# Interfacing QGIS spatial processing algorithms from R


Floris Vanderhaeghe
Dewey Dunnington
Nyall Dawson
Jan Caha
Jannes Muenchow
Jakub Nowosad
Robin Lovelace

html slides

useR! Salzburg 2024

Flanders State of the Art

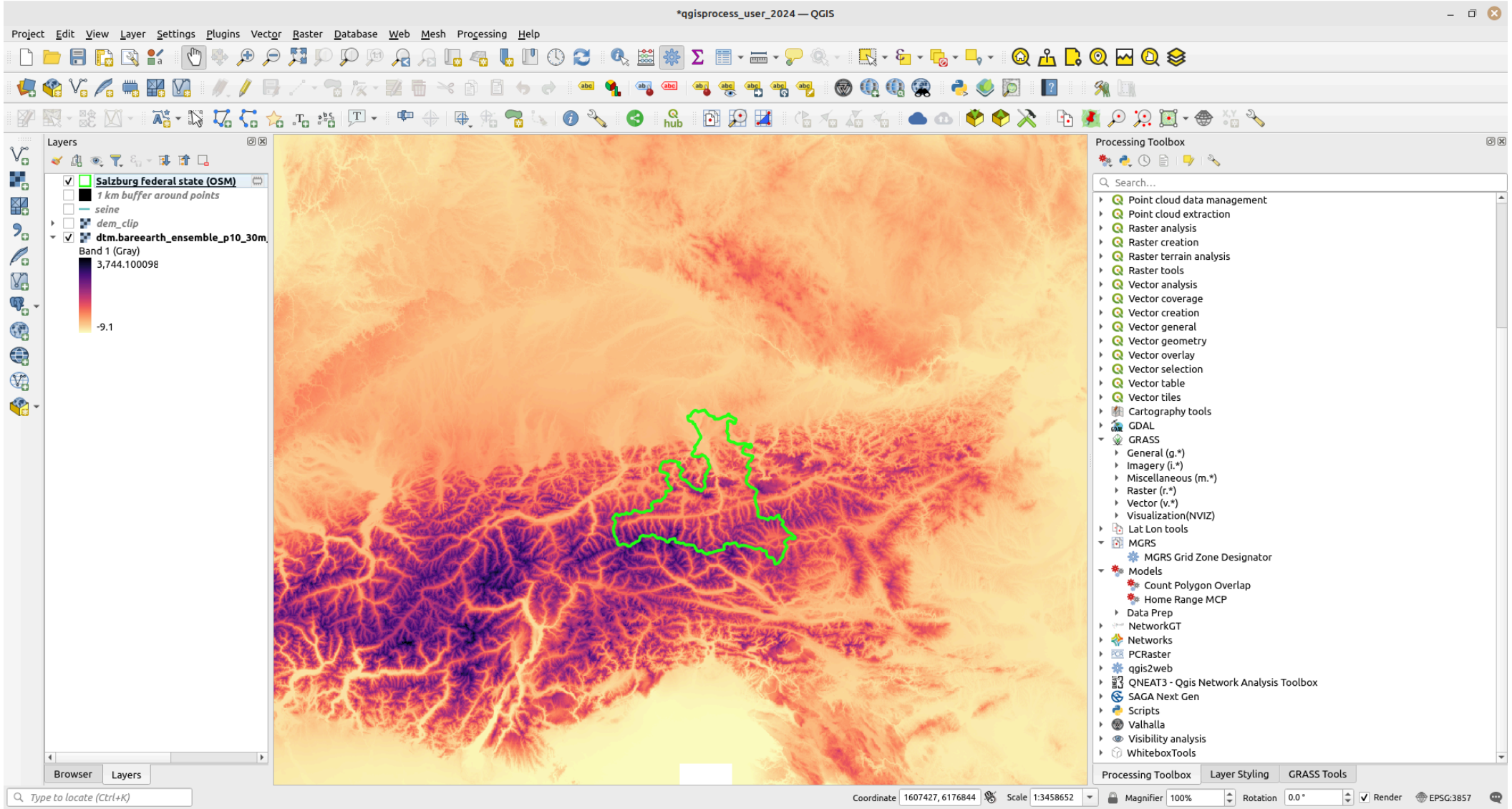
RESEARCH INSTITUTE NATURE AND FOREST

# About QGIS

# QGIS

Popular open-source geographic information system (GIS) program to:

- view geospatial data interactively
- create and edit spatial data layers
- perform advanced geoprocessing with algorithms of QGIS, GRASS GIS, GDAL ...
- make sophisticated maps, atlases and reports


*qgisprocess_user_2024 — QGIS
Project Edit View Layer Settings Plugins Vector Raster Database Web Mesh Processing Help
Layers
Salzburg federal state (OSM)
1 km buffer around points
seine
dem_clip
dtm.bareearth_ensemble_p10_30m
Band 1 (Gray)
3,744.100098
-9.1
Processing Toolbox
Search...
Point cloud data management
Point cloud extraction
Raster analysis
Raster creation
Raster terrain analysis
Raster tools
Vector analysis
Vector coverage
Vector creation
Vector general
Vector geometry
Vector overlay
Vector selection
Vector table
Vector tiles
Cartography tools
GDAL
GRASS
General (g.*)
Imagery (i.*)
Miscellaneous (m.*)
Raster (r.*)
Vector (v.*)
Visualization(NVIZ)
Lat Lon tools
MGRS
MGRS Grid Zone Designator
Models
Count Polygon Overlap
Home Range MCP
Data Prep
NetworkGT
Networks
PCRaster
qgis2web
QNEAT3 - Qgis Network Analysis Toolbox
SAGA Next Gen
Scripts
Valhalla
Visibility analysis
WhiteboxTools
Browser
Layers
Processing Toolbox
Layer Styling
GRASS Tools
Type to locate (Ctrl+K)
Coordinate 1607427, 6176844
Scale 1:3458652
Magnifier 100%
Rotation 0.0 °
Render
EPSG:3857

# Why interface QGIS processing from R?

# Main advantages

- Expand geospatial processing abilities in R
- Still have all other processing capabilities of R
- Reproducibility
- No QGIS project needed
- Unified interface to QGIS, GRASS GIS, SAGA, GDAL and other processing providers (1000+ geospatial algorithms)

# Interfacing QGIS processing from R

- Formerly: packages **RQGIS** and **RQGIS3** (Muenchow & Schratz) for QGIS 2 and 3
  - use the QGIS Python API
  - set QGIS environment variables
  - **RQGIS3** was hard to maintain wrt provider changes in QGIS; also, crashes were observed in RStudio IDE
  - no longer developed

# Interfacing QGIS processing from R

- Since QGIS 3.16: packages **qgisprocess** & **qgis**
  - use the recent standalone `qgis_process` shell command from QGIS = a unified entry to all providers and algorithms!
  - no more QGIS environment variables needed
  - actively developed

# Interfacing QGIS processing from R

```
$ qgis_process
QGIS Processing Executor - 3.38.0-Grenoble 'Grenoble' (3.38.0-Grenoble)
Usage: /usr/bin/qgis_process.bin [--help] [--version] [--json] [--verbose]
          [--no-python] [--skip-loading-plugins]
          [command] [algorithm id, path to model file, or path to Python script] [parameters]

...
```

```
$ qgis_process run <algorithm> [parameters]

$ qgis_process run <algorithm> -

$ qgis_process plugins
```

# Current R-packages

- **qgisprocess** (Dewey Dunnington et al.): direct interface to `qgis_process`
  - https://r-spatial.github.io/qgisprocess/

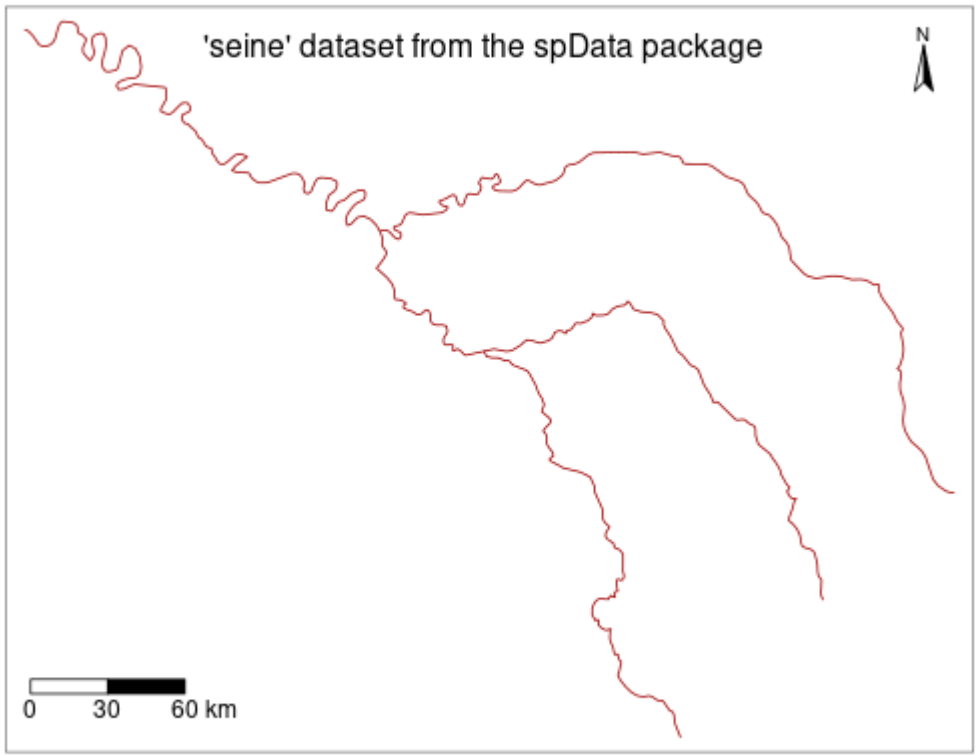
```
install.packages("qgisprocess")
```

- **qgis** (Jan Caha): functions for each algorithm; uses **qgisprocess** package
  - https://jancaha.github.io/r_package_qgis/

```
remotes::install_github("jancaha/qgis")
```

# A simple example

# Basic usage



# Basic usage

Load the R package:

```
library(qgisprocess)
```

```
#> Attempting to load the package cache ... Success!
#> QGIS version: 3.38.0-Grenoble
#> Having access to 2059 algorithms from 18 QGIS processing providers.
#> Run `qgis_configure(use_cached_data = TRUE)` to reload cache and get more details.
```

# Basic usage

```
seine_path <- "data/seine.gpkg"
```

```
(seine <- sf::read_sf(seine_path))
```

```
#> Simple feature collection with 3 features and 1 field
#> Geometry type: MULTILINESTRING
#> Dimension:     XY
#> Bounding box:  xmin: 518344.7 ymin: 6660431 xmax: 879955.3 ymax: 6938864
#> Projected CRS: RGF93 v1 / Lambert-93
#> # A tibble: 3 × 2
#>   name                                                          geom
#>   <chr>                                                <MULTILINESTRING [m]>
#> 1 Marne ((879955.3 6755725, 878440.9 6755688, 876653.8 6756227, 874212.2 675791…
#> 2 Seine ((828893.6 6713873, 828216.3 6715450, 827937.9 6716999, 828199.2 671851…
#> 3 Yonne ((773482.1 6660431, 771342.9 6665712, 771043 6667566, 770931.7 6669151,…
```

# Basic usage

Run algorithm:

```
result <- qgis_run_algorithm(
  algorithm = "native:pointsalonglines",
  INPUT = seine_path,
  DISTANCE = 1e4
)
#> Argument `START_OFFSET` is unspecified (using QGIS default value).
#> Argument `END_OFFSET` is unspecified (using QGIS default value).
#> Using `OUTPUT = qgis_tmp_vector()`
```

or:

```
result <- qgis::qgis_pointsalonglines(
  INPUT = seine_path,
  DISTANCE = 1e4
)
```

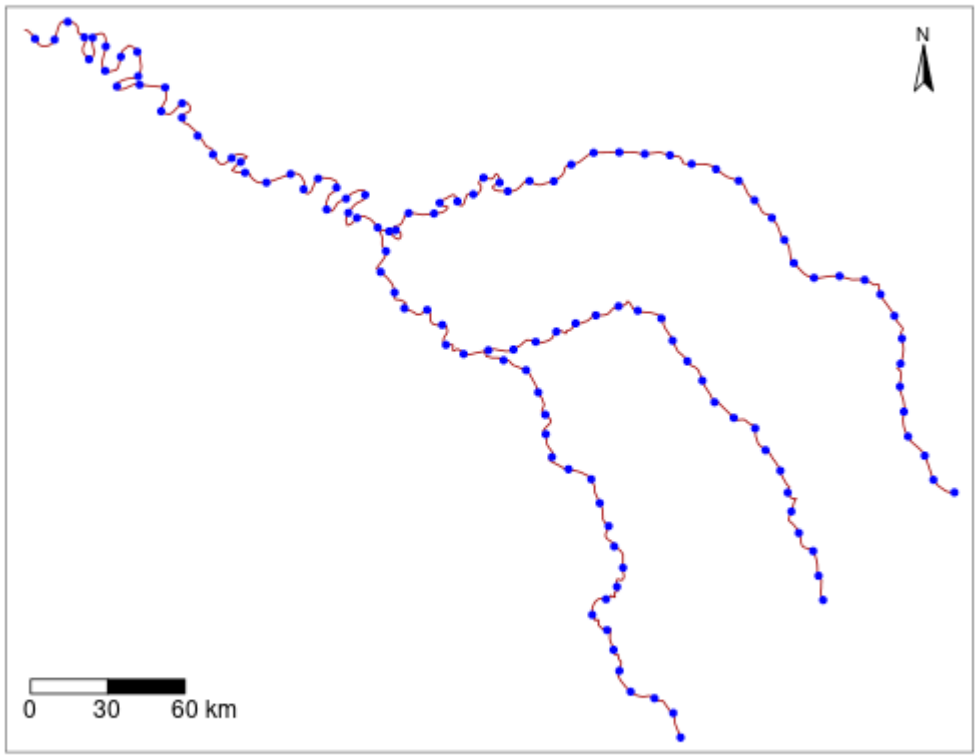
# Basic usage

Extract output (defaults to the `"OUTPUT"` element of `result`).

```
qgis_extract_output(result)
```

```
#> [1] "/tmp/RtmpUtuVpc/file5288578598f1/file52883dccf462.gpkg"
#> attr(,"class")
#> [1] "qgis_outputVector"
```

# Basic usage



# Convenient functions

# Finding algorithms

```
qgis_search_algorithms(algorithm = "point.*line")
```

| provider | algorithm | algorithm_title |
|---|---|---|
| gdal | gdal:pointsalonglines | Points along lines |
| native | native:interpolatepoint | Interpolate point on line |
| native | native:pointsalonglines | Points along geometry |
| native | native:randompointsonlines | Random points on lines |
| qgis | qgis:generatepointspixelcentroidsalongline | Generate points (pixel centroids) along line |
| qgis | qgis:randompointsalongline | Random points along line |
| sagang | sagang:convertpointstolines | Convert points to line(s) |
| sagang | sagang:pointtolinedistances | Point to line distances |
| sagang | sagang:snappointstolines | Snap points to lines |

# Finding algorithms

```
qgis_providers()
```

```
#> # A tibble: 18 × 3
#>    provider         provider_title                          algorithm_count
#>    <chr>            <chr>                                             <int>
#>  1 cartographytools Cartography tools                                   5
#>  2 gdal             GDAL                                               56
#>  3 grass            GRASS                                             307
#>  4 latlontools      Lat Lon tools                                      10
#>  5 model            Models                                              3
#>  6 NetworkGT        NetworkGT                                          33
#>  7 Networks         Networks                                           48
#>  8 pcraster         PCRaster                                          102
#>  9 qgis             QGIS                                               42
#> 10 3d               QGIS (3D)                                           1
#> 11 native           QGIS (native c++)                                 263
#> 12 pdal             QGIS (PDAL)                                        17
#> 13 qneat3           QNEAT3 - Qgis Network Analysis Toolbox             14
#> 14 sagang           SAGA Next Gen                                     587
#> 15 script           Scripts                                             1
#> 16 Valhalla         Valhalla                                           20
#> 17 visibility       Visibility analysis                                 4
#> 18 wbt              WhiteboxTools                                     546
```

# Finding algorithms

```
qgis_plugins()
```

```
#> # A tibble: 12 × 2
#>    name                    enabled
#>    <chr>                   <lgl>
#>  1 QNEAT3                  TRUE
#>  2 ViewshedAnalysis        TRUE
#>  3 cartography_tools       TRUE
#>  4 grassprovider           TRUE
#>  5 latlontools             TRUE
#>  6 network_gt              TRUE
#>  7 networks                TRUE
#>  8 pcraster_tools          TRUE
#>  9 processing              TRUE
#> 10 processing_saga_nextgen TRUE
#> 11 valhalla                TRUE
#> 12 wbt_for_qgis            TRUE
```

# Algorithm documentation

```
qgis_show_help("native:pointsalonglines")
```

```
## Points along geometry (native:pointsalonglines)
##
## ----------------
## Description
## ----------------
## Creates regularly spaced points along line features.
## This algorithm creates a points layer, with points distributed along the lines of an input vector layer. The dist
##
## Start and end offset distances can be defined, so the first and last point will not fall exactly on the line's fi
##
## ----------------
## Arguments
## ----------------
##
## INPUT: Input layer
##     Argument type:    source
##     Acceptable values:
##         - Path to a vector layer
## DISTANCE: Distance
##     Default value:    1
## .....
```

# Supports various R objects as input arguments

Spatial QGIS argument types

| input argument | R object |
|---|---|
| vector layer | **sf** or **terra** |
| raster layer | **stars**, **terra** and **raster** |
| multilayer | list (preferrably as `qgis_list_input()`) |
| extent | various 'bounding box' and 'extent' objects |
| crs | various CRS objects |

# Supports various R objects as input arguments

Non-spatial QGIS argument types:

| input argument | R object |
|---|---|
| expression | string |
| enum | integer or character |
| range | vector |
| matrix | matrix or dataframe |
| color | R color string |
| hierarchical types (e.g. aggregates) | nested list |
| ... | ... |

# Supports various R objects as input arguments

```
library(terra)
elev <- rast(system.file("ex/elev.tif", package = "terra"))
class(elev)
```

```
#> [1] "SpatRaster"
#> attr(,"package")
#> [1] "terra"
```

```
qgis_run_algorithm("native:rasterlayerstatistics", INPUT = elev, BAND = 1)
```

```
#> <Result of `qgis_run_algorithm("native:rasterlayerstatistics", ...)`>
#> List of 9
#>  $ COUNT           : num 4608
#>  $ MAX             : num 547
#>  $ MEAN            : num 348
#>  $ MIN             : num 141
#>  $ OUTPUT_HTML_FILE: 'qgis_outputHtml' chr "/tmp/RtmpUtuVpc/file5288578598f1/file52887ef8bb06"
#>  $ RANGE           : num 406
#>  $ STD_DEV         : num 80.2
#>  $ SUM             : num 1605135
#>  $ SUM_OF_SQUARES  : num 29646349
```

# Result handling

## Extracting elements from the result

By default, the `OUTPUT` element is selected by `qgis_extract_output()`.

- typically contains a file path

# Result handling

## Extracting elements from the result

Which output elements are generated by an algorithm?

```
qgis_get_output_specs("grass:r.flow")
```

```
#> # A tibble: 3 × 3
#>   name             description       qgis_output_type
#>   <chr>            <chr>             <chr>
#> 1 flowaccumulation Flow accumulation outputRaster
#> 2 flowlength       Flow path length  outputRaster
#> 3 flowline         Flow line         outputVector
```

So, sometimes you need to specify the output name:

```
qgis_extract_output(result, "flowline")
```

# Result handling

## Coercing output to R objects

Result object or output element can be coerced to an R object:

```
sf::st_as_sf(result) # takes OUTPUT by default
```

```
result |> qgis_extract_output("flowline") |> sf::st_as_sf()
```

```
qgis_as_terra(result)
```

```
qgis_as_raster(result)
```

```
stars::st_as_stars(result)
```

Extra!

Extra!

# Algorithm piping

```
seine_points_buffer <- seine |>
  qgis_run_algorithm_p("native:pointsalonglines", DISTANCE = 1e4) |>
  qgis_run_algorithm_p("native:buffer", DISTANCE = 1000,
                       OUTPUT = "data/buffer.gpkg")
seine_points_buffer
```

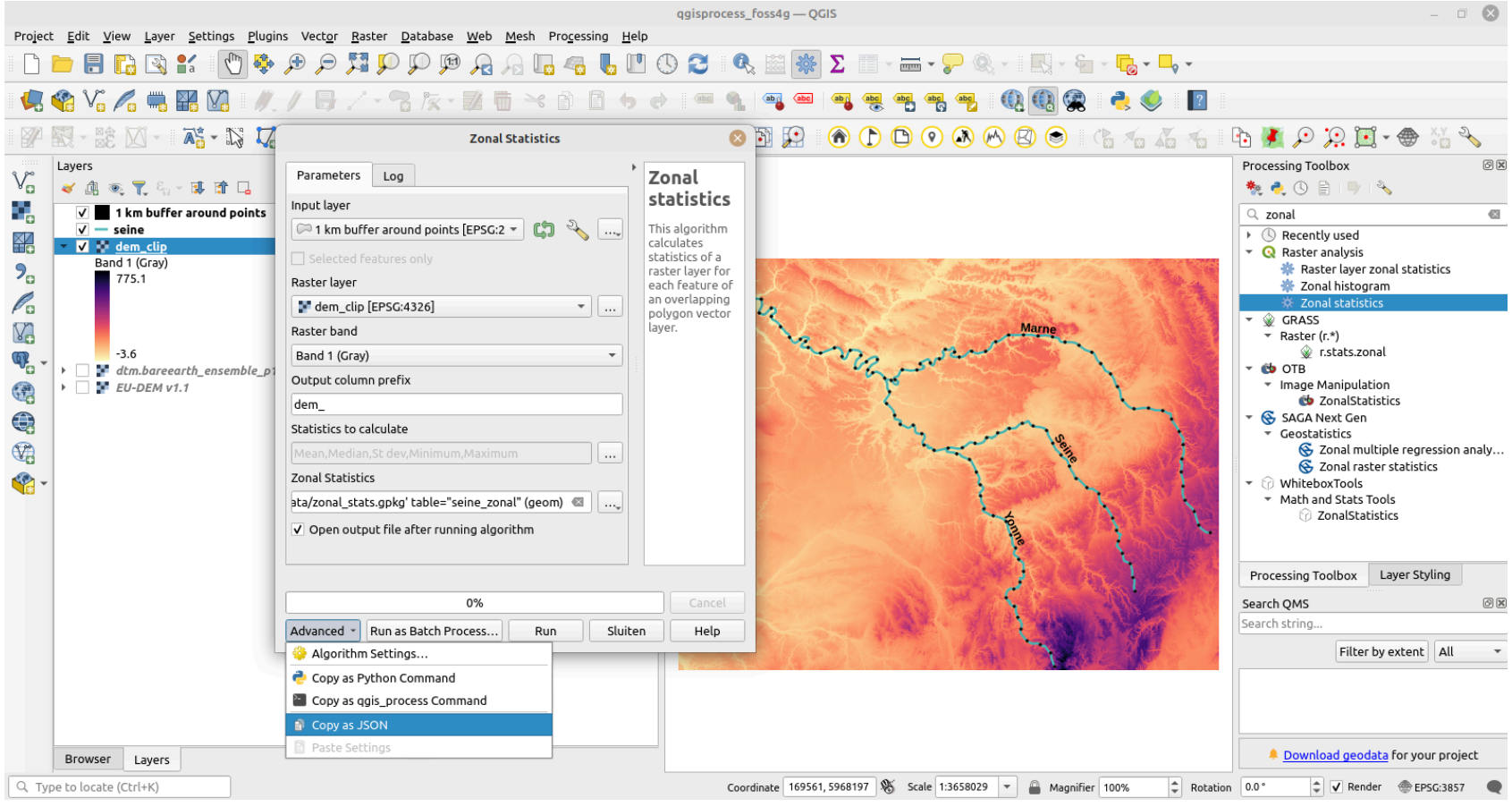
```
#> <Result of `qgis_run_algorithm("native:buffer", ...)`>
#> List of 1
#>  $ OUTPUT: 'qgis_outputVector' chr "data/buffer.gpkg"
```

Alternative with **qgis** package:

```
seine |>
  qgis::qgis_pointsalonglines(DISTANCE = 1e4) |>
  qgis_extract_output() |>
  qgis::qgis_buffer(DISTANCE = 1000)
```

```
#> <Result of `qgis_run_algorithm("native:buffer", ...)`>
#> List of 1
#>  $ OUTPUT: 'qgis_outputVector' chr "/tmp/RtmpUtuVpc/file5288578598f1/file528896a5b20.gpkg"
```

# Taking parameters from the QGIS GUI



# Taking parameters from the QGIS GUI

```
jsonlite::prettify(json_from_qgis)
```

```
#> {
#>     "area_units": "m2",
#>     "distance_units": "meters",
#>     "ellipsoid": "EPSG:7030",
#>     "inputs": {
#>         "COLUMN_PREFIX": "dem_",
#>         "INPUT": "/home/floris/git_repositories/foss4g-2023-qgisprocess/data/buffer.gpkg|layername=buffer",
#>         "INPUT_RASTER": "/home/floris/git_repositories/foss4g-2023-qgisprocess/data/dem_clip.tif",
#>         "OUTPUT": "ogr:dbname='/home/floris/git_repositories/foss4g-2023-qgisprocess/data/zonal_stats.gpkg' table
#>         "RASTER_BAND": 1,
#>         "STATISTICS": [
#>             2,
#>             3,
#>             4,
#>             5,
#>             6
#>         ]
#>     }
#> }
#>
```

# Taking parameters from the QGIS GUI

```
zonal_stats_result <- qgis_run_algorithm(
  "native:zonalstatisticsfb",
  .raw_json_input = json_from_qgis
)
```

```
zonal_stats_result
```

```
#> <Result of `qgis_run_algorithm("native:zonalstatisticsfb", ...)`>
#> List of 1
#>  $ OUTPUT: 'qgis_outputVector' chr "/home/floris/git_repositories/foss4g-2023-qgisprocess/data/zonal_stats.gpkg|l
```

# Result

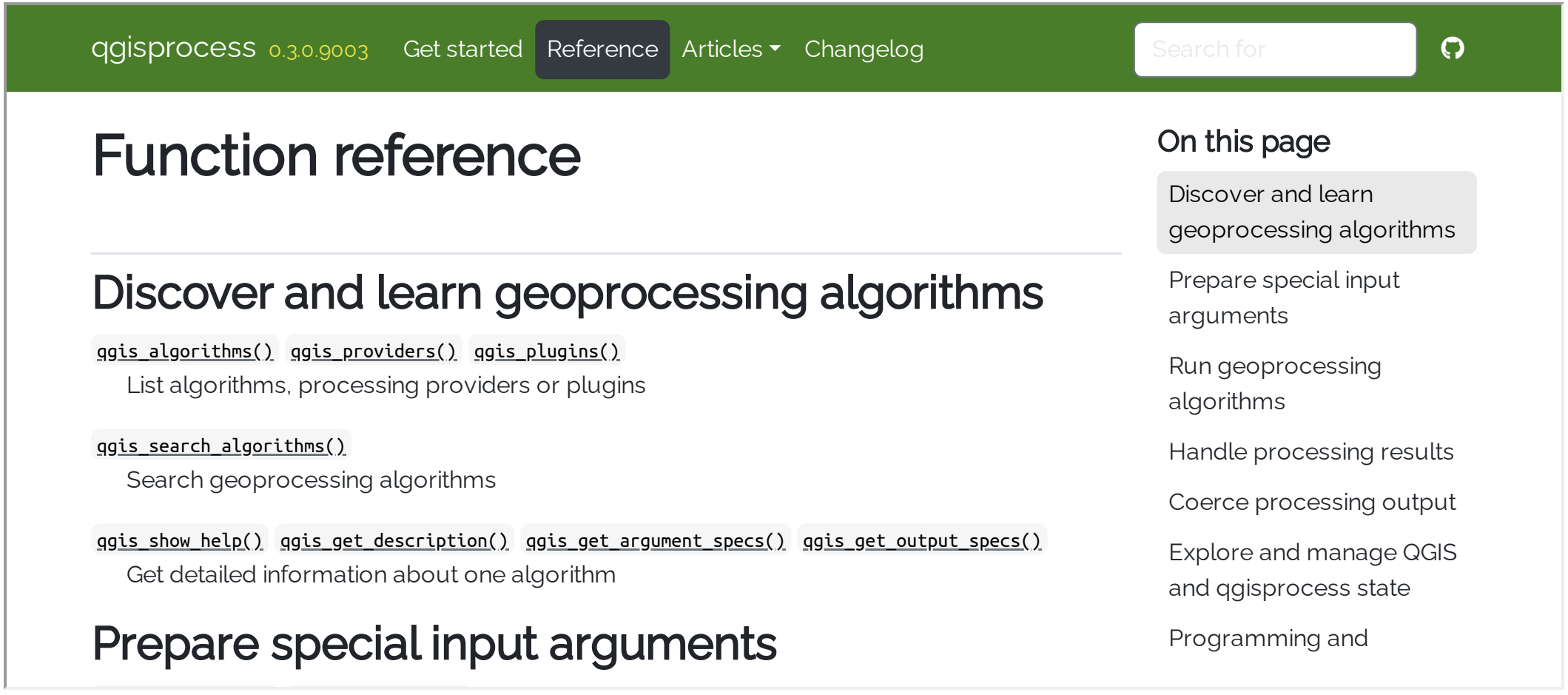
```
zonal_stats <- sf::st_as_sf(zonal_stats_result) |>
  sf::st_drop_geometry() |>
  mutate(distance = set_units(distance, "m") |> set_units("km"))

zonal_stats
```

```
#> # A tibble: 123 × 8
#>    name  distance  angle dem_mean dem_median dem_stdev dem_min dem_max
#>    <chr>     [km]  <dbl>    <dbl>      <dbl>     <dbl>   <dbl>   <dbl>
#>  1 Marne        0 269.       372.       366.      22.0    344.    412.
#>  2 Marne       10 322.       340.       338       21.9    309.    416.
#>  3 Marne       20 327.       331.       334.      23.9    287.    382.
#>  4 Marne       30 332.       296.       280.      29.5    264.    359.
#>  5 Marne       40   5.76     286.       295.      24.3    238.    317.
#>  6 Marne       50 358.       269.       254.      45.4    221.    378.
#>  7 Marne       60  22.4      235.       220.      32.7    201.    312.
#>  8 Marne       70 350.       202.       188.      24.8    183.    306.
#>  9 Marne       80 335.       196.       184.      26.9    167.    276
#> 10 Marne       90 334.       182.       184.      19.6    150.    209.
#> # ℹ 113 more rows
```

# Online documentation

# https://r-spatial.github.io/qgisprocess



# https://jancaha.github.io/r_package_qgis

## Reference

### QGIS

Algorithms provided directly by QGIS.

| | |
|---|---|
| `qgis_3d_tessellate()` | QGIS algorithm - Tessellate |
| `qgis_addautoincrementalfield()` | QGIS algorithm - Add autoincremental field |
| `qgis_addfieldtoattributestable()` | QGIS algorithm - Add field to attributes table |
| `qgis_adduniquevalueindexfield()` | QGIS algorithm - Add unique value index field |
| `qgis_addxyfields()` | QGIS algorithm - Add X/Y fields to layer |

Contents

# Cheat sheet !

# QGIS in R with qgisprocess :: CHEAT SHEET

## Mission

The goal of qgisprocess is to provide an R interface to the geoprocessing algorithms of QGIS, a popular and open source desktop geographic information system (GIS) program. 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 processing algorithms and some from GDAL, GRASS and many others (like SAGA).

| Providers | Algorithms |
|---|---|
| qgis | 50 + 242 ( c ++) + 1 (3D) |
| gdal | 56 |
| grass | 304 |
| third-party providers | x |
| Total counts | 653 + x |

```
# Show a tibble with processing providers
> qgis_providers( )
# Show a tibble with algorithms
> qgis_algorithms( )
# Search algorithms using regular expressions
> qgis_search_algorithms(
      algorithm = <x>,
      provider = <y>,
      group = <z>
      )
```

## Installation

```
> install.packages("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.

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

# Show all the parameters of the function
> qgis_get_argument_specs(algorithm = "native:
    creategrid")


# 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")


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

## Pipe integration

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

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

## Workflow

**Vector data**

```
# Hexagrid of 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**

```
# TWI processing
 > 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"))
```

Created by Antony Barja — Revised by Floris Vanderhaeghe and updated: 23/06/2023

# Section in book 'Geospatial Computation with R'

Geocomputation with R

Search

Table of contents

Welcome

Foreword (1st Edition)

Foreword (2nd Edition)

Preface

1 Introduction

Foundations

2 Geographic data in R

3 Attribute data operations

4 Spatial data operations

## 10.2 qgisprocess: a bridge to QGIS and beyond

QGIS is the most popular open-source GIS (Table 10.1; Graser and Olaya (2015)). QGIS provides a unified interface to QGIS's native geoalgorithms, GDAL, and — when they are installed — from other *providers* such as GRASS GIS, and SAGA (Graser and Olaya 2015). Since version 3.14 (released in summer 2020), QGIS ships with the `qgis_process` command line utility for accessing a bounty of functionality for geocomputation. `qgis_process` provides access to 300+ geoalgorithms in the standard QGIS installation and 1,000+ via plugins to external providers such as GRASS GIS and SAGA.

The **qgisprocess** package provides access to `qgis_process` from R. The package requires QGIS — and any other relevant plugins such as GRASS GIS and SAGA, used in this chapter — to be installed and available to the system. For installation instructions, see **qgisprocess**'s documentation.

Second Edition

Visit the geocompx website 🌐

Install updated packages

Open an issue ?

Chat on Discord

Check exercise solutions ✔

Support Ukraine 🇺🇦

On this page

10 Bridges to GIS software

https://r.geocompx.org

# Recent developments

# Recent developments

- Speed-up since QGIS 3.36 & **qgisprocess** 0.4.0
- Added vector support for R package **terra**
- Vignettes:
  - which R objects are accepted as algorithm arguments?
  - how to configure behaviour with options or environment variables?
  - how to pass QGIS expressions?
- Improved handling of deprecated algorithms

# Questions?

# Ideas?

Slides: https://florisvdh.github.io/user-2024-qgisprocess/