

Package ‘GRASS’

January 20, 2010

Version 0.3-9

Date 2010-01-19

Title Interface between GRASS 5.0 geographical information system and R

Author Roger Bivand

Maintainer Roger Bivand <Roger.Bivand@nhh.no>

Description Interface between GRASS 5.0 geographical information system and R, based on starting R from within the GRASS environment using values of environment variables set in the GISRC file. Interface examples should be run outside GRASS, others may be run within. Wrapper and helper functions are provided for a range of R functions to match the interface metadata structures.

Depends R (>= 2.2.0)

Suggests spatial

License GPL (>= 2)

URL <http://grass.itc.it/statsgrass/index.html>

Repository CRAN

Date/Publication 2010-01-20 07:59:55

R topics documented:

contourG	2
get.mapsets	3
gmeta	4
GRASS.connect	6
kde2d.G	7
list.grass	9
pcbs	10
plot.grassmeta	11
rast.get	13

rast.put	15
sites.get	16
sites.put	18
summary.grassmeta	20
trmat.G	21
utm.maas	22

Index	25
--------------	-----------

contourG	<i>Equal scale contour plots for GRASS raster and site data</i>
----------	---

Description

contour.G provides a simple interface between grass data objects and the `contour()` function.

Usage

```
contourG(x, layer=NULL, xlab="", ylab="", reverse=NULL, add=FALSE, ...)
```

Arguments

x	GRASS metadata from <code>gmeta()</code>
layer	GRASS-ordered raster layer passed on to <code>contour()</code>
xlab	a title for the x axis, passed to <code>plot()</code>
ylab	a title for the y axis, passed to <code>plot()</code>
reverse	see <code>help(reverse)</code>
add	FALSE for new plot, TRUE to overlay layer on prior plot
...	parameters passed through to <code>contour()</code>

Details

The function uses `asp=1` from the `plot.window()` function to ensure that the scales chosen to be equal on both axes, that is 1cm represents the same units on each.

Value

none

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[plot.window](#), [contour](#), [plot.grassmeta](#)

Examples

```
data(utm.maas)
Zn.o <- as.ordered(cut(utm.maas$Zn, labels=c("insignificant", "low",
"medium", "high", "crisis"), breaks=c(100, 200, 400, 700, 1000, 2000),
include.lowest=TRUE))
G <- maas.metadata
contourG(G)
points(utm.maas$east, utm.maas$north, pch=unclass(Zn.o))
legend(x=c(269800, 270500), y=c(5652300, 5653000), pch=c(1:5),
legend=levels(Zn.o))
title("Note equal east and north scales")
contourG(G, kde2d.G(utm.maas$east, utm.maas$north, h=c(300,300), G=G, Z=utm.maas$Zn)*maasmas
points(utm.maas$east, utm.maas$north, pch=unclass(Zn.o))
legend(x=c(269800, 270500), y=c(5652300, 5653000), pch=c(1:5),
legend=levels(Zn.o))
title("Kernel density representation of soil Zn")
```

get.mapsets

Access GRASS readable mapsets search path

Description

Access GRASS readable mapsets search path by reading the internal structure read from the SEARCH_PATH file in the current mapset root.

Usage

```
get.mapsets()
refresh.mapsets()
```

Details

The functions are needed because the R interface is run as a GRASS program within which the `SEARCH_PATH` file is only read once. If, during an R/GRASS session, a command such as `system("g.mapsets mapset=new.mapset")` is given, the `SEARCH_PATH` file will be updated, but not the values earlier read into the interface. Consequently, after any `system("g.mapsets mapset=...")`, `refresh.mapsets()` should be run to update the internal values.

Value

`get.mapsets()` returns the current setting as a character vector of mapsets, while `refresh.mapsets()` returns the updated mapset names, also as a character vector.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. *Computers and Geosciences*, 26, pp. 1043-1052.

gmeta

Reads GRASS metadata from the current LOCATION

Description

GRASS LOCATION metadata are read into a list in R, and processed to provide other functions with parameters and structures they require.

Usage

```
gmeta(interp=FALSE)
east(object)
north(object)
## Default S3 method:
east(object)
## Default S3 method:
north(object)
## S3 method for class 'grassmeta':
east(object)
## S3 method for class 'grassmeta':
north(object)
obsno(G)
reverse(G)
make.maas.location()
```

Arguments

<code>interp</code>	if <code>interp</code> is TRUE, the interface uses text transfer through <code>system()</code>
<code>G</code>	a <code>grassmeta</code> object returned by <code>gmeta</code>
<code>object</code>	a <code>grassmeta</code> object returned by <code>gmeta</code>

Details

The function retrieves metadata from the GRASS LOCATION from which R was started, and stores it in a list. The metadata is taken from the current, active region, and the implied resolution for raster layers. Next, this is used to create a number of auxiliary objects used by other interface routines. Other auxiliary objects may be constructed using the access functions `east.grassmeta`, `north.grassmeta`, `obsno`, and `reverse`. The function returns a list with class `grassmeta`:

Value

<code>LOCATION</code>	GRASS LOCATION name
<code>MAPSET</code>	GRASS MAPSET within the LOCATION
<code>proj</code>	GRASS projection description
<code>n</code>	Northern edge: numeric
<code>s</code>	Southern edge: numeric
<code>w</code>	Western edge: numeric
<code>e</code>	Eastern edge: numeric
<code>nsres</code>	North-South resolution in measurement units, typically metres or decimal degrees
<code>ewres</code>	East-West resolution in measurement units, typically metres or decimal degrees
<code>Nrow</code>	Number of rows of raster cells
<code>Ncol</code>	Number of columns of raster cells
<code>Ncells</code>	Number of raster cells
<code>xlim</code>	East-West range
<code>ylim</code>	North-South range
<code>xseq</code>	Sequence of raster cell East-West midpoint coordinates
<code>yseq</code>	Sequence of raster cell North-South midpoint coordinates
<code>ryseq</code>	Reversed sequence of raster cell North-South midpoint coordinates

In addition, a number of internal GRASS environment variables are exposed for reading and manipulation - these are at present visible but should be used only with caution, and signal the future deprecation of the `gmeta()` mechanism for handling metadata.

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[summary.grassmeta](#)

Examples

```
if(!get("maas.loc", env = .GRASS.meta)) make.maas.location()
G <- gmeta()
summary(G)
```

GRASS.connect

Access to underlying environment variables within GRASS

Description

Access functions to variables set within GRASS, should normally not be used directly.

Usage

```
GRASS.connect(manual=FALSE, gisdbase=NULL, loc=NULL, mapset=NULL)
get.GISDBASE()
set.GISDBASE(gisdbase)
get.LOCATION()
set.LOCATION(loc)
get.MAPSET()
set.MAPSET(mapset)
get.cygwinstring()
set.cygwinstring(cygwin)
get.GRASSChkGISRC()
```

Arguments

manual	TRUE to set location parameters manually, FALSE by default
gisdbase	string: full path name to GISDBASE
loc	string: location name
mapset	string: mapset name
cygwin	string: CygWin prefix for prepending to GISRC and GISDBASE values

Details

For users running GRASS under Cygwin with a standard single cygwin root, follow the instructions given as the library is loaded:

```
> library(GRASS) If GRASS.connect() fails in this way and you are running under CygWin, please set the CygWin root file system prefix using: set.cygwinstring() and re-run GRASS.connect()
```

In such cases, the cygwin root is for example "c:/cygwin", so the cygwin GISRC variable "/home/rsb/.grassrc5" can be seen by R as "c:/cygwin/home/rsb/.grassrc5", and the GISDBASE field in that file, for example "/usr/local/grass5/data" as "c:/cygwin/usr/local/grass5/data".

Where the Cygwin setup is more complex, you will need to set the environment values manually:

Optionally:

```
> gisrcname <- Sys.getenv("GISRC") > fix(gisrcname) \# to edit it manually to something readable  
> scan(gisrcname, character(0)) \# to have an on-screen version of values
```

Then:

```
> GRASS.connect(manual=TRUE, gisdbase="the R readable full path name", + loc="LOCATION",  
mapset="MAPSET")
```

GRASS.connect() will attempt to insert these values and trap erroneous values.

Value

Most of the functions have side effects (apart from those getting values), but otherwise do not return much of use.

In addition, a number of internal GRASS environment variables are exposed for reading and manipulation - these are at present visible but should be used only with caution, and signal the future deprecation of the gmeta() mechanism for handling metadata.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

Description

A version of `kde2d()` in the MASS package of the VR collection using the metadata of the underlying GRASS LOCATION. `kde2d()` is for "two-dimensional kernel density estimation with an axis-aligned bivariate normal kernel, evaluated on a square grid." An optional feature is the introduction of a weighted moving average through the Z argument.

Usage

```
kde2d.G(G, x, y, h, reverse=NULL, Z=NULL)
```

Arguments

G	GRASS metadata from <code>gmeta()</code>
x	x coordinate of data
y	y coordinate of data
h	vector of bandwidths for x and y directions. Defaults to normal reference bandwidth.
reverse	see <code>help(reverse)</code>
Z	vector of attribute values

Value

A vector of `G$Ncells` values is returned, with the results of density/weighted moving average calculations ordered as a GRASS raster file. Density values are points per unit area, while moving average values are in the units of the Z variable.

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Original `kde2d()`, `bandwidth.nrd()`: Brian D. Ripley and Bill Venables as detailed in the MASS package documentation; adapted by Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. *Computers and Geosciences*, 26, pp. 1043-1052.

See Also

[kde2d](#)

Examples

```
data(utm.maas)
G <- maas.metadata
inregion <- (utm.maas$east >= G$w & utm.maas$east <= G$e) &
  (utm.maas$north >= G$s & utm.maas$north <= G$n)
```

```

if(all(!inregion)) stop("None of the site locations are inside the current GRASS region")
if(any(!inregion)) warning("Some site locations are outside the current GRASS region")
plot(G, kde2d.G(G=G, utm.maas$east, utm.maas$north, h=c(300,300))*maasmask)
points(utm.maas$east, utm.maas$north)
rug(utm.maas$east, side=1, ticksize=0.02)
rug(utm.maas$north, side=2, ticksize=0.02)
title(main="Kernel estimates of density of soil sample placing",
      xlab="(Bailey & Gatrell, pp. 84-88)")
plot(G, kde2d.G(G=G, utm.maas$east, utm.maas$north, h=c(300,300), Z=utm.maas$Zn)*maasmask)
points(utm.maas$east, utm.maas$north)
rug(utm.maas$east, side=1, ticksize=0.02)
rug(utm.maas$north, side=2, ticksize=0.02)
title(main="Kernel density weighted average, Zn ppm",
      xlab="(Bailey & Gatrell, pp. 159-161)")

```

list.grass

List GRASS database elements

Description

A helper function to list GRASS database elements, using compiled access to GRASS - for an interpreted alternative use system("g.list").

Usage

```

list.grass(type = "cell")
## S3 method for class 'glist':
print(x, ...)

```

Arguments

type	one of "cell", "site_lists", or "dig_plus"
x	a glist object returned by list.grass()
...	arguments passed to print methods

Value

A glist object, a named list of database elements in character vectors, one character vector for each MAPSET in the mapsets returned by get.mapsets()

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>

pcbs

PCBs in an area of South Wales

Description

The `pcbs` data frame has 70 rows and 3 columns. It records contamination of the environment with polychlorinated biphenyls around the site of a large plant for the incineration of chemical wastes near Pontypool, South Wales.

Usage

```
data(pcbs)
```

Format

This data frame contains the following columns:

east a numeric vector - UTM zone 30 eastings coordinates

north a numeric vector - UTM zone 30 northings coordinates

pcbs a numeric vector - standardised scores for seven types of PCB

Details

The original data were positioned in relation to National Grid coordinates, and have been reprojected to UTM zone 30, using the WGS84 ellipsoid.

Note

This dataset will only work if R is started from inside GRASS, choosing the location for the Pontypool PCBs data. GRASS installations differ in permissions needed to establish a new location, and system administrator privileges may be needed to create the Pontypool location. Information about the necessary metadata are as follows. The actual data may be accessed from R: `data(pcbs)`.

projection UTM

ellipsoid WGS84

zone 30

north 77200

south 74500

west 68000

east 70200

nsres 25

ewres 25

rows 108

cols 88

Source

Bailey, T. C., Gatrell, A. C. 1995 Interactive Spatial Data Analysis (Longman, Harlow); pages 149-150, courtesy of Andrew Lovett, School of Environmental Sciences, University of East Anglia.

Examples

```
data(pcbs)
pcbs.o <- as.ordered(cut(pcbs$pcbs, labels=c("insignificant", "low",
"medium", "high", "crisis"), breaks=c(1,20,100,500,1000,5000),
include.lowest=TRUE))
table(pcbs.o)
plot(pcbs$east, pcbs$north, pch=unclass(pcbs.o), xlab="", ylab="", asp=1)
legend(x=c(67980, 68480), y=c(74710, 75180), pch=c(1:5), legend=levels(pcbs.o))
```

plot.grassmeta *Equal scale plots for GRASS raster and site data*

Description

plot.grassmeta provides a simple interface between grass data objects and the image() function; category layers may be plotted by taking unclass() of the layer, and setting zlim to non-default values. If layer is not set, a blank base map is plotted, for instance for use with points().

Usage

```
## S3 method for class 'grassmeta':
plot(x, layer=NULL, xlab="", ylab="", reverse=NULL, add=FALSE,
     breaks=NULL, ...)
legtext(break.levels)
leglabs(x1, under="under", over="over", between="-")
```

Arguments

x	GRASS metadata from gmeta()
layer	GRASS-ordered raster layer passed on to image()
xlab	a title for the x axis, passed to plot()
ylab	a title for the y axis, passed to plot()
reverse	see help(reverse)
add	FALSE for new plot, TRUE to overlay layer on prior plot
...	parameters passed through to image()
break.levels	vector of image break levels
breaks	vector of image break levels
x1	in leglabs(), a vector of breaks
under	character value to denote under
over	character value to denote over
between	character value to denote between

Details

The function uses `asp=1` from the `plot.window()` function to ensure that the scales chosen to be equal on both axes, that is 1cm represents the same units on each. The `legtext` function is a small helper to aid in converting image break levels into legend texts, taking a numeric argument as `image(break=)`, and returning a character vector.

Value

none

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[plot.window](#), [image](#)

Examples

```
data(utm.maas)
Zn.o <- as.ordered(cut(utm.maas$Zn, labels=c("insignificant", "low",
"medium", "high", "crisis"), breaks=c(100, 200, 400, 700, 1000, 2000),
include.lowest=TRUE))
G <- maas.metadata
plot(G)
points(utm.maas$east, utm.maas$north, pch=unclass(Zn.o))
legend(x=c(269800, 270500), y=c(5652300, 5653000), pch=c(1:5),
legend=levels(Zn.o))
title("Note equal east and north scales")
example(kde2d.G)
```

 rast.get

Import GRASS raster files

Description

`rast.get()` moves one or more GRASS 5.0 raster files to a list, returning the filled object. Setting `catlabels` elements, corresponding to the files named in `rlist`, to `TRUE`, imports category labels instead of codes yielding an ordered factor rather than a numeric vector, and requires more memory.

Usage

```
rast.get(G, rlist="", catlabels=NULL, drop.unused.levels=FALSE,
make.ordered=TRUE, debug=FALSE, interp=FALSE)
```

Arguments

<code>G</code>	GRASS metadata from <code>gmeta()</code>
<code>rlist</code>	character vector of GRASS raster file names; may include mapset qualifiers
<code>catlabels</code>	logical vector of length <code>length(rlist)</code> : if <code>TRUE</code> , GRASS category raster layers are imported as ordered factors, if <code>FALSE</code> , as numeric only
<code>drop.unused.levels</code>	default <code>FALSE</code> , if <code>TRUE</code> , drop unused levels from factor. Beware: some levels may be absent from a region of a location but should be retained, use <code>TRUE</code> with caution. The argument will be applied to all layers read during a single call to <code>read.get()</code> .
<code>make.ordered</code>	The default behaviour has been to treat all GRASS categorical rasters as ordered factors. This can be modified by setting to <code>FALSE</code> , when the "ordered" class tag will be removed. The argument will be applied to all layers read during a single call to <code>read.get()</code> .
<code>debug</code>	if <code>TRUE</code> , the temporary ASCII file used for transfer is not deleted on exit from the function
<code>interp</code>	if <code>TRUE</code> , the interpreted version of the function is used instead of the loaded compiled version

Details

`rast.get()` assumes firstly that the region of the GRASS LOCATION has not been changed since `gmeta()` was last run, secondly that resolutions of the requested map layers and the GRASS LOCATION are in accord, and thirdly that the data cover the selected region. If the second or third assumptions are not met for the interpreted version, the GRASS program `r.stats` run through `system()` will read from the underlying map layers according to the new region setting, yielding cell values at the current resolution, and inserting NAs in areas for which there are no data. For the compiled version, the user is required to work within a given map region at the same resolution.

Value

A list is returned with `length(rlist)` vectors and `G$Ncells` values in each vector; the columns are named using the names of the imported GRASS raster files. If the names of the GRASS raster files are qualified by mapset, the name of the mapset will be appended after an underscore. Those imported with category labels are ordered factors, and their names are suffixed with `.f`

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[rast.put](#)

Examples

```
if(!get("maas.loc", env = .GRASS.meta)) make.maas.location()
data(utm.maas)
Zn.o <- as.ordered(cut(utm.maas$Zn, labels=c("insignificant", "low",
"medium", "high", "crisis"), breaks=c(100, 200, 400, 700, 1000, 2000),
include.lowest=TRUE))
G <- gmeta()
if(length(ls(pat="nameR"))==0){example(rast.put)}
exget <- rast.get(G, rlist=c("ex.tr3.in",
"ex.Zn.s3.o.in"), catlabels=as.logical(c(FALSE, TRUE)))
plot(G, exget$ex.tr3.in)
points(utm.maas$east, utm.maas$north, pch=18)
title("Cubic trend surface")
table(exget$ex.Zn.s3.o.in)
```

rast.put

*Exports to GRASS raster data files***Description**

`rast.put()` moves a single numeric vector to GRASS, using the metadata from `gmeta()`. If the compiled function is loaded, ordered factors are assigned category labels from the factor levels.

Usage

```
rast.put(G, lname="", layer, title="", cat=FALSE, DCELL=FALSE, breaks=NULL,
        col=NULL, nullcol=NULL, defcol=NULL, debug=FALSE, interp=FALSE, check=TRUE)
```

Arguments

<code>G</code>	GRASS metadata from <code>gmeta()</code>
<code>lname</code>	Name for new GRASS raster data file
<code>layer</code>	numeric vector or factor for export with length of <code>G\$Ncells</code> in GRASS order
<code>title</code>	character string to describe new GRASS raster data file
<code>cat</code>	if TRUE, the layer is read into GRASS as integer, for codes of category factors
<code>DCELL</code>	if TRUE, the layer is read into GRASS in double precision, otherwise (default) in single precision, unless the GRASS environment variable <code>GRASS_FP_DOUBLE</code> is set, overriding <code>DCELL</code>
<code>breaks</code>	a set of breakpoints for quantization if required: must give one more breakpoint than colour; only for <code>cat=FALSE</code> and <code>interp=FALSE</code> .
<code>col</code>	a list of colours such as that generated by ‘rainbow’, ‘heat.colors’, ‘topo.colors’, ‘terrain.colors’ or similar functions; the default is a grey ramp; only for <code>interp=FALSE</code> , for <code>cat=TRUE</code> must equal number of factor levels.
<code>nullcol</code>	Colour to be used for NA values - "honeydew"
<code>defcol</code>	Default colour to be used where no other is defined - "pale turquoise"
<code>debug</code>	if TRUE, the temporary ASCII file used for transfer in interpreted mode is not deleted on exit from the function
<code>interp</code>	if TRUE, the interpreted version of the function is used instead of the loaded compiled version
<code>check</code>	if TRUE, existing raster layers will not be overwritten

Details

If `cat=TRUE`, then the R factor layer is moved to the current GRASS location and mapset as a CELL layer, with labels taken from the factor levels; for this case `breaks` is disregarded (they are implicit in the factor), and `col` must be either `NULL` or a vector of colours equal in length to the number of levels. When `col=NULL`, `grey()` is used to generate the colours. If `cat=FALSE`, the R numeric layer is moved to GRASS as an FCELL or DCELL layer (depending on `DCELL` or the GRASS environmental variable `GRASS_FP_DOUBLE`). In this case `breaks` are used for quantization and colour assignment in connection with `col`. If `breaks=NULL`, `pretty()` is used to yield about 20 "nice" values for quantization, and their values are used to generate labels.

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[rast.get](#)

Examples

```
if(!get("maas.loc", env = .GRASS.meta)) make.maas.location()
data(utm.maas)
G <- gmeta()
inregion <- (utm.maas$east >= G$w & utm.maas$east <= G$e) & (utm.maas$north >= G$s & utm.maas$north <= G$N)
if(all(!inregion)) stop("None of the site locations are inside the current GRASS region")
if(any(!inregion)) warning("Some site locations are outside the current GRASS region")
require(spatial)
s3 <- trmat.G(G, surf.ls(3, utm.maas$east, utm.maas$north,
  utm.maas$Zn))*maasmask
Zn.s3.o <- as.ordered(cut(s3, labels=c("insignificant", "low",
  "medium", "high", "crisis"), breaks=c(-300, 200, 400, 700, 1000, 5000), include.lowest=TRUE))
nameR <- c("ex.tr3.in", "ex.Zn.s3.o.in")
rast.put(G, lname=nameR[1], layer=s3, title="Cubic trend surface",
  check=FALSE)
rast.put(G, lname=nameR[2], layer=Zn.s3.o, cat=TRUE,
  title="Interpolated surface categories", check=FALSE)
```

sites.get

Import GRASS sites file

Description

sites.get moves one GRASS 5.0 sites file to a data frame, returning the filled object.

Usage

```
sites.get(G, slist="", all.sites=FALSE, collapse.labels=TRUE, debug=FALSE, interp=FALSE)
```

Arguments

<code>G</code>	GRASS metadata from <code>gmeta()</code>
<code>slist</code>	GRASS sites file name
<code>all.sites</code>	if FALSE, retrieve only sites in current GRASS region, if TRUE, retrieve all sites
<code>collapse.labels</code>	If the sites file is old-style or malformed (d.sites.pg map=), collapse label strings to single string per point
<code>debug</code>	if TRUE, the temporary ASCII file used for transfer in interpreted mode is not deleted on exit from the function
<code>interp</code>	if TRUE, the interpreted version of the function is used instead of the loaded compiled version

Value

A data frame with columns: east, north, and var1, ..., varn, where n is the number of attributes held in the GRASS sites file, some of which may be factors, and which may include the site id number. Under the interpreted interface, the attributes are named var1, var2, etc., under the compiled interface as id, str1, str2, etc., num1, num2, etc., unless the sites file has a labels header containing the same number of labels as the number of columns in the data table returned, in which case the labels are used.

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[sites.put](#)

Examples

```

if(!get("maas.loc", env = .GRASS.meta)) make.maas.location()
G <- gmeta()
example(sites.put)
ex.Zn.log <- sites.get(G, slist="ex.Znlog.in")
mean(ex.Zn.log$log.Zn - log(utm.maas$Zn))
ex.Zn.cat <- sites.get(G, slist="ex.Zncat.in")
table(Zn.o, ex.Zn.cat$Zn.o)
utm.maas.new <- sites.get(G, slist="ex.utm.maas")
names(utm.maas.new)
table(Zn.o, utm.maas.new$Zn.o)

```

sites.put

Export GRASS sites file

Description

`sites.put()` moves a single numeric or character vector site to GRASS, using the metadata from `gmeta()`. `sites.put2()` moves a data frame to a GRASS sites list, and requires the specification of the columns of the data frame to be treated as dimensions.

Usage

```

sites.put(G, lname="", east, north, var, debug=FALSE)
sites.put2(G, data, id=NULL, dims, lname="", all.sites=FALSE,
check=TRUE)

```

Arguments

<code>G</code>	GRASS metadata from <code>gmeta()</code>
<code>lname</code>	Name for new GRASS sites file
<code>east</code>	Eastings of points
<code>north</code>	Northings of points
<code>var</code>	numeric or character vector of attribute values at these points for export
<code>debug</code>	if TRUE, the temporary ASCII file used for transfer is not deleted on exit from the function
<code>data</code>	data frame to be exported as a GRASS 5.0 format sites list
<code>id</code>	default NULL, can be the index or name of a numeric column in data
<code>dims</code>	character vector of column names or integer vector of column indices in data for sites list coordinates, expected to be in x, y, z1, ... order
<code>all.sites</code>	if FALSE, retrieve only sites in current GRASS region, if TRUE, retrieve all sites
<code>check</code>	check to see if <code>lname</code> already exists as a sites list in the GRASS database, and abort if it does; if FALSE <code>sites.put2()</code> will overwrite existing files (default TRUE)

Value

none

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[sites.get](#)

Examples

```
if(!get("maas.loc", env = .GRASS.meta)) make.maas.location()
data(utm.maas)
Zn.o <- as.ordered(cut(utm.maas$Zn, labels=c("insignificant", "low",
"medium", "high", "crisis"), breaks=c(100, 200, 400, 700, 1000, 2000),
include.lowest=TRUE))
G <- gmeta()
nameQ <- c("ex.Zn.in", "ex.Znlog.in", "ex.Zncat.in")
Zn <- data.frame(east=utm.maas$east, north=utm.maas$north, Zn=utm.maas$Zn)
sites.put2(G, Zn, dims=c("east", "north"), lname=nameQ[1], check=FALSE)
log.Zn <- data.frame(east=utm.maas$east, north=utm.maas$north,
  log.Zn=log(utm.maas$Zn))
sites.put2(G, log.Zn, dims=c("east", "north"), lname=nameQ[2], check=FALSE)
Zn.o.df <- data.frame(east=utm.maas$east, north=utm.maas$north, Zn.o=Zn.o)
sites.put2(G, Zn.o.df, dims=c("east", "north"), lname=nameQ[3], check=FALSE)
newdf <- data.frame(utm.maas, Zn.o)
sites.put2(G, newdf, dims=c("east", "north", "elev"), lname="ex.utm.maas",
  check=FALSE)
```

summary.grassmeta *Display GRASS metadata*

Description

summary.grassmeta() displays the metadata collected by gmeta().

Usage

```
## S3 method for class 'grassmeta':  
summary(object, ...)
```

Arguments

object	GRASS metadata from gmeta()
...	other generic arguments

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. Computers and Geosciences, 26, pp. 1043-1052.

See Also

[gmeta](#)

Examples

```
if(!get("maas.loc", env = .GRASS.meta)) make.maas.location()  
G <- gmeta()  
summary(G)
```

`trmat.G`*Evaluate Trend Surface over a GRASS Grid*

Description

A wrapper for `trmat()` from the `spatial` package in the VR bundle, using the metadata of the underlying GRASS LOCATION.

Usage

```
trmat.G(G, obj, east=NULL, north=NULL)
```

Arguments

<code>G</code>	GRASS metadata from <code>gmeta()</code>
<code>obj</code>	object returned by <code>surf.ls</code> or <code>surf.gls</code>
<code>east</code>	see <code>help(east)</code>
<code>north</code>	see <code>help(north)</code>

Value

A vector of `G$Ncells` values is returned, with the results of interpolation ordered as a GRASS raster file.

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Author(s)

Original `trmat()`: Brian D. Ripley and Bill Venables as detailed in the `spatial` package documentation; adapted by Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no.>

References

<http://grass.itc.it/statsgrass/index.html>, Bivand, R. S., (2000) Using the R statistical data analysis language on GRASS 5.0 GIS data base files. *Computers and Geosciences*, 26, pp. 1043-1052; Unwin, D. J., Wrigley, N. (1987) Towards a general-theory of control point distribution effects in trend surface models. *Computers and Geosciences*, 13, pp. 351-355.

See Also

[trmat](#)

Examples

```

data(utm.maas)
G <- maas.metadata
if(G$Ncells != length(maasmask))
  stop("example not running in Maas GRASS location")
require(spatial)
s1 <- surf.ls(1, utm.maas$east, utm.maas$north, utm.maas$Zn)
s2 <- surf.ls(2, utm.maas$east, utm.maas$north, utm.maas$Zn)
s3 <- surf.ls(3, utm.maas$east, utm.maas$north, utm.maas$Zn)
s4 <- surf.ls(4, utm.maas$east, utm.maas$north, utm.maas$Zn)
summary(s2)
summary(s3)
summary(s4)
anova(s2,s3,s4)
plot(G, trmat.G(G, s3)*maasmask, breaks=c(-10000, seq(-100, 1500, 200),
  10000), col=c("yellow", grey(8:1/8), "red"))
plot(G, ifelse(is.na(maasmask), 1, NA), breaks=c(0,2), col="wheat", add=TRUE)
plot(s3, add=TRUE)
legend(c(269900, 270600), c(5652000, 5652900), bty="n", bg="wheat",
  legend=c("> 1500", rev(legtext(seq(-100,1500,200))), "< -100", "mask=NA"),
  fill=c("red", rev(grey(8:1/8)), "yellow", "wheat"))
oldpar <- par(mfrow=c(2,2))
plot(G, trmat.G(G, s1)*maasmask, breaks=c(-10000, seq(-100, 1500, 200),
  10000), col=c("yellow", grey(8:1/8), "red"))
plot(G, ifelse(is.na(maasmask), 1, NA), breaks=c(0,2), col="wheat", add=TRUE)
plot(s1, add=TRUE)
plot(G, trmat.G(G, s2)*maasmask, breaks=c(-10000, seq(-100, 1500, 200),
  10000), col=c("yellow", grey(8:1/8), "red"))
plot(G, ifelse(is.na(maasmask), 1, NA), breaks=c(0,2), col="wheat", add=TRUE)
plot(s2, add=TRUE)
plot(G, trmat.G(G, s3)*maasmask, breaks=c(-10000, seq(-100, 1500, 200),
  10000), col=c("yellow", grey(8:1/8), "red"))
plot(G, ifelse(is.na(maasmask), 1, NA), breaks=c(0,2), col="wheat", add=TRUE)
plot(s3, add=TRUE)
plot(G, trmat.G(G, s4)*maasmask, breaks=c(-10000, seq(-100, 1500, 200),
  10000), col=c("yellow", grey(8:1/8), "red"))
plot(G, ifelse(is.na(maasmask), 1, NA), breaks=c(0,2), col="wheat", add=TRUE)
plot(s4, add=TRUE)
par(oldpar)

```

 utm.maas

Soil pollution data set

Description

The utm.maas data frame has 98 rows and 13 columns. It records contamination of the environment by selected metals along the Dutch bank of the River Maas just north of Maastricht and contains the following columns (topsoil data were collected as bulked samples in 1990 within a radius of 5m according to Burrough and McDonnell 1998: 102, 309):

Usage

```
data(utm.maas)
```

Format

This data frame contains the following columns:

east a numeric vector - UTM zone 32 eastings coordinates m
north a numeric vector - UTM zone 32 northings coordinates m
x a numeric vector - local coordinates m
y a numeric vector - local coordinates m
elev a numeric vector - elevation above local reference level in meters
d.river a numeric vector - distance from main River Maas channel in meters
Cd a numeric vector - Cadmium in ppm
Cu a numeric vector - Copper in ppm
Pb a numeric vector - Lead in ppm
Zn a numeric vector - Zinc in ppm
LOI a numeric vector - percentage organic matter loss on ignition
Fldf a numeric vector - flood frequency class, 1: annual, 2: 2-5 years, 3: every 5 years
Soil a numeric vector - 3 unnamed soil types

Details

Maas river bank soil pollution data

The Maas river bank soil pollution data (Limburg, The Netherlands) are sampled along the Dutch bank of the river Maas (Meuse) north of Maastricht. This is a flood plain of the river Maas, not far from where the Maas enters the Netherlands (Borgharen, Ifteren, about 3 to 5 km north of Maastricht).

The river Maas is at the north-west border of the project area, traversing the area in north-east direction. Burrough et al. 1998 use a subset of the same data set in their book (reduced area). The data have been re-projected for the GRASS/R interface from the Dutch standard coordinate system (TDN coordinates in stereographic projection) to UTM coordinate system zone 32, on WGS84 ellipsoid. A GRASS LOCATION has to be defined with following parameters: projection UTM, ellipsoid WGS84, zone 32, north 5652930, south 5650610, west 269870, east 272460, nsres 10, ewres 10, rows 232, cols 259. The pre-defined location including the data stored column-wise in sites lists can be downloaded from the GRASS web site, and is also contained in the "grassmeta" object `maas.metadata`.

Maas river bank dat GRASS LOCATION, http://grass.itc.it/statsgrass/maas_grass_location.tar.gz

Note

The functions in this package are intended to work with the GRASS geographical information system. The examples for wrapper functions will work whether or not R is running in GRASS, and whether or not the current location is that of the data set used for the examples. Examples of interface functions will however (from version 0.2-2) only work outside GRASS, to avoid possible overwriting of GRASS database locations and/or files.

Source

Burrough, P. & McDonnell, R. (1998) *Principles of geographical information systems*, New York: Oxford, pp. 309–311; Rikken, M.G.J., Van Rijn, R.P.G., 1993. Soil pollution with heavy metals — An Inquiry into Spatial Variation, Cost of Mapping and the Risk evaluation of Copper, Cadmium, Lead and Zinc in the Floodplains of the Meuse West of Stein, The Netherlands. Doctoraalveldwerkverslag, Dept. of Physical Geography, Utrecht University.

Examples

```
data(utm.maas)
Zn.o <- as.ordered(cut(utm.maas$Zn, labels=c("insignificant", "low",
"medium", "high", "crisis"), breaks=c(100, 200, 400, 700, 1000, 2000),
include.lowest=TRUE))
table(Zn.o)
plot(utm.maas$east, utm.maas$north, pch=unclass(Zn.o), xlab="", ylab="", asp=1)
legend(x=c(269900, 270500), y=c(5652000, 5652700), pch=c(1:5),
legend=levels(Zn.o))
cat("Burrough & McDonnell p. 107\n")
round(tapply(utm.maas$Zn, as.factor(utm.maas$Fldf), mean), 2)
round(tapply(utm.maas$Zn, as.factor(utm.maas$Fldf), sd), 2)
round(tapply(log(utm.maas$Zn), as.factor(utm.maas$Fldf), mean), 3)
round(tapply(log(utm.maas$Zn), as.factor(utm.maas$Fldf), sd), 3)
round(exp(round(tapply(log(utm.maas$Zn), as.factor(utm.maas$Fldf), mean), 3)), 2)
hist(utm.maas$Zn, breaks=seq(0,2000,100), col="grey")
hist(log(utm.maas$Zn), breaks=seq(3.5,8.5,0.25), col="grey")
t.test(utm.maas$Zn[utm.maas$Fldf == 2], utm.maas$Zn[utm.maas$Fldf == 3])
cat("NB: B&McD p. 108, their relative variance is (1-(RSquared))\n")
anova(lm(Zn ~ as.factor(Fldf), data=utm.maas))
summary(lm(Zn ~ as.factor(Fldf), data=utm.maas))
anova(lm(log(Zn) ~ as.factor(Fldf), data=utm.maas))
summary(lm(log(Zn) ~ as.factor(Fldf), data=utm.maas))
```

Index

*Topic **IO**

- contourG, 2
- get.mapsets, 3
- gmeta, 4
- GRASS.connect, 6
- kde2d.G, 7
- plot.grassmeta, 11
- rast.get, 12
- rast.put, 14
- sites.get, 16
- sites.put, 18
- summary.grassmeta, 19
- trmat.G, 20

*Topic **datasets**

- pcbs, 9
- utm.maas, 22

*Topic **hplot**

- plot.grassmeta, 11

*Topic **spatial**

- contourG, 2
- get.mapsets, 3
- gmeta, 4
- GRASS.connect, 6
- kde2d.G, 7
- list.grass, 9
- plot.grassmeta, 11
- rast.get, 12
- rast.put, 14
- sites.get, 16
- sites.put, 18
- summary.grassmeta, 19
- trmat.G, 20

contour, 3

contourG, 2

east (*gmeta*), 4

get.cygwinstring (*GRASS.connect*),
6

get.GISDBASE (*GRASS.connect*), 6

get.GRASSChkGISRC
(*GRASS.connect*), 6

get.LOCATION (*GRASS.connect*), 6

get.MAPSET (*GRASS.connect*), 6

get.mapsets, 3

gmeta, 4, 20

GRASS.connect, 6

image, 12

kde2d, 8

kde2d.G, 7

leglabs (*plot.grassmeta*), 11

legtext (*plot.grassmeta*), 11

list.grass, 9

maas.metadata (*utm.maas*), 22

maasmask (*utm.maas*), 22

make.maas.location (*gmeta*), 4

north (*gmeta*), 4

obsno (*gmeta*), 4

pcbs, 9

plot.grassmeta, 3, 11

plot.window, 3, 12

print.glist (*list.grass*), 9

rast.get, 12, 15

rast.put, 14, 14

refresh.mapsets (*get.mapsets*), 3

reverse (*gmeta*), 4

set.cygwinstring (*GRASS.connect*),
6

set.GISDBASE (*GRASS.connect*), 6

set.LOCATION (*GRASS.connect*), 6

set.MAPSET (*GRASS.connect*), 6

`sites.get`, [16](#), [19](#)
`sites.put`, [17](#), [18](#)
`sites.put2(sites.put)`, [18](#)
`summary.grassmeta`, [5](#), [19](#)

`trmat`, [21](#)
`trmat.G`, [20](#)

`utm.maas`, [22](#)