Package 'RQGIS'

August 13, 2018

Date 2018-08-14

Type Package

Title Integrating R with QGIS

Version 1.0.4

Description Establishes an interface between R and 'QGIS', i.e. it allows the user to access 'QGIS' functionalities from the R console. It achieves this by using the 'QGIS' Python API via the command line. Hence, RQGIS extends R's statistical power by the incredible vast geo-functionality of 'QGIS' (including also 'GDAL', 'SAGA'- and 'GRASS'-GIS among other third-party providers). This in turn creates a powerful environment for advanced and innovative (geo-)statistical geocomputing. 'QGIS' is licensed under GPL version 2 or greater and is available from <http://www.qgis.org/en/site/>.

URL https://github.com/jannes-m/RQGIS

BugReports https://github.com/jannes-m/RQGIS/issues

License GPL-3

LazyData TRUE

Encoding UTF-8

RoxygenNote 6.1.0

ByteCompile true

Depends R (>= 3.2.0), reticulate (>= 1.2)

Imports raster, RCurl, readr, rgdal, sf (>= 0.4-2), sp, stringr, tools, XML

SystemRequirements Python (>= 2.7), QGIS (>= 2.14 & < 3)

Suggests knitr, rgrass7, rmarkdown, testthat, RSAGA

VignetteBuilder knitr

NeedsCompilation no

Author Jannes Muenchow [aut, cre] (<https://orcid.org/0000-0001-7834-4717>), Patrick Schratz [aut] (<https://orcid.org/0000-0003-0748-6624>)

Maintainer Jannes Muenchow <jannes.muenchow@uni-jena.de>

Repository CRAN

Date/Publication 2018-08-13 10:30:03 UTC

R topics documented:

`RQGIS-package` *RQGIS: Integrating R with QGIS*

Description

RQGIS establishes an interface between R and 'QGIS', i.e. it allows the user to access 'QGIS' functionalities from the R console. It achieves this by using the QGIS' Python API via the command line. Hence, RQGIS extends R's statistical power by the incredible vast geo-functionality of 'QGIS' (including also 'GDAL', 'SAGA'- and 'GRASS'-GIS among other third-party providers). This in turn creates a powerful environment for advanced and innovative (geo-)statistical geocomputing. 'QGIS' is licensed under GPL version 2 or greater and is available from http://www.qgis.org/en/site/. Before running RQGIS you need to make sure to have installed correctly all external software such as QGIS, GRASS and SAGA.

Details

Our vignette helps to correctly install all third-party dependencies (e.g., QGIS, GRASS, SAGA):

```
vignette("install_guide", package = "RQGIS")
```

To get started with RQGIS, have a peak at the example on our github page:

https://github.com/jannes-m/RQGIS

Author(s)

Maintainer: Jannes Muenchow <jannes.muenchow@uni-jena.de> (0000-0001-7834-4717)

Authors:

- Patrick Schratz <patrick.schratz@uni-jena.de> (0000-0003-0748-6624)

See Also

Useful links:

- https://github.com/jannes-m/RQGIS
- Report bugs at https://github.com/jannes-m/RQGIS/issues

---

comm *Community matrix of the Mt. Mongón*

Description

A community matrix with species as columns and sites as rows. The rownames correspond to the id which can be also found in random_points. Please note that in fact 100 sites have been visited but in 16 of them no species could be found (see again random_points).

Format

An dataframe with 84 sites (rows) and 69 species (columns). Species presence is given in percentage points.

References

Muenchow, J., Bräuning, A., Rodríguez, E.F. & von Wehrden, H. (2013): Predictive mapping of species richness and plant species' distributions of a Peruvian fog oasis along an altitudinal gradient. Biotropica 45, 5, 557-566, doi: 10.1111/btp.12049.

---

dem *Digital elevation model (DEM) of the Mongón study area.*

Description

A `raster::raster()` object (EPSG:32717) representing altitude (ASTER GDEM, LP DAAC 2012). For more details, please refer to Muenchow et al. (2013).

Format

A `raster::raster()` with 117 rows and 117 columns:

dem Altitude in m asl.

References

Muenchow, J., Bräuning, A., Rodríguez, E.F. & von Wehrden, H. (2013): Predictive mapping of species richness and plant species' distributions of a Peruvian fog oasis along an altitudinal gradient. Biotropica 45, 5, 557-566, doi: 10.1111/btp.12049.

LP DAAC (2012): Land Processes Distributed Active Archive Center, located at the U.S. Geological Survey (USGS) Earth Resources Observation and Science (EROS) Center. Available at: https://lpdaac.usgs.gov/ (last accessed 25 January 2012).

`find_algorithms` *Find and list available QGIS algorithms*

Description

`find_algorithms` lists or queries all QGIS algorithms which can be accessed via the QGIS Python API.

Usage

```
find_algorithms(search_term = NULL, name_only = FALSE,
  qgis_env = set_env())
```

Arguments

`search_term`	If (`NULL`), the default, all available functions will be returned. If `search_term` is a character, all available functions will be queried accordingly. The character string might also contain a regular expression (see examples).
`name_only`	If `TRUE`, the function returns only the name(s) of the found algorithms. Otherwise, a short function description will be returned as well (default).
`qgis_env`	Environment settings containing all the paths to run the QGIS API. For more information, refer to `set_env()`.

Details

Function `find_algorithms` simply calls `processing.alglist` using Python.

Value

The function returns QGIS function names and short descriptions as an R character vector.

Author(s)

Jannes Muenchow, Victor Olaya, QGIS core team

Examples

```
## Not run:
# list all available QGIS algorithms on your system
algs <- find_algorithms()
algs[1:15]
# find a function which adds coordinates
find_algorithms(search_term = "add")
# find only QGIS functions
find_algorithms(search_term = "qgis:")
# find QGIS and SAGA functions related to centroid computations
find_algorithms(search_term = "centroid.+(qgis:|saga:)")

## End(Not run)
```

`get_args_man` *Get GIS arguments and respective default values*

Description

`get_args_man` retrieves automatically function arguments and respective default values for a given QGIS geoalgorithm.

Usage

```
get_args_man(alg = "", options = TRUE, qgis_env = set_env())
```

Arguments

`alg`	The name of the algorithm for which one wishes to retrieve arguments and default values.
`options`	Sometimes one can choose between various options for a function argument. Setting option to `TRUE`, the default, will automatically assume one wishes to use the first option (QGIS GUI behavior).
`qgis_env`	Environment containing all the paths to run the QGIS API. For more information, refer to `set_env()`.

Details

`get_args_man` basically mimics the behavior of the QGIS GUI. That means, for a given GIS algorithm, it captures automatically all arguments and default values. In the case that a function argument has several options, one can indicate to use the first option (see also `get_options()`), which is the QGIS GUI default behavior.

Value

The function returns a list whose names correspond to the function arguments one needs to specify. The list elements correspond to the argument specifications. The specified function arguments can serve as input for `run_qgis()`'s params argument. Please note that although `get_args_man` tries to retrieve default values, one still needs to specify some function arguments manually such as the input and the output layer.

Note

Please note that some default values can only be set after the user's input. For instance, the GRASS region extent will be determined automatically by `run_qgis()` if left blank.

Author(s)

Jannes Muenchow, Victor Olaya, QGIS core team

Examples

```
## Not run:
get_args_man(alg = "qgis:addfieldtoattributestable")
# and using the option argument
get_args_man(alg = "qgis:addfieldtoattributestable", options = TRUE)

## End(Not run)
```

---

get_options *Get options of parameters for a specific GIS option*

Description

`get_options` lists all available parameter options for the required GIS function.

Usage

```
get_options(alg = "", intern = FALSE, qgis_env = set_env())
```

Arguments

`alg`	Name of the GIS function for which options should be returned.
`intern`	Logical, if `TRUE` the function captures the command line output as an `R` character vector. If `FALSE`, the default, the output is printed to the console in a pretty way.
`qgis_env`	Environment containing all the paths to run the QGIS API. For more information, refer to `set_env()`.

Details

Function `get_options` simply calls `processing.algoptions` of the QGIS Python API.

Author(s)

Jannes Muenchow, Victor Olaya, QGIS core team

Examples

```
## Not run:
get_options(alg = "saga:slopeaspectcurvature")

## End(Not run)
```

`get_usage` *Get usage of a specific QGIS geoalgorithm*

Description

`get_usage` lists all function parameters of a specific QGIS geoalgorithm.

Usage

```
get_usage(alg = NULL, intern = FALSE, qgis_env = set_env())
```

Arguments

`alg`	Name of the function whose parameters are being searched for.
`intern`	Logical, if `TRUE` the function captures the command line output as an `R` character vector. `IfFALSE`, the default, the output is printed to the console in a pretty way.
`qgis_env`	Environment containing all the paths to run the QGIS API. For more information, refer to `set_env()`.

Details

Function `get_usage` simply calls `processing.alghelp` of the QGIS Python API.

Author(s)

Jannes Muenchow, Victor Olaya, QGIS core team

Examples

```
## Not run:
# find a function which adds coordinates
find_algorithms(search_term = "add")
# find function arguments of saga:addcoordinatestopoints
get_usage(alg = "saga:addcoordinatestopoints")

## End(Not run)
```

ndvi *Normalized difference vegetation index for the Mongón study area.*

Description

NDVI `raster::raster()` (EPSG:32717) computed from a Landsat scene (path 9, row 67, acquisition date: 09/22/2000; USGS 2013). For more details, please refer to Muenchow et al. (2013).

Format

A `raster::raster()` with 117 rows and 117 columns:

ndvi Normalized difference vegetation index.

References

Muenchow, J., Bräuning, A., Rodríguez, E.F. & von Wehrden, H. (2013): Predictive mapping of species richness and plant species' distributions of a Peruvian fog oasis along an altitudinal gradient. Biotropica 45, 5, 557-566, doi: 10.1111/btp.12049.

USGS (2013): U.S. Geological Survey. Earth Explorer. Available at: http://earthexplorer.usgs.gov/ (last accessed 1 March 2013).

open_app *Open a QGIS application*

Description

`open_app` first sets all the correct paths to the QGIS Python binary, and secondly opens a QGIS application while importing the most common Python modules.

Usage

```
open_app(qgis_env = set_env())
```

Arguments

`qgis_env`	Environment settings containing all the paths to run the QGIS API. For more information, refer to `set_env()`. Basically, the function defines a few new environment variables which should not interfere with other settings.

Value

The function enables a 'tunnel' to the Python QGIS API.

Note

Please note that the function changes your environment settings via `base::Sys.getenv()` which is necessary to run the QGIS Python API.

Author(s)

Jannes Muenchow

Examples

```
## Not run:
open_app()

## End(Not run)
```

`open_help` *Access the QGIS/GRASS online help for a specific (Q)GIS geoalgorithm*

Description

`open_help` opens the online help for a specific (Q)GIS geoalgorithm. This is the online help one also encounters in the QGIS GUI. In the case of GRASS algorithms this is actually the GRASS online documentation.

Usage

```
open_help(alg = "", qgis_env = set_env())
```

Arguments

`alg`	The name of the algorithm for which one wishes to retrieve arguments and default values.
`qgis_env`	Environment containing all the paths to run the QGIS API. For more information, refer to `set_env()`.

Details

Bar a few exceptions `open_help` works for all QGIS, GRASS and SAGA geoalgorithms. The online help of other third-party providers, however, has not been tested so far.

Value

The function opens the default web browser, and displays the help for the specified algorithm.

Note

Please note that `open_help` requires a **working Internet connection**.

Author(s)

Jannes Muenchow, Victor Olaya, QGIS core team

Examples

```
## Not run:
# QGIS example
open_help(alg = "qgis:addfieldtoattributestable")
# GRASS example
open_help(alg = "grass:v.overlay")

## End(Not run)
```

---

pass_args *Specifying QGIS geoalgorithm parameters the R way*

Description

The function lets the user specify QGIS geoalgorithm parameters as R named arguments or a a parameter-argument list. When omitting required parameters, defaults will be used if available as derived from `get_args_man()`. Additionally, the function checks thoroughly the user-provided parameters and arguments.

Usage

```
pass_args(alg, ..., params = NULL, NA_flag = -99999,
  qgis_env = set_env())
```

Arguments

`alg`	The name of the geoalgorithm to use.
`...`	Triple dots can be used to specify QGIS geoalgorithm arguments as R named arguments.
`params`	Parameter-argument list for a specific geoalgorithm, see `get_args_man()` for more details. Please note that you can either specify R arguments directly via the triple dots (see above) or via the parameter-argument list. However, you may not mix the two methods.
`NA_flag`	Value used for NAs when exporting raster objects through `save_spatial_objects()` (default: -99999).
`qgis_env`	Environment containing all the paths to run the QGIS API. For more information, refer to `set_env()`.

Details

In detail, the function performs following actions and parameter-argument checks:

- Were the right parameter names used?
- Were the correct argument values provided?
- The function collects all necessary arguments (to run QGIS) and respective default values which were not set by the user with the help of `get_args_man()`.
- If an argument value corresponds to a spatial object residing in R (`sp`-, `sf`- or `raster`-objects are supported), the function will save the spatial object to `tempdir()`, and use the corresponding file path to replace the spatial object in the parameter-argument list. If the QGIS geoalgorithm parameter belongs to the `ParameterMultipleInput`-instance class (see for example `get_usage(grass7:v.patch)`) you may either use a character-string containing the paths to the spatial objects separated by a semi-colon (e.g., "shape1.shp;shape2.shp;shape3.shp" - see also QGIS documentation) or provide a `base::list()` where each spatial object corresponds to one list element.
- If a parameter accepts as arguments values from a selection, the function replaces verbal input by the corresponding number (required by the QGIS Python API). Please refer to the example section for more details, and to `get_options()` for valid options for a given geoalgorithm.
- If `GRASS_REGION_PARAMETER` is "None" (the QGIS default), `run_qgis` will automatically determine the region extent based on the user-specified input layers. If you do want to specify the `GRASS_REGION_PARAMETER` yourself, please do it in accordance with the QGIS documentation, i.e., use a character string and separate the coordinates with a comma: "xmin, xmax, ymin, ymax".

Value

The function returns the complete parameter-argument list for a given QGIS geoalgorithm. The list is constructed with the help of `get_args_man()` while considering the R named arguments or the `params`-parameter specified by the user as additional input. If available, the function returns the default values for all parameters which were not specified.

Note

This function was inspired by `rgrass7::doGRASS()`.

Author(s)

Jannes Muenchow

Examples

```
## Not run:
data(dem, package = "RQGIS")
alg <- "grass7:r.slope.aspect"
get_usage(alg)
# 1. using R named arguments
pass_args(alg, elevation = dem, slope = "slope.asc")
# 2. doing the same with a parameter argument list
```

```
pass_args(alg, params = list(elevation = dem, slope = "slope.asc"))
# 3. verbal input replacement (note that "degrees" will be replaced by 0)
get_options(alg)
pass_args(alg, elevation = dem, format = "degrees")

## End(Not run)
```

`qgis_session_info` *QGIS session info*

Description

`qgis_session_info` reports the version of QGIS and installed third-party providers (so far GRASS 6, GRASS 7, and SAGA). Additionally, it figures out with which SAGA versions the QGIS installation is compatible.

Usage

```
qgis_session_info(qgis_env = set_env())
```

Arguments

`qgis_env`	Environment settings containing all the paths to run the QGIS API. For more information, refer to `set_env()`.

Value

The function returns a list with following elements:

1. qgis_version: Name and version of QGIS used by RQGIS.
2. grass6: GRASS 6 version number, if installed to use with QGIS.
3. grass7: GRASS 7 version number, if installed to use with QGIS.
4. saga: The installed SAGA version used by QGIS.
5. supported_saga_versions: character vector representing the SAGA versions supported by the QGIS installation.

Author(s)

Jannes Muenchow, Victor Olaya, QGIS core team

Examples

```
## Not run:
qgis_session_info()

## End(Not run)
```

random_points *Random points.*

Description

An sf (EPSG:32717) object with 100 randomly sampled points (stratified by altitude). For more details, please refer to Muenchow et al. (2013).

Format

An sf object with 100 rows and 3 variables:

id Plot ID.

spri Number of vascular plant species per plot (species richness).

geometry Simple feature point geometry.

References

Muenchow, J., Bräuning, A., Rodríguez, E.F. & von Wehrden, H. (2013): Predictive mapping of species richness and plant species' distributions of a Peruvian fog oasis along an altitudinal gradient. Biotropica 45, 5, 557-566, doi: 10.1111/btp.12049.

reset_path *Reset PATH*

Description

Since `run_ini()` starts with a clean PATH, this function makes sure to add the original paths to PATH. Note that this function is a Windows-only function.

Usage

```
reset_path(settings)
```

Arguments

`settings`	A list as derived from `as.list(Sys.getenv())`.

Author(s)

Jannes Muenchow

run_qgis *Interface to QGIS commands*

Description

run_qgis calls QGIS algorithms from within R while passing the corresponding function arguments.

Usage

```
run_qgis(alg = NULL, ..., params = NULL, load_output = FALSE,
  show_output_paths = TRUE, NA_flag = -99999, qgis_env = set_env())
```

Arguments

alg	Name of the GIS function to be used (see find_algorithms()).
...	Triple dots can be used to specify QGIS geoalgorithm arguments as R named arguments. For more details, please refer to pass_args().
params	Parameter-argument list for a specific geoalgorithm. Please note that you can either specify R named arguments directly via the triple dots (see above) or via a parameter-argument list. However, you may not mix the two methods. See the example section, pass_args() and get_args_man() for more details.
load_output	If TRUE, all QGIS output files (sf::sf()-object in the case of vector data and raster::raster()-object in the case of a raster) specified by the user (i.e. the user has to indicate output files) will be loaded into R. A list will be returned if there is more than one output file (e.g., grass7:r.slope.aspect). See the example section for more details.
show_output_paths	Logical. QGIS computes all possible output files for a given geoalgorithm, and saves them to a temporary location in case the user has not specified explicitly another output location. Setting show_output to TRUE (the default) will print all output paths to the console after the successful geoprocessing.
NA_flag	Value used for NAs when exporting raster objects through pass_args() and save_spatial_objects() (default: -99999).
qgis_env	Environment containing all the paths to run the QGIS API. For more information, refer to set_env().

Details

This workhorse function calls the QGIS Python API, and specifically processing.runalg.

Value

The function prints a list (named according to the output parameters) containing the paths to the files created by QGIS. If not otherwise specified, the function saves the QGIS generated output files to a temporary folder (created by QGIS). Optionally, function parameter load_output loads spatial QGIS output (vector and raster data) into R.

Note

Please note that one can also pass spatial R objects as input parameters where suitable (e.g., input layer, input raster). Supported formats are `sp::SpatialPointsDataFrame()`-, `sp::SpatialLinesDataFrame()`-, `sp::SpatialPolygonsDataFrame()`-, `sf::sf()`- (of class `sf`, `sfc` as well as `sfg`), and `raster::raster()`-objects. See the example section for more details.

GRASS users do not have to specify manually the GRASS region extent (function argument GRASS_REGION_PARAMETE If "None" (the QGIS default), `run_qgis` (see `pass_args()` for more details) will automatically determine the region extent based on the user-specified input layers. If you do want to specify it yourself, please do it in accordance with the QGIS documentation, i.e., use a character string and separate the coordinates with a comma: "xmin, xmax, ymin, ymax".

Author(s)

Jannes Muenchow, Victor Olaya, QGIS core team

Examples

```
## Not run:
# calculate the slope of a DEM
# load dem - a raster object
data(dem, package = "RQGIS")
# find out the name of a GRASS function with which to calculate the slope
find_algorithms(search_term = "grass7.*slope")
# find out how to use the function
alg <- "grass7:r.slope.aspect"
get_usage(alg)
# 1. run QGIS using R named arguments, and load the QGIS output back into R
slope <- run_qgis(alg, elevation = dem, slope = "slope.asc",
                  load_output = TRUE)
# 2. doing the same with a parameter-argument list
params <- list(elevation = dem, slope = "slope.asc")
slope <- run_qgis(alg, params = params, load_output = TRUE)
# 3. calculate the slope, the aspect and the pcurvature.
terrain <- run_qgis(alg, elevation = dem, slope = "slope.asc",
                    aspect = "aspect.asc", pcurvature = "pcurv.asc",
                    load_output = TRUE)
# the three output rasters are returned in a list of length 3
terrain

## End(Not run)
```

`set_env` *Retrieve the environment settings to run QGIS from within R*

Description

`set_env` tries to find all the paths necessary to run QGIS from within R.

Usage

```
set_env(root = NULL, new = FALSE, dev = FALSE, ...)
```

Arguments

root	Root path to the QGIS-installation. If left empty, the function looks for qgis.bat first in the most likely locations (C:/OSGEO4~1, C:/OSGEO4~2), and secondly on the C: drive under Windows. On a Mac, it looks for QGIS.app under "Applications" and "/usr/local/Cellar/". On Linux, set_env assumes that the root path is "/usr".
new	When called for the first time in an R session, set_env caches its output. Setting new to TRUE resets the cache when calling set_env again. Otherwise, the cached output will be loaded back into R even if you used new values for function arguments root and/or dev.
dev	If set to TRUE, set_env will use the development version of QGIS (if available). Since RQGIS so far does not support QGIS 3 (developer version), setting dev to TRUE will result in an error message under Windows.
...	Currently not in use.

Value

The function returns a list containing all the path necessary to run QGIS from within R. This is the root path, the QGIS prefix path and the path to the Python plugins.

Author(s)

Jannes Muenchow, Patrick Schratz

Examples

```
## Not run:
# Letting set_env look for the QGIS installation might take a while depending
# on how full the C: drive is (Windows)
set_env()
# It is much faster (0 sec) to explicitly state the root path to the QGIS
# installation
set_env("C:/OSGEO4~1")  # Windows example

## End(Not run)
```

study_area *Mask of the study area*

Description

An sf (EPSG:32717) object of geometry class polygon.

Format

An sf object with 1 row and 2 variables:

name Name.

geometry Simple feature polygon geometry.

References

Muenchow, J., Bräuning, A., Rodríguez, E.F. & von Wehrden, H. (2013): Predictive mapping of species richness and plant species' distributions of a Peruvian fog oasis along an altitudinal gradient. Biotropica 45, 5, 557-566, doi: 10.1111/btp.12049.

Index