QGIS en R con qgisprocess :: CHEAT SHEET

Misión

El objetivo de qgisprocess es proporcionar una interfaz de R para los algoritmos de geoprocesamiento de QGIS, un popular programa de sistema de información geográfica (SIG) de código abierto. Este paquete es una reimplementación de la funcionalidad proporcionada por el paquete archivado **RQGIS** o la que parcialmente fue revivido en el paquete **RQGIS3**

Características

Este paquete facilita el uso de algoritmos de procesamiento nativos y algunos de GDAL, GRASS y muchos otros (como SAGA).

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

```
# Mostrar un tibble con proveedores de processing
> qgis_providers( )
# Mostrar un tibble con algoritmos
> qgis_algorithms( )
# Buscar algoritmos con expresiones regulares
> qgis_search_algorithms(
      algorithm = <x>,
      provider = <y>,
      group = <z>
      )
```

Instalación

```
> install.packages("qgisprocess")
> library(qgisprocess)
```

GNU/Linux, macOS, Windows

Si es necesario, especifique la ruta a la instalación de QGIS antes de cargar qgisprocess:

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

Usando docker

1.Instalación de docker.
2.Descargar la imagen de docker llamado geocomputación.

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

3. Iniciar la imagen de gecomputación con docker.

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

Funciones de entrada

El paquete ofrece nuevas funcionalidades de entrada para tener un flujo de trabajo más amigable en el entorno de R.

```
# Mostrar una descripciön de ayuda de la funciön a
    usar
> qgis_show_help(algorithm ="native:creategrid")

# Mostrar todos los parämetros de la funciön
> qgis_get_argument_specs(algorithm = "native:
    creategrid")



# Inicar la ejecucion del algoritmo
> qgis_run_algorithm(
    algorithm = "native:creategrid",
    TYPE = 4,
    EXTENT = c("794599, 798208, 8931775,8935384"),
    HSPACING = 1000 ,
    VSPACING = 1000,
    CRS = "EPSG:32717",
    OUTPUT = "grid"
      )

# Crear una funciön basada en un algoritmo especifico
> 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"
    )
```

Funciones de salida

qgisprocess nos brinda nuevas funcionalidades para la generación de archivos en formatos vectoriales, raster y otros. Además, permite cargar fácilmente a nuestro entorno de trabajo.

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



# Un vector de caracteres que indica la ubicaciön de
    un archivo temporal.
> qgis_tmp_base( )
> qgis_tmp_file( ".csv" )
> qgis_tmp_vector( )
> qgis_tmp_raster( )
```

Uso de Pipe

qgisprocess proporciona una funciön llamada qgis_run_algorithm_p() que integra de mejor forma el uso de pipelines.

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

Flujo de trabajo

Vector data

```
# Creaciön de un hexagrid de 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

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