QGIS in R with qgisprocess :: CHEAT SHEET

Mission

The main objective is provide to the R interface the most popular open-source desktop GIS program like QGIS. 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 functions from QGIS and some from GDAL, GRASS and many others (like SAGA).

Provider	Algorithms
qgis	50 + 242 (c ++) + 1 (3D)
gdal	56
grass	304
< third-party providers >	x
Total count	653 + x

```
> qgis_providers( )
```
Show a tibble with processing providers

```
> qgis_algorithms( )
```
Show a tibble with algorithms

```
> qgis_search_algorithms(algorithm = <x>,
provider = <y>, group = <z>)
```
Search algorithms using regular expressions

Installation

```
> install.packages('remotes')
> 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.

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

Algorithm name

```
qgis_get_argument_specs(algorithm =
  'native:creategrid')
```
Show all the parameters of the function.

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.

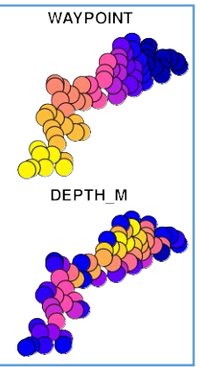
```
qgis_extract_output(result_run_alg, 'OUTPUT')
```

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

Pipe integration

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

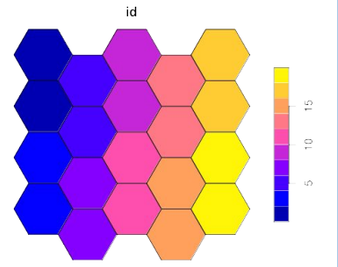
```
library(sf)
system.file('longlake/longlake_depth.gpkg',
              package = 'qgisprocess') |>
  qgis_run_algorithm_p(
    algorithm = 'native:buffer',
    DISTANCE = 100,
  ) |>
  st_as_sf( ) |>
  plot( )

qgis_fun(...)
```



Workflows

Vector data

```
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_af() |>
  select(id) |>
  plot()
```



Raster data

```
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') %>%
  st_as_stars() |>
  plot(col =
    cptcity::cpt(pal = 'ocal_blues'))
```