-

GRASS 4.0 Inference Engine: r.infer

Mary Martin
James Westervelt

U.S. Army Corps of Engineers
Construction Engineering Research Laboratory
Environmental Division
Spatial Analysis Systems Team
P.O. Box 9005
Champaign, IL 61826-9005

December 1991

Table of Contents

1. INTRODUCTION

1.1. Background

This document details the capabilities of **r.infer**, an inference engine included among Geographic Resources Analysis Support System (GRASS) software. GRASS is a public domain, image processing and geographic information system (GIS) originally developed by researchers in the Environmental Division of the U.S. Army Corps of Engineers Construction Engineering Research Laboratory (USACERL-EN) in Champaign, Illinois. The system is used to input, manipulate, analyze, and output geographic data by users in both military and non-military and public and private agencies based in North America, Europe, and other parts of the world. **r.infer** uses an expert-system approach and logic-based syntax to perform user-constructed analyses on map data in GRASS raster format.

Although most GRASS development has been conducted at USACERL, system integration, development, testing, distribution, training, and support is performed by numerous publicly- and privately-operated sites throughout the world. GRASS version 4.0 implemented significant additions and modifications to system libraries and programming code. Mechanisms are needed to transfer current technology and information about GRASS to user and development sites.

This document reflects modfications made to GRASS 4.0.

1.2. Objective

The objective of this work is to transfer knowledge of GRASS version 4.0 raster map analysis functions to the field. This document guides the user through the process of using **r.infer** to analyze raster data layers in GRASS using expert-system type rules.

1.3. Approach

Current GRASS **r.infer** functions are examined in this report.

1.4. Scope

This document discusses the 4.0-release version of the GRASS **r.infer** program, completed in Summer of 1991. Some elements of this document will be dated and inaccurate for post-4.0 software releases.

1.5. Mode of Technology Transfer

The information in this report will aid the technical transfer of USACERL’s GRASS image processing and geographic information system. GRASS is being transferred to the field through the following mechanisms: training programs, hands-on experience, a user support center, newsletters, extensive documentation, institutional structures at the Army and interagency levels, communication networks, and other forums.

User feedback on GRASS program capabilities, documentation, and other technology transfer mechanisms is important to the development of the system. Users are encouraged to communicate such feedback to GRASS development staff at USACERL, via existing electronic communication networks and via the GRASS Information Center at USACERL, P.O. Box 9005, Champaign, IL 61826-9005; phone 217-373-7220; fax 217-373-7222; or e-mail grassbug@zorro.cecer.army.mil.

2. ORIENTATION

This paper describes the use and syntax of the GRASS program **r.infer. r.infer** uses an expert system approach and logic-based syntax to perform analyses similar to those made by the GRASS program **r.combine**. Both programs allow the user to analyze and manipulate information contained in existing raster map layers, and use the results of these analyses to generate a new raster map layer. Unlike **r.combine**, however, **r.infer** is an "inference engine" designed to apply expert system type rules to the user's selected map layers, using a logical syntax comfortable for the user.

It is assumed that you are familiar with the nature of raster data and the concepts of a current location, mapset search path, geographic region settings, and masks, as these are used in GRASS. For a review of these concepts refer to *An Introduction to GRASS*.

Land analysts rely heavily on maps as a source of geographic information. This geographic information may be collected from many maps, of various scales and compiled from sources of differing dates. The land analyst therefore frequently attempts to create a new map layer which combines data extracted from numerous maps and contains only data relevant to the land analyst's current purpose. The GRASS program **r.infer** provides the user with a tool for combining and manipulating such geographic information, once it is entered into a GRASS database. Using **r.infer,** the land analyst can rapidly identify and highlight the presence of specified landcover characteristics.

r.infer is an inference engine which applies expert system type rules to a set of user-specified maps. The results are used to generate a new map in the user's current mapset under the name "infer."

Expert systems are simply sets of logical rules that mirror the analysis process of an expert in some field. For example, an expert system which might be used to locate suitable landfill sites contains questions that a waste disposal expert might ask of a landscape: Is the slope steep? Is the clay content of the soil sufficiently high to inhibit the leaching of landfilled materials into groundwater supplies? How far is the site from water supplies? Is the site within reasonable transport distance of the population centers generating the wastes? Etc. The line of questioning will take different courses, depending on the user's answers to questions along the way. An expert system embodies these lines of reasoning in a data file. The program used to work its way through the questions and answers is called an inference engine. **r.infer** is such a program. An expert system asks a user to answer simple yes/no questions. **r.infer** instead asks these questions directly of the map database. Each individual map layer cell in turn becomes the "user" answering questions about its contents. Results of this questioning are stored in a new raster map layer.

Rules are provided to **r.infer** via standard input. The user may type these rules into a data file during a GRASS session, but more frequently will create this data file prior to entering GRASS using a system editor (e.g., the UNIX text editor *vi*).

r.infer differs from the GRASS program **r.combine** in its more pleasing syntax and approach. Like **r.combine, r.infer** requires the user to determine only whether or not a given land characteristic is good or bad for the user's purpose; another similar GRASS program called **r.weight** requires the user to assign numeric weights to each map category quantifying gradations of goodness and badness. Although **r.weight** allows the user to perform more detailed analyses of locational suitabilities than do either **r.infer** or **r.combine,** the latter programs are more easily used and their outputs more easily understood. Of the three programs, only **r.infer** attempts to apply an expert system approach to geographic analysis. Numerous other programs are available in GRASS to manipulate raster map data. Refer to the manual entries for **r.mapcalc**, **r.mfilter**, **r.neighbors**, **r.surf.idw**, and **r.watershed**, for examples of these.

3. SYNTAX

The user enters a series of logical statements into a data file which is subsequently run by **r.infer.** Each logical statement is composed of one or more *conditions* followed by either a *hypothesis* or a *conclusion.* In the jargon of the field, conditions are called *antecedents* while hypotheses and conclusions are termed *consequences.* Suppose we wish to determine locations with our map database which meet the following conditions.

If a cell contains category values 6, 7, or 8 in the map layer *landuse*, and also contains category values 1 or 2 in the map layer *soils.Tfactor*, but does not contain category values 10 through 89 in the *slope* map layer, then assign that cell a category value of 3 (with the category label ‘‘Suitable Location") in a new map layer called *infer.*

The **r.infer** equivalent of this statement is:

```
IFMAP        landuse 6-8
ANDIFMAP     soils.Tfactor 1 2
ANDNOTMAP    slope 10-89
THENMAPHYP   3 Suitable Location
```

The first three lines in the above statement state the map conditions that must be met to reach the map conclusion stated on the fourth line. This shorthand allows for the quick entry of quite involved and complex logic branches using many statements resulting in the possibility of many different conclusions.

Each logical statement in **r.infer** is composed of one or more conditions followed by a consequence. The user’s input to **r.infer** can contain a series of such logical statements. This input file must contain at least one Map Condition and at least one Map Conclusion. It can also contain Statement Conditions and Statement Hypotheses. Map Conditions are questions about the presence or absence of stated categories in specified GRASS raster map layers. Map Conclusions designate the category to be assigned to the cells which meet stated conditions in the new raster map layer created by **r.infer**. Statement Conditions are similar to Map Conditions, but are used to refer to Statement Hypotheses, rather than to actual map layers. Statement Hypotheses are used to embody the truth of a set of conditions, without actually assigning a value to cells which meet these conditions in the new map layer *infer*. The **r.infer** language uses its own words to form these statements; descriptions of these words follow:

1. Map Conditions:

IFMAP
checks for the existence of stated contents. In the example above, IFMAP checks to see if each cell in the raster map layer *landuse* contains categories 6, 7, or 8.

IFNOTMAP
is similar to IFMAP, but checks for the existence of the inverse of the stated contents (i.e., it checks the cell for the absence of the stated map layer category values).

2. Map Conclusions:

THENMAPHYP
When this conclusion statement is reached, application of the rules to the data ceases, the current cell is assigned the stated category value in the new growing map layer, and the trailing text is saved for the creation of the category labels for the new map layer. In the above example, if the current cell met all

stated conditions, the line:

THENMAPHYP 3 Suitable Location

would determine that the current cell would be assigned a category value of 3 and analysis would stop for that cell. The trailing label text ‘‘Suitable Location" would be entered beside category 3 in the category support file. The inference engine then moves on to the next cell in the map layer and starts the analysis again from the top of the logical statement.

3. Statement Hypotheses:

THEN
Instead of making a Map Conclusion after successfully meeting the stated conditions (antecedents), a statement may instead be assigned a Truth Condition. In the above example, we might have said

THEN Suitable Location

(rather than *THENMAPHYP 3 Suitable Location*), and added additional conditions beneath this THEN statement.

Use of a THEN statement does not result in the assignment of a specified category value to the current cell in the new map. Instead, the hypothesis ‘‘Suitable Location" will be set to the truth condition "true," and analysis of the current cell will continue; succeeding rules will continue to be applied to this current cell until a THENMAPHYP statement is reached. The hypothesis made using the THEN statement can be used in conditional statements using the keywords below. This means that the phrase named by the THEN statement (here, "Suitable Location") can be used further down in the logical statement to refer to the fulfillment of the set of conditions specified by the THEN statement (here, that the cell contains *landuse* category values 6, 7 or 8, and *soils.Tfactor* category values 1 or 2, and does not contain *slope* category values 10 through 89).

4. Statement Conditions:

IF
Hypothesis strings, the truth of which can be determined in other statements which use the **THEN** conclusion keyword, can be used with these condition keywords. Our above example,

```
IFMAP        landuse 6-8
ANDIFMAP     soils.Tfactor 1 2
ANDNOTMAP    slope 10-89
THENMAPHYP   3 Suitable Location
```

could be rewritten to make use of the **IF** keyword:

```
IFMAP        landuse 6-8
THEN         suitable landuse
!
IFMAP        soils.Tfactor 1 2
ANDNOTMAP    slope 10-89
THEN         erosion limited
!
IF           suitable landuse
ANDIF        erosion limited
THENMAPHYP   3 Suitable Location
```

IFNOT
states the inverse of that which is stated.

It should be noted that the spacing of words on a line in the data file input to **r.infer** shown above was done for clarity; it is not necessary that keywords be separated by more than one space from either map layer names, category designations, or truth conditions. It is necessary that map layer names (or category values) be separated from map layer category values by exactly one space on a given line. Exclamation points identify comments which will be ignored by **r.infer,** but which may greatly contribute to the readability of your logical statement.

4. ALIASES

Several of the above keywords (used to express antecedents and consequences) have aliases with which they are interchangeable. All keywords must be capitalized. The following table lists those keyword aliases that are currently available:

KEYWORD:	Can be replaced with:	
Map Conditions:		
IFMAP	ANDIFMAP	ANDMAP
IFNOTMAP	ANDNOTMAP	
Map Conclusions:		
THENMAPHYP		
Statement Hypotheses:		
THEN	ANDTHEN	
Statement Conditions:		
IF	AND	ANDIF
IFNOT	ANDNOT	

5. SAMPLE SESSION

This sample session can be replicated by users who run GRASS and choose *spearfish* as their current GRASS location.

Normally, before entering the GRASS session in which the user actually executes the relevant commands, the user will already have made a list of the map layers relevant to a specific problem, and will have created a set of logical statements to be input to **r.infer** designed to answer each map layer's ability to solve this problem. However, here we will both construct and run the analysis within a single GRASS session, proceeding one step at a time. This will entail the

1. Statement of the Problem
2. Invocation of the Command-Driven Version of GRASS
3. Examination of Available Map Layers and Map Layer Categories
4. Construction of Logical Statements
5. Execution of **r.infer**
6. Examination of the Resultant Map Layer Created
7. Exiting of GRASS.

Remember that **r.infer** will *run* your analysis, but it will *not construct* that analysis. Before running **r.infer** you must have a clear understanding of the problem you wish to analyze, and know how to translate this problem into the syntax accepted by **r.infer.** You, not **r.infer,** construct the line of reasoning (in the form of a logical statement) that is entered into a data file and used to analyze selected map layers. You determine which map layers and map layer categories are relevant to your analysis.

Commands that you are requested to enter are displayed in bold in the manner shown below:

PROMPT:> **my_command**

Results that you should see after typing a command are displayed like this:

```
Some result displayed here
```

Let us begin. This sample session with **r.infer** can be reproduced on any normal text terminal. **r.infer,** unlike **r.combine** and **r.weight,** does not print map layers out to the screen as it processes geographic data. However, if you wish to see the map layer created as a result of your **r.infer** run, you can simply use the GRASS program **r.combine,** which prints numeric maps to the screen. On color graphics monitors, these numeric maps are turned into color maps. If the color graphics monitor is used, you can display your **r.infer** map in color using such GRASS programs as **d.display** and **d.rast** (as well as **r.combine**).

5.1. Statement of the Problem

Let us assume we wish to examine the suitability of various portions of the *spearfish* database for the construction of an electric steam-generating plant. We now require a set of rules which an expert in the field of power plant siting might use to analyze landscape features for their suitability for this purpose. We can state that we require a suitable construction site (implying a site with relatively deep depth to bedrock, little slope, good soils, the absence of existing fixed structures, and the absence of flooding on the site itself). We also require a site that is close to a flowing water source which can supply water to the plant, and would like the site to be near a reliable transportation network.

We will now enter GRASS, and subsequently create a data file containing these site requirements after having translated them into a logical statement interpretable by **r.infer.**

5.2. Entering GRASS

Users should now enter GRASS. The below sample session assumes that users are running GRASS version 4.0. If you are running a version of GRASS previous to 4.0, note that **r.infer** must be accessed through the quick access (not the menu driven) GRASS.

Once you have entered GRASS by invoking the command **grass4.0,**

```
PROMPT:>  grass4.0
```

you must declare to GRASS the LOCATION_NAME, MAPSET, and DATABASE on which you wish to work. To duplicate this session, you must request LOCATION_NAME *spearfish.* The mapset name you choose is arbitrary, but must be a one-word name less than 14 characters long. A default database directory will be presented to you.

5.3. Examination of Map Layers and Map Layer Categories

You have formulated a problem statement and invoked (command driven) GRASS. You will now wish to examine available map layers and map layer categories within the location *spearfish* that appear relevant to your problem. To do this, we will make brief use of the GRASS program **r.weight.** Enter the command **r.weight** to begin your **r.weight** session.

```
GRASS 4.0 >  r.weight
```

You will be asked:

```
Do you want graphics to go to the color monitor? (y/n) [n] >
```

Figure 1

Press RETURN or enter the command **n** here, because we have no **r.infer**-created map as yet to display, and do not now intend to display other maps. After your response is entered a screen with general information concerning **r.weight** will appear. **r.weight** is a language-driven program, whose use is explained in the *GRASS Tutorial: r.weight*. Here, we will assume the user is already familiar with basic **r.weight** commands. We are in any case only currently interested in using **r.weight** to obtain a listing of available map layers and the categories within each map layer. A listing of available map layers can be obtained without entering **r.weight,** simply by entering the command

```
GRASS 4.0 >  g.list rast
```

From within **r.weight,** the command **list maps** will produce the same results. Try this:

```
r.weight:>  list maps
```

```
RASTER files available in mapset PERMANENT:
aspect        mss.image     rushmore        soils.ph         trn.sites
bugsites      owner         slope           soils.range      vegcover
density       quads         slope.7         soils.texture
elevation     railroads     soils           streams
geology       roads         soils.Tfactor   strm.dist
landuse       rstrct.areas  soils.br.depth  transport.misc
Hit RETURN to Continue-->
```

Figure 2

Hit RETURN to return to the **r.weight** prompt.

Given the problem you wish to analyze, it appears that the *spearfish* map layers *soils.br.depth, soils.texture, slope, streams, transport.misc, roads,* and *railroads* will be useful to your analysis. You will now wish to obtain a listing of the categories within each of the map layers you want included in your analysis. One convenient way to obtain such category listings for particular map layers is through use of the **r.weight** command,

```
r.weight:>  list categories map_name
```

where *map_name* is the name of an available map layer in the database. Here, we will obtain listings of the categories present in the *spearfish* map layers noted above, by invoking the following series of commands:

```
r.weight:>  list categories soils.br.depth
r.weight:>  list categories soils.texture
r.weight:>  list categories slope
r.weight:>  list categories streams
r.weight:>  list categories transport.misc
r.weight:>  list categories roads
r.weight:>  list categories railroads
```

The map layer category listings output by these commands are summarized in Figure 3.

```
---------------------------------------------------------------------

map|       Slope             Soils Texture         Streams

---------------------------------------------------------------------
 c | 0 No Data             0 No Data             0 No Data
 a | 1 0 degrees           1 Loam                1 Perennial Stream
 t | 2 1 degree            2 Stony loam          2 Intermittent Stream
 e | 3 2 degrees           3 Silty loam          3 Aqueduct
 g | 4 3 degrees           4 Silt loam, loam     4 Shoreline
 o | 5 4 degrees           5 Fine sandy loam
 r | 6 5 degrees           6 Loam, fine sandy loam
 y | 7 6 degrees           7 Loam, very fine sandy loam
   | 8 7 degrees           8 Gravelly loam
   | 9 8 degrees           9 Gravelly silt loam
   |10 9 degrees          10 Channery silt loam, loam
   |11 10 degrees         11 Cobbly loam
   |12 11 degrees         12 Very gravelly silt loam,
   |                           gravelly silt loam,silt loam
   |13 12 degrees         13 Clay
   |14 13 degrees         14 Clay loam
   |15 14 degrees         15 Silt clay loam
   |16 15 degrees         16 Silty clay loam, clay
   |17 16 degrees         17 Silty clay loam, silty clay
   |18 17 degrees
   |19 18 degrees         (...More Slope:)
   |20 19 degrees         39 38 degrees
   |21 20 degrees         40 39 degrees
   |21 21 degrees         41 40 degrees
   |22 22 degrees         42 41 degrees
   |23 23 degrees         43 42 degrees
   |24 24 degrees         44 43 degrees
   |25 25 degrees         45 44 degrees
   |26 26 degrees         46 45 degrees
   |27 27 degrees         47 46 degrees
   |28 28 degrees         48 47 degrees
   |30 29 degrees         50 49 degrees
   |31 30 degrees         53 52 degrees
   |32 31 degrees         54 53 degrees
   |33 32 degrees         82 81 degrees
   |34 33 degrees         83 82 degrees
   |35 34 degrees         84 83 degrees
   |36 35 degrees         85 84 degrees
   |37 36 degrees         87 86 degrees
   |38 37 degrees         88 87 degrees
   |                      89 88 degrees

------------------------------------------------------------------
```

```
----------------------------------------------------------------
map:| Miscellaneous        Roads                       Railroads
    | Transportation
    | Features
----------------------------------------------------------------
  c | 0 No data            0 No data                   0 No data
  a | 1 Power transmission 1 Primary route, undivided  1 railroad
    |   line
  t | 2 Landing strip      2 Road or Street, class 3   2
  e |                      3 Road or Street, class 4
  g |                      4 Trail, class 4, non-
                             4-wheel drive vehicle
  o |                      5 Cloverleaf or Exchange
  r |
  y |

----------------------------------------------------------------


----------------------------------------------------------------
map:| Depth to Bedrock
----------------------------------------------------------------
  c | 0 No data
  a | 1 < 14 inches
  t | 2 > = 14 and < 20 inches
  e | 3 > = 20 and < 40 inches
  g | 4 > = 40 and < 60 inches
  o | 5 > = 60 inches
  r |
  y |
----------------------------------------------------------------
```

Figure 3

It should be noted that you need not use map layers present within the PERMANENT mapset to run **r.infer;** you might use other raster map layers from other GRASS locations and mapsets in your mapset search path, including map layers you have generated using programs like **r.combine** and **r.weight** (refer to GRASS **g.mapsets** command).

Having obtained a listing of relevant raster map layer categories, you are ready to leave **r.weight.** To exit **r.weight,** use the command **exit**.

r.weight:> **exit**

You now exit the r.weight program and are returned to your GRASS shell prompt.

5.4. Construction of Logical Statements

Using the syntax rules described earlier, we can translate our problem statement into the following logical statement. **r.infer** will later apply the rules embodied in this logical statement to each cell in the map layers specified. You can use your system editor to create a new data file containing this logical statement. Do this now, inserting the text shown below:

```
!
IFMAP streams 1 3 4
THENMAPHYP 1 Water Present
!
IFMAP roads 1 2 5
THENMAPHYP 2 Transport Lines Available
!
IFMAP transport.misc 2
THENMAPHYP 2 Transport Lines Available
!
IFMAP railroads 1
THENMAPHYP 2 Transport Lines Available
!
IFMAP soils.br.depth 3-5
ANDIFMAP soils.texture 1-7
ANDNOTMAP slope 0 9-89
THENMAPHYP 3 Suitable for Construction
```

5.5. Execution of r.infer

Having constructed a data file containing the set of rules by which you will analyze the suitability of specified landscape features to support the construction of an electric steam-generating plant, you now wish the inference engine **r.infer** to evaluate the ability of each *spearfish* cell to satisfy these requirements. However, you feel a bit queasy about your newly constructed logical statement. Does it contain the questions you think it contains? Will the line of reasoning embodied in the data file follow the courses you expect it to take under all possible sets of circumstances? You can alleviate this uncertainty by testing your logical statement, using the command:

```
GRASS 4.0 > r.infer -t filename
```

where *filename* is the name of your data file containing the logical statement described above, and - *t* denotes the test option available with **r.infer.** (Note that you must also specify the full pathname location of this filename if it is not in your current working directory.)

```
Compiling input rules from file /fullpath/filename

mark_cats (6-8)
mark_cats (1-1)
mark_cats (2-2)
mark_cats (9-89)

Maps used:
Map [slope] in mapset [PERMANENT]
Map [soils.texture] in mapset [PERMANENT]
Map [soils.br.depth] in mapset [PERMANENT]
Map [railroads] in mapset [PERMANENT]
Map [transport.misc] in mapset [PERMANENT]
Map [roads] in mapset [PERMANENT]
Map [streams] in mapset [PERMANENT]
Is the following statement true?
streams 1 3 4 :
Is the following statement true?
roads 1 2 5 :
Is the following statement true?
transport.misc 2 :
Is the following statement true?
railroads 1 :
Is the following statement true?
soils.br.depth 3 4 5 :
Is the following statement true?
soils.texture 1 2 3 4 5 6 7 :
Is the following statement true?
slope 0 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 50 53 54 82 82 84 85
87 88 89:
________________________________________________________________
New cell value =

Mapset <mapset-name> in Location <spearfish>
GRASS 4.0 >
________________________________________________________________
```

Figure 4

Enter either **yes** or **no** (or **true** or **false**) as to the truth of each statement. The new cell value shown should vary between zero and three, depending on your answers to each of these questions. A cell category value of zero implies the user (i.e., the map layer cell) has failed to satisfy at least one of the sets of map conditions tested for within the data file. **r.infer** works from the top of your logical statement to its bottom; as soon as you satisfy all map conditions necessary to making a map conclusion, a new cell category value will appear. You will not then necessarily be asked all of the questions present in the data file; you will only continue to be asked questions until a conclusion about the cell's value can be drawn. Given the site requirements we earlier stated within our problem statement, it is therefore important that we ordered questions within the data file as we did. For example, we sought a site that was both close to water sources, but did not itself often flood. It was therefore necessary to identify cells containing or adjacent to water, prior to identifying sites ‘‘suitable for construction." It is necessarily implied that those sites assigned to category 3 (Suitable for Construction) do not satisfy the conditions necessary to be assigned to category 1 (Water Present). Had they fallen into category 1, the analysis of the cell would have ceased, and the cell been assigned to category 1 without progressing further down through the data file to see if the cell also satisfied the conditions necessary to be classed in category 3.

Again invoke the command:

GRASS 4.0 > **r.infer -t filename**

This time entering different responses (**yes** or **no**) along the way. Again, check to make sure that the New Cell Value output as a result of your responses matches up with what you intended to occur under such circumstances. Do this repeatedly, entering different answers as to the truth of the questions posed you each time, until you are comfortable that the logical statement contained in your data file *filename* will indeed output the results you expect.

Thus far, you have only tested your data file contents, without producing any new raster map layer. After you are satisfied with your data file, you are ready to actually run **r.infer.** Do this now, by simply entering the command:

GRASS 4.0 > **r.infer filename**

```
Compiling input rules from file /pathname/filename

mark_cats (6-8)
mark_cats (1-1)
mark_cats (2-2)
mark_cats (9-89)

Map [slope] in mapset [PERMANENT] opened
Map [soils.texture] in mapset [PERMANENT] opened
Map [soils.br.depth] in mapset [PERMANENT] opened
Map [railroads] in mapset [PERMANENT] opened
Map [transport.misc] in mapset [PERMANENT] opened
Map [roads] in mapset [PERMANENT] opened
Map [streams] in mapset [PERMANENT] opened
Working to row 280
At row: 280
Creating support files
Mapset <mapset-name> in Location <spearfish>
GRASS 4.0 >
```

Figure 5

If you now type,

GRASS 4.0 > **g.list rast**

your new map should appear as a raster map layer called *infer* stored within your current GRASS MAPSET:

```
RASTER files available in mapset <mapset-name>:
infer

RASTER files available in mapset PERMANENT:
aspect          mss.image       rushmore        soils.ph        trn.sites
bugsites        owner           slope           soils.range     vegcover
density         quads           slope.7         soils.texture
elevation       railroads       soils           streams
geology         roads           soils.Tfactor   strm.dist
landuse         rstrct.areas    soils.br.depth  transport.misc
 Hit RETURN to Continue-->
```

Figure 6

Hit the RETURN key to return to the GRASS 4.0 prompt. You can display your raster map layer on a color graphics monitor after running the GRASS program **r.support**. All support files except a color file will already have been created for you by **r.infer**. You can also modify the cell header, category, miscellaneous, and history files associated with your new raster data file by running **r.support** if desired.

5.6. Examination of the Resultant Raster Map Layer

Now that you have successfully created the raster map layer *infer* containing the results of your analysis, and have created a color table for this map by running **r.support**, you can display or print out this map layer. If you have a color graphics monitor, you can display your map using GRASS programs like **d.rast** or **d.display**. If you do not have a color graphics monitor, you can display a numeric representation of your map layer on your dumb terminal, by using a GRASS program like **r.combine**. This is demonstrated below. If your system is hooked up to a printer, you can print out your map using **p.map**.

Let us take a brief look at a numeric representation of your new raster map layer within **r.combine.** Enter the following command to begin your session.

```
GRASS 4.0 >  r.combine
```

```
Do you want graphics to go to the color monitor? (y/n) [n] >
```

If you are not working with a color monitor, enter "no" at this prompt. (Note that if you are working with a color graphics monitor, you can more easily display your new raster map layer *infer* using such GRASS programs as **d.display** and **d.rast**.)

```
Analyses will be displayed with symbols during execution.

Help is available for r.combine:  enter (help)
```

Let us look at our new map layer named *infer*. A correct and complete **r.combine** request is called a map expression (EXPR). A viable data layer name is, by itself, an EXPR.

```
[1]:> (infer)
```

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0000000000000000000000000000000000000000000000000002200000000000
0000020000000000000000000000000000000000000000000002000000000000
0100000200000002000300000000000000000220000002000000000020111010
0001020000220020000000020000000000000000000000000000000222200000
0001020000020000000000000000000000000330000000000000000000000200
0000000000200000000000000002030000030200000000000000000000002000
0010000000020000000000000003000000000002000000000000000200000020
0100000000000000000000000000000000000000000000000000000000000000
0000000000000020000000000000000000000000000000000000000000022010
0000000000000220000000000000000000000000000000000000000000000100
0000000000000000000000000000000000000000000000000000000000000000
0000000000002000000000000000000000000000000000100000000110000000
0000000000002000000000000000000000000000000000000000000000000000
0000000000000200000000000000000000000000200000000000000000000000
0000000000000000020000000000000002000020200000000000000000000000
0000000000000000000000000000002000000000000200000000000200000000
0000000000000020000020000000002200000001200000000000000000000000
0000000000000000000000000020000000020200020000000200000000000000
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

It should be emphasized that the raster map layer *infer* in which your results are stored will be overwritten each time you run **r.infer.** If you wish to save a particular map layer created using **r.infer,** you should therefore remember to rename this raster map layer before running a different **r.infer** analysis. The GRASS command **g.rename** can be used when you are either outside or inside of **r.combine** to change the name of the raster map layer *infer*. When outside of the **r.combine** program (at the GRASS 4.0 prompt), enter the command **g.rename:**

```
GRASS 4.0 >  g.rename
```

This command can conveniently be run from within **r.combine** by preceding the command with the shell escape character **!** (an exclamation point):

```
[2]:>  !g.rename
```

After invoking either command, the user is prompted for the type of file to be renamed (here, a raster data file), the existing map name the user wishes to change (*infer*), and the new map name to which the user desires to change it.

At this point, we might conveniently edit and run additional data files through **r.infer** while remaining within **r.combine.** As shown above, commands that can be run from within GRASS can also be run from within **r.combine,** by preceding command names with the shell escape character **!** (refer to the *GRASS Tutorial: r.combine* for further instructions on the use of **r.combine**).

When editing logical statements, the user should be careful to specify the full path name of the directory in which the user wants to place the data file. Similarly, when running **r.infer** from within **r.combine,** the user should specify the full path name of the directory in which the user's **r.infer** data file appears. For example, the following commands would allow a user to first edit an input file called *test.1* located in the directory specified by the path name */usr/cerl/login-name* using the visual editor *vi*, and subsequently run this data file through **r.infer**:

```
[3]:>  !vi /usr/cerl/login-name/test.1
[4]:>  !r.infer /usr/cerl/login-name/test.1
```

r.combine has now demonstrated its usefulness. You should give the following command to leave **r.combine.**

```
[5]:> BYE
```

5.7. Exiting GRASS

At this point, you might also edit and execute additional logical statements through **r.infer.** Now, however, we will exit GRASS entirely. When you then exit GRASS with the **exit** command you will be given the opportunity to save or remove the new maps you have made using **r.infer:**

```
GRASS 4.0 > exit
```

Unless your new maps must be saved, it is always best to remove them at this point.

6. CONCLUSIONS AND RECOMMENDATIONS

This manual is part of an ongoing process to document Geographic Resources Analysis Support System (GRASS) program functions. It is one of many technology transfer mechanisms being implemented to explain current, and future, GRASS program capabilities to resource managers and system users.

Documentation, along with training programs, hands-on use, a user-support center, newsletters, institutional structures at the Army and interagency levels, communication networks, and other forums, can be used to aid this ongoing technology transfer effort.

User feedback on GRASS program capabilities, documentation, and other technology transfer mechanisms is also needed. Users are encouraged to communicate such feedback to GRASS development staff at USACERL, via existing electronic communication networks and via the GRASS Information Center at USACERL, P.O. Box 9005, Champaign, IL 61826-9005; phone 217-373-7220; fax 217-373-7222; or e-mail: grassbug@zorro.cecer.army.mil.

REFERENCES

Ruiz, Marilyn S., and Jean M. Messersmith, *Cartographic Issues in the Development of a Digital GRASS Database*, USACERL Special Report N-90/16 (USACERL, September 1990).

Westervelt, James, and William D. Goran, *Introduction to GRASS 4*, Technical Manuscript N-92/xx (USACERL, xx 1992).

Westervelt, James, Mary Martin, and Deborah Brinegar, *GRASS 4.0 Programs*, USACERL Technical Manuscript N-92/xx (U.S. Army Construction Engineering Research Laboratory [USACERL], xx 1992).

Westervelt, James, Michael Shapiro, William D. Goran, et al., *GRASS User's Reference Manual*, ADP Report N-87/22 rev (USACERL, September 1988, last revised December 1991).