QGIS in R with qgisprocess :: CHEAT SHEET

Mission

The goal of qgisprocess is to provide an R interface to the geoprocessing algorithms of QGIS, a popular and open source desktop geographic information system (GIS) program. This package is a re-implementation of the functionality provided by the archived **RQGIS** package, which was partially revived in the **RQGIS3** package.

Features

This package makes it easier to use native processing algorithms and some from GDAL, GRASS and many others (like SAGA).

Providers	Algorithms
qgis	50 + 242 (c ++) + 1 (3D)
gdal	56
grass	304
third-party providers	x
Total counts	653 + x

```
# Show a tibble with processing providers
> qgis_providers( )
# Show a tibble with algorithms
> qgis_algorithms( )
# Search algorithms using regular expressions
> qgis_search_algorithms(
      algorithm = <x>,
      provider = <y>,
      group = <z>
      )
```

Installation

```
> install.packages("remotes")
> install_github("r-spatial/qgisprocess")
> library(qgisprocess)
```

GNU/Linux, macOS, Windows

If needed, specify path to QGIS installation before loading qgisprocess:

```
> options("qgisprocess.path" = "C:/Program Files/
    QGIS 3.30/bin/qgis_process-qgis.bat")
```

Using docker

1.Get started with the installation of docker in your machine.
2.Download the image of geocomputation

```
> docker pull geocompr/geocompr:qgis-ext
```

3. Run to image of geocomputation with docker

```
> docker run -d -p 8786:8787 -v $(pwd):/home/rstudio/
    data -e PASSWORD=pw geocompr/geocompr:qgis-ext
```

Input functions

The package offers new functionalities of Input to have a workflow of an easy manner inside of R.

```
# Show a description of the function to use
> qgis_show_help(algorithm ="native:creategrid")

# Show all the parameters of the function
> qgis_get_argument_specs(algorithm = "native:
    creategrid")
```

```
# Run the algorithms
> qgis_run_algorithm(
    algorithm = "native:creategrid",
    TYPE = 4,
    EXTENT = c("794599, 798208, 8931775,8935384"),
    HSPACING = 1000 ,
    VSPACING = 1000,
    CRS = "EPSG:32717",
    OUTPUT = 'grid'
      )

# Create a function based on the algorithm to use
> grid_fun <- qgis_function('native:creategrid')
> grid_fun(
    TYPE = 4,
    EXTENT = c("794599,798208,8931775,8935384"),
    HSPACING = 1000,
    VSPACING = 1000,
    CRS = "EPSG:32717",
    OUTPUT = "grid"
     )
```

Output functions

qgisprocess give us new functionalities of output for vector, raster and other format file, and it is possible loads it to our environment work.

```
> qgis_extract_output(result_run_alg, 'OUTPUT')
```

```
# A character vector indicating the location of a
    temporary file.
> qgis_tmp_base( )
> qgis_tmp_file( ".csv" )
> qgis_tmp_vector( )
> qgis_tmp_raster( )
```

Pipe integration

qgisprocess also provides qgis_run_algorithm_p() that works better in pipelines.

```
# Buffer processing
> library(sf)
> system.file(
    "longlake/longlake_depth.gpkg",
    package = "qgisprocess"
    ) |>
  qgis_run_algorithm_p(
    algorithm = "native:buffer",
    DISTANCE = 100
    ) |> st_as_sf() |>
    plot()
```

Workflow

Vector data

```
# Hexagrid of 400x400
> library(sf)
> grid_fun <- qgis_function("native:creategrid")
> grid_fun(
    TYPE = 4,
    EXTENT = c("409967, 411658, 5083354, 5084777"),
    HSPACING = 400,
    VSPACING = 400,
    CRS = "EPSG:26920",
    OUTPUT = "grid"
    ) |>
    st_as_sf() |>
    select(id) |>
    plot()
```

Raster data

```
# TWI processing
 > library(stars)
 > dem <- read_stars(
    system.file(
      "raster/nz_elev.tif",
       package = "spDataLarge")
      )
 > qgis_run_algorithm(
    algorithm ="sagang:sagawetnessindex",
    DEM = dem,
    TPI = "tpi.sdat") |>
    qgis_extract_output("TWI") |>
    qgis_as_terra() |>
    plot(col = cptcity::cpt(pal = "ocal_blues"))
```