# rnet

*Release 0.2.0*

**Kota Sakazaki**

**May 17, 2022**

# CONTENTS:

CHAPTER

# ONE

# MODEL

**class** rnet.models.**Model**(*name=None*)

Bases: object

Class for representing a road network model.

**Parameters** **name** (str, optional) – Model name. If None, a name is generated automatically. Default: None.

**Variables**

- **name** (str) – Model name.
- **sources** (Dict[str, DataContainer]) – Dictionary mapping source type to data container.
- **built** (bool) – Whether the model has been built.
- **crs** (int) – EPSG code in which node coordinates are represented.
- **nodes** (pandas.DataFrame) – Data frame containing node data.
- **edges** (pandas.DataFrame) – Data frame containing edge data.
- **node_count** (int) – Number of nodes.
- **edge_count** (int) – Number of edges.

**add**(*source*)

Add data from source to the model.

**Paramaters:** source (str or Data):

**build**(**, crs=4326, include='all', exclude=None, r=0.0005, p=2, verbose=True*)

**Keyword Arguments**

- **crs** (int, optional) – EPSG code for node coordinates. Default: 4326.
- **include** ('all' or List[str], optional) – List of tags to include. If 'all', all tags are included. Default: 'all'.
- **exclude** (List[str], optional) – List of tags to exclude. If None, no tags are excluded. Default: None.
- **r** (float, optional) – Radius for nearest neighbor search in IDW interpolation. Default: 0.0005.
- **p** (int, optional) – Power setting for IDW interpolation. Default: 2.

**Note:** The keyword *include* takes precedence over *exclude*.

**dump**()

Print information about the model.

CHAPTER

TWO

# DATA

## 2.1 Classes

The following classes are used to represent data sources.

### 2.1.1 The MapData Class

**class** rnet.**MapData**(*vertices*, *links*, ***, *crs*, *name*)

Bases: rnet.data.classes.Data

Class for representing map data.

**Parameters**

- **vertices** (pandas.DataFrame) – Frame containing vertex data.
- **links** (pandas.DataFrame) – Frame containing link data.

**Keyword Arguments**

- **crs** (int) – EPSG code of the CRS in which vertex coordinates are represented.
- **name** (str) – Data source name.

**Variables**

- **vertices** (pandas.DataFrame) – Frame containing vertex data.
- **links** (pandas.DataFrame) – Frame containing link data.
- **vertex_count** (int) – Number of vertices.
- **link_count** (int) – Number of links.

**bounds**()

Return the coordinates that define the bounding box for the set of vertices.

**Returns** 4-tuple of the form (xmin, ymin, xmax, ymax).

**Return type** Tuple[float]

**dump**()

Print information about the instance.

**classmethod from_osm**(*path_to_osm*, ***kwargs*)

Instantiate class from an OSM data source.

**Parameters**

- **path_to_osm** (str) – Path to OSM file.
- **name** (str, optional) – Data source name. If unspecified, then the OSM file name is used.

**Keyword Arguments**

- **include** (List[str], optional) – List of tags to include. All tags are included by default.
- **exclude** (List[str], optional) – List of tags to exclude. No tags are excluded by default.

---

**Note:** If required, either the *include* or *exclude* keyword should be given, not both. In the case that both are given, *include* takes precedence and *exclude* is ignored.

---

**out**(*, *crs=None*, *include='all'*, *exclude=None*, *keep_indices=False*)

Export vertex and link data frames.

**Keyword Arguments**

- **crs** (int, optional) – EPSG code of CRS for vertex coordinates. If different from .crs, coordinates are transformed to *crs*. If None, coordinates are not transformed. Default: None.
- **include** (*'all'* or List[str], optional) – List of tags to include. If 'all', all tags are included. Default: 'all'.
- **exclude** (List[str], optional) – List of tags to exclude. If None, no tags are excluded. Default: None.

**Returns** 2-tuple containing .vertices and .links frames with links filtered and vertices transformed.

**Return type** Tuple[pandas.DataFrame, pandas.DataFrame]

---

**Note:** The keyword *include* takes precedence over *exclude*.

---

### 2.1.2 The ElevationData Class

**class** rnet.**ElevationData**(*x*, *y*, *z*, *, *crs*, *name*)

Bases: rnet.data.classes.Data

Class for representing a grid of elevation data points.

**Parameters**

- **x** (numpy.ndarray, shape (*nx*,)) – *x*-coordinates of grid.
- **y** (numpy.ndarray, shape (*ny*,)) – *y*-coordinates of grid.
- **z** (numpy.ndarray, shape (*ny*, *nx*)) – Array of *z*-coordinates.

**Keyword Arguments**

- **crs** (int) – EPSG code of CRS in which $(x, y)$ coordinates are represented.
- **name** (str) – Data source name.

**Variables**

- **x** (numpy.ndarray, shape (*nx*,)) – *x*-coordinates of grid.
- **y** (numpy.ndarray, shape (*ny*,)) – *y*-coordinates of grid.
- **z** (numpy.ndarray, shape (*ny*, *nx*)) – Array of *z*-coordinates.
- **nx** (int) – Grid width.
- **ny** (int) – Grid height.
- **point_count** (int) – Number of data points.
- **xmin** (float) – Minimum *x*-coordinate.
- **xmax** (float) – Maximum *x*-coordinate.
- **ymin** (float) – Minimum *y*-coordinate.
- **ymax** (float) – Maximum *y*-coordinate.
- **zmin** (float) – Minimum *z*-coordinate.
- **zmax** (float) – Maximum *z*-coordinate.

**bounds**()

Return the coordinates that define the three-dimensional bounding box for the set of data points.

**Returns** 6-tuple of the form (xmin, ymin, zmin, xmax, ymax, zmax).

**Return type** Tuple[float]

**dump**()

Print information about the instance.

**classmethod from_tif**(*path_to_tif*, ***kwargs*)

Instantiate class from a TIF data source.

**Parameters** **path_to_tif** (str) – Path to TIF file.

**Keyword Arguments** **name** (str, optional) – Data source name. If unspecified, TIF file name is used.

**get_elev**(*x*, *y*, ***, *r=0.001*, *p=2*)

Return elevation at a single point.

**Parameters**

- **x** (float) – *x*-coordinate.
- **y** (float) – *y*-coordinate.

**Keyword Arguments**

- **r** (float, optional) – Radius for neighboring point search. Default: 0.001.
- **p** (int, optional) – Power setting for IDW interpolation. Default: 2.

**Returns** Elevation at point $(x, y)$.

**Return type** float

**See also:**

**get_elevs()** Returns elevations at multiple points.

**get_elevs**(*points*, ***, *r=0.001*, *p=2*)

Return elevations at multiple points.

**Parameters** **points** (numpy.ndarray, shape (*N*, 2)) – The *N* points at which to compute elevations.

**Keyword Arguments**

- **r** (float, optional) – Radius for neighbor search. Default: 50.
- **p** (int, optional) – Power setting. Default: 2.

**Returns**

**elevations** elevations[i] is the elevation at points[i].

**Return type** List[float]

**See also:**

**get_elev()** Returns elevation at a single point.

**out**()

Export data frame with index .y, columns .x, and values .z.

**Returns** Frame with *y*-coordinates in index and *x*-coordinates in columns.

**Return type** pandas.DataFrame

**query**(*i*, *j*)

Return elevation at (x[i], y[j]).

**Parameters**

- **i** (int) – Row number.
- **j** (int) – Column number.

**Return type** float

**Raises** **IndexError** – If *i* or *j* is out of bounds.

## 2.2 Containers

### 2.2.1 The MapDataContainer Class

**class** rnet.**MapDataContainer**(*name=None*)

Bases: rnet.data.containers.DataContainer

Container for map data.

**Parameters** **name** (str, optional) – Container name. If None, a name is generated automatically. Default: None.

**add**(*source*, *crs=None*)

Add map data to the container.

**Parameters**

- **source** (str or MapData) – Either (1) path to OSM file, (2) path to directory containing vertices.csv and links.csv pair, or (3) MapData instance.

- **crs** (int, optional) – EPSG code of the CRS in which vertex coordinates are represented. Required only if *source* is of type (2).

**out**(*, *assume_unique=False*, *crs=4326*, *include='all'*, *exclude=None*)

Creates a MapData instance containing concatenated frames.

**Keyword Arguments**

- **assume_unique** (bool, optional) – If True, vertices and links in all data sources are assumed to be unique. If False, data sources are checked for uniqueness and only unique features are retained. Default: False.
- **crs** (int, optional) – EPSG code of CRS for vertex coordinates. Default: 4326.
- **include** (*'all'* or List[str], optional) – List of tags to include. If 'all', all tags are included. Default: 'all'.
- **exclude** (List[str], optional) – List of tags to exclude. If None, no tags are excluded. Default: None.

**Returns** MapData instance.

**Return type** *MapData*

**See also:**

**MapData** Class for representing map data.

### 2.2.2 The ElevationDataContainer Class

**class** rnet.**ElevationDataContainer**(*name=None*)

Bases: rnet.data.containers.DataContainer

Container for elevation data.

**Parameters** **name** (str, optional) – Container name. If None, a name is generated automatically. Default: None.

**add**(*source*, *crs=None*)

Adds elevation data to the container.

**Parameters**

- **source** (str or ElevationData) – Either (1) path to TIF file, (2) path to CSV file, or (3) ElevationData instance.
- **crs** (int, optional) – EPSG code of the CRS in which point coordinates are represented. Required only if *source* is of type (2).

**out**(*, *assume_unique=False*, *crs=4326*)

Creates an ElevationData instance containing concatenated frames.

**Keyword Arguments**

- **assume_unique** (bool, optional) – If True, points in all data sources are assumed to be unique. If False, data sources are checked for uniqueness and only unique features are retained. Default: False.
- **crs** (int, optional) – EPSG code of CRS for $(x, y)$ coordinates. Default: 4326.

**Returns** ElevationData instance.

**Return type** *ElevationData*

**See also:**

`ElevationData` Class for representing elevation data.

CHAPTER

# THREE

# TOOLKITS

## 3.1 The Graph Toolkit

rnet.toolkits.graph.**clean_points**(*points*, *connections*)

Removes points that are not used in any connections.

**Parameters**

- **points** (pandas.DataFrame) – Frame containing points with unique indices.
- **connections** (pandas.DataFrame) – Frame containing connections, $(i, j)$, with indices corresponding to those in *points*.

**Returns** *points* frame excluding unused points.

**Return type** pandas.DataFrame

rnet.toolkits.graph.**compute_lengths**(*vertices*, *links*)

Computes the length of each link and inserts or updates the 'length' column.

**Parameters**

- **vertices** (pandas.DataFrame) – Frame containing vertex data.
- **links** (pandas.DataFrame) – Frame containing link data.

**Returns** *links* frame with 'length' column inserted or updated.

**Return type** pandas.DataFrame

rnet.toolkits.graph.**concatenate**(**frames*)

**Parameters** ***frames** (Tuple[pandas.DataFrame, pandas.DataFrame]) – Frames containing points and connections data.

rnet.toolkits.graph.**extract_edges**(*links*, *nodes*, *directed*)

Extracts edges from a set of links. Edges are constructed by chaining one or more links together such that the endpoints are both nodes.

**Parameters**

- **links** (pandas.DataFrame) – Frame containing edge data.
- **nodes** (pandas.DataFrame) – Frame containing node data.
- **directed** (bool) – If True, then $(i, j)$ pairs in the links frame are interpreted as ordered pairs. The resulting set of edges will also be directed.

**Returns** Frame containing edge data.

**Return type** pandas.DataFrame

rnet.toolkits.graph.**extract_nodes**(*vertices*, *links*, *directed*)

Extracts nodes from a set of vertices. Nodes are the subset of vertices that have exactly one or more than two neighbors.

**Parameters**

- **vertices** (pandas.DataFrame) – Frame containing vertex data.
- **links** (pandas.DataFrame) – Frame containing link data.
- **directed** (bool) – If True, then $(i, j)$ pairs in the links frame are interpreted as ordered pairs.

**Returns** *vertices* frame containing only the points with exactly one or more than two neighbors.

**Return type** pandas.DataFrame

rnet.toolkits.graph.**filter_connections**(*connections*, *action*, *tags*)

Filters connections based on their tag.

**Parameters**

- **connections** (pandas.DataFrame) – Frame containing connection data.
- **action** ('include' or 'exclude') – Whether to include or exclude the specified tags.
- **tags** (List[str]) – List of tags to include or exclude.

**Returns** *connections* frame with specified tags included or excluded.

**Return type** pandas.DataFrame

rnet.toolkits.graph.**get_neighbors**(*connections*, *directed*)

Returns dictionary mapping point ID to set of neighboring point IDs.

**Parameters**

- **connections** (pandas.DataFrame) – Frame containing connections.
- **directed** (bool) – If True, then $(i, j)$ pairs in are interpreted as ordered pairs.

**Returns** Mapping from point ID to set of neighboring point IDs.

**Return type** Dict[int, Set[int]]

rnet.toolkits.graph.**reindex_points**(*points*, *connections*, *start=0*)

Resets indices of points to a consecutive range.

**Parameters**

- **points** (pandas.DataFrame) – Frame containing points with unique indices.
- **connections** (pandas.DataFrame) – Frame containing connections, $(i, j)$, with indices corresponding to those in *points*.
- **start** (int, optional) – Starting index. Default: 0.

**Returns** 2-tuple containing *points* and *connections* frames with new point indices.

**Return type** Tuple[pandas.DataFrame, pandas.DataFrame]

## 3.2 The Coordinates Toolkit

rnet.toolkits.coords.**concatenate_points**(**frames*)

Concatenate frames containing point coordinates.

**Parameters** ***frames** (pandas.DataFrame) – Frames containing point coordinates.

**Returns** Concatenated frame.

**Return type** pandas.DataFrame

rnet.toolkits.coords.**create_tree**(*points*)

Return two-dimensional *k*-d tree for quick nearest-neighbor query.

**Parameters** **points** (numpy.ndarray, shape(N,2)) – The N points at which to compute elevations.

**Return type** scipy.spatial.cKDTree

rnet.toolkits.coords.**get_bounds2d**(*coords*)

Return coordinate bounds.

**Parameters** **coords** (numpy.ndarray, shape(N,2)) – Coordinates.

**Returns** 4-tuple containing xmin, ymin, xmax, ``ymax.

**Return type** Tuple[float]

rnet.toolkits.coords.**get_bounds3d**(*coords*)

Return coordinate bounds.

**Parameters** **coords** (numpy.ndarray, shape(N,3)) – Coordinates.

**Returns**

**6-tuple containing xmin, ymin, zmin, xmax,** ymax, zmax.

**Return type** Tuple[float]

rnet.toolkits.coords.**get_crs_info**(*crs*)

Return information about a CRS.

**Parameters** **crs** (int) – EPSG code.

**Returns** Dictionary mapping descriptor to value.

**Return type** Dict[str, Any]

rnet.toolkits.coords.**get_elev**(*xdata*, *ydata*, *zdata*, *x*, *y*, *r*, *p*)

Compute elevation at a single point via inverse distance weighting (IDW) interpolation.

**Parameters**

- **xdata** (numpy.ndarray, shape(nx,)) – *x*-coordinates.
- **ydata** (numpy.ndarray, shape(ny,)) – *y*-coordinates.
- **zdata** (numpy.ndarray, shape(nx,ny)) – *z*-coordinates.
- **x** (float) – *x*-coordinate for elevation query.
- **y** (float) – *y*-coordinate for elevation query.
- **r** (float) – Radius for neighboring point search.
- **p** (int) – Power setting for IDW interpolation.

**Returns** Elevation at point $(x, y)$.

**Return type** float

rnet.toolkits.coords.**get_elevs**(*data*, *tree*, *points*, *r*, *p*)

Return elevations at multiple points. The elevations are computed via inverse distance weighting (IDW) interpolation.

**Parameters**

- **data** (pandas.DataFrame) – Elevation data.
- **tree** (scipy.spatial.cKDTree) – *k*-d tree for nearest-neighbor query.
- **points** (numpy.ndarray, shape(N,2)) – The N points at which to compute elevations.
- **r** (float) – Radius for neighbor search.
- **p** (int) – Power setting for IDW interpolation.

**Return type** List[float]

rnet.toolkits.coords.**indices_in_circle**(*xdata*, *ydata*, *x*, *y*, *r*)

**Parameters**

- **xdata** (numpy.ndarray, shape(nx,)) –
- **ydata** (numpy.ndarray, shape(ny,)) –
- **x** (float) – *x*-coordinate.
- **y** (float) – *y*-coordinate.
- **r** (float) – Circle radius.

**Returns** Array with shape (N,2), where N is the number of points located within the circle.

**Return type** numpy.ndarray

rnet.toolkits.coords.**transform2d**(*coords*, *source*, *destination*)

Transform two-dimensional coordinates from *source* to *destination* CRS.

**Parameters**

- **coords** (numpy.ndarray, shape(N,2)) – Coordinates to transform.
- **source** (int) – EPSG code of source CRS.
- **destination** (int) – EPSG code of destination CRS.

**Returns** Transformed coordinates.

**Return type** numpy.ndarray, shape(N,2)

CHAPTER

# FOUR

# UTILITIES

rnet.utils.layerutils.**get_config**(*layer_type*)

Return configuration dictionary for given layer type.

**Parameters** **layer_type** (LayerType) – Layer type.

**Returns**

Dictionary containing configuration settings. Possible keys are the following:

- **geometry** (*str*) – {'point', 'linestring', 'polygon'}
- **size** (*float*) – point size, default: 1.0
- **width** (*float*) – line width, default: 0.5
- **color** (*Tuple[int, int, int]*) – render color, default: (0, 0, 0)
- **opacity** (*float*) – opacity in range [0, 1)
- **maxscale** (*int* or *float*) – maximum scale
- **minscale** (*int* or *float*) – minimum scale
- **renderer** (*str*) – {'rulebased', 'categorized'}
- **outlinecolor** (*Tuple[int, int, int]*) – outline color for polygon geometry, default: (0, 0, 0)
- **outlinewidth** (*float*) – outline width for polygon geometry, default: 0.5

**Return type** Dict[str, Any]

# PYTHON MODULE INDEX

## r

# INDEX