Performing Boolean Operations in GRASS 4.0: r.combine Program Tutorial

L. Van Warren
James Westervelt
Scott Tweddale

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

Table of Contents 2
1. Introduction 3
2. Orientation 4
3. Getting Out 4
4. EXPR: The Map Expression 4
5. Aliases 9
6. Sample Session 10
7. Conclusions and Recommendations 14
References 15

1. INTRODUCTION

1.1. Background

This document details the capabilities of **r.combine**, a program that allows the user to manipulate and create raster map data using Boolean operations, 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 nonmilitary and public and private agencies based in North America, Europe, and other parts of the world. **r.combine** allows the user to construct and execute analyses on map data in GRASS raster format using Boolean operations.

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.

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.combine** to manipulate and analyze raster data layers in GRASS using Boolean operators.

1.3. Approach

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

1.4. Scope

This document discusses the 4.0-release version of the GRASS **r.combine** program, completed in Summer, 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 usage and syntactic details of a GRASS program called **r.combine**. **r.combine** is geared for the manipulation and analysis of raster maps and their components. Parts of available maps can be combined with the operators: GROUP, AND, OR, NOT, OVER, and COVER. Using **r.combine,** the user can freely build maps highlighting chosen characteristics and combinations of characteristics.

It is assumed that you are familiar with raster and vector map layer formats, GRASS mapsets and locations, regions, and masks. For a review of these concepts refer to the GRASS hardcopy or on-line GRASS help facility in *S**** *on Demand: Help for GRASS Users* and the GRASS **g.help** command.

GRASS provides you with programs for combining, manipulating, and displaying geographic information. Projects requiring the analysis of land areas rely heavily on maps to supply geographic information. The geographic information sought for a specific project may be divided among many maps, which were drawn at various scales and prepared on different dates. The land analyst therefore frequently attempts to sketch a new map which combines data extracted from numerous maps and contains only data relevant to a specific purpose. **r.combine,** a program within GRASS, replaces the sketch pads. Using **r.combine**'s simple yet powerful language, the land analyst can isolate and highlight desired information in a way that elicits quick yet accurate answers.

r.combine allows the user to combine information from several map layers by using boolean operators like AND, OR, NOT, and GROUP. Use of these boolean operators lets the user combine information existing on several different map layers within a single, new map layer. **r.combine** differs from the GRASS program *r.weight* by being somewhat simpler and requiring the user to make fewer value judgements. **r.combine** requires the user to determine only whether or not a given land characteristic is good or bad for the user's purpose; *r.weight* requires the user to assign numeric weights to each map category quantifying gradations of good and bad. While *r.weight* allows more precise analyses of locational suitabilities to be made, *g.weight*'s analyses are more difficult to create and to understand than are those of **r.combine.**

3. GETTING OUT

Before getting too involved, it is useful to know how to exit **r.combine.** This can be done at any time by typing:

BYE

4. EXPR: The Map Expression

r.combine's basic input command is the map-expression (EXPR). An EXPR is either a map layer already existing in the database, or is a new map layer resulting from the user's use of a mathematical (boolean) operation performed on one or two other EXPRs. The boolean operation words are called OPERATORS. Note that these operators can be typed in either uppercase or lowercase letters.

OPERATORS used with EXPRs:

AND	Identify locations where two requests are true.
OR	Identify locations where one or both requests are true.
GROUP	Identify locations that contain one of a set of map categories.
NAME	Save the results of a EXPR in a new map layer for later use.
OVER	Transparently overlay the result of a EXPR on named saved map layer.
COVER	Opaquely overlay the result of a EXPR on a named saved map layer.
NOT	Negate an EXPR to identify areas not containing a set of map categories.

4.1. MAP LAYER NAME

Let us try some examples of some basic EXPRs. The simplest is to enter the name of an existing and available map layer enclosed in parentheses (all EXPRs are enclosed in parentheses):

(slope)

Assuming the map layer slope is available, **r.combine** will recognize that simple command as a valid and complete EXPR. Once **r.combine** sees the end of the first EXPR it goes to work trying to fulfill your request. The command

(slope)

will result in your region of the slope map layer being displayed on the color monitor. (Note: Most EXPRs result in a boolean, or yes-no, map. In a few special cases, such as this expression, the map categories are retained, resulting in the production of a multicolored map.)

4.2. (AND EXPR1 EXPR2)

This example of the use of the AND operator demonstrates that EXPRs are arguments to other EXPRs. Consider the example:

(AND (soils) (slope))

Here, the data layer "soils" is EXPR1 while "slope" is EXPR2. The resulting complete EXPR will result in a single value (single color) map displayed on the monitor, showing all the locations where *both* the soil type and the degrees of slope are defined. Note that this resulting map is represented in a single color, unlike the multicolor map of the "(slope)" example above. To explain this, we will have to suspend our discussion of the AND operator for a moment. Entering the EXPR above:

(slope)

results in a multivalue (multicolor) map display. Similarly, substituting "soils" for "slope" will result in a similar multivalue display. To combine the two maps using an operator (e.g., AND) and still retain information about the original values associated with each location, would require too many values. For example, consider a soils map with 54 different categories, and a slope map with 58 categories (and category range 0-89), as is the case in the sample *spearfish* database provided with your GRASS code. Combining the categories of these two map layers in all possible ways results in a new map, potentially containing a total of 54x58=3132 new categories. GRASS version 4.0 will easily accommodate this number of categories. However, it is questionable that more than 256 colors easily distinguishable on a computer screen by human eyes could be created.

To make life manageable, these boolean operators therefore treat any non-zero value in a map layer as a one (as "YES," or "TRUE"). Soil categories 1 through 54 are then all considered "TRUE" (because all contain defined soil types). The example AND command above conceptually takes the soils map and turns it into a TRUE/FALSE map. Grid cell locations which contain defined soils types are assigned the value TRUE in the new TRUE/FALSE map; grid cell locations that do not have defined soil types in the soils layer (i.e., locations with soils category 0) are assigned the value FALSE in the new TRUE/FALSE map. The same is done with the slope map. The AND operator in the above command:

(AND (soils) (slope))

deals with only TRUES and FALSES (ones and zeros, really), and produces an output map layer containing only categories one and zero.

Hence, the result will be a map showing all the ("TRUE") grid cell locations that have been assigned both a soils category *and* a slope category.

4.3. (OR EXPR1 EXPR2)

The OR operator works very much like the AND operator above. If we substitute ‘‘OR" for ‘‘AND" in the above example, we obtain the valid EXPR:

(OR (soils) (slope))

The result will be a map showing all the locations that have been assigned a soil category and/or a slope category. Note that this OR operator allows for BOTH the assignment of a soil AND a slope category, as well as for the presence of EITHER category alone.

4.4. (GROUP NUMBERS EXPR)

The above examples for AND and OR are very restrictive in that they treat their map layers (EXPRs) as if they contained only two types of data (TRUES (ones), or FALSES (zeros)). Both the slope and the soils map layers really contain many more types of much more interesting information. The GROUP operator allows you to specify which of all the available categories in a map layer are of interest to you. For example:

(GROUP 8 (soils))

(GROUP 8 11 (soils))

(GROUP 1-8 11 (soils))

are all valid EXPRs. The first one says that you wish to consider only map category 8 in the map layer ‘‘soils." The second one, that you wish to consider soils categories 8 and 11; and the third, soils categories 1 through 8 and 11. Each of the three resulting maps displayed will show only those locations where the soils data layer contains soils types corresponding to the relevant category numbers.

Let us construct a more interesting example. Assume that you would like to see all the grid cell locations that contain soil types 8 through 11 that also contain slope categories 3, 5, and 7. The following single command should do the trick:

(AND (GROUP 8-11 (soils)) (GROUP 3 5 7 (slope)))

The above example begins to reveal **r.combine**’s ability to nest EXPRs within EXPRs within EXPRs, ad infinitum. Note that the EXPR ‘‘soils" has been replaced with ‘‘(GROUP 8-11 soils)." This ability to embed EXPRs within EXPRs can lead to very lengthy and complex analyses. Note that only one map is displayed on your monitor, not three (although there are really three EXPRs contained within the above command). Only the newly combined EXPR is displayed. For readability, you are allowed to change the above to:

```
(AND
    (GROUP 8-11 (soils))
    (GROUP 3 5 7 (slope)) )
```

r.combine ignores extra spaces, tabs, and newlines.

For practice, work through the following analysis request:

```
(OR
   (AND
      (GROUP 1-3 (roads))
      (GROUP 5-7 (slope))
   )
   (AND
      (GROUP 1-3 (roads))
      (GROUP 2-6 (vegcover))
   )
)
```

You should begin to see some interesting applications.

4.5. (NAME MAP_NAME EXPR)

Up to this point, all the results have been mapped to your display device only. No retrievable record has been made of your work. (Actually, this is good because these maps generally consume a substantial amount of computer memory!) But, now and again you will create a result that you need saved for future reference. To satisfy this need, the operator ‘‘NAME" is available. Let us say you are again GROUPing soils categories, and want to save the results. The following command sequence will store this soils grouping in a new map called ‘‘mymap":

```
(name mymap (GROUP 8-10 (soils)))
```

You can then reference the newly created "mymap" map layer like any other map layer. Before **r.combine** will let you continue, however, it will prompt you for some information about the category names you wish to assign to your new map’s categories.

If you are going to start creating more maps, it is wise to learn how to use the GRASS program called "g.remove" as well. You do not want to stop all work on your system because too many old maps stored there have filled the available memory. Map layers can be removed at the end of your GRASS session (where you are asked whether or not you wish to save your MAPSET).

4.6. (OVER MAP_NAME COLOR EXPR)

Two more operators can be used to create new map layers: OVER and COVER. Both operators permit the user to add new information onto existing map layers. OVER acts like a transparent map overlay process; this means that any overlay stacked atop other map overlays will not destroy the data able to be seen on the overlays underneath, but will only add information to them. COVER, on the other hand, works like someone painting over data on an existing map; the paint (which is the data contained on the added map overlay) is opaque, and where it occurs it obscures any information formerly visible on the maps underneath.

Consider the above example that uses AND on groups of soils and slope categories:

```
(AND
    (GROUP 8-11 (soils))
    (GROUP 3 5 7 (slope))
)
```

The resulting map is in one color, and does not reveal those locations where either *only* the soils categories or *only* the slope categories exist separately. To see both the soils of interest, the slopes of interest, and the locations where both occur together, try the following command:

```
(NAME mymap  (OVER RED (GROUP 8-11 (soils))))
(NAME myover (OVER mymap YELLOW (GROUP 3 5 7 (slope))))
```

These are two, separately run, OVER EXPRs. The first EXPR will create a "RED" overlay based on the GROUPing of "soils" categories shown. The NAME command then saves this result in the new file "mymap." The second OVER EXPR demonstrates how a second overlay can be written on top of an already saved overlay. Here the composite results are saved in "myover." At this point, two new maps have been created. Users should be careful to discard intermediate maps such as "mymap" when finishing a GRASS session. Let us look briefly at the mathematics at work here.

The results of the "soils" GROUPing are put into "mymap" as overlay "RED." This is done by assigning the value "1" to each location where soils categories 8 through 11 occur in the map layer. "Mymap" now contains only "0s" (no data) and "1s." The "slope" overlay is then overlaid **onto** "mymap" by adding the value "2" (for YELLOW) to each location where slope categories 3, 5, and 7 are found. The results are stored through use of the NAME command, in "myover." Note that at the end of this process, the resulting map can potentially contain "1s", "2s", and "3s" (3 representing the combination (intersections) of 1s and 2s).

OVER allows up to four overlays to be combined. The value added onto each location for these different overlays are as follows:

overlay	value
RED	1
YELLOW	2
BLUE	4
GREY	8

This schema preserves for each location which overlay(s) occur where. With four overlays available, there are 16 different combinations of these overlays possible. The following chart lists the 16 possible combinations, their resulting cell values, and assigned colors used for display.

overlay combination	value	color	
1	1	red	
2	2	yellow	primary
3	4	blue	colors
4	8	lt gray	
1 & 2	3	orange	
1 & 3	5	violet	
1 & 4	9	dark-red	
2 & 3	6	green	
2 & 4	10	dark-yellow	
3 & 4	12	dark-blue	
1 & 2 & 3	7	white	
1 & 2 & 4	11	dark-orange	
1 & 3 & 4	13	dark-violet	
2 & 3 & 4	14	dark-green	
1 & 2 & 3 & 4	15	dark-gray	

4.7. (COVER MAP_NAME EXPR)

COVER is similar to OVER, but does not provide a means to retain information already stored in a map layer. It is used to "paint" information over an existing map layer. For example, the following command might often be used:

(NAME mycover (GROUP 8-11(soils)))
(NAME mycover1 (COVER mycover (roads)))
(NAME mycover2 (COVER mycover1 (streams)))

These three COVER EXPRs create a map that has all locations from "streams" overwritten on all locations from "roads" overwritten on all locations having categories 8 through 11 from "soils." The effect here is to get the information from "soils" overlaid with some reference lines. At the end of these three requests, mycover will have only categories 1, 2, and 3. "1" will represent the soils that have not been overwritten with "roads" or "streams." "2" will represent those areas containing roads that have not been overwritten by "streams." Category "3", then, will show all locations containing "streams."

5. ALIASES

Each person will prefer to refer to these different commands with different words. In an attempt to satisfy the mnemonic thinking patterns of a wide variety of people, each of the above OPERATORS has several different aliases. The following table lists those available at the last writing:

OPERATOR:	Can be replaced with:				
QUIT	bye	BYE	q	stop	exit
AND	and	&	&&		
OR	or	|	||		
GROUP	group	grp			
NAME	name				
OVER	over	overlay			
COVER	cover				
NOT	not	~			

6. SAMPLE SESSION

Included here is a sample **r.combine** session using the database, *spearfish.* You should be able to completely replicate the interactive session with **r.combine** illustrated here. It is assumed that you can already get yourself into an interactive GRASS session. If you have difficulties with this please refer to the GRASS **g.help** program or its accompanying documentation.

Commands that you are requested to enter are displayed in this manner:

<Prompt> **my_command**

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

```
Some result displayed here
```

This sample session with **combine** can be reproduced on any normal text terminal. While **r.combine** processes data, it prints number maps out to the screen. On the color monitors, these number maps are turned into color maps. If the color monitor is used, your **r.combine** session will be displayed in color. If you do not have access to a color monitor, type **r.combine -s** to send **r.combine**'s graphic output to the terminal screen in the form of numerical maps. If you have a color monitor, type **combine** without the *-s* option. Your choice will affect only the graphics display; the same map files will be produced in either case.

To duplicate this session, you must request LOCATION *spearfish.* The mapset name you choose is arbitrary, but must be a one-word name less than 14 characters long. Hit the escape (ESC) key when you are satisfied with your LOCATION and MAPSET. You are now running the *spearfish* sample database provided with GRASS. Simply enter the following to begin your **r.combine** session.

GRASS 4.0> **r.combine**
or
GRASS 4.0> **r.combine -s**

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

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

Let us examine the list of available map layers for our sample database *spearfish.* The list of available

map layers can be seen by using the shell escape command:

[1]: **!g.list rast**

```
Running command: g.list rast
raster files available in mapset PERMANENT:
aspect          geology      rush.bnoise     soils.Tfactor    streams
bugsites        landuse      rushmore        soils.br.depth   streams.recl
density         owner        sensitive       soils.ph         strm.dist
elevation.dem   quads        slope           soils.range      transport.misc
elevation.dted  railroads    slope.7         soils.texture    trn.sites
erode.index     roads        soils           spot.image       vegcover
fields          rstrct.areas soils.Kfactor   streamdist
```

```
hit RETURN to continue-->
```

Hit the RETURN key to continue:

[2]: **RETURN**

Region information is also available within **r.combine** using the command:

[3]: **window map_name**

where "map_name" is the name of an available map layer in the database. The numbers you are most interested in are the eastings and northings. If, when running your own database you desire a different region, you should exit **r.combine** and run the GRASS program *g.region*. For this session use the defaults that describe the entire database.

Now, let us just look at the two existing data layers soils and vegcover. A correct and complete **r.combine** request is called a map expression (EXPR). A viable data layer name is, by itself, an EXPR:

[4]: **(soils)**

Now look at the vegetation cover data layer:

[5]: **(vegcover)**

Assume you are interested in the soil types represented by the category code: 9. The **r.combine** operator GROUP allows you to display only those categories of interest. GROUP, like all data operators, must be enclosed in parentheses to make a correct EXPR. It must contain the GROUP command followed by a series of numbers followed by an EXPR. Our command below has just one category number followed by a map layer name (that is itself a valid EXPR).

[6]: **(GROUP 9 (soils))**

Similarly, we will separate out categories 2 and 3 from the data layer "vegcover." Note that this

operation produces a binary result; this means that all grid cell locations containing vegetation cover categories 2 or 3 are now designated on the resulting map with the number 1, as shown below:

[7]: **(GROUP 2 3 (vegcover))**

Unless you make a direct request, analyses are displayed while being processed, and are not saved. The NAME command requests that results be saved. For example, the following command saves the results of the (GROUP 2 3 (vegcover)) analysis in "map1."

[8]: **(NAME map1 (GROUP 2 3 (vegcover)))**

This new map can now be used just like any other map:

[9]: **(map1)**

If you wanted to see all the locations that do not include areas containing vegetation cover type 2 or 3, you would use the operator "NOT." The next command does precisely this. Note that each of the GROUP commands is followed by one or more numbers and then followed by a valid EXPR. This ability to embed EXPRs makes the **r.combine** language extremely powerful.

[10]: **(NOT (GROUP 2 3 (vegcover)))**

Now we will introduce the AND command. Like the OR command, it requires two valid EXPRs. The following command says: display all the locations that contain vegetation cover type 2 or 3 AND also contain soil type 9:

[11]: **(AND (GROUP 2 3 (vegcover)) (GROUP 9 (soils)))**

Similarly, the next command says: display all the locations that contain vegetation type 2 or 3 OR contain soil type 9. This OR is an "inclusive or." In common language this might be expressed: and/or; that is, those areas that contain either vegetation type 1 or 2 AND/OR also contain soil type 9 are included.

[12]: **(OR (GROUP 2 3 (vegcover)) (GROUP 9 (soils)))**

Besides saving analyses with the NAME command, OVER and COVER are available for making composite maps. OVER provides the "cellophane" or "see through" overlay capability. It accommodates up to 4 overlays, but 2 to 3 overlays can sometimes overwhelm your eyes. Overlay number 1 is represented by the value 1, overlay 2 by 2, overlay 3 by 4, and overlay 4 by 8. In this manner, any combination of overlays can be represented by a unique value that is the sum of the individual overlay values. Thus, if a cell is assigned overlays 2 and 3, it is assigned the value 6 which codes only for the combination of overlays 2 and 3. (See the table of overlay combinations at the end of section **4.6.(OVER MAP_NAME COLOR EXPR)**.

[13]: **(NAME map2 (OVER RED (GROUP 2 3 (vegcover))))**

To demonstrate further, overlay the chosen vegetation types with soil type 9. In the following new map

(called "map3") then, the value 1 represents overlay 1 only, the value 2 represents overlay 2 only, and the value 3 represents the occurrence of both overlays 1 and 2. It uses the results stored in your previously made map called "map2," shown above.

[14]: **(NAME map3 (OVER YELLOW map2 (GROUP 9 (soils))))**

COVER performs an overlay as well, but unlike OVER, performs its overlay work opaquely. No overlay numbers are given. The first COVER overlay retains the values provided by its EXPR (here the GROUP expression).

[15]: **(NAME map4 (GROUP 2 3 (vegcover)))**

Subsequent COVER overlays are given the category value that is one greater than the current highest value. In this case the highest current category value was 3; the following command will therefore cover or paint over the existing map with the value "4" in the locations where the new EXPR is satisfied. The locations that contain soil category 9 painted over the existing map, map4, are shown below:

[16]: **(NAME map5 (COVER map4 (GROUP 9 (soils))))**

Continue experimenting with these commands. When finished, give the following command to leave **r.combine.**

[17]: **BYE**

When you then exit GRASS using the **exit** command, you will be given the opportunity to save or remove the new maps you have made:

GRASS 4.0> **exit**

GRASS SESSION WRAPUP

You have just finished working on mapset: <login-name>
The following RASTER, VECTOR, etc....data layers belong to it:
(any map names created appear here)

Shall the mapset <login-name> be saved y/n [y]

<prompt>: **y**

Do you wish to selectively remove data files y/n [n]

It is a good idea to remove any files which you no longer wish to keep at this time. Therefore, answer **y**, and then select option **1** to remove raster files.

```
Enter raster file to be removed
Enter 'list' for a list of existing raster files
Hit RETURN to cancel request
>
```

After entering the name of a map layer you wish to remove, you will be given the chance to remove additional map layers, or to exit GRASS:

<prompt>: **map_name**

```
REMOVE <map_name>
 cell
 cell header
 category
 color
 history

enter raster file to be removed
Enter 'list' for a list of existing raster files
Hit RETURN to cancel request
>
```

<prompt>: **RETURN**

```
GOOD BYE FROM GRASS
```

You will now be returned to your normal computer prompt outside of GRASS.

7. 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 (U.S. Army Construction Engineering Research Laboratory [USACERL], September 1990).

Westervelt, James, and William D. Goran, *Introduction to GRASS 4*, Draft ADP Report (USACERL, July 1991).

Westervelt, James, Mary Martin, and Deborah Brinegar, "GRASS 4.0 Programs," in *GRASS User's Reference Manual*, ADP Report N-87/22 rev (USACERL, September 1988, last revised December 1991).

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).