Introducing the RQGIS package

Jannes Muenchow

2018-04-27

Find the slides and the code

https://github.com/jannes-m/erum18_geocompr

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...

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

Installing RQGIS

You can either install the developer...

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

... or the CRAN version

```r
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:

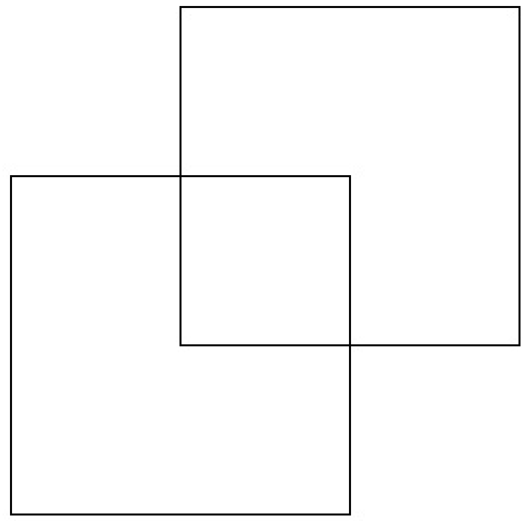
https://github.com/jannes-m/RQGIS

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



Want to learn more about `sf` and geocomputation with R

```r
vignette(package = "sf")
```

Want to learn more about `sf` and geocomputation with R

```
vignette(package = "sf")
```

And there's a new book to appear but you can already read it online:

http://robinlovelace.net/geocompr/

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] "C:/OSGeo4W64"
##
## $qgis_prefix_path
## [1] "C:/OSGeo4W64/apps/qgis-ltr"
##
## $python_plugins
## [1] "C:/OSGeo4W64/apps/qgis-ltr/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>
```

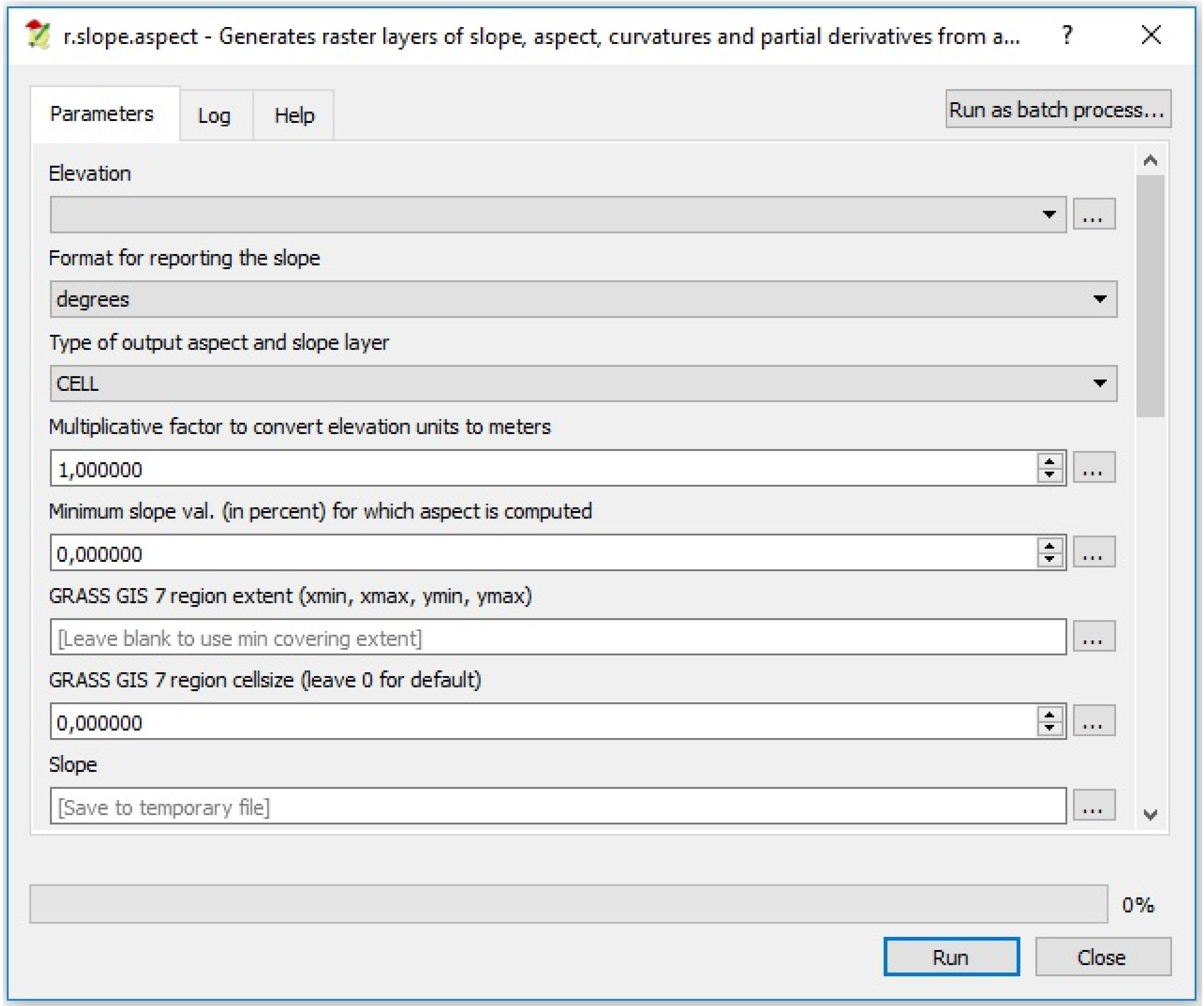
Here, we only have three function arguments, and automatic parameter collection is not necessary, but when I first looked at...

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

```
ALGORITHM: r.slope.aspect - Generates raster layers of slope, aspect, curva
    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...



r.slope.aspect - Generates raster layers of slope, aspect, curvatures and partial derivatives from a...

Elevation

Format for reporting the slope

degrees

Type of output aspect and slope layer

CELL

Multiplicative factor to convert elevation units to meters

1,000000

Minimum slope val. (in percent) for which aspect is computed

0,000000

GRASS GIS 7 region extent (xmin, xmax, ymin, ymax)

[Leave blank to use min covering extent]

GRASS GIS 7 region cellsize (leave 0 for default)

0,000000

Slope

[Save to temporary file]

Convenience function

get_args_man

```r
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.
```

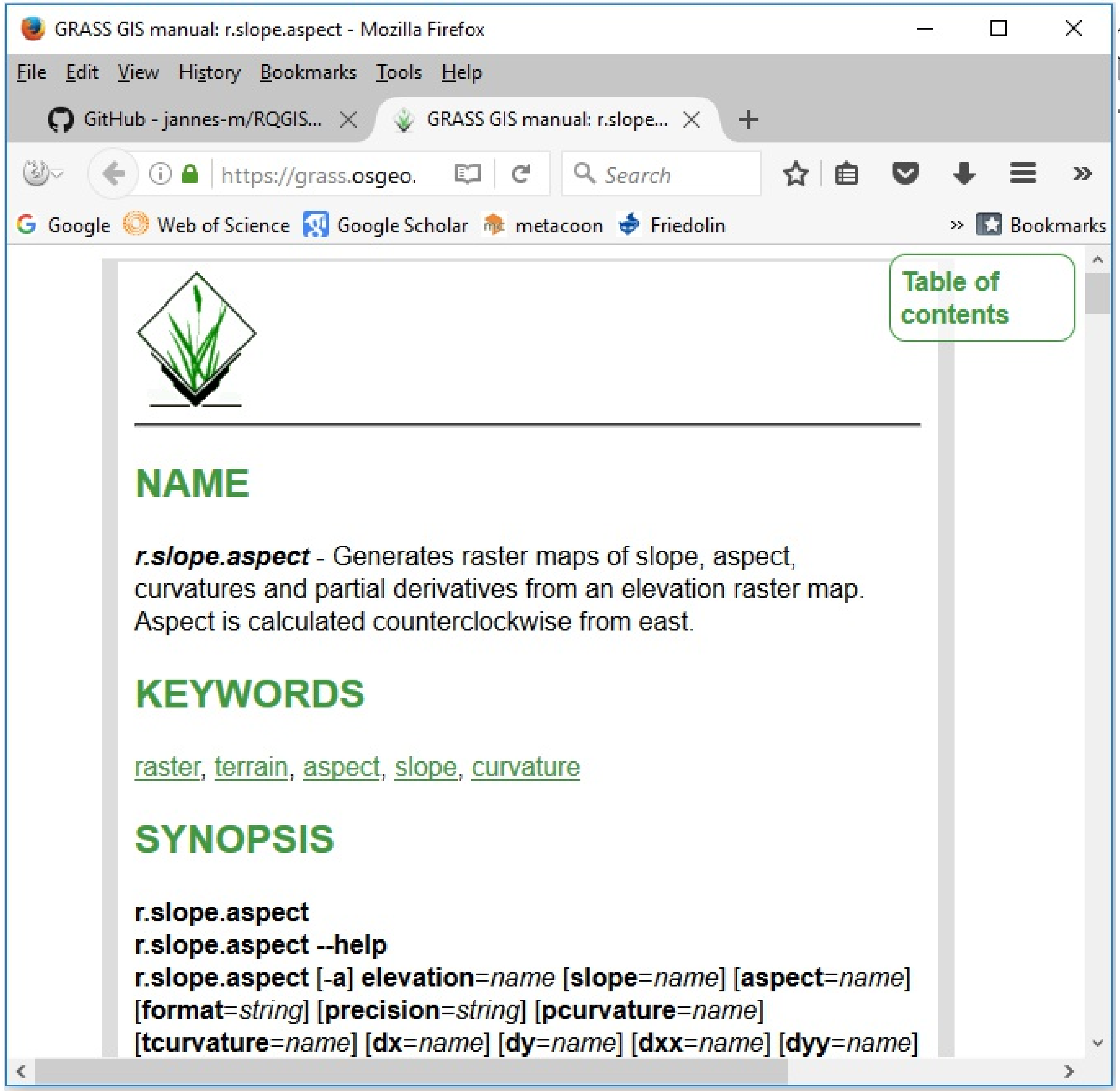
```
## $elevation
## [1] "None"
##
## $format
## [1] "0"
##
## $precision
## [1] "0"
##
## $`-a`
## [1] "True"
##
## $zscale
## [1] "1.0"
```

```
## $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:

```r
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

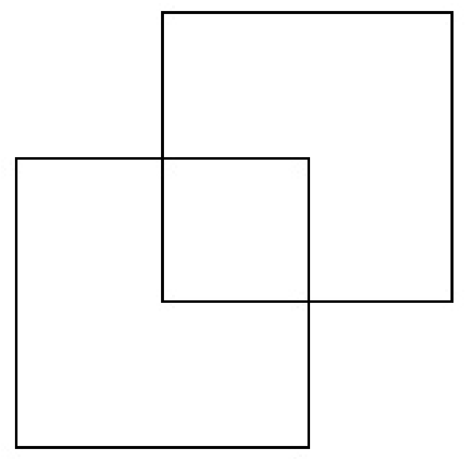
raster, terrain, aspect, slope, curvature

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

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

```
## $OUTPUT
## [1] "C:/Users/pi37pat/AppData/Local/Temp/Rtmpc1gfhf/out.shp"
```

Spatial objects as inputs

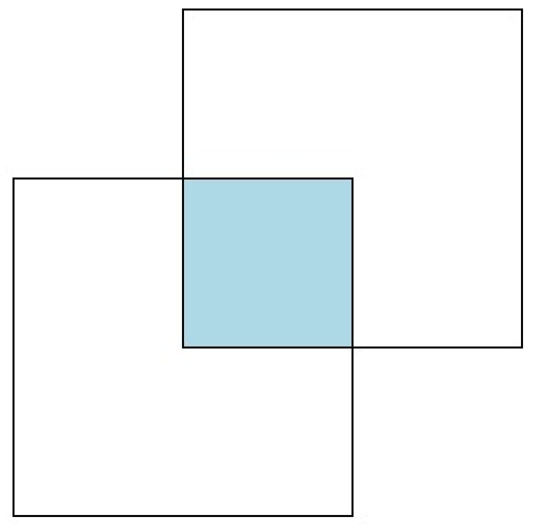
```
int = run_qgis("qgis:intersection",
               INPUT = poly_1,
               INPUT2 = poly_2,
               OUTPUT = file.path(tempdir(), "out.shp"),
               load_output = TRUE)
```

Load QGIS output into R

```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)
```



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.