Geodata and algorithms in R

RQGIS3

Jannes Muenchow

DAAD summer school

https://github.com/giscience-fsu/daad_summerschool

Please install following packages:

```
install.packages(c("sf", "raster", "spData", "dplyr", "RQGIS"))
```

Or use our geocompr docker image. See the geocompr landing page for instructions how to use it.

Installing QGIS3

- **Follow** the steps described in `vignette(install_guide, package = "RQGIS3")`!

Installing QGIS3

- **Follow** the steps described in `vignette(install_guide, package = "RQGIS3")`!
- Windows users: Use the OSGeo-network-installer (also described in the vignette)!

Installing RQGIS3

You have to install the developer version

```
devtools::install_github("jannes-m/RQGIS3")
```

For more information and a short introduction by example refer to:
https://github.com/jannes-m/RQGIS3

A word of warning

On UNIX-based systems **RQGIS3** and RStudio **do not** work in harmony, i.e., trying to run any **RQGIS3** algorithm will crash the RStudio R session. However, running **RQGIS3** in the shell works perfectly fine, i.e., it is an RStudio issue. Unfortunately, the RStudio guys could not solve the problem so far (see this issue). But the good news it that **RQGIS3** works with RStudio Server, which means if you are on a UNIX-based system, simply use our docker image by running:

How to use Docker (on UNIX-based systems)

First, you have to install docker, use the package manager of your OS to do so (e.g., `sudo apt-get install docker` under Ubuntu/Debian). To use docker type the following commands in a shell (see here). Probably this does not work for MacOS, please visit https://docs.docker.com/docker-for-mac/install/ for how to set up docker for MacOS.

```
# start and enable the Docker daemon usinng system
sudo systemctl start docker.service  # start immediately the Docker d
sudo systemctl enable docker.service  # ensure that daemon will start
# note that you can run Docker only as root. To run Docker as a regul
# create a new group called docker
sudo groupadd docker
# add your user to the group
sudo gpasswd -a "$USER" docker

docker run -d -e PASSWORD=pass -p 8787:8787 robinlovelace/geocompr
```

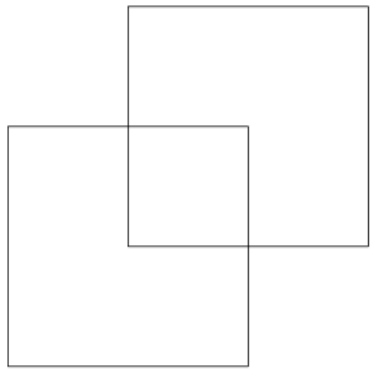
Open http://localhost:8787 in your preferred browser and use `rstudio` as username and `pass` as password.

RQGIS3 by example

To introduce the **RQGIS3** package, let's find the intersection between two polygons. For this we create two polygons using the `sf`-package.

```
library(sf)
coords_1 =
  matrix(data =
           c(0, 0, 1, 0, 1, 1,0, 1, 0, 0),
         ncol = 2, byrow = TRUE)
coords_2 =
  matrix(data =
           c(-0.5, -0.5, 0.5, -0.5, 0.5,
             0.5,-0.5, 0.5, -0.5, -0.5),
         ncol = 2, byrow = TRUE)

poly_1 = st_polygon(list((coords_1))) %>%
  st_sfc %>%
  st_sf
poly_2 = st_polygon(list((coords_2))) %>%
  st_sfc %>%
  st_sf
plot(poly_1$geometry, xlim = c(-1, 1), ylim = c(
plot(poly_2$geometry, add = TRUE)
```



Find a QGIS3 algorithm

Now we would like to know which QGIS3 geoalgorithm we can use for this task. We assume that the word `intersec` will be part of the short description of the searched geoalgorithm.

```
library("RQGIS3")
find_algorithms("intersec", name_only = TRUE)
```

```
## [1] "native:intersection"      "native:lineintersections"
```

How to use it

To find out the parameter names and corresponding default values, use `get_usage()`.

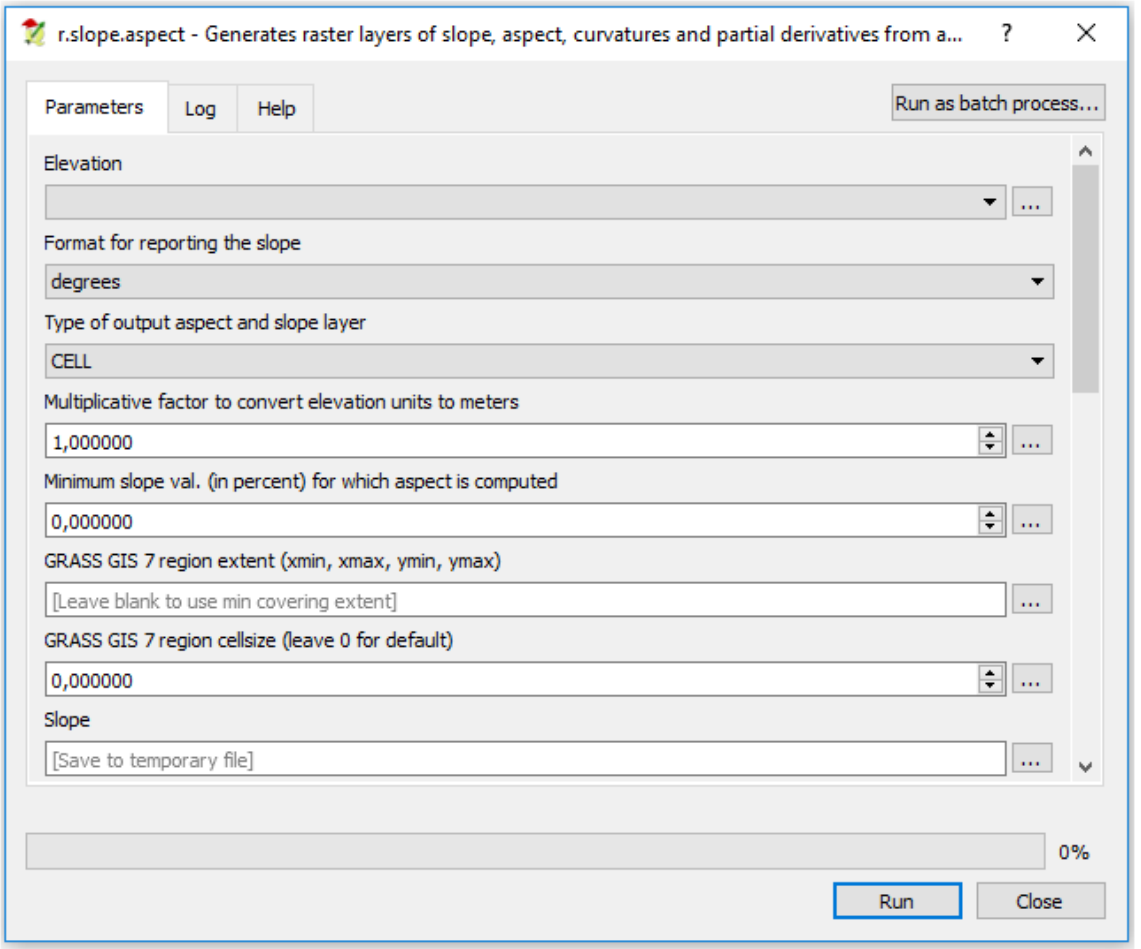
```
get_usage("native:intersection")
```

```
## Intersection (native:intersection)
##
## This algorithm extracts the overlapping portions of features in the Input and Overlay layers. Features in the ou
##
##
## ----------------
## Input parameters
## ----------------
##
## INPUT: Input layer
##
##     Parameter type:    QgsProcessingParameterFeatureSource
##
##     Accepted data types:
##         - str: layer ID
##         - str: layer name
##         - str: layer source
##         - QgsProcessingFeatureSourceDefinition
##         - QgsProperty
##         - QgsVectorLayer
##
## OVERLAY: Overlay layer
##
##     Parameter type:    QgsProcessingParameterFeatureSource
##
##     Accepted data types:
##         - str: layer ID
##         - str: layer name
```

Here, we only have three function arguments, and automatic parameter collection is not necessary, but when I first looked at...

```
get_usage("grass7:r.slope.aspect")
```

But looking at the QGIS GUI...



Convenience function `get_args_man`

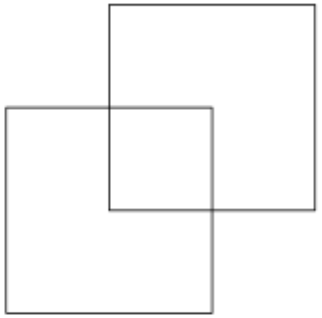

```
params = get_args_man(alg = "grass7:r.slope.aspect")
params[1:10]
```

```
## Choosing default values for following parameters:
## format: 0
## precision: 0
## See get_options('grass7:r.slope.aspect') for all available options.


## $elevation
## [1] "None"
##
## $format
## [1] "0"
##
## $precision
## [1] "0"
##
## $`-a`
## [1] "True"
##
## $zscale
## [1] "1.0"
##
## $min_slope
## [1] "0.0"
##
## $slope
## [1] "None"
##
## $aspect
## [1] "None"
##
## $pcurvature
## [1] "None"
##
## $tcurvature
## [1] "None"
```

Back to our use case

We have created two polygons using `sf`, and would like to find the intersection between the two.



Back to our use case

We also know the name of the geoalgorithm (`native:intersection`), and its parameters

```
## Intersection (native:intersection)
##
## This algorithm extracts the overlapping portions of features in the Input
##
##
## ----------------
## Input parameters
## ----------------
##
## INPUT: Input layer
##
##     Parameter type:    QgsProcessingParameterFeatureSource
##
##     Accepted data types:
##         - str: layer ID
##         - str: layer name
##         - str: layer source
##         - QgsProcessingFeatureSourceDefinition
##         - QgsProperty
##         - QgsVectorLayer
```

Back to our use case

We also know the name of the geoalgorithm (`native:intersection`), and its parameters

```
## Intersection (native:intersection)
##
## This algorithm extracts the overlapping portions of features in the Input
##
##
## ----------------
## Input parameters
## ----------------
##
## INPUT: Input layer
##
##     Parameter type:    QgsProcessingParameterFeatureSource
##
##     Accepted data types:
##         - str: layer ID
##         - str: layer name
##         - str: layer source
##         - QgsProcessingFeatureSourceDefinition
##         - QgsProperty
##         - QgsVectorLayer
```

Run QGIS3 from within R


```
int = run_qgis("native:intersection",
               INPUT = poly_1,
               OVERLAY = poly_2,
               OUTPUT = file.path(tempdir(), "out.shp"),
               load_output = TRUE)
```

```
## $OUTPUT
## [1] "/tmp/Rtmp4OxNtm/out.shp"
```

Spatial objects as inputs

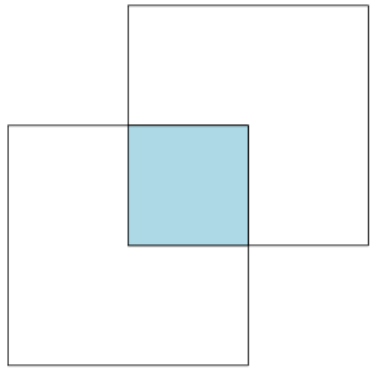

```
int = run_qgis("native:intersection",
               INPUT = poly_1,
               OVERLAY = poly_2,
               OUTPUT = file.path(tempdir(), "out.shp"),
               load_output = TRUE)
```

Load QGIS3 output into R


```
int = run_qgis("native:intersection",
               INPUT = poly_1,
               OVERLAY = poly_2,
               OUTPUT = file.path(tempdir(), "out.shp"),
               load_output = TRUE)
```

Visualizing the result

```
plot(poly_1$geometry,
     xlim = c(-1, 1),
     ylim = c(-1, 1))
plot(poly_2$geometry,
     add = TRUE)
plot(int$geometry,
     col = "lightblue",
     add = TRUE)
```



Further reading

- RQGIS R journal paper
- https://geocompr.robinlovelace.net/gis.html

Your turn

1. Let us (together) reproduce the `native:intersection` example (download code).

Your turn

1. Let us (together) reproduce the `native:intersection` example (download code).
2. Since we could also use **sf** to do the intersection (see also task 3), we will now compute the SAGA wetness index - an geoalgorithm unavailable in R. Calculate the SAGA wetness index of `data(dem)` using **RQGIS3**. If you are faster than the others or if you have trouble using SAGA, calculate the slope, the aspect (and the curvatures) of `data(dem)` using GRASS through **RQGIS3**.

Your turn

1. Let us (together) reproduce the `native:intersection` example (download code).
2. Since we could also use **sf** to do the intersection (see also task 3), we will now compute the SAGA wetness index - an geoalgorithm unavailable in R. Calculate the SAGA wetness index of `data(dem)` using **RQGIS3**. If you are faster than the others or if you have trouble using SAGA, calculate the slope, the aspect (and the curvatures) of `data(dem)` using GRASS through **RQGIS3**.
3. Optional: calculate the intersection of `poly_1` and `poly_2` with the help of **sf**, **RSAGA** and/or **rgrass7** (Chapter 9 of Geocomputation with R might be of help here).

Your turn

1. Let us (together) reproduce the `native:intersection` example (download code).
2. Since we could also use **sf** to do the intersection (see also task 3), we will now compute the SAGA wetness index - an geoalgorithm unavailable in R. Calculate the SAGA wetness index of `data(dem)` using **RQGIS3**. If you are faster than the others or if you have trouble using SAGA, calculate the slope, the aspect (and the curvatures) of `data(dem)` using GRASS through **RQGIS3**.
3. Optional: calculate the intersection of `poly_1` and `poly_2` with the help of **sf**, **RSAGA** and/or **rgrass7** (Chapter 9 of Geocomputation with R might be of help here).
4. Optional: Select randomly a point from `random_points` and find all `dem` pixels that can be seen from this point (hint: viewshed). Visualize your result. Plot a hillshade, and on top of it the digital elevation model, your viewshed output and the point. Additionally, give `mapview` a try.