Using the R statistical data analysis language on GRASS 5.0 GIS data base files

Roger S. Bivand
Department of Geography,
Norwegian School of Economics and Business Administration,
Breiviksveien 40, N-5045 Bergen, Norway.

24th August 1999

1 Introduction

With the release of the open-source GIS GRASS 5.0 in early 1999, opportunities are presented for integration with the open-source R statistical data analysis programming environment (Ihaka and Gentleman, 1996, code obtained from [1]). In the examples presented, R is run interactively within the GRASS 5.0 environment, transfering data by writing and reading temporary text files; the operating system here is Linux. The note describes the implementation in R of functions needed to move data between GRASS and R, providing the user with a basic interface between the two environments.

Development of the leading Open Source GIS — GRASS — has been moved to Baylor University in Texas, where work on a new release incorporating floating-point raster cell values and NULL values different from zero is now in beta testing (Byars and Clamons, 1998, Linux binary obtained from [3]). In parallel with this, the R statistical and data analysis language, also Open Source, is maturing very rapidly, and can now execute most S and S-PLUS code in an unmodified form. In the past, when S was available on academic license, integration between GRASS and S existed in a loose-coupled form for integer raster cell values sampled at points given in a site layer.

The issues involved in linking two complex and fast-changing programming environments are encapsulated in a comprehensive way in the R functions included in the code accompanying this note. While the progress reported in this paper is based on Open Source Unix-like operating systems, it is worth noting that both GRASS and R have been compiled for MS Windows systems. Programming techniques for R are covered in Venables and Ripley (1997), and in materials available at the R archive [2].

In work to date, the interface used is that of the statistical analysis system, run from within the GIS environment. Given major design differences in memory management — GRASS uses the underlying file system, while R maps all active objects into a static area of memory allocated when the program is started, managed by a garbage collector — and other problems, it has been necessary to decide on a representation suiting the data analysis and visualization tasks being performed. This means here that the statistical programming environment is run from within GRASS, permitting GRASS command line instructions, including those requiring interaction, to be issued from within R using the system() function.

Running under Unix-family operating systems, GRASS only customizes the user's program execution environment, adding specific definitions needed for GRASS programs to be able to find the files and metadata required for their work. GRASS does not then represent a major memory overhead, and R can be launched with plenty of space for its computations. The examples reported below did not need more than 12Mb heap memory for analysis of a data set with 57600 raster cells, and with the judicious deletion of data objects from the heap, less would have surficed.

2 Interface functions

General issues involved in interfacing statistical and GIS software are discussed by Anselin et al. (1993) and Anselin and Bao (1997). Bao and Anselin (forthcoming) present work on an interface between S-PLUS and a commercial derivative of GRASS, using a more deeply embedded technique than that covered in this note. The motivation for writing an interface in the interpreted language of one of the linked software systems as presented below is to make a simple interface available to any analyst willing and able to download two open source programs and install them, at present under a Unix-like operating system (see also Bivand, 1996).

Two interface functions are included in the accompanying code, get.GRASS() to read one or more GRASS data base files (map layers) into R, with an option to retain the category labels of the values, and put.GRASS() to export single R numerical vectors to GRASS. In both cases, the enhancements to GRASS 5.0 have been used, taking account of floating point numerical values and NULL (not available) values different from zero. In previous releases of GRASS, values of zero might be numerical zero or NULL, and only integer values were used, inhibiting or complicating analysis. The functions are written using the R language, supplemented with a small utility function, list.GRASS(), to list existing GRASS raster data base files.

At present, R can only read from named files or sockets, while GRASS can only write to named files or pipes. While it would have been possible to use an additional interpreted language, for instance Perl, as glue, it was decided to write ASCII data to, and read ASCII data from, temporary files, despite the consequences for performance of this choice. Numerical GRASS map layers are read directly into an R data.frame object, prepending GRASS grid coordinates as variables “east” and “north” before variables named from the requested GRASS data base file names. Since the GRASS r.mapcalc command and R impose similar conventions on map layer and variable names, arithmetically ambiguous names like “a-b” should not occur on either side, and no sanity check is made; layer and variable names are assumed to belong to the subset of names valid in both environments. When category labels are imported and R factors created, substantially more internal processing is involved, both in terms of copies of data in memory and time.

The get.GRASS() function creates an object of class “grass”, using the R method despatch object naming convention. Objects of this class are lists with two elements: “data” and “meta”. The “data” list element is the data.frame object described above. If category labels have been imported, then the data.frame will contain factor columns for each map layer that possessed category labels. The “meta” list element contains a wide range of metadata components, some of which are reported by summary.grass(), using the method despatch mechanism. Method despatch is employed to permit generic functions, such as summary() or plot(), to call class-specific functions by concatenating their own names with the class of the object given as their first argument.

The metadata are also used by plot.grass(), which provides a simple interface between “grass” data objects and the image() function, to give the dimensions of the plotting window, and to reverse the row order of the image matrix. GRASS, as many GIS and image display systems, looks at the world from top left down, while R assumes that data begin from bottom left. Factor variables, originally GRASS map layers with category labels, may be plotted by taking their integer code() values, and passing through an appropriate range for these values in the image() function zlim argument.

3 Examples

Two brief examples will be presented to illustrate the uses of the interface. The first is taken from the North-West Leicestershire GRASS data set made available with the “GRASS-seeds” tutorial by the Assist project [1]. This covers a 12km by 12km section of central England with 50m by 50m raster cell sizes. Two map layers have been imported into R, “topo”, the elevation in meters interpolated from digitised contours, and “landcov”, a category labelled land cover classification from Landsat TM data. Exercise B.1 in the “GRASS-seeds” tutorial involves the use of the r.average and r.report commands to calculate average height by land cover category. Here we not only use the R summary() function by category: tapply(topo, landcov, summary) to add a Tukey five number summary to the mean, as shown in Table 1, but also display in Figure 1 boxplots of elevation by land cover category: boxplot(topo ~ landcov). Such exploratory and graphical statistical functions give a much clearer impression of the underlying distributions than their means.

The second example is from a geomorphometric classification of the topography of Kosovo, based on GTOPO30 elevation data (30 second grid data from [4], and projected to UTM zone 34), described in more detail by Bivand

Land cover	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
Arable	40	63	81	104.00	153	270
Industry	40	42	59	83.61	120	220
Pasture	40	75	145	133.50	180	278
Quarry	40	59	161	135.30	177	198
Residential	40	53	60	76.30	80	218
Scrub	40	100	161	148.20	196	270
Water	47	65	99	99.27	131	221
Woodland	40	118	156	142.90	174	248

Table 1: Summary statistics of elevation in meters by land cover categories for North-West Leicestershire.

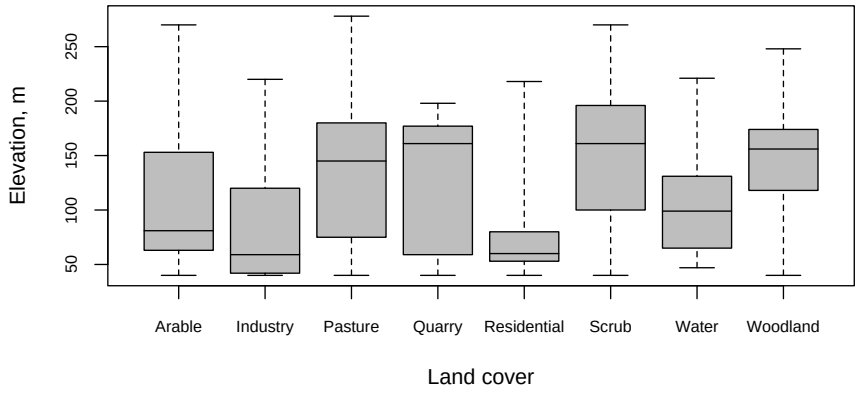

Figure 1: Boxplots of elevation by land cover categories for North-West Leicestershire.

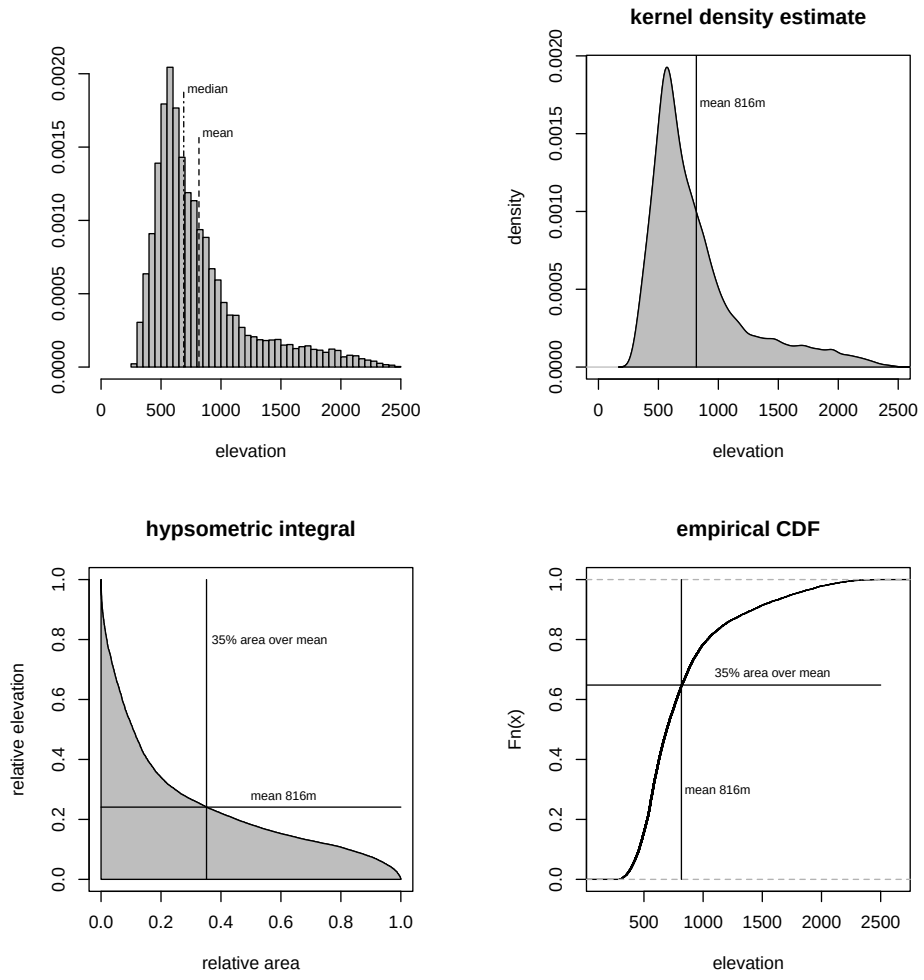

Figure 2: Graphical data analysis of elevation in Kosovo, using a variety of R functions; for comparison with Luo (1998), skewness: 1.583, kurtosis: 2.219.

(1999). This data set is interesting because it is imported into R from floating point GRASS data base files, which also use NULL values to represent cell locations beyond the borders of the province; the data set has 163 rows and 164 columns, with a cell size of 1000m in each direction. Before undertaking the classification, the elevation data were explored as shown in Figure 2, using a histogram function (truehist(), Venables and Ripley, 1997) permitting control of bin widths and the starting point of the display, and a density trace with default bandwidth (Jacoby, 1997). A hypsometric analysis was performed, and is also shown in Figure 2; unlike Luo (1998), use was made of Pike and Wilson's (1971) result that the hypsometric integral and the elevation-relief ratio are the same. Finally, essentially the same diagram was made by plotting the empirical cumulative density function of the elevation data (Jacoby, 1997).

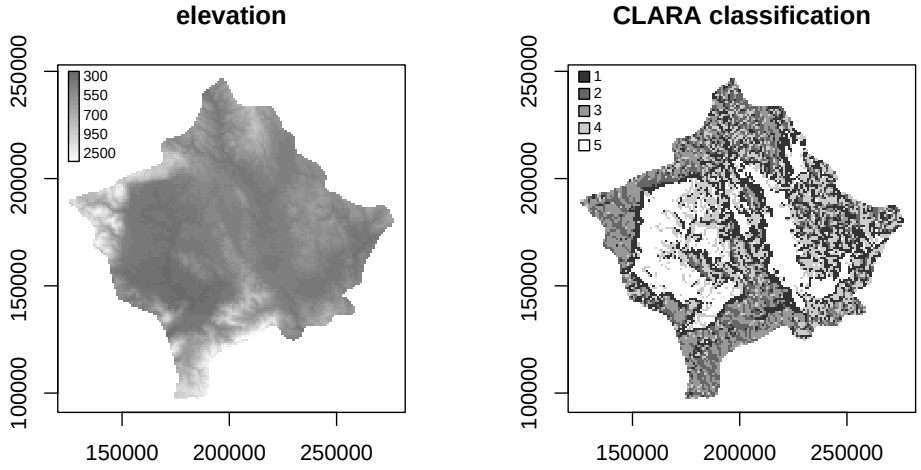

Figure 3: Left: elevation in Kosovo; right: geomorphometric classification into five surface topography classes; scale in grid meters (both figures were made using plot.grass() supplemented by extra commands to construct and place the legends).

Class	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1	290.3	572.1	655.6	690.9	771.5	1811
2	318.1	717.0	908.7	1036.0	1298.0	2244
3	513.1	939.2	1223.0	1327.0	1674.0	2474
4	332.3	589.0	699.6	747.0	856.3	2405
5	293.8	428.9	505.5	495.8	558.1	1023

Table 2: Summary statistics of elevation in meters by geomorphometric classes for Kosovo.

As Jones (1998) has shown, the Zevenbergen and Thorne (1987) method for computing slopes from gridded elevation data performs better than many alternatives (see also Burrough and McDonnell, 1998). Using this method, slope gradient and plan and profile curvature were added to elevation, local relief in a 3×3 window, and the local elevation-relief ratio, again in a 3×3 window, as a basis for classification, logarithms being taken of three of the variables. Figure 3 shows the topography of Kosovo, and the results of a classification of 10948 cells into 5 classes using the clara() function from the cluster package, while Table 2 displays the distributions of elevation by class.

4 Conclusion

The code accompanying this note is a first step in making GRASS data base files available for analysis in R, and could be extended by reading data into R directly in binary form, and by including file types other than raster. In particular, the use of R functions for interpolation from irregularly spaced points to regular grids, or for point pattern analysis or other spatial statistical techniques in connection with GRASS would be welcome. The functions presented do not use language constructs specific to R, and should therefore work with S; since they use simple file transfer, they could also easily be adapted to work with other raster GIS with command line interfaces

at the Unix shell level. Finally, it would be interesting to see whether the same approach can be used to link R and GRASS under MS Windows operating systems; R runs well under Windows, while GRASS has been made available for the same platform, although both are developed under Unix-like operating systems.

References

Anselin, L. and Bao S. 1997. Exploratory spatial data analysis linking SpaceStat and ArcView. In: Fischer, M. M., Getis, A. (Eds) Recent developments in spatial analysis. Springer-Verlag, Berlin, pp. 35–59.

Anselin, L., Dodson, R. and Hudak, S. 1993. Linking GIS and spatial data analysis in practice. Geographical Systems 1 (1), 3–23.

Bao, S. and Anselin, L. forthcoming. Linking spatial statistics and GIS: operational issues in the SpaceStat–ArcView link and the S+Grassland link. In: Proceedings ASA Section on Statistical Graphics, American Statistical Association, Alexandria.

Bivand, R. S. 1996 Scripting and toolbox approaches to spatial analysis in a GIS context. In Fischer, M. M., Scholten, H., Unwin D. (Eds) Spatial analytical perspectives on GIS in the environmental and socio-economic sciences. Taylor & Francis, London, pp. 39–52.

Bivand, R. S., 1999. Integrating GRASS 5.0 and R: GIS and modern statistics for data analysis. In: Proceedings 7th Scandinavian Research Conference on Geographical Information Science, Aalborg, Denmark, pp. 111–127.

Burrough, P. A., McDonnell, R. A. 1998. Principles of geographical information systems. Oxford University Press, Oxford, 333pp.

Byars, B., Clamons, S. 1998. GRASS is back! GIS World 10(2), 60–63.

Ihaka, R., Gentleman, R. 1996. R: a language for data analysis and graphics. Journal of Computational and Graphical Statistics, 5 (*), 299–314.

Jacoby, W. G. 1997. Statistical graphics for univariate and bivariate data. Sage Publications, Thousand Oaks, 97pp.

Jones, K. H. 1998. A comparison of algorithms used to comute hill slope as a property of the DEM. Computers & Geosciences 24 (4), 315–323.

Luo, W., 1998. Hypsometric analysis with a geographic information system. Computers & Geosciences 24 (8), 815–821.

Pike, R. J., Wilson, S. E. 1971. Elevation-relief ratio, hypsometric integral, and geomorphic area-altitude analysis. Geological Society of America Bulletin 82 (*), 1079–1084.

Venables, W. N., Ripley, B. D. 1997. Modern applied statistics with S-PLUS. Springer, New York, 548pp.

Zevenbergen, L. W., Thorne, C. R. 1987. Quantitative analysis of land surface topography. Earth Surface Processes and Landforms, 12 (*), 47–56.

Internet references

[1] Assist project data set. `http://www.geog.le.ac.uk/assist/grass/data/leics/leics.tar.Z`

[2] Comprehensive R Archive Network. `http://www.ci.tuwien.ac.at/R/`

[3] GRASS home page. `http://www.baylor.edu/~grass`

[4] Source of Kosovo elevation data. `http://edcwww.cr.usgs.gov/landdaac/gtopo30/gtopo30.html`

Appendix:
R function code

```
#
# R.GRASS.R
# utility functions for moving raster data between GRASS 5.0 and R
# Copyright (C) 1999 Roger Bivand
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details (Free Software
# Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.)

#
# get.GRASS moves one or more GRASS 5.0 raster files to an R data object
# of class "grass", returning the filled object. Setting catlabels to
# TRUE imports category labels instead of codes, and requires more memory.
#
get.GRASS <- function(rlist = "", catlabels=FALSE) {
if (! is.character(rlist))
stop("character vector of GRASS data base file names required")
G.list <- list.GRASS(type="rast")
res <- rlist %in% G.list
if (! all(res)) {
warning("The following GRASS data base files were not found:")
print(rlist[res == FALSE])
stop("transfer terminated")
}
FILE <- tempfile("GRtoR")
on.exit(unlink(FILE))
rstats <- "r.stats -1g input="
if (catlabels) rstats <- "r.stats -1gl fs=\":\" input="
for (i in 1:(length(rlist)))
rstats <- paste(rstats, rlist[i], ",", sep="")
rstats <- paste(rstats, " output=", FILE, sep="")
system(rstats)
G <- vector(mode="list")
if (catlabels) {
data0 <- read.table(FILE, sep=":", na.strings="*")
varlist <- c("east", "north", rlist)
d0ncol <- ncol(data0)
passthru <- 1:d0ncol
for (i in 1:(d0ncol)) if (all(is.na(data0[,i])))
passthru[i] <- -1
for (i in 2:(d0ncol)) if (is.factor(data0[,i]))
passthru[i-1] <- -1
G$data <- data0[,passthru > 0]
if (ncol(G$data) != length(varlist)) {
warning("mismatch in column names")
}
else {names(G$data) <- varlist}
}
else {
G$data <- read.table(FILE, na.strings="*")
names(G$data) <- c("east", "north", rlist)
}
```

```
G$meta$LOCATION <- system("g.gisenv LOCATION_NAME", intern=TRUE)
G$meta$MAPSET <- system("g.gisenv MAPSET", intern=TRUE)
G$meta$rstats <- rstats
G$meta$who <- system("whoami", intern=TRUE)
G$meta$when <- system("date", intern=TRUE)
G$meta$obsno <- 1:nrow(G$data)
G$meta$reverse <- order(G$data$north, G$data$east)
G$meta$Ncol <- length(unique(G$data$east))
G$meta$Nrow <- length(unique(G$data$north))
G$meta$we.range <- range(G$data$east)
G$meta$sn.range <- range(G$data$north)
G$meta$xr <- (G$meta$we.range[2] - G$meta$we.range[1])
G$meta$yr <- (G$meta$sn.range[2] - G$meta$sn.range[1])
G$meta$maxr <- max(G$meta$xr, G$meta$yr)
G$meta$xstep <- (G$meta$xr)/(G$meta$Ncol - 1)
G$meta$ystep <- (G$meta$yr)/(G$meta$Nrow - 1)
G$meta$xlim <- c(G$meta$we.range[1] - (G$meta$xstep/2),
(G$meta$we.range[1] + G$meta$maxr) + (G$meta$xstep/2))
G$meta$ylim <- c(G$meta$sn.range[1] - (G$meta$ystep/2),
(G$meta$sn.range[1] + G$meta$maxr) + (G$meta$ystep/2))
G$meta$xseq <- seq(from=G$meta$we.range[1], to=G$meta$we.range[2],
by=G$meta$xstep)
G$meta$yseq <- seq(from=G$meta$sn.range[1], to=G$meta$sn.range[2],
by=G$meta$ystep)
class(G) <- "grass"
invisible(G)
}

#
# put.GRASS moves a single numeric vector to GRASS, using the metadata
# prepared when layers were got from the GRASS data base.
#
put.GRASS <- function(G, lname="", layer, title="") {
if (class(G) != "grass") stop("Data not a grass object")
if (length(lname) != 1)
stop("Single new GRASS data base file name required")
G.list <- list.GRASS(type="rast")
res <- lname %in% G.list
if (any(res))
stop(paste(lname, ": GRASS raster file already exists", sep=""))
if (length(layer) != G$meta$Ncol * G$meta$Nrow)
stop("GRASS object metadata do not match layer length")
FILE <- tempfile("RtoGR")
on.exit(unlink(FILE))
outstr <- paste("north:   ", G$meta$sn.range[2] + (G$meta$ystep/2),
"\nsouth:   ", G$meta$sn.range[1] - (G$meta$ystep/2),
"\neast:    ", G$meta$we.range[2] + (G$meta$xstep/2),
"\nwest:    ", G$meta$we.range[1] - (G$meta$xstep/2),
"\nrows:    ", G$meta$Nrow, "\ncols:    ", G$meta$Ncol, "\n",
sep="")
cat(outstr)
cat(outstr, file=FILE)
write(t(matrix(layer, nrow=G$meta$Nrow, ncol=G$meta$Ncol, byrow=T)),
file=FILE, append=T, ncolumns=G$meta$Ncol)
system(paste("r.in.ascii input=", FILE, " nv=NA output=", lname,
" title=\"", title, "\"", sep=""))
}

#
# summary.grass displays the metadata prepared when map layers are moved to R
# using get.GRASS.
#
summary.grass <- function(G) {
```

```
if (class(G) != "grass") stop("Data not a grass object")
cat("Data from GRASS 5.0 LOCATION ", G$meta$LOCATION,
", imported using r.stats:\n", G$meta$rstats, "\nby ", G$meta$who,
", on ", G$meta$when, ".\nThe data table has the following layers:\n",
sep="")
cat(names(G$data), "\n", sep=" ")
cat("in ", G$meta$Ncol,
" columns and ", G$meta$Nrow, " rows. The west-east range is:\n",
G$meta$we.range[1], ", ", G$meta$we.range[2], "\nand the south-north:\n",
G$meta$sn.range[1], ", ", G$meta$sn.range[2],
"\nWest-east cell sizes are in ", G$meta$xstep,
" measurement units\nSouth-north in ", G$meta$ystep,
" measurement units.", sep="")
cat("\nFor plotting, xlim: ", G$meta$xlim, ", ylim: ",
G$meta$ylim, "\n\n", sep=" ")
}

#
# plot.grass provides a simple interface between grass data
# objects and the image() function; category layers may be plotted
# by taking codes() of the layer, and setting zlim to non-default values.
#
plot.grass <- function(G, layer, ...) {
if (class(G) != "grass") stop("Data not a grass object")
if (length(layer) != G$meta$Ncol * G$meta$Nrow)
stop("GRASS object metadata do not match layer length")
oldpty <- par("pty")
on.exit(par(pty=oldpty))
par(pty="s")
image(x=G$meta$xseq, y=G$meta$yseq,
z=t(matrix(layer[G$meta$reverse], nrow=G$meta$Nrow,
ncol=G$meta$Ncol, byrow=T)),
xlim=G$meta$xlim, ylim=G$meta$ylim, xlab="", ylab="",
...)
}

#
# list.GRASS lists available GRASS data base files of the user-
# specified data type (at present only "rast"), returning a
# character vector.
#
list.GRASS <- function(type = "rast") {
res <- system(paste("g.list -f ", type, sep=""), intern=TRUE)
l.2 <- length(res)-2
res1 <- strsplit(res[3:l.2], " ")
G.list <- character(length(res1))
for(i in 1:length(res1)) G.list[i] <- res1[[i]][1]
invisible(G.list)
}
```