Package ‘rgrass’

October 14, 2022

Version 0.3-6
Date 2022-10-12
Title Interface Between ‘GRASS’ Geographical Information System and ‘R’

Description An interface between the ‘GRASS’ geographical information system (‘GIS’) and ‘R’, based on starting ‘R’ from within the ‘GRASS’ ‘GIS’ environment, or running a free-standing ‘R’ session in a temporary ‘GRASS’ location; the package provides facilities for using all ‘GRASS’ commands from the ‘R’ command line. The original interface package for ‘GRASS 5’ (2000-2010) is described in Bivand (2000) <doi:10.1016/S0098-3004(00)00057-1> and Bivand (2001) <https://www.r-project.org/conferences/DSC-2001/Proceedings/Bivand.pdf>. This was succeeded by ‘spgrass6’ for ‘GRASS 6’ (2006-2016) and ‘rgrass7’ for ‘GRASS 7’ (2015-present). The ‘rgrass’ package modernizes the interface for ‘GRASS 8’ while still permitting the use of ‘GRASS 7’.

Depends R (>= 3.3.0), XML
Imports stats, utils, methods
Suggests terra (>= 1.6-16), sp (>= 0.9), knitr, rmarkdown, sf, stars, raster (>= 3.6-3)

VignetteBuilder knitr
SystemRequirements GRASS (>= 7)
License GPL (>= 2)


BugReports https://github.com/rsbivand/rgrass/issues/


NeedsCompilation no

Author Roger Bivand [cre, aut] (<https://orcid.org/0000-0003-2392-6140>), Rainer Krug [ctb] (<https://orcid.org/0000-0002-7490-0066>),
rgrass-package

Description

Interpreted interface between GRASS geographical information system, versions 7 and 8, and R, based on starting R from within the GRASS environment, or on running R stand-alone and creating a throw-away GRASS environment from within R. The interface uses classes defined in the sp package to hold spatial data.

Details

Index:

- read_RAST: read GRASS raster files
- write_RAST: write GRASS raster files
- read_VECT: read GRASS vector object files
- write_VECT: write GRASS vector object files
- gmeta: read GRASS metadata from the current LOCATION
- getLocationProj: return a WKT2 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

Note that the examples now use the smaller subset North Carolina location: https://grass.osgeo.org/sampledata/north_carolina/nc_basic_spm_grass7.tar.gz
Author(s)

Roger Bivand

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

Examples

```
run <- FALSE
if (nchar(Sys.getenv("GISRC")) > 0 &&
    read.dcf(Sys.getenv("GISRC"))[1, "LOCATION_NAME"] == "nc_basic_spm_grass7") run <- TRUE
run <- run && require("terra", quietly=TRUE) && require("sp", quietly=TRUE)
if (run) {
  grd <- gmeta2 grd(ignore.stderr=TRUE)
  grd
}
if (run) {
  elevation <- read_RAST("elevation", ignore.stderr=TRUE)
}
if (run) {
  set.seed(1)
  smple <- spatSample(ext(elevation), lonlat=is.lonlat(elevation), size=200,
                      method="random", as.points=TRUE)
  values(smple) <- extract(elevation, smple)[,2]
  smple
}
if (run) {
  write_VECT(smple, "sp_dem", flags=c("overwrite", "o"), ignore.stderr=TRUE)
}
if (run) {
  schoolsDF <- read_VECT("schools", ignore.stderr=TRUE)
  summary(schoolsDF)
}
if (run) {
  vInfo("streams", ignore.stderr=TRUE)
}
if (run) {
  vColumns("streams", ignore.stderr=TRUE)
}
if (run) {
  vDataCount("streams", ignore.stderr=TRUE)
}
if (run) {
  streams <- read_VECT("streams", type="line", ignore.stderr=TRUE)
  summary(streams)
}
```

execGRASS

Run GRASS commands
Description

The functions provide an interface to GRASS commands run through the system, based on the values returned by the --interface description flag using XML parsing. If required parameters are omitted, and have declared defaults, the defaults will be used.

Usage

execGRASS(cmd, flags = NULL, ..., parameters = NULL, intern = NULL, ignore.stderr = NULL, Sys_ignore.stdout=FALSE, Sys_wait=TRUE, Sys_input=NULL, Sys_show.output.on.console=TRUE, Sys_minimized=FALSE, Sys_invisible=TRUE, echoCmd=NULL, redirect=FALSE, legacyExec=NULL)

stringexecGRASS(string, intern = NULL, ignore.stderr = NULL, Sys_ignore.stdout=FALSE, Sys_wait=TRUE, Sys_input=NULL, Sys_show.output.on.console=TRUE, Sys_minimized=FALSE, Sys_invisible=TRUE, echoCmd=NULL, redirect=FALSE, legacyExec=NULL)
doGRASS(cmd, flags = NULL, ..., parameters = NULL, echoCmd=NULL, legacyExec=NULL)

parseGRASS(cmd, legacyExec=NULL)

## S3 method for class 'GRASS_interface_desc'
print(x, ...)

getXMLencoding()

setXMLencoding(enc)

Arguments

- **cmd**
  - GRASS command name

- **flags**
  - character vector of GRASS command flags

- **...**
  - for execGRASS and doGRASS, GRASS module parameters given as R named arguments directly. For the print method, other arguments to print method.

- **parameters**
  - list of GRASS command parameters, used if GRASS parameters are not given as R arguments directly; the two methods for passing GRASS parameters may not be mixed. The storage modes of values passed must match those required in GRASS, so a single GRASS string must be a character vector of length 1, a single GRASS integer must be an integer vector of length 1 (may be an integer constant such as 10L), and a single GRASS float must be a numeric vector of length 1. For multiple values, use vectors of suitable length

- **string**
  - a string representing one full GRASS statement, using shell syntax: command name, optionally followed by flags and parameters, all separated by whitespaces. Parameters follow the key=value format; if 'value' contains spaces, then 'value' must be quoted

- **intern**
  - default NULL, in which case set internally from get.useInternOption; 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
ignore.stderr default NULL, taking the value set by set.ignore.stderrOption, a logical indicating whether error messages written to 'stderr' should be ignored

Sys.ignore.stdout, Sys.wait, Sys.input
pass extra arguments to system

Sys.show.output.on.console, Sys.minimized, Sys.invisible
pass extra arguments to system on Windows systems only

echoCmd default NULL, taking the logical value set by set.echoCmdOption, print GRASS command to be executed to console

redirect default FALSE, if TRUE, add “2>&1” to the command string and set intern to TRUE; only used in legacy mode

legacyExec default NULL, taking the logical value set by set.legacyExecOption which is initialised to FALSE on “unix” platforms and TRUE otherwise. If TRUE, use system, if FALSE use system2 and divert stderr to temporary file to record error messages and warnings from GRASS modules

x object to be printed

enc character string to replace UTF-8 in header of XML data generated by GRASS module –interface-description 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 (from 0.7-4, the ... argument is not used for passing extra arguments for 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. Support for defaults and for direct use of GRASS parameters instead of a parameter list suggested by Rainer Krug.

stringexecGRASS is a wrapper around execGRASS, and accepts a single shell statement as a string (following GRASS’s command syntax).

Value
parseGRASS returns a GRASS_interface_desc object, doGRASS returns a character string with a proposed GRASS command - the expanded command name is returned as an attribute, and execGRASS and stringexecGRASS return what system or system2 return, particularly depending on the intern argument when the character strings output by GRASS modules are returned. If intern is FALSE, system returns the module exit code, while system2 returns the module exit code with “resOut” and “resErr” attributes.

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  
```r
run <- FALSE
if (nchar(Sys.getenv("GISRC")) > 0 &&
    read.dcf(Sys.getenv("GISRC"))[1,"LOCATION_NAME"] == "nc_basic_spm_grass7") run <- TRUE
  oechoCmd <- get.echoCmdOption()
  set.echoCmdOption(TRUE)
if (run) {
  print(parseGRASS("r.slope.aspect"))
}
if (run) {
  doGRASS("r.slope.aspect", flags=c("overwrite"),
    elevation="elevation.dem", slope="slope", aspect="aspect")
}
if (run) {
  pars <- list(elevation="elevation", slope="slope", aspect="aspect")
  doGRASS("r.slope.aspect", flags=c("overwrite"), parameters=pars)
}
if (run) {
  print(parseGRASS("r.buffer"))
}
if (run) {
  doGRASS("r.buffer", flags=c("overwrite"), input="schools", output="bmap",
    distances=seq(1000,15000,1000))
}
if (run) {
  pars <- list(input="schools", output="bmap", distances=seq(1000,15000,1000))
  doGRASS("r.buffer", flags=c("overwrite"), parameters=pars)
}
if (run) {
  set.echoCmdOption(oechoCmd)
  try(res <- execGRASS("r.stats", input = "fire_blocksgg", # no such file
    flags = c("C", "n"), silent=FALSE))
}
if (run) {
  res <- execGRASS("r.stats", input = "fire_blocksgg", flags = c("C", "n"),
    legacyExec=TRUE)
  print(res)
}
if (run) {
  if (res != 0) {
    resERR <- execGRASS("r.stats", input = "fire_blocksgg",
      flags = c("C", "n"), redirect=TRUE, legacyExec=TRUE)
    print(resERR)
  }
```
```r
if (run) {
  res <- stringexecGRASS("r.stats -p -l input=geology", intern=TRUE)
  print(res)
}
if (run) {
  stringexecGRASS(paste("r.random.cells --overwrite --quiet output=samples",
    "distance=1000 ncells=100 seed=1"))
}
if (run) {
  execGRASS("r.random.cells", flags=c("overwrite", "quiet"), output="samples", distance=1000,
    ncells=100L, seed=1L)
}
```

---

**gmeta**  
*Reads GRASS metadata from the current LOCATION*

**Description**

GRASS LOCATION metadata are read into a list in R; helper function `getLocationProj` returns a WKT2 string of projection information. The helper function `gmeta2grd` creates a `GridTopology` object from the current GRASS mapset region definitions.

**Usage**

```r
gmeta(ignore.stderr = FALSE, g.proj_WKT=NULL)
getLocationProj(ignore.stderr = FALSE, g.proj_WKT=NULL)
gmeta2grd(ignore.stderr = FALSE)
## S3 method for class 'gmeta'
print(x, ...)
get.ignore.stderrOption()
get.stop_on_no_flags_parasOption()
get.echoCmdOption()
get.useInternOption()
get.legacyExecOption()
get.defaultFlagsOption()
get.suppressEchoCmdInFuncOption()
set.ignore.stderrOption(value)
set.stop_on_no_flags_parasOption(value)
set.echoCmdOption(value)
set.useInternOption(value)
set.legacyExecOption(value)
set.defaultFlagsOption(value)
set.suppressEchoCmdInFuncOption(value)
```

**Arguments**

- `ignore.stderr`  
  default FALSE, can be set to TRUE to silence system() output to standard error; does not apply on Windows platforms
g.proj_WKT default NULL: return WKT2 representation in GRASS >= 7.6 and Proj4 in GRASS < 7.6; may be set to FALSE to return Proj4 for GRASS >= 7.6

x S3 object returned by gmeta

... arguments passed through print method

value logical value for setting options on ignore.stderr set by default on package load to FALSE, stop_on_no_flags_paras set by default on package load to TRUE, echoCmd set by default on package load to FALSE. useIntern sets the intern argument globally; legacyExec sets the legacyExec option globally, but is initialized to FALSE on unix systems (all but Windows) and TRUE on Windows; defaultFlags is initialized to NULL, but may be a character vector with values from c("quiet", "verbose") suppressEchoCmdInFunc default TRUE suppresses the effect of echoCmd within package functions, may be set FALSE for debugging.

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

run <- FALSE
if (nchar(Sys.getenv("GISRC")) > 0 &
   read.dcf(Sys.getenv("GISRC"))[1,"LOCATION_NAME"] == "nc_basic_spm_grass7") run <- TRUE
if (run) {
  G <- gmeta()
  print(G)
}
if (run) {
  cat(getLocationProj(), "\n")
}
if (run) {
  cat(getLocationProj(g.proj_WKT=FALSE), "\n")
}
if (run) {
  grd <- gmeta2grd()
  print(grd)
}
if (run) {
  ncells <- prod(slot(grd, "cells.dim"))
  df <- data.frame(k=rep(1, ncells))
  mask_SG <- sp::SpatialGridDataFrame(grd, data=df)
  print(summary(mask_SG))
}
Description

Run GRASS interface in an R session not started within GRASS. In general, most users will use `initGRASS` in throwaway locations, to use GRASS modules on R objects without the need to define and populate a location. The function initializes environment variables used by GRASS, the `.gisrc` used by GRASS for further environment variables, and a temporary location.

On Windows, if OSGeo4W GRASS is being used, the R session must be started in the OSGeo4W shell. If not, the non-standard placing of files and of environment variables confuses the function. If `toupper(gisBase)` contains “OSGEO4W64/APPS/GRASS” or “OSGEO4W/APPS/GRASS” (and after converting “\” to “/”), but the environment variable OSGEO4W_ROOT is not defined, `initGRASS()` will exit with an error before confusion leads to further errors. For further details, see https://github.com/rsbivand/rgrass/issues/16 and https://lists.osgeo.org/pipermail/grass-stats/2018-November/001800.html.

The locking functions are used internally, but are exposed for experienced R/GRASS scripters needing to use the GRASS module "g.mapset" through `initGRASS` in an existing GRASS location. In particular, “g.mapset” may leave a `.gislock` file in the current MAPSET, so it may be important to call `unlink_.gislock` to clean up before quitting the R session. `remove_GISRC` may be used to try to remove the file given in the “GISRC” environment variable if created by `initGRASS` with argument `remove_GISRC=TRUE`.

Usage

```r
initGRASS(gisBase=NULL, home, SG, gisDbase, addon_base, location, mapset, 
override = FALSE, use_g.dirseps.exe = TRUE, pid, remove_GISRC=FALSE, 
ignore.stderr=get.ignore.stderrOption())
get.GIS_LOCK()
set.GIS_LOCK(pid)
unset.GIS_LOCK()
unlink_.gislock()
remove_GISRC()
```

Arguments

- `gisBase`: The directory path to GRASS binaries and libraries, containing bin and lib subdirectories among others; if NULL, set from environment variable `GRASS_INSTALLATION` if found, if not found, `system("grass --config path")` is tried
- `home`: The directory in which to create the `.gisrc` file; defaults to `$HOME` on Unix systems and to `USERPROFILE` on Windows systems; can usually be set to `tempdir()`
- `SG`: An optional `SpatRaster` or `SpatialGrid` object to define the `DEFAULT_WIND` of the temporary location.
- `gisDbase`: if missing, `tempdir()` will be used; GRASS `GISDBASE` directory for the working session
addon_base
if missing, assumed to be “$HOME/grass7/addons” on Unix-like platforms, on MS Windows “%APPDATA%\GRASS7\addons”, and checked for existence

location
if missing, basename(tempfile()) will be used; GRASS location directory for the working session

mapset
if missing, basename(tempfile()) will be used; GRASS mapset directory for the working session

override
default FALSE, set to TRUE if accidental trashing of GRASS .gisrc files and locations is not a problem

use_g.dirseps.exe
default TRUE; when TRUE appears to work for WinGRASS Native binaries, when FALSE for QGIS GRASS binaries; ignored on other platforms.

pid
default as.integer(round(runif(1, 1, 1000))), integer used to identify GIS_LOCK; the value here is arbitrary, but probably should be set correctly

remove_GISRC
default FALSE; if TRUE, attempt to unlink the temporary file named in the “GISRC” environment variable when the R session terminates or when this package is unloaded

ignore.stderr
default taking the value set by set.ignore.stderrOption; can be set to TRUE to silence system() output to standard error; does not apply on Windows platforms

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 gmeta 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.

Author(s)
Roger S. Bivand, e-mail: <Roger.Bivand@nhh.no>

See Also
gmeta
Examples

```r
GRASS_INSTALLATION <- Sys.getenv("GRASS_INSTALLATION")
run <- FALSE
if (nzchar(GRASS_INSTALLATION)) run <- file.info(GRASS_INSTALLATION)$isdir
run <- run && require(terra, quietly=TRUE)
if (run) {
  f <- system.file("ex/elev.tif", package="terra")
  r <- rast(f)
  plot(r, col=grDevices::terrain.colors(50))
}
if (run) {
  (loc <- initGRASS(GRASS_INSTALLATION, home=tempdir(), SG=r, override=TRUE))
}
if (run) {
  write_RAST(r, "elev", flags="overwrite")
  execGRASS("r.info", map="elev")
}
if (run) {
  s <- rast(r)
  values(s) <- values(r)
  write_RAST(s, "elev1", flags="overwrite")
  execGRASS("r.info", map="elev1")
}
if (run) {
  execGRASS("r.slope.aspect", flags="overwrite", elevation="elev", slope="slope", aspect="aspect")
}
if (run) {
  u1 <- read_RAST(c("elev", "slope", "aspect"), return_format="terra")
  plot(u1[,"elev"], col=grDevices::terrain.colors(50))
}
```

---

**readRAST**

Read and write GRASS raster files

**Description**

Read GRASS raster files from GRASS into R **terra** "SpatRaster" or **sp** "SpatialGridDataFrame" objects, and write single columns of **terra** "SpatRaster" or **sp** "SpatialGridDataFrame" objects to GRASS. When `return_format="terra"`, temporary binary files and `r.out.bin` and `r.in.bin` are used for speed reasons. `read_RAST()` and `write_RAST()` by default use "RRASTER" files written and read by GDAL.

**Usage**

```r
read_RAST(vname, cat=NULL, NODATA=NULL, ignore.stderr=get.ignore.stderrOption(),
          return_format="terra", close_OK=return_format="SGDF", flags=NULL)
write_RAST(x, vname, zcol = 1, NODATA=NULL, flags=NULL,
            ignore.stderr = get.ignore.stderrOption(), overwrite=FALSE, verbose=TRUE)
```
Arguments

**vname**  
A vector of GRASS raster file names

**cat**  
default NULL; if not NULL, must be a logical vector matching vname, stating which (CELL) rasters to return as factor

**return_format**  
default "terra", optionally "SGDF"

**ignore.stderr**  
default taking the value set by set.ignore.stderrOption; can be set to TRUE to silence system() output to standard error; does not apply on Windows platforms

**close_OK**  
default TRUE - clean up possible open connections used for reading metadata; may be set to FALSE to avoid the side-effect of other user-opened connections being broken

**x**  
A terra "SpatRaster" or sp "SpatialGridDataFrame" object

**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 for FCELL rasters or the range maximum for CELL Byte, UInt16 and Uint32 rasters (with no negative values), and an attempt is made to set NODATA to the upper Int16 and Int32 range if the lower range is occupied; 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"

**verbose**  
default TRUE, report how writing to GRASS is specified

Value

read_RAST by default returns a SpatRaster object, but may return a legacy SpatialGridDataFrame object if return_format="SGDF". write_RAST silently returns the object being written to GRASS.

Author(s)

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

Examples

```r
run <- FALSE
if (nchar(Sys.getenv("GISRC")) > 0 &
   read.dcf(Sys.getenv("GISRC"))[1, "LOCATION_NAME"] == "nc_basic_spm_grass7") run <- TRUE

GV <- Sys.getenv("GRASS_VERBOSE")
Sys.setenv("GRASS_VERBOSE"="")
ois <- get.ignore.stderrOption()
set.ignore.stderrOption(TRUE)
run <- run && require("terra", quietly=TRUE)
if (run) {
  v1 <- read_RAST("landuse", cat=TRUE, return_format="terra")
  inMemory(v1)
}
```
if (run) {
    write_RAST(v1, "landuse1", flags=c("o", "overwrite"))
    execGRASS("r.stats", flags="c", input="landuse1")
    execGRASS("g.remove", flags="f", name="landuse1", type="raster")
}
run <- run && require("sp", quietly=TRUE)
if (run) {
    nc_basic <- read_RAST(c("geology", "elevation"), cat=c(TRUE, FALSE),
                          return_format="SGDF")
    print(table(nc_basic$geology))
} else if (run) {
    execGRASS("r.stats", flags=c("c", "l", "quiet"), input="geology")
} else if (run) {
    boxplot(nc_basic$elevation ~ nc_basic$geology)
} else if (run) {
    nc_basic$sqdem <- sqrt(nc_basic$elevation)
} else if (run) {
    write_RAST(nc_basic, "sqdemSP", zcol="sqdem", flags=c("quiet", "overwrite"))
    execGRASS("r.info", map="sqdemSP")
} else if (run) {
    print(system.time(sqdemSP <- read_RAST(c("sqdemSP", "elevation"),
                                           return_format="SGDF")))
} else if (run) {
    print(system.time(sqdem <- read_RAST(c("sqdemSP", "elevation"),
                                          return_format="terra")))
} else if (run) {
    execGRASS("g.remove", flags="f", name="sqdemSP", type="raster")
    execGRASS("r.mapcalc", expression="basins0 = basins - 1", flags="overwrite")
    execGRASS("r.stats", flags="c", input="basins0")
} else if (run) {
    basins0 <- read_RAST("basins0", return_format="SGDF")
    print(table(basins0$basins0))
} else if (run) {
    execGRASS("g.remove", flags="f", name="basins0", type="raster")
} Sys.setenv("GRASS_VERBOSE"=GV)
set.ignore.stderrOption(ois)
Description

read_VECT moves one GRASS vector object file with attribute data through a temporary GeoPackage file to a terra "SpatVector" object; write_VECT moves a terra "SpatVector" object through a temporary GeoPackage file 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

read_VECT(vname, layer, type=NULL, flags="overwrite", ignore.stderr = NULL)
write_VECT(x, vname, flags="overwrite", ignore.stderr = NULL)
vInfo(vname, layer, ignore.stderr = NULL)
vColumns(vname, layer, ignore.stderr = NULL)
vDataCount(vname, layer, ignore.stderr = NULL)
vect2neigh(vname, ID=NULL, ignore.stderr = NULL, remove=TRUE, vname2=NULL, units="k")

Arguments

vname A GRASS vector file name
layer a layer name (string); if missing set to default of “1”
type override type detection when multiple types are non-zero, passed to v.out.ogr
ignore.stderr default the value set by set.ignore.stderrOption; NULL, taking the value set by set.ignore.stderrOption, can be set to TRUE to silence system() output to standard error; does not apply on Windows platforms
x A "SpatVector" object moved to GRASS
flags Character vector containing additional optional flags and/or options for v.in.ogr, particularly "o" and "overwrite"
ID A valid DB column name for unique identifiers (optional)
remove default TRUE, remove copied vectors created in vect2neigh
vname2 If on a previous run, remove was FALSE, the name of the temporary vector may be given to circumvent its generation
units default "k"; see GRASS 'v.to.db' manual page for alternatives

Value

read_VECT imports a GRASS vector object into a "SpatVector" object.

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.
Examples

```r
run <- FALSE
if (nchar(Sys.getenv("GISRC")) > 0 &&
    read.dcf(Sys.getenv("GISRC"))[1,"LOCATION_NAME"] == "nc_basic_spm_grass7") run <- TRUE
GV <- Sys.getenv("GRASS_VERBOSE")
Sys.setenv("GRASS_VERBOSE"="0")
ois <- get.ignore.stderrOption()
set.ignore.stderrOption(TRUE)
if (run) {
    execGRASS("v.info", map="schools", layer="1")
}
if (run) {
    print(vInfo("schools"))
    schs <- read_VECT("schools")
    print(summary(schs))
}
if (run) {
    write_VECT(schs, "newsch", flags=c("o", "overwrite"))
    execGRASS("v.info", map="newsch", layer="1")
}
if (run) {
    nschs <- read_VECT("newsch")
    print(summary(nschs))
}
if (run) {
    print(all.equal(names(nschs), as.character(vColumns("newsch")[,2])))
}
if (run) {
    print(vInfo("roadsmajor"))
}
if (run) {
    roads <- read_VECT("roadsmajor")
    print(summary(roads))
}
if (run) {
    cen_neig <- vect2neigh("census")
    str(cen_neig)
}
if (run) {
    execGRASS("g.remove", flags="f", name=c("newsch", "newsch1"), type="vector")
}
Sys.setenv("GRASS_VERBOSE"=GV)
set.ignore.stderrOption(ois)
```
Index

* package
  rgrass-package, 2

  * spatial
    execGRASS, 3
    gmeta, 7
    initGRASS, 9
    readRAST, 11
    readVECT, 13
    rgrass-package, 2

doGRASS (execGRASS), 3

execGRASS, 3

get.defaultFlagsOption (gmeta), 7
get.echoCmdOption (gmeta), 7
get.GIS_LOCK (initGRASS), 9
get.ignore.stderrOption (gmeta), 7
get.legacyExecOption (gmeta), 7
get.stop_on_no_flags_parasOption (gmeta), 7
get.suppressEchoCmdInFuncOption (gmeta), 7
get.useInternOption (gmeta), 7
getXMLencoding (execGRASS), 3
set.defaultFlagsOption (gmeta), 7
set.echoCmdOption (gmeta), 7
set.GIS_LOCK (initGRASS), 9
set.ignore.stderrOption (gmeta), 7
set.legacyExecOption (gmeta), 7
set.stop_on_no_flags_parasOption (gmeta), 7
set.suppressEchoCmdInFuncOption (gmeta), 7
set.useInternOption (gmeta), 7
setXMLencoding (execGRASS), 3
stringexecGRASS (execGRASS), 3
system, 6

unlink_.gislock (initGRASS), 9
unset.GIS_LOCK (initGRASS), 9

vColumns (readVECT), 13
vDataCount (readVECT), 13
vect2neigh (readVECT), 13
vInfo (readVECT), 13

write_RAST (readRAST), 11
write_VECT (readVECT), 13

rgrass (rgrass-package), 2
rgrass-package, 2

read_RAST (readRAST), 11
read_VECT (readVECT), 13
readRAST, 11
readVECT, 13
remove_GISRC (initGRASS), 9