# Exporting networks to other formats using nx\_pgnet Python module

[Exporting networks to other formats using nx\_pgnet Python module 1](#_Toc346009779)

[Supported Formats 2](#_Toc346009780)

[Spatial Networks 3](#_Toc346009781)

[Example exports – create a database connection 3](#_Toc346009782)

[GEXF 3](#_Toc346009783)

[Pajek 3](#_Toc346009784)

[YAML 4](#_Toc346009785)

[GraphML 4](#_Toc346009786)

[GML 4](#_Toc346009787)

[Gephi-compatible node/edge lists 5](#_Toc346009788)

[Aspatial Networks 7](#_Toc346009789)

[Example exports – create a database connection 7](#_Toc346009790)

[GEXF 7](#_Toc346009791)

[Pajek 7](#_Toc346009792)

[YAML 7](#_Toc346009793)

[GraphML 7](#_Toc346009794)

[GML 7](#_Toc346009795)

[Gephi-compatible node/edge lists 8](#_Toc346009796)

## Supported Formats

NetworkX supports the ability to export graphs / networks to a number of different formats, that can then be used and linked to other software packages capable of handling network data. The definitive list of supported formats can be found at the NetworkX website: <http://networkx.lanl.gov/reference/readwrite.html>.

A subset of those formats are supported within the Python nx\_pgnet module, allowing a user to create a graph / network, store the result in PostGIS, and then also export that to another format. This could be useful if a user wishes to use the network within an alternative software package, perhaps for visualisation for example, or for analysis. Of the formats listed at the above URL, the nx\_pgnet module supports exporting to:

* Graph Exchange XML (GEXF) - <http://gexf.net/format/>
* YAML - <http://www.yaml.org/>
* GraphML - <http://graphml.graphdrawing.org/>
* GML - <http://www.infosun.fim.uni-passau.de/Graphlet/GML/gml-tr.html>
* Gephi-compatible node/edge lists
  + Within the “Data Table” view of Gephi, it is possible to select “Import Spreadsheet” data for defining both nodes, or edges, or both. When importing a set of edges into Gephi, a “Source”, a “Target”, and a “Type” column must be defined. By exporting a network to a set of Gephi-compatible edge lists using pg\_net, the user can be ensured that a “Source” , “Target” and “Type” column are defined.
  + A user will receive a set of coordinates for nodes and edges in SRS EPSG:4326 (WGS84, lat/lng) and SRS EPSG:900913 (Google projection), that can allow the user to use the GeoLayout layout available within Gephi to visualise the network using the coordinates of the nodes and edges.
* Pajek - <http://vlado.fmf.uni-lj.si/pub/networks/pajek/>

**NOTE: Each of the examples listed below do not include how to create a network and store it inside the schema. It is assumed that a user already has a network, and just wishes to understand how they can export that to another format.**

# Spatial Networks

## Example exports – create a database connection

Import nx\_pgnet as nx\_pgnet

Import osgeo.ogr as ogr

db\_connection = ogr.Open("PG: host=’<insert\_server\_host>’ dbname=’<insert\_database\_name>’ user=’<insert\_user\_name>’ password=’<insert\_password>’”)

db\_connection = ogr.Open("PG: host='127.0.0.1' dbname='database' user='postgres', password='password'")

### GEXF

nx\_pgnet.export\_graph(<insert\_database\_connection>).export\_to\_gexf(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_gexf\_to>’, ‘<insert\_name\_of\_output\_gexf\_file>’, ‘<insert\_encoding>’, <insert\_prettyprint\_output\_status>)

nx\_pgnet.export\_graph(db\_connection).export\_to\_gexf(graph,’C:/TEMP/’, ‘TransportNetwork’, ‘utf-8’, True)

**NOTE: node id and label values set to equal a tuple of coordinates denoting the geometry of the node e.g.**

<node id="(137623.20999999999, 28977.740000000002)" label="(137623.20999999999, 28977.740000000002)">

**NOTE: edge id and label values set to equal a tuple of coordinates denoting the geometry of the source and target node of an edge e.g.**

<edge id="20" source="(247865.09, 666985.34999999998)" target="(512423.73999999999, 220830.37)">

### Pajek

nx\_pgnet.export\_graph(db\_connection).export\_to\_pajek(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_pajek\_to>’, ‘<insert\_name\_of\_output\_pajek\_file>’, ‘<insert\_node\_attribute\_name\_for\_label>’,<insert\_edge\_attribute\_name\_for\_edge\_weight>,’<insert\_encoding>’)

nx\_pgnet.export\_graph(db\_connection).export\_to\_pajek(graph, ‘C:/TEMP/’,’TransportNetwork’,’name’,’value’,’utf-8’)

**NOTE: Pajek requires nodes to be defined in the format defined below. As such, the export using export\_to\_pajek, results in nodes having their label value set to be equal to a tuple representing the coordinates of the node. Furthermore, the x and y values are also set to the appropriate x and y components of the coordinate.**

<vertex\_number> <label> x y …

For example, this could look like:

\*network NetworkX

\*vertices 48

1 "(137623.20999999999, 28977.740000000002)" 137623.21 28977.74 ellipse y 28977.74 x 137623.21 name "Land's End / St. Just Airport"

2 "(247865.09, 666985.34999999998)" 247865.09 666985.35 ellipse y 666985.35 x 247865.09 name Glasgow

**NOTE: Currently edge coordinates are not output to the Pajek file format. Start and end points of edges within the Pajek format are effectively integer references to the start and end nodes that would make up that edge. For example, an edge between “Land’s End / St. Just Airport” and “Glasgow” (see above for node definition in Pajek format), would look like the following (with the final 1.0 value denoting some edge weighting)**

\*edges

1 2 1.0

**NOTE: Currently the exporting of a network to the Pajek .net format results in all edge attributes being removed. A user is able to specify a particular edge attribute to use as a weight for an edge in the output Pajek format, or set a default value for this weight, that would then be applied to all edges in the output Pajek file (see above example)**

### YAML

nx\_pgnet.export\_graph(db\_connection).export\_to\_yaml(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_yaml\_to>’,’<insert\_name\_of\_output\_yaml\_file>’,’<insert\_encoding>’)

nx\_pgnet.export\_graph(db\_connection).export\_to\_yaml(graph, ‘C:/TEMP/’,’TransportNetwork’,’utf-8’)

### GraphML

nx\_pgnet.export\_graph(db\_connection).export\_to\_graphml(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_graphml\_to>’, ’<insert\_name\_of\_output\_graphml\_file>’,’<insert\_encoding>’, <insert\_prettyprint\_output\_status>)

nx\_pgnet.export\_graph(db\_connection).export\_to\_graphml(graph, ‘C:/TEMP/’, ‘TransportNetwork’, ‘utf-8’, True)

**NOTE: node id values are set to equal a tuple of coordinates denoting the geometry of the node e.g.**

<node id="(137623.20999999999, 28977.740000000002)">

**NOTE: edge id and label values set to equal a tuple of coordinates denoting the geometry of the source and target node of an edge e.g.**

<edge source="(128198.08, 537323.33999999997)" target="(314388.65999999997, 673846.98999999999)">

### GML

nx\_pgnet.export\_graph(db\_connection).export\_to\_graphml(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_graphml\_to>’, ’<insert\_name\_of\_output\_graphml\_file>’)

nx\_pgnet.export\_graph(db\_connection).export\_to\_gml(graph, ‘C:/TEMP/’, ‘TransportNetwork’)

**NOTE: JSON and WKT representations of node coordinates kept in GML output, to denote geometry of node e.g.**

node [

id 0

label 1

city "Land's End"

name "Land's End / St. Just Airport"

country "United Kingdom"

altitude 401.0

iata\_faa\_c "LEQ"

longitude -5.67056

airportid 5572

Json "{ "type": "Point", "coordinates": [ 137623.21, 28977.74 ] }"

icao\_code "EGHC"

latitude 50.1028

timezone 0.0

dst "E"

Wkt "POINT (137623.21 28977.74)"

]

**NOTE: JSON and WKT representations of edge coordinates kept in GML output, to denote geometry of edge e.g.**

edge [

source 8

target 24

source\_airport\_id 521

equipment "DH4 "

destination\_airport "SOU "

stops 0.0

airlineid 2421

codeshare None

Json "{ "type": "LineString", "coordinates": [ [ 419803.84, 571486.05 ], [ 445278.65, 116955.37 ] ] }"

airline "BE "

destination\_airport\_id 495

source\_airport "NCL "

Wkt "LINESTRING (419803.84 571486.05,445278.65 116955.37)"

]

### Gephi-compatible node/edge lists

nx\_pgnet.export\_graph(db\_connection).export\_to\_gephi\_node\_edge\_lists(‘<insert\_path\_to\_write\_to>’,’<insert\_node\_viewname>’,’<insert\_edge\_viewname>’,’<insert\_node\_view\_geometry\_column\_name>’,’<insert\_edge\_view\_geometry\_column\_name>’,<insert\_directed\_status>)

nx\_pgnet.export\_graph(db\_connection).export\_to\_gephi\_node\_edge\_lists(‘C:/TEMP/’, ‘TransportNetwork\_View\_Nodes’, ‘TransportNetwork\_View\_Edges\_Edge\_Geometry’,’geom’,’geom’,False)

The geometry of the nodes and edges within a network are output WKT representations within the Gephi-compatible node and edge lists, along with any other parameters defined e.g.

**Node Example (not exhaustive list of attributes)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| view\_id | GraphID | geometry\_text | srid | google\_node\_x | google\_node\_y | wgs84\_node\_x | wgs84\_node\_y |
| 1 | 957 | POINT(166906.72 622106.05) | 27700 | -633006 | 7447184 | -5.68639 | 55.4372 |

The edge table would contain the following values, alongside any other attributes defined:

* Google\_StartPoint\_X – X Component of coordinates in EPSG 900913 of start point of edge
* Google\_StartPoint\_Y– Y Component of coordinates in EPSG 900913 of start point of edge
* Google\_EndPoint\_X– X Component of coordinates in EPSG 900913 of end point of edge
* Google\_EndPoint\_Y– Y Component of coordinates in EPSG 900913 of end point of edge
* WGS84\_StartPoint\_X– X Component of coordinates in EPSG 4326 of start point of edge
* WGS84\_StartPoint\_Y– Y Component of coordinates in EPSG 4326 of start point of edge
* WGS84\_EndPoint\_X– X Component of coordinates in EPSG 4326 of end point of edge
* WGS84\_EndPoint\_Y– Y Component of coordinates in EPSG 4326 of end point of edge

# Aspatial Networks

## Example exports – create a database connection

Import nx\_pgnet as nx\_pgnet

Import osgeo.ogr as ogr

db\_connection = ogr.Open("PG: host=’<insert\_server\_host>’ dbname=’<insert\_database\_name>’ user=’<insert\_user\_name>’ password=’<insert\_password>’”)

db\_connection = ogr.Open("PG: host='127.0.0.1' dbname='database' user='postgres', password='password'")

### GEXF

nx\_pgnet.export\_graph(<insert\_database\_connection>).export\_to\_gexf(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_gexf\_to>’, ‘<insert\_name\_of\_output\_gexf\_file>’, ‘<insert\_encoding>’, <insert\_prettyprint\_output\_status>)

nx\_pgnet.export\_graph(db\_connection).export\_to\_gexf(a\_network, ‘C://Temp//’, ‘a\_network’, ‘utf-8’, True)

### Pajek

nx\_pgnet.export\_graph(db\_connection).export\_to\_pajek(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_pajek\_to>’, ‘<insert\_name\_of\_output\_pajek\_file>’, ‘<insert\_node\_attribute\_name\_for\_label>’,<insert\_edge\_attribute\_name\_for\_edge\_weight>,’<insert\_encoding>’)

nx\_pgnet.export\_graph(db\_connection).export\_to\_pajek(a\_network, ‘C://Temp//’, ‘a\_network’, ‘node\_attribute\_name’, ‘edge\_attribute\_name’, False)

### YAML

nx\_pgnet.export\_graph(db\_connection).export\_to\_yaml(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_yaml\_to>’,’<insert\_name\_of\_output\_yaml\_file>’,’<insert\_encoding>’)

nx\_pgnet.export\_graph(db\_connection).export\_to\_yaml(a\_network, ‘C://Temp//’, ‘a\_network’, ‘utf-8’)

### GraphML

nx\_pgnet.export\_graph(db\_connection).export\_to\_graphml(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_graphml\_to>’, ’<insert\_name\_of\_output\_graphml\_file>’,’<insert\_encoding>’, <insert\_prettyprint\_output\_status>)

nx\_pgnet.export\_graph(db\_connection).export\_to\_graphml(a\_network, ‘C://Temp//’, ‘a\_network’, ‘utf-8’, True)

### GML

nx\_pgnet.export\_graph(db\_connection).export\_to\_graphml(<insert\_networkx\_graph>, ‘<insert\_path\_to\_write\_graphml\_to>’, ’<insert\_name\_of\_output\_graphml\_file>’)

nx\_pgnet.export\_graph(conn).export\_to\_gml(a\_network, 'C://Temp//', 'a\_network')

### Gephi-compatible node/edge lists

nx\_pgnet.export\_graph(db\_connection).export\_to\_gephi\_node\_edge\_lists(‘<insert\_path\_to\_write\_to>’,’<insert\_node\_viewname>’,’<insert\_edge\_viewname>’,’<insert\_node\_view\_geometry\_column\_name>’,’<insert\_edge\_view\_geometry\_column\_name>’,<insert\_directed\_status>)

nx\_pgnet.export\_graph(conn).export\_to\_gephi\_node\_edge\_lists('C://Temp//', 'a\_node\_view\_name', 'an\_edge\_view\_name', spatial=False)