Package 'Rsagacmd'

June 12, 2020

Type Package

Title Linking R with the Open-Source 'SAGA-GIS' Software

Version 0.0.9

Date 2020-06-11

Maintainer Steven Pawley <dr.stevenpawley@gmail.com>

Description Provides an R scripting interface to the open-source 'SAGA-GIS' (System for Automated Geoscientific Analyses Geographical Information System) software. 'Rsagacmd' dynamically generates R functions for every 'SAGA-GIS' geoprocessing tool based on the user's currently installed 'SAGA-GIS' version. These functions are contained within an S3 object and are accessed as a named list of libraries and tools. This structure facilitates an easier scripting experience by organizing the large number of 'SAGA-GIS' geoprocessing tools (>700) by their respective library. Interactive scripting can fully take advantage of code autocompletion tools (e.g. in 'Rstudio'), allowing for each tools syntax to be quickly recognized. Furthermore, the most common types of spatial data (via the 'raster', 'sp', and 'sf' packages) along with non-spatial data are automatically passed from R to the 'SAGA-GIS' command line tool for geoprocessing operations, and the results are loaded as the appropriate R object. Outputs from individual 'SAGA-GIS' tools can also be chained using pipes from the 'magrittr' and 'dplyr' packages to combine complex geoprocessing operations together in a single statement. 'SAGA-GIS' is available under a GPLv2 / LGPLv2 licence from <https://sourceforge.net/projects/saga-gis/> including Windows x86/x84 binaries. SAGA-GIS is also included in Debian/Ubuntu default software repositories and is available for macOS using homebrew (<https://brew.sh/>) from the osgeo/osgeo4mac (<https://github.com/OSGeo/homebrew-osgeo4mac>) formula tap. Rsagacmd has currently been tested on 'SAGA-GIS' versions from 2.3.1 to 7.6.0 on Windows, Linux and macOS.

License GPL-3

Encoding UTF-8

SystemRequirements SAGA-GIS (>= 2.3.1)

LazyData true

RoxygenNote 7.1.0

Depends raster, R (>= 2.10)

Imports XML, sf, tools, rgdal, foreign, minpack.lm, magrittr, stringr, rlang, tibble

Suggests dplyr, testthat, covr

NeedsCompilation no

Author Steven Pawley [aut, cre]

Repository CRAN

Date/Publication 2020-06-12 04:20:06 UTC

R topics documented:

`mrvbf_threshold` *Calculate the t_slope value based on DEM resolution for MRVBF*

Description

Calculates the t_slope value for the Multiresolution Index of Valley Bottom Flatness (Gallant and Dowling, 2003) based on input DEM resolution. MRVBF identified valley bottoms based on classifying slope angle and identifying low areas by ranking elevation in respect to the surrounding topography across a range of DEM resolutions. The MRVBF algorithm was developed using a 25 m DEM, and so if the input DEM has a different resolution then the slope threshold t_slope needs to be adjusted from its default value of 16 in order to maintain the relationship between slope and DEM resolution. This function provides a convenient way to perform that calculation.

Usage

```
mrvbf_threshold(res, plot = FALSE)
```

Arguments

`res`	numeric, DEM resolution
`plot`	logical, produce plot of relationship

Value

numeric, t_slope value for MRVBF

Examples

```
mrvbf_threshold(res = 10, plot = TRUE)
```

`print.saga_tool` *Generic function to display help and usage information for any SAGA-GIS tool*

Description

Generic function to display help and usage information for any SAGA-GIS tool

Usage

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

Arguments

`x`	A ‘saga_tool‘ object.
`...`	Additional arguments to pass to print. Currently not used.

Examples

```
## Not run:
# Intialize a saga object
saga <- saga_gis()

# Display usage information on a tool
print(saga$ta_morphometry$slope_aspect_curvature)

# Or simply:
saga$ta_morphometry$slope_aspect_curvature

## End(Not run)
```

Rsagacmd *Rsagacmd: Linking R with the open-source SAGA-GIS software.*

Description

Rsagacmd provides an R scripting interface to the open-source System for Automated Geoscientific Analyses Geographical Information System software SAGA-GIS. The current version has been tested using SAGA-GIS versions 2.3.1 to 7.6.0 on Windows (x64), OS X and Linux. Rsagacmd provides a functional approach to scripting with SAGA-GIS by dynamically generating R functions for every SAGA-GIS tool based on the user's current SAGA-GIS installation. These functions are generated by the `saga_gis` function and are included within an S3 object as a named list of libraries and tools. This structure facilitates an easier scripting experience by organizing the large number of SAGA-GIS tools (>700) by their respective library. Interactive scripting can also fully take advantage of code autocompletion tools (e.g. in Rstudio), allowing for each tool's syntax to be quickly recognized. Furthermore, the most common types of spatial data (rasters using the **raster** package, and vector data using the **sp** or simple features **sf** packages) along with non-spatial data are seamlessly passed from R to the SAGA-GIS command line tool for geoprocessing operations, and the results are automatically loaded as the appropriate R object. Outputs from individual SAGA-GIS tools can also be chained using pipes from the **magrittr** and **dplyr** packages to chain complex geoprocessing operations together in a single statement.

Handling of geospatial and tabular data

Rsagacmd aims to facilitate a seamless interface to the open-source SAGA-GIS by providing access to all SAGA-GIS geoprocessing tools in a 'R-like' manner. In addition to generating R functions that correspond to each SAGA-GIS tool, Rsagacmd automatically handles the passing of geospatial and tabular data contained from the R environment to SAGA-GIS.

Rsagacmd uses the SAGA-GIS command line interface to perform geoprocessing operations. Therefore, spatial data can be passed to any Rsagacmd function as a path to the input data, assuming that the data is stored in the appropriate file formats (e.g. GDAL-supported single-band rasters, OGR supported vector data, and comma- or tab-delimited text files for tabular data). In addition, Rsagacmd also supports the following R object classes to pass data to SAGA-GIS, and to load the results back into the R environment:

- Raster data handling is provided by the R **raster** package Raster-based outputs from SAGA-GIS tools are loaded as RasterLayer objects. For more details, see the 'Handling of raster data'.
- Vector features that result from SAGA-GIS geoprocessing operations are output in ESRI Shapefile format and are loaded into the R environment as simple features objects
- Tabular data from SAGA-GIS tools are loaded as data frames

The results from tools that return multiple outputs are loaded into the R environment as a named list of the appropriate R object classes.

Multi-band raster data and RasterStack/RasterBrick objects

SAGA-GIS does not handle multi-band rasters and the native SAGA GIS Binary file format (.sgrd) supports only single band data. Therefore when passing raster data to most SAGA-GIS tools using Rsagacmd, the data should represent single raster bands, specified as either the path to the single raster band, or when using the R **raster** package, a RasterLayer (or less commonly a RasterStack or RasterBrick) object that contains only a single layer. Subsetting of raster data is performed automatically by Rsagacmd in the case of when a single band from a RasterStack or RasterBrick object is passed to a SAGA-GIS tool. This occurs in by either passing the filename of the raster to the SAGA-GIS command line, or by writing the data to a temporary file. A few SAGA-GIS functions will accept a list of single band rasters as an input. In this case if this data is in the form of a RasterStack or RasterLayer object, it is recommended to use pass the output from the unstack function in the **raster** package, which will return a list of RasterLayer objects, and then Rsagacmd will handle the subsetting automatically.

Combining SAGA-GIS tools with pipes

For convenience, outputs from SAGA-GIS tools are automatically saved to tempfiles if outputs are not explicitly stated as arguments when calling the function. This was implemented so that the user can create complex workflows based on little code. It is also means that several processing steps can be combined or chained in a convenient manner using pipes from the **magritrr** package. When using pipes, all of the intermediate processing steps are dealt with automatically by saving the outputs as tempfiles, and then in turn passing the output to the next function in the chain. Note that when dealing with high-resolution and/or larger raster data, these tempfiles can start to consume a significant amount of disk space during a session. If required, these temporary files can be cleaned during the session in a similar way to the raster package, using the saga_remove_tmpfiles function.

Notes

SAGA-GIS compressed .sg-grd-z files are not currently supported, although support may be added in future package updates.

Author(s)

Steven Pawley, <dr.stevenpawley@gmail.com>

Examples

```
## Not run:
library(Rsagacmd)
library(magrittr)

# initialize a saga object
saga <- saga_gis(opt_lib = c("grid_calculus", "ta_morphometry"))

# example of executing a tool using a tempfile to store the tool outputs
dem <- saga$grid_calculus$random_terrain(radius = 100, iterations = 500)

# Example of chaining operations using pipes and using tempfile to
# store tool outputs
tri <- dem %>%
```

```
  saga$ta_morphometry$terrain_ruggedness_index_tri()

# Remove tempfiles generated by Rsagacmd during a session
saga_remove_tmpfiles(h = 0)

## End(Not run)
```

---

saga_configure *Generates a custom saga_cmd configuration file*

Description

Creates and edits a saga_cmd configuration file in order to change saga_cmd settings related to file caching and number of available processor cores. Intended to be used internally by saga_gis

Usage

```
saga_configure(
  senv,
  grid_caching = FALSE,
  grid_cache_threshold = 100,
  grid_cache_dir = NULL,
  cores = NULL,
  saga_vers
)
```

Arguments

senv	A saga environment object. Contains the SAGA-GIS environment and settings.
grid_caching	Whether to use file caching. The default is FALSE.
grid_cache_threshold	Any number to use as a threshold (in Mb) before file caching for loaded raster data is activated.
grid_cache_dir	Optionally specify a path to the used directory for temporary files. The default uses 'base::tempdir'.
cores	An integer specifying the maximum number of processing cores. Needs to be set to 1 if file caching is activated because file caching in SAGA-GIS is not thread-safe.
saga_vers	A 'numeric_version' that specifies the version of SAGA-GIS. The generation of a saga_cmd configuration file is only valid for versions > 4.0.0.

Value

A character that specifies the path to custom saga_cmd initiation file.

saga_env — *Parses valid SAGA-GIS libraries and tools into a nested list of functions*

Description

Establishes the link to SAGA GIS by generating a SAGA help file and parsing all libraries, tools and options from the help files into a nested list of library, module and options, that are contained within an saga environment object object. Intended to be used internally by saga_gis

Usage

```
saga_env(saga_bin = NULL, opt_lib = NULL)
```

Arguments

saga_bin	An optional character vector to specify the path to the saga_cmd executable. Otherwise the function will perform a search for saga_cmd.
opt_lib	A character vector of a subset of SAGA-GIS tool libraries to generate dynamic functions that map to each tool. Used to save time if you only want to import a single library.

Value

A saga environment S3 object containing paths, settings and a nested list of libraries tools and options.

saga_execute — *Function to execute SAGA-GIS commands through the command line tool*

Description

Intended to be used internally by each function

Usage

```
saga_execute(lib, tool, senv, .intern = TRUE, .all_outputs = TRUE, ...)
```

Arguments

lib	A character specifying the name of SAGA-GIS library to execute.
tool	A character specifying the name of SAGA-GIS tool to execute.
senv	A saga environment object.
.intern	A logical specifying whether to load the outputs from the SAGA-GIS geoprocessing operation as an R object.
.all_outputs	A logical to specify whether to automatically output all results from the selected SAGA tool and load them results as R objects (default = TRUE). If .all_outputs = FALSE then the file paths to store the tool's results will have to be manually specified in the arguments.
...	Named arguments and values for SAGA tool.

Value

output of SAGA-GIS tool loaded as an R object.

---

saga_gis *Initiate a SAGA-GIS geoprocessor object*

Description

Dynamically generates functions to all valid SAGA-GIS libraries and tools. These functions are stored within a saga S3 object as a named list of functions

Usage

```
saga_gis(
  saga_bin = NULL,
  grid_caching = FALSE,
  grid_cache_threshold = 100,
  grid_cache_dir = NULL,
  cores = NULL,
  opt_lib = NULL,
  temp_path = NULL
)
```

Arguments

saga_bin	The path to saga_cmd executable. If this argument is not supplied then an automatic search for the saga_cmd executable will be performed.
grid_caching	A logical whether to use file caching in saga_cmd geoprocessing operations for rasters that are too large to fit into memory.
grid_cache_threshold	A number to act as a threshold (in Mb) before file caching is activated for loaded raster data.

grid_cache_dir	The path to directory for temporary files generated by file caching. If not provided then the result from 'base::tempdir()' is used.
cores	An integer for the maximum number of processing cores. By default all cores are utilized. Needs to be set to 1 if file caching is activated.
opt_lib	A character vector with the names of a subset of SAGA-GIS libraries. Used to link only a subset of named SAGA-GIS tool libraries, rather than creating functions for all available tool libraries.
temp_path	The path to use to store any temporary files that are generated as data is passed between R and SAGA-GIS. If not specified, then the system 'base::tempdir()' is used.

Value

A S3 'saga' object containing a nested list of functions for SAGA-GIS libraries and tools.

Examples

```
## Not run:
# Initialize a saga object
saga <- saga_gis()

# Alternatively intialize a saga object using file caching to handle large
# raster files
saga <- saga_gis(grid_caching = TRUE, grid_cache_threshold = 250, cores = 1)

# Example terrain analysis
# Generate a random DEM
dem <- saga$grid_calculus$random_terrain(radius = 100)

# Use Rsagacmd to calculate the Terrain Ruggedness Index
tri <- saga$ta_morphometry$terrain_ruggedness_index_tri(dem = dem)
plot(tri)

# Optionally run command and do not load result as an R object
saga$ta_morphometry$terrain_ruggedness_index_tri(dem = dem, .intern = FALSE)

## End(Not run)
```

---

saga_remove_tmpfiles *Removes temporary files created by Rsagacmd*

Description

For convenience, functions in the Rsagacmd package create temporary files if any outputs for a SAGA-GIS tool are not specified as arguments. Temporary files in R are automatically removed at the end of each session. However, when dealing with raster data, these temporary files potentially can consume large amounts of disk space. These temporary files can be observed during a session by using the saga_show_tmpfiles function, and can be removed using the saga_remove_tmpfiles function. Note that this function also removes any accompanying files, i.e. the '.prj' and '.shx' files that may be written as part of writing a ESRI Shapefile '.shp' format

Usage

```
saga_remove_tmpfiles(h = 0)
```

Arguments

h	Remove temporary files that are older than h (in number of hours).

Value

Nothing is returned.

Examples

```
## Not run:
# Remove all temporary files generated by Rsagacmd
saga_remove_tmpfiles(h = 0)

## End(Not run)
```

---

saga_search *Automatic search for the path to a SAGA-GIS installation*

Description

Returns the path to the saga_cmd executable On windows, automatic searching will occur first in 'C:/Program Files/SAGA-GIS'; 'C:/Program Files (x86)/SAGA-GIS'; 'C:/SAGA-GIS'; 'C:/OSGeo4W'; and 'C:/OSGeo4W64'. On linux or OS X, saga_cmd is usually included in PATH, if not an automatic search is performed in the '/usr' folder. If multiple versions of SAGA-GIS are installed on the system, the path to the newest version is returned

Usage

```
saga_search()
```

Value

The path to installed saga_cmd binary.

saga_show_tmpfiles *List temporary files created by Rsagacmd*

Description

For convenience, functions in the Rsagacmd package create temporary files if any outputs for a SAGA-GIS tool are not specified as arguments. Temporary files in R are automatically removed at the end of each session. However, when dealing with raster data, these temporary files potentially can consume large amounts of disk space. These temporary files can be observed during a session by using the saga_show_tmpfiles function, and can be removed using the saga_remove_tmpfiles function.

Usage

```
saga_show_tmpfiles()
```

Value

returns the file names of the files in the temp directory that have been generated by Rsagacmd. Note this list of files only includes the primary file extension, i.e. '.shp' for a shapefile without the accessory files (e.g. .prj, .shx etc.).

Examples

```
## Not run:
# Show all temporary files generated by Rsagacmd
saga_remove_tmpfiles(h = 0)

## End(Not run)
```

saga_version *Return the installed version of SAGA-GIS*

Description

Intended to be used internally by saga_env. Uses a system call to saga_cmd to output version of installed SAGA-GIS on the console

Usage

```
saga_version(saga_cmd)
```

Arguments

saga_cmd	The path of the saga_cmd binary.

Value

A numeric_version with the version of SAGA-GIS found at the cmd path.

`save_object` *Generic methods to save R in-memory objects to file to SAGA-GIS to access*

Description

Designed to be used internally by Rsagacmd for automatically pass data to SAGA-GIS for geoprocessing

Usage

```
save_object(x, ...)
```

Arguments

`x`	An R object.
`...`	Other parameters such as the temporary directory to use for storage.

Value

A character that specifies the file path to where the R object was saved.

`search_tools` *Search for a SAGA-GIS tool*

Description

Search for a SAGA-GIS tool

Usage

```
search_tools(x, pattern)
```

Arguments

`x`	saga object
`pattern`	character, pattern of text to search for within the tool name

Value

dataframe, tools that match the pattern of the search text and their host library

Examples

```
## Not run:
# initialize Rsagacmd
saga <- saga_gis()

# search for a tool
search_tools(x = saga, pattern = "terrain")

## End(Not run)
```

`tile_geoprocessor` *Split a raster grid into tiles for tile-based processing*

Description

Split a raster grid into tiles. The tiles are saved as Rsagacmd temporary files, and are loaded as a list of R objects for further processing. This is a function to make the the SAGA-GIS grid_tools / tiling tool more convenient to use.

Usage

```
tile_geoprocessor(x, grid, nx, ny, overlap = 0, file_path = NULL)
```

Arguments

`x`	A 'saga' object.
`grid`	A path to a GDAL-supported raster to apply tiling, or a RasterLayer.
`nx`	An integer with the number of x-pixels per tile.
`ny`	An integer with the number of y-pixels per tile.
`overlap`	An integer with the number of overlapping pixels.
`file_path`	An optional file file path to store raster tiles.

Value

A list of RasterLayer objects representing tiled data.

Examples

```
## Not run:
# Initialize a saga object
saga <- saga_gis()

# Generate a random DEM
dem <- saga$grid_calculus$random_terrain(radius = 15, iterations = 500)

# Return tiled version of DEM
tiles <- tile_geoprocessor(x = saga, grid = dem, nx = 20, ny = 20)

## End(Not run)
```

Index