nearneighbor

nearneighbor - Grid table data using a “Nearest neighbor” algorithm

Synopsis

nearneighbor [*table*] **-G***out_grdfile* **-I***increment* **-N***sectors*[/*min_sectors*] **-R***region* **-S***search_radius*[*unit*] [**-E***empty*] [**-V**[*level*]] [**-W**] [**-bi**binary] [**-di**nodata] [**-e**regexp] [**-f**flags] [**-h**headers] [**-i**flags] [**-n**flags] [**-r**] [**-:**[**i**|**o**]]

Note: No space is allowed between the option flag and the associated arguments.

Description

nearneighbor reads arbitrarily located (x,y,z[,w]) triples [quadruplets] from standard input [or *table*] and uses a nearest neighbor algorithm to assign an average value to each node that have one or more points within a radius centered on the node. The average value is computed as a weighted mean of the nearest point from each sector inside the search radius. The weighting function used is w(r) = 1 / (1 + d ^ 2), where d = 3 * r / search_radius and r is distance from the node. This weight is modulated by the weights of the observation points [if supplied].

Required Arguments

-G*out_grdfile*

Give the name of the output grid file.

-I*xinc*[*unit*][**+e**|**n**][/*yinc*[*unit*][**+e**|**n**]]

x_inc [and optionally *y_inc*] is the grid spacing. Optionally, append a suffix modifier. **Geographical (degrees) coordinates**: Append **m** to indicate arc minutes or **s** to indicate arc seconds. If one of the units **e**, **f**, **k**, **M**, **n** or **u** is appended instead, the increment is assumed to be given in meter, foot, km, Mile, nautical mile or US survey foot, respectively, and will be converted to the equivalent degrees longitude at the middle latitude of the region (the conversion depends on PROJ_ELLIPSOID). If *y_inc* is given but set to 0 it will be reset equal to *x_inc*; otherwise it will be converted to degrees latitude. **All coordinates**: If **+e** is appended then the corresponding max *x* (*east*) or *y* (*north*) may be slightly adjusted to fit exactly the given increment [by default the increment may be adjusted slightly to fit the given domain]. Finally, instead of giving an increment you may specify the *number of nodes* desired by appending **+n** to the supplied integer argument;

the increment is then recalculated from the number of nodes and the domain. The resulting increment value depends on whether you have selected a gridline-registered or pixel-registered grid; see GMT File Formats for details. Note: if **-R***grdfile* is used then the grid spacing has already been initialized; use **-I** to override the values.

-N*sectors*[/*min_sectors*]

The circular area centered on each node is divided into *sectors* sectors. Average values will only be computed if there is at least one value inside each of at least *min_sectors* of the sectors for a given node. Nodes that fail this test are assigned the value NaN (but see **-E**). If *min_sectors* is omitted it is set to be at least 50% of *sectors* (i.e., rounded up to next integer). [Default is a quadrant search with 100% coverage, i.e., *sectors* = *min_sectors* = 4]. Note that only the nearest value per sector enters into the averaging; the more distant points are ignored.

-R*xmin*/*xmax*/*ymin*/*ymax*[**+r**][**+u***unit*] (more …)

Specify the region of interest.

-S*search_radius*[*unit*]

Sets the *search_radius* that determines which data points are considered close to a node. Append the distance unit (see UNITS).

Optional Arguments

table

3 [or 4, see **-W**] column ASCII file(s) [or binary, see **-bi**] holding (x,y,z[,w]) data values. If no file is specified, **nearneighbor** will read from standard input.

-E*empty*

Set the value assigned to empty nodes [NaN].

-V[*level*] (more …)

Select verbosity level [c].

-W

Input data have a 4th column containing observation point weights. These are multiplied with the geometrical weight factor to determine the actual weights used in the calculations.

-bi[*ncols*][**t**] (more …)

Select native binary input. [Default is 3 (or 4 if **-W** is set) columns].

-di*nodata* (more …)

Replace input columns that equal *nodata* with NaN.

-e[**~**]*"pattern"* **|** **-e**[**~**]/*regexp*/[**i**] (more …)

Only accept data records that match the given pattern.

-f[**i**|**o**]*colinfo* (more …)

Specify data types of input and/or output columns.

-h[**i**|**o**][*n*][**+c**][**+d**][**+r***remark*][**+r***title*] (more …)

Skip or produce header record(s).

-i*cols*[**+l**][**+s***scale*][**+o***offset*][,*…*] (more …)

Select input columns and transformations (0 is first column).

-n[**b**|**c**|**l**|**n**][**+a**][**+b***BC*][**+t***threshold*]

Append **+b***BC* to set any boundary conditions to be used, adding **g** for geographic, **p** for periodic, or **n** for natural boundary conditions. For the latter two you may append **x** or **y** to specify just one direction, otherwise both are assumed. [Default is geographic if grid is geographic].

-r (more …)

Set pixel node registration [gridline].

-:[**i**|**o**] (more …)

Swap 1st and 2nd column on input and/or output.

-^ or just **-**

Print a short message about the syntax of the command, then exits (NOTE: on Windows just use **-**).

-+ or just **+**

Print an extensive usage (help) message, including the explanation of any module-specific option (but not the GMT common options), then exits.

-? or no arguments

Print a complete usage (help) message, including the explanation of all options, then exits.

Units

For map distance unit, append *unit* **d** for arc degree, **m** for arc minute, and **s** for arc second, or **e** for meter [Default], **f** for foot, **k** for km, **M** for statute mile, **n** for nautical mile, and **u** for US survey foot. By default we compute such distances using a spherical approximation with great circles. Prepend **-** to a distance (or the unit is no distance is given) to perform “Flat Earth” calculations (quicker but less accurate) or prepend **+** to perform exact geodesic calculations (slower but more accurate).

Grid Values Precision

Regardless of the precision of the input data, GMT programs that create grid files will internally hold the grids in 4-byte floating point arrays. This is done to conserve memory and furthermore most if not all real data can be stored using 4-byte floating point values. Data with higher precision (i.e., double precision values) will lose that precision once GMT operates on the grid or writes out new grids. To limit loss of precision when processing data you should always consider normalizing the data prior to processing.

Examples

To create a gridded data set from the file seaMARCII_bathy.lon_lat_z using a 0.5 min grid, a 5 km search radius, using an octant search with 100% sector coverage, and set empty nodes to -9999:

```
gmt nearneighbor seaMARCII_bathy.lon_lat_z -R242/244/-22/-2
                 -E-9999 -Gbathymetry.nc -S5k -N8/8
```

To make a global grid file from the data in geoid.xyz using a 1 degree grid, a 200 km search radius, spherical distances, using an quadrant search, and set nodes to NaN only when fewer than two quadrants contain at least one value:

```
gmt nearneighbor geoid.xyz -R0/360/-90/90 -I1 -Lg -Ggeoid.n
```

See Also

blockmean, blockmedian, blockmode, gmt, greenspline, sphtriangulate, surface, triangulate