Table of contents

NAME

v.net - Performs network maintenance.

KEYWORDS

vector, network, network maintenance

SYNOPSIS

v.net
v.net --help
v.net [-**cs**] [**input**=*name*] [**points**=*name*] [**output**=*name*] **operation**=*string* [**arc_layer**=*string*] [**arc_type**=*string*[,*string*,...]] [**node_layer**=*string*] [**threshold**=*float*] [**file**=*name*] [**turn_layer**=*string*] [**turn_cat_layer**=*string*] [--**overwrite**] [--**help**] [--**verbose**] [--**quiet**] [--**ui**]

Flags:

-c
Assign unique categories to new points
For operation 'nodes'
-s
Snap points to network
For operation 'connect'. By default, a new line from the point to the network is created.
--overwrite
Allow output files to overwrite existing files
--help
Print usage summary
--verbose
Verbose module output
--quiet
Quiet module output
--ui
Force launching GUI dialog

Parameters:

input=*name*
Name of input vector line map (arcs)
Required for operation 'nodes', 'connect', 'report' and 'nreport'
points=*name*
Name of input vector point map (nodes)
Required for operation 'connect' and 'arcs'
output=*name*
Name for output vector map
operation=*string* **[required]**

Table of contents

Operation to be performed
Options: *nodes, connect, arcs, report, nreport, turntable*
nodes: new point is placed on each node (line end) if doesn't exist
connect: connect still unconnected points to vector network by inserting new line(s)
arcs: new line is created from start point to end point
report: print to standard output {line_category start_point_category end_point_category}
nreport: print to standard output {point_category line_category[,line_category...]}
turntable: create turntable on vector network

arc_layer=*string*
Arc layer
Vector features can have category values in different layers. This number determines which layer to use. When used with direct OGR access this is the layer name.
Default: *1*

arc_type=*string[,string,...]*
Arc type
Input feature type
Options: *line, boundary*
Default: *line,boundary*

node_layer=*string*
Node layer
Vector features can have category values in different layers. This number determines which layer to use. When used with direct OGR access this is the layer name.
Default: *2*

threshold=*float*
Threshold
Required for operation 'connect'. Connect points in given threshold.

file=*name*
Name of input file
Required for operation 'arcs' ('-' for standard input)

turn_layer=*string*
Turntable layer
Layer where turntable will be attached. Format: layer number[/layer name].Required for operation 'turntable'.
Default: *3*

turn_cat_layer=*string*
Layer with unique categories used in turntable
Layer with unique categories for every line in arc_layer and point on every node. The categories are used in turntable. Format: layer number[/layer name]. Required for operation 'turntable'.
Default: *4*

DESCRIPTION

v.net is used for network preparation and maintenance. Its main use is to create a vector network from vector lines (*arcs*) and points (*nodes*) by creating nodes from intersections in a map of vector lines (*node* operator), by connecting a vector lines map with a points map (*connect* operator), and by creating new lines between pairs of vector points (*arcs* operator).

A GIS network consists of topologically correct lines (arcs). That is, the lines must be connected by shared vertices where real connections exist. In GRASS GIS you also can add nodes to the network. These are

Table of contents

specially designated vertices used for analyzing network properties or computing cost/distance measures. That is, **not all vertices are treated as nodes by default**. Only *v.net.path* can use a network without nodes, they are required for all the other network modules. In GRASS, network arcs are stored in one data layer (normally layer 1) and nodes are stored in a different data layer (normally layer 2).

v.net offers two ways to add nodes to a network of arcs and one method to add arcs to a set of nodes:

1. Use the *connect* operation to create nodes from a vector points file and add these nodes to an existing vector network of arcs (i.e., lines/boundaries). This is useful when the goal is to analyze a set of places (points) in relation to a network--for example travel costs between places. Only points within the *thresh* (threshold) distance to a line/boundary will be connected as network nodes. There are two ways to connect nodes. By default, *v.net* will create new lines connecting each point to the closest line of the network. If you use the *-s* flag, however, the new nodes will be added on the closest line of the network at the point closest to the point you wish to add. When using the *connect* operation, some lines will share the same category. In order to assign unique costs to each line, a new layer needs to be created with
   ```
   v.category map=yourmap option=add cat=1 step=1 layer=3
   output=newmap
   ```
   followed by
   ```
   v.db.addtable map=newmap layer=3 table=tablename.
   ```
2. Create nodes and arcs from a vector line/boundary file using the *node* operation. This is useful if you are mostly interested in the network itself and thus you can use intersections of the network as start and end points. Nodes will be created at all intersections of two or more lines. For an *arc* that consists of several segments connected by vertices (the typical case), only the starting and ending vertices are treated as network nodes.
3. Create straight-line arcs between pairs of nodes with the *arcs* option. This produces networks like those of airline flights between airports. It is also similar to the kind of network created with social networking software, making it possible to create georeferenced social networks.

While the arcs created with v.net will retain any attribute information associated with the input vector line/boundary file in data layer 1, nodes created and stored in data layer 2 will not have any associated attribute information.

For nodes created using the *connect* and *arcs* operations (methods 1 and 3 above), the nodes can be reconnected to the attribute table of the input vector points file using the attribute table manager ("manage layers" tab) or by running *v.db.connect*.

For nodes created using the *nodes* operation (method 2 above), it is possible to create an attribute table for the new nodes in layer 2 using the attribute table manager and connect it to layer 2 ("manage layers" tab) or to create a table with *v.db.addtable*, connect it to layer 2 with *v.db.connect*, and update the new table with cat values with *v.to.db*.

The *turntable* operation creates a turntable with the costs for every possible turn on every possible node (intersection, crossroad) in given

Table of contents

layer (arc_layer). U-turns are taken in account too. Turntable is created in **turn_layer** and **turn_cat_layer**. Building the turntable allows you to model e.g. traffic code, where some turns may be prohibited. If features in analyzed network are changed, the turntable must be created again (e.g. it includes v.net connect operation). Turntable name consists of output vector map name + "_turntable_" + "t" + "_" + turn_layer + "_" + "tuc" + "_" + turn_cat_layer + "_" + "a" + "_" + arc_layer e. g. roads_turntable_t_3_tuc_4_a_1

These modules are able to work with the turntable: *v.net.alloc*, *v.net.iso*, *v.net.path*, *v.net.salesman* For more information about turns in the vector network analyses see wiki page.

Once a vector network has been created, it can be analyzed in a number of powerful ways using the suite of *v.net.** modules. The shortest route between two nodes, following arcs, can be computed (*v.net.path*), as can the shortest route that will pass through a set of nodes and return to the starting node (*v.net.salesman*). Least cost routes through the network can be calculated on the basis of distance only or on the basis of distance weighted by an attribute associated with each arc (for example, travel speed along a network segment). A network can be divided into concentric zones of equal travel cost around one or more nodes (*v.net.iso*) or subdivided so that each node is surrounded by a zone in which all arcs can be reached with the same travel costs as all arcs surrounding each other node (*v.net.alloc*). In addition to the modules listed above, the GRASS vector networking suite includes numerous other modules for analysis of network costs and connectivity. These include: *v.net.allpairs*, *v.net.bridge*, *v.net.centrality*, *v.net.components*, *v.net.distance*, *v.net.flow*, *v.net.spanningtree*, *v.net.steiner*, *v.net.timetable*, and *v.net.visibility*.

NOTES

For a vector map prepared for network analysis in GRASS, nodes are represented by the grass-internal geometry type *node* and arcs by the geometry type *line*. If vector editing is required to modify the graph, *g.gui.vdigit* or *v.edit* can be used. See also the Linear Referencing System available in GRASS GIS.

EXAMPLES

The examples are North Carolina dataset based.

Create nodes globally for all line ends and intersections

```
v.net input=streets_wake output=streets_node operation=nodes
# verify result
v.category streets_node option=report
```

Merge in nodes from a separate map within given threshold

```
v.net input=streets_wake points=firestations out=streets_net \
      operation=connect threshold=500
# verify result
v.category streets_net option=report
```

The nodes are stored in layer 2 unless `node_layer=1` is used.

Table of contents

Generating network for vector point map

For generating network for given vector point map an input file in the following format is required:

```
[category of edge] [category of start node] [category of end node]
```

Option 1: Save the file (e.g. "points.txt") and generate the map:

```
v.net points=geodetic_swwake_pts output=geodetic_swwake_pts_net \
      operation=arcs file=points.txt
# verify result
v.category geodetic_swwake_pts_net option=report
```

Option 2: Read in from command line:

```
v.net points=geodetic_swwake_pts output=geodetic_swwake_pts_net \
      operation=arcs file=- << EOF
1 28000 28005
2 27945 27958
3 27886 27897
EOF

# verify result
v.category geodetic_swwake_pts_net option=report
```

Generating network with turntable for vector point map

Following example generates a vector map with turntable:

```
v.net operation=turntable in=railroads out=railroads_ttb
```

SEE ALSO

g.gui.vdigit, *v.edit*

v.net.alloc, *v.net.allpairs*, *v.net.bridge*, *v.net.centrality*, *v.net.components*, *v.net.connectivity*, *v.net.distance*, *v.net.flow*, *v.net.iso*, *v.net.path*, *v.net.salesman* *v.net.spanningtree*, *v.net.steiner*, *v.net.timetable*, *v.net.visibility*

AUTHORS

Radim Blazek, ITC-irst, Trento, Italy
Martin Landa, FBK-irst (formerly ITC-irst), Trento, Italy and CTU in Prague, Czech Republic (operation 'connect' and 'arcs')
Markus Metz: important fixes and improvements

TURNS SUPPORT

The turns support was implemnented as part of GRASS GIS turns cost project at Czech Technical University in Prague, Czech Republic. Eliska Kyzlikova, Stepan Turek, Lukas Bocan and Viera Bejdova participated at the project. Implementation: Stepan Turek Documentation: Lukas Bocan Mentor: Martin Landa

Last changed: $Date: 2016-11-13 15:05:32 -0800 (Sun, 13 Nov 2016) $

SOURCE CODE

Available at: v.net source code (history)

Table of contents

Main index | Vector index | Topics index | Keywords index | Graphical index | Full index