

Package ‘spgrass6’

November 15, 2009

Version 0.6-12

Date 2009-11-08

Title Interface between GRASS 6 geographical information system and R

Author Roger Bivand

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

Description Interpreted interface between GRASS 6 geographical information system and R, based on starting R from within the GRASS environment, or running free-standing R in a temporary GRASS location; the package provides facilities for using all GRASS commands from the R command line.

Depends R (>= 2.4), sp (>= 0.9), rgdal (>= 0.5.26), XML

SystemRequirements GRASS (>= 6.3)

License GPL (>= 2)

URL <http://grass.osgeo.org/>

Collate AAA.R spgrass6.R bin_link.R vect_link.R initGRASS.R xml1.R zzz.R

Repository CRAN

Date/Publication 2009-11-15 13:51:29

R topics documented:

spgrass6-package	2
execGRASS	3
gmeta6	4
initGRASS	5
readRAST6	7
readVECT6	9

Index	12
--------------	-----------

spgrass6-package *Interface between GRASS 6.0 geographical information system and R*

Description

Interpreted interface between GRASS 6.0 geographical information system and R, based on starting R from within the GRASS environment. The interface uses classes defined in the sp package to hold spatial data.

Details

Index:

readRAST6	read GRASS 6 raster files
writeRAST6	write GRASS 6 raster files
readVECT6	read GRASS 6 vector object files
writeVECT6	write GRASS 6 vector object files
gmeta6	read GRASS metadata from the current LOCATION
getLocationProj	return a PROJ.4 string of projection information
gmeta2grd	create a GridTopology object from the GRASS region
vInfo	return vector geometry information
vColumns	return vector database columns information
vDataCount	return count of vector database rows
vect2neigh	return area neighbours with shared boundary length

Further information may be found in the document `doc/spgrass_0.3.pdf` in the directory returned by `system.file("", package="spgrass6")`.

Author(s)

Roger Bivand

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

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  soilsph <- readRAST6("soils.ph", ignore.stderr=TRUE, plugin=FALSE)
  summary(soilsph)
  grd <- gmeta2grd(ignore.stderr=TRUE)
  grd
  set.seed(1)
  smple <- overlay(soilsph, spsample(soilsph, 200, "random"))
  summary(smple)
  writeVECT6(smple, "sp_dem", v.in.ogr_flags="overwrite", ignore.stderr=TRUE)
  bugsDF <- readVECT6("bugsites", ignore.stderr=TRUE, mapset="PERMANENT")
  summary(bugsDF)
  vInfo("streams", ignore.stderr=TRUE)
  vColumns("streams", ignore.stderr=TRUE)
```

```

vDataCount("streams", ignore.stderr=TRUE)
streams <- readVECT6("streams", type="line,boundary", remove.duplicates=FALSE, ignore.stde
summary(streams)
}

```

execGRASS

Run GRASS commands

Description

The functions provide an interface to GRASS commands run through `system`, based on the values returned by the `--interface` description flag using XML parsing.

Usage

```

execGRASS(cmd, flags = NULL, parameters = NULL, intern = FALSE, ignore.stderr = FALSE)
doGRASS(cmd, flags = NULL, parameters = NULL)
parseGRASS(cmd)
## S3 method for class 'GRASS\_interface\_desc':
print(x, ...)
getXMLencoding()
setXMLencoding(enc)

```

Arguments

<code>cmd</code>	GRASS command name
<code>flags</code>	character vector of GRASS command flags
<code>parameters</code>	list of GRASS command parameters
<code>intern</code>	a logical (not 'NA') which indicates whether to make the output of the command an R object. Not available unless 'popen' is supported on the platform
<code>ignore.stderr</code>	a logical indicating whether error messages written to 'stderr' should be ignored
<code>x</code>	object to be printed
<code>...</code>	other arguments to print method
<code>enc</code>	character string to replace UTF-8 in header of XML data generated by GRASS module <code>-interface-description</code> output when the internationalised messages are not in UTF-8 (known to apply to French, which is in latin1)

Details

`parseGRASS` checks to see whether the GRASS command has been parsed already and cached in this session; if not, it reads the interface description, parses it and caches it for future use. `doGRASS` assembles a proposed GRASS command with flags and parameters as a string, wrapping `parseGRASS`, and `execGRASS` is a wrapper for `doGRASS`, running the command through `system`. The command string is termed proposed, because not all of the particular needs of commands are provided by the interface description, and no check is made for the existence of input objects. Support for multiple parameter values added with help from Patrick Caldon.

Value

`parseGRASS` returns a `GRASS_interface_desc` object, `doGRASS` returns a character string with a proposed GRASS command, and `execGRASS` returns what `system` returns, particularly depending on the `intern` argument.

Note

If any package command fails with a UTF-8 error from the XML package, try using `setXMLencoding` to work around the problem that GRASS modules declare `-interface-description` output as UTF-8 without ensuring that it is (French is of 6.4.0 RC5 latin1).

Author(s)

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

See Also

[system](#)

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  print(parseGRASS("r.slope.aspect"))
  pars <- list(elevation="elevation.dem", slope="slope", aspect="aspect")
  doGRASS("r.slope.aspect", flags=c("overwrite"), parameters=pars)
  print(parseGRASS("r.buffer"))
  pars <- list(input="bugsites", output="bmap", distances=seq(1000,15000,1000))
  doGRASS("r.buffer", flags=c("overwrite"), parameters=pars)
}
```

gmeta6

Reads GRASS metadata from the current LOCATION

Description

GRASS LOCATION metadata are read into a list in R; helper function `getLocationProj` returns an sproj-compliant PROJ.4 string of projection information. The helper function `gmeta2grd` creates a GridTopology object from the current GRASS mapset region definitions.

Usage

```
gmeta6(ignore.stderr = FALSE)
getLocationProj(ignore.stderr = FALSE)
gmeta2grd(ignore.stderr = FALSE)
## S3 method for class 'gmeta6':
print(x, ...)
```

Arguments

`ignore.stderr` default FALSE, can be set to TRUE to silence `system()` output to standard error; does not apply on Windows platforms

`x` S3 object returned by `gmeta6`

`...` arguments passed through print method

Value

Returns list of `g.gisenv`, `g.region -g3`, and `g.proj` values

Author(s)

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

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  G <- gmeta6()
  print(G)
  CRS(getLocationProj())
  grd <- gmeta2grd()
  print(grd)
  ncells <- prod(slot(grd, "cells.dim"))
  df <- data.frame(k=rep(1, ncells))
  mask_SG <- SpatialGridDataFrame(grd, data=df)
  print(summary(mask_SG))
}
```

initGRASS

Initiate GRASS session

Description

Run GRASS interface in an R session not started within GRASS. The function initializes environment variables used by GRASS, the `.gisrc` used by GRASS for further environment variables, and a temporary location.

Usage

```
initGRASS(gisBase, home, SG, gisDbase, location, mapset, override = FALSE)
```

Arguments

<code>gisBase</code>	The directory path to GRASS binaries and libraries
<code>home</code>	The directory in which to create the <code>.gisrc</code> file; defaults to <code>\$HOME</code> on Unix systems and to <code>USERPROFILE</code> on Windows systems; can usually be set to <code>tempdir()</code>
<code>SG</code>	An optional <code>SpatialGrid</code> object to define the <code>DEFAULT_WIND</code> of the temporary location
<code>gisDbase</code>	if missing, <code>tempdir()</code> will be used; GRASS GISDBASE directory for the working session
<code>location</code>	if missing, <code>basename(tempfile())</code> will be used; GRASS location directory for the working session
<code>mapset</code>	if missing, <code>basename(tempfile())</code> will be used; GRASS mapset directory for the working session
<code>override</code>	default <code>FALSE</code> , set to <code>TRUE</code> if accidental trashing of GRASS <code>.gisrc</code> files and locations is not a problem

Details

The function establishes an out-of-GRASS working environment providing GRASS commands with the environment variable support required, and may also provide a temporary location for use until the end of the running R session if the `home` argument is set to `tempdir()`, and the `gisDbase` argument is not given. Running `gmeta6` shows where the location is, should it be desired to archive it before leaving R.

Value

The function runs `gmeta6` before returning the current values of the running GRASS session that it provides.

Note

If any package command fails with a UTF-8 error from the XML package, try using `setXMLencoding` to work around the problem that GRASS modules declare `-interface-description` output as UTF-8 without ensuring that it is (French is of 6.4.0 RC5 latin1).

Author(s)

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

See Also

[gmeta6](#)

Examples

```
## Not run:
initGRASS("/usr/local/grass-6.4.0", home=tempdir())
initGRASS("C:/GRASS", home=tempdir())

## End(Not run)
```

readRAST6

Read and write GRASS 6 raster files

Description

Read GRASS 6 raster files from GRASS 6 into R SpatialGridDataFrame objects, and write single columns of R SpatialGridDataFrame objects to GRASS 6. `readRAST6` and `writeRAST6` use temporary binary files and `r.out.bin` and `r.in.bin` rather than the temporary ASCII files used in earlier implementations. The earlier versions may still be used in a transition period.

Usage

```
readRAST6(vname, cat=NULL, ignore.stderr = FALSE, NODATA=NULL, plugin=NULL, mapset=
writeRAST6(x, vname, zcol = 1, NODATA=NULL, ignore.stderr = FALSE, useGDAL=TRUE, ov
```

Arguments

<code>vname</code>	A vector of GRASS 6.0 raster file names
<code>cat</code>	default NULL; if not NULL, must be a logical vector matching <code>vname</code> , stating which (CELL) rasters to return as factor
<code>ignore.stderr</code>	default FALSE, can be set to TRUE to silence <code>system()</code> output to standard error; does not apply on Windows platforms
<code>plugin</code>	default NULL does auto-detection, changes to FALSE if <code>vname</code> is longer than 1, and a sanity check will be run on raster and current region, and the function will revert to FALSE if mismatch is found; if TRUE, the plugin is available and the raster should be read in its original region and resolution; if the plugin is used, no further arguments other than <code>mapset</code> are respected
<code>mapset</code>	default NULL, if <code>plugin</code> is TRUE, the <code>mapset</code> of the file to be imported will be autodetected; if not NULL and if <code>plugin</code> is TRUE, a character string overriding the autodetected <code>mapset</code> , otherwise ignored

useGDAL	default TRUE use r.out.gdal or plugin and readGDAL if autodetected or plugin=TRUE; or for writing writeGDAL, GTiff, and r.in.gdal, if FALSE using r.out.bin or r.in.bin
x	A SpatialGridDataFrame object for export to GRASS as a raster layer
zcol	Attribute column number or name
NODATA	by default NULL, in which case it is set to one less than floor() of the data values, otherwise an integer NODATA value (required to be integer by GRASS r.out.bin)
overwrite	default FALSE, if TRUE inserts "overwrite" into the value of the flags argument if not already there to allow existing GRASS rasters to be overwritten
flags	default NULL, character vector, for example "overwrite"

Value

readRAST6 returns a SpatialGridDataFrame objects with an data.frame in the data slots, and with the projection argument set. Note that the projection argument set is the the GRASS rendering of proj4, and will differ from the WKT/ESRI rendering returned by readVECT6 in form but not meaning. They are exchangeable but not textually identical, usually with the +ellps= term replaced by ellipsoid parameters verbatim.

Author(s)

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

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  spear <- readRAST6(c("geology", "elevation.dem"), cat=c(TRUE, FALSE),
    ignore.stderr=TRUE, plugin=NULL)
  spear <- readRAST6(c("geology", "elevation.dem"), cat=c(TRUE, FALSE),
    ignore.stderr=TRUE, useGDAL=FALSE, plugin=FALSE)
  spear <- readRAST6(c("geology", "elevation.dem"), cat=c(TRUE, FALSE),
    ignore.stderr=TRUE, useGDAL=TRUE, plugin=FALSE)
  print(table(spear$geology))
  execGRASS("r.stats", flags=c("c", "l", "quiet"),
    parameters=list(input="geology"), ignore.stderr=TRUE)
  boxplot(spear$elevation.dem ~ spear$geology)
  spear$sqdem <- sqrt(spear$elevation.dem)
  if ("GRASS" %in% gdalDrivers()$name) {
    dem1 <- readRAST6("elevation.dem", plugin=TRUE, ignore.stderr=TRUE,
      mapset="PERMANENT")
    print(summary(dem1))
  }
  writeRAST6(spear, "sqdemSP", zcol="sqdem", ignore.stderr=TRUE)
  execGRASS("r.info", parameters=list(map="sqdemSP"), ignore.stderr=TRUE)
  execGRASS("g.remove", parameters=list(rast="sqdemSP"), ignore.stderr=TRUE)
  writeRAST6(spear, "sqdemSP", zcol="sqdem", ignore.stderr=TRUE, useGDAL=TRUE)
  execGRASS("r.info", parameters=list(map="sqdemSP"), ignore.stderr=TRUE)
  execGRASS("g.remove", parameters=list(rast="sqdemSP"), ignore.stderr=TRUE)
  execGRASS("r.mapcalculator", parameters=list(outfile="quads0",
```

```

    amap="quads", formula='A - 1'), ignore.stderr=TRUE)
execGRASS("r.stats", flags="c", parameters=list(input="quads0"),
  ignore.stderr=TRUE)
quads0 <- readRAST6("quads0", ignore.stderr=TRUE)
print(table(quads0$quads0))
quads0 <- readRAST6("quads0", ignore.stderr=TRUE, plugin=FALSE)
print(table(quads0$quads0))
execGRASS("g.remove", parameters=list(rast="quads0"), ignore.stderr=TRUE)
}

```

readVECT6

Read and write GRASS 6 vector object files

Description

readVECT6 moves one GRASS 6.0 vector object file with attribute data through a temporary shapefile to a Spatial*DataFrame object of type determined by the GRASS 6.0 vector object; writeVECT6 moves a Spatial*DataFrame object through a temporary shapefile to a GRASS vector object file. vect2neigh returns neighbour pairs with shared boundary length as described by Markus Neteler, in <https://stat.ethz.ch/pipermail/r-sig-geo/2005-October/000616.html>. cygwin_clean_temp can be called to try to clean the GRASS mapset-specific temporary directory under cygwin.

Usage

```

readVECT6(vname, type=NULL, plugin=NULL, remove.duplicates = TRUE, ignore.stderr =
writeVECT6(SDF, vname, v.in.ogr_flags=NULL, ignore.stderr = FALSE)
vInfo(vname, ignore.stderr = FALSE)
vColumns(vname, ignore.stderr = TRUE)
vDataCount(vname, ignore.stderr = TRUE)
vect2neigh(vname, ID=NULL, ignore.stderr = FALSE)
putSites6sp(SPDF, vname, ignore.stderr = FALSE)
putSites6(df, vname, ignore.stderr = FALSE)
getSites6sp(vname, ignore.stderr = FALSE, with_prj=TRUE)
getSites6(vname, ignore.stderr = FALSE, with_prj=TRUE)
cygwin_clean_temp(verbose=TRUE, ignore.stderr = FALSE)

```

Arguments

vname	A GRASS 6.0 vector file name
type	override type detection when multiple types are non-zero, passed to v.out.ogr
plugin	default NULL for auto-detection, may be set to FALSE to avoid or TRUE if the plugin is known to be available; if the plugin is used, no further arguments other than mapset are respected
remove.duplicates	In line and area vector objects, multiple geometrical features may be associated with a single cat number, leading to duplication of data rows; this argument

	attempts to combine the geometrical features so that they match a single data row
<code>ignore.stderr</code>	default FALSE, can be set to TRUE to silence <code>system</code> output to standard error; does not apply on Windows platforms
<code>with_prj</code>	default TRUE, write ESRI-style PRJ file for transferred data
<code>with_c</code>	if TRUE, export features with category (labeled) only; by default all features are exported
<code>mapset</code>	if plugin is TRUE, the mapset of the file to be imported may be changed from the current mapset by passing a character string
<code>pointDropZ</code>	default FALSE, if TRUE, discard third coordinates for point geometries; third coordinates are always discarded for line and polygon geometries
<code>SDF</code>	A <code>Spatial*DataFrame</code> to be moved to GRASS6 as a vector object, for <code>SpatialPointsDataFrame</code> , <code>SpatialLinesDataFrame</code> , and <code>SpatialPolygonsDataFrame</code> objects
<code>v.in.ogr_flags</code>	Character vector containing additional optional flags and/or options for <code>v.in.ogr</code> , particularly "o" and "overwrite"
<code>ID</code>	A valid DB column name for unique identifiers (optional)
<code>SPDF</code>	A <code>SpatialPointsDataFrame</code> to be moved to GRASS6 as vector points
<code>df</code>	A data frame with at least columns named x, y and cat (no z support yet)
<code>verbose</code>	TRUE, can be set to false

Value

`readVECT6` imports a GRASS6 vector object into a `Spatial*DataFrame` object with the type determined by the type of the GRASS6 vector object; `getSites6` returns a data frame. `vect2neigh` returns a data frame object with left and right neighbours and boundary lengths, also given class `GRASSneigh` and `spatial.neighbour` (as used in `spdep`). The incantation to retrieve the neighbours list is `sn2listw(vect2neigh())$neighbours`, and to retrieve the boundary lengths: `sn2listw(vect2neigh())$weights`. The `GRASSneigh` object has two other useful attributes: `external` is a vector giving the length of shared boundary between each polygon and the external area, and `total` giving each polygon's total boundary length.

Note

Please note that the OGR drivers used may not handle missing data gracefully. From `rgdal` release 0.5-27, missing values are taken as unset OGR field values. If the OGR driver encodes them in this way, NAs will be moved across the interface correctly from R to GRASS, and from GRASS to R using the OGR GRASS vector plugin. Work is continuing to correct `v.out.ogr` so that it emits unset fields, which affects users with no OGR GRASS plugin for the present. Thanks to Dylan Beaudette for helping with missing data handling.

Please also note that, on Windows and Cygwin systems, the temporary shapefiles are not removed by the interface functions, nor can GRASS remove them on termination - they must for the time being be removed manually. Windows believes that the GDAL/OGR library is still using them.

Author(s)

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

Examples

```
if (nchar(Sys.getenv("GISRC")) > 0) {
  execGRASS("v.info", parameters=list(map="bugsites"))
  print(vInfo("bugsites", ignore.stderr=TRUE))
  bugs <- readVECT6("bugsites", ignore.stderr=TRUE, plugin=NULL)
  print(summary(bugs))
  bugsl <- readVECT6("bugsites", ignore.stderr=TRUE, plugin=FALSE)
  print(summary(bugsl))
  writeVECT6(bugs, "newbugs", v.in.ogr_flags=c("o", "overwrite"),
    ignore.stderr=TRUE)
  execGRASS("v.info", parameters=list(map="newbugs"))
  nbugs <- readVECT6("newbugs", ignore.stderr=TRUE)
  print(summary(nbugs))
  print(vInfo("roads", ignore.stderr=TRUE))
  roads <- readVECT6("roads", ignore.stderr=TRUE)
  print(summary(roads))
}
```

Index

*Topic **package**

spgrass6-package, 2

*Topic **spatial**

execGRASS, 3

gmeta6, 4

initGRASS, 5

readRAST6, 7

readVECT6, 9

spgrass6-package, 2

cygwin_clean_temp (readVECT6), 9

doGRASS (execGRASS), 3

execGRASS, 3

getLocationProj (gmeta6), 4

getSites6 (readVECT6), 9

getSites6sp (readVECT6), 9

getXMLencoding (execGRASS), 3

gmeta2grd (gmeta6), 4

gmeta6, 4, 6

initGRASS, 5

parseGRASS (execGRASS), 3

print.gmeta6 (gmeta6), 4

print.GRASS_interface_desc
(execGRASS), 3

putSites6 (readVECT6), 9

putSites6sp (readVECT6), 9

rast.get6 (readRAST6), 7

rast.put6 (readRAST6), 7

readBinGrid (readRAST6), 7

readCELL6sp (readRAST6), 7

readFLOAT6sp (readRAST6), 7

readRAST6, 7

readVECT6, 9

setXMLencoding (execGRASS), 3

spgrass6 (spgrass6-package), 2

spgrass6-package, 2

system, 4

vColumns (readVECT6), 9

vDataCount (readVECT6), 9

vect2neigh (readVECT6), 9

vInfo (readVECT6), 9

writeBinGrid (readRAST6), 7

writeRAST6 (readRAST6), 7

writeRast6sp (readRAST6), 7

writeVECT6 (readVECT6), 9