Geocomputation with R

⚔

Introducing RQGIS and RSAGA

Jannes Muenchow

GeoStats 2018

Find the slides and the code

https://github.com/geocompr/geostats_18

Installing QGIS

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

Installing QGIS

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

Installing RQGIS

You can either install the developer...

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

Installing RQGIS

You can either install the developer...

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

... or the CRAN version

```
install.packages("RQGIS")
```

Installing RQGIS

You can either install the developer...

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

... or the CRAN version

```
install.packages("RQGIS")
```

For more information and a short introduction by example refer to:

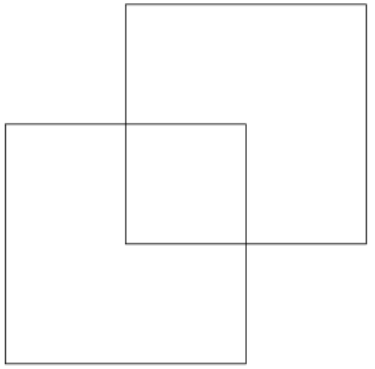
https://github.com/jannes-m/RQGIS

RQGIS by example

To introduce the RQGIS 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$geometry, xlim = c(-1, 1), ylim = c(-1
plot(poly_2$geometry, add = TRUE)
```



Find a QGIS algorithm

Now we would like to know which QGIS 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(RQGIS)
set_env(dev = FALSE)
```

```
## $root
## [1] "/usr"
##
## $qgis_prefix_path
## [1] "/usr/bin/qgis"
##
## $python_plugins
## [1] "/usr/share/qgis/python/plugins"
```

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

```
## [1] "qgis:intersection"             "qgis:lineintersections"
## [3] "saga:fuzzyintersectionand"     "saga:intersect"
## [5] "saga:linepolygonintersection" "saga:polygonselfintersection"
## [7] "saga:polygonlineintersection"
```

How to use it

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

```
get_usage("qgis:intersection")
```

```
## ALGORITHM: Intersection
##      INPUT <ParameterVector>
##      INPUT2 <ParameterVector>
##      IGNORE_NULL <ParameterBoolean>
##      OUTPUT <OutputVector>
```

How to use it

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

```
get_usage("qgis:intersection")
```

```
## ALGORITHM: Intersection
##      INPUT <ParameterVector>
##      INPUT2 <ParameterVector>
##      IGNORE_NULL <ParameterBoolean>
##      OUTPUT <OutputVector>
```

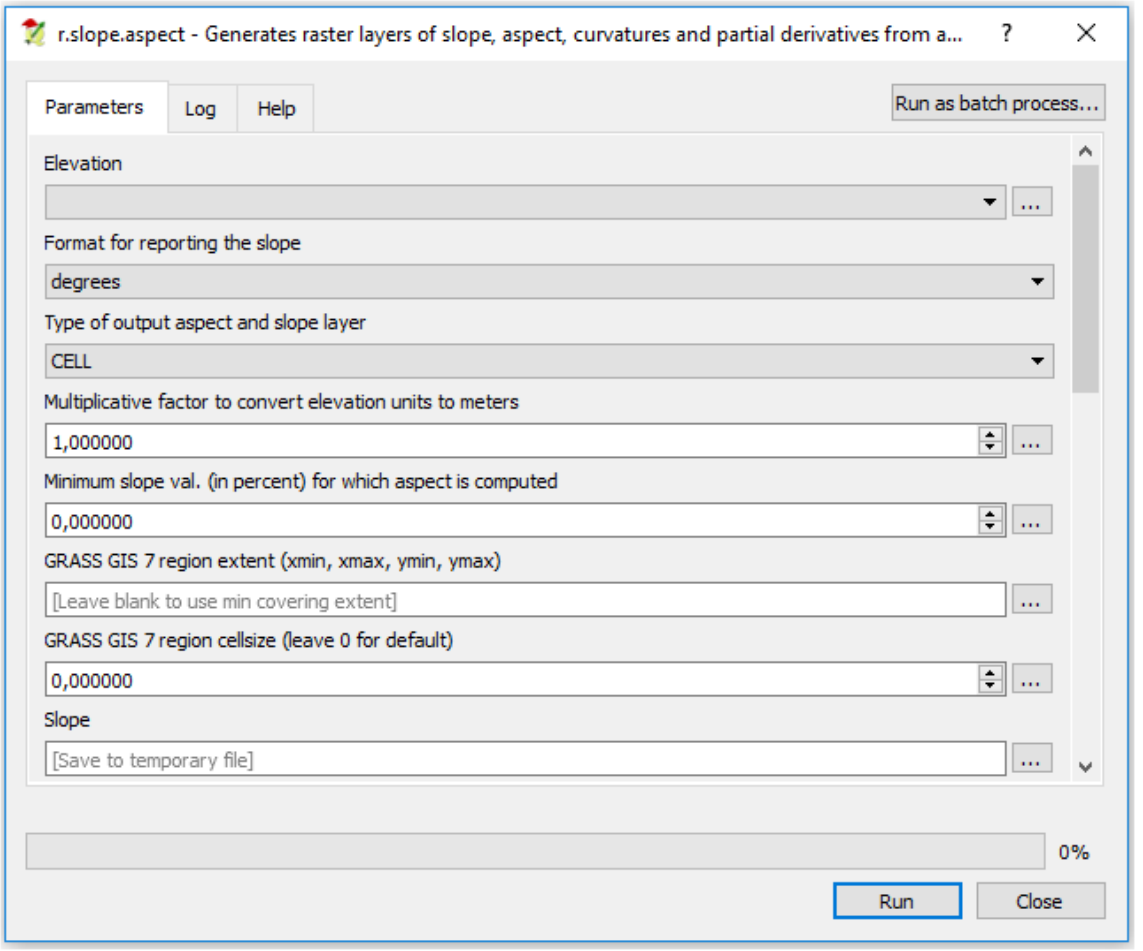
Here, we only have three function arguments, and automatic parameter collection is not necessary, but when I first looked at...

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

```
ALGORITHM: r.slope.aspect - Generates raster layers of slope, aspect, curvat
    elevation <ParameterRaster>
    format <ParameterSelection>
    precision <ParameterSelection>
    -a <ParameterBoolean>
    zscale <ParameterNumber>
    min_slope <ParameterNumber>
    GRASS_REGION_PARAMETER <ParameterExtent>
    GRASS_REGION_CELLSIZE_PARAMETER <ParameterNumber>
    slope <OutputRaster>
    aspect <OutputRaster>
    pcurvature <OutputRaster>
    tcurvature <OutputRaster>
    dx <OutputRaster>
    dy <OutputRaster>
    dxx <OutputRaster>
    dyy <OutputRaster>
    dxy <OutputRaster>

format(Format for reporting the slope)
    0 - degrees
    1 - percent
precision(Type of output aspect and slope layer)
    0 - FCELL
    1 - CELL
    2 - DCELL
```

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.
```

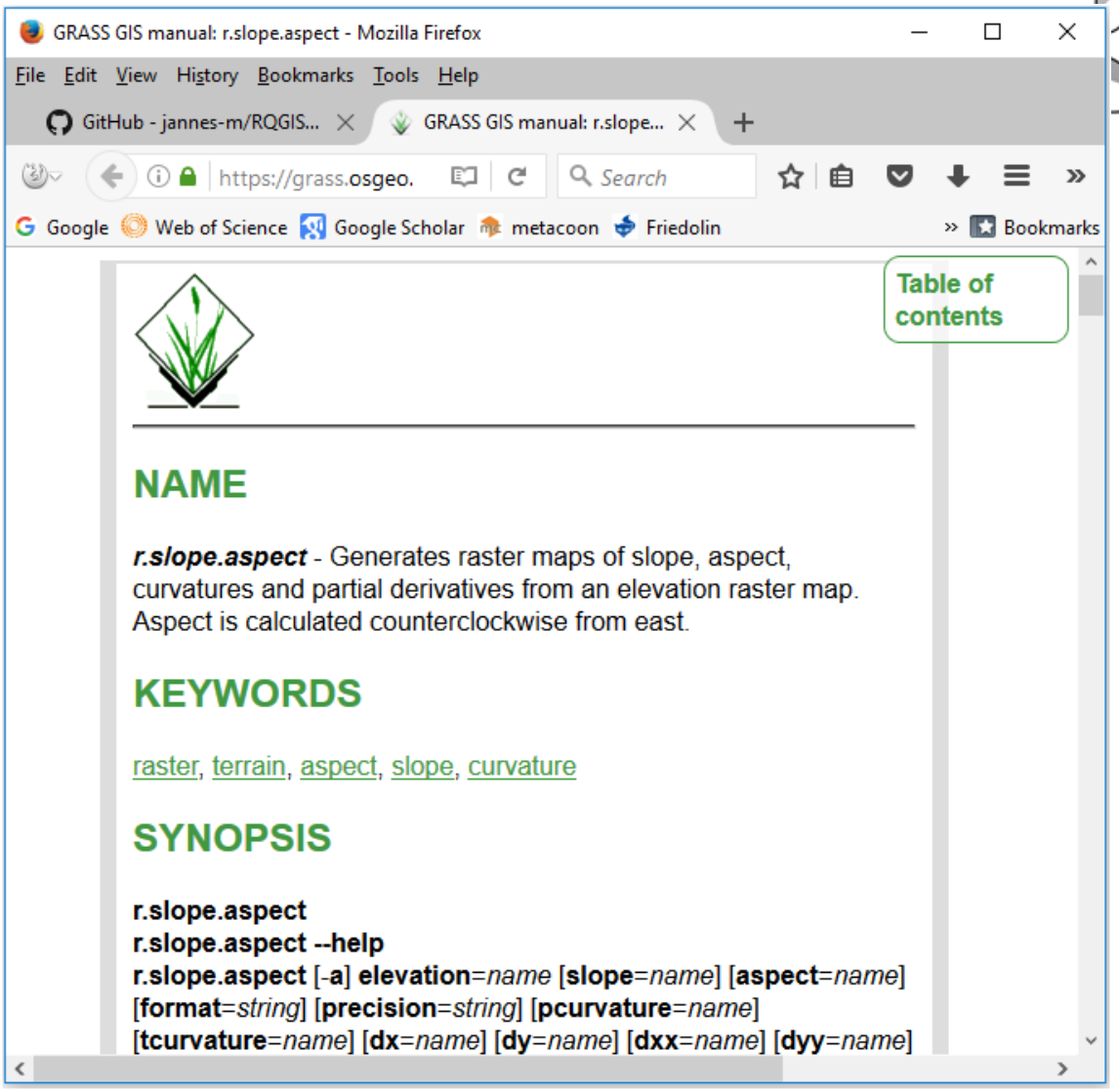
```
## $elevation
## [1] "None"
##
## $format
## [1] "0"
##
## $precision
## [1] "0"
##
## $`-a`
## [1] "True"
##
## $zscale
## [1] "1.0"
```

```
## $min_slope
## [1] "0.0"
##
## $GRASS_REGION_PARAMETER
## [1] "\"None\""
##
## $GRASS_REGION_CELLSIZE_PARAMETER
## [1] "0.0"
##
## $slope
## [1] "None"
##
## $aspect
## [1] "None"
```

Access the online help

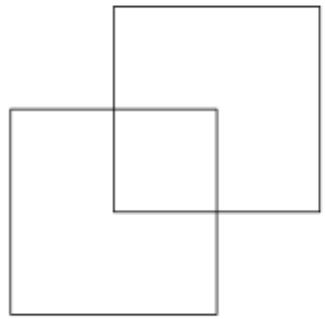
By the way, use `open_help` to access the online help and possibly find out more about a specific geoalgorithm:

```
library(RQGIS)
open_help(alg = "grass7:r.slope.aspect")
```


GRASS GIS manual: r.slope.aspect - Mozilla Firefox
File Edit View History Bookmarks Tools Help
GitHub - jannes-m/RQGIS...
GRASS GIS manual: r.slope...
https://grass.osgeo.
Search
Google
Web of Science
Google Scholar
metacoon
Friedolin
Bookmarks
Table of contents
NAME
r.slope.aspect - Generates raster maps of slope, aspect, curvatures and partial derivatives from an elevation raster map. Aspect is calculated counterclockwise from east.
KEYWORDS
raster, terrain, aspect, slope, curvature
SYNOPSIS
r.slope.aspect
r.slope.aspect --help
r.slope.aspect [-a] elevation=name [slope=name] [aspect=name] [format=string] [precision=string] [pcurvature=name] [tcurvature=name] [dx=name] [dy=name] [dxx=name] [dyy=name]

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 (`qgis:intersection`), and its parameters

```
## ALGORITHM: Intersection
##     INPUT <ParameterVector>
##     INPUT2 <ParameterVector>
##     IGNORE_NULL <ParameterBoolean>
##     OUTPUT <OutputVector>
```

Back to our use case

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

```
## ALGORITHM: Intersection
##     INPUT <ParameterVector>
##     INPUT2 <ParameterVector>
##     IGNORE_NULL <ParameterBoolean>
##     OUTPUT <OutputVector>
```

Hence, we have to specify `INPUT`, `INPUT2` and `OUTPUT`. We can do so using R named arguments.

Run QGIS from within R

```
int = run_qgis("qgis:intersection",
               INPUT = poly_1,
               INPUT2 = poly_2,
               OUTPUT = file.path(tempdir(), "out.shp"),
               load_output = TRUE)
```

```
## $OUTPUT
## [1] "/tmp/RtmpmcNs0Z/out.shp"
```

Spatial objects as inputs

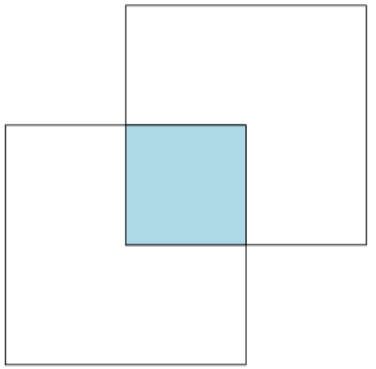
```
int = run_qgis("qgis:intersection",
               INPUT = poly_1,
               INPUT2 = poly_2,
               OUTPUT = file.path(tempdir(), "out.shp"),
               load_output = TRUE)
```

Load QGIS output into R

```
int = run_qgis("qgis:intersection",
               INPUT = poly_1,
               INPUT2 = 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 (R)QGIS reading

- RQGIS R Journal paper (Muenchow, Schratz, and Brenning, 2017).
- Nice paper on QGIS and its architecture (Graser and Olaya, 2015).
- https://geocompr.robinlovelace.net/gis.html

RSAGA

Ok, let's do the same using **RSAGA**. First, we need to tell our system where SAGA is installed, `rsaga.env()` searches our system automatically for a SAGA installation.

```
library("RSAGA")
env = rsaga.env()
```

```
## Search for SAGA command line program and modules...
## Done
```

```
# if this doesn't work, specifically set the path to your SAGA inst
# rsaga.env(path = "C:/OSGeo4W64/apps/saga-ltr/")
```

SAGA modules

Remember SAGA is structured in modules. It also might to have a look at the SAGA GUI. Let's have a look at the available module libraries.

```
library("dplyr")
rsaga.get.libraries() %>%
  grep("shapes", ., value = TRUE)
```

```
## Search for SAGA command line program and modules...
## Done
```

```
## [1] "io_shapes_dxf"   "io_shapes"       "shapes_grid"     "shapes_lines"
## [5] "shapes_points"   "shapes_polygons" "shapes_tools"    "shapes_transe
```

Geoalgorithms

We want to intersect two polygon layers, so we would assume to find a corresponding function in module `shapes_polygons`.

```
algs = rsaga.get.modules(libs = "shapes_polygons")
```

```
## Search for SAGA command line program and modules...
## Done
```

```
tail(algs[[1]], 10)
```

```
##    code                                name interactive
## 12   12         Polygon Self-Intersection       FALSE
## 13   14                         Intersect       FALSE
## 14   15                        Difference       FALSE
## 15   16            Symmetrical Difference       FALSE
## 16   17                             Union       FALSE
## 17   18                            Update       FALSE
## 18   19                          Identity       FALSE
## 19   20 Add Point Attributes to Polygons       FALSE
## 20   21             Flatten Polygon Layer       FALSE
## 21   22              Shared Polygon Edges       FALSE
```

How to use a specific geoalgorithm

Now that we found out that there is an `Intersect` command, we need to know its parameters.

```
rsaga.get.usage(lib = "shapes_polygons", module = "Intersect")
```

```
## Search for SAGA command line program and modules...
## Done
## library path: /usr/lib/saga/
## library name: libshapes_polygons
## library     : Polygons
## Usage: saga_cmd shapes_polygons 14 [-A <str>] [-B <str>] [-RESULT <str>]
##   -A:<str>          Layer A
##     Shapes (input)
##   -B:<str>          Layer B
##     Shapes (input)
##   -RESULT:<str>     Intersect
##     Shapes (output)
##   -SPLIT:<str>      Split Parts
##     Boolean
```

Run SAGA

Ok, before running SAGA, we need to export our `sf`-objects.

```
file_1 = file.path(tempdir(), "poly_1.shp")
file_2 = file.path(tempdir(), "poly_2.shp")
write_sf(poly_1, file_1, quiet = TRUE)
write_sf(poly_2, file_2, quiet = TRUE)
```

Now, run SAGA

`rsaga.geoprocessor` is the workhorse that calls the SAGA algorithms using the command-line API. Parameters and corresponding arguments have to be specified in a list (looks a bit like `RQGIS:get_args_man()`).

```
rsaga.geoprocessor(lib = "shapes_polygons", module = "Intersect",
                   list(A = file_1,
                        B = file_2,
                        RESULT = file.path(tempdir(), "int.shp"),
                        SPLIT = FALSE),
                   show.output.on.console = FALSE)
```

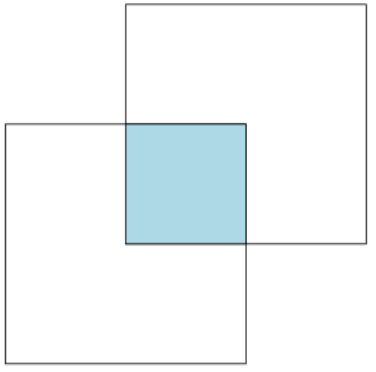
```
## Search for SAGA command line program and modules...
## Done
```

Visualize it

```
file = file.path(tempdir(), "int.shp")
int = st_read(file)
plot(st_geometry(poly_1))
plot(st_geometry(poly_2), add = TRUE)
plot(st_geometry(int), add = TRUE,
     col = "lightblue")
```

Visualize it

```
file = file.path(tempdir(), "int.shp")
int = st_read(file)
plot(st_geometry(poly_1))
plot(st_geometry(poly_2), add = TRUE)
plot(st_geometry(int), add = TRUE,
     col = "lightblue")
```



Further (R)SAGA reading

- We recommend reading `vignette("RSAGA")` for a deeper look at **RSAGA**.
- Nice paper on SAGA, it's history and architecture (Conrad, Bechtel, Bock, Dietrich, Fischer, Gerlitz, Wehberg, Wichmann, and Böhner, 2015)
- https://geocompr.robinlovelace.net/gis.html

Your turn

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

Your turn

1. Let us (together) reproduce the `qgis: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 RQGIS. 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 RQGIS.

Your turn

1. Let us (together) reproduce the `qgis: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 RQGIS. 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 RQGIS.
3. Optional: calculate the intersection of `poly_1` and `poly_2` with the help of `sf`, SAGA and/or GRASS (hint: overlay and `open_help`).

Your turn

1. Let us (together) reproduce the `qgis: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 RQGIS. 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 RQGIS.
3. Optional: calculate the intersection of `poly_1` and `poly_2` with the help of `sf`, SAGA and/or GRASS (hint: overlay and `open_help`).
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.

References

Conrad, O, B. Bechtel, M. Bock, et al. (2015). "System for Automated Geoscientific Analyses (SAGA) v. 2.1.4". In: *Geosci. Model Dev.* 8.7, pp. 1991-2007. ISSN: 1991-9603. DOI: 10.5194/gmd-8-1991-2015. URL: http://www.geosci-model-dev.net/8/1991/2015/ (visited on Jun. 12, 2017).

Graser, Anita and Victor Olaya (2015). "Processing: A Python Framework for the Seamless Integration of Geoprocessing Tools in QGIS". En. In: *ISPRS International Journal of Geo-Information* 4.4, pp. 2219-2245. ISSN: 2220-9964. DOI: 10.3390/ijgi4042219. URL: http://www.mdpi.com/2220-9964/4/4/2219 (visited on Jul. 21, 2018).

Muenchow, Jannes, Patrick Schratz and Alexander Brenning (2017). "RQGIS: Integrating R with QGIS for Statistical Geocomputing". In: *The R Journal* 9 (2). Accepted for publication on 2017-12-04, pp. 409-428. URL: https://rjournal.github.io/archive/2017/RJ-2017-067/RJ-2017-067.pdf.