xyz2grd

xyz2grd - Convert data table to a grid file

Synopsis

xyz2grd [*table*] **-G***grdfile* **-I***increment* **-R***region* [**-A**[**d**|**f**|**l**|**m**|**n**|**r**|**S**|**s**|**u**|**z**]] [**-D**[**+x***xname*][**+y***y-name*][**+z***zname*][**+s***scale*][**+o***offset*][**+n***invalid*][**+t***title*][**+r***remark*]] [**-J***parameters*] [**-S**[*zfile*]] [**-V**[*level*]] [**-Z**[*flags*]] [**-bi**binary] [**-di**nodata] [**-e**regexp] [**-f**flags] [**-h**headers] [**-i**flags] [**-r**] [**-:**[**i**|**o**]]

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

Description

xyz2grd reads one or more z or xyz tables and creates a binary grid file. **xyz2grd** will report if some of the nodes are not filled in with data. Such unconstrained nodes are set to a value specified by the user [Default is NaN]. Nodes with more than one value will be set to the mean value. As an option (using **-Z**), a 1-column z-table may be read assuming all nodes are present (z-tables can be in organized in a number of formats, see **-Z** below.) Note: **xyz2grd** does not grid the data, it simply reformats existing data to a grid structure. For gridding, see **surface**, **greenspline**, **nearneighbor**, or **triangulate**.

Required Arguments

-G*grdfile*

grdfile is the name of the binary output grid file. (See GRID FILE FORMAT below.)

-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.

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

Specify the region of interest.

Optional Arguments

table

One or more ASCII [or binary, see **-bi**] files holding z or (x,y,z) values. The xyz triplets do not have to be sorted. One-column z tables must be sorted and the **-Z** must be set.

-A[**d**|**f**|**l**|**m**|**n**|**r**|**S**|**s**|**u**|**z**]

By default we will calculate mean values if multiple entries fall on the same node. Use **-A** to change this behavior, except it is ignored if **-Z** is given. Append **f** or **s** to simply keep the first or last data point that was assigned to each node. Append **l** or **u** or **d** to find the lowest (minimum) or upper (maximum) value or the difference between the maximum and miminum value at each node, respectively. Append **m** or **r** or **S** to compute mean or RMS value or standard deviation at each node, respectively. Append **n** to simply count the number of data points that were assigned to each node (this only requires two input columns *x* and *y* as *z* is not consulted). Append **z** to sum multiple values that belong to the same node.

-D[**+x***xname*][**+y***yname*][**+z***zname*][**+s***scale*][**+o***offset*][**+n***invalid*][**+t***title*][**+r***remark*]

Give one or more combinations for values *xname*, *yname*, *zname* (give the names of those variables and in square bracket their units, e.g., “distance [km]”), *scale* (to multiply grid values after read [normally 1]), *offset* (to add to grid after scaling [normally 0]), *invalid* (a value to represent missing data [NaN]), *title* (anything you like), and *remark* (anything you like). Items not listed will remain untouched. Give a blank name to completely reset a particular string. Use quotes to group texts with more than one word. Note that for geographic grids (**-fg**) *xname* and *yname* are set automatically.

-J*parameters* (more …)

Select map projection. Use the **-J** syntax to save the georeferencing info as CF-1 compliant metadata in netCDF grids. This metadata will be recognized by GDAL.

-S[*zfile*]

Swap the byte-order of the input only. No grid file is produced. You must also supply the **-Z** option. The output is written to *zfile* (or stdout if not supplied).

-V[*level*] (more …)

Select verbosity level [c].

-Z[*flags*]

Read a 1-column ASCII [or binary] table. This assumes that all the nodes are present and sorted according to specified ordering convention contained in *flags*. If incoming data represents rows, make *flags* start with **T**(op) if first row is y = ymax or **B**(ottom) if first row is y = ymin. Then, append **L** or **R** to indicate that first element is at left or right end of row. Likewise for column formats: start with **L** or **R** to position first column, and then append **T** or **B** to position first element in a row. Note: These two row/column indicators are only required for grids; for other tables they do not apply. For gridline registered grids: If data are periodic in x but the incoming data do not contain the (redundant) column at x = xmax, append **x**. For data periodic in y without redundant row at y = ymax, append **y**. Append **s***n* to skip the first *n* number of bytes (probably a header). If the byte-order or the words needs to be swapped, append **w**. Select one of several data types (all binary except **a**):

A ASCII representation of one or more floating point values per record

a ASCII representation of a single item per record

c int8_t, signed 1-byte character

u uint8_t, unsigned 1-byte character

h int16_t, signed 2-byte integer

H uint16_t, unsigned 2-byte integer

i int32_t, signed 4-byte integer

I uint32_t, unsigned 4-byte integer

l int64_t, long (8-byte) integer

L uint64_t, unsigned long (8-byte) integer

f 4-byte floating point single precision

d 8-byte floating point double precision

Default format is scanline orientation of ASCII numbers: **-ZTLa**. Note that **-Z** only applies to 1-column input. The difference between **A** and **a** is that the latter can decode both *date***T***clock* and *ddd:mm:ss[.xx]* formats while the former is strictly for regular floating point values.

-bi[*ncols*][**t**] (more …)

Select native binary input. [Default is 3 input columns]. This option only applies to xyz input files; see **-Z** for z tables.

-di*nodata* (more …)

Replace input columns that equal *nodata* with NaN. Also sets nodes with no input xyz triplet to this value [Default is 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). Not used with binary data.

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

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

-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.

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.

Grid File Formats

By default GMT writes out grid as single precision floats in a COARDS-complaint netCDF file format. However, GMT is able to produce grid files in many other commonly used grid file formats and also facilitates so called “packing” of grids, writing out floating point data as 1- or 2-byte integers. To specify the precision, scale and offset, the user should add the suffix =*ID*[**+s***scale*][**+o***offset*][**+n***invalid*], where *ID* is a two-letter identifier of the grid type and precision, and *scale* and *offset* are optional scale factor and offset to be applied to all grid values, and *invalid* is the value used to indicate missing data. See grdconvert and Section Grid file format specifications of the GMT Technical Reference and Cookbook for more information.

When writing a netCDF file, the grid is stored by default with the variable name “z”. To specify another variable name *varname*, append **?***varname* to the file name. Note that you may need to escape the special meaning of **?** in your shell program by putting a backslash in front of it, or by placing the filename and suffix between quotes or double quotes.

Geographical And Time Coordinates

When the output grid type is netCDF, the coordinates will be labeled “longitude”, “latitude”, or “time” based on the attributes of the input data or grid (if any) or on the **-f** or **-R** options. For example, both **-f0x -f1t** and **-R**90w/90e/0t/3t will result in a longitude/time grid. When the x, y, or z coordinate is time, it will be stored in the grid as relative time since epoch as specified by TIME_UNIT and TIME_EPOCH in the gmt.conf file or on the command line. In addition, the **unit** attribute of the time variable will indicate both this unit and epoch.

Swapping Limitations

All data types can be read, even 64-bit integers, but internally grids are stored using floats. Hence, integer values exceeding the float type’s 23-bit mantissa may not be represented exactly. When **-S** is used no grids are implied and we read data into an intermediate double container. This means all but 64-bit integers can be represented using the double type’s 53-bit mantissa.

Examples

To create a grid file from the ASCII data in hawaii_grv.xyz, use

```
gmt xyz2grd hawaii_grv.xyz -D+xdegree+ydegree+zGal+t”Hawaiian
Gravity”+r”GRS-80 Ellipsoid used”
     -Ghawaii_grv_new.nc -R198/208/18/25 -I5m -V
```

To create a grid file from the raw binary (3-column, single-precision scanline-oriented data raw.b, use

```
gmt xyz2grd raw.b -D+xm+ym+zm -Graw.nc -R0/100/0/100 -I1 -V -Z -bi3f
```

To make a grid file from the raw binary USGS DEM (short integer scanline-oriented data topo30.b on the NGDC global relief Data CD-ROM, with values of -9999 indicate missing data, one must on some machine reverse the byte-order. On such machines (like Sun), use

```
gmt xyz2grd topo30.b -D+xm+ym+zm -Gustopo.nc -R234/294/24/50 -I30s
-di-9999 -ZTLhw
```

Say you have received a binary file with 4-byte floating points that were written on a machine of different byte-order than yours. You can swap the byte-order with

```
gmt xyz2grd floats.bin -Snew_floats.bin -V -Zf
```

See Also

gmt, grd2xyz, grdedit, grdconvert, greenspline, nearneighbor, surface, triangulate