the betweenness centrality. These two concepts of centrality give information about how central roads or junction nodes are, respectively. Though these two variants of the betweenness centrality are conceptional different, the results are closely related, as central edges start and end at central nodes. Like for the random network model, there are differences in whether the weighted or the unweighted version is used. Especially the region around Epfenbach displayed in figure Fig. 7 illustrates this difference. For the unweighted betweenness centrality, the street passing Reichartshausen[2] in the north has a high centrality, indicating that many shortest paths avoid crossing the village with its many junctions. In contrast, for the weighted version, which takes street length into account, shortest paths crossing through the village are determined, as distance-wise bypassing the village is a detour. This example indicates that depending on the weight chosen different roads render important and frequently used. This demonstrates how slight adjustments in the transportation model used are leading to vastly different instructions on how street networks should be optimised for transportation.

As for the random network model, roads connecting villages tend to have higher centrality scores than streets inside villages. Importantly, *central paths* through villages render exceptions to this. Most prominently, this is observable in the region around Epfenbach in Baden-Wuerttemberg and for the village Huelsen in Lower Saxony, see figures Fig. 7,8. In contrast, this isn't observable for the region between Nossen and Rosswein in Saxony, where several paths connect the two villages.

Checking the distance measure for the different regions, both the euclidean distance distribution and the weighted street distance distribution show peaked distributions, similar to the random network model. For all three regions, a big peak close to zero is identifiable corresponding to the denser street network inside the villages. However, the peaks in the distribution are spread out and information about the number of villages considered in the respective region cannot be obtained from the distributions. Summarising, the street distributions reflect the two intrinsic spatial scales of rural street networks both in the random network model and for real world rural street networks.

Focusing on the impact of road blockage on the average betweenness centrality, we find differences between the random network model and the real world rural street networks. Checking the abundance of roads that decrease

[2]Reichartshausen is the big village in the east of the region in Baden-Wuerttemberg.

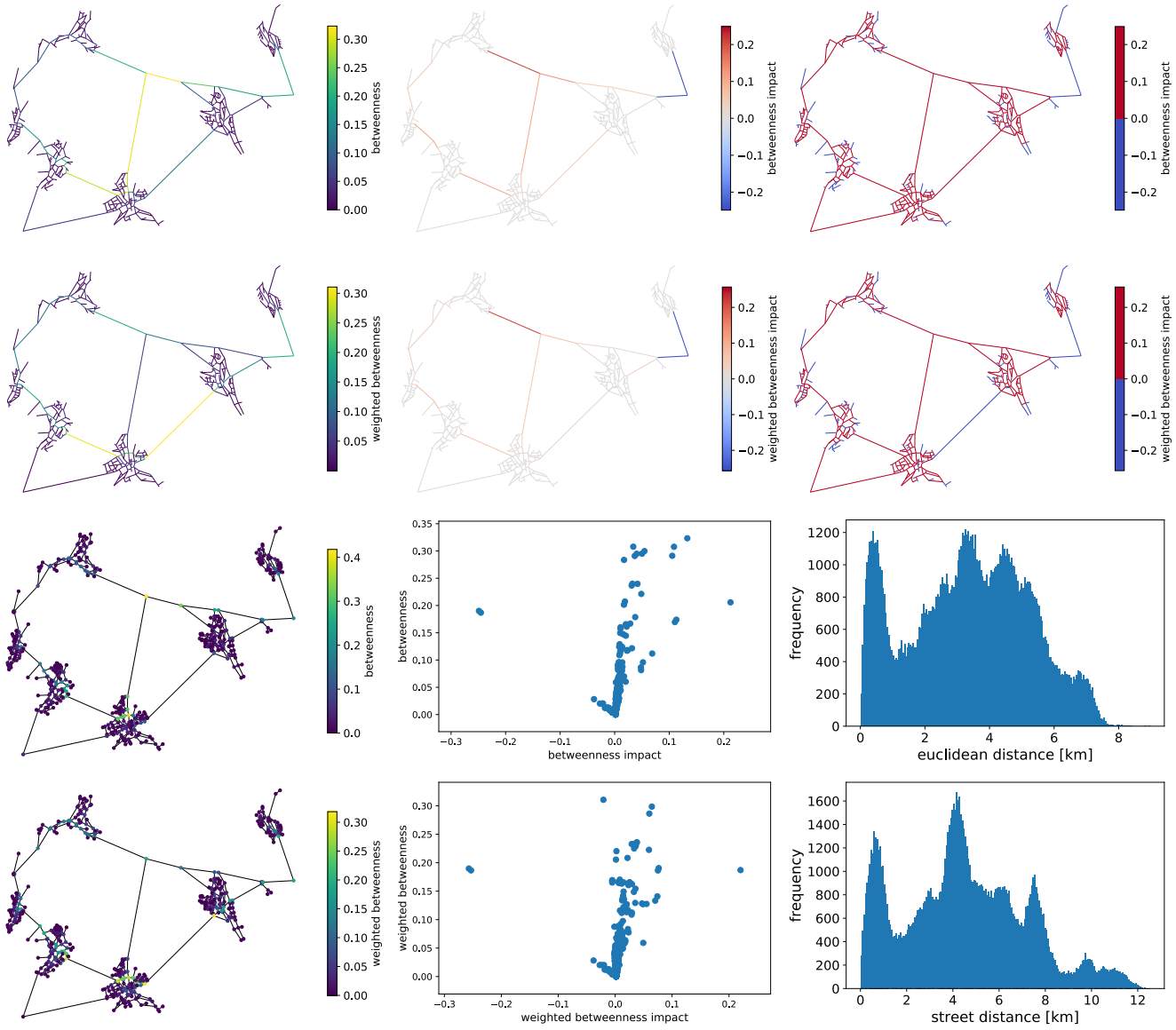


Fig. 7. Analysis of the rural street network located in Baden-Wuerttemberg. The measures described in section 4 were employed to analyse the rural street network between the region around the village Epfenbach in Baden-Wuerttemberg. Epfenbach is the big village in the south of the region. A annotated satellite image of the region can be found in figure Fig. 1. The networks were obtained following the instructions of section 3.1. Compare also with the analogous figures Fig. 8,9. In this region, several clearly separated villages are shown. The streets connecting the villages are more central than streets inside villages. Depending on the choice of weighted or unweighted edge betweenness centrality, streets connecting villages render more or less central.

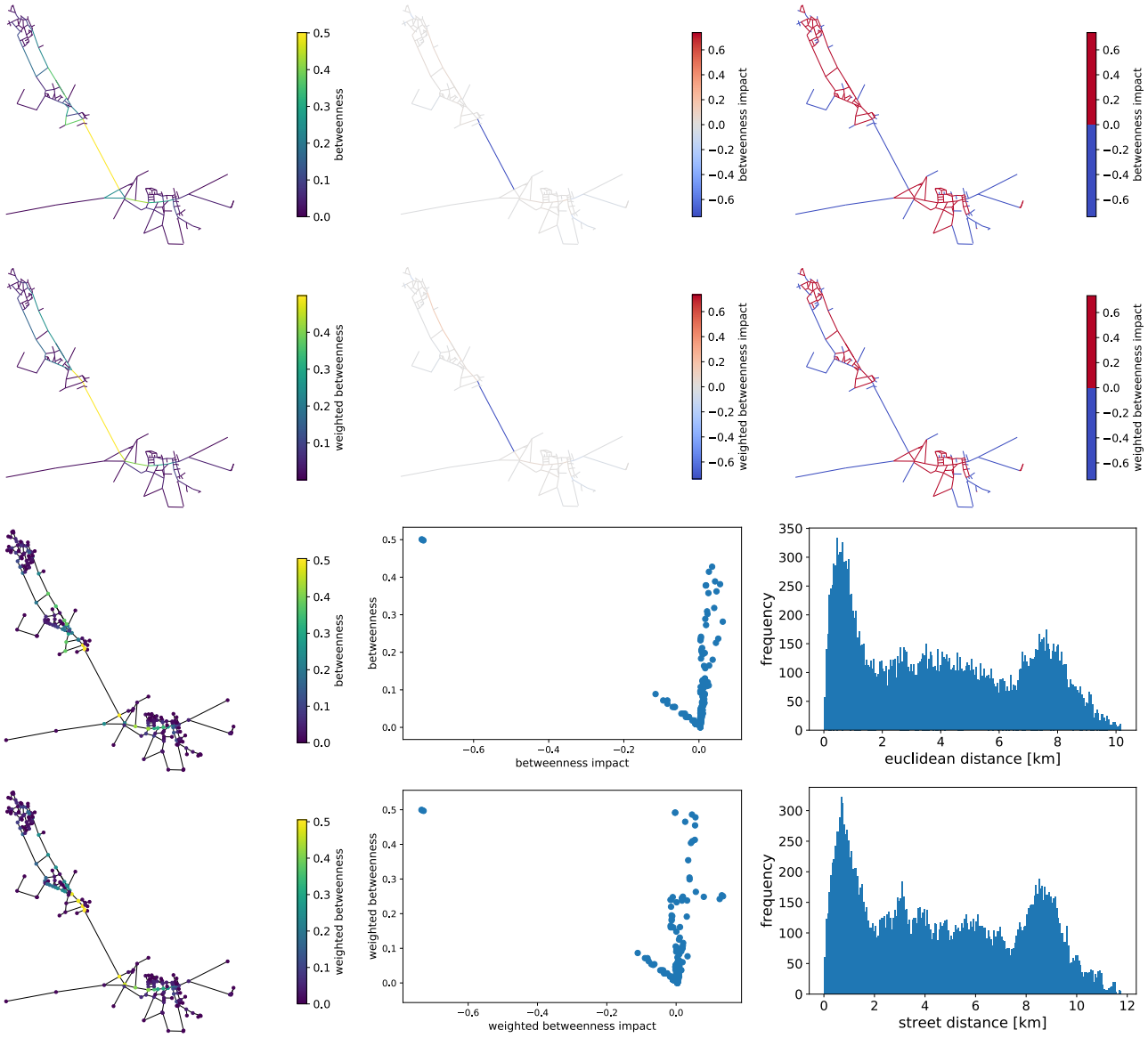


Fig. 8. Analysis of the rural street network located in Lower Saxony. The measures described in section 4 were employed to analyse the rural street network between the villages Rethem and Huelsen in Lower Saxony. A annotated satellite image of the region is shown in figure Fig. 1. The networks were obtained following the instructions of section 3.1. Compare also with the analogous figures Fig. 7,9. In this region, only two villages are connected two each other. The street connecting both villages is the most central street in the network and has the highest impact on the average betweenness centrality when blocked.

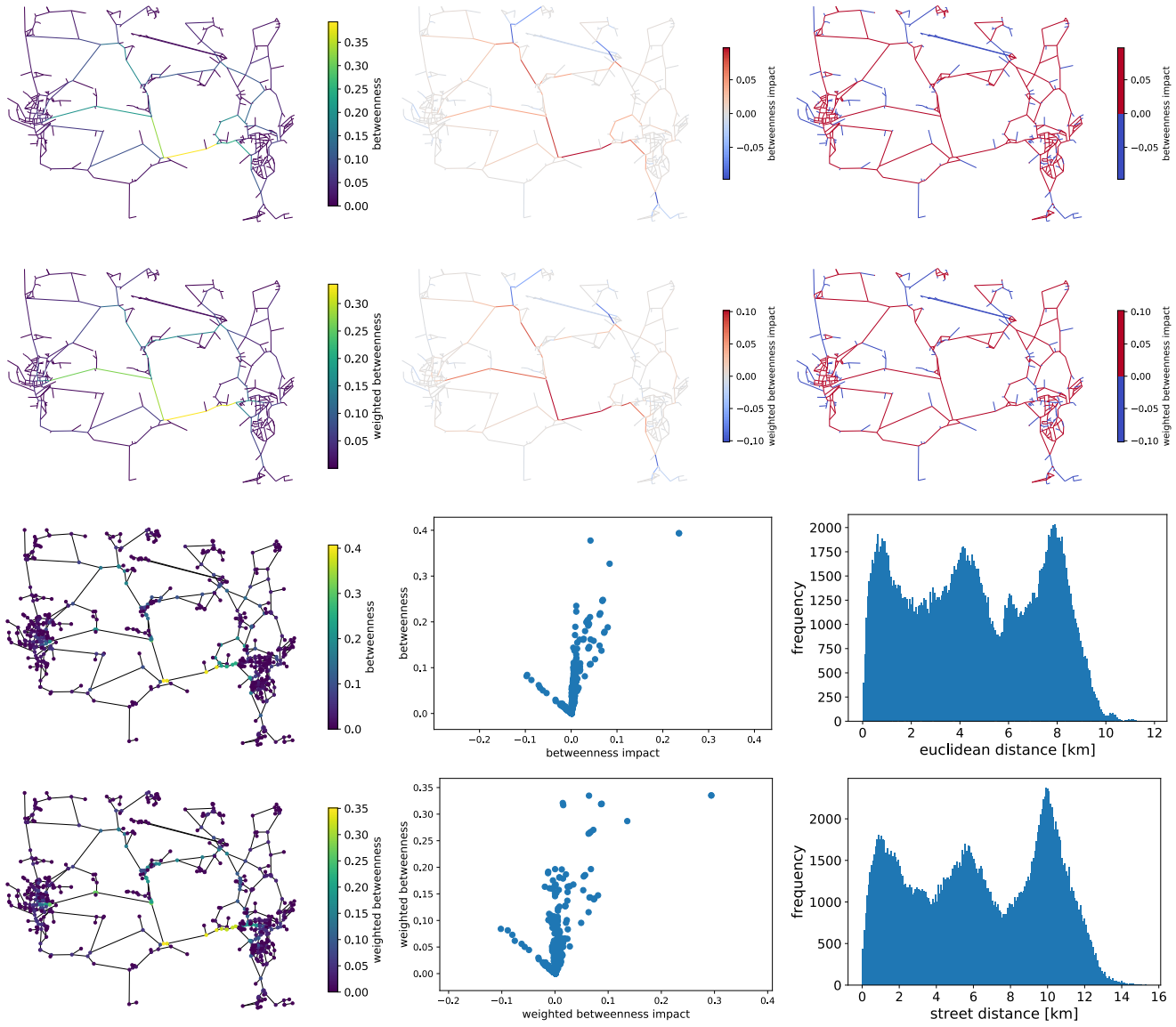


Fig. 9. Analysis of the rural street network located in Saxony. The measures described in section 4 were employed to analyse the rural street network between the villages Rosswein and Nossen in Saxony. A annotated satellite image of the region is displayed in figure Fig. 1. The networks were obtained following the instructions of section 3.1. Compare also with the analogous figures Fig. 7,8. In contrast to the other regions, there are multiple streets connecting the two villages, such that a single road blockage cannot isolate the villages from each other.

the average betweenness centrality when blocked, we find that for both the weighted and the unweighted version of the impact measure these roads are frequent for all three regions. Checking the position of these nodes using a binary colour-scale, we find these roads are predominantly dead ends, which were hardly present in the random network model. Besides this major difference, there are a number of similarities between the random network model and the real world rural street networks. Like for the random network model, blocking roads that cut off villages from the rest of the network decrease the average betweenness centrality strongly. This can be seen for the region in Lower Saxony as the villages Rethem and Huelsen are only connected by one road, or in the region in Baden-Wuerttemberg, where the village Michelsbach[3] is only connected via on street with the other villages. Also, as for the random network model, blocking roads connecting villages leads to a strong increase in the average betweenness centrality if this blockage doesn't isolate villages.

Notably, focusing on the street connecting the village Epfenbach[4] and Reichartshausen and considering the weighted version of the edge centrality and impact measure, we find a different behaviour. While this road has a high centrality connecting to villages and no close parallel road is displayed, blocking this road strongly reduces the average betweenness centrality. This isn't the case for the unweighted version of the impact measure. In contrast, the previously observed theme, where for two parallel roads blocking one of the roads increases and for the other road decreases the average betweenness centrality, can be also observed for the region in Lower Saxony in the village Rethem (the more northern city in the region considered).

Linking betweenness centrality and impact measure in scatter plots, we find no proportionality for roads increasing the average betweenness centrality with edge centrality, similar to the random network model. The scatter plot reflects the difference whether streets are connecting different village centers, or whether the streets are located inside villages, as already observed for the random network model. In contrast to this, there seems to be a proportionality for the roads decreasing the average betweenness centrality when blocked. While this proportionality appears sharp when considering the unweighted versions, few roads with high betweenness centrality but small negative impact dilute this result for the weighted version.

6.2 Evaluation of the random network model

Having contrasted the random network model with the real world rural street networks, we evaluate the advantages and the limitations of the random network model. The random network model is capable of inferring the two intrinsic spatial scales of rural street networks reflecting dense streets inside villages connected by few long roads. The distance distributions are highly comparable. The random network model allows for easily changing the village size and its effect on the distance distribution, which cannot be explored systematically in real world networks. Yet, the information of the distance distribution is itself limited.

Moreover, the importance of streets connecting villages is predicted by the random network model and verified by real world rural street networks. Also, the impact measure shows qualitatively a good agreement between the real world rural street networks and the random network model. Yet, it should be noted, that the random network model is ineffectively creating dead-ends, which are frequent in real world villages. This connects to the major drawback of the random network model, which is its inability of creating realistic street networks inside villages. Hence, the analysis of roads inside villages should be taken with caution when using the random network model.

6.3 Speculation about different categories of roads regarding their impact measure

Regarding the impact measure introduced in equation Eq. (4), we have identified two categories of roads: Roads that increase the average betweenness centrality when blocked and roads that decrease the average betweenness centrality when blocked. We found that some roads change the category depending on whether the streets are weighted with their euclidean length or not. For the unweighted impact measure, the categories can easily be connected to structural characteristics of roads in the network. Only if roads disconnect junctions from the remaining network and by this create an unconnected graph, the average betweenness centrality is reduced. This is directly reflected in Eq. (4) where the definition $0/0 = 0$ is used. This definition implies that the number of shortest paths is reduced, which reduces also the on average betweenness centrality. Vice versa, if alternative bypasses are available for the betweenness centrality this increases the number of edges that are included in the shortest paths, which increases the average betweenness centrality.

Things are more difficult when interpreting the weighted version of the impact measure. While cutting edges that disconnect the graph decrease the average betweenness centrality, there are also other edges that decrease the

[3]The village in the northeast of the region.

[4]The large village in the south of the region.

average betweenness centrality that doesn't fulfill this structural condition. Studying the random network model and the real rural street networks, we observed that this predominantly happens if two paths with high betweenness centrality run parallel to each other. However, we also found counterexamples for parallel pathways. Here, it should be noted that the average betweenness centrality doesn't hold the full information about the distribution of differential changes resulting from blocking a single road. To disentangle the impact of these edges, it would advisable to check the differential change mapped to the network when these roads are blocked.

7 Relevance for other transportation network, discussion of the methods employed, and outlook

In this report, we analyzed rural street networks which are specific types of transportation networks. Characteristic for rural street networks is a spatial inhomogeneity reflected in two intrinsic spatial scales of these networks, reflecting dense street networks in villages connected by few large roads. While these two spatial scales can be depicted from distance measures, centrality measures give information about the importance of edges. Another measure to estimate the importance of roads is mapping the effect of these roads when blocked. For both measures, roads that connect villages render most important in both a random network model and real world rural street networks.

The impact measure revealed two categories of roads. Roads that increase the average betweenness centrality when blocked and roads that decrease the average betweenness centrality when blocked. While these two categories could roughly be associated with network structures when no edge weights are considered, no structural conditions could be established if the euclidean length of each road was set as edge weight. Here, mapping the differential change on the network instead of just considering the change in the average could prove a fruitful way to deepen the understanding of transportation networks.

A major drawback of the random network model is its *ad hoc* creation, which is in contrast to most transportation networks that grow and optimise over time [10; 1; 5; 6]. Nevertheless, the random network model effectively captures key statistics of real world rural street networks. It could be insightful to contrast this random network with a random network model that is grown iteratively to further shed light on the general building principles of transportation networks.

Importantly, the analysis in this report completely neglected the width and hierarchy of roads. Highways and dirt roads were treated equally, which is unrealistic. Distinguishing the width and capacity of roads could have a strong impact on the results presented in this report. However, introducing a hierarchy for streets inside villages could be a major obstacle for the random network model. This might make refinements of the random network model necessary. We considered the rural street networks as undirected graphs, yet interpreting rural street networks as directed graphs instead is a plausible improvement, given the fact that one-way roads exist and that cars parked at the side of the road might hinder traffic dominantly in one direction.

While rural networks are special in respect of having dense centers connected by few edges, this theme is not uncommon for transportation networks. Here one could think of the vasculature of animals, where dense capillary networks inside organs are connected by few big veins and arteries. It might be interesting to investigate how the insight generated in this report translates to these types of directed flow networks.

In this report, we investigated the impact of road blockage. This concept is not only important to various types of human-made good transportation networks but is also of high importance for biological transportation networks. A prominent example here would be the blockage of arterioles during strokes. Even more suited is the analysis of leaf venation patterns with the impact measure defined in Eq. (4). A prominent publication in the field of biological fluid flow networks linked the resilience to rodents eating holes in leaves with the loopy network structure in leaves [5]. While to the author's knowledge street networks and leaf venation patterns have been compared before [1], no connection has been made between blocking roads and the resilience ability in biological fluid flow networks. The measure defined in Eq. (4) together with an understanding under which structural conditions road blockage increases or decreased the average betweenness centrality could help to improve the mechanisms underlying the optimization processes of biological transportation networks.

Noting general themes in human-made and biological transportation networks, it might be fruitful to discuss and apply concepts developed either for human-made or biological transportation networks to the other type of transportation network. This could generate a deeper understanding of how network morphology sets physical limits on transport and might lead to new and unconditional optimisation approaches in human-made transportation networks.

References

[1] BARTHELEMY, M. From paths to blocks: New measures for street patterns. *Environment and Planning B: Urban Analytics and City Science 44*, 2 (Mar. 2017), 256–271. Publisher: SAGE Publications Ltd STM.

[2] BRANDES, U. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology 25*, 2 (June 2001), 163–177. Publisher: Routledge _eprint: https://doi.org/10.1080/0022250X.2001.9990249.

[3] CORSON, F. Fluctuations and Redundancy in Optimal Transport Networks. *Phys. Rev. Lett. 104*, 4 (Jan. 2010), 048703. Publisher: American Physical Society.

[4] KARSCHAU, J., SCHOLICH, A., WISE, J., MORALES-NAVARRETE, H., KALAIDZIDIS, Y., ZERIAL, M., AND FRIEDRICH, B. M. Resilience of three-dimensional sinusoidal networks in liver tissue. *PLOS Computational Biology 16*, 6 (June 2020), e1007965. Publisher: Public Library of Science.

[5] KATIFORI, E., SZÖLLŐSI, G. J., AND MAGNASCO, M. O. Damage and Fluctuations Induce Loops in Optimal Transport Networks. *Phys. Rev. Lett. 104*, 4 (Jan. 2010), 048704. Publisher: American Physical Society.

[6] MEIGEL, F. J., AND ALIM, K. Flow rate of transport network controls uniform metabolite supply to tissue. *Journal of The Royal Society Interface 15*, 142 (May 2018), 20180075. Publisher: Royal Society.

[7] MEIGEL, F. J., CHA, P., BRENNER, M. P., AND ALIM, K. Robust Increase in Supply by Vessel Dilation in Globally Coupled Microvasculature. *Phys. Rev. Lett. 123*, 22 (Nov. 2019), 228103. Publisher: American Physical Society.

[8] NEWMAN, M. E. J. *Networks*, second edition ed. Oxford University Press, Oxford, 2018.

[9] SINSKE, S. A., AND ZIETSMAN, H. L. Sewer-system analysis with the aid of a geographical information system. *Water SA 28*, 3 (2002), 243–248. Number: 3.

[10] STRANO, E., NICOSIA, V., LATORA, V., PORTA, S., AND BARTHÉLEMY, M. Elementary processes governing the evolution of road networks. *Sci Rep 2*, 1 (Mar. 2012), 296.

RandomNetworkModel

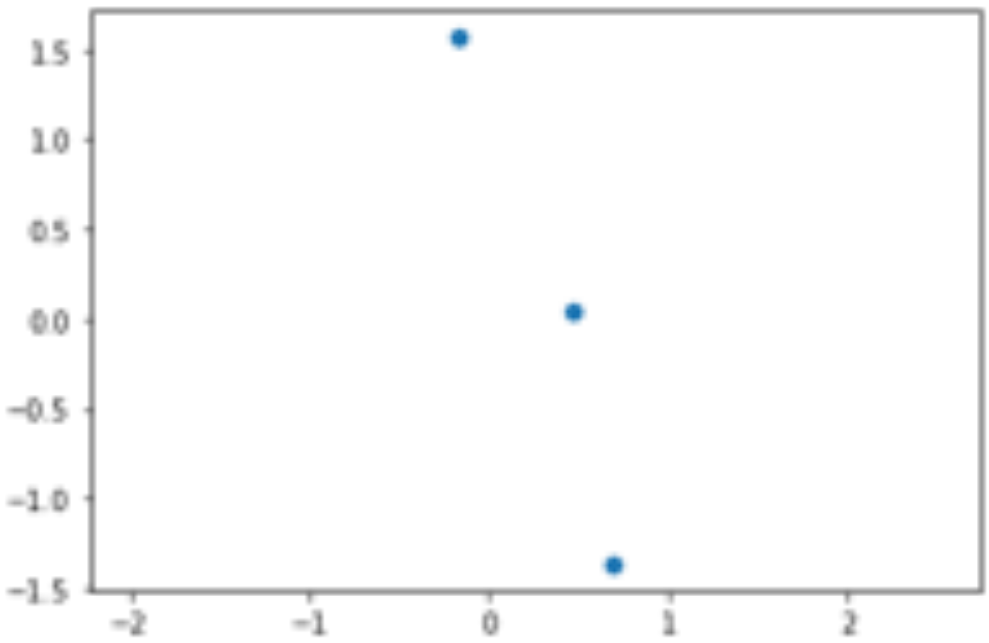
August 14, 2021

[1]:
```
#Load Libraries

import pickle as pkl
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import csv
import collections
from scipy.spatial import Delaunay
```

[2]:
```
#Generate village centers
def genRandCirc(n,radius):
    sr=np.random.uniform(0,1,size=n)
    r=np.sqrt(sr*radius)
    theta=np.random.uniform(0,1,size=n)*2*np.pi
    y=r*np.cos(theta)
    x=r*np.sin(theta)
    return [x,y]
```

[3]:
```
#Generate junction nodes
def genRandRural(n,radius,listx,listy):
    numcenter=len(listx)
    xveclist=[]
    yveclist=[]
    for counter in range(0,numcenter):
        [xlistr,ylistr]=genRandCirc(n,radius)
        xveclist.append(listx[counter]+xlistr)
        yveclist.append(listy[counter]+ylistr)
    xveclista=np.asarray(xveclist)
    yveclista=np.asarray(yveclist)
    return [xveclista.flatten(),yveclista.flatten()]
```

[7]:
```
#generate village centers
numvilagecenters=3
[xvec,yvec]=genRandCirc(numvilagecenters,numvilagecenters)
plt.scatter(xvec,yvec)
plt.axis('equal')
plt.show()
```



[8]:
```
numCC=numvilagecenters
#create rural spots
ruralspots=20
ruralradius=0.1

#generate random rural centers
[xvec2,yvec2]=genRandRural(ruralspots,ruralradius,xvec,yvec)

#plot and check
plt.scatter(xvec2,yvec2)
plt.axis('equal')
plt.show()

#make Delaunay triangulation
posvec=np.asarray([xvec2,yvec2])
posvect=posvec.transpose()
vertices=Delaunay(posvect)


#create edge list
edges=set()
for n in range(vertices.nsimplex):
    edge=sorted([vertices.vertices[n,0],vertices.vertices[n,1]])
    edges.add((edge[0],edge[1]))
    edge=sorted([vertices.vertices[n,0],vertices.vertices[n,2]])
```

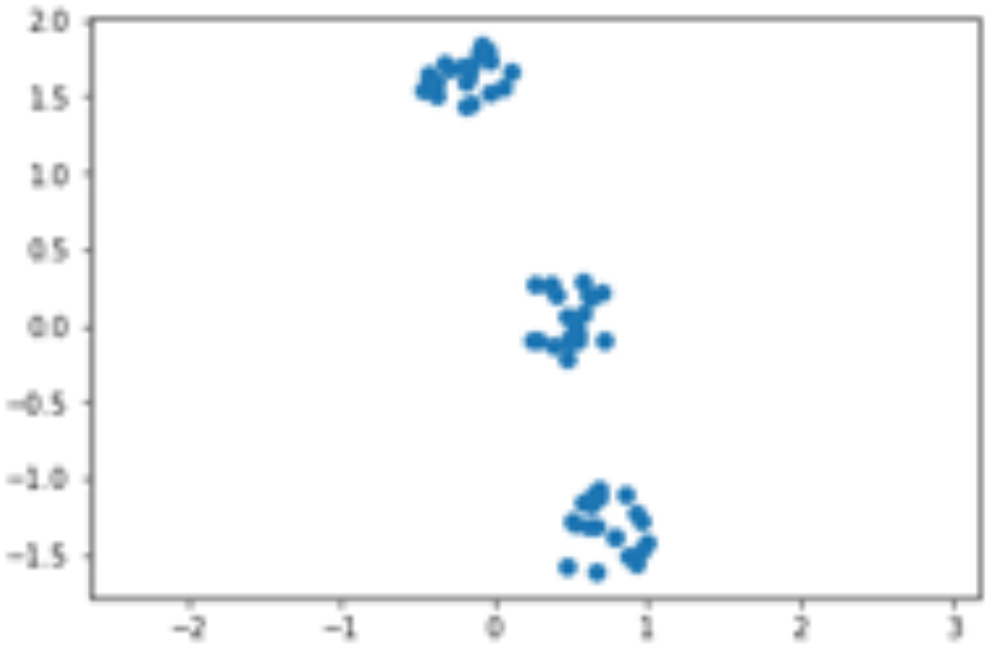
```
    edges.add((edge[0],edge[1]))
    edge=sorted([vertices.vertices[n,1],vertices.vertices[n,2]])
    edges.add((edge[0],edge[1]))

#make a graph
Grural = nx.Graph()
Grural.add_nodes_from(range(len(posvect)))
Grural.add_edges_from(edges)
for counter in range(len(posvect)):
    Grural.nodes[counter]['pos']=[posvect[counter,0],posvect[counter,1]]



#create Gabriel graph
listofnextneighbours=[]
startpoint=0

A = nx.adjacency_matrix(Grural)  #adjacency matrix
Ac=A.toarray()
Anew=np.zeros((len(posvect),len(posvect)))
A2=A*A
A2c=A2.toarray()
indices=np.where(Ac[startpoint,:]>0)


#delete edges for Gabriel graph
for counter in range(len(posvect)):
    indicesy=np.where(Ac[counter,:]>0)
    indexlisty=indicesy[0].tolist()
    indexx=counter
    for countery in range(len(indexlisty)):
        indexy=indexlisty[countery]
        if (indexx!=indexy):
            indicesx=np.where(Ac[indexy,:]>0)
            indexlistx=indicesx[0].tolist()
            fullindexlist=indexlisty+indexlistx
            #print(fullindexlist)
            midpointx=(posvect[indexx,0]+posvect[indexy,0])/2
            midpointy=(posvect[indexx,1]+posvect[indexy,1])/2
            distance=np.
 ↪sqrt((posvect[indexx,0]-posvect[indexy,0])**2+(posvect[indexx,1]-posvect[indexy,1])**2)/
 ↪2
            edgeremains=True
            for countern in range(len(fullindexlist)):
                indexn=fullindexlist[countern]
                if (indexn!=indexx) and (indexn!=indexy):
```

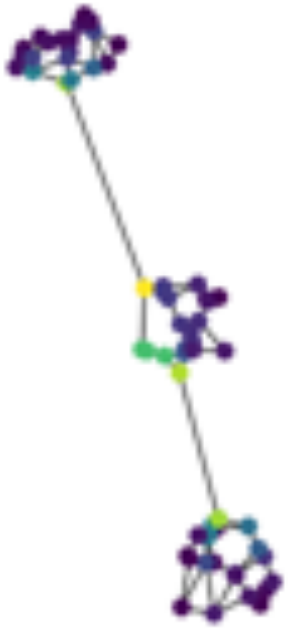
```
                    newdistance=np.
 ↪sqrt((midpointx-posvect[indexn,0])**2+(midpointy-posvect[indexn,1])**2)
                    #print([indexx,indexy,indexn,distance,newdistance])
                    if newdistance<distance:
                        edgeremains=False
            if edgeremains:
                Anew[indexx,indexy]=1

#create Gabriel graph in networkX
rows, cols = np.where(Anew == 1)
edges2 = zip(rows.tolist(), cols.tolist())

Grural2 = nx.Graph()
Grural2.add_nodes_from(range(len(posvect)))
Grural2.add_edges_from(edges2)
for counter in range(len(posvect)):
    Grural2.nodes[counter]['pos']=[posvect[counter,0],posvect[counter,1]]


#draw Gabriel graph and check
nx.draw(Grural2, nx.get_node_attributes(Grural2, 'pos'), with_labels=False,␣
 ↪node_size=0)
plt.axis('equal')
filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_GabGraph.
 ↪pdf'
#plt.tight_layout()
plt.savefig(filename)
plt.show()

#create copy of Gabriel graph for edge betweenness centrality
edges3=nx.edge_betweenness_centrality(Grural2)
nodes3=nx.betweenness_centrality(Grural2)

#print(edges3)

Grural3 = nx.Graph()
Grural3.add_nodes_from(range(len(posvect)))
Grural3.add_edges_from(edges3)
for counter in range(len(posvect)):
    Grural3.nodes[counter]['pos']=[posvect[counter,0],posvect[counter,1]]

nx.set_edge_attributes(Grural3, edges3, "betweenness")
nx.set_node_attributes(Grural3, nodes3, "betweenness")


#create and plot graph with edge color
```

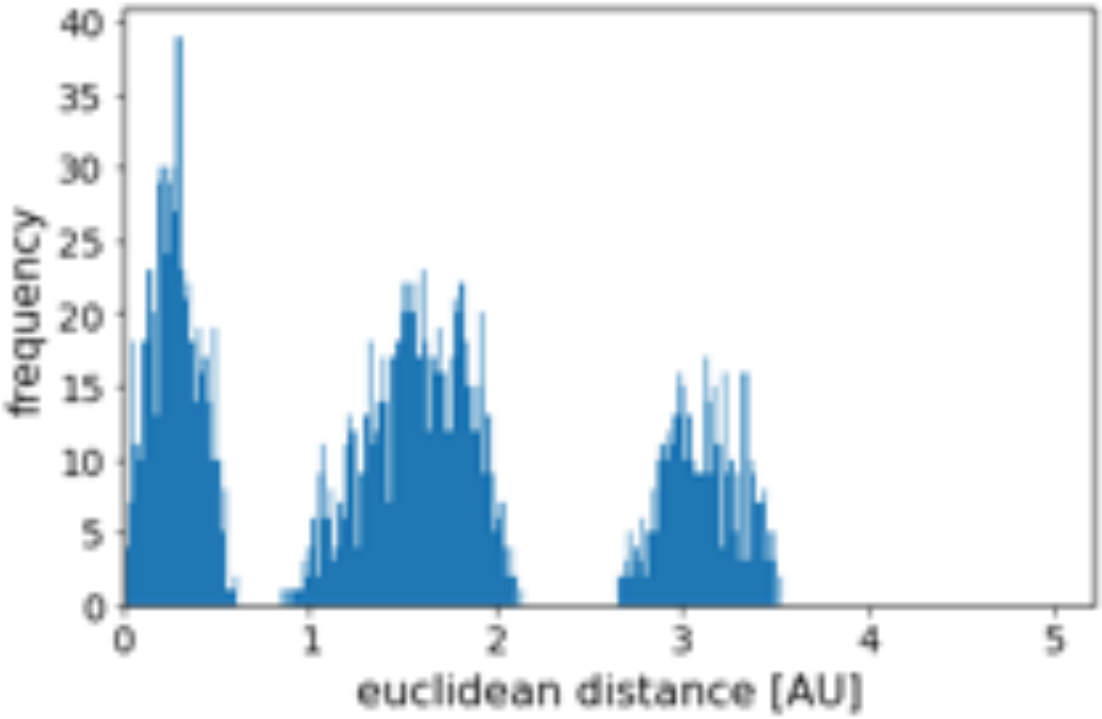
```
edges,weights = zip(*nx.get_edge_attributes(Grural3,'betweenness').items())
nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges, edge_color=weights)
plt.axis('equal')
filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_GraphEdgeBetween.
 ↪pdf'
#plt.tight_layout()
plt.savefig(filename)
plt.show()

#create and plot graph with node color
nodes,weights = zip(*nx.get_node_attributes(Grural3,'betweenness').items())
nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
 ↪node_size=40,nodelist=nodes, node_color=weights)
plt.axis('equal')
filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_GraphNodeBetween.
 ↪pdf'
#plt.tight_layout()
plt.savefig(filename)
plt.show()

#Generate euclidean distance between junctions
totaldistanceED=[]
for u in range(len(posvect)):
    for v in range(len(posvect)):
        if u<v:
            newdistance=np.
 ↪sqrt((posvect[u,0]-posvect[v,0])**2+(posvect[u,1]-posvect[v,1])**2)
            totaldistanceED.append(newdistance)

#plot histogram
plt.hist(totaldistanceED,200)
plt.ylabel('frequency',fontsize=16)
plt.xlabel('euclidean distance [AU]',fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlim(0,5.2)
filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_DistanceEuc.
 ↪pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()
```

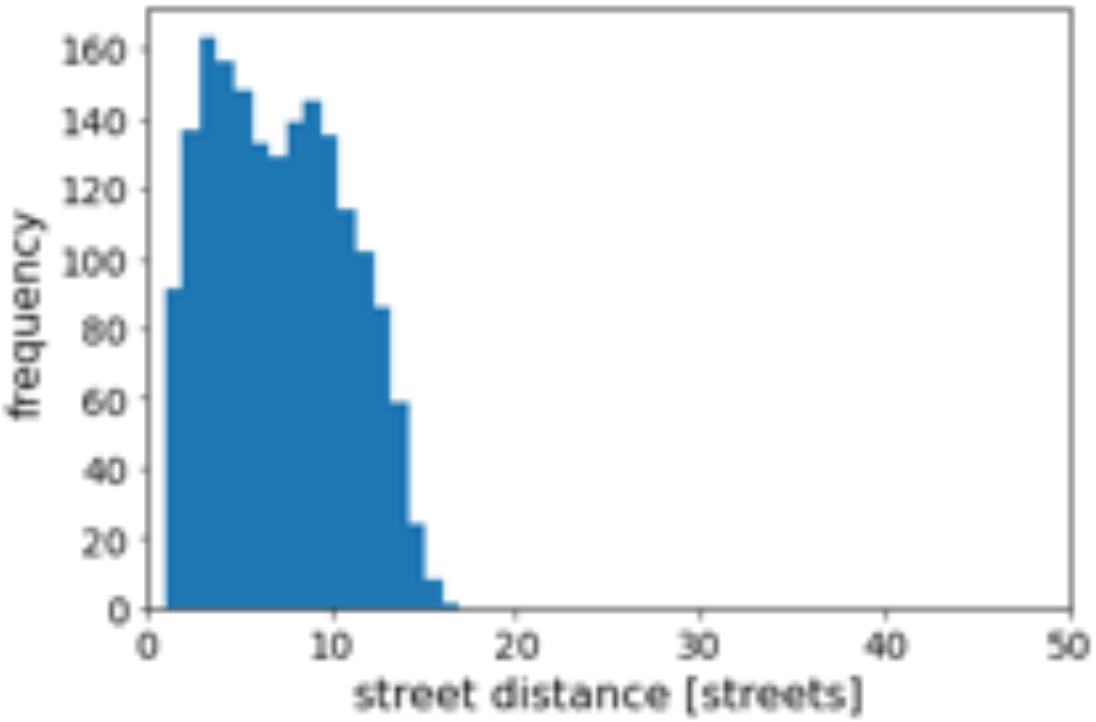
```
#Generate edge distance between junctions
totaldistanceEDD=[]
lendistance=dict(nx.all_pairs_shortest_path_length(Grural3))
for u in range(len(posvect)):
    for v in range(len(posvect)):
        if u<v:
            newdistance=lendistance[u][v]
            totaldistanceEDD.append(newdistance)

#plot histogram
plt.hist(totaldistanceEDD,np.max(totaldistanceEDD))
plt.ylabel('frequency',fontsize=16)
plt.xlabel('street distance [streets]',fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlim(0,50)
filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_DistanceStreets.
 ↪pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()


#Generate weigthed edge distance between junctions
collectalllengths=[]
for u,v in Grural3.edges(data=False):
    Grural3.edges[u,v]['length'] = np.
 ↪sqrt((posvect[u,0]-posvect[v,0])**2+(posvect[u,1]-posvect[v,1])**2)
    collectalllengths.append(np.
 ↪sqrt((posvect[u,0]-posvect[v,0])**2+(posvect[u,1]-posvect[v,1])**2))


totaldistanceWED=[]
lendistance=dict(nx.all_pairs_dijkstra_path_length(Grural3, cutoff=None,␣
 ↪weight='length'))
for u in range(len(posvect)):
    for v in range(len(posvect)):
        if u<v:
            newdistance=lendistance[u][v]
            totaldistanceWED.append(newdistance)


#plot histogram
plt.hist(totaldistanceWED,200)
plt.ylabel('frequency',fontsize=16)
plt.xlabel('street distance [AU]',fontsize=16)
```

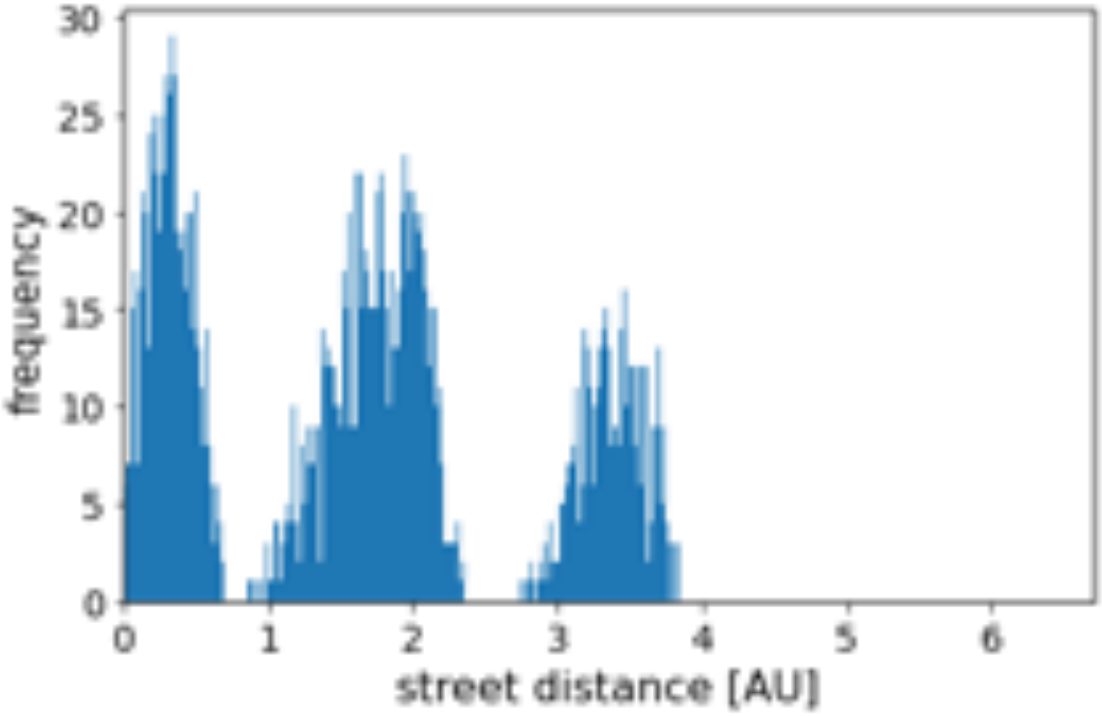
```
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlim(0,6.7)
filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_DistanceStreetsWeight.
 ↪pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()

plt.hist(collectalllengths,60)
plt.ylabel('frequency',fontsize=16)
plt.xlabel('length [AU]',fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
plt.xlim(0,3)
filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_LengthStreets.
 ↪pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()
```

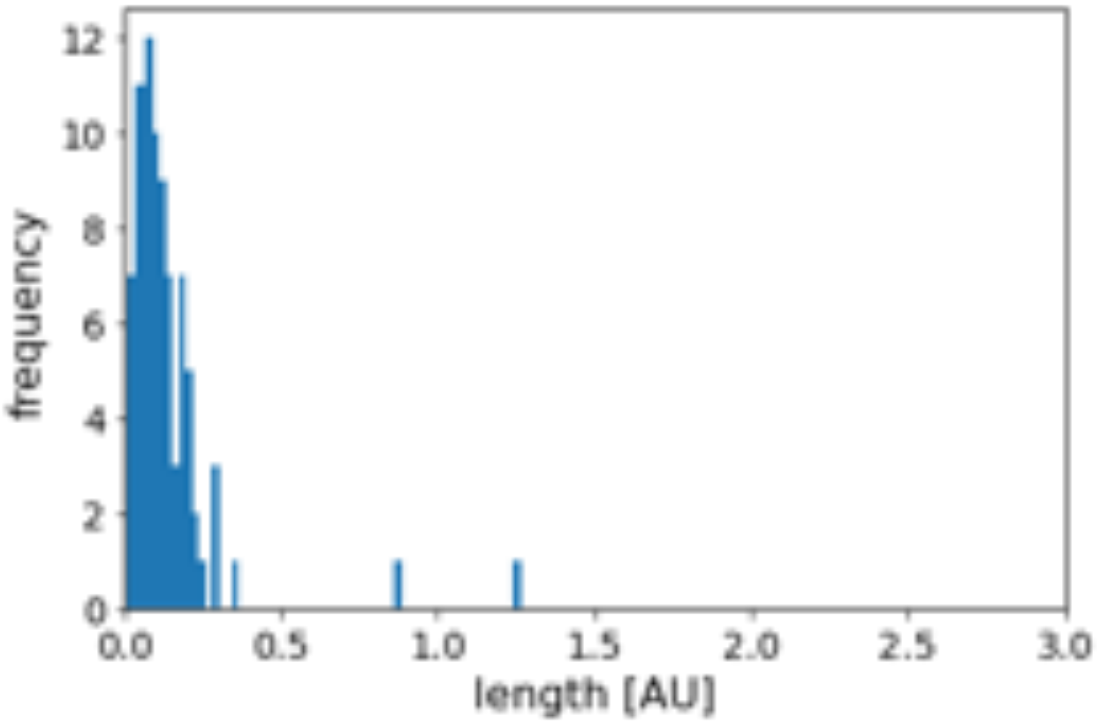





frequency
euclidean distance [AU]
0
5
10
15
20
25
30
35
40
0
1
2
3
4
5


frequency
0
20
40
60
80
100
120
140
160
0
10
20
30
40
50
street distance [streets]


frequency
0
5
10
15
20
25
30
0
1
2
3
4
5
6
street distance [AU]

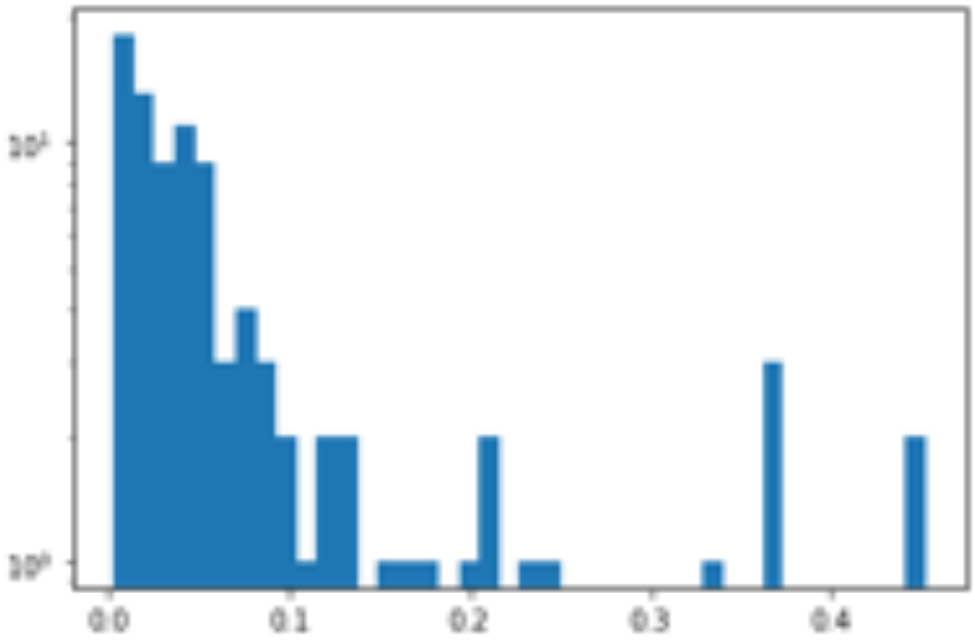


[12]:
```
#compute the weighted betweenness centrality
edges4=nx.edge_betweenness_centrality(Grural3,weight='length')
nodes4=nx.betweenness_centrality(Grural3,weight='length')
nx.set_edge_attributes(Grural3, edges4, "betweenness2")
nx.set_node_attributes(Grural3, nodes4, "betweenness2")
#print(Grural3.edges[0,7]['length'])
#print(Grural3.edges(data=True))


#compute average betweenness centrality
averageBetweeness=[]
for u,v in Grural3.edges(data=False):
    averageBetweeness.append(Grural3.edges[u,v]['betweenness'])
meanaverage=np.mean(averageBetweeness)
print(meanaverage)

averageBetweeness2=[]
for u,v in Grural3.edges(data=False):
    averageBetweeness2.append(Grural3.edges[u,v]['betweenness2'])
meanaverage2=np.mean(averageBetweeness2)
print(meanaverage2)

#compute impact (unweighted)
```

```
GruralDel = nx.Graph()
GruralDel.add_nodes_from(range(len(posvect)))
GruralDel.add_edges_from(Grural3.edges(data=True))
for counter in range(len(posvect)):
    GruralDel.nodes[counter]['pos']=[posvect[counter,0],posvect[counter,1]]
counter=0
for u,v in Grural3.edges(data=False):
    counter=counter+1
    #print(counter/len(averageBetweeness))
    savelen=Grural3.edges[u,v]['length']
    GruralDel.remove_edge(u, v)
    edgesDel=nx.edge_betweenness_centrality(GruralDel)
    nx.set_edge_attributes(GruralDel, edgesDel, "betweennessDEL")
    averageBetweenessDel=[]
    for x,y in GruralDel.edges(data=False):
        averageBetweenessDel.append(GruralDel.edges[x,y]['betweennessDEL'])

    meanaverageDel=np.mean(averageBetweenessDel)
    #print([u,v,(meanaverageDel-meanaverage)/meanaverage])
    Grural3.edges[u,v]['del'] = ((meanaverageDel-meanaverage)/meanaverage)
    GruralDel.add_edge(u, v,length=savelen)

#compute impact (unweighted)
GruralDel = nx.Graph()
GruralDel.add_nodes_from(range(len(posvect)))
GruralDel.add_edges_from(Grural3.edges(data=True))
for counter in range(len(posvect)):
    GruralDel.nodes[counter]['pos']=[posvect[counter,0],posvect[counter,1]]

counter=0
for u,v in Grural3.edges(data=False):
    counter=counter+1
    #print(counter/len(averageBetweeness))
    savelen=Grural3.edges[u,v]['length']
    GruralDel.remove_edge(u, v)
    edgesDel=nx.edge_betweenness_centrality(GruralDel,weight='length')
    nx.set_edge_attributes(GruralDel, edgesDel, "betweennessDEL")
    averageBetweenessDel=[]
    for x,y in GruralDel.edges(data=False):
        averageBetweenessDel.append(GruralDel.edges[x,y]['betweennessDEL'])

    meanaverageDel=np.mean(averageBetweenessDel)
    #print([u,v,(meanaverageDel-meanaverage)/meanaverage])
    Grural3.edges[u,v]['delW'] = ((meanaverageDel-meanaverage2)/meanaverage2)
    GruralDel.add_edge(u, v,length=savelen)
```

```
plt.hist(averageBetweeness,40)
plt.yscale('log')
#plt.xscale('log')
plt.show()

edges,weights = zip(*nx.get_edge_attributes(Grural3,'del').items())

nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
↪node_size=0,edgelist=edges, edge_color=weights)
plt.axis('equal')
plt.show()


edges,weights = zip(*nx.get_edge_attributes(Grural3,'del').items())

nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
↪node_size=0,edgelist=edges, edge_color=weights,edge_vmin=-0.0,edge_vmax=0.02)
plt.axis('equal')
plt.show()
```

```
0.07830756813807661
0.08337989693921898
```



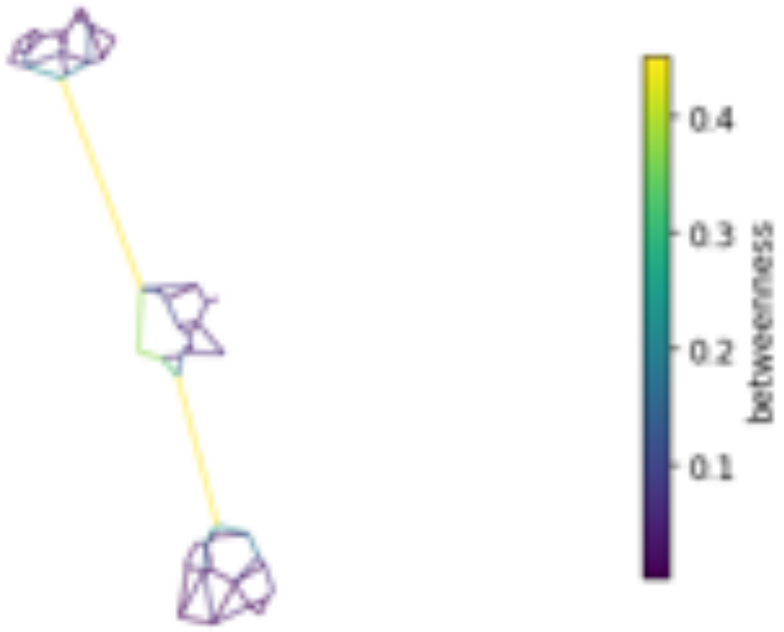
[11]:
```
nodes3=nx.closeness_centrality(Grural3)

#print(edges3)


nx.set_node_attributes(Grural3, nodes3, "closeness")

seismic = plt.cm.get_cmap('coolwarm', 100)
viridis = plt.cm.get_cmap('viridis', 100)

#create and plot graph with edge color

edges,weights = zip(*nx.get_edge_attributes(Grural3,'betweenness').items())

vmin = min(weights)
vmax = max(weights)

nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges, edge_color=weights,edge_vmin=vmin,␣
 ↪edge_vmax=vmax,edge_cmap=viridis)

sm = plt.cm.ScalarMappable(cmap=viridis, norm=plt.Normalize(vmin = vmin,␣
 ↪vmax=vmax))
sm._A = []
cbar=plt.colorbar(sm,shrink=0.7)
cbar.ax.tick_params(labelsize=12)
cbar.ax.set_ylabel('betweenness',fontsize=12)

plt.tight_layout()
plt.axis('equal')

filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_Betweenness.
 ↪pdf'
plt.savefig(filename)

plt.show()

###########

#create and plot graph with edge color

edges,weights = zip(*nx.get_edge_attributes(Grural3,'del').items())

#delprop=[]
#for u,v in GBawue2.edges(data=False):
#    delprop.append(GBawue2.edges[u,v]['del'])
```

```
vmin = -max(weights)
vmax = max(weights)

#nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges,␣
 ↪edge_color=weights,edge_cmap=seismic,edge_vmin=min(delprop),edge_vmax=-min(delprop))
nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges, edge_color=weights,edge_vmin=vmin,␣
 ↪edge_vmax=vmax,edge_cmap=seismic)

sm = plt.cm.ScalarMappable(cmap=seismic, norm=plt.Normalize(vmin = vmin,␣
 ↪vmax=vmax))
sm._A = []
cbar=plt.colorbar(sm,shrink=0.5)
cbar.ax.tick_params(labelsize=10)
cbar.ax.set_ylabel('betweenness impact',fontsize=10)

plt.tight_layout()
plt.axis('equal')

filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_BetweennessImpact.
 ↪pdf'
plt.savefig(filename)

plt.show()

###########




for u,v in Grural3.edges(data=False):
    if Grural3.edges[u,v]['del']<0:
        Grural3.edges[u,v]['del2'] = min(weights)
    elif Grural3.edges[u,v]['del']>0:
        Grural3.edges[u,v]['del2'] = -min(weights)
    else:
        Grural3.edges[u,v]['del2'] = 0


#create and plot graph with edge color

edges,weights = zip(*nx.get_edge_attributes(Grural3,'del2').items())
seismic2 = plt.cm.get_cmap('coolwarm', 2)
```

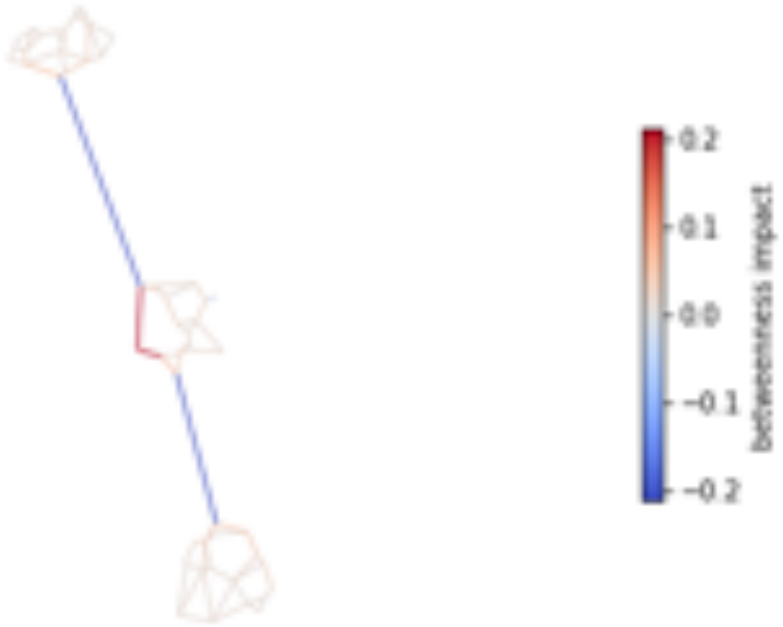
```
#delprop=[]
#for u,v in GBawue2.edges(data=False):
#    delprop.append(GBawue2.edges[u,v]['del'])

vmin = min(weights)
vmax = -min(weights)

#nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
↪node_size=0,edgelist=edges,␣
↪edge_color=weights,edge_cmap=seismic,edge_vmin=min(delprop),edge_vmax=-min(delprop))
nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
↪node_size=0,edgelist=edges, edge_color=weights,edge_vmin=vmin,␣
↪edge_vmax=vmax,edge_cmap=seismic2)

sm = plt.cm.ScalarMappable(cmap=seismic2, norm=plt.Normalize(vmin = vmin,␣
↪vmax=vmax))
sm._A = []
cbar=plt.colorbar(sm,shrink=0.4)
cbar.ax.tick_params(labelsize=10)
cbar.ax.set_ylabel('betweenness impact',fontsize=10)

plt.tight_layout()
plt.axis('equal')

filename='Figures1/
↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_BetweennessImpact2.
↪pdf'
plt.savefig(filename)

plt.show()

###########


#create and plot graph with edge color

nodes,weights = zip(*nx.get_node_attributes(Grural3,'betweenness').items())

vmin = min(weights)
vmax = max(weights)

nx.draw(Grural3, nx.get_node_attributes(Grural3, 'pos'), with_labels=False,␣
↪node_size=10,nodelist=nodes, node_color=weights,vmin=vmin,␣
↪vmax=vmax,cmap=viridis)

sm = plt.cm.ScalarMappable(cmap=viridis, norm=plt.Normalize(vmin = vmin,␣
↪vmax=vmax))
```

```
sm._A = []
cbar=plt.colorbar(sm,shrink=0.7)
cbar.ax.tick_params(labelsize=12)
cbar.ax.set_ylabel('betweenness',fontsize=12)

plt.tight_layout()
plt.axis('equal')

filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_BetweennessNode.
 ↪pdf'
plt.savefig(filename)

plt.show()

############


#scatter plot

delprop2=[]
for u,v in Grural3.edges(data=False):
    delprop2.append([Grural3.edges[u,v]['del'],Grural3.
 ↪edges[u,v]['betweenness']])
delprop2a=np.asarray(delprop2)

plt.scatter(delprop2a[:,0],delprop2a[:,1])
plt.xlabel('betweenness impact',fontsize=12)
plt.ylabel('betweenness',fontsize=12)

plt.tight_layout()
plt.axis('equal')

filename='Figures1/
 ↪CC-'+str(numCC)+'_CS-'+str(ruralspots)+'_RR-'+str(ruralradius)+'_ScatterBetween.
 ↪pdf'
plt.savefig(filename)
plt.show()
```

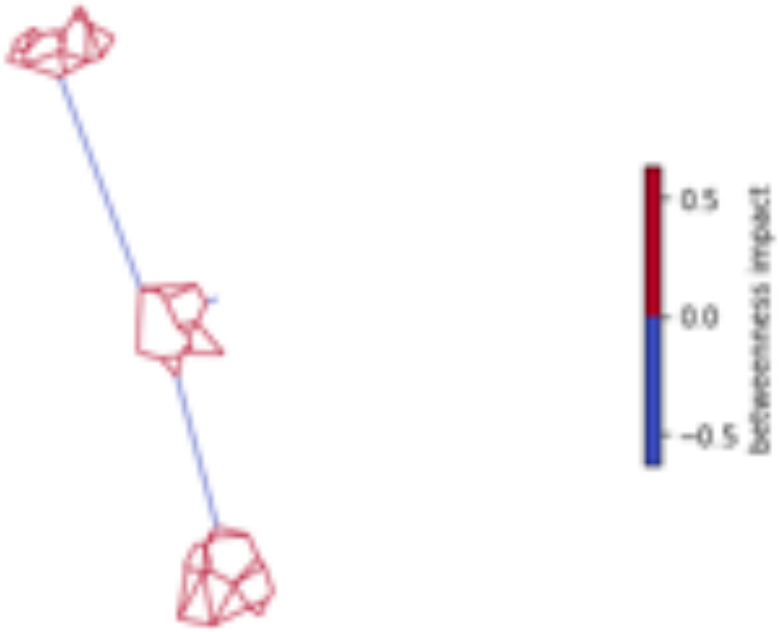
```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:26: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```



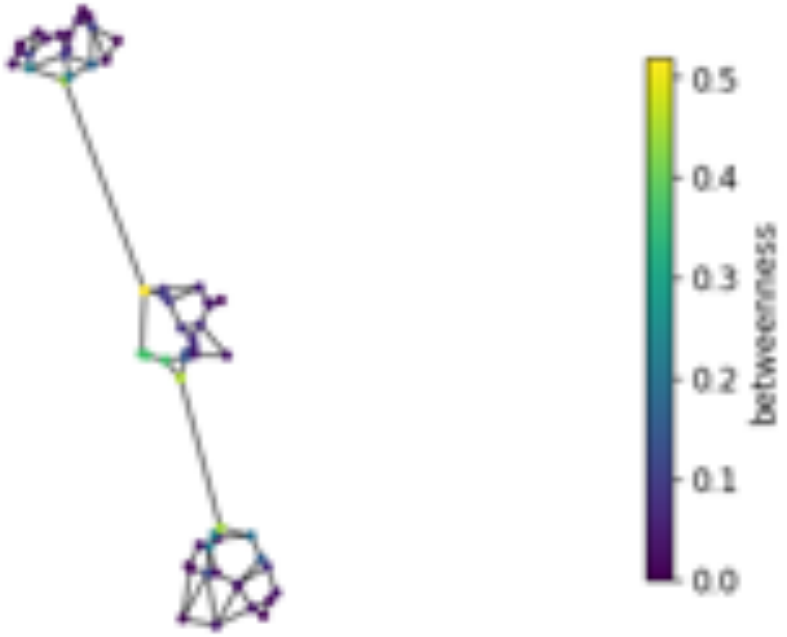

```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:56: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```

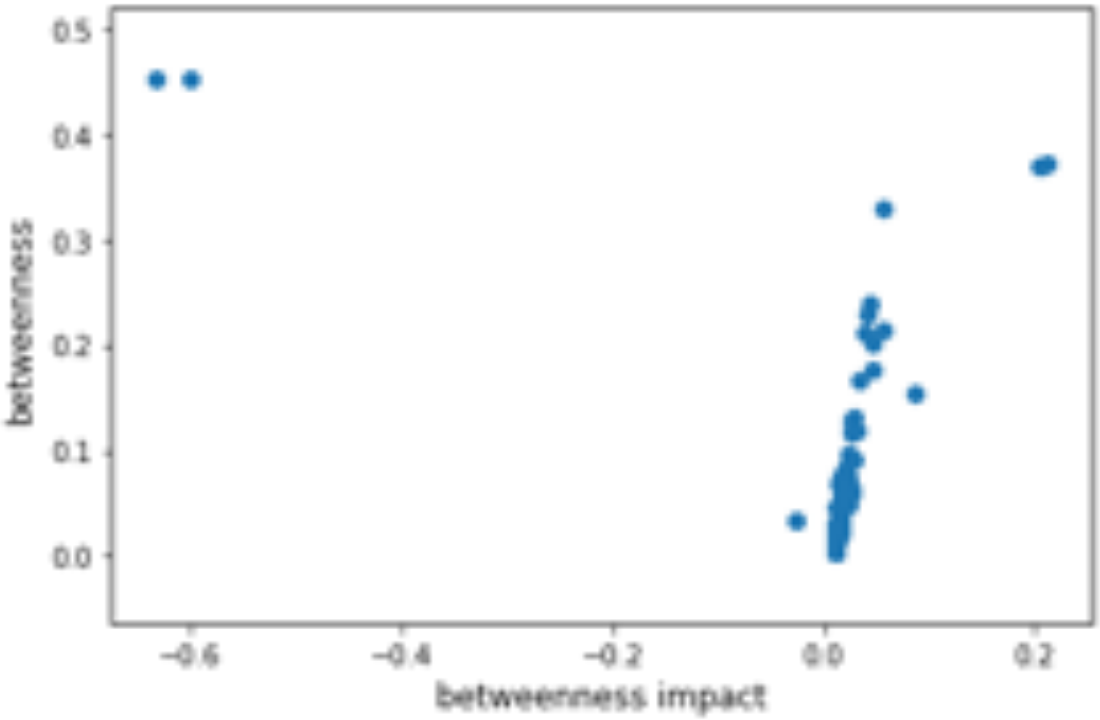



```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:99: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```




```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:125: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```


0.5
0.4
0.3
0.2
0.1
0.0
betweenness


0.5
0.4
0.3
0.2
0.1
0.0
betweenness
−0.6
−0.4
−0.2
0.0
0.2
betweenness impact

[ ]:

RuralStreetBaWue

August 15, 2021

[1]:
```
#load libraries
import pickle as pkl
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
import csv
import collections
from scipy.spatial import Delaunay
```

[2]:
```
# Load copies of network
G = pkl.load(open("StreetNetworkData/bawue_ruralgraph_drivenet.pkl", "rb"))
GBawue=pkl.load(open("StreetNetworkData/bawue_ruralgraph_drivenet.pkl", "rb"))
GBawue2=pkl.load(open("StreetNetworkData/bawue_ruralgraph_drivenet.pkl", "rb"))
```

[3]:
```
plt.subplot(121)
nx.draw(G, nx.get_node_attributes(G, 'pos'), with_labels=False, node_size=0)
plt.axis('equal')
plt.show()
```

[5]:
```
#Delete Selfloops

for u,v in G.edges:
    if u==v:
        G.remove_edge(u, v)
        GBawue.remove_edge(u, v)
        GBawue2.remove_edge(u, v)

#Plot for Checking
plt.subplot(121)
nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=0)
plt.axis('equal')
plt.show()
```

[6]:
```
#correct positions

#get mean lattitude
newpos=[]
collectlat=[]
for e in G.nodes:

    poss=G.nodes[e]['pos']
    #print(poss)
    collectlat.append(poss[1])
```

```
meanlat=(max(collectlat)+min(collectlat))/2
for e in G.nodes:

    poss=G.nodes[e]['pos']
    #print(poss)
    long=poss[0]
    latt=poss[1]
    ycoord=113.2*latt
    xcoord= 40075 * np.cos( meanlat*np.pi/180 ) / 360*long
    newpos.append([xcoord,ycoord])
    GBawue.nodes[e]['pos']=[xcoord,ycoord]
    GBawue2.nodes[e]['pos']=[xcoord,ycoord]

newposa=np.asarray(newpos)

#Plot for checking plausibility
print([max(newposa[:,0]),min(newposa[:,0])])
print([max(newposa[:,1]),min(newposa[:,1])])
plt.xlim([min(newposa[:,0]),max(newposa[:,0])])
plt.ylim([min(newposa[:,1]),max(newposa[:,1])])
nx.draw(GBawue, nx.get_node_attributes(GBawue, 'pos'), with_labels=False,␣
 ↪node_size=0)
plt.axis('equal')
plt.show()

#Generate weigthed edge distance between junctions
collectalllengths=[]
for u,v in GBawue2.edges(data=False):
    newposa1=GBawue.nodes[u]['pos']
    newposa2=GBawue.nodes[v]['pos']
    GBawue2.edges[u,v]['length'] = np.
 ↪sqrt((newposa1[0]-newposa2[0])**2+(newposa1[1]-newposa2[1])**2)
    collectalllengths.append(np.
 ↪sqrt((newposa1[0]-newposa2[0])**2+(newposa1[1]-newposa2[1])**2))
```

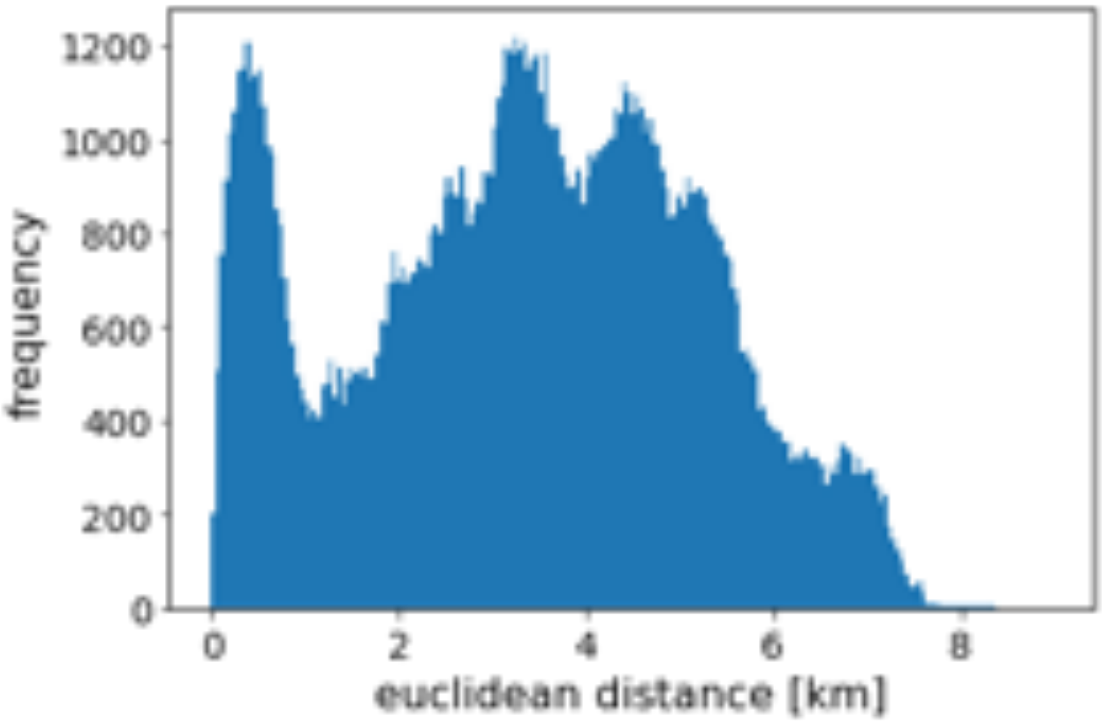
```
[650.4372070684761, 642.9492954523464]
[5590.19112216, 5584.26264572]
```

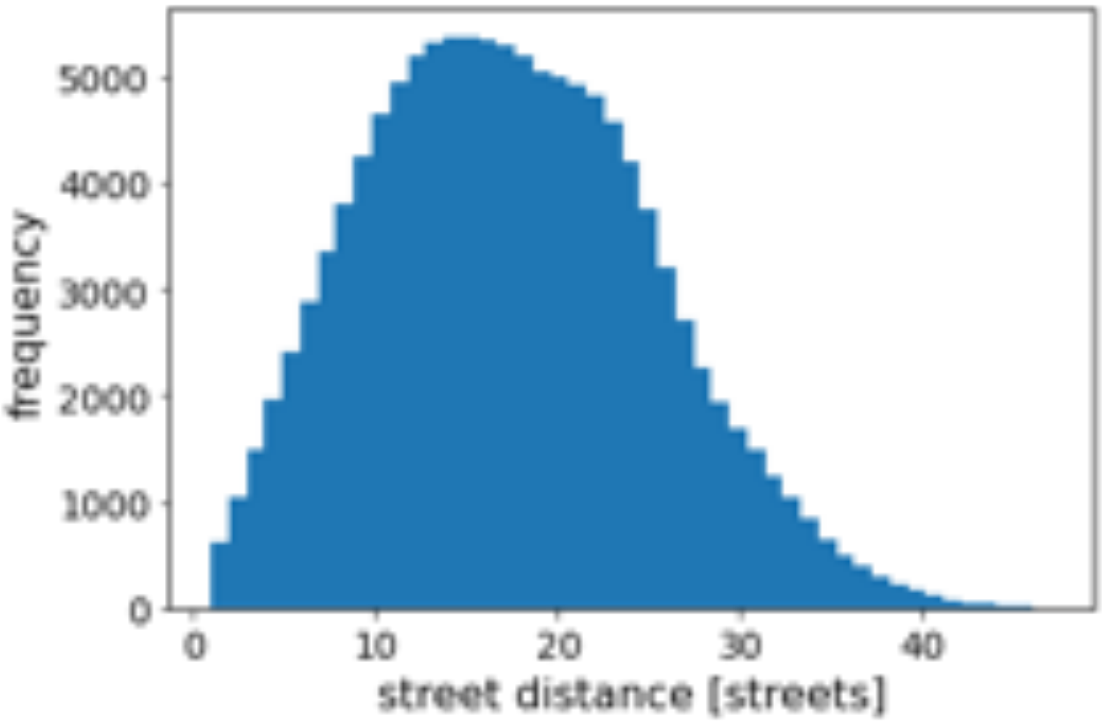
[7]:
```
#get betweeness
edges3=nx.edge_betweenness_centrality(GBawue2,normalized=True)
nodes3=nx.betweenness_centrality(GBawue2,normalized=True)
edges4=nx.edge_betweenness_centrality(GBawue2,weight='length',normalized=True)
nodes4=nx.betweenness_centrality(GBawue2,weight='length',normalized=True)

#set betweeness
nx.set_edge_attributes(GBawue2, edges4, "betweenness2")
nx.set_node_attributes(GBawue2, nodes4, "betweenness2")

nx.set_edge_attributes(GBawue2, edges3, "betweenness")
nx.set_node_attributes(GBawue2, nodes3, "betweenness")

#get average betweeness
averageBetweeness=[]
for u,v in GBawue2.edges(data=False):
    averageBetweeness.append(GBawue2.edges[u,v]['betweenness'])
meanaverage=np.mean(averageBetweeness)
print(meanaverage)

averageBetweeness2=[]
for u,v in GBawue2.edges(data=False):
    averageBetweeness2.append(GBawue2.edges[u,v]['betweenness2'])
meanaverage2=np.mean(averageBetweeness2)
print(meanaverage2)
```

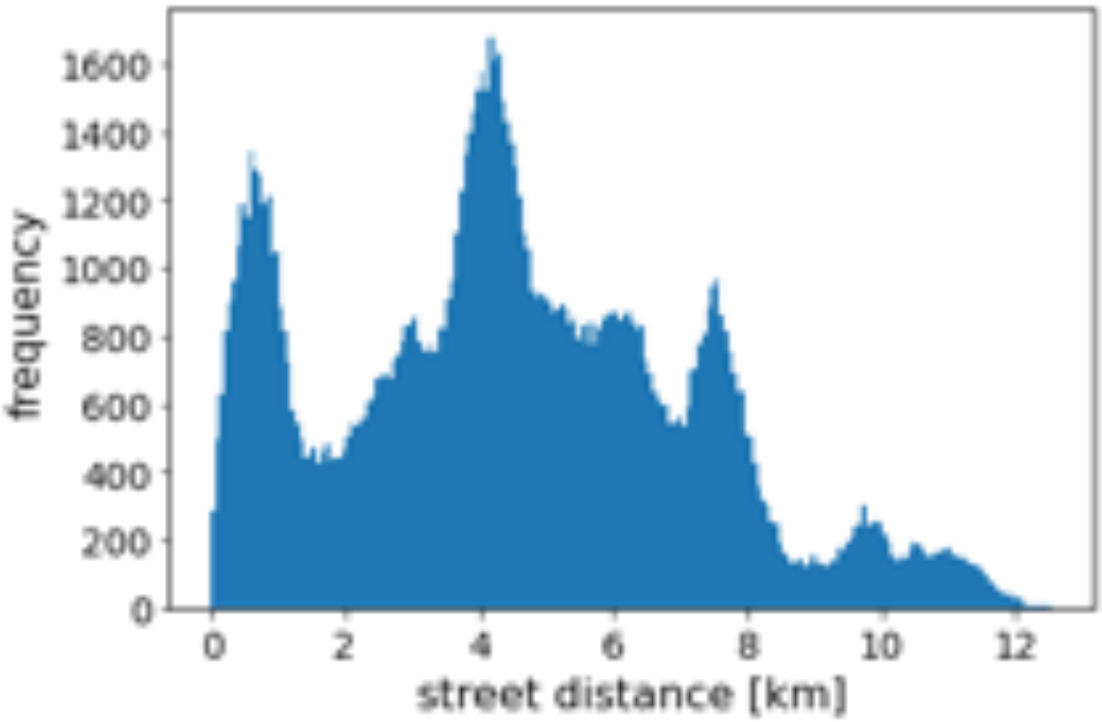
```
#Generate euclidean distance between junctions
totaldistanceED=[]
for u in range(len(newposa)):
    for v in range(len(newposa)):
        if u<v:
            newdistance=np.
 ↪sqrt((newposa[u,0]-newposa[v,0])**2+(newposa[u,1]-newposa[v,1])**2)
            totaldistanceED.append(newdistance)

#plot histogram
plt.hist(totaldistanceED,200)
plt.ylabel('frequency',fontsize=16)
plt.xlabel('euclidean distance [km]',fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.xlim(0,5.2)
filename='Figures2/BaWue_DistanceEuc.pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()

#Generate edge distance between junctions
totaldistanceEDD=[]
lendistance=dict(nx.all_pairs_shortest_path_length(GBawue2))
#print(lendistance[0])
for u in GBawue2.nodes:
    for v in GBawue2.nodes:
        if u<v:
            newdistance=lendistance[u][v]
            totaldistanceEDD.append(newdistance)

#plot histogram
plt.hist(totaldistanceEDD,np.max(totaldistanceEDD))
plt.ylabel('frequency',fontsize=16)
plt.xlabel('street distance [streets]',fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.xlim(0,50)
filename='Figures2/BaWue_DistanceStreets.pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()


#Generate edge distance between junctions (weighted)
totaldistanceWED=[]
```

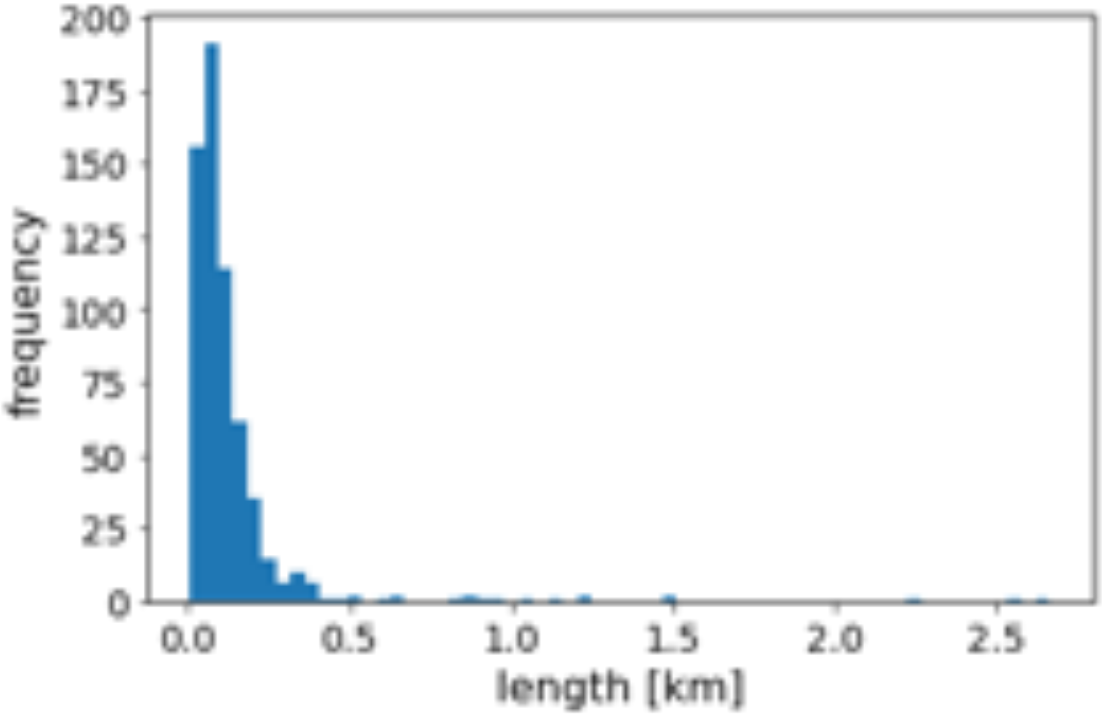
```
lendistance=dict(nx.all_pairs_dijkstra_path_length(GBawue2, cutoff=None,␣
 ↪weight='length'))
for u in GBawue2.nodes:
    for v in GBawue2.nodes:
        if u<v:
            newdistance=lendistance[u][v]
            totaldistanceWED.append(newdistance)


#plot histogram
plt.hist(totaldistanceWED,200)
plt.ylabel('frequency',fontsize=16)
plt.xlabel('street distance [km]',fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.xlim(0,6.7)
filename='Figures2/Bawue_DistanceStreetsWeight.pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()

plt.hist(collectalllengths,60)
plt.ylabel('frequency',fontsize=16)
plt.xlabel('length [km]',fontsize=16)
plt.xticks(fontsize=14)
plt.yticks(fontsize=14)
#plt.xlim(0,3)
filename='Figures2/BaWue_LengthStreets.pdf'
plt.tight_layout()
plt.savefig(filename)
plt.show()
```

```
0.028133733059979255
0.03164682511361092
```


1200
1000
800
600
400
200
0
frequency
0
2
4
6
8
euclidean distance [km]


5000
4000
3000
2000
1000
0
frequency
0
10
20
30
40
street distance [streets]


frequency
1600
1400
1200
1000
800
600
400
200
0
0
2
4
6
8
10
12
street distance [km]


frequency
200
175
150
125
100
75
50
25
0
0.0
0.5
1.0
1.5
2.0
2.5
length [km]

[8]:
```
#compute impact

#create copy of graph
GBawueDel=pkl.load(open("StreetNetworkData/bawue_ruralgraph_drivenet.pkl",␣
↪"rb"))

for x,y in GBawueDel.edges(data=False):
    if x==y:
        GBawueDel.remove_edge(x, y)
    else:
        newposa1=GBawue.nodes[x]['pos']
        newposa2=GBawue.nodes[y]['pos']
        GBawueDel.edges[x,y]['length'] = np.
↪sqrt((newposa1[0]-newposa2[0])**2+(newposa1[1]-newposa2[1])**2)

counter=0

#compute new average impact after having deleted edges
for u,v in GBawue2.edges(data=False):
    newposa1=GBawue.nodes[u]['pos']
    newposa2=GBawue.nodes[v]['pos']

    GBawueDel.remove_edge(u, v)
    edgesDel=nx.edge_betweenness_centrality(GBawueDel,normalized=True)
    nx.set_edge_attributes(GBawueDel, edgesDel, "betweenness")
    averageBetweenessDel=[]
    for x,y in GBawueDel.edges(data=False):
        averageBetweenessDel.append(GBawueDel.edges[x,y]['betweenness'])

    meanaverageDel=np.mean(averageBetweenessDel)
    #print([u,v,(meanaverageDel-meanaverage)/meanaverage])
    GBawue2.edges[u,v]['del'] = ((meanaverageDel-meanaverage)/meanaverage)
    GBawueDel.add_edge(u, v, length=np.
↪sqrt((newposa1[0]-newposa2[0])**2+(newposa1[1]-newposa2[1])**2))
    counter=counter+1
    #print(counter/len(averageBetweeness2))

print('done1')
```

```
done1
```

[9]:
```
#Plot results

nodes3=nx.closeness_centrality(GBawue)

#print(edges3)
```

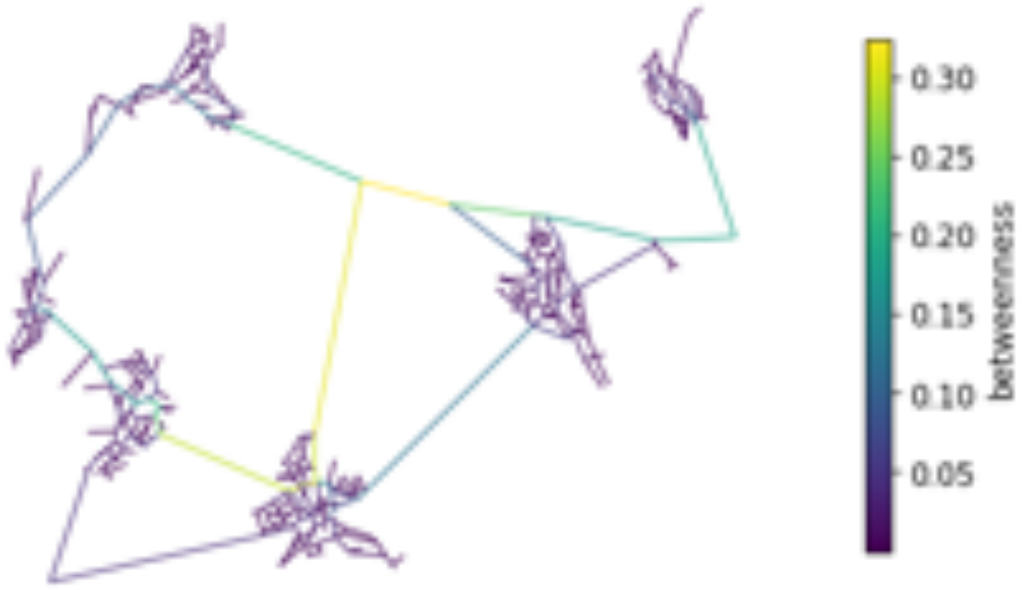
```
nx.set_node_attributes(GBawue2, nodes3, "closeness")

seismic = plt.cm.get_cmap('coolwarm', 100)
viridis = plt.cm.get_cmap('viridis', 100)

#create and plot graph with edge color

edges,weights = zip(*nx.get_edge_attributes(GBawue2,'betweenness').items())

vmin = min(weights)
vmax = max(weights)

nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges, edge_color=weights,edge_vmin=vmin,␣
 ↪edge_vmax=vmax,edge_cmap=viridis)

sm = plt.cm.ScalarMappable(cmap=viridis, norm=plt.Normalize(vmin = vmin,␣
 ↪vmax=vmax))
sm._A = []
cbar=plt.colorbar(sm,shrink=0.7)
cbar.ax.tick_params(labelsize=12)
cbar.ax.set_ylabel('betweenness',fontsize=12)

plt.tight_layout()
plt.axis('equal')

filename='Figures2/BaWue_Betweenness.pdf'
plt.savefig(filename)

plt.show()

###########

#create and plot graph with edge color

edges,weights = zip(*nx.get_edge_attributes(GBawue2,'del').items())

#delprop=[]
#for u,v in GBawue2.edges(data=False):
#    delprop.append(GBawue2.edges[u,v]['del'])

vmin = min(weights)
vmax = -min(weights)
```

```
#nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges,␣
 ↪edge_color=weights,edge_cmap=seismic,edge_vmin=min(delprop),edge_vmax=-min(delprop))
nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges, edge_color=weights,edge_vmin=vmin,␣
 ↪edge_vmax=vmax,edge_cmap=seismic)

sm = plt.cm.ScalarMappable(cmap=seismic, norm=plt.Normalize(vmin = vmin,␣
 ↪vmax=vmax))
sm._A = []
cbar=plt.colorbar(sm,shrink=0.7)
cbar.ax.tick_params(labelsize=12)
cbar.ax.set_ylabel('betweenness impact',fontsize=12)

plt.tight_layout()
plt.axis('equal')

filename='Figures2/BaWue_BetweennessImpact.pdf'
plt.savefig(filename)

plt.show()

############




for u,v in GBawue2.edges(data=False):
    if GBawue2.edges[u,v]['del']<0:
        GBawue2.edges[u,v]['del2'] = min(weights)
    elif GBawue2.edges[u,v]['del']>0:
        GBawue2.edges[u,v]['del2'] = -min(weights)
    else:
        GBawue2.edges[u,v]['del2'] = 0


#create and plot graph with edge color

edges,weights = zip(*nx.get_edge_attributes(GBawue2,'del2').items())
seismic2 = plt.cm.get_cmap('coolwarm', 2)

#delprop=[]
#for u,v in GBawue2.edges(data=False):
#    delprop.append(GBawue2.edges[u,v]['del'])

vmin = min(weights)
vmax = -min(weights)
```

```
#nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges,␣
 ↪edge_color=weights,edge_cmap=seismic,edge_vmin=min(delprop),edge_vmax=-min(delprop))
nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=0,edgelist=edges, edge_color=weights,edge_vmin=vmin,␣
 ↪edge_vmax=vmax,edge_cmap=seismic2)

sm = plt.cm.ScalarMappable(cmap=seismic2, norm=plt.Normalize(vmin = vmin,␣
 ↪vmax=vmax))
sm._A = []
cbar=plt.colorbar(sm,shrink=0.7)
cbar.ax.tick_params(labelsize=12)
cbar.ax.set_ylabel('betweenness impact',fontsize=12)

plt.tight_layout()
plt.axis('equal')

filename='Figures2/BaWue_BetweennessImpact2.pdf'
plt.savefig(filename)

plt.show()

###########


#create and plot graph with edge color

nodes,weights = zip(*nx.get_node_attributes(GBawue2,'betweenness').items())

vmin = min(weights)
vmax = max(weights)

nx.draw(GBawue2, nx.get_node_attributes(GBawue2, 'pos'), with_labels=False,␣
 ↪node_size=10,nodelist=nodes, node_color=weights,vmin=vmin,␣
 ↪vmax=vmax,cmap=viridis)

sm = plt.cm.ScalarMappable(cmap=viridis, norm=plt.Normalize(vmin = vmin,␣
 ↪vmax=vmax))
sm._A = []
cbar=plt.colorbar(sm,shrink=0.7)
cbar.ax.tick_params(labelsize=12)
cbar.ax.set_ylabel('betweenness',fontsize=12)

plt.tight_layout()
plt.axis('equal')
```

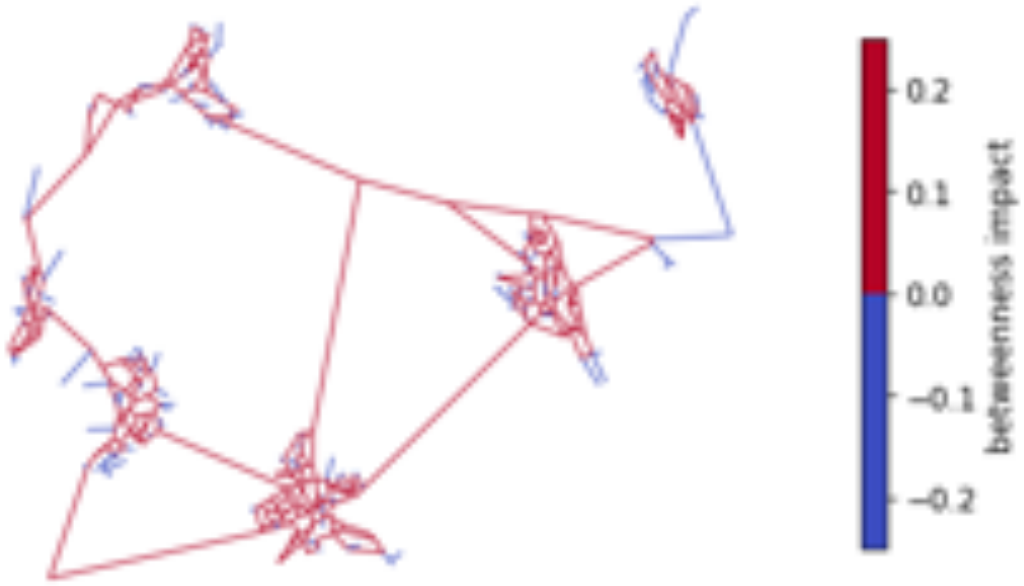
```
filename='Figures2/BaWue_BetweennessNode.pdf'
plt.savefig(filename)

plt.show()

###########


#scatter plot

delprop2=[]
for u,v in GBawue2.edges(data=False):
    delprop2.append([GBawue2.edges[u,v]['del'],GBawue2.
 ↪edges[u,v]['betweenness']])
delprop2a=np.asarray(delprop2)

plt.scatter(delprop2a[:,0],delprop2a[:,1])
plt.xlabel('betweenness impact',fontsize=12)
plt.ylabel('betweenness',fontsize=12)

plt.tight_layout()
plt.axis('equal')

filename='Figures2/BaWue_ScatterBetween.pdf'
plt.savefig(filename)
plt.show()
```

```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:28: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```

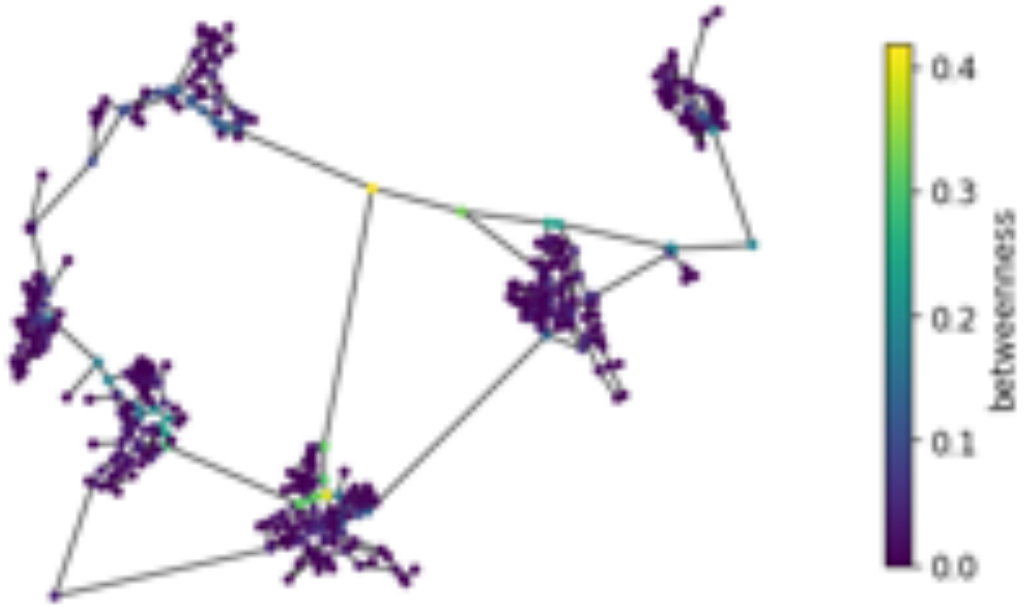


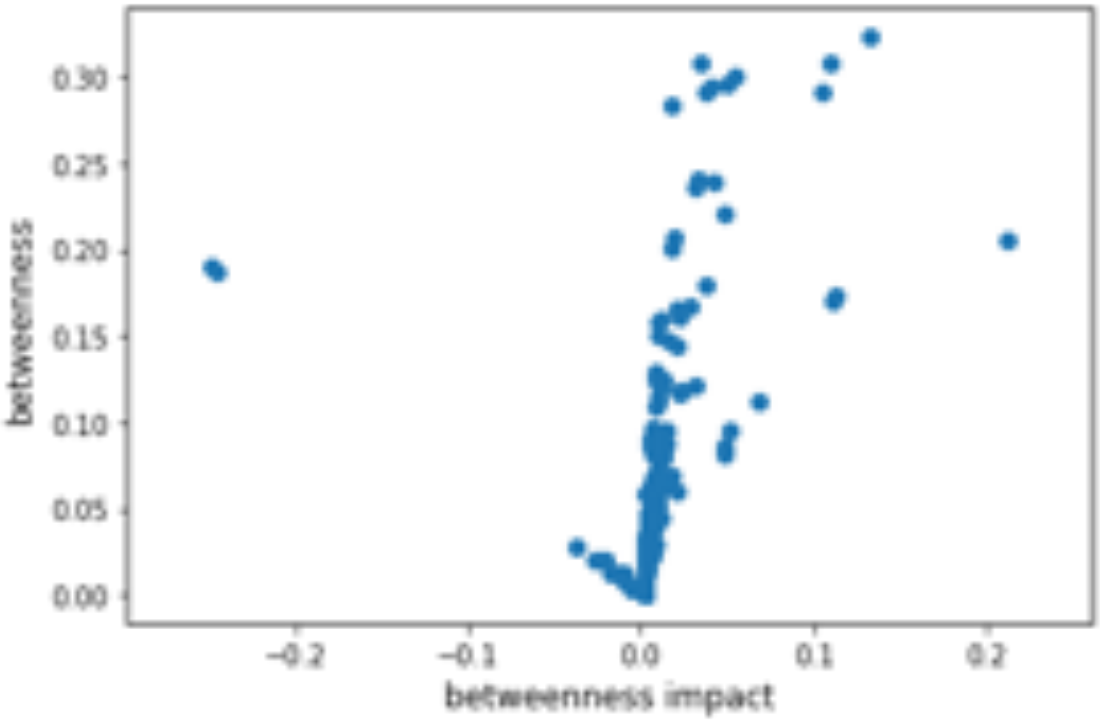

```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:58: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```


```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:101: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```




```
//anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:127: UserWarning:
This figure includes Axes that are not compatible with tight_layout, so results
might be incorrect.
```


0.4
0.3
0.2
0.1
0.0
betweenness


0.30
0.25
0.20
0.15
0.10
0.05
0.00
betweenness
−0.2
−0.1
0.0
0.1
0.2
betweenness impact

[ ]: