NAME

s.buffer − generates vector buffer around a site point. The area is labelled with site category.
(GRASS Sites Program)

SYNOPSIS

s.buffer
s.buffer help
s.buffer input=*name* **output=***name* **radius=***value*

DESCRIPTION

s.buffer generates vector buffer around a site point. The area is labelled with site category. If the value is not given or its 0, then the radius is taken from the attribute of the site point.

PARAMETERS

input site_list file to read

output vector (digit) file to create

radius radius of circle

Values for radius:

0=take category for radius,
−1=the 4th value of site list (category label),
positive value= use given value for radius. Then the buffer is the same with all points.

SEE ALSO

v.circle, v.bubble

AUTHOR

Janne Soimasuo, Finland

Last changed: $Date: 2003/07/04 08:35:34 $

NAME

s.cellstats − Univariate statistics for a GRASS sites list within cells.
(GRASS Sites Program)

SYNOPSIS

s.cellstats
s.cellstats help
s.cellstats [**−q**] **input=***name* **[output=***name***] [field=***value***] [min=***value***] [stats=***stat[,stat,...]***]**

DESCRIPTION

s.cellstats calculates univariate statistics of sites lists for each cluster of input sites falling within each cell of the current region. This includes the number of sites, mean, standard deviation, coefficient of variation, mininum, first quartile, median, third quartile, and maximum. The output is a sites list where the coordinates are cell centers, the category is the count of sites falling within the cell, and the decimal attributes are the statistics. Sites falling outside the current region are discarded.

OPTIONS

Flags:

−q
Quiet. Turn off informational messages.

Parameters:

input=*name*
Name of an existing sites file.
output=*name*
Name of the output sites file. If not specified, the program writes to standard output.
field=*value*
Decimal attribute to use for calculation
default: 1
min=*value*
Set a minimum sample size per cell. If there are fewer than *value* sites found for a given cell, statistics are not calculated for that cell.
default: 1
stats=*stat[,stat,...]*
Specify which statistics should be reported. Statistics will be reported in the order specified, otherwise they are output in the order below. The statistics are:
m − mean

s − standard deviation
cv − coefficient of variation
skw − skewness
kur − kurtosis
mse − mean of squares
mav − mean of absolute values
min − minimum
q1 − first quartile
med − median
q3 − third quartile
max − maximum
sum − sum
n − number of sites in the sample

SEE ALSO

s.univar, *s.windavg*, *s.normal*, *r.univar*

NOTES

Requires at least 16 bytes of memory per input site plus some other overhead. Very large input sites files may cause excessive swap file usage or cause the program to fail with an out of memory error.

Certain statistics are invalid when calculated for a single site. These statistics are the standard deviation, coefficient of variation, skewness and kurtosis. The *min* parameter may be used to insure the sample size is large enough to approximate normality. Additionally, the cell size for the current region may be enlarged to capture more sites per cell. No test for normality is performed on the input and some statistics may not be valid if normality is violated.

BUGS

None known.

Please send all bug fixes and comments to the grass development team.
`http://grass.itc.it`.

AUTHOR

Eric G. Miller, adapted from s.univar written by James Darrell McCauley <darrell@mccauley−usa.com>, when he was at Agricultural Engineering, Purdue University

Last changed: $Date: 2003/08/20 13:29:14 $

NAME

s.datum.shift − Allows the user to apply a datum transformation on a sites file
(GRASS Sites Program)

SYNOPSIS

s.datum.shift
s.datum.shift help
s.datum.shift input=*name* **output=***name***[idatum=***name*] **odatum=***name* **[−m]**

DESCRIPTION

Allows the user to perform a datum transformation on a sites file. The map datum of the output file is read from the current location if not specified on the command line. Each site will retain all associated category, double, and string attributes. The header information is copied and an additional header comment is written containing the *s.datum.shift* command line used to create the new file.

Parameters:

input
: Name of the input sites list.

output
: Name of the output sites list.

idatum
: Map Datum of the input sites list.

odatum
: Map Datum of the output sites list.

−m
: use Molodensky formula, faster but less precise.

SEE ALSO

s.proj, *m.datum.shift*

AUTHOR

Andreas Lange, andreas.lange@rhein−main.de

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.db.rim − RIM data base management/query interface for GRASS sites data.
(GRASS Sites Management Program)

SYNOPSIS

s.db.rim
s.db.rim *data_base*

DESCRIPTION

For some informations on LINUX and RIM look at the BUGS section

s.db.rim allows users to create, manage and query information about site locations (sites) across the landscape. Required inputs can be entered interactively, or from the command line. Command line input may be entered through a prepared text file or from the keyboard (standard input). The *s.db.rim* command language is defined in SECTION ONE below. The menu−driven interactive version is described in SECTION TWO of these manual pages.

These programs are actually a marriage of the GRASS environment and the programmer's interface library of the relational data base management program RIM, distributed publically by the University of Washington Academic Computing Services as FORTRAN 77 code. Your system must have a FORTRAN 77 compiler to use *s.db.rim*.

SECTION ONE −− THE COMMAND VERSION

The command−line driven version of *s.db.rim* is run by typing the below command, where *data_base* is the name of an existing RIM sites data base:

> **s.db.rim** *data_base*

The sites data bases are stored in a subdirectory named 'rim/sites' in the user's current mapset. Data bases in other mapsets, selectable through the GRASS *g.mapsets* command, can be accessed for 'read−only' retrieval of records. Each mapset may have many data bases. Each data base within a mapset must have a different name; user−supplied names for data bases are limited to seven (7) characters in order to maintain compatibility with the standard version of RIM. As with other GRASS commands, mapsets are searched in the mapset SEARCH_PATH order when a data base needs to be opened.

Each site data base is composed of multi−field records (rows or tuples, in DBMS jargon). Each field and its position in the site form is defined via input to the .make command when a data base is originally defined. It is possible to add new fields or change the length of existing fields after data has been loaded, however this is not straightforward and is not described here; deleting of fields is also possible, but requires even more experience and knowledge. The user needs to carefully design the data base fields and form (layout) and check the operation with a few pieces of test data before loading data for a large number of sites.

COMMANDS

(Note: For each of the "dot commands", i.e., .make, described below there is a menu choice to selected when running the interactive version. The interactive menus are described in the SECTION TWO of this document. Some display capabilities exist in the interactive version which are not directly implemented in the command version.)

The commands are given alphabetically here for easy reference. The .make command is required to create a data base and, therefore, will be the first to be entered by a new user. Abbreviations down to the string shown in ( ) are accepted; this is primarily for those giving *s.db.rim* commands from a terminal, but abbreviations may also be used in batch files.

Each command is introduced with an input record (line) which starts with a period and is followed by one of the words shown below; for some commands the command line also contains one or more required or optional parameters. Additional or optional input instructions/data for a command are supplied on successive lines; a .end line is needed by some commands to signify the end of these input lines.

Alphabetical Command Summary

"!command"

This is the only *s.db.rim* command not starting with a period. "command" is a single shell command line which is executed by a "G_system()" call (see GRASS gis library). Many UNIX utilities (e.g., *vi, ls, print*) and most GRASS commands (e.g., *d.rast*, *d.points*, *g.list, g.region*, *d.zoom*, *r.mask*) may be executed. It is permitted, and often useful, to change "region" and "MASK" within *s.db.rim.* Multiple commands may be separated by ";" in the standard UNIX way. Note that a "!cd directory; ls" will change to the specified directory and list files, but the effective working directory for *s.db.rim* will not be changed when the command terminates.

".add (.a)"

Add a new site record (row) to the open data base. Each line following contains a field name followed by spaces and/or tabs then the value or character string to store for that field. Field information lines end with .end. Some fields may be absent and fields may appear in any order. Checks are made for the input of data for the one required field (site number), for string length for string type fields, and for duplicate site numbers. If split fields are used in the data base layout (see .make), text data for each split field must be added as a separate line. If there are any problems, the record will not be stored and a message will be output. This format makes it relatively easy to import data from most other DBMS. The ".print −a" command, see below, outputs data in this list format.

```
Example:

 .add
 site_id   204
 north     4690673.30
 east      601410.00
 reference Jones (1987)
 .end
```

".backup (.b) file_name"

The .backup command is used to dump the entire data base from the RIM binary files to a text file format (see UNLOAD in the RIM User's Manual). The file_name can be a relative path name or full path name. The file will contain the data base definition, screen layout information, and tabular data. This text file is transportable to RIM or *s.db.rim* running on any other computer; it may also be reloaded to recreate the *s.db.rim* data base. A message will be output if there is any problem writing the .backup file. Backup can only be done on data bases in the user's current mapset.
To reload your data base from the backup file (normally not necessary):

```
GRASS 4.1> cd $LOCATION/rim/sites #right directory
GRASS 4.1> rm db_name.rimdb1      #remove data base (or mv to somewhere)
GRASS 4.1> rm db_name.rimdb2      #remove data base (or mv to somewhere)
GRASS 4.1> rm db_name.rimdb3      #remove data base (or mv to somewhere)
GRASS 4.1> rim                    #run RIM manually
RIM> input "path/file"            #RIM rebuilds data base from data
                                   written by .backup
RIM> exit
```

".change (.c) [−l]"
Without the "−l" flag, each line following .change is in the same format as for the .add command. The site number field is required and the site number must match an existing site in the data base. Only those fields for which lines are provided are changed in the record. After the .end the changed record is stored, if all is ok, otherwise a message is output.
If the "−l" flag (for "list") is given, the site number field is omitted and the specified field values are changed for all sites currently selected by .find and/or .query.

".delete (.d)"
This command is used to delete data records for sites. Deletion of sites is permanent. A backup of the data base, or copies of the data base files, are the ways to protect your valuable data.
The lines following the .delete command should contain only the site numbers, with a .end line being last.

The following command sequence will delete all the sites currently on the internal site list (the result of the last .query or .find command) after asking for approval.

```
 .delete
 .end
```

".end (.e)"
Ends multi−line input for several other commands.

".exit (.ex)"
Use .exit to end operation of *s.db.rim* cleanly. In general, do not use CTRL−C to exit unless absolutely necessary. When .exit is encountered in a batch file, input will revert back to the previous file, or the terminal, if any, which called the batch file.

".find (.f) [−a | −d] [−m] [−r]"
The .find command is used to find the site(s) closest to a given point (the target). The target can be defined in one of several ways. The found sites are stored on an internal sites list for output by other commands; however, see note 2, below. The found sites are stored on the internal sites list in order of proximity to the target location.
The optional .find command line parameter specifies the current MASK (−m), if any, or the current region (−r), as a filter on the retrieved sites. −m automatically implies −r, as the MASK is not defined outside the current region. If the −a flag is given, the retrieved sites will be appended to those previously retrieved with a .query or .find; duplicates will be automatically discarded. The −d flag causes the retrieved sites to be deleted

from the internal site list, if present there. Very complex selections can be done by interspersing appends and deletes to arrive at a final list of sites. For instance, selecting those sites within 2000 meters of a target and then deleting those within 1500 meteres of the target will give a final list of those from 1500 to 2000 meters.

The single required line following the .find line gives the program the necessary target information. The following examples show the possibilities.

```
find> 602793.90 4379010.00
```

will find the one site nearest these coordinates and store it on the internal site list.

```
find> 619840 4599000 10
```

will find the 10 sites (or fewer, if there are not that many) closest to the given location.

```
find> site 132 10
```

will find the 10 sites closest to the location of site 132 in the data base (including site 132). If site 132 does not exist, no action is taken.

```
find> distance from 472910.06 5732001.0 5000
```

will find all sites within 5000 (meters, in UTM or Lat−Long coordinates) of the target location.

```
find> distance from site 16 −2500
```

will find all sites greater than 2500 (meters) from the location of site 16.

Notes for .find:

1. All sites found are stored on the site list in order of proximity to the target location (sorted by distance from target).

2. The number of sites found is automatically printed to the active output device/file.

3. If mask is specified, the effective region is automatically set to the current region (because the GRASS mask is only defined for the current region).

4. Region and mask filtering uses the current resolution for the region to test if a point falls within a cell in the masking map.

5. In the last two examples the string "distance from" must be exactly matched. Also, the word "site" must be exactly matched.

6. If the "distance from" radius is given as a negative value, points outside the target circle are selected; whereas, if a positive value is given, points inside the circle are selected.

7. The current region may be changed with !g.region or !d.zoom prior to doing a .find, and the mask may be set or removed with a variety of GRASS commands.

8. The "find>" prompt is given only when input is from a terminal.

".help (.h)"

Prints a help screen to the output device or file. Useful to have when using *s.db.rim* from a terminal, or when writing a script file of commands.

".input (.i) [file]"

The lines in the file given are read and processed as commands or data until an end of file is reached or until a .exit command is found. Input files may call other input files to a nesting depth of eight. Without a file name, stdin is used as the input file.

".list (.l)"

Lists the available data bases in the current mapset search path.

".make"

Using the .make command you create a new data base in the current mapset by specifying the following items which define the screen (page) layout for displaying and printing the site records, as well as the information fields:
1) The fixed text part of the screen layout.
2) The positions, types, names and lengths of data fields.

Three fields must always exist in a data base; each of these field types may only occur once in a data base layout:

1) Type 's' Site identification number field (an integer).
2) Type 'x' Easting coordinate of the site (a double float).
3) Type 'y' Northing coordinate of the site (a double float).

The other field types, which may occur in any combination and order, are:

4) type 'i' An integer field.
5) type 'f' A double precision float field.
6) type 't' A text field.

Each of the fields can be positioned anywhere within the screen layout, which has a limit of 19 lines by 80 columns. A maximum of 70 fields may be defined within this space. A field is specified in the screen layout by a tilde (~), a field type character, a field name and enough trailing tildes to fill out the desired field length.

Each line following the .make command is taken to define a line of the screen layout until a .end is reached. If a mistake is made on any of the input lines, the .make will fail. The .make information may be prepared in advance as a text file (this facilitates fixing mistakes) and the .input command can be used to read in this file. An example text file for a data base screen layout follows, with some important explanatory notes.

```
 .make
              Archaeological Sites Database
              ==============================
```

Site #: ~sSite~~~ Entered By: ~tEnter_by~~~~~~~~ Description: C−14 Date: ~iAge~~~
~tDescript.1~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~tDescript.2~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~tDescript.3~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Type: ~tType~~~~~~~~~~~~~ (Should be Arch. or Hist.) Date: ~tEnter_Date~~~~~~~ Square Miles: ~fArea.4~~~ North: ~yNorth~~~~~ East:

~xEast.1~~~~ .end Notes:

1) Any text not preceded by a tilde (~) character is taken to be part of the constant or fixed text portion of the form.

2) A field definition begins with a tilde (~) character immediately followed by a single character which indicates the data type of the field (s,x,y,i,f or t). Immediately following the data type character is the field name of 1 to 16 characters. Field names can be composed of any characters from the following set: [A−Z,a−z,_,0−9]; the RIM program and library do not distinguish upper and lower case in field names, so you should avoid making names which differ only in case. Field names may not begin with a numeral [0−9]. The rest of the field length is padded with tilde (~) characters to the required maximum length.

3) The minimum field width is three characters; e.g., "~tA". Be sure field widths for all fields are wide enough for the values and strings you expect to store there; e.g., UTM northings require at least 11 spaces.

4) For text fields it is possible to continue a field across more than one line. This is done by appending a .1 to the field name forming first portion of this "split field", a .2 for the second portion, etc. This text field splitting affects how information is organized for input and output; the composite text string is concatenated (unused portions of fields are retained as spaces) and treated as a unit for storage and queries to the data base.

5) For the double precision floating point fields (types x, y and f), the number of decimal places to print may be specified by appending ".n" to the field name, where n is the number of decimals places required. Values of n from 1 to 12 may be used. Two decimal places is the default if ".n" is not specified. Be certain that you make the field wide enough to print the integer and decimal portions of the values that will be stored in the field; include space for a sign and the decimal point. (If it is desired to print zero decimals, whole numbers, use an "i" type field.) In the example above, the northing coordinate (y) would be output with two decimals, the easting (x) would have only one decimal place, and the "Area" would be printed with four decimal places.

".output (.o) [file or | process]"

Causes all output (except some error messages) from *s.db.rim*, including that from the .print command, to go to the named path/file (may be a full or relative path name), or to be used as standard input by the process (a pipe). If no parameter is given, output returns to stdout, usually the user's terminal. An example of the pipe usage would be

```
      .output | grep "easting" | wc −l > /tmp/my_count
```

A pipe is closed whenever the .output command is given again, or on a .exit command.

".pack (.pa)"

This should be used when numerous data records have been deleted to recover disk space in the RIM binary data base files. It works by doing a .backup to a temporary file; moving the data base files to new names (*.bakdb*); running RIM to rebuild the data base; and, if the rebuilt data base can be opened and read, the temporary files are deleted. The user is informed if this process fails. Packing can only be done on an open data base located in the user's current mapset.

".print (.p) [−a | −l] "

This command outputs the full site records for the sites currently stored on the internal sites list (result of last .query or .find). Without the flag, the screen layout format is used. With the −l flag, for list format, the field name followed by the contents are output one field per line. The −a flag also outputs in the list format but with a .add line and a .end line surrounding each site record printed; data files in this form can be read with .input, thus they form one kind of backup mechanism and can be used to transfer data (not the data base layout) from one GRASS system to another. The destination for the output is set by a previous .output command (default is stdout).

".query (.q) [−a | −d] [−m] [−r]"

The .query command is used to retrieve sites via an SQL−like request to RIM, including a user specified "where clause." All fields for each site meeting the selection criteria are retrieved.
The optional .query command line parameters cause points not in the region (−r) and/or mask (−m) to be rejected, so these conditions need not be tested in the "where clause." The −a flag causes the retrieved sites to be appended to those previously retrieved by .query or .find; duplicate entries are automatically discarded. The −d flag causes selected sites to be deleted from the current list, if present.

After the query command line, any number of lines may be entered to define the SQL "where" clause. A .end line is required to finish the request and begin data retrieval. See examples below.

The "distance from" clause may also be used as additional selection criteria exactly as described in the examples and notes for .find. It must be entered as a separate line to the query prompt.

The retrieved records may be printed at time of retrieval, rather than after the completion of the query command by including a .print (.p) line with the same options for print format as in the .print command (see above); e.g. .p −a to output in the "list add" format. The .print clause must be entered as a separate line to the query prompt. This feature is most useful when working with very large data bases where retreval time is significant. See example 2 below.

```
Example 1

query> where density < 20 and (date = "10/14/89"
query> or county eq "San Marcos")
query> .end


Example 2

query> where east <600000 and name like "*Jones*"
query> distance from site 12 3000
query> .print −a
query> .end


Example 3

query>.end
```

The where and distance from clauses are each optional. If both are omitted, only the mask and region on the .query command line restrict the search; if mask and region are also omitted, all sites will be retrieved (Example 3). When querying for sites the where clause is processed first, the current region and mask tested next (if requested), then the distance from clause is applied; a site must pass all tests to be put on the internal site list for output by other commands.

Notes: (Also see Notes for .find)

1. The retrieved sites are stored on the internal site list in the order returned from the data base by RIM, not necessarily in site number order or the order the data was loaded. A "distance from" clause results in a final sorting by proximity to target.

2. See the RIM User's Manual for additional information on the "where" clause in the "select" command, especially the quotes required for matching character string fields, and the allowed comparison operators.

3. In the where clauses of the examples, "density", "date", "county", east", and "name" are field names (column names in RIM) defined when the user initially makes the data base.

4. Each .query or .find resets the internal site list (even unsuccessful ones), unless the −a or −d flags are used.

".read_site (.re) site_list [comment_field]"

This command reads an existing GRASS site list and creates a data base record for each site. If the comment or description field of all entries in the site list begin with # and a number, the number becomes the site number in the data base. If some of the sites in the GRASS site list do not have a # at the beginning of the comment field, the sites are numbered sequentially starting with 1. (These options are similar to the way the GRASS sites−to−raster [in s.menu] works.)
if a data base field name "comment_field" is entered on the command line, the comment will be stored in that field for each site. If an integer or float field is specified, and attempt is made to interpret the comment as that type of number; if this interpretation fails, 0 or 0.0 is stored.

If the site number duplicates one already in the data base or found earlier in the site list, it is not added.

Once the sites have been loaded by .read_site, use .change (or the interactive version) to add data to other fields for those sites.

".remove"

This command, which requires a "y" as confirmation on the next line, entirely removes the three binary files which constitute your RIM data base. Use with care. Backup files must be removed individually by the user, if desired, from the $LOCATION/rim/sites directory.

".show (.sh)"

This command is used to output the screen or page layout as defined for the current data base. It serves as documentation of the data base definition and as a reminder for field names, types and lengths. By using an editor to surround the output of .show with .make and .end lines, it can be used to reload the data base definition with .input.

".site_list (.si) file_name [field_name]"

This command writes the site locations and the site numbers to the specified file in the site_list directory in the current mapset. If the file exists, the sites are appended to the current list, otherwise, a new site list file is created. A "field_name" may be optionally specified; if so, the contents of that field (retrieved from the appropriate site record) are inserted as the comment (following a '#') in the site list. The site number is used if no field name is supplied.
A comment line is inserted in the site_list file with the current date and time and the name of the data base producing the site locations. The format used for each site is:

```
easting|northing|#comment
```

".tables (.t)"

Prints the table structure of the currently opened RIM data base. This is the same output generated by a "list *" command when running RIM manually. The information for the table named "data" is useful for review of the user's field definitions. The information for the two other tables is for internal use by *s.db.rim.*

SECTION TWO −− THE INTERACTIVE VERSION MENUS AND COMMENTS

SYNOPSIS

s.db.rim

DESCRIPTION

The interactive version of *s.db.rim* allows you to create, manage and query information about site locations (sites) across the landscape. Operations are done on a data base through a series of menus explained below. Most of the menus use VASK screens; the user should become familiar with keys that move the cursor among the fields to be entered (RETURN, ENTER, CTRL−L, CTRL−K, etc.).

THE MAIN MENU

Below is the main menu. Option 1 is the default. Note the status line at the top of the menu, and the fact that 8 records have been selected by the latest find or query operation (between items 2 and 3). Note, also, that CTRL−C can be used to exit from this menu (and most other menus in the program) back to the GRASS prompt. The specifics of each menu choice are described below.

```
    s.db.rim              MAIN  MENU                  Version 1.4
        Data base <water> in mapset <rono> open.  25 records.

       1  Open a data base
       2  List available data bases
  --------  Retrieve/Output Site Records (8 currently)  --
       3  Find sites in proximity to a Target point
       4  Query to select site records (SQL)
       5  Show selected site records on Terminal
       6  Display maps/selected sites on graphics terminal
       7  Output selected site records to Printer or File
       8  Create a site_list from selected records
  ------------  Add/Edit Site Records  ----------
       9  View a single site record
      10  Add a site record
      11  Change a site record
      12  Delete a single record or all selected records
  ------- Other functions −− Shell Command −− Exit ---------
      13  Make a new data base & Management Functions
      14  Execute a shell command

       0  Done −− Exit from s.db.rim

    AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
            (OR <Ctrl−C> TO EXIT THIS PROGRAM)
```

1. Open a data base. If a data base is already open, it is closed before the requested one is opened. Only data bases in the user's current mapset may be modified; others are opened in read−only mode; this will be indicated on line 2.

2. List available data bases. For each mapset in the current GRASS mapset search path, the names of the existing data bases are listed.

3. "Find" sites in the data base relative to a specified target location. This is used to select sites based on proximity to the target and, optionally, sites within the current region and, optionally, sites falling in active cells within the current GRASS mask. Two modes of targeting are provided: the N sites closest to the target, and all sites within (or outside) a circle of specified radius from the target. The FIND/QUERY TARGET MENU discussed below accepts region/mask/target specifications from the user. The selected sites are then displayed one at a time until CTRL−C is entered; then other operations, choices 5−8, can be done with these sites. The line on the menu between 2 and 3 shows the number of sites currently selected by choices 3 or 4.

4. "Query" sites in the data base using an SQL−like "where clause," including specifications for region/mask/target (circle only) as in 3, above; see FIND/QUERY TARGET MENU section below. The where clause can test for ranges or matches for numeric data base fields, or matches on full strings or substrings for text fields. The selected sites are then displayed one at a time until CTRL−C is entered; then other operations, choices 5−8, can be done with these sites. This clause is entered on a menu described below; see QUERY COMMAND MENU section, below.

The where clause may use parentheses ( ) to control the order of comparisons. Field names are not case sensitive within where clauses. The following comparison operators are valid for all types of fields:

```
          eq   or   =           ne   or   <>
          ge   or  >=           le   or   <=
          gt   or   >           lt   or    <
```

String comparisons are case sensitive and are done character by character. Substrings comparisons may be done with the "like" operator as in:

```
          where name like "*Jones*"
```

Note that the string being tested against the name field for each record is in quotes (single or double) and that wild card comparisons can be done in the standard way with '*' and '?' characters.

Logical comparisons may also be combined with those operators above. The permitted logical operators are:

```
and       or       not
```

The following complex example should be examined. The line breaks can occur between any tokens (words, values, operators), except within quoted strings.

```
    where (name like "*Jones*" or name = "Smith")
    and ( ( site < 300 and not (site = 251 or site eq 15) )
    or east < 601000 )
```

5. This choice will display the site records resulting from the last find/query one at a time on the terminal. Use ESC or enter a number to display another record and CTRL−C to end the display.

6. If a graphics monitor is active, the locations of the selected sites will be displayed. The user may choose to erase the screen; display raster, vector, and/or site maps; or display the selected sites from the data base. These maps are requested through the following interactive screen. Just enter ESC to skip this step. If no data base

sites are currently selected, that section of the menu will not appear; but the menu can still be used to display the other types of maps. This display function is a major added function of the interactive version of the program; display is not so easy in the command version.

```
               SELECTION MENU FOR ITEMS TO DISPLAY



Enter raster and/or vector map names, if desired



 ______________  Raster map to display
 ______________  Vector map to display in color: _________
 ______________  Site list to display
                 Dpoints with: size=3_ type=box____ color=white____
               _ Display currently selected sites (enter x)
                 d.sites with: size=6_ type=x______ color=red______
               _ Erase graphics screen (enter x)
                 d.erase  black____


              AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
                            (OR <Ctrl−C> TO CANCEL)
```

7. This selection results in a screen prompting for the name of the file to output the selected site records to, and for optional formatting selection. If the file name is lp, the site records are sent to the printer. The optional formatting choices are for export of data in list format (see .print in the first part of this manual page for *s.db.rim* for information and examples).

8. Using this choice you can write (or append) the currently selected sites to a GRASS site_list file in your current mapset. A short menu prompts for the name of the site_list file, and also for the name of a field to be used for the "comment" in the site_list (the site number is the default field). The current date and time, and the names of the mapset and data base in use are entered as an information line in the site_list file. Note that various kinds of raster map layers can be produced from a *s.db.rim* data base by writing site_lists with different fields as "comments" then converting the site_lists to raster files with *s.menu*.

9. Choices 9−12 operate on only a single site and do not use or modify the internal list of sites selected by find/query (choices 3 or 4). Choice 9 is the way to view a single site record, selected by site number. After viewing, ESC will allow entry of another site number and CTRL−C will exit to the main menu.

10. Use this selection to add a new site record to the data base. (A new site is one whose site number does not currently exist in the open data base.) After making this selection, the data base layout will be displayed and you should enter the available information appropriate to each field; the only required entry is the site number field. If values for numeric fields are not entered, zero values will be stored. Unused portions of text fields are stored as strings of spaces.

11. After making this selection and specifying the site number to change field information for, the data is entered as for choice 10, except that the site number cannot be changed. (The command version of the program has provision for making bulk changes after a find or query; see .change.)

12. To delete a single record, enter its site number when requested. All site records chosen by the last find/query operation may be deleted by entering "list" in place of the site number. BE CAREFUL with this,

deleted records are really gone.

13. This choice starts a new menu with less commonly used functions. See MANAGEMENT MENU section below.

14. The program will prompt you for one−line Shell Commands until you enter just a <RETURN> to return to the main menu. Often useful for changing the GRASS region, setting a MASK, etc.

FIND/QUERY TARGET MENU

This is the screen to set up the region/mask/target information for the find choice (3) and the query choice (4), except that item B is omitted for choice (4). The choice to append or delete selected records will only be given after a successful find or query has stored some records on the internal record list. See .find and .query for more information.

If a graphics monitor is not active, the "mouse" item is omitted from the screen; and, if a mask is not set, that choice is omitted. The choices entered on this example screen will result in all the sites within a 1500 (meters) radius of the target point (to be chosen with the mouse) being selected and stored on the internal site list by find or query. They are stored in order of proximity to the target. If a site is used as the target, it is always the first in the retrieved list (useful for just selecting one site by number). If a mouse is chosen to select the target point, a menu to display reference maps is presented, exactly as in choice (6), prior to actually activating the mouse.

```
        QUERY/FIND:  REGION/MASK/TARGET SELECTION MENU
  Data base <arch> (READONLY) in mapset <PERMANENT> open.  113 records.
    Mark requests with 'x' and enter required values.

               Respect current region   _
               Respect current MASK     x
              (forces current region)

A.  Find all sites within (or outside) a circular target   x
             and give the radius (negative for outside)  1500.00_____
                        OR
B.  Find a number of sites nearest a point   _
         and the number of sites requested   ________

    After selecting A or B, complete one(!) of these:
        1. x to select target point with mouse    x
        2. Enter site number for target point     __________
        3. Target coordinates            east    0.00________
                                        north    0.00________

 Append/Delete to current FIND/QUERY site list (a | d) _
 Reset to default choices for this menu _

         AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
                       (OR <Ctrl−C> TO CANCEL)
```

QUERY COMMAND MENU

The following screen completes the information for a query (choice 4). It may be left blank if no "where clause" is required. After a successful query, the selected records are displayed one at a time by hiting escape;

CTRL−C will quit the display and return to the main menu where several choices of operation on the retrieved sites are offered.

```
        QUERY COMMAND CONSTRUCTION SCREEN
 Data base <A> in mapset <rim_test> open.  25 records.
 The SQL select query will use the current region
 and a target clause of 'distance from 596463.15 4919041.88'

where date = 10/16/89_____________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________
__________________________________________________________

(Enter .show on a line to review screen layout and field names.)

  AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
              (OR <Ctrl−C> TO CANCEL)
```

MANAGEMENT MENU

Choice 13 from the main menu presents this menu. Each item is discussed below.

```
  s.db.rim       DATA  BASE  MANAGEMENT  MENU
Data base <A> in mapset <rim_test> open.  15025 records.

  1   Make a New Data Base in Current Mapset
  2   List Available Data Bases
  3   Remove (PERMANENTLY) Data Base from Current Mapset
  4   Recover a Data Base from a RIM ASCII File
  5   Show Screen Layout of Current Data Base
  6   Backup (UNLOAD) Data Base to RIM ASCII Format File
  7   Pack the Current Data Base
  8   Read a Site list into the Current Data Base

  0   Return to Main Menu

 0_ Your selection

 AFTER COMPLETING ALL ANSWERS, HIT <ESC> TO CONTINUE
             (OR <Ctrl−C> TO CANCEL)
```

1. Use this choice to create a new *s.db.rim* data base in the current GRASS mapset. See section below on MAKE A NEW DATA BASE.

2. List available data bases. Like 2 on MAIN MENU.

3. Delete an entire data base from the current mapset. The name of the data base and additional confirmation of the action are prompted for.

4. Choice 6 allows backup of the definition and data parts of a data base to a transportable text file. To rebuild (or build for the first time) a *s.db.rim* data base from one of these text files do the following steps:

```
   # see if the rim directory exists.
ls $LOCATION/rim/sites
   # if the directory was not found, make it.
mkdir $LOCATION/rim/sites
   # change directory to it.
cd $LOCATION/rim/sites
   # have rim build the binary data base files.
rim
RIM> input '/path/to/your/textfile'
RIM> exit
```

The data base is thus created in the current mapset. Several *s.db.rim* commands should be run to verify the integrity of the newly created data base.

5. This merely shows the screen layout of the currently open data base. It is a useful way to quickly see the layout and review the field names and types.

6. When backing up to a text file, the RIM UNLOAD command is run with the output directed to a file of the user's choice. See 4 above. It is wise to do this operation after extensive changes or additions of data records. The resulting text file can be written to tape for preservation, or shared with other GRASS systems, if desired.

7. After deleting a large number of site records, some "wasted" disk space will be present in the binary data base files. This procedure will perform an unload and a reload automatically to recover this unusable disk space. If there is any problem reopening the data base after packing, the user is notified and can recover in various ways depending on the backups which have been done.

8. Data may be loaded into a data base from an existing GRASS site_list. This procedure will prompt for the site_list name and then add the sites to the currently open data base. If all sites in the list have a comment field of the form "#value ....", the value is used as the data base site number, otherwise the sites are numbered sequentially beginning with 1. Only the site number and location coordinates are loaded for each site record by this procedure; other fields may be later added with the "change" function. See .read_sites.

MAKE A NEW DATA BASE

After entering the name of the new data base you wish to create (7 characters maximum), you then decide how to input the information required. This input may be from a text file, or may be entered directly using the editor of your choice; the former is recommended.

See .make for the way to define a data base and record (form) layout.

NOTES

This program is included in the GRASS 4.0 release, but is not automatically compiled with other GRASS commands. The user must compile this program separately.

s.db.rim interfaces to the RIM program. Both *rim* and *s.db.rim* contain FORTRAN code. The user must have access to a FORTRAN compiler in order to compile and use *s.db.rim*. See the FILES section, below, for the location of source code.

A "date" type field should be added to future versions. This version only allows storing of dates as strings (unless the user codes them to integers), and thus only string type searches can be made for dates.

BUGS

On LINUX I have some problems:

> − If You select menu 13/8 and hit ESC, there is no output to the screen. *s.db.rim* expects here the name of the site file or the keyword *list*.
> − If You select menu 8, You hould be asked for the site_list filename, but the screen remains empty. Simply enter the name.

FILES

The source code for RIM is located under $GISBASE/../src.related/rim

The source code for *s.db.rim* is located under $GISBASE/../src.garden/grass.rim/s.db.rim

SEE ALSO

The RIM User's Manual by Jim Fox, Academic Computing Services, Univ. of Washington. See especially Appendix B on redistribution of RIM.
The RIM Installers manual.

GRASS 4.1 Installation Guide, by Jim Westervelt and Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

d.icons
d.points
g.mapsets
g.region
r.mask
s.in.ascii
s.menu
s.out.ascii
v.db.rim

AUTHORS

James Hinthorne and David Satnik, GIS Laboratory, Central Washington University, Ellensburg, WA.

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.delaunay − Module to create a vector delaunay triangulation from a sites file.

SYNOPSIS

s.delaunay [−aql] sites=*name* **vect**=*name*

DESCRIPTION

s.delaunay uses an existing sites list (sitesname) to do Delaunay triangulation, putting the results in a binary vector file (vectname).

Flag:

−a
: Use all sites (do not limit to current region)

−q
: Quiet

−l
: Output triangulation as a graph, not areas

Parameters:

sites
: Sites file to process

vect
: Vector map layer to contain delaunay triangulation.

s.delaunay can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies the name of an existing site list file and a name for a vect file, using the form

```
              s.delaunay [−aql] sitesname vectname
```

where sitesname is the name of an existing site list file and vectname is the name of vector output file. Alternately, the user can simply type *s.delaunay* on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS parser interface described in the manual entry for parser. In either case, v.support should be run on the output.

REFERENCE

Steve J. Fortune, (1987). A Sweepline Algorithm for Voronoi Diagrams, Algorithmica 2, 153−174.

SEE ALSO

v.autocorr, *v.spag*, *v.support*, *s.hull*, *s.voronoi*, *s.sweep*

AUTHOR

James Darrell McCauley,
GRASS 5 update, improvements: Andrea Aime, Modena, Italy

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.geom − Computes Delaunay triangulation, MinMax−Angle triangulation, MinMax−Slope triangulation, MaxMin−Height triangulation, Regular triangulation, planesweep triangulation, Voronoi diagram, and convex hull of sites in 2 and 2 1/2 dimensions.

SYNOPSIS

s.geom
s.geom help
s.geom input=*name* **output=***name* [**precision=***value*] [**operation=***name*]

DESCRIPTION

s.geom takes a sites file as input and computes various triangulations, the Voronoi diagram, or the convex hull of the sites. The z−coordinate is read from the description field if it is specified, otherwise 0 is assumed. The z−coordinate is used for the MinMax−slope triangulation and for the regular triangulation (where it is interpreted as the weight of the site). For all other computations the z−coordinate is ignored.

The MinMax−angle triangulation is the triangulation for the sites which minmizes (lexicographically) the sorted vector of all the angles of triangles in the triangulation. The MaxMin−height and MinMax−slope triangulations are similar. The algorithms used for the computations are not heuristics, they actually achieve the optimum.

The regular triangulation is the weighted version of the delaunay triangulation (weights are assigned to the sites, the delaunay triangulation corresponds to the regular triangulation where all the sites have identical weights).

The output is saved in vector file format.

OPTIONS

Parameters:

input=*name*
Input vector (level 2) file.
output=*name*
Output vector file.
precision=*value*
Number of significant positions after the decimal point. (default is 0).
operation=*name*

One of the following: *sweep*, *delaunay*, *angle*, *height*, *slope*, *hull*. These correspond to the planesweep triangulation, Delaunay triangulation, MinMax−angle triangulation, MaxMin−height triangulation, MinMax−slope triangulation, regular triangulation, Voronoi diagram, and convex hull, respectively. (default is Delaunay triangulation).

NOTE

Only the sites which fall into the current region are used for the computations.

The computation times for the various operations depends strongly on the algorithm used.

The plansweep triangulation and convex hull computation require *O(n log n)* operations in the worst case [Ed]. The Delaunay heuristic needs *O(n^2)* time in the worst case, however it performs much faster in practice. The MinMax−angle and MaxMin−height triangulations need *O(n^2 log n)* operations [BeEd, EdTa], and the MinMax−slope triangulation needs *O(n^3)* operations [BeEd].

Internally, the coordinates of the sites are stored in fix−point format. Therefore, the number of decimal digits cannot exceed 64 bit (or apprx. 16 decimal digits).

BUGS

Some fields of the header in the output file are not properly set.

REFERENCES

[BeEd] M.Bern, H. Edelsbrunner, D. Eppstein, S. Mitchel, T.S. Tan. Edge Insertion for Optimal Triangulations. *In Proc. 1st Latin American Sympos. Theoret. Informatics 1992*, 46−−60.

[Ed] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer−Verlag, Heidelberg, Germany, 1987.

[EdSh] H. Edelsbrunner, N. R. Shah. Incremental Flipping Works for Regular Triangulations. *In Proc. 8th Ann. Sympos. Comput. Geom. 1992*, 43−52.

[EdTa] H. Edelsbrunner, T.S. Tan and R. Waupotitsch. An O(n^2 log n) Time Algorithm for the MinMax Angle Triangulation. *SIAM J. Sci. Statist. Comput. 13 1992*, 994−1008.

SEE ALSO

v.geom

AUTHOR

Roman Waupotitsch

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

shade.clr.sh − Creates a color shaded relief map based on current resolution settings and sun altitude and color azimuths values entered by the user.
(GRASS Shell Script)

SYNOPSIS

shade.clr.sh
shade.clr.sh help
shade.clr.sh [**altitude=***value*] [**r_azimuth=***value*] [**g_azimuth=***value*] [**b_azimuth=***value*] [**elevation=***name*] [**shade=***value*|*m*|*f*]

DESCRIPTION

shade.clr.sh is a Bourne shell (sh(1)) script that creates a colored raster shaded relief map based on current resolution settings and on sun altitude and color light azimuth values entered by the user. The new shaded relief map is named *<elevation>.shade* and stored in the user's current mapset.

If no parameters are provided on startup, this program is interactive; thus if the user enters the command:

> **shade.clr.sh**

The program then prompts the user to enter values for:

1. The **altitude** of the sun in degrees above the horizon (a value between 0 and 90 degrees), and
2. The **azimuth** of the red, green, and blue lights in degrees to the east of north (a value between −1 and 360 degrees).
3. The name of a raster map layer whose cell category values are to provide **elevation** values for the shaded relief map. Typically, this would be a map layer of elevation; however, any raster map layer can be named.
4. The scaling parameter, which compensates for a different horizontal **scale** than vertical scale. For example, when a latitude−longitude projection is used with an elevation map measured in meters. If 'scale' is a number then the ewres and nsres are multiplied by that scale to calculate the shading. If 'scale' is the letter M (either case) the number of meters in a degree of latitude is used as the scale. If 'scale' is the letter F (either case) then the number of feet in a degree is used. The script scales latitude and longitude equally, so it's only approximately right, but for shading its close enough. It makes the difference between a usable and unusable shade.

Specifically, *shade.clr.sh* executes the following *r.mapcalc* statement:

```
r.mapcalc << EOF
```

```
        $ELEV.shade = eval( \\
        x=($elev[-1,-1] + 2.*$elev[0,-1] + $elev[1,-1] \\
        -$elev[-1,1] - 2.*$elev[0,1] - $elev[1,1])/(8.*ewres()*$shade) , \\
        y=($elev[-1,-1] + 2.*$elev[-1,0] + $elev[-1,1] \\
        -$elev[1,-1] - 2.*$elev[1,0] - $elev[1,1])/(8.* nsres()*$shade) , \\
        slope=90.-atan(sqrt(x*x + y*y)), \\
        a=round(atan(x,y)), \\
        a=if(isnull(a),1,a), \\
        aspect=if(x!=0||y!=0,if(a,a,360.)), \\
        rang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
        cos($raz-aspect),\\
        red = int(if(rang < 0.,0.,4.9*rang)), \\
        gang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
        cos($gaz-aspect),\\
        green = int(if(gang < 0.,0.,4.9*gang)), \\
        bang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
        cos($baz-aspect), \\
        blue = int(if(bang < 0.,0.,4.9*bang)), \\
        1. + red + 5. * green + 25. * blue )
EOF
```

Refer to the manual entry for *r.mapcalc* for an explanation of the filtering syntax shown in the above expression. See, for example, the section on "The Neighborhood Modifier".

shade.clr.sh then runs *r.colors* to assign a color table to the new shaded relief map *<elevation>.shade*.

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/shade.clr.sh`**.

SEE ALSO

An Algebra for GIS and Image Processing, by Michael Shapiro and Jim Westervelt, U.S. Army Construction Engineering Research Laboratory (March/1991) (get from GRASS web site).

shade.rel.sh
blend.sh
g.ask
g.region
r.colors
r.mapcalc

AUTHOR

Jim Westervelt, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2003/08/26 07:03:47 $

NAME

shade.rel.sh − Creates a shaded relief map based on current resolution settings and sun altitude and azimuth values entered by the user.
(GRASS Shell Script)

SYNOPSIS

shade.rel.sh
shade.rel.sh help
shade.rel.sh [**altitude=***value*] [**azimuth=***value*] [**elevation=***name*] [**shade=***value|m|f*]

DESCRIPTION

shade.rel.sh is a Bourne shell (sh(1)) script that creates a raster shaded relief map based on current resolution settings and on sun altitude and azimuth values entered by the user. The new shaded relief map is named *<elevation>.shade* and stored in the user's current mapset. The map is assigned a grey−scale color table.

If no parameters are provided on startup, this program is interactive; thus if the user enters the command:

> **shade.rel.sh**

The program then prompts the user to enter values for:

1. The **altitude** of the sun in degrees above the horizon (a value between 0 and 90 degrees), and
2. The **azimuth** of the sun in degrees to the east of north (a value between −1 and 360 degrees).
3. The name of a raster map layer whose cell category values are to provide **elevation** values for the shaded relief map. Typically, this would be a map layer of elevation; however, any raster map layer can be named.
4. The scaling parameter, which compensates for a different horizontal **scale** than vertical scale. For example, when a latitude−longitude projection is used with an elevation map measured in meters. If 'scale' is a number then the ewres and nsres are multiplied by that scale to calculate the shading. If 'scale' is the letter M (either case) the number of meters in a degree of latitude is used as the scale. If 'scale' is the letter F (either case) then the number of feet in a degree is used. The script scales latitude and longitude equally, so it's only approximately right, but for shading its close enough. It makes the difference between a usable and unusable shade.

Specifically, *shade.rel.sh* executes the following *r.mapcalc* statement:

```
r.mapcalc << EOF
     $ELEV.shade = eval( \\
     x=($elev[−1,−1] + 2*$elev[0,−1] + $elev[1,−1] \\
     −$elev[−1,1] − 2*$elev[0,1] − $elev[1,1])/(8.*ewres()*$scale) , \\
     y=($elev[-1,-1] + 2*$elev[-1,0] + $elev[-1,1] \\
     -$elev[1,-1] - 2*$elev[1,0] - $elev[1,1])/(8.*nsres()*$scale) , \\
     slope=90.-atan(sqrt(x*x + y*y)), \\
     a=round(atan(x,y)), \\
     a=if(isnull(a),1,a), \\
     aspect=if(x!=0||y!=0,if(a,a,360.)), \\
     cang = sin($alt)*sin(slope) + cos($alt)*cos(slope) *
     cos($az-aspect), \\
     if(cang < 0.,0.,100.*cang), \\
     if(isnull(cang), null(), 100.*cang))
EOF
```

Refer to the manual entry for *r.mapcalc* for an explanation of the filtering syntax shown in the above expression. See, for example, the section on "The Neighborhood Modifier".

shade.rel.sh then runs *r.colors* to assign a grey−scale color table to the new shaded relief map *<elevation>.shade*, by executing the command:

> **r.colors shade color**=grey

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/shade.rel.sh`**.

SEE ALSO

An Algebra for GIS and Image Processing, by Michael Shapiro and Jim Westervelt, U.S. Army Construction Engineering Research Laboratory (March/1991) (get from GRASS web site).

shade.clr.sh
blend.sh
g.ask
g.region
r.colors
r.mapcalc

AUTHOR

Jim Westervelt, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2003/08/26 07:03:47 $

NAME

show.color.sh − Displays and names available primary colors used by GRASS programs, in frames on the graphics monitor.
(GRASS Shell Script)

SYNOPSIS

show.color.sh

DESCRIPTION

show.color.sh is a UNIX Bourne shell macro that displays and names available primary colors used by GRASS programs in frames on the graphics monitor. Available colors are: *red*, *orange*, *yellow*, *green*, *blue*, *indigo*, *violet*, *white*, *black*, *gray*, *brown*, *magenta*, and *aqua*.

No program arguments are required to run this program.

NOTES

The full monitor screen is made the active frame after this program ends.

This macro is located in **`$GISBASE/scripts/show.color.sh`**.

SEE ALSO

d.colormode
d.colors
d.colortable
d.display
d.frame
grass.logo.sh
show.fonts.sh

AUTHOR

David Gerdes, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

show.fonts.sh − Displays and names available font types in the active display frame on the graphics monitor. *(GRASS Shell Script)*

SYNOPSIS

show.fonts.sh

DESCRIPTION

show.fonts.sh is a UNIX Bourne shell macro which runs the *d.erase* −*a* command, then names and displays the font types that can be selected using *d.font*. This macro also runs the GRASS commands *d.font* and *d.text*. See the manual entry for *d.font* for instructions on choosing a font type.

No program arguments are required to run this program.

BUGS

The font is set to *romans* (Roman simplex) after running *show.fonts.sh*. There is no mechanism to query the current font, so there is no way to automatically restore the font. The user will have to reset the font type using *d.font* if *romans* is not desired.

FILES

This program is simply a shell script stored under the **`$GISBASE/scripts`** directory. Users are encouraged to examine the shell script programs stored here and to produce others for their own use.

SEE ALSO

d.display
d.erase
d.font
grass.logo.sh
d.label
d.legend
d.paint.labels
d.text
d.title

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.hull − Uses a GRASS sites list to produce a convex hull vector map
(GRASS Sites Program)

SYNOPSIS

s.hull [**−as**] **sites=***name* **vect=***name*

DESCRIPTION

s.hull computes the convex hull of a sites list and outputs it in a binary vector file (vect).

OPTIONS

Flags:

−a
: Use all sites found in the named sites file, rather than limiting output to sites falling within the current geographic region.

−s
: Automatically run "v.support" on newly created vector file.

Parameters:

sites=*name*
: Name of existing sites file.

vect=*name*
: Name of the output file (binary vector file).

s.hull can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies the name of an existing site list file and a name for a vect file, using the form

```
            s.hull [-a] sites=name1 vect=name2
```

where name1 is the name of an existing site list file and name2 is the name of vector output file.

Alternately, the user can simply type s.hull on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS parser interface described in the manual entry for parser. In either case, v.support should be run on the output (the "−s" flag can be used for that).

EXAMPLES

REFERENCES

M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf, (2000). Computational geometry, chapter 1.1, 2−8.

SEE ALSO

v.support

AUTHOR

Andrea Aime, Modena, Italy

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.in.ascii − Converts an ASCII listing of site locations and their descriptions into a GRASS site list file.
(GRASS Sites Program)

SYNOPSIS

s.in.ascii
s.in.ascii help
s.in.ascii sites=*name* [**input=***name*] [**d=***value*] [**fs=***character|space|tab*] [**date=***timestamp*[/*timestamp*]]

DESCRIPTION

s.in.ascii converts an ASCII listing of site locations and category labels into a file in GRASS site list file format.

Input can be entered via standard input or from the file **input=***name*. Each line of input should contain the easting, northing, and either the category value or category label associated with a site. The **fs=***name* option (where *name* is either a character, a space, or a tab) can be used to specify the use of a particular field separator between these three input fields. This is useful when input is obtained from other programs (see NOTES, below). Output is stored in the file **sites=***name* and placed in the site_lists/ directory under the user's current mapset.

The GRASS program *s.out.ascii* can be used to perform the reverse function, converting a file in GRASS site list format into an ASCII listing of eastings, northings, and category labels associated with site locations.

Parameters:

sites=*name*
Name of the new GRASS site list file to be output.
input=*name*
Name of an existing ASCII file containing site locations and labels.
d=*value*
number of dimensions (default=2)
fs=*character|space|tab*
The field separator separating the easting, northing, and category label in each line of the *input* file.
The field separator can be a character, a space, or a tab.
Default: space
date=*timestamp[/timestamp]*
String specifying timestamp or timestamp range.

s.in.ascii can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies a name to be assigned to the **sites** file output, the name of an existing ASCII file containing

input, and (optionally) a field separator **fs** appearing in the **input** file.

Alternately, the user can simply type **s.in.ascii** on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS parser interface described in the manual entry for *parser*. If the user does not specify the name of an **input** file containing site locations and category attributes, these should be entered to the program via standard input. The **d** parameter allows the user to specify that more than 2 dimensions will be imported. Otherwise the third (or further) column in the **input** file will be treated as an attribute.

To define a **date** (timestamp), several date strings are accepted. Please see *r.timestamp* for details.

NOTES

Importing from other GRASS programs

Other GRASS programs can be used to produce output in a format suitable for input to *s.in.ascii*. For example, the user might pipe output produced by *d.where* into *s.in.ascii* to create a site list file called *my.sites* containing site locations pointed to with the mouse, as illustrated below. In this example it is unnecessary to specify the field separator used in the input, since *d.where* output separates the easting and northing values with spaces, and spaces are the default field separator assumed by *s.in.ascii*.

d.where | **s.in.ascii sites**=my.sites

Importing from a spreadsheet

Data may be imported from many spreadsheet programs by saving the spreadsheet as a comma separated variable (.csv) text file, and then using the "**fs=**," command line parameter with *s.in.ascii*. Note *s.in.ascii* currently requires there to be at least three columns of data for the import to work properly.

Importing latitude/longitude data

Latitude/longitude data may be imported either in decimal degree format:

```
8.314824 54.921730 site1
8.897605 54.872353 site2
9.549371 54.834080 site3
```

or in DMS (degree, minutes, seconds) format:

```
8:18:53.3664E 54:55:18.228N site1
8:53:51.378E  54:52:20.4708N site2
9:32:57.7356E 54:50:02.688N site3
```

Time as String Attributes

In this example, we will work with the following site list:

```
10.8 0 9.8 Fri Sep 13 10:00:00 1986 31.4
11 5.5 9.9 Fri Sep 14 00:20:00 1985 36.4
5.1 3.9 10 Fri Sep 15 00:00:30 1984 28.4
```

This data has three dimensions (assume easting, northing, and elevation), five string attributes, and one decimal attribute.

GRASS 5.0 internal sites format in "site_lists/"

If the user wishes to generate site list files directly, the internal format is described below.

Header:

```
#any comment, can be inserted anywhere
name|mysoils.site
desc|3D soils data for location A
time|15 Jan 1999
labels|east north elevation ID pH Corg color
form||||#%%@
```

Data:

Each line of the ASCII file should contain either two or three coordinates (x,y and optionally z) separated by a pipe ("|") character. After the coordinates there should be trailing pipe character, and optionally attribute fields may follow separated from each other with blank spaces.

The attribute field may be a category number, a decimal value or a string. Category numbers must come first and be preceded by the "#" character, while string values must be preceded by the "@" character. Floating point values may be preceded by the "%" character but if there is no "#" or "@" preceding the attribute then it is assumed to be a floating point (FP) value. Also, string values that contain blanks must be quoted or the part of the string following the first blank will be parsed as a separate field, which may cause an error.

General form
easting|northing|[z|[d4|]...][#category_int] [[@attr_text OR %FP] ...]

Example
`739865.8|4279785.5|#2965 %396685 %194919 %2.473222 @"St.Louis" @MO @city`

There can be many dimensions between pipes (|), but no #%@.

There can be only one cat, preceded by #, after which there may be many FP or text attributes.

For more information, see the *GRASS ASCII formats* page or the *GRASS 5 Programmer's manual*.

SEE ALSO

d.points, *d.sites*, *d.what.rast*, *d.where*, *r.timestamp*, *s.out.ascii*, *parser*, *GRASS ASCII formats*

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

s.in.atkisdgm − Imports a German ATKIS DEM file (.dgm) into a GRASS site list file.
(GRASS Sites Import Program)

SYNOPSIS

s.in.atkisdgm
s.in.atkisdgm help
s.in.atkisdgm *dgmfile*

DESCRIPTION

s.in.atkisdgm imports a German ATKIS DEM file (.dgm) ("Digitales Geländemodell") into a GRASS site list file. ATKIS is the "Amtliches Topographisch−Kartographisches Informationssystem".

Input can be entered from the file *dgmfile*. The standardized ATKIS DGM structure is recognized properly. Output is stored in the file *dgmfile* (identical name) and placed in the site_lists directory under the user's current mapset.

Parameters:

dgmfile
  Name of an ATKIS DGM file containing site locations and labels.

NOTES

As the ATKIS−DGM files represent a regular elevation grid, these maps can be directly converted into raster format with *s.to.rast*.

SEE ALSO

s.info, s.in.ascii, s.in.atkisktb, s.out.shape, s.out.ascii, s.to.rast

AUTHORS

Markus Neteler, University of Hannover
Otto Dassau

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.in.atkisktb − Imports a German ATKIS KTB file (.ktb) into a GRASS site list file.
(GRASS Sites Import Program)

SYNOPSIS

s.in.atkisktb
s.in.atkisktb help
s.in.atkisktb *ktbfile*

DESCRIPTION

s.in.atkisktb imports a German ATKIS KTB file (.ktb) ("Kotentabelle") into a GRASS site list file. ATKIS is the "Amtliches Topographisch−Kartographisches Informationssystem".

Input can be entered from the file *ktbfile*. The standardized ATKIS KTB structure is recognized properly. Output is stored in the file *ktbfile* (identical name) and placed in the site_lists directory under the user's current mapset.

Parameters:

ktbfile
Name of an ATKIS KTB file containing site locations and labels.

NOTES

The ATKIS KTB files contain irregularly distributed elevation points.

SEE ALSO

s.info, s.in.ascii, s.in.atkisdgm, s.out.shape, s.out.ascii, s.surf.rst

AUTHORS

Markus Neteler, University of Hannover
Otto Dassau

Last changed: $Date: 2002/03/01 00:08:31 $

NAME

s.in.dbf − Imports a dBase (.dbf) table of site locations and their descriptions into a GRASS site list file.
(GRASS Sites Import Program)

SYNOPSIS

s.in.dbf
s.in.dbf help
s.in.dbf [−lz] **input=***name* [**sites=***name*] [**date=***timestamp*[/*timestamp*]]

DESCRIPTION

s.in.dbf converts a dBase (.dbf) table of site locations and category labels into a file in GRASS site list file format.

Input can be entered from the file **input=***name*. Each line of input should contain the easting, northing, and (optionally) the category label associated with a site. Output is stored in the file **sites=***name* and placed in the site_lists/ directory under the user's current mapset.

Flags:

−l
List fields (including the field types and field widths) of DBF file, then exit.
−z
treat third column as third dimension (z)

Parameters:

input*=name*
Name of an existing dBase file containing the data to be imported.
sites*=name*
Name of the new GRASS site list file to be output.
date*=timstamp[/timestamp]*
String specifying timestamp or timestamp range.

s.in.dbf can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies a name to be assigned to the **sites** file output, and the name of an existing DBF file containing **input**.

Alternately, the user can simply type **s.in.dbf** on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

To define a **date** (timestamp), several date strings are accepted. Please see *r.timestamp* for details.

GRASS 5.0 internal sites format in "site_lists/"

See *s.in.ascii*

TODO

A field selection/reordering parameter is not yet provided. So the EAST/NORTH columns have to be the first and second column in the table.

SEE ALSO

s.in.ascii
s.out.shape
s.out.ascii
r.timestamp
parser

AUTHORS

Markus Neteler, University of Hannover, with code from:
Alex Shevlakov (pg.in.dbf)
Frank Warmerdam (DBF Routines in pg.in.dbf)
David D. Gray (−l flag in v.in.shape)

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

s.info − reports attribute, label, and other information about a sites file.
(*GRASS Sites Program*)

SYNOPSIS

s.info
s.info help
s.info [−qwQ] sites=name

DESCRIPTION

s.info reads the entire sites file and reports header & label information, type of category, number of attributes & dimensions, and min and max values for each dimension, attribute, or category. For best results sites should conform to the multiple attribute format as specified in GRASS5.0.

OPTIONS

This program can be run non−interactively or interactively. The program will be run non−interactively; if the user spcifies program arguments and flag settings on the command line using the form

s.info [−qwQ] sites=name

Alternately, the user can type simply s.info on the command line; in this case, the program will prompt the user for parameter values using the standard GRASS parser interface described in the manual entry for *parser*.

Flags

−q Load quietly
−w Only count sites in current window for min/max.
−Q Show quartiles for decimal attributes

Parameters

sites = *name*
Name of sites file.

Example output

SITES FILENAME: r_analys −−−−−−−−−−−−−− Header Information: −−−−−−−−−−−−−−−−−− name r_analys description soil data from Germany format |||%%%%%%% labels X|Y|Horizon|%pH%Kcal%Pcal%Corg%Ccarb%Nt%Trd Number of DIMENSIONS: 3 −−−−−−−−−−−−−−−−−−−−− − − MIN − − − − MAX − − dim 1 4457986.000000 4459536.000000 X dim 2 5372514.000000 5374314.000000 Y dim 3 0.000000 120.000000 Horizon Type of CATEGORY information: NO CATEGORY −−−−−−−−−−−−−−−−−−−−−−−−−−−−− Number of DOUBLE attributes: 7 −−−−−−−−−−−−−−−−−−−−−−−−−−−− − − MIN − − − − Q1 − − − − MED − − − − Q3 − − − − MAX − − dbl 1 0 5.4 6 6.2 7.7 pH dbl 2 0 0.058 0.1 0.174 1.212 Kcal dbl 3 0 0.009 0.017 0.044 0.415 Pcal dbl 4 0 0.18 0.34 1.09 13.68 Corg dbl 5 0 0 0 0 3.82 Ccarb dbl 6 0 0.031 0.05 0.117 0.744 Nt dbl 7 0.31 1.45 1.55 1.6 1.89 Trd Number of STRING attributes: 0 −−−−−−−−−−−−−−−−−−−−−−−−−−−−− TOTAL SITES COUNTED: 2343

SITES FILENAME: US.places −−−−−−−−−−−−−− Header Information: −−−−−−−−−−−−−−−−−−− name US Places description names of populated places in the United States format ||#@@@%%%%% labels easting|northing|#FIPS@place@state@place_type%population%households%land_area%water_area%population_density time 21 Jul 1995 Number of DIMENSIONS: 2 −−−−−−−−−−−−−−−−−−−−− − − MIN − − − − MAX − − dim 1 −124.610970 −67.012039 easting dim 2 24.562840 48.996136 northing Type of CATEGORY information: CELL_TYPE −−−−−−−−−−−−−−−−−−−−−−−−−−−−−− − − MIN − − − − MAX − − 100124 5686665 FIPS Number of DOUBLE attributes: 5 −−−−−−−−−−−−−−−−−−−−−−−−−−−−− − − MIN − − − − Q1 − − − − MED − − − − Q3 − − − − MAX − − dbl 1 0 437 1371 4605 7.32256e+06 population dbl 2 0 190 583 1905 2.99217e+06 households dbl 3 5 1659 3973 10396 1.96495e+06 land_area dbl 4 0 0 0 76 781914 water_area dbl 5 0 0.191793 0.393334 0.712237 17.692 population_density Number of STRING attributes: 3 −−−−−−−−−−−−−−−−−−−−−−−−−−−−− − MIN_LEN − − MAX_LEN − str 1 3 35 place str 2 2 2 state str 3 3 7 place_type TOTAL SITES COUNTED: 22993 −−

BUGS

If the **−Q** flag is given, the sites are cached in order to determine quartile values; otherwise the sites are read sequentially without caching. Therefore, use of the **−Q** option may result in an out−of−memory error for very large sites files. The algorithm used to find quartile values matches that used by *s.univar*, so if the number of sites is not evenly divisible by 4, the resulting Q1 and Q3 values may not exist in the data, but is an average of the two neighboring values. Likewise, if the number of sites is not evenly divisible by 2, the resulting median will be an average of its neighbors.

SEE ALSO

d.sites, d.siter, d.sites.qual, s.univar

AUTHOR

Bill Brown, UI GMS Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.in.garmin.sh − Import gps data from garmin receiver into GRASS sites file
(GRASS Script)

SYNOPSIS

s.in.garmin.sh
s.in.garmin.sh *−h*
s.in.garmin.sh ***name***=*sitesfile* ***port***=*/dev/gps −v −w −r −t*

DESCRIPTION

s.in.garmin.sh allows to import waypoint, route and track data from a locally connected garmin gps receiver via the *gpstrans* program of Carsten Tschach.

Use at your own risk. This software comes with absolutely no warranty.

No checks are performed for datum, projection and format of data. You must check by yourself that your receiver, gpstrans and GRASS use the same map datum and projection (this means as it is now that you can only use a GRASS database in lat/lon projection and in wgs84 datum).

Parameters:

name=*sitesfile*
    name for new sites file
port=*/dev/gps*
    port garmin receiver is connected to
−v
    verbose output
−w
    upload Waypoints
−r
    upload Routes
−t
    upload Track
−h
    print this message

SEE ALSO

v.in.garmin.sh
gpstrans manual

AUTHOR

Andreas Lange, Andreas.Lange@Rhein−Main.de
gpstrans was written by Carsten Tschach
gpstrans is found at http://www.metalab.unc.edu/pub/Linux/science/cartography/

Last changed: $Date: 2002/06/16 15:29:18 $

NAME

s.in.grid − convert Arc GRIDASCII output to site list.
(GRASS Shell Script)

SYNOPSIS

s.in.grid file

DESCRIPTION

s.in.grid is a *awk* script that reads the output of Arc/INFO's GRIDASCII command and converts it to a GRASS site list.

OPTIONS

Parameters:

file
: Name of the output of GRIDASCII.

EXAMPLE

Typing the following:

> Arc: **GRIDASCII nitrates tmp**
> Arc: **quit**
> ...
> GRASS GRID> **s.in.grid tmp | s.in.ascii nitrates fs=**'|'
> GRASS GRID> **rm tmp**

This procedure exports the GRID file *nitrates* from Arc/INFO and imports it as the GRASS site list *nitrates*.

NOTES

awk may not be able to handle large ASCII grids. In these instances, try using *nawk*, or better yet, *gawk*.

FILES

`$GISBASE/scripts/s.in.grid`

SEE ALSO

s.in.ascii

AUTHOR

James Darrell McCauley, Agricultural Engineering, Purdue University

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.in.mif − Import of MapInfo point data files
(GRASS Sites Data Import Program)

SYNOPSIS

s.in.mif help s.in.mif [−l] in=*name* [**logfile**=*name*] [**out**=*name*]

Flag:

−l
List fields of coverage

Parameters:

in
Name of .MIF/.MID file to be imported
logfile
Name of file where log operations
Default: mif.log
out
Name of sites−list file

AUTHOR

David D. Gray

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.in.shape − Read an ArcView Shapefile with points or multipoint shapes

SYNOPSIS

s.in.shape [**−l**] [**−z**] [**−m**] **in=***name* [**attribute=***attribute category*] [**height=***constant height value*]

DESCRIPTION

The *s.in.shape* module imports point and multipoint themed ESRI shapefiles into a GRASS site_lists.

s.in.shape can be run with command line options as follows:

s.in.shape [**−l**] [**−z**] [**−m**] **in=***name* [**attribute=***attribute category*] [**height=***constant height value*]

Alternatively, an interactive interface can be started:
s.in.shape

In this case, the user will be prompted for parameter values using the standard GRASS user interface described in the manual entry for *parser*.

FEATURES

Grass files created have the name extracted from the basename of the shapefile.

All the fields in the associated dbf file are imported, but those not recognised as INTEGER, DOUBLE OR STRING are simply recorded as INTEGER with a common value of zero. This is a place−holder for possibly improved type support in the future.

A field may be chosen as the attribute field, or if not, a sequential ID field is created.

Multipoint themes are imported with individual points being each given a separate record, but the original record number is preserved in a special field. It is always the first (after the attribute).

If a measure field is defined in the input theme, this is recorded in a special field. It will be the second if present. A module or script using these sites lists is expected to know if this field is present. This fact is not recorded.

An optional height record for 3d points can be stored if present as a third dimension.

FLAGS

−l	List the fields contained in the DBF file associated with the input shapefile, then exit.
−z	Include a height with each record. The height will only be recorded if this flag is set, and then will always be given. If the input data does not contain a height field, a constant value is set, which defaults to 0. The default can be over−ridden with the *height* option − see below.
−m	Include a field in the sites file with the measure field from the input theme. This is only used if the input theme defines this field, **and** the user specifies this flag. Otherwise it is ignored. If present, the measure field will be the second field.

OPTIONS

in=*name*	Name of input shape file. Provide a full path name or the name of a file in the current directory. Any of the full pathname, basename, or prefix only will suffice.
height=*z−value*	In the event you select the **−z** option, **and** there is no in−built height field (zval) in the input file, this can provide a constant height value. Otherwise it is ignored if given.
attribute=*a_name*	Name of the field in the dbf file to use as the attribute in the sites list. Defaults to a sequential index if no attribute field is given, or the given field is a string field.

BUGS AND CAVEAT

The projection must be set in the mapset before input.

Some site maps do not display all markers.

The time stamp facility does not appear to work at present.

SEE ALSO

g.mapset, g.region, g.proj, v.in.shape.

AUTHORS

David D. Gray

Last changed: $Date: 2003/08/20 08:10:13 $

GRASS 5.0 Site Record Format Specification (DRAFT!)

Site files contain records describing punctual information. Records are limited to files containing only characters from the US−ASCII character set. Records are separated by a newline character (ASCII 0x0a). There are three types of records: comment records, header records, and data records. The formats of each these types of records are described in the following sections.

Site Data Record Description

A site record in the GRASS Sites Format is divided into two parts, each with a different field separator. Part 1 contains location in 2 or more dimensions and part 2 optionally contains attribute information for this location. Both types of fields (and thus site records) are variable length.

Part 1 of a Site Record: Location

Part 1 of a site record gives information about location. The field separator in part 1 of the site record is a "pipe" (ASCII 0x7c) character. The last (non−escaped) pipe signifies the end of part 1 (an escaped character is defined as one prefixed by a "backslash" (ASCII 0x5c)). Any additional fields are considered attribute information.

Each field in part 1 indicates a coordinate in some space. There must be at least two fields in part 1: the first describing a geographic easting and the second describing a geographic northing. These may be in either decimal or degrees−minutes−second format.

Additional fields in part 1 are optional but must be stored in decimal format. They should only be used to represent coordinate information about some space (e.g., elevation, time; depending upon how a space is defined).

Part 2 of a Site Record: Attributes

Part 2 contains attribute information for the location given in part 1. The field separator in part 2 of the site record is a "space" character (ASCII 0x20), except when the space character is contained in double quotes (ASCII 0x22). The three types of attributes are: category, decimal, and string. These attributes may be in any order. Each of these attributes have an associated identifier tag defining the type of attribute in a field: # (ASCII 0x23), % (ASCII 0x25), and @ (ASCII 0x40), for category, decimal, and string, respectively. No space character may immediately follow an identifier tag.

Category Attributes

Categories are a special kind of attribute. They are used to represent vector or raster categories when sites are transformed into these different data formats. There may be only one category field per record and it must be prefixed with a "pound" or "number" symbol (#). Categories must be integers.

Decimal Attributes

Decimal attributes include both integers and floating−point numbers. They are prefixed with a "percent" symbol (%). There may be be zero, one, or more decimal attributes in a site record.

String Attributes

String attributes are fields that contain possibly non−numeric information and are prefixed with the "at" or "each" symbol (@). There may be be zero, one, or more string attributes in a site record. String attributes may contain space (ASCII 0x20) characters if the entire attribute, not including the attribute tag (@), is contained within pairs of "double quotes" ("). String attributes may also contain double quotes if they are escaped by prefixing a "backslash" (\).

Default

If no identifier tag is prefixed (i.e., none of #, %, or @), the type of attribute defaults to string.

Header and Comment Record Format

In addition to the data record format, the site file may contain comment lines (records containing a pound symbol, 0x23, in the first column) and header lines, both of which are optional. Header records must precede all data records while comment records may occur anywhere within a sites data file.

There are five types of header records: (1) name, (2) description, (3) timestamp, (4) label, and (5) format.

name
: A name record contains the string "name|" beginning in column 1 and optionally specifies the name of the database file.

description
: A description record contains the string "desc|" beginning in column 1 and optionally describes the database file (metadata).

timestamp
: A timestamp record is special type of metadata that contains the string "time|" beginning in column 1 and optionally gives a time and date associated with the entire sites file. Valid time data strings are described in the documentation for the DateTime library (URL?).

label
: A label record describes what each dimension and attribute field in site data records represent. It contains the string "labels|" beginning in column 1 and optionally contains field descriptions. No special formatting is required since this record is for user convenience only.

format
: A format record describes the format of site data records. It contains the string "form|" beginning in column 1 and a special sample data record beginning in column 6. The special sample data record is a site data record (as describe above) containing only field separators and identifier tags (i.e., all data removed).

All header records are optional. If present in a sites data file, header records must occur in the before any data records in a site file.

Darrell McCauley Last modified on Friday, 25−Oct−96 15:55:23 CDT

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

sites.occur − Description of the site occurrence report produced by *s.menu*.

DESCRIPTION

This is a discussion of the site occurrence report produced by *s.menu*.

SAMPLE OUTPUT

```
                         SITE OCCURRENCE REPORT
                    Yakima Firing Center, Washington

Location:  yakima
Mapset:    sample
Site List: arch_erosion − archeology (34 sites)

Analysis Region:
                    north: 5199975.00
   west:  692975.00                      east:  737975.00
                    south: 5154975.00

============================================================================

Layer: arch_survey          Archeologic Survey Areas (arch_survey)

cells in analysis region:     810000
sites in analysis region:         34

                                                                    degrees
                              cells     %    expected actual   chi     of
Site Characteristics          cover   cover   sites    sites  square freedom
--------------------          ------- ------- -------- ------ -------- -------
(  0) No data                  764310                      11

(  1) Hartmann,phase 1          25821    56.5     19.2      6    9.088      1
(  2) Hartmann,phase 2          11032    24.1      8.2     14    4.084      1
(  3) Wapora,phase 3             8837    19.3      6.6      3    1.945      1
                              ------- ------- -------- ------ -------- -------
      Totals                    45690   100.0     34.0     23   15.117      2

============================================================================
```

EXPLANATION

The "Site Occurrence Report" provides aggregate site information organized by raster map layers.

The report header contains the following information:

Location
: GRASS database location (e.g., yakima),

Mapset
: GRASS mapset (e.g., sample)

Analysis Region
: the rectangular area in which the analysis was run, reflecting the current geographic settings of the user's mapset.

Site List
: name and description of the site list used for the report and the number of sites in the list. See manual entry for *s.menu* for more discussion about these site lists.

The rest of the report is divided into reports for each raster map layers specified by the user. In this SAMPLE OUTPUT, only one map layer (arch_survey) was specified.

Certain information heads up each of these reports:

layer
: identifies the raster map layer along with the map layer TITLE.

cells in analysis region
: the size of the analysis region (in cells).

sites in analysis region
: the number of sites in the analysis region.

For each map layer category which occurs in a cell falling within the boundaries of the analysis region, as well as totals for the map layer as a whole, various statistics are presented:

Site Characteristics
: These are the numbers (i.e., values) and names (i.e., descriptions) for each category (i.e., characteristic) which occurs in the map layer.

cells cover
: the number of cells in the analysis region having the category value (i.e., site characteristic).

% cover
: the predominance of the category within the analysis region.

expected sites
: the number of sites in the analysis region which would be expected to be observed having the category value if all sites were evenly distributed by in the region.

actual sites
: the number of sites in the analysis region which were occurred in cells having the category value.

chi square
: gives the user a statistical test of the hypothesis that 'site distribution is dependent on environmental factors'. For this hypothesis to be true, the chi−square test must fail since we are calculating expected sites according to the hypothesis that 'sites are randomly distributed'.

degrees of freedom
: Number of degrees of freedom for the chi−square test. Each category value will have 1 degree of freedom, and can be tested alone. The entire map layer will have 1 less degree of freedom than the number of categories.

NOTE

Category zero (0) throughout GRASS represents "no data". Therefore, it is reported, but not included in the totals or in the computation of the chi−square statistics.

SEE ALSO

s.db.rim
s.in.ascii
s.menu
s.out.ascii
s.surf.idw
sites.S
sites.format
sites.report

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

sites.report − Description of the machine readable report output by *s.menu*.

DESCRIPTION

This is a description of the machine readable report produced by the *s.menu* program.

SAMPLE REPORT

```
location|yakima|Yakima Firing Center, Washington
mapset|sample
region north|5199975.000000
region south|5154975.000000
region west|692975.000000
region east|737975.000000
resolution|50.000000
site list|sample|sample site list

point|1|728220|5182440|27
point|2|727060|5181710|28
point|3|725500|5184000|29
point|4|719800|5187200|30

layer|1|arch_survey|Archeologic Survey Areas (arch_survey)
layer|2|vegetation|Vegetation Study Plots

cat|1|0|764310|No data
cat|1|1|25821|Hartmann,phase 1
cat|1|2|11032|Hartmann,phase 2
cat|1|3|8837|Wapora,phase 3
cat|2|0|774333|no data
cat|2|1|9299|stiff sagebrush/sandberg bluegrass type (S−1)
cat|2|2|732|big sagebrush/sandberg bluegrass type (S−2)
cat|2|3|397|buckwheat/sandberg bluegrass type (S−3)
cat|2|4|2526|bluebunch wheatgrass/sandberg wheatgrass type (S−4)
cat|2|5|20184|big sagebrush/bluebunch wheatgrass type (L−1)
cat|2|6|82|big sagebrush/needle−and−thread type (L−2)

matrix size|9

matrix|−1|−1|0|−1|1|−1
matrix|−1|0|0|0|1|0
matrix|−1|1|0|1|1|1

data|1|1|0|0|0|0|0|0|0|0|0
data|1|2|1|1|1|1|1|1|1|1|1
data|1|3|1|1|1|1|1|1|1|1|1
data|1|4|1|1|1|1|1|1|1|1|1
data|2|1|0|0|0|0|0|0|0|0|0
```

```
data|2|2|0|0|0|0|0|0|0|0|0
data|2|3|4|4|4|0|0|0|0|0|0
data|2|4|1|5|5|5|5|5|5|5|5
```

EXPLANATION

The machine readable report file consist of lines of text, with fields that are separated by the pipe character |. The first field on a line defines the record type, and the remaining fields are defined according to the record type.

location

This record gives the GRASS database location.
location <name> <full name>

<name>

database location

<full name>

long name for the location

mapset

This record gives the GRASS mapset.
mapset <mapset>

<mapset>

mapset under which the report was run

region

The region records give the mapset region coordinates.
region north <N>
region south <S>
region west <W>
region east <E>

<N>

north for current region (float)

<S>

south for current region (float)

<W>

west for current region (float)

<E>

east for current region (float)

resolution

This record gives the GRASS database resolution in meters.
resolution <res>

<res>

resolution of the database, in meters (float)

point

These records describe the sites that were used in the report. There is one 'point' record for each site.
point <site_ref> <E> <N> <desc>

<site_ref>

a reference number, used by other lines in the report to reference this site

<E>

site easting

<N>

site northing

<desc>
a description of the site (usually an site number)

layer

These records describe the map layers that were used in the report. There is one 'layer' record for each layer.

layer <layer_ref> <name> <full_name>

<layer_ref>
a reference number, used by other lines in the report to reference this layer

<name>
name of the map layer (raster file name)

<full name>
full name (TITLE of raster file)

cat

These records give information about each category in each layer.

cat <layer_ref> <cat> <count> <name>

<layer_ref>
reference number of the layer

<cat>
category number

<count>
number of times this category occurred

<name>
full name of the category

matrix size

This record gives the number of cells in each site.

matrix size <matrix size>

<matrix size>
number of cells per site.

matrix

These records describe the locations of each cell in the site relative to the center of the site.

matrix <ew_disp> <ns_disp> ...

<ew_disp>
east−west displacement from the site

<ns_disp>
north−south displacement from the site

The displacements are in units of 1 cell, which is equal to the resolution of the database. They are paired, and there will be <matrix size> pairs. To compute the coordinates for the cell:
north = site_north − <ns_disp> * resolution
east = site_east + <ew_disp> * resolution

data

This is the site data. There is one 'data' record for each site for each layer. The <cat> field is repeated <matrix size> times.

data <layer_ref> <site_ref> <cat> ...

<layer_ref>
reference number of the layer

<site_ref>
reference number of the site

<cat>
category which occurred

SEE ALSO

s.db.rim
s.in.ascii
s.menu
s.out.ascii
s.surf.idw
sites.S
sites.format
sites.occur

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

sites.S − Description of the GRASS *s.menu* "S" interface option.

DESCRIPTION

This describes the GRASS *s.menu* "S" statistical package interface option. See also the manual entry for *s.menu*.

SPECIAL ADJUSTMENT

Due to the fact that "S" references all subscripts beginning with 1, and the GIS data begins with 0, it was necessary to add 1 to the category numbers. Therefore, category value 0 ("no data") becomes category value 1, category 1 becomes category 2, etc.

S DATA STRUCTURES

Below are descriptions of the data structures created by the interface. As a general comment, within "S", simply typing the data structure name will display the values contained in the structure. You may also find it interesting to display the data structures using the "S" function *dput()*.

cat.histo, (cat.#.histo)

This structure contains the histogram for the categories in each map layer. The histogram contains the number of cells of each category which occur in the user's geographic region (see wind.n, et al., below). It is a 2−dimensional integer array (matrix). The first subscript references the map layer. The second subscript references the category value.

Examples:

cat.histo[2,5] is the cell count for layer 2, category 5.
cat.histo[2,] is the full histogram for layer 2.

Note: since more than one layer may occur in the data, and the number of categories in each layer varies, it was necessary to create this structure with sufficient dimension to hold the largest category value for all layers. Histogram data for categories which do not occur are set to NA ("S" notation for "no data"). However, there is also an individual histogram structure for each layer: cat.1.histo, cat.2.histo, etc. These are simple vectors.

cat.name

This structure contains the names for the categories in each map layer. It is a 2−dimensional character array (matrix). The first subscript references the map layer. The second subscript references the category.
Examples:

cat.name[2,5] is the name of category 5 for layer 2.
cat.name[2,] are all category names for layer 2.

Note: since more than one layer may occur in the data, and the number of categories in each layer varies, it was necessary to create this structure with sufficient dimension to hold the largest category value for all layers. Names for categories which do not occur are set to ''. However, there is also an individual category name structure for each layer: cat.1.name, cat.2.name, etc. These are simple vectors.

site.data

This structure contains the data for each site. It is a 3−dimensional integer array. The first subscript references the categories which occurred at the site. The second subscript references the site. The third subscript references the layer.

Examples:

site.data[,5,3] is the data for site 5 in layer 3.
site.data[,2,] is the data for site 2 in all layers.
site.data[,,1] is all site data for layer 1.

Note: the size of the first dimension will be the number of cells in a site, the size of the second dimension will be the number of sites, the size of the third dimension will be the number of layers.

site.mode

Since "S" does not provide a statistical mode function, this structure contains the most frequently occurring category for each site in each layer. It is a 2−dimensional integer array (matrix). The first subscript references the site. The second subscript references the layer.

Examples:

site.mode[5,3] is the mode for site 5 in layer 3.
site.mode[2,] are the modes for site 2 in all layers.
site.mode[,1] are all site modes for layer 1.

Note: this 'mode' is not the strict definition of the mode. Since category 0 (which in "S" is category 1) represents "no data" in the GIS databases, it was excluded from the mode calculations (essentially as if it had been NA). For example, if the data for a site are 1 1 1 1 2 2 3 2 3 1 1, the mode will be 2.

layer.name

This structure contains the raster map layer names. It is a 2−dimensional array (i.e., a matrix). The first subscript references the map layer. The second subscript selects either the map layer name, or the map layer TITLE.

Examples:

layer.name[3,1] is the name of layer 3.
layer.name[3] is also the name of layer 3.
layer.name[2] is the name of layer 2.
layer.name[3,2] is the TITLE for layer 3.

location

This is a simple character vector giving the GRASS location from which the data was extracted.

mapset

This is a simple character vector giving the GRASS mapset.

nlayers

This is a simple integer giving the number of map layers.

nsites

This is a simple integer giving the number of sites.

site.e

This is a simple integer vector giving the geographic easting for each site.

site.n

This is a simple integer vector giving the geographic northing for each site.

site.name

This is a simple character vector giving the description for each site.

sitelist

This is a simple character vector giving the name and description of the site lists file from which the sites were taken.

wind.n, wind.s, wind.w, wind.e

These are simple real numbers giving the north, south, west, east of the mapset's current geographic region.

wind.res
This is a simple real number giving the GRASS database resolution (in meters).

S MACROS

You may find the following "S" macros helpful when referencing the 'site.data' and 'site.mode' structures, since they allow you to specify parameters:

```
MACRO site.data(site, layer)
({
        site.data[ , site, layer]
})
END


MACRO site.mode(site, layer)
({
        site.mode[site, layer]
})
END
```

MACRO USAGE

To select the site.data for all sites for layer 2:

?site.data(layer=2)

To select the site.data for site 4 in all layers:

?site.data(site=4)

To select the site.mode for site 10 in layer 1:

?site.mode(layer=1,site=10)

SORRY, BUT ...

These macros are not provided by the interface. To use these macros, you will have to type them into a text file and then, from "S", issue the command:

define("<file>")

SEE ALSO

s.db.rim
s.in.ascii
s.menu
s.out.ascii
s.surf.idw
sites.format
sites.occur
sites.report

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.kcv − Randomly partition sites into test/train sets. *(GRASS Sites Program)*

SYNOPSIS

s.kcv
s.kcv help
s.kcv [**−dq**] **k=value sites**=*name*

DESCRIPTION

s.kcv randomly divides a sites lists into *k* sets of test/train data (for **k**−fold **c**ross **v**alidation). Test partitions are mutually exclusive. That is, a site will appear in only one test partition and *k−1* training partitions. The program generates a random point using the selected random number generator and then finds the closest site to it. This site is removed from the candidate list (meaning that it will not be selected for any other test set) and saved in the first test partition file. This is repeated until enough points have been selected for the test partition. The number of sites chosen for test partitions depends upon the number of sites available and the number of partitions chosen (this number is made as consistent as possible while ensuring that all sites will be chosen for testing). This process of filling up a test partition is done *k* times.

Flags:

−d Use *drand48()* (default is *rand()).*
−q Run quietly. Don't report progress.

Parameters:

k=*value* Positive integer value indicating the number of partitions.
sites=*name* Name of a sites file to store random points in.

Test/train pairs are saved as sites list using *name* as a basename. Test sites are saved in *name*−test.*i* while training sites are saved in *name*−train.*i,* where *i* ranges from zero to *k.*

NOTES

Existing files are silently overwritten.

An ideal random sites generator will follow a Poisson dis only be as random as the original sites. This program simply divides sites up in a random manner.

Be warned that random number generation occurs over the intervals defined by the current region.

This program may not work properly with Lat−long data.

SEE ALSO

s.random and *g.region*

BUGS

Please send all bug fixes and comments to the author or the grass development team.
`http://www.geog.uni−hannover.de/grass/`

AUTHOR

James Darrell McCauley <darrell@mccauley−usa.com>,
when he was at: Agricultural Engineering Purdue University

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.kernel − Generates a raster density map from sites data using a moving 2D isotropic Gaussian kernel
(GRASS Sites Program)

SYNOPSIS

s.kernel
s.kernel help
s.kernel sitefile=*name* **output=***name* [**stddeviation=***value*] [**resolution=***value*]

DESCRIPTION

s.kernel generates a raster density map from sites data using a moving 2D isotropic Gaussian kernel.

OPTIONS

Parameters:

sitefile=*name*
  Name of sites list.
output=*name*
  output raster map
stddeviation
  stddeviation in map units
  default: 1
resolution
  resolution of the output raster map
  default: 25.0

NOTES

s.kernel uses moving 2D isotropic Gaussian kernel.

LIMITATIONS

The modules only considers the presence of sites, but not (yet) any attribute values.

SEE ALSO

s.surf.rst

BUGS

Please send all bug fixes and comments to the author or the grass development team.
`http://grass.itc.it/`

AUTHOR

Stefano Menegon, ITC−irst, Trento, Italy

Last changed: $Date: 2003/07/04 08:35:34 $

NAME

s.label − makes a label−file from a GRASS sites list.
(GRASS Sites Program)

SYNOPSIS

s.label
s.label help
s.label [−r] sites=name label=name [xoffset=value] [yoffset=value] [reference=name] [font=name] [ps_font=name] [size=value] [color=name] [width=value] [hcolor=name] [hwidth=value] [background=name] [border=name] [opaque=name]

FLAG

−r make sites from label file

DESCRIPTION

s.label makes a label−file from a GRASS sites list.

PARAMETERS

sites Name of a site list

label Name of a paint−label file

xoffset Offset label in x−direction
default: 0

yoffset Offset label in y−direction
default: 0

reference Reference position
options: center,left,right,upper,lower
default: center

font Font
options: standard,cyrilc,gothgbt,gothgrt,gothitt,greekc, greekcs, grrekp,greeks,italicc,italiccs,italict, romanc,romancs,romand,romanp,romans, scriptc,scripts
default: standard

ps_font PostScript font (for use with ps.map); N.B. screen display will be in standard font!
options: Helvetica,Helvetica−Bold,Helvetica−Oblique, Helvetica−BoldOblique,Times−Roman,Times−Bold, Times−Italic,Times−BoldItalic,Courier,Courier−Bold, Courier−BoldOblique,Symbol,AvantGarde−Book, AvantGarde−BookOblique,AvantGarde−Demi, AvantGarde−DemiOblique,Bookman−Demi, Bookman−DemiItalic, Bookman−Light,Bookman−LightItalic, Helvetica−Narrow,Helvetica−Narrow−Bold, Helvetica−Narrow−Oblique, Helvetica−Narrow−BoldOblique,NewCenturySchlbk−Roman, NewCenturySchlbk−Italic,NewCenturySchlbk−Bold, NewCenturySchlbk−BoldItalic,Palatino−Roman, Palatino−Italic,Palatino−BoldItalic, ZapfChancery−MediumItalic

size Label size (in map−units)
options: 1−1000
default: 100

color Text color
options: aqua,black,blue,brown,cyan,gray,green,grey,indigo, magenta, orange,purple,red,violet,white,yellow
default: black

width Line width of text (only for p.map output)
options: 1−100
default: 1

hcolor Highlight color for text (only for p.map output)
options: none,aqua,black,blue,brown,cyan,gray,green,grey, indigo,magenta, orange,purple,red,violet,white,yellow
default: none

hwidth Line width of highlight color (only for p.map output)
options: 0−100
default: 0

background Background color
options: none,aqua,black,blue,brown,cyan,gray,green,grey, indigo,magenta, orange,purple,red,violet,white,yellow
default: none

border Border color
options: none,aqua,black,blue,brown,cyan,gray,green,grey, indigo,magenta, orange,purple,red,violet,white,yellow
default: none

opaque Opaque to vector (only relevant if background color is selected)
options: yes,no
default: yes

AUTHOR

Philip Verhagen
S t i c h t i n g R A A P
Regionaal Archeologisch Archiverings Projekt
adress: Plantage Muidergracht 14

1018 TV Amsterdam
THE NETHERLANDS
e−mail: motte@xs4all.nl

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

slide.show.sh − Displays a series of raster/vector map layers existing in the user's current mapset search path on the graphics monitor.
(GRASS Shell Script)

SYNOPSIS

slide.show.sh [**across=***value*] [**down=***value*] [**prefix=***name*] [**mapsets=***list*]

DESCRIPTION

slide.show.sh is a UNIX Bourne shell macro which clears the entire screen, creates a series of display frames on the graphics monitor, and displays in slideshow format each of the raster/vector map layers listed in the user−specified *mapsets*. This is a shell script example which makes extensive use of GRASS and UNIX commands. Users are encouraged to examine this macro and develop similar on−line demos using their own data files.

OPTIONS

Parameters:

across=*value*
: The number of display frames across the graphics monitor screen to be used for map display.
Default: 4

down=*value*
: The number of display frames down the graphics monitor screen to be used for map display.
Default: 3

prefix=*name*
: Specify character(s) to view selected maps only.
default: * (show all maps)

mapsets=*list*
: The names of the mapsets under the user's current location whose raster map layers are to be displayed, separated by commas.
Default: All mapsets listed in the user's current mapset search path.

FILES

See the file **`$GISBASE/scripts/slide.show.sh`**.

SEE ALSO

d.display
d.erase
d.text
g.mapsets
3d.view.sh
grass.logo.sh
show.fonts.sh

AUTHOR

James Westervelt, U.S.Army Construction Engineering Research Laboratory
Vector update, fixes: Markus Neteler

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.mask − Filter a sites list with a raster mask
(GRASS Sites Program)

SYNOPSIS

s.mask
s.mask help
s.mask [**−r**] **input=***name* **output=***name* **raster=***name*

DESCRIPTION

Use *s.mask* to filter a sites list with a raster mask. The raster can be any available raster. Sites are filtered first by the region, then by whether the site falls in a cell with a non−NULL value. The region used for filtering is the current region by default. However, the user may specify that the raster mask's region be used instead. If the raster mask's region is used, both it's extents and resolution are used in the filtering process.

Upon succesful completion, *s.mask* will print out the total number of sites examined and the number of sites output.

OPTIONS

Flags:

−r
: Use the raster's extents and cell resolution, rather than the current region settings.

Parameters:

input=*name*
: The name of the existing sites list to filter.

output=*name*
: The name of the sites list to output.

raster=*name*
: The name of the raster to use as the mask.

SEE ALSO

d.sites parser

AUTHOR

Eric G. Miller

Inspired by code supplied by Helen Mitasova and Jaro Hofierka.

Last changed: $Date: 2002/03/16 22:27:57 $

NAME

s.medp − Median polish for a GRASS sites list.
(GRASS Sites Program)

SYNOPSIS

s.medp
s.medp help
s.medp [**−aeqs**] **sites=***name* **vect=***name* **output=***name* [**report=***name*] [**thresh=***n*]

DESCRIPTION

s.medp performs median polish on an existing sites list. The sites are overlayed onto a grid (**vect=***name*) and each site is associated with the closest node. Therefore, sites do not necessarily have to be oriented on a grid. Each node may be associated with zero, one, or more sites and the grid may be rotated. Also, the length of each box does not have to be equal to width of each box defining the grid.

Once sites are associated with nodes on a *p* by *q* grid, *p+q+1* extra storage locations are created (initialized to zero) to store *all*, *row*, and *column* effects. The median of each row is removed from the data and added to the extra *p* cells. Then, medians are removed from the data as well as from the *p* cells containing *row* effects. The medians of data in this pass are stored in the extra *q* cells (*column* effects) and the median of the row effects is stored in the extra (*p+1, q+1*) cell (*all* effect). This is repeated until each successive iteration leaves each site unchanged (within a tolerance, described by [**thresh=***n*]).

OPTIONS

Flags:

−a
: Use all sites found in the named *sites* file, rather than limiting output to sites falling within the current geographic region.

−e
: Store row, column, and all effects in output file.

−s
: Write results to a sites list file (default is to write points in a binary vector file).

−q
: Quiet. Cut out the chatter.

Parameters:

sites=*name*
: Name of an existing sites file.

vect=*name*
: Name of the grid file (binary vector file).

output=*name*
: Name of the output file (binary vector file or sites list).

report=*name*
: Name of an ASCII file which shows original and residual data in tabular form.

thresh=*n*
: Threshold to determine when convergence of median polish is obtained. (default = 1).

s.medp can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies the name of an existing *sites* list file, name for a *vect* file, and name of an *output* file using the form

s.medp [**−aeqs**] **sites**=*name* **vect**=*name* **output**=*name* [**report**=*name*] [**thresh**=*n*]

Alternately, the user can simply type **s.medp** on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS *parser* interface.

NOTES

When using the **report** option, the tabular report may be more than 128 columns wide. Therefore, it may be useful to add a *TeX* or *PostScript* wrapper before printing (e.g., **enscript −h −r −fCourier−Bold5 −p report.ps report**) so that the page is in landscape orientation and a very small font is used.

The input vector file defining the grid should be large enough so that the extreme east−west and north−south nodes (for non−rotated grids) can be used to store row and column effects. Otherwise, when residuals *and* effects are written to the output file, residuals may share the same spatial location as effects when written to a vector output file.

Currently, nothing is done with the residuals, but I would like implement some sort of analysis of these in the future (suggestions?). Kriging using these residuals as new data set is being planned.

SEE ALSO

v.mkgrid
v.transform
parser

Cressie, Noel A. C. (1991). *Statistics for Spatial Data*, New York, NY: John Wiley & Sons. pp. 186−187.

BUGS

This should still be consider beta−release software. Although I have repeated results shown in Cressie (1991), I would greatly appreciate someone testing this further.

The **−s** flag is yet to be implemented.

This should probably be re−written to use points in a vector file as input (thus creating *v.medp*).

Ideally, I would also like to output three data files for plotting by a graphing program (e.g., *gnuplot*), but I never got around to doing this. The classic plot is to show three surfaces: data = trend + residuals. The trend can be obtained by extrapolation and/or interpolation.

Extrapolating and interpolating using the row, column, and all effects to create a raster file would be a nice feature to include in the future. Instead, I am considering retrieving the effects from the output file and creating the surface with separate program.

Please send all bug fixes and comments to the author.

AUTHOR

James Darrell McCauley, Agricultural Engineering Purdue University

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.menu − Accesses and manipulates GRASS site location data.
(GRASS Sites Program)

SYNOPSIS

s.menu

OVERVIEW

The *s.menu* program provides the user with the capability of interfacing site location data with the geographic data in raster map layers. Two types of spatial analysis reports on sites can be generated, and an interface to the "S" statistical package is provided.

DESCRIPTION

The *s.menu* program is an interface to functions that allow the user to manipulate GRASS site "lists". A site list is a list of eastings and northings describing the location of some point feature. It can also contain a category value and category label for each site location. The program is interactive. After typing **s.menu** on the command line, the user selects site functions from a menu.

The main menu is shown below:

```
SITES MAIN MENU    (current list: no sites)

LOCATION: spearfish    REGION 4928000.00(N)  4914000.00(S) 100.00 (RES)
MAPSET:   PERMANENT            609000.00(E)   590000.00(W) 100.00 (RES)
MASK:  none

Please select one of the following

   1  Read an existing site list
   2  Mask current site list
   3  Save the current site list in your mapset

   4  Check site list for duplicate sites
   5  Edit site list using a UNIX editor

   6  Convert site list to raster file (0/1)
   7  Convert site list to raster file (frequency of occurrence)

   8  Run reports on the current site list

stop  Leave the s.menu program
```

At the top of the menu is general information about the user's current MAPSET, LOCATION, etc. Note the above message in parentheses "current list: no sites." This message will vary depending on the status of the list. For example, after the user reads the existing site list file *arch_sites*, the message would read (given the geographic region indicated): "current list: 25 sites, 24 in current region."

1. Read an existing site list
   This option will copy an existing site list file into the current site list within the sites server. Existing site lists are stored under a GRASS data base and are pulled into the *s.menu* server via this option. Other sites menu functions operate only on the current site list file in the server −− you must therefore "read an existing site list" file BEFORE performing any of the other sites menu functions.
   Note: Site lists can be created and placed into a GRASS data base using *s.menu* option 5 (edit) followed by option 3 (save). However, the user can also create site lists using other methods or programs. One simple way to do this is to create a site list file in the appropriate format using any text editor (e.g., "vi"), and to put this site list file under the *site_lists* directory under the user's current GRASS mapset (i.e., under $LOCATION/site_lists). The user can do this either inside of GRASS or outside of GRASS. Alternately, the user can run other GRASS programs which format their output as a GRASS site list file (*r.random*, *s.db.rim*, *v.mkquads*, *v.to.sites*), or the user can use UNIX programs like *awk* and *sed* to format other GRASS programs' output in the form of a site list file (*d.what.rast*, *d.what.vect*).

2. Mask current site list
   The site list can be reduced to a subset that includes only sites which fall in specific categories within a specified raster map layer. The user will be asked to specify the name of a raster map layer to form the basis for the mask, and will then be allowed to specify categories from this raster map that will limit the site list. As with *r.mask*, the category values selected designate the areas of the map in which information will remain. Areas assigned category values *not* selected will be re−assigned to category value "0" ("no data").
   Note: This masking operation is performed only against the site list itself and not against other raster map layers. If the user wishes to analyze masked raster map layers, a mask should be create using the *r.mask* program.

3. Save the current site list in your mapset
   The current site list can be stored permanently in your current mapset with this option. You will be asked to name the saved site list and to provide a short description of it. Saved site lists can be retrieved (option 1) during later runs of *s.menu*. Once saved, these site list files can be used with other GRASS programs, like *d.sites*, *d.points*, *d.icons*, *p.icons*, *s.surf.idw*, and *s.db.rim*.
   Note: Saved lists will be removed if the GRASS mapset
   under which they are stored is removed.

4. Check site list for duplicate sites
   It is not desirable that a site list contain multiple references to the same site. This option attempts to recognize duplicate sites. Duplicates are displayed to the user and can be removed automatically if the user desires. Duplicates can also be removed by hand using option 5
   (edit).

5. Edit site list using a UNIX editor
   The user can modify the current site list or create a new site list by hand using a UNIX editor. You will be asked to specify the text editor you prefer to use. You should exercise some care if you select this option. Lines in the site list which have invalid formats will be (silently) ignored by *s.menu*. See the GRASS manual entry *sites.format* for a description of the site list format.
   Note: This option will only modify the site list copied into the server. It does not modify the original

site list stored under a GRASS mapset. If you wish to modify a stored site list file, you will have to combine options 1 (read), 5 (edit) and 3 (save).

6. Convert site list file to raster file (0/1)
   You can create a raster data representation of the site list in your GRASS mapset.
   Once created, this raster map layer can be used with other raster map layers in further analyses.
   Allowing the user to create a raster map layer of sites opens up the full analysis capabilities for site data that are available for raster map layers within GRASS.
   You have the option of specifying the number of cells to represent a site. The minimum is one cell per site. The alternatives are squares around the site: 3x3, 5x5, 7x7, etc.
   The number of categories present in the new raster map layers will depend on the format of your site list file (see *sites.format*). You can create a non−binary raster map layer representation of your site list by creating the site list in the format:
   "E|N|" *"#n label"*
   where **E** is the Easting, **N** is the Northing, and *#n label* is the description field. The description field consists of a pound sign # followed by the category value *n* to be associated with the site's cell location, and the category label *label* for *n*.
   If the user does not include a description field starting with
   #n
   beside the Easting and Northing on *every line* in the sites list file, a binary raster map layer will be created instead. In the binary raster file, each site will be represented as the category value 1. Non−site cells will be coded as category value 0.

   Note that only sites within the current geographic region will be considered. However, if the size of the sites is more than one cell (3x3, 5x5, etc.) and a site lies near an edge of the geographic region, some of the cells for the site may fall outside the geographic region. These cells will not appear in the raster map layer, and the site will no longer be 3x3 or 5x5 but will be clipped to fit the geographic region.

7. Convert site list file to raster file (frequency of occurrence)
   You can also create a frequency of occurrence raster map layer representation of the site list file.
   The raster category values will be coded as the number of sites that fall within the cell.
   In this function, you do NOT have the option of specifying the number of cells to represent a site.

8. Run reports on the current site list
   The current list of sites is passed to the report server after removing sites which do not fall within the geographic region of the user's current GRASS mapset (see *g.region*). The user then selects the names of one or more raster map layers for analysis. Data at (or near) the sites extracted from these raster map layers form the basis for all reports.
   The user specifies the 'size' of a site in cells. The 'size' may be specified as a single cell, or as a 3x3 square around the site, or 5x5, or 7x7, etc (where the size is an odd integer).

   The following menu of reports is then presented:

```
SITES REPORT MENU

Please select one of the following

   1  Site characteristics report
   2  Site occurrence report

   3  Convert data to S input format
```

```
      4  Produce machine−readable data file

   stop  return to SITES MAIN MENU
```

1. Site characteristics report
   This report provides geographic information about each site.
   Each site is identified by description and locational information. The 'description' is an identification of the site. The site location is an easting and a northing. (The location does not denote the extent of the site, since, for example, an archeologic site which takes up two hectares would be represented as a single point).
   The information reported for each site is displayed by raster map layer, and, within each map layer, gives the categories (i.e., characteristics) that occurred at the site (as well as a count of the number of cells in each category).
   This can easily generate a massive amount of information, which is difficult to handle or digest quickly. Therefore, option 2 produces a synopsis of the information.

2. Site occurrence report
   This report provides aggregate site characteristic information organized by raster map layer. The report produces chi−square statistics for each raster map layer, measuring number of expected sites (assuming a random distribution of sites) against actual site occurrence. The site characteristic is the most frequently occurring cell category in the site (i.e., the statistical mode). See the GRASS manual entry for *sites.occur* for details on this report.

3. Convert data to S input format
   This function converts the GRASS data extracted for the sites into a form usable by the S statistical package. The user provides a file to contain the information. Once the file is written, the user must exit *s.menu*, run S on an S data base, and issue the S command

   source *file*

   to bring the data into the S data base. (Of course, *file* would be the name of the actual file supplied by the user.) See manual entry *sites.S* for an explanation of the S data structures created by this interface.

4. Produce machine−readable data file
   This option provides a mechanism for the user to write his/her own reports. The data is written into a user−specified file as a text file, which has a format readable by UNIX utilities (e.g., *awk*) or user−written programs. See GRASS manual entry *sites.report* for details on this format.

FILES

$LOCATION/site_list/<file >

SEE ALSO

d.icons	*d.graph*	*d.points*
d.sites	*p.icons*	*r.random*
d.what.rast	*d.what.vect*	*r.what*

s.in.ascii	*s.out.ascii*	*s.db.rim*
s.surf.idw	*s.surf.rst*	*v.db.rim*
v.mkquads	*v.to.sites*	*sites.format*
sites.report	*sites.occur*	*sites.S*

AUTHORS

Michael Shapiro, U.S.Army Construction Engineering Research Laboratory

James Farley, Arkansas Archeological Survey, University of Arkansas
contributed the frequency of occurrence sites to cell function

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.normal − tests for normality for sites.
(GRASS Sites Program)

SYNOPSIS

s.normal
s.normal help
s.normal [**−q**] **sites=***name* [**tests=***range*[,*range*,...]] **[field=***value***]**

DESCRIPTION

s.normal computes tests of normality on a site list.

OPTIONS

Flag:

−q
Quiet. Cut out the chatter when reading site list.

Parameters:

sites=*name*
Name of sites list.
tests=*range[,range,...]]*
Tests of normality requested (see below).
field
Number of z−field attribute to use for calculation
default: 1

NOTES

The tests that *s.normal* performs are indexed below. The tests that are performed are specified by giving an index, ranges of indices, or multiple thereof.

1. Sample skewness and kurtosis
2. Geary's a−statistic and an approximate normal transformation
3. Extreme normal deviates
4. D'Agostino's D−statistic

5. Modified Kuiper V−statistic
6. Modified Watson U^2−statistic
7. Durbin's Exact Test (modified Kolmogorov)
8. Modified Anderson−Darling statistic
9. Modified Cramer−Von Mises W^2−statistic
10. Kolmogorov−Smirnov D−statistic (modified for normality testing)
11. Chi−Square test statistic (equal probability classes) and the number of degrees of freedom
12. Shapiro−Wilk W Test
13. Weisberg−Binghams W'' (similar to Shapiro−Francia's W')
14. Royston's extension of W for large samples
15. Kotz Separate−Families Test for Lognormality vs. Normality

EXAMPLE

s.normal sites=*soils* **tests=***1−3,14*

computes the sample skewness and kurtosis, Geary's a−statistic and an approximate normal transformation, extreme normal deviates, and Royston's W for the *soils* site list.

SEE ALSO

s.univar

BUGS

Please send all bug fixes and comments to the author or the grass development team.
`http://www.geog.uni−hannover.de/grass/`

AUTHOR

James Darrell McCauley <darrell@mccauley−usa.com>,
when he was at: Agricultural Engineering Purdue University

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.out.ascii − Converts a GRASS site list file into an ASCII listing of site locations and their descriptions. *(GRASS Sites Program)*

SYNOPSIS

s.out.ascii
s.out.ascii help
s.out.ascii [**−adi**] **sites=***name* [**fs=**character|space|tab]

DESCRIPTION

s.out.ascii converts an existing site list file (**sites=***name*) into an ASCII listing of site locations and (optionally) their category labels, in a format suitable for input to other programs (e.g., *d.points*, *m.u2ll*, etc.).

Each line of output consists of the easting, northing, and category label for a site listed in the named **sites** file. The **fs=***name* option (where *name* is either a character, a space, or a tab) can be used to place a particular field separator between these three output fields. This is useful when output is to be manipulated by other programs, like *awk* or *sed*.

The GRASS program *s.in.ascii* can be used to perform the reverse function, converting a UNIX file containing eastings, northings, and category labels associated with site locations into GRASS site list file format.

OPTIONS

Flags:

−a
: Output all sites found in the named **sites** file, rather than limiting output to sites falling within the current geographic region.

−d
: Include site descriptions (category labels) in the output.

−i
: Include site attribute identifiers in the output

Parameters:

sites=*name*
: Name of an existing site list file.

fs=*character|space|tab*

The field separator to be placed between the easting, northing, and (optionally) category label on each line of output. The field separator can be a character, a space, or a tab.
Default: space

s.out.ascii can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies the name of an existing site list file and (optionally) a value for **fs**, using the form

s.out.ascii [**−ad**] **sites=***name* [**fs=**character|space|tab]

where *name* is the name of an existing site list file to be converted to a brief ASCII listing, and **fs** is the field separator to be placed between output fields. The user can also the **−a** and **−d** options to use all sites in the named **sites** file and to include site descriptions in the output.

Alternately, the user can simply type **s.out.ascii** on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS parser interface.

NOTES

To output GRASS data in x,y,z ASCII style, the s.out.ascii may be used:

s.out.ascii −d sites=sitesmap

Resulting to:

```
3566393.75 5773293.75 168.3
3566818.75 5773293.75 158.8
3568843.75 5773293.75 114.3
3568981.25 5773293.75 117.5
[...]
```

To export raster maps into this x,y,z format, the raster map can be converted to sites format using *r.to.sites* and *s.out.ascii* subsequently as described above.

The output from *s.out.ascii* may be placed into a file by using the UNIX redirection mechanism; e.g.:

s.out.ascii sites=archsites > out.file

s.out.ascii output may also be redirected into other programs; e.g.:

s.out.ascii sites=archsites | d.points **color=**red **size=**10 **type=**diamond

SEE ALSO

d.points
d.sites
m.ll2u
m.u2ll
s.in.ascii
r.to.sites
parser

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.out.e00 − Write an Arc−Info point coverage in e00 format

SYNOPSIS

s.out.e00
s.out.e00 help
s.out.e00 input=*name* [**output=***name*]

DESCRIPTION

The *s.out.e00* program is designed to create an ESRI Arc/Info e00 ascii file from a Grass site file.

s.out.e00 will be run non−interactively if the user specifies program arguments on the command line, using the form:

s.out.e00 input=*name* [**output=***name*]

Alternately, the user can simply type:

s.out.e00

on the command line without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS user interface described in the manual entry for *parser*.

FEATURES

s.out.e00 attempts to create an Arc/Info export file (.e00) for points coverage. The attribut of each Grass site will be the Arc/Info coverage−ID. If there is no attribute, the coverage−ID will be the same as the coverage−# (numbered from 1 to the number of sites).

OPTIONS

s.out.e00 requires the user to enter the following information:

Parameters:

output=*name*
: Name of output file. If no name is given, *s.out.e00* will write on the standard output.

BUGS AND CAVEAT

Since e00 is NOT a public format, this program is mainly based on analysis of existing files.

NOTES

SEE ALSO

m.in.e00, *v.out.e00*, *v.in.shape*.

AUTHOR

Michel J. Wurtz, CEREG,

Ecole Nationale du Genie de l'Eau et de l'Environnement.

mw@engees.u−strasbg.fr

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.perturb − Random location perturbations of GRASS sites.
(GRASS Sites Program)

SYNOPSIS

s.perturb
s.perturb help
s.perturb [**−q**] **input=***name* **output=***name* **distribution=**[*uniform*|*normal*] **parameters=***value*,[*value*]

DESCRIPTION

s.perturb reads a site list and writes the same list but *perturbs* the eastings and northings by adding either a uniform or normal delta value. Perturbation means that a variating spatial deviation is added to the coordinates.

OPTIONS

Flags:

−q
Quiet. Cut out the chatter.

Parameters:

input=*name*
Name of an existing sites file.
output=*name*
Name of output sites file.
distribution=*[uniform|normal]*
Distribution of perturbation.
parameters=*value[,value]*
Parameter(s) of distribution. If the distribution is uniform, only one parameter, the maximum, is needed. For a normal distribution, two parameters, the mean and standard deviation, are required.

NOTES

The uniform distribution is always centered about zero. The associated parameter is constrained to be positive and specifies the maximum of the distribution; the minimum is the negation of that parameter.

Usually, the mean (first parameter) of the normal distribution is zero (i.e., the distribution is centered at zero). The standard deviation (second parameter) is naturally constrained to be positive.

Output sites are not guaranteed to be contained within the current geographic region.

SEE ALSO

g.region
s.random
s.univar
s.kcv

BUGS

Please send all bug fixes and comments to the author or the GRASS development team.

AUTHOR

James Darrell McCauley <darrell@mccauley−usa.com>,
when he was at: Agricultural Engineering Purdue University

Random number generators originally written in FORTRAN by Wes Peterson and translated to C using *f2c*

Last changed: $Date: 2003/03/16 12:25:19 $

NAME

split.sh − Divides the graphics monitor into two frames and then displays two maps in these frames.
(GRASS Shell Script)

SYNOPSIS

split.sh mapname mapname [**cmd=***GRASS_command*] [**cmd2=***GRASS_command*] [**view=horiz**]

DESCRIPTION

split.sh is a Bourne shell (*sh*) script that clears the entire graphics screen and divides it into two display frames. Map layers are then displayed in each of the two frames. This command is very useful for visually comparing maps (raster, vector, and 3−d views) and can be used by other GRASS shell macros. It is also useful for creating demos. Program parameters are given below.

OPTIONS

Parameters:

view=horiz
: The graphics screen can be split either horizontally or vertically. The default view splits the screen into two frames, one on the left and one on the right (a vertical split). Some maps (3−d views) are better represented with more width then height (horizontal split). The first map name listed on the command line will be displayed in the top or left window (depending on whether the screen was split horizontally or vertically), and The second map will be displayed in the bottom or right window.

***cmd**=GRASS_command*
: The GRASS command used to display the named *mapname*s. If no command is specified by the user, *d.rast* is used by default. However, any GRASS display command (e.g., *d.3d*, *d.vect*, etc...) can be entered.

***cmd2**=GRASS_command*
: This command will be used to display map data in the second frame only.
If the user fails to specify the values of both *cmd* and *cmd2*, *split.sh* will use the default command (*d.rast*) to display user−specified map layer names in both frames. If the user specifies only the value of *cmd* on the command line, then that command will be executed for both frames. If the user specifies the values of both *cmd* and *cmd2* on the command line, the *cmd* command will be executed in frame 1 and the *cmd2* command will be executed in frame 2.

EXAMPLES

split.sh soils vegcover

split.sh soils **cmd2=d.legend** "soils color=red"

split.sh elevation vegcover **cmd=**d.3d **view=**horiz

NOTES

split.sh leaves the frame that the last map was drawn in as the active frame. The order in which the options (*cmd*, *cmd2*, *view*) are placed on the command line doesn't matter, but the order is important for the map names.

FILES

This program is simply a shell script. Users are encouraged to make their own shell scripts using similar techniques. See **`$GISBASE/scripts/split.sh`**.

SEE ALSO

d.3d
d.frame
d.rast
d.sites
d.vect

AUTHOR

Michael Higgins, U.S.Army Construction Engineering Research Laboratory

Last changed: $Date: 2003/03/16 12:25:19 $

NAME

s.probplot − Normal probability plot of a GRASS site list.
(GRASS Sites Program)

SYNOPSIS

s.probplot
s.probplot help
s.probplot [**−alq**] **sites=***name* **width=***value* [**graph=***name*]

DESCRIPTION

s.probplot does normal or lognormal probability plots of site values.

OPTIONS

Flags:

−a
Use all sites found in the named *sites* file, rather than limiting output to sites falling within the current geographic region.
−l
Lognormal probability plot instead of normal.
−q
Quiet. Cut out the chatter.

Parameters:

sites=*name*
Name of an existing sites file.
width=*value*
Width of bins.
graph=*name*
Basename to save graphing data/commands files. Graphs are saved in the current working directory with the extensions *.gp* and *.dat*

NOTES

The plotting program can be selected by setting the environment variable **GRASS_GNUPLOT** .

EXAMPLE

Given a sites file named *example* in the following format:

```
83.8|92.2|3.5689
83.8|82.2|3.9269
83.8|80.2|3.5389
83.8|69.2|3.7452
```

Suppose that we wish to examine normality of the site values (third column). The first step is to examine minimum and maximum and determine with the histogram bin width using **s.univar**:

> **s.univar −gq sites**=example

This command outputs:

```
n=216
min=1.489
max=3.9419
...
```

along with other useful statistics. For this range and number of observations, we randomly select 0.1 as the histogram bin width. Then, the following command graphs a probability plot and saves it in the current working directory with a basename of *myplot*:

> **s.probplt −q sites**=example **width**=0.1 **graph**=myplot

To view the graph again, try

> **gnuplot** myplot.gp

Using *gnuplot*, the graphs may be output as PostScript, LaTeX, FrameMaker MIF, or many other formats.

SEE ALSO

s.univar
s.normal
gnuplot

BUGS

Please send all bug fixes and comments to the author or the grass development team.
`http://www.geog.uni-hannover.`

AUTHOR

James Darrell McCauley <darrell@mccauley−usa.com>,
when he was at: Agricultural Engineering Purdue University

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.proj − Allows the user to re−project a sites file from one location to the current location.
(GRASS Sites Program)

SYNOPSIS

s.proj
s.proj help
s.proj [−l] input=*name* location=*name* **[mapset=***name***] [dbase=***name***] [output=***name***]**

DESCRIPTION

Allows the user to create a new sites file in the current location by re−projecting a sites file from another location, similar to *v.proj* and *r.proj*. Each site will retain all associated category, double, and string attributes. The header information is copied and an additional header comment is written containing the *s.proj* command line used to create the new file.

Flags:

−l
List sites files in input location and exit

Parameters:

input
Name of the input sites list.
mapset
Mapset of input sites list.
location
Location of the input sites list.
dbase
path to GRASS database of input location.
output
Name of the output sites list.

NOTES

v.proj supports transformations between nad27 and nad83. It performs the transformation automatically based on the datum specified in the mapset region. See BUGS section.

BUGS

Currently v.proj does not support general datum conversions. It only supports conversion between nad27 and nad83, and only within the CONUS conversion grid; 20 degrees to 50 degrees north latitude, 63 degrees to 131 degrees west longitude. That covers all of the conterminous USA plus Mexico north of Mexico City and most of Canada farther south than Winnipeg, Manitoba.

SEE ALSO

m.proj, r.proj, v.proj, g.setproj, i.rectify, i.rectify3, r.support, r.stats, s.sample, s.surf.idw, s.surf.rst

AUTHOR

Bill Brown

Last changed: $Date: 2002/04/22 16:09:22 $

NAME

s.qcount − indices for quadrat counts of sites lists
(GRASS Sites Program)

SYNOPSIS

s.qcount
s.qcount help
s.qcount [**−ciq**] **sites=***name* **n=***value* **r=***value*

DESCRIPTION

s.qcount chooses **n** circular quadrats of radius **r** such that they are completely within the bounds of the current region and no two quadrats overlap. The number of sites falling within each quadrat are counted and indices are calculated to estimate the departure of site locations from complete spatial randomness.

OPTIONS

Flags:

−c
Print count data for each quadrat in the form of a sites list (E|N|#n %count, where E is the easting and N is the northing of the quadrat center, n is the quadrat number, and count is the number of sites contained within the quadrat).
−i
Suppress printing table of indices (default is to print indices and not the count data).
−q
Quiet. Cut out the chatter.

Parameters:

sites=*name*
Name of sites list defining qcount points.
n=*value*
Number of quadrats (integer).
r=*value*
Radius of quadrats (floating point).

NOTES

This program may not work properly with lat−long data. It uses *hypot()* in two files: *count.c* and *findquads.c*.

SEE ALSO

s.random

Complete Spatial Randomness and Quadrat Methods − GRASS Tutorial on *s.qcount* (available as s.qcount−tutorial.ps),

General references include:
Noel A. C. Cressie. *Statistics for Spatial Data*. Wiley Series in Probability and Mathematical Statistics. John Wiley & Sons, New York, NY, 1st edition, 1991.
Brian D. Ripley. *Spatial Statistics*. John Wiley \& Sons, New York, NY, 1981.

References to the indices include:
F. N. David and P. G. Moore. Notes on contagious distributions in plant populations. *Annals of Botany*, 18:47−53, 1954.
J. B. Douglas. Clustering and aggregation. *Sankhya B*, 37:398−417, 1975.
R. A. Fisher, H. G. Thornton, and W. A. Mackenzie. The accuracy of the plating method of estimating the density of bacterial populations. *Annals of Applied Biology*, 9:325−359, 1922.
M. Lloyd. Mean crowding. *Journal of Animal Ecology*, 36:1−30, 1967.
M. Morista. Measuring the dispersion and analysis of distribution patterns. *Memoires of the Faculty of Science, Kyushu University, Series E. Biology*, 2:215−235, 1959.

BUGS

Timestamp not working for header part of counts output. (2000−10−28)

Please send all bug fixes and comments to the author or the grass development team.
`http://www.geog.uni−hannover.`

AUTHOR

James Darrell McCauley <darrell@mccauley−usa.com>,
when he was at: Agricultural Engineering Purdue University

Modified for GRASS 5.0 by Eric G. Miller (2000−10−28)

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.random − Randomly generate a GRASS sites list.
(GRASS Sites Program)

SYNOPSIS

s.random
s.random help
s.random [**−d**] **n=***value* **sites=***name*

DESCRIPTION

s.random randomly generates sites within the current region using the selected random number generator.

OPTIONS

Flags:

−d
Use *drand48()*. [default is *rand()*]

Parameters:

n=*value*
Positive integer value indicating the number of sites to be created.
sites=*name*
Name of a sites file to store random points in.

SEE ALSO

UNIX man pages for *rand(3)* and *drand48(3)*.

g.region
s.perturb

BUGS

The RNG used by *s.perturb* should probably be added to this program.

Please send any bug fixes and comments to the grass development team.

http://grass.itc.it

AUTHOR

James Darrell McCauley <darrell@mccauley−usa.com>,
when he was at: Agricultural Engineering Purdue University

Last changed: $Date: 2003/10/12 04:48:20 $

NAME

s.sample − Sample a raster file at site locations.
(GRASS Sites Program)

SYNOPSIS

s.sample
s.sample help
s.sample [**−BClq**] **input=***name* [**attr=***type*] [**field=***value*] [**output=***name*] **rast=***name* [**z=***name*]

DESCRIPTION

s.sample samples a GRASS raster map at the site locations in the input file by either cubic convolution interpolation, bilinear interpolation, or nearest neighbor sampling (default). Category label values may sampled.

This program may be especially useful when sampling for cross validation of interpolations whose output is a raster map.

OPTIONS

Flags:

−B
Use bilinear interpolation.
−C
Use cubic convolution interpolation.
−l
Use raster category label values instead of cell values. Labels must be numeric.
−q
Quiet. Cut out the chatter.

Parameters:

input=*name*
Name of sites list defining sample points.
attr=*type*
Site attribute type to use for comparison.
Options: *none, dim, cat, decimal*
If *none* is chosen no comparison will be done.
field=*value*

Site attribute field to use for comparison.
Ignored when the attribute is *none* or *cat*.

output=*name*
: Optional name of sites list in which output will be stored. Standard output is used if this is missing.

rast=*name*
: Name of raster map to be sampled.

z=*value*
: Option scaling factor for values read from raster map. Sampled values will be multiplied by this factor. If omitted, this is set to 1.0.

NOTES

The output site list will have the easting and northing of the input site list. The category value will either be the same as the value in the input site list or a unique value if there were no categories in the input site list. One decimal attribute will be output containing either, the cell value of the raster if no comparison was done (i.e. **attr=none**) or it will be the difference between the cell value and the site list attribute value.

If **−l** is specified, it is important that the raster category label is numeric. No error checking is done except for "no data" values as raster category labels. In this instance, a warning is issued and a zero value is assumed.

When NULL values are encountered for a cell, zero value is used instead. In these cases, more acurrate results may be obtained by using the default nearest neighbor comparisons.

This program may not work properly with lat−long data when the **−BC** flags are used.

When interpolation is done (i.e., the **−BC** flags are used), values are assumed to be located at the centroid of grid cells. Therefore, current resolution settings are important.

SEE ALSO

s.random
s.kcv
g.region
Image Sampling Methods − GRASS Tutorial on *s.sample* (available as s.sample−tutorial.ps.gz)

BUGS

Please send all bug fixes and comments to the author or the grass development team.
`http://www.geog.uni−hannover/grass/`

AUTHOR

James Darrell McCauley <darrell@mccauley−usa.com>,
when he was at: Agricultural Engineering Purdue University

Updated for GRASS 5.0 by Eric G. Miller

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.surf.idw − Surface interpolation from sites data by Inverse Distance Squared Weighting.
(GRASS Sites Program)

SYNOPSIS

s.surf.idw [−n] input=*name* **output=***name* **[npoints=***count***] [field=***number***]**

DESCRIPTION

s.surf.idw fills a raster matrix with interpolated values generated from a set of irregularly spaced data points using numerical approximation (weighted averaging) techniques. The interpolated value of a cell is determined by values of nearby data points and the distance of the cell from those input points. In comparison with other methods, numerical approximation allows representation of more complex surfaces (particularly those with anomalous features), restricts the spatial influence of any errors, and generates the interpolated surface from the data points.

This program allows the user to use a GRASS site list file, rather than a raster map layer, as input.

The program will be run non−interactively if the user specifies the values of needed program parameters and any desired optional parameter values on the command line, using the form:

> **s.surf.idw [−n] input=***name* **output=***name* **[npoints=***count***] [field=***number***]**

Alternately, the user can simply type **s.surf.idw** on the command line, without program arguments. In this case, the user will be prompted for needed inputs and option choices using the standard GRASS *parser* interface.

OPTIONS

Flags:

−n
: Don't index sites by cell. (for very large regions; uses less memory, but is very slow if there is a large number of sites)

Parameters:

input=*name*
: Name of an input site list file that contains sites with one or more floating point attributes.

output=*name*

Name to be assigned to the new output raster map layer containing a smooth surface generated from the known data values in the input site list file.

npoints=*count*

The number of points to use for interpolation. By default, the 12 nearest points are used for interpolation.
Default: 12

field=*number*

Number of floating point attribute to use for interpolation.
Default: 1

NOTES

The amount of memory used by this program is related to the number of sites in the current region. If the site list is very dense (i.e., contains many data points), the program may not be able to get all the memory it needs from the system. The time required to execute is related to the resolution of the current region, after an initial delay determined by the time taken to read the input sites file.

If the user has a mask set, then interpolation is only done for those cells that fall within the mask. However, all sites in the current region are used even if they fall outside the mask. Sites outside the current region are not used in the interpolation. A larger region may be set and a mask used to limit interpolation to a smaller area if it is desired to use sites from outside the region in the interpolation.

If more than *count* sites fall into one target raster cell, the mean of all the site values will determine the cell value (unless the −n flag is specifed, in which case only the *count* sites closest to the centre of the cell will be interpolated).

SEE ALSO

d.sites
g.region
r.mask
r.surf.contour
r.surf.idw
r.surf.idw2
r.surf.gauss
r.surf.fractal
r.surf.random
v.surf.rst
s.surf.rst
parser

AUTHOR

Michael Shapiro, U.S. Army Construction Engineering Research Laboratory
Improved algorithm (indexes sites according to cell and ignores sites outside current region) by Paul Kelly

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

s.surf.krig
Surface interpolation from site data via kriging

SYNOPSIS

s.surf.krig
s.surf.krig −help
s.surf.krig input=*name* [**zdimension**=*index*] [**znumeric**=*index*]
[**zstring**=index] **outz=**name **outvarz=**name **model=**name
[**npoints=**count] [**range=**semivariogram range] [**power=**exponetial power]
[**nugget=**semivariogram nugget] [**sill=**semivariogram sill] [**max_lag=**max_lag]

DESCRIPTION

Is missing.

Parameters:

input=*name*
Name of input site map
zdimension=*index*
Index of dimension field if appropriate
znumeric=*index*
Index of numeric field if appropriate
zstring=*index*
Index of string field if appropriate
outz=*name*
Name of output z value raster map
outvarz=*name*
Name of output z variance raster map
outvarz=*name*
Name of output z variance raster map
model=*type*
Type of semivariogram model to be used
options: power,spherical,exp,log,gaussian
npoints=*value*
Number of interpolation points
default: 12
range=*value*
Not used with log or power models
default: 1.0
power=*value*

Only for use with the power model
default: 1.0

nugget=*value*
The variogram nugget variance
default: 0.0

sill=*value*
The variogram sill scaled to the sample variance
default: 1.0

max_lag=*value*
The max_lag is only used with the power model
default: 1000.0

NOTE

You have to play with the parameters to get a reasonable result and not a segmentation violation

At least one of the options zdimension | znumeric | zstring must be selected and supplied with a valid (0−based) index value. This will contain the values of the spot heights.

AUTHORS

Chris Skelly
School of Earth Sciences
Macquarie Univerity
North Ryde 2109 NSW Australia

Darrel McCauley

David D. Gray <ddgray@armadce.demon.co.uk> 09/2000

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.surf.rst− interpolation and topographic analysis from given site data to GRASS floating point raster format using regularized spline with tension (this program replaces *s.surf.tps*)
(GRASS Site Program)

SYNOPSIS

s.surf.rst
s.surf.rst help
s.surf.rst [−d] [−t] input = name [**elev** = name] [**field**=val] [**slope** = name] [**aspect** = name] [**pcurv** = name] [**tcurv** = name] [**mcurv** = name] [**maskmap** = name] [**dmin** = val] [**zmult =** val] [**tension** = val] [**smooth** = val] [**smatt=**val] [**segmax =** val] [**npmin** = val] [**theta** = val] [**scalex** = val] [**devi** = name] [**treefile** = name] [**overfile** = name]

DESCRIPTION

s.surf.rst
This program interpolates the z_i−values from point data (e.g., elevations, climatic stations, drill holes, etc.) given in a sites file named *input* as x|y|%z_1 %z_2 ...%z_i... to grid cells in the output raster file *elev* representing a surface. The user can select which floating point attribute will be interpolated by setting the parameter *field* to the value i for the i−th floating point attribute**.**

As an option, simultaneously with interpolation, topographic parameters slope, aspect, profile curvature (measured in the direction of steepest slope), tangential curvature (measured in the direction of a tangent to contour line) or mean curvature are computed and saved as raster files specified by the options *slope, aspect, pcurv, tcurv, mcurv* respectively. If *−d* flag is set the program outputs partial derivatives f_x, f_y, f_{xx}, f_{yy}, f_{xy} instead of slope, aspect, profile, tangential and mean curvatures respectively.

User can define a raster file named *maskmap*, which will be used as a mask. The interpolation is skipped for cells which have zero or NULL value in mask. Data points are checked for identical points and the points that are closer to each other than the given *dmin* are removed. Parameter *zmult* allows user to rescale the z−values for sites (useful e.g. for transformation of elevations given in feet to meters, so that the proper values of slopes and curvatures can be computed).

Regularized spline with tension and smoothing is used for interpolation and approximation. The *tension* parameter tunes the character of the resulting surface from thin plate to membrane. The experimental flag *−t* can be set to use "dnorm independent tension", (see notes for more details about the tension behavior). For noisy data, it is possible to define either a constant smoothing parameter *smooth* or a variable smoothing parameter by setting the parameter *smatt* to the value j for the j−th floating point attribute in the *input* site file, representing the smoothing parameter for each point. When smoothing is used, it is possible to output site file *devi* containing deviations of the resulting surface from the given data.

If the number of given points is greater than *segmax*, segmented processing is used. The region is split into rectangular segments, each having less than *segmax* points and interpolation is performed on each segment of the region. To ensure smooth connection of segments the interpolation function for each segment is computed using the points in given segment and the points in its neighborhood which are in the rectangular window surrounding the given segment. The number of points taken for interpolation is controlled by *npmin*, the value of which must be larger than *segmax*. User can choose to output vector files *treefile* and *overfile* which represent the quad tree used for segmentation and overlapping neighborhoods from which additional points for interpolation on each segment were taken. Anisotropic surfaces can be interpolated setting anisotropy angle *theta* and scaling factor *scalex*. The program writes several important values to history file of raster map *elev*. If the input data have time stamp, the program creates time stamp for all output files.

The user must run *g.region* before the program to set the region and resolution for interpolation.

OPTIONS

The user can run this program either interactively or non−interactively. The program will be run non−interactively if the user specifies program arguments and flag settings on the command line using the form:

s.surf.rst [−d] [−t] input = name **elev** = name [**field**=val] [**slope** = name] [**aspect** = name] [**pcurv** = name] [**tcurv** = name] [**mcurv** = name] [**maskmap** = name] [**dmin** = val] [**zmult =** val] [**tension** = val] [**smooth** = val] [**smatt**=val] [**segmax =** val] [**npmin** = val] [**theta** = val] [**scalex** = val] [**devi** = name] [**treefile** = name] [= name]

Alternately, the user can simply type **s.surf.rst** on the command line without program arguments. In this case, the user will be prompted for parameter values and flag settings using the standard GRASS parser interface described in the manual entry for *parser*.

Flags

−d Output partial derivatives instead of aspect, slope and curvatures.
−t Use dnorm independent tension (experimental)

Parameters:

input = *name*
Use the existing site file name as input.

elev = *name*
Output elevation values to raster file *name*.

field=*val*
decimal attribute to use for elevation (1=first) options (1−100), default is 1.

slope = *name*
Output slope or dx values to raster file *name*.

aspect = *name*
Output aspect or dy values to raster file *name*.

pcurv = *name*
Output profile curvature or dxx values to raster file *name*.

tcurv = *name*
Output tangential curvature or dyy values to raster file *name*.

mcurv = *name*
Output mean curvature or dxy values to raster file *name*.

maskmap = *name*
Use the existing raster file *name* as a mask.

dmin = *val*
Set min distance between points to *val*. Default value is set to 0.5 grid cell size.

zmult = *val*
Convert z−values using conversion factor *val*. Default value is 1.

tension =*val*
Set tension to *val*. Default value is 40.

smooth = *val*
Set smoothing parameter to *val*. Default value is 0.1.

smatt=*val*
order of floating point attribute to use for variable smoothing parameter

segmax = *val*
Set max number of points per segment to *val*. Default value is 40.

npmin = *val*
Set min number of points for interpolation to *val*. Default value is 200, for data with heterogeneous spatial distribution higher value is suggested (see notes).

theta = *val*
Set anisotropy angle in degrees (measured from East counterclockwise) to *val*.

scalex = *val*
Set anisotropy scaling factor to *val*. Values 0 and 1 give no anisotropy.

devi = *name*
Output deviations to a site file *name*.

treefile = *name*
Output quad tree used for segmentation to vector file *name*

overfile = name
Output overlapping neighborhoods used for segmentation to vector file *name*.

NOTES

*s.surf.rst*uses regularized spline with tensionfor interpolation from point data. Point data should be in a new site format , that means format x|y|%z_1 %z_2 ...%z_i..., instead of the old format which confused categories with values (x|y|#z). If program detects the old format it will allow users to have the site file rewritten to a new format automatically.

The implementation has a segmentation procedure based on quadtrees which enhances the efficiency for large data sets. The GRASS5.0 version has enhanced segmentation which takes more points for the large segments, to reduce the potential for visibility of segmentens in areas with sparse data.

Special color tables are created by the program for output raster files.

Topographic parameters are computed directly from the interpolation function so that the important relationships between these parameters are preserved. The equations for computation of these parameters and their interpretation are described in (Mitasova and Hofierka 1993). Slopes and aspect are computed in degrees (0−90 and 1−360 respectively). The aspect raster file has value 0 assigned to flat areas (with slope less than 0.1%) and to singular points with undefined aspect. Aspect points downslope and is 90 to the North, 180 to the West, 270 to the South and 360 to the East, the values increase counterclockwise. Curvatures are positive for convex and negative for concave areas. Singular points with undefined curvatures have assigned zero values.

Tension and *smooth*ing allow user to tune the surface character. For most landscape scale applications the default should work fine.The program gives warning when significant overshoots appear in the resulting surface and higher tension or smoothing should be used.
While it is possible to automatize the selection of suitable *tension* and *smooth*ing, it has not been done yet, so here are some hints which may help to choose the proper parameters if the results look "weird". It is useful to know that the method is scale dependent and the *tension* works as a rescaling parameter (high *tension* "increases the distances between the points" and reduces the range of impact of each point, low *tension* "decreases the distance" and the points influence each other over longer range). Surface with *tension* set too high behaves like a membrane (rubber sheet stretched over the data points) with peak or pit ("crater") in each given point and everywhere else the surface goes rapidly to trend. If digitized contours are used as input data, high tension can cause artificial waves along contours. Lower tension and higher smoothing is suggested for such a case.
Surface with *tension* set too low behaves like a stiff steel plate and overshoots can appear in areas with rapid change of gradient and segmentation can be visible. Increase tension should solve the problems.

There are two options how *tension* can be applied in relation to *dnorm* (dnorm rescales the coordinates depending on the average data density so that the size of segments with *segmax*=40 points is around 1 − this ensures the numerical stability of the computation):

1. Default (used also in s.surf.tps): the given *tension* is applied to normalized data (x/*dnorm*..), that means that the distances are multiplied (rescaled) by *tension/dnorm*. If density of points is changed, e.g., by using higher *dmin*, the *dnorm* changes and *tension* needs to be changed too to get the same result. Because the *tension* is applied to normalized data its suitable value is usually within the 10−100 range and does not depend on the actual scale (distances) of the original data (which can be km for regional applications or cm for field experiments).
2. Flag **−t** (experimental for s.surf.rst)**:** The given *tension* is applied to un−normalized data (rescaled tension = *tension***dnorm*/1000 is applied to normalized data (x/*dnorm*) and therefore *dnorm* cancels out) so here *tension* truly works as a rescaling parameter. For regional applications with distances between points in km the

suitable tension can be 0.1 or smaller, for detailed field scale analysis with distances in cm it can be 500 or more. To help select how much the data need to be rescaled the program writes *dnorm* and rescaled tension=*tension*dnorm*/1000 at the beginning of the program run. This rescaled *tension* should be around 20−30. If it is lower or higher, the given *tension* parameter should be changed accordingly.

The default is a recommended choice, however for the applications where the user needs to change density of data and preserve the interpolation character the **−t** flag can be helpful.

Anisotropic data (e.g. geologic phenomena) can be interpolated using *theta* and *scalex* defining orientation and ratio of the perpendicular axes put on the longest/shortest side of the feature, respectively. *Theta* is measured in degrees from East, counterclockwise. *Scalex* is a ratio of axes sizes. Setting *scalex* in the range 0−1, you will get pattern prolonged in the direction defined by *theta*. *Scalex* value 0.5 means that your feature is approximatelly 2 times longer in the direction of *theta* than in the perpendicular direction. *Scalex* value 2 means that axes ratio is reverse and you will get pattern perpendicular to the previous example. Please note that anisotropy option has not been extensively tested and may include bugs − if there are problems, please report to GRASS bugtracker (accessible from http://grass.itc.it/).

For data with values changing over several magnitudes (sometimes the concentration or density data) it is suggested to interpolate the log of the values rather than the original ones.

The program checks the numerical stability of the algorithm by computing the values in given points. The root mean square deviation (rms) between interpolated and given values is written into the history file of raster map *elev*. For computation with smoothing set to 0. the rms should be 0. Significant increase in tension is suggested if the rms is unexpectedly high for this case. With smoothing parameter greater than zero the surface will not pass exactly through the data points and the higher the parameter the closer the surface will be to the trend. The rms then represents a measure of smoothing effect on data. More detailed analysis of smoothing effects can be performed using the output deviations option and running s.univar on the site file with deviations.

The program writes the values of parameters used in computation into the comment part of history file *elev* as well as the following values which help to evaluate the results and choose the suitable parameters: minimum and maximum z values in the data file (zmin_data, zmax_data) and in the interpolated raster map (zmin_int, zmax_int), rescaling parameter used for normalization (dnorm), which influences the tension.

When the number of points in a site file is not too large (less than 800), the user can skip segmentation by setting *segmax* to the number of data points or segmax=700.

The program gives warning when user wants to interpolate outside the rectangle given by minimum and maximum coordinates in site file, zoom into the area where the points are is suggested in this case.

When a mask is used, the program takes all points in the given region for interpolation, including those in the area which is masked out, to ensure proper interpolation along the border of the mask. It therefore does not mask out the data points, if this is desirable, it must be done outside s.surf.rst (e.g. using r.mask.points).

For examples of applications see http://skagit.meas.ncsu.edu/~helena/gmslab/viz/
and http://skagit.meas.ncsu.edu/~helena/gmslab/

SEE ALSO

r.slope.aspect, r.surf.idw , r.surf.idw2 , r.surf.contour , s.surf.idw , v.to.sites , g.region , r.mask , v.surf.rst , r.resamp.rst

AUTHORS

Original version of program (in FORTRAN) and GRASS enhancements:
Lubos Mitas, NCSA, University of Illinois at Urbana−Champaign, Illinois, USA
Helena Mitasova, Department of Geography, University of Illinois at Urbana−Champaign, Champaign, Illinois, USA

Modified program (translated to C, adapted for GRASS, new segmentation procedure):
Irina Kosinovsky, US Army CERL, Champaign, Illinois, USA
Dave Gerdes, US Army CERL, Champaign, Illinois, USA

Modifications for new sites format and timestamping:
Darrel McCauley, Purdue University, West Laffayette, Indiana, USA

REFERENCES

Hofierka J., Parajka J., Mitasova H., Mitas L., 2002, Multivariate Interpolation of Precipitation Using Regularized Spline with Tension. *Transactions in GIS* 6(2), pp. 135−150.

Mitas, L., Mitasova, H., 1999, Spatial Interpolation. In: P.Longley, M.F. Goodchild, D.J. Maguire, D.W.Rhind (Eds.), *Geographical Information Systems: Principles, Techniques, Management and Applications*, Wiley, pp.481−492

Mitasova H., Mitas L., Brown W.M., D.P. Gerdes, I. Kosinovsky, Baker, T.1995, Modeling spatially and temporally distributed phenomena: New methods and tools for GRASS GIS. *International Journal of GIS*, 9 (4), special issue on Integrating GIS and Environmental modeling, 433−446.

Mitasova H. and Mitas L. 1993 : Interpolation by Regularized Spline with Tension: I. Theory and Implementation, *Mathematical Geology* 25, 641−655.

Mitasova H. and Hofierka J. 1993 : Interpolation by Regularized Spline with Tension: II. Application to Terrain Modeling and Surface Geometry Analysis, *Mathematical Geology* 25, 657−667.

Mitasova, H., 1992 : New capabilities for interpolation and topographic analysis in GRASS, *GRASSclippings* 6, No.2 (summer), p.13.

Mitas, L., Mitasova H., 1988 : General variational approach to the interpolation problem, *Computers and Mathematics with Applications* 16, p. 983

Talmi, A. and Gilat, G., 1977 : Method for Smooth Approximation of Data, *Journal of Computational Physics*, 23, p.93−123.

Wahba, G., 1990, : Spline Models for Observational Data, CNMS−NSF Regional Conference series in applied mathematics, 59, SIAM, Philadelphia, Pennsylvania.

Updated April 2, 2002 by Jaro Hofierka and Helena Mitasova

Last changed: $Date: 2003/02/05 12:46:03 $

NAME

s.sv − Sample semivariogram of a GRASS sites list.
(GRASS Sites Program)

SYNOPSIS

s.sv
s.sv help
s.sv [**−alq**] [**sites=***name*] **lag=***value* [**lagtol=***value*] [**direction=***value*] [**angtol=***value*] [**graph=***name*] [**field=***value***]**

DESCRIPTION

s.sv calculates a sample semivariogram and either plots it using *gnuplot* or writes it to standard output.

For more information, refer to the tutorial or see the example below.

OPTIONS

Flags:

−q
Quiet. Cut out the chatter.
−p
Plot the sample semivariogram (requires *gnuplot*).
field
Number of z−field attribute to use for calculation
default: 1

Parameters:

sites=*name*
Name of an existing sites file. Default is standard input with no field separators.
lag=*value*
Nominal lag distance.
lagtol=*value*
Tolerance on lag distance. Default is half of nominal distance.
direction=*value*
Direction of semivariogram. Default is omnidirectional semivariogram.
angtol=*value*
Angular tolerance on direction.

graph=*name*
: Basename to save graphing data/commands files. Graphs are saved in the current working directory with the extensions *.gp* and *.dat*. Implies the **−p** flag. If unspecified, semivariogram is written to standard output.

NOTES

The plotting program can be selected by setting the environment variable **GRASS_GNUPLOT** .

Without the **−p** flag, three columns of data are written to standard output: lag distance (*h*), semivariogram value (*gamma*), and the number of data pairs used to compute it (*N(h)*). When the **graph** parameter is set, these same three columns of data are written to *name*.dat. Therefore, to replot a sample semivariogram, use:

gnuplot *name*.gp

To plot a histogram of *N(h)*, simply edit *name*.gp and redo the previously given command.

For more information, refer to the tutorial.

SEE ALSO

s.univar
s.surf.krig
s.normal
m.svfit
gnuplot
Semivariogram Modeling − A GRASS Tutorial on Exploratory Data Analysis and Semivariogram Modeling.

BUGS

Will not work correctly with lat−long data. Should *G_azimuth()* be used to calculate the angle between points?

Only Matheron's classical estimator is available with *s.sv*. Others may be added in the future.

AUTHOR

James Darrell McCauley, Agricultural Engineering, Purdue University

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.sweep − computes Voronoi diagram or Delaunay triangulation (used by s.voronoi and s.delaunay)

SYNOPSIS

s.sweep [−dstp]

DESCRIPTION

s.sweep reads the standard input for a set of points in the plane and writes either the Voronoi diagram or the Delaunay triangulation to the standard output. Each input line should consist of two real numbers, separated by white space. The program is used by s.voronoi and s.delaunay and is not meant to be useful as a standalone program. If option −t is present, the Delaunay triangulation is produced. Each output line is a triple i j k, which are the indices of the three points in a Delaunay triangle. Points are numbered starting at 0. If option −t is not present, the Voronoi diagram is produced. There are four output record types.

```
       s a b  indicates  that  an  input point at coordinates a b
              was seen.

       l a b c
              indicates a line with equation ax + by = c.

       v a b  indicates a vertex at a b.

       e l v1 v2
              indicates a Voronoi segment which is  a  subsegment
              of line number l with endpoints numbered v1 and v2.
              If v1 or v2 is −1, the line extends to infinity.
```

The other options are:

```
       s      The input is sorted by  y  coordinate  (the  second
              number  in  the pair).  The first input line should
              be npoints xmin xmax ymin ymax.  This describes the
              number of points and the range of the points.  This
              line is used to determine internal hash table size;
              it  need not be exact but performance suffers if it
              is grossly wrong.

       p      Produce output suitable  for  input  to  plot  (1),
              rather than the forms described above.
```

On unsorted data uniformly distributed in the unit square, s.sweep uses about 20n+140/n bytes of storage. On sorted data, s.sweep uses about 160/n bytes.

REFERENCES

Steve J. Fortune, (1987). A Sweepline Algorithm for Voronoi Diagrams, Algorithmica 2, 153−174.

SEE ALSO

v.autocorr, *v.spag*, *v.support*, *s.delaunay*, *s.sweep*

AUTHOR

James Darrell McCauley,
GRASS 5 update, improvements: Andrea Aime, Modena, Italy

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.territory − Calculates territory consumption for animal based on distributed positions.
(GRASS Sites Program)

SYNOPSIS

s.territory
s.territory help
s.territory rast=*value* **sites=***name* [**output=***name*] [**claim=***name*] [**thresh=***name*] [**incr=***name*]

DESCRIPTION

s.territory calculates territory consumption for animal based on distributed positions.

OPTIONS

Parameters:

rast=*value*
  Name of incidence or density file
sites=*name*
  Name of sites file with threshold locations
output=*name*
  Name of new sites file to contain radius
claim=*name*
  Simultaneous growth map. (ignored if site output given)
thresh=*name*
  Field in sites file containing thresholds. Threshold is target amount of resources that site wants to consume
incr=*name*
  Field in sites file containing radius increment (meters). incr is radius increment for each iteration
   − default incr is min distance between cells
   − not currently implemented for sequential, only simult.

NOTES

OUTPUT is a sites file containing:

```
easting|northing|#cat %threshold %radius
```

where radius is the minimum radius of a circle which surrounds enough cells in the incidence map to sum up to threshold.
If neither a sites output file nor a claim map is named, output is simply the radius followed by a newline for each site (for easier use in scripts).

EXAMPLES:

- Lion's territory

1. you have sites for lion's lairs with food needs as the "threshold"
2. you have an incidence map for frequency of prey or hunting success probability (Kg/week or something)
3. s.territory creates a circular range for each lair which you might use to create vector or raster files and predict areas of conflict in overlapping "territories"

- School children

1. you have sites for schools with school capacity as the "threshold"
2. you have population density maps for school−age children
3. s.territory creates ranges for each school that encompass an area in which enough children live to fill the school

ALGORITHM:

```
radius increment set to lesser of ns_res and ew_res
( or incr if given )

for each site
{
  for (radius = increment; cells remaining; incr+=incr)
  {
    sum incidence cell centers which fall within radius
    if >= threshold, break
  }
}
```

- *implementation (fast, high memory):*
  read density map into buffer
  use additional buffer to calculate distances(squared)
  to current site (as needed)
  if distance_squared > prev radius, add it in

- *implementation for simultaneous growth (much much slower):*
  reads density map, then repeatedly traverses
  output map looking for null cells, then checking to
  see if it's in any of the sites current radius, etc.

- *alternatives:*

```
   1. let max increase until entire map is covered,
      using zero for outside incidence
      >>> currently implemented

   2. use a rectangular(square) territory rather than circle?

   3. somehow "weight" expansion toward successful areas

   4. save combined nearest distances map


ADDENDUM

Think about:

1.  Make already claimed territory uninhabitable − as each site is
    processed, zero out resources within territory.
    >>> This can be done in scripts, using masks. (slower though)

2.  Instead of just going through in order, attack them all at the
    same time, zeroing out resources as it goes and assigning a
    category value to a "claim" map.
```

AUTHOR

Bill Brown, GMSL, 1999

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.to.rast3 − Converts a site file to a 3dcell volume (G3D)
(GRASS Sites Program)

SYNOPSIS

s.to.rast3
s.to.rast3 help
s.to.rast3 [**−qs**] **input=***name* **output=***name* **field=***value*

DESCRIPTION

s.to.rast3 converts a GRASS site file to a GRASS 3dcell file.

The quad size (ie the number of cells used to represent one site) is fixed to 0 (ie one cell/site). If the output file already exists, the program exits gracefully.

OPTIONS

Parameters:

input=*name*
  Name of input site list
output=*name*
  Name of output cell file
field=*value*
  decimal field number to use for operation
  default: 1

NOTES

When converting from a sites layer to a 3dcell volumne the site is converted into a single cube representing the location of the site.

SEE ALSO

g3.region, r3.to.sites

AUTHOR

Jaro Hofierka, Geomodel s.r.o.

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.to.rast − Converts a site file to a raster map
(GRASS Sites Program)

SYNOPSIS

s.to.rast
s.to.rast help
s.to.rast [**−qs**] **input=***name* **output=***name* **size=***value* **title=***"phrase"* **field=***value* **findex=***value* **string=***value*

DESCRIPTION

s.to.rast converts a GRASS site file to a GRASS raster file.

The quad size (ie the number of cells used to represent one site) is fixed to 0 (ie one cell/site). If the output file already exists, the program exits gracefully.

OPTIONS

Flag:

−q
Run the module quietly.
−s
Create a single−valued (0/1, binary) raster map

Parameters:

input=*name*
Name of input site list
output=*name*
Name of output cell file
size=*value*
Number of cells to surround site cell
title=*"phrase"*
Title for the resulting raster map
field=*value*
Attribute field type to use for operation
options: dim,decimal,cat
default: decimal
findex=*value*

Attribute field number to use for operation
default: 1

string=*value*

String attribute number to use for description
default: 1

NOTES

When converting from a sites layer to a raster layer with size=0, the site is converted into a single cell representing the location of the site, which can be hard to see (depending on the raster resolution).
size=2 (or higher) makes the sites show up as fatter in the raster layer.

Alternaticely the user can convert the sites to vector format and use v.bubble or v.circle to represent the location.

As site files can consist of several attribute fields (decimals or string categories), the user can select the column of interest for conversion using the filed selection parameters (findex, string). The attribute field type to use for operation is selected by parameter *field*.

The user will be warned if two or more sites fall into one raster cell. However, the module will continue to import (the last site value will determine the cell value). If such overwriting is not wanted, the cell resolution has to be changed using g.region.

The user will be warned if sites fall out of the current region.

SEE ALSO

g.region, *s.menu*, *s.to.vect*, *v.bubble*, *v.circle*,

AUTHORS

Katarina Johnsson, CCRS
GRASS 5 improvements: Eric G. Miller

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.to.vect − Converts a GRASS site_lists file into a vector file.
(GRASS Sites Program)

SYNOPSIS

s.to.vect
s.to.vect help
s.to.vect [**−p**] **input=***name* **output=***name* [**cat=***type,index*]

DESCRIPTION

s.to.vect converts GRASS **`site_lists`** file into vector files. The resulting vector file can be treated as any other vector file. The requirements of the site_lists file are standard (i.e., a regular **`site_lists`** file format is required). **`site_lists`** file values are used as **`dig_cats`** category values.

OPTIONS

Flags:

−p
Don't prompt for map header information.

Parameters:

input=*name*
Name of input GRASS **`site_lists`** file to be converted.
output=*name*
Name to be assigned to the vector output file.
cat=*type,index*
Field type ("string", "dim", or "decimal") and index to use for the vector category value. By default, the category number is used. The *type* is the **`site_lists`** attribute type of "string", "dim" (or dimension), or "decimal". The *index* is a positive integer attribute number for the specified *type*. That is, the second "string" attribute would be "2" or the fourth dimension would be "four". For the "dim" type, the *index* must be greater than 2.

NOTES

The specification for **`site_lists`** allows category numbers to be missing or to be floating point. Since attributes in **`vector`** layers are linked by positive integers, when a **`site_lists`** file has floating point or missing category numbers, a positive sequential numbering is used instead. This may have the unfortunate side−effect of generating different category numbers for category values that are exactly the same.

AUTHOR

R.L. Glenn, USDA, SCS, NHQ−CGIS
GRASS 5.0 Team

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.univar − Univariate statistics for a GRASS sites list.
(GRASS Sites Program)

SYNOPSIS

s.univar
s.univar help
s.univar [**−aglq**] **sites=***name* **[field=***value***]**

DESCRIPTION

s.univar calculates univariate statistics of sites lists. This includes the number of sites, mean, standard deviation, coefficient of variation, mininum, first quartile, median, third quartile, and maximum.

OPTIONS

Flags:

−a
Use all sites found in the named *sites* file, rather than limiting output to sites falling within the current geographic region.

−g
Print the statistics in shell script style (similar to *g.region*).

−l
Use site descriptions (category labels) for z values. Sites must be in the (E|N|#n desc) format for this to work. Default format is just three pipe−separated floating point numbers (E|N|desc).

−q
Quiet. Cut out the chatter.

Parameters:

sites=*name*
Name of an existing sites file.

field
Number of z−field attribute to use for calculation
default: 1

SEE ALSO

s.normal, *r.univar*,

BUGS

Needs to do as many calculations as possible without storing individual site records (requires a lot of memory).

BUGS

Please send all bug fixes and comments to the author or the grass development team.
`http://www.geog.uni-hannover.`

AUTHOR

James Darrell McCauley <darrell@mccauley-usa.com>,
when he was at: Agricultural Engineering Purdue University

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.vol.idw − Interpolates point data to a 3D grid.
(GRASS 3D Program)

SYNOPSIS

s.vol.idw
s.vol.idw help
s.vol.idw input=*name* **output=***name* **[npoints=***count***] [field=***value***]**

DESCRIPTION

s.vol.idw fills a GRID3D raster volume matrix with interpolated values generated from a set of irregularly spaced data points using numerical approximation (weighted averaging) techniques. The interpolated value of a tile is determined by values of nearby data points and the distance of the cell from those input points. In comparison with other methods, numerical approximation allows representation of more complex volumes (particularly those with anomalous features), restricts the spatial influence of any errors, and generates the interpolated volume from the data points.

Parameters:

input
: Name of input 3D sites file

output
: Name of output 3D − G3D raster file

npoints
: Number of interpolation points
Default: 12

field
: Number of z−field attribute to use for calculation
default: 1

NOTES

If two or more sites fall into one voxel, the last site value will determine the 3dcell value (no warning yet).

SEE ALSO

s.vol.rst, *s.to.rast3*

AUTHOR

Jaro Hofierkahofierka@geomodel.sk

Last changed: $Date: 2002/01/25 05:45:35 $

NAME

s.vol.rst − Interpolates point data to a G3D grid volume using regularized spline with tension (RST) algorithm *(GRASS 3D Program)*

SYNOPSIS

s.vol.rst input=*name* **[cellinp=***name***] [field=***value***] [tension=***value***] [smooth=***value***] [devi**=*name***]** **[maskmap=** *name***] [segmax=***value***] [dmin=***value***] [npmin=***value***] [wmult=***value***] [zmult=***value***]** **[cellout=***name***] [elev=***name***] [gradient=***name***] [aspect1=***name***] [aspect2=***name***] [ncurv=***name***]** **[gcurv=***name***] [mcurv=***name***]**

DESCRIPTION

s.vol.rst interpolates the values to 3−dimensional grid from point data (climatic stations, drill holes etc.) given in a 3D sites file named *input*. Output g3d file is *elev*. The 3−dimensional grid is given by the current 3D region. If the options *cellinp* and *cellout* are specified then the output raster file *cellout* contains crossection of interpolated volume with surface defined by input cell file . As an option, simultaneously with interpolation, geometric parameters magnitude of gradient, both aspects, change of gradient, Gauss−Kronecker curvature, or mean curvature are computed and saved as g3d file as specified by the options *gradient, aspect1, aspect2, ncurv, gcurv, mcurv* respectively.

At first, data points are checked for identical points and points that are closer to each other than given *dmin* are removed. Parameters *wmult* and *zmult* allow user to re−scale the w−values and z−values for sites (useful e.g. for transformation of elevations given in feet to meters, so that the proper values of gradient and curvatures can be computed).

Regularized spline with tension is used for the interpolation. The *tension* parameter tunes the character of the resulting volume from thin plate to membrane. Higher values of tension parameter reduce the overshoots that can appear in volumes with rapid change of gradient. For noisy data, it is possible to define a smoothing parameter, *smooth*. With the smoothing parameter set to zero (*smooth=0*) the resulting volume passes exactly through the data points. When smoothing is used, it is possible to output site file *devi* containing deviations of the resulting volume from the given data.

User can define a 2D raster file named *maskmap*, which will be used as a mask. The interpolation is skipped for 3−dimensional cells whose 2−dimensional projection has zero value in mask. Zero values will be assigned to these cells in all output g3d files.

If the number of given points is greater than 700, segmented processing is used. The region is split into 3−dimensional "box" segments, each having less than *segmax* points and interpolation is performed on each segment of the region. To ensure the smooth connection of segments the interpolation function for each segment is computed using the points in given segment and the points in its neighborhood. The minimum number of points taken for interpolation is controlled by *npmin* , the value of which must be larger than *segmax* and less than 700. This limit of 700 was selected to ensure the numerical stability and efficiency of the algorithm.

s.vol.rst uses regularized spline with tension for interpolation from point data (as described in Mitasova and Mitas, 1993). The implementation has an improved segmentation procedure based on Oct−trees which enhances the efficiency for large data sets.

Geometric parameters − magnitude of gradient (*gradient*), horizontal (*aspect1*) and vertical (*aspect2)* aspects, change of gradient (*ncurv*), Gauss−Kronecker (*gcurv*) and mean curvatures (*mcurv*) are computed directly from the interpolation function so that the important relationships between these parameters are preserved. More information on these parameters can be found in Mitasova et al., 1995 or Thorpe, 1979.

The program gives warning when significant overshoots appear and higher tension should be used. However, with tension too high the resulting volume changes its behavior to membrane(rubber sheet stretched over the data points resulting in a peak in each given point and everywhere else the volume goes rapidly to trend). With smoothing parameter greater than zero the volume will not pass through the data points and the higher the parameter the closer the volume will be to the trend. For theory on smoothing with splines see Talmi and Gilat, 1977 or Wahba, 1990.

If a visible connection of segments appears, the program should be rerun with higher *npmin* to get more points from the neighborhood of given segment.

If the number of points in a site file is less then 400, *segmax* should be set to 400 so that segmentation is not performed when it is not necessary.

The program gives warning when user wants to interpolate outside the "box" given by minimum and maximum coordinates in site file, zoom into the area where the points are is suggested in this case.

For large data sets (thousands of data points) it is suggested to zoom into a smaller representative area and test whether the parameters chosen (e.g. defaults) are appropriate.

The user must run *g3.region* before the program to set the region for interpolation.

Parameters:

input
Name of the site file (format see NOTES below)

field
decimal attribute to use for value w (1=first) options (1−100), default is 1.

cellinp
Name of the surface cell file to use for crossection

tension
Tension
Default: 40

smooth
Smoothing parameter
Default: 0.1

devi
Output deviations to a site file

maskmap

Name of the raster file used as mask

segmax
Max number of points in segment (=700)
Default: 50

dmin
Min distance between points (extra points ignored)
Default: Default value is set to 0.5 cell size.

npmin
Min number of points for interpolation
Default: 200

wmult
Conversion factor for w−values
Default: 1.0

zmult
Conversion factor for z−values
Default: 1.0

cellout
Name of the crossection cell file

elev
Elevation g3d−file

gradient
Gradient g3d−file

aspect1
Aspect1 g3d−file

aspect2
Aspect2 g3d−file

ncurv
Change of gradient g3d−file

gcurv
Gauss−Kronecker curvature g3d−file

mcurv
Mean curvature g3d−file

NOTES

The sites volume format is as follows:

```
   x|y|z|#n %w1 %w2 %w3
```

with x,y,z (spatial coordinates), n (optional integer number) and w (data values).

SEE ALSO

g3.region, s.in.ascii, s.vol.idw, r3.mask, s.surf.rst

AUTHOR

Original version of program (in FORTRAN) and GRASS enhancements:
Lubos Mitas, NCSA, University of Illinois at Urbana−Champaign, Illinois, USA,lubos_mitas@ncsu.edu

Helena Mitasova, Department of Geography, University of Illinois at Urbana−Champaign, Champaign, Illinois, USA, hmitaso@unity.ncsu.edu

Modified program (translated to C, adapted for GRASS, new segmentation procedure):
Irina Kosinovsky, US Army CERL, Champaign, Illinois, USA
Dave Gerdes, US Army CERL, Champaign, Illinois, USA

Modifications for g3d library, geometric parameters, deviations:
Jaro Hofierka, GeoModel s.r.o., Bratislava, Slovakia, hofierka@geomodel.sk, http://www.geomodel.sk

REFERENCES

Hofierka J., Parajka J., Mitasova H., Mitas L., 2002, Multivariate Interpolation of Precipitation Using Regularized Spline with Tension. Transactions in GIS 6, pp. 135−150.

Mitas, L., Mitasova, H., 1999, Spatial Interpolation. In: P.Longley, M.F. Goodchild, D.J. Maguire, D.W.Rhind (Eds.), Geographical Information Systems: Principles, Techniques, Management and Applications, Wiley, pp.481−492

Mitas L., Brown W. M., Mitasova H., 1997, Role of dynamic cartography in simulations of landscape processes based on multi−variate fields. Computers and Geosciences, Vol. 23, No. 4, pp. 437−446 (includes CDROM and WWW: www.elsevier.nl/locate/cgvis)

Mitasova H., Mitas L., Brown W.M., D.P. Gerdes, I. Kosinovsky, Baker, T.1995, Modeling spatially and temporally distributed phenomena: New methods and tools for GRASS GIS. International Journal of GIS, 9 (4), special issue on Integrating GIS and Environmental modeling, 433−446.

Mitasova, H., Mitas, L., Brown, B., Kosinovsky, I., Baker, T., Gerdes, D. (1994): Multidimensional interpolation and visualization in GRASS GIS

Mitasova H. and Mitas L. 1993: Interpolation by Regularized Spline with Tension: I. Theory and Implementation, *Mathematical Geology* 25, 641−655.

Mitasova H. and Hofierka J. 1993: Interpolation by Regularized Spline with Tension: II. Application to Terrain Modeling and Surface Geometry Analysis, *Mathematical Geology* 25, 657−667.

Mitasova, H., 1992 : New capabilities for interpolation and topographic analysis in GRASS, GRASSclippings 6, No.2 (summer), p.13.

Wahba, G., 1990 : Spline Models for Observational Data, CNMS−NSF Regional Conference series in applied mathematics, 59, SIAM, Philadelphia, Pennsylvania.

Mitas, L., Mitasova H., 1988 : General variational approach to the interpolation problem, Computers and Mathematics with Applications 16, p. 983

Talmi, A. and Gilat, G., 1977 : Method for Smooth Approximation of Data, Journal of Computational Physics, 23, p.93−123.

Thorpe, J. A. (1979): Elementary Topics in Differential Geometry. Springer−Verlag, New York, pp. 6−94.

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

s.voronoi − Uses a GRASS sites list to produce a Voronoi diagram
(GRASS Sites Program)

SYNOPSIS

s.voronoi [**−aq**] **sites=***name* **vect=***name* [**catnum=***no|keep|gen*] [**labels=***no|cat|str,#|dec,#*]

DESCRIPTION

s.voronoi uses an existing sites list (**sites**) to create a Voronoi diagram (Thiessen polygons) in a binary vector file (**vect**).

OPTIONS

Parameters:

sites=*name*
: Name of existing sites file.

vect=*name*
: Name of the output file (binary vector file).

catnum=*no|keep|gen*
: What to use as category numbers for the vector output: *no* category numbers, *keep* from input file, or *gen*erate as serial number.
options: *no*, *keep*, or *gen*
default: *keep*

labels=*no|cat|str,#|dec,#*
: Site file field to use for generating the label. "#" is the *n*−th string or decimal field associated with the site record.
options: *no*; *cat*; *str,#*; or *dec,#*
default: *no*

Flags:

−a
: Use all sites found in the named sites file, rather than limiting output to sites falling within the current geographic region.

−q
: Quiet. Cut out the chatter.

NOTES

s.voronoi can be run either non−interactively or interactively. The program will be run non−interactively if the user specifies the name of an existing site list file and a name for a vect file, using the form

s.voronoi [−aq] sites=name1 vect=name2 catnum=optCatnum labels=optLabel

where name1 is the name of an existing site list file and name2 is the name of vector output file. optCatnum can be "keep" only if the site file category number is integer. optLabel can be "no", no labels; "cat", use site file category number as label; str,*num*, use the *num* string type field of the site file as label; or dec,*num*, use the *num* decimal type field of the site file as label.

Alternately, the user can simply type **s.voronoi** on the command line, without program arguments. In this case, the user will be prompted for parameter values using the standard GRASS parser interface described in the manual entry for parser.

EXAMPLES

To create a Voronoi diagram keeping the original category number and using the first decimal field as the output label use:
s.voronoi sites=*inputSite* vect=*voro* catnum=*keep* labels=*dec,1*

REFERENCES

Steve J. Fortune, (1987). A Sweepline Algorithm for Voronoi Diagrams, Algorithmica 2, 153−174.

SEE ALSO

v.autocorr, v.spag, v.support, s.delaunay, s.hull, s.sweep

AUTHOR

James Darrell McCauley, Purdue University
GRASS 5 update, improvements: Andrea Aime, Modena, Italy

Last changed: $Date: 2003/08/20 08:10:13 $

NAME

s.what − Allows the user to query site list descriptions.
(GRASS Sites Program)

SYNOPSIS

s.what
s.what help
s.what [**−q**] **map=***name*[,*name*,...] [**east_north=***east*,*north*[,*east*,*north*,...]

DESCRIPTION

s.what outputs the category values and (optionally) the category labels associated with user−specified locations on sites input map(s). Locations are specified as geographic x,y coordinate pairs (i.e., pair of eastings and northings).

OPTIONS

Flags:

−q
Load quietly

Parameters:

map=
The name(s) of one or more existing sites map layers to be queried.
east_north=
One or multiple coordinate pairs for query

EXAMPLES

The contents of the ASCII *inputfile* to *s.what* can be typed in at the keyboard, redirected from a file, or piped from another program (like *d.where*). Each line of the input consists of an easting, a northing, and an optional label, which are separated by spaces. The word **end** is typed to end input of coordinates to *s.what*. For example:

```
635342.21 7654321.09 site 1
653324.88 7563412.42 site 2
end
```

s.what output consists of the input geographic location and label, and, for each user−named sites map layer, the category value. Sample input (in regular font) to and output (in plain text) from *s.what* are given below.

```
s.what map=soils,aspect
635342.21 7654321.09 site 1
653324.88 7563412.42 site 2
end
sites in mapset 635342.21|7654321.09|site 1|45|21
sites in mapset 653324.88|7563412.42|site 2|44|20
```

NOTES

The maximum number of sites map layers that can be queried at one time is 14.

SEE ALSO

d.sites, *d.where*, *s.info*, *s.univar*, *parser*

AUTHOR

Huidae Cho <grass@geni.cemtlo.com>

Last changed: $Date: 2002/06/16 15:29:18 $

NAME

s.windavg − Average an attribute of a site_list for all the sites within each cell of the current region.
(GRASS Sites Program)

SYNOPSIS

s.windavg
s.windavg help
s.windavg [**−qzp**] **input=***name* **attr=***name* [**index=***value*] [**output=***name*]

DESCRIPTION

s.windavg reads a sites list averaging the selected attribute for each site that falls within each cell defined by the current region settings. The output gives the coordinates of the center of the cell and the average (mean) of the attribute values.

This program may be especially useful for generalizing data to a specific cell resolution.

OPTIONS

Flags:

−q
: Run quietly

−z
: Don't output cells without sites (averages of 0.0)

−p
: Create output in *gnuplot* format (overrides −z and writes to Standard Output)

Parameters:

input=*name*
: Name of the input sites list.

attr=*type*
: Site attribute type to average values over.
Options: *cat, dim, decimal, string*
Default: *cat*

index=*value*
: Site attribute index to average values over.
Ignored when the attribute is *cat*.
Default: 1

output=*name*

Optional name of sites list in which output of means will be written. Standard output is used if this is missing.

NOTES

Results are highly dependent on the settings of the current region. Only those sites within the current region will be considered. It may be desirable to decrease the cell resolution (make cells bigger) to get good results.

Processing time increases geometrically with increasing cell resolution (smaller cells). However, available memory should never be an issue.

SEE ALSO

g.region
s.rand
s.sample

BUGS

Please send all bug fixes and comments to the author or the grass development team.
`http://www.geog.uni-hannover/grass/`

AUTHOR

James Darrell McCauley <darrell@mccauley-usa.com>,
when he was at: Agricultural Engineering Purdue University

Updated for GRASS 5.0 by Eric G. Miller (30 Oct 2000)

Last changed: $Date: 2002/01/25 05:45:35 $