Modeling and Analyzing Urban Networks and Amenities with OSMnx [*]

Geoff Boeing [†]

University of Southern California

July 2024

Abstract

OSMnx is a Python package for downloading, modeling, analyzing, and visualizing urban networks and any other geospatial features from OpenStreetMap data. A large and growing body of literature uses it to conduct scientific studies across the disciplines of geography, urban planning, transport engineering, computer science, and others. The OSMnx project has recently developed and implemented many new features, modeling capabilities, and analytical methods. The package now encompasses substantially more functionality than was previously documented in the literature. Accordingly, this paper introduces the OSMnx package's new capabilities, usage, and design—in addition to the scientific theory and logic underlying them. Finally, it reflects on tool building in geographical information science and the implications for urban modeling and analysis in the mode of open science.

1. Introduction

Modeling networked urban infrastructure—such as streets, rails, canals, etc.—and geospatial features—such as amenities, points of interest, etc.—underpins research into travel behavior, accessibility, public health, sustainability, and spatial equity. This research spans the disciplines of urban morphology (e.g., Gervasoni et al., 2017; Nilsson

[*]Citation info: Boeing, G. 2024. Modeling and Analyzing Urban Networks and Amenities with OSMnx. Working paper. https://geoffboeing.com/publications/osmnx-paper/

[†]Correspondence: boeing@usc.edu

and Gil, 2019; Coutrot et al., 2022), transportation planning (e.g., Merchán et al., 2020; Liao et al., 2020; Natera Orozco et al., 2020), and network science (e.g., Feng and Porter, 2020; Yin et al., 2020; Young and Eccles, 2020). However, traditional limitations of data availability, inconsistent digitization standards, and lack of well-documented, reusable tools historically limited the reproducibility, generalizability, scalability, and usefulness of empirical urban network modeling (Liu et al., 2022). In response to similar challenges across many disciplines, the open science movement calls for reproducibility and transparency—in particular through building, sharing, and utilizing open-source analytical tools (Rey, 2009; Singleton et al., 2016; Rey, 2019; Kedron et al., 2021). Like their peers across disciplines, urban researchers need good tools for good science.

This paper describes OSMnx, a Python package that allows users to easily download, model, analyze, and visualize urban networks and geospatial features from OpenStreetMap data. Users can download and model walking, driving, biking, or custom networks with a single line of code and then analyze and visualize them. Users can easily work with urban amenities and points of interest, building footprints, transit stops, elevation data, street orientations, speed and travel time, and routing.

The OSMnx package was first released as a beta in 2016 and that inchoate version was documented in a preliminary paper (Boeing, 2017). At the time, it was the first tool enabling users to automatically download and model the street network geometry and topology of any city in the world as a graph and then analyze and visualize it (Boeing, 2020b). It has since become a widespread tool for a broad range of urban modeling and analysis in the intervening years (e.g., Coutrot et al., 2022; Nilsson and Gil, 2019; Feng and Porter, 2020; Gervasoni et al., 2017; Liao et al., 2020; Natera Orozco et al., 2020; Yin et al., 2020; Young and Eccles, 2020).

However, since its initial release, the package has evolved substantially by expanding its functionality, improving performance, and stabilizing a user-friendly API. To document these start-of-the-art capabilities and their scientific underpinnings in the literature, this paper presents OSMnx's current modular structure, functionality, and uses for urban modeling and analysis. The following section introduces the theory, data, and tools of street network modeling and analysis, including OSMnx's place in that landscape. Then it presents the OSMnx package's functionality, organization, and usage. Finally this paper concludes with a brief discussion of implications for urban network analysis and tool building in geographical information science.

2. Street Network Models and Analysis

Numerous sources cover spatial network theory and analysis (e.g., Tinkler, 1979; Barnes and Harary, 1983; Gastner and Newman, 2006; Barthelemy, 2011; Ducruet and Beauguitte, 2014; O'Sullivan, 2014; Marshall et al., 2018). This section focuses on street networks and offers an overview of necessary concepts then briefly reviews the current tool landscape.

2.1. Model Fundamentals

Real-world networks are commonly modeled as mathematical *graphs* (Trudeau, 1994; Barthelemy, 2022). A graph, G, is a data structure consisting of two sets: one set, N, contains *nodes* that are linked to each other by the second set, E, containing node pairs called *edges*. Let $G = (N, E)$ and $\{u, v\} \subseteq N$ and $\{u, v\} \in E$. We can say for graph G that: 1) edge $\{u, v\}$ links nodes u and v, 2) edge $\{u, v\}$ is *incident* to u and to v, 3) u is *adjacent* to v and vice versa, and 4) u and v are *neighbors* (Newman, 2010). An adjacency matrix can fully represent a graph by defining these adjacent node pairs.

A node's *degree* is how many edges are incident to that node. For example, a node with degree 2 has two incident edges which are consequently adjacent to each other. An edge can be *undirected* (linking two nodes bidirectionally), *directed* (linking one-way from a source node to a target node), or a *self-loop* (linking one node back to itself). A graph with directed edges is a directed graph, or *digraph*. A graph that allows multiple edges to link a single pair of nodes is a *multigraph* and those multiple edges are called *parallel* edges.

A *path* is a sequence of edges linking a sequence of nodes. The graph *distance* between two nodes is the count of edges in the shortest such path between them. A *weighted* graph's distance is the sum of some edge impedance attribute (e.g., length or time) along the shortest path minimizing that sum. A directed graph is strongly *connected* if a path exists between each ordered pair of nodes, and it is weakly connected if such a path exists only if its edges are undirected. A disconnected graph contains multiple connected *components*—each a disjoint set of nodes forming its own connected subgraph.

Spatial networks' nodes and/or edges are embedded in space (Barthelemy, 2022). Spatial graphs thus model both topology and geometry (O'Sullivan, 2014). *Topology* refers to the structure and configuration of the nodes and edges, whereas *geometry* encompasses positions, lengths, angles, etc. A *planar* graph's topology can be represented in a two-dimensional plane such that its edges intersect only at nodes (Barthelemy and Flammini, 2008). Most street networks are nonplanar due to the occasional presence of overpasses and underpasses, but their spatial embedding and "approximate" planarity constrain their topological characteristics relative to other kinds of complex networks (Boeing, 2020a). A *primal* graph models a street network's intersections and dead-ends as nodes and its street segments as edges (Porta et al., 2006b). A *line* graph (sometimes called a *dual* graph) inverts this topology to instead model streets as nodes and their intersections as edges (Porta et al., 2006a), though it discards much of the network's geometry (Ratti, 2004).

2.2. Street Network Analysis

Many common geometric measures of spatial networks are often applied to street networks. In general, these should use undirected representations of the graph to avoid double-counting bidirectional streets relative to one-ways. Intersection density (i.e., the count of nodes with degree >1, normalized by network area) is perhaps the most common such measure of network "grain" in transport planning and urban design (e.g., Ewing and Cervero, 2010). The average street segment length (i.e., mean edge length) offers a linear proxy of block size. Street density is the sum of edge lengths normalized by network area. Circuity can take on different interpretations, including average circuity: the sum of edge lengths divided by the sum of great-circle distances between adjacent node pairs, representing the inverse of network edge straightness (Boeing, 2019).

Additionally, many topological measures in network science are often applied to street networks (Barthelemy, 2011). The *average node degree* (i.e., mean number of edges incident to the nodes) indicates graph *connectedness*[1] and is perhaps the most common topological measure in transport planning and urban design (e.g., Barrington-Leigh and Millard-Ball, 2015, 2017a, 2020). Networks with high connectedness can be more robust against perturbation as they offer alternate routing options if parts of the network fail (Boeing and Ha, 2024). Various measures of *centrality* are also common (Crucitti et al., 2006). For example, a node's *betweenness centrality* measures the share of all graph shortest paths that pass through the node (Barthelemy, 2004; Barthelemy et al., 2013). The graph's maximum betweenness centrality indicates the share of shortest paths that rely on its most important node: high values suggest possible chokepoints that represent single points of failure, such as a bridge connecting a city's halves across a river (Boeing and Ha, 2024).

In street network analysis, researchers use these geometric and topological measures to characterize a network's form. Such analyses often also employ path solving (Miller, 1999; Wang et al., 2020). For example, accessibility analyses solve shortest paths (e.g., by length or travel time) from origin nodes (e.g., homes) to destination nodes representing the locations of amenities (e.g., workplaces, schools, transit stops, parks, greengrocers, hospitals, etc.) to measure access (Foti, 2014; Liu et al., 2022). Disaster analyses often simulate emergency responses or evacuations along the network to understand the flows following a disaster (Sasabe et al., 2020; Tamakloe et al., 2021). These kinds of analyses can reveal differential outcomes for different communities or locations with the city, offering guidance for practitioners' interventions.

2.3. Street Network Tools and Data

Various tools exist to model and analyze spatial networks like street networks. For example, ESRI's ArcGIS software includes a Network Analyst extension and QGIS offers plug-ins for network analysis. However, such GIS tools' network capabilities are usually fairly limited. Conversely, dedicated network analysis tools such as Gephi, igraph, graph-tool, and NetworkX offer robust network analysis functionality but limited geospatial capabilities. Other dedicated spatial network tools exist, often for specific analytical purposes, including PySAL spaghetti for network inference (Gaboardi et al., 2021; Rey et al., 2022), Pandana for accessibility analysis (Foti, 2014), momepy for urban morphology (Fleischmann, 2019), and stplanr for transport planning (Lovelace and Ellison, 2019).

Street network data come from various sources, including governmental sources like the US Census Bureau's TIGER/Line shapefiles which represent network geometry but lack sufficient topological information to build a properly nonplanar model. Alternatively, OpenStreetMap offers a worldwide, public web mapping platform and geospatial database that anyone contribute to, with some editorial oversight, and includes streets, highways, transit, buildings, footpaths, cycleways, points of interest, and political boundaries (Jokar Arsanjani et al., 2015). Although its coverage varies, it includes network geometry and topology and its data quality is generally high—particularly so in urban areas, with notable exceptions in China and sub-Saharan Africa (Barron et al., 2014; Barrington-Leigh and Millard-Ball, 2017b). OpenStreetMap's data model comprises three *element* types: *nodes* (points), *ways* (either open ways representing lines or closed ways representing polygons), and *relations* (i.e., between nodes and/or ways). These elements can possess one or more *tags*: key-value pairs containing attribute data. OpenStreetMap data are available to download from third-party services, such as Geofabrik, and web APIs including Overpass and Nominatim.

The problem, however, was historically that building a street network model from raw OpenStreetMap data was fairly difficult. The Overpass API can be cumbersome to extract the right data to construct an appropriate graph model. Analysts usually had to write extensive ad hoc code to process the raw data into a useful graph model and then write more code to conduct analyses. Their decisions on how to handle ubiquitous street network characteristics—such as directedness, planarity, self-loops, parallel edges, or culs-de-sac—frequently went undocumented. With every research team writing its own ad hoc code, innumerable small modeling decisions inevitably would go unreported in the resulting peer-reviewed literature—but these small decisions add up to significantly restrict interpretability and replicability.

OSMnx fits into this landscape by allowing users to build a spatial network model anywhere in the world by automatically downloading raw data from OpenStreetMap and building a NetworkX model with spatial information. This allows users to sidestep the ad hoc coding and replace the many small modeling decisions with a well-documented and transparent common tool. Accordingly, OSMnx has become a standard tool in the literature for both retrieving OpenStreetMap data broadly and for modeling street networks specifically (Boeing, 2020b). The following section provides a current[2] introduction to the package's organization, capabilities, and usage.

3. The OSMnx Package

OSMnx (pronounced as the initialism: "oh-ess-em-en-ex") is a free, open-source, and fully type-hinted package written in pure Python. The package is structured as a collection of modules that organize related functionality (Table 1). It is built on top of and uses the data structures of NetworkX (Hagberg et al., 2008), a network analysis Python package, and GeoPandas (Van den Bossche et al., 2024), a Python package for working with geospatial dataframes. It interacts with three public web APIs to collect data: the OpenStreetMap Nominatim API, the Overpass API, and the Google Maps Elevation API (or equivalent with the same interface). This section presents OSMnx's organization and capabilities.

3.1. Geocoding and Querying

OSMnx geocodes place names and addresses with the Nominatim API. Users can use the `geocoder` module to geocode place names or addresses to latitude-longitude point coordinates. Or, they can retrieve place boundaries or any other OpenStreetMap elements by name or ID. Using the `features` and `graph` modules, as described below, users can download OpenStreetMap data from the Overpass API by latitude-longitude point coordinates, address, bounding box, bounding polygon/multipolygon, or place name (e.g., neighborhood, city, county, etc.).

3.2. Urban Amenities

Using OSMnx's `features` module, users can search for and download any geospatial features (e.g., building footprints, grocery stores, schools, public parks, transit stops, etc.) from the OpenStreetMap Overpass API as a GeoPandas `GeoDataFrame` object. This uses OpenStreetMap tags to search for matching elements.

3.3. Modeling Networks

OSMnx models spatial networks as primal, nonplanar, weighted, directed multigraphs with possible self-loops—specifically, these are NetworkX `MultiDiGraph` data structures (Hagberg et al., 2008). Using OSMnx's `graph` module, users can download any spatial network data (such as streets, paths, rail, canals, power lines, etc.) from the Overpass API and model them as a `MultiDiGraph`.

OSMnx models a one-way street as a single directed edge from node u to node v, but a bidirectional street is modeled with two reciprocal directed edges (with identical

Table 1. The OSMnx public API's modules and the functionality they expose.

Module	Functionality
`bearing`	Calculate graph edge compass bearings and orientation entropy.
`convert`	Convert graph to/from different data types.
`distance`	Calculate spatial distances and find nearest graph node/edge(s) to point(s).
`elevation`	Attach node elevations from raster files or a Google Maps compatible elevation API, and calculate edge grades.
`features`	Download OSM geospatial features' geometries and attributes, such as points of interest, building footprints, transit stops, etc.
`geocoder`	Geocode place names or addresses or retrieve OSM elements by place name or ID, via the Nominatim API.
`graph`	Download and create graphs from OSM data, using filters to query the Overpass API for built-in network types or a custom filter.
`io`	Save/load graphs to/from GraphML, GeoPackage, or OSM XML files.
`plot`	Visualize street networks, routes, orientations, and geospatial features.
`projection`	Project spatial graph to a different coordinate reference system.
`routing`	Calculate graph edge speeds, travel times, and weighted shortest paths between nodes.
`settings`	Configure global package settings.
`simplification`	Simplify and consolidate spatial graph nodes and edges.
`stats`	Calculate geometric and topological network measures.
`truncate`	Truncate spatial graph by distance, bounding box, or polygon.
`utils`	General utility functions.
`utils_geo`	Miscellaneous geospatial utility functions.

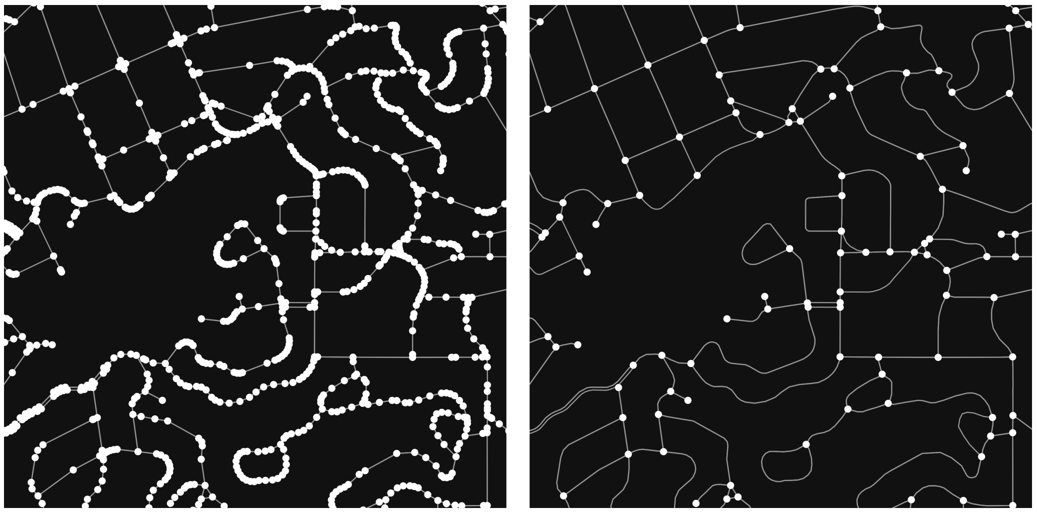

Figure 1. Graph of a town's street network before (left) and after (right) edge simplification.

geometries)—one from u to v and another from v to u—to represent both possible directions of flow. Because these graphs are nonplanar, they correctly model the topology of interchanges, overpasses, and underpasses. That is, edge crossings in a two-dimensional plane are not nodes in an OSMnx model unless they represent true junctions in the three-dimensional real world.

The `graph` module uses filters to query the Overpass API: users can either specify a built-in network type or provide their own custom filter written in OverpassQL. Under the hood, OSMnx does several things to generate the best possible model. It initially buffers the query area by 500 meters to create the initial graph before truncating it to the user's desired query area. This ensures accurate streets-per-node counts by attenuating graph perimeter effects. As the graph may contain many small disconnected components around the perimeter, by default it returns the largest weakly connected component. OSMnx also automatically simplifies the graph topology as discussed below.

3.4. Graph Simplification

The `simplification` module automatically processes the network's topology from the original raw OpenStreetMap data to ensure that individual nodes represent individual intersections or dead-ends and edges represent the street segments that link them. This graph simplification is of two primary types—edge simplification and node consolidation—described in detail in Boeing (2024) and summarized here.

Edge simplification, put simply, merges adjacent edges for a better model. It cleans up the graph's topology so that nodes represent intersections or dead-ends and edges represent street segments as illustrated in Figure 1. This is important because in OpenStreetMap's raw data, ways comprise sets of straight-line segments between nodes: that is, nodes are vertices for streets' curving line geometries, not just intersections and dead-ends. By default, OSMnx simplifies this topology by deleting non-intersection/dead-end nodes, merging the edges between them into a new "simplified" edge, and retaining the complete true edge geometry as an edge attribute. When multiple OpenStreetMap ways are thus merged into a new graph edge, the ways' attribute values can be aggregated into a single value.

Node consolidation, put simply, merges nearby nodes for a better model. This is important because many real-world street networks feature complex intersections and traffic circles, resulting in a cluster of graph nodes where there is really just one true intersection as we would consider it in transport planning or urban design. Similarly, divided roads are often represented by separate centerline edges. The intersection of 2 divided roads thus creates 4 nodes where each edge intersects a perpendicular edge—but these 4 nodes represent a single intersection in the real world. OSMnx can consolidate such complex intersections into a single node and optionally rebuild the graph's edge topology accordingly. When multiple OpenStreetMap nodes are thus merged into a new graph node, the nodes' attribute values can be aggregated into a single value.

Edge simplification and node consolidation offer several benefits. They produce a more accurate model that better represents the real world. This in turn yields more accurate network measures, for example by not overcounting complex intersections when calculating intersection density or by not underrepresenting street segment lengths. Finally, many graph algorithms' time complexity scales with node or edge count. By generating a graph with (often drastically) fewer nodes and edges—yet no loss of accuracy—many algorithms will complete much faster. This matters most when analyzing large urban networks, where runtime becomes an issue.

3.5. Converting, Projecting, and Saving

OSMnx's `convert` module can convert a NetworkX `MultiDiGraph` model to a NetworkX `MultiGraph` if the user prefers an undirected representation of the network for specific analytical purposes, as discussed in the background section. It can also convert to a NetworkX `DiGraph` if the user prefers a directed graph without any parallel edges. OSMnx can also convert a `MultiDiGraph` to and from node and edge GeoPandas `GeoDataFrame` objects. The resulting nodes `GeoDataFrame` is indexed by OpenStreetMap node ID, and the resulting edges `GeoDataFrame` is multi-indexed by endpoint node u, endpoint node v, and a *key* (to differentiate parallel edges), just as a `MultiDiGraph` edge is identified by its u, v, *key* ordered triplet. This also allows users to load arbitrary node and edge ShapeFiles or GeoPackage layers as `GeoDataFrames` then convert them to a `MultiDiGraph` for network analysis.

As these models are all spatial graphs, they have coordinate reference system (CRS) metadata. OSMnx's default CRS is EPSG:4326, but users can project a graph to any other CRS using the `projection` module. If a user is unsure which CRS to project to, OSMnx can automatically determine an appropriate Universal Transverse Mercator CRS for the operation, based on the graph nodes' centroid.

Finally, using the `io` module, users can save a graph to disk as a GraphML file (to load into other network analysis software, including OSMnx), a GeoPackage (to load into other GIS software), or an OSM XML file (the standard OpenStreetMap data interchange format).

3.6. Elevation

Topography is essential to understanding street network form, but street network analyses too often ignore elevation as it can be difficult to acquire and attach to a graph model (Boeing, 2022). OSMnx's `elevation` module lets users automatically add elevation attributes to a spatial graph's nodes from either a local raster file or the Google Maps Elevation API. Once all nodes have elevation attributes, users can automatically calculate edge grades (i.e., rise-over-run inclines), analyze the steepness of certain streets, or use elevation change in an impedance function for routing, as discussed below.

3.7. Map Matching and Routing

OSMnx offers basic map matching and routing functionality. The `distance` module can match a list of coordinates to each's nearest node or edge using a fast spatial index and vectorized operation. This can be useful for converting an origin-destination matrix of coordinates (such as geocoded addresses) into corresponding nearest nodes for route solving.

The `routing` module can solve shortest paths for network routing—parallelized with multiprocessing—using different weights (e.g., distance, travel time, elevation change, etc.). OpenStreetMap has a "maxspeed" tag representing streets' maximum speed limits, but it tends to be sparse for most cities. To address this problem, the `routing` module can impute missing edge maximum speeds based on observed values across other edges of the same type in the graph. Such imputation can be imprecise, but the user can override it by passing per-type local speed limits. Once all edges have maximum speed attributes, the module can also automatically calculate free-flow traversal times for each edge.

3.8. Network Measures

The `stats` module can calculate a variety of geometric and topological measures of the network (Boeing, 2022). These measures define streets as the edges in an undirected representation of the graph to prevent double-counting the bidirectional edges of a two-way street. Users can automatically calculate common measures from transport planning, urban design, and network science, including intersection density, circuity, average node degree, centrality, and many others. Users can also use NetworkX directly to calculate dozens of additional topological network measures. Additionally, OSMnx's `bearing` module can calculate the streets' compass bearings and the network's orientation entropy (Boeing, 2019).

3.9. Visualization

Users can visualize network characteristics, routes, network figure-ground diagrams (Boeing, 2021), building footprints, and network orientation polar histograms (Boeing, 2019) using OSMnx's `plot` module. Users can also easily explore networks, routes, or urban amenities as interactive web maps.

3.10. Installation and Configuration

The OSMnx project's source code[3] is publicly hosted on GitHub. Users can install the package from the PyPI package repository using pip or from the Anaconda package repository using conda, as detailed in the package documentation[4] and its installation instructions. To get started, users can read the documentation's Getting Started guide then work through the OSMnx Examples Gallery[5] for step-by-step tutorials, feature demonstrations, and sample code.

Once installed and imported, users can configure OSMnx through its `settings` module. Here they can adjust logging behavior, server response caching, server endpoints (including pointing to locally hosted instances), and much more. Users can also configure OSMnx to retrieve historical snapshots of OpenStreetMap data as of a certain date.

4. Discussion

The open science movement calls for accessible, reusable, and well-documented research tools to support better science. The OSMnx project seeks to contribute to these goals in urban science by offering such a tool for modeling and analyzing urban networks and amenities anywhere in the world from OpenStreetMap data. It embodies the relevant domain theory from urban planning, network science, and geographical information science in an easy-to-use tool. In turn, researchers no longer have to reinvent the wheel and make dozens of ad hoc modeling decisions for each analytics project.

Along these lines, the OSMnx project aims to improve reproducibility by allowing users to automatically model networks with clearly defined spatial extents, thoroughly documented modeling decisions, and theoretically-sophisticated results. For example, its graph models are primal, nonplanar, directed, weighted, multigraphs with possible self-loops to capture the diversity of network types and characteristics around the world. Its graph creation methods attenuate perimeter effects, and its graph simplification methods generate models that flexibly match real-world expectations and use.

This paper documented the package's current organization, capabilities, and scientific underpinnings. Better tools, led by academic domain experts, represent one of the key paths forward toward better and more open science.

Acknowledgments

The author wishes to thank the developers and maintainers of the many other packages on which OSMnx depends, as well as the contributors who have lent their time and expertise in proposing features, solving bugs, and contributing code to the OSMnx code base over the years.

Notes

[1]The separate term *connectivity* has a distinct definition in graph theory, but is less useful for spatial network analysis because approximate planarity sharply constrains it: almost all street networks have connectivity equal to 1.

[2]This working paper documents OSMnx version 2.0.0 prior to its official release.

[3]The OSMnx source code is hosted at https://github.com/gboeing/osmnx

[4]The OSMnx documentation is available at https://osmnx.readthedocs.io/en/latest/

[5]The OSMnx Examples Gallery is hosted at https://github.com/gboeing/osmnx-examples

References

Barnes, J. A. and Harary, F. (1983). Graph Theory in Network Analysis. *Social Networks*, 5(2):235–244.

Barrington-Leigh, C. and Millard-Ball, A. (2015). A century of sprawl in the United States. *Proceedings of the National Academy of Sciences*, 112(27):8244–8249.

Barrington-Leigh, C. and Millard-Ball, A. (2017a). More connected urban roads reduce US GHG emissions. *Environmental Research Letters*, 12(4):044008.

Barrington-Leigh, C. and Millard-Ball, A. (2017b). The world's user-generated road map is more than 80% complete. *PLOS ONE*, 12(8):e0180698.

Barrington-Leigh, C. and Millard-Ball, A. (2020). Global trends toward urban street-network sprawl. *Proceedings of the National Academy of Sciences*, 117(4):1941–1950.

Barron, C., Neis, P., and Zipf, A. (2014). A Comprehensive Framework for Intrinsic OpenStreetMap Quality Analysis. *Transactions in GIS*, 18(6):877–895.

Barthelemy, M. (2004). Betweenness centrality in large complex networks. *The European Physical Journal B: Condensed Matter and Complex Systems*, 38(2):163–168.

Barthelemy, M. (2011). Spatial Networks. *Physics Reports*, 499(1-3):1–101.

Barthelemy, M. (2022). *Spatial Networks: A Complete Introduction*. Springer International, Cham.

Barthelemy, M., Bordin, P., Berestycki, H., and Gribaudi, M. (2013). Self-organization versus top-down planning in the evolution of a city. *Scientific Reports*, 3.

Barthelemy, M. and Flammini, A. (2008). Modeling Urban Street Patterns. *Physical Review Letters*, 100(13).

Boeing, G. (2017). OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks. *Computers, Environment and Urban Systems*, 65:126–139.

Boeing, G. (2019). Urban Spatial Order: Street Network Orientation, Configuration, and Entropy. *Applied Network Science*, 4(1):67.

Boeing, G. (2020a). Planarity and Street Network Representation in Urban Form Analysis. *Environment and Planning B: Urban Analytics and City Science*, 47(5):855–869.

Boeing, G. (2020b). The Right Tools for the Job: The Case for Spatial Science Tool-Building. *Transactions in GIS*, 24(5):1299–1314.

Boeing, G. (2021). Spatial Information and the Legibility of Urban Form: Big Data in Urban Morphology. *International Journal of Information Management*, 56:102013.

Boeing, G. (2022). Street Network Models and Indicators for Every Urban Area in the World. *Geographical Analysis*, 54(3):519–535.

Boeing, G. (2024). Graph simplification solutions to the street intersection miscount problem.

Boeing, G. and Ha, J. (2024). Resilient by Design: Simulating Street Network Disruptions across Every Urban Area in the World. *Transportation Research Part A: Policy and Practice*, 182:104016.

Coutrot, A., Manley, E., Goodroe, S., Gahnstrom, C., Filomena, G., Yesiltepe, D., Dalton, R. C., Wiener, J. M., Hölscher, C., Hornberger, M., and Spiers, H. J. (2022). Entropy of city street networks linked to future spatial navigation ability. *Nature*, 604(7904):104–110.

Crucitti, P., Latora, V., and Porta, S. (2006). Centrality measures in spatial networks of urban streets. *Physical Review E*, 73(3):036125.

Ducruet, C. and Beauguitte, L. (2014). Spatial Science and Network Science: Review and Outcomes of a Complex Relationship. *Networks and Spatial Economics*, 14(3-4):297–316.

Ewing, R. and Cervero, R. (2010). Travel and the Built Environment: A Meta-Analysis. *Journal of the American Planning Association*, 76(3):265–294.

Feng, M. and Porter, M. A. (2020). Spatial applications of topological data analysis: Cities, snowflakes, random structures, and spiders spinning under the influence. *Physical Review Research*, 2(3):033426.

Fleischmann, M. (2019). momepy: Urban Morphology Measuring Toolkit. *Journal of Open Source Software*, 4(43):1807.

Foti, F. (2014). *Behavioral Framework for Measuring Walkability and its Impact on Home Values and Residential Location Choices*. Dissertation, University of California, Berkeley, CA.

Gaboardi, J., Rey, S., and Lumnitz, S. (2021). spaghetti: spatial network analysis in PySAL. *Journal of Open Source Software*, 6(62):2826.

Gastner, M. T. and Newman, M. E. J. (2006). The spatial structure of networks. *The European Physical Journal B: Condensed Matter and Complex Systems*, 49(2):247–252.

Gervasoni, L., Bosch, M., Fenet, S., and Sturm, P. (2017). Calculating spatial urban sprawl indices using open data. In *15th International Conference on Computers in Urban Planning and Urban Management*, Adelaide, Australia.

Hagberg, A. A., Schult, D. A., and Swart, P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. In Varoquaux, G., Vaught, T., and Millman, J., editors, *Proceedings of the 7th Python in Science Conference*, pages 11–15. SciPy 2008, Pasadena, CA.

Jokar Arsanjani, J., Zipf, A., Mooney, P., and Helbich, M., editors (2015). *OpenStreetMap in GIScience*. Springer International, Cham, Switzerland.

Kedron, P., Li, W., Fotheringham, S., and Goodchild, M. (2021). Reproducibility and replicability: opportunities and challenges for geospatial research. *International Journal of Geographical Information Science*, 35(3):427–445.

Liao, Y., Gil, J., Pereira, R. H. M., Yeh, S., and Verendel, V. (2020). Disparities in travel times between car and transit: Spatiotemporal patterns in cities. *Scientific Reports*, 10(1):4056.

Liu, S., Higgs, C., Arundel, J., Boeing, G., Cerdera, N., Moctezuma, D., Cerin, E., Adlakha, D., Lowe, M., and Giles-Corti, B. (2022). A Generalized Framework for Measuring Pedestrian Accessibility around the World Using Open Data. *Geographical Analysis*, 54(3):559–582.

Lovelace, R. and Ellison, R. (2019). stplanr: A Package for Transport Planning. *The R Journal*, 10(2):7.

Marshall, S., Gil, J., Kropf, K., Tomko, M., and Figueiredo, L. (2018). Street Network Studies: from Networks to Models and their Representations. *Networks and Spatial Economics*, 18:735–749.

Merchán, D., Winkenbach, M., and Snoeck, A. (2020). Quantifying the impact of urban road networks on the efficiency of local trips. *Transportation Research Part A: Policy and Practice*, 135:38–62.

Miller, H. J. (1999). Measuring space-time accessibility benefits within transportation networks: basic theory and computational procedures. *Geographical analysis*, 31(1):1–26.

Natera Orozco, L. G., Battiston, F., Iñiguez, G., and Szell, M. (2020). Data-driven strategies for optimal bicycle network growth. *Royal Society Open Science*, 7(12):201130.

Newman, M. E. J. (2010). *Networks: An Introduction*. Oxford University Press, Oxford, England.

Nilsson, L. and Gil, J. (2019). The Signature of Organic Urban Growth: Degree Distribution Patterns of the City's Street Network Structure. In D'Acci, L., editor, *The Mathematics of Urban Morphology*, pages 93–121. Springer, Cham, Switzerland.

O'Sullivan, D. (2014). Spatial Network Analysis. In Fischer, M. M. and Nijkamp, P., editors, *Handbook of Regional Science*, pages 1253–1273. Springer-Verlag, Berlin, Germany.

Porta, S., Crucitti, P., and Latora, V. (2006a). The Network Analysis of Urban Streets: A Dual Approach. *Physica A: Statistical Mechanics and its Applications*, 369(2):853–866.

Porta, S., Crucitti, P., and Latora, V. (2006b). The network analysis of urban streets: a primal approach. *Environment and Planning B: Planning and Design*, 33(5):705–725.

Ratti, C. (2004). Space syntax: some inconsistencies. *Environment and Planning B: Planning and Design*, 31(4):487–499.

Rey, S. J. (2009). Show Me the Code: Spatial Analysis and Open Source. *Journal of Geographical Systems*, 11(2):191–207.

Rey, S. J. (2019). PySAL: The First 10 Years. *Spatial Economic Analysis*, 14(3):273–282.

Rey, S. J., Anselin, L., Amaral, P., Arribas-Bel, D., Cortes, R. X., Gaboardi, J. D., Kang, W., Knaap, E., Li, Z., Lumnitz, S., Oshan, T. M., Shao, H., and Wolf, L. J. (2022). The PySAL Ecosystem: Philosophy and Implementation. *Geographical Analysis*, 54(3):467–487.

Sasabe, M., Fujii, K., and Kasahara, S. (2020). Road network risk analysis considering people flow under ordinary and evacuation situations. *Environment and Planning B: Urban Analytics and City Science*, 47(5):759–774.

Singleton, A. D., Spielman, S., and Brunsdon, C. (2016). Establishing a framework for Open Geographic Information science. *International Journal of Geographical Information Science*, 30(8):1507–1521.

Tamakloe, R., Hong, J., Tak, J., and Park, D. (2021). Finding evacuation routes using traffic and network structure information. *Transportation Research Part D: Transport and Environment*, 95:102853.

Tinkler, K. J. (1979). Graph theory. *Progress in Geography*, 3(1):85–116.

Trudeau, R. J. (1994). *Introduction to Graph Theory*. Dover Publications, New York, NY, 2nd edition.

Van den Bossche, J., Jordahl, K., Fleischmann, M., Richards, M., McBride, J., Wasserman, J., Badaracco, A. G., Snow, A. D., Ward, B., Tratner, J., Gerard, J., Perry, M., Farmer, C., Hjelle, G. A., Taves, M., Hoeven, E. t., Cochran, M., Bell, R., rraymondgh, Bartos, M., Roggemans, P., Culbertson, L., Caria, G., Eubank, N., sangarshanan, Flavin, J., Rey, S., Gardiner, J., and Kaushik (2024). geopandas/geopandas: v0.14.4.

Wang, M., Chen, Z., Mu, L., and Zhang, X. (2020). Road network structure and ride-sharing accessibility: A network science perspective. *Computers, Environment and Urban Systems*, 80:101430.

Yin, Y., Varadarajan, J., Wang, G., Wang, X., Sahrawat, D., Zimmermann, R., and Ng, S.-K. (2020). A Multi-task Learning Framework for Road Attribute Updating via Joint Analysis of Map Data and GPS Traces. In *Proceedings of The Web Conference 2020*, WWW '20, pages 2662–2668, Taipei, Taiwan. Association for Computing Machinery.

Young, D. L. and Eccles, C. (2020). Automatic construction of Markov decision process models for multi-agent reinforcement learning. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II*, volume 11413, page 114130Y. International Society for Optics and Photonics.