

# Package ‘RPyGeo’

May 7, 2026

**Type** Package

**Title** ArcGIS Geoprocessing via Python

**Version** 1.0.0

**Date** 2018-11-12

**Description** Provides access to ArcGIS geoprocessing tools by building an interface between R and the ArcPy Python side-package via the reticulate package.

**URL** <https://github.com/fapola/RPyGeo>

**BugReports** <https://github.com/fapola/RPyGeo/issues>

**License** GPL-3

**LazyData** TRUE

**Imports** reticulate (>= 1.2), sf, raster, tools, stringr, utils,  
rmarkdown, magrittr, stats, purrr

**SystemRequirements** Python (>= 2.6.0), ArcGIS (>= 10.0)

**RoxygenNote** 6.1.1

**Suggests** testthat, knitr, spData, rstudioapi, bookdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Alexander Brenning [aut, cre],  
Fabian Polakowski [aut],  
Marc Becker [aut],  
Jannes Muenchow [ctb] (ORCID: <<https://orcid.org/0000-0001-7834-4717>>)

**Maintainer** Alexander Brenning <[alexander.brenning@uni-jena.de](mailto:alexander.brenning@uni-jena.de)>

**Repository** CRAN

**Date/Publication** 2018-11-14 11:00:11 UTC

## Contents

RPyGeo-package . . . . .	2
rpygeo_build_env . . . . .	3
rpygeo_help . . . . .	4
rpygeo_load . . . . .	5
rpygeo_save . . . . .	7
rpygeo_search . . . . .	8
%rpygeo_-% . . . . .	9
%rpygeo_+% . . . . .	10
%rpygeo_!% . . . . .	11
%rpygeo_.*% . . . . .	12
<b>Index</b>	<b>14</b>

---

RPyGeo-package	<i>RPyGeo: ArcGIS Geoprocessing in R via Python</i>
----------------	---

---

### Description

Provide access to (virtually any) ArcGIS geoprocessing tool from within R by running Python geoprocessing without writing Python code or touching ArcGIS.

### Details

The package utilizes the ArcPy Python site-package or the ArcGIS API in order to access ArcGIS functionality. The function `rpygeo_build_env` can be applied to generate an ArcPy or arcgis object.

### Author(s)

**Maintainer:** Alexander Brenning <alexander.brenning@uni-jena.de>

Authors:

- Fabian Polakowski <fabian.polakowski@gmail.com>
- Marc Becker <marc.becker@uni-jena.de>

Other contributors:

- Jannes Muenchow (0000-0001-7834-4717) [contributor]

### See Also

Useful links:

- <https://github.com/fapola/RPyGeo>
- Report bugs at <https://github.com/fapola/RPyGeo/issues>

## Examples

```
# load the ArcPy module related to ArcGIS Pro (and save it as a R
# object called "arcpy_m") in R and also set the overwrite parameter
# to FALSE and add some extensions. Note that we do not have to set the path
# because the Python version is located in the default location
# (C:/Program Files/ArcGIS/Pro/bin/Python/envs/arcgispro-py3/)in this example.
## Not run: arcpy <- rpygeo_build_env(overwrite = TRUE,
                                     extensions = c("3d", "Spatial", "na"),
                                     pro = TRUE)

## End(Not run)
# Suppose we want to calculate the slope of a Digital Elevation Model.
# It is possible to get the description of any ArcPy function as a R list:
## Not run: py_function_docs("arcpy$Slope_3d")
# Now we can run our computation:
## Not run: arcpy$Slope_3d(arcpy$Slope_3d(in_raster = "dem.tif", out_raster = "slope.tif"))
```

---

rpygeo\_build\_env      *Initialize ArcPy site-package in R*

---

## Description

Initialises the Python ArcPy site-package in R via the reticulate package. Additionally environment settings and extensions are configured.

## Usage

```
rpygeo_build_env(path = NULL, overwrite = TRUE, extensions = NULL,
                 x64 = FALSE, pro = FALSE, arcgisAPI = FALSE, workspace = NULL,
                 scratch_workspace = NULL)
```

## Arguments

path	Full path to folder containing Python version which is linked to the ArcPy site-package. If left empty, the function looks for python.exe in the most likely location (C:/Python27/). It is also possible to provide a path to the ArcGIS API for Python here. In order to do so you need to provide the path to the python anaconda library were the arcgis package is installed. Additionally arcgisAPI must be set to true.
overwrite	If TRUE (default), existing ArcGIS datasets can be overwritten (does not work while using ArcGIS API for Python).
extensions	Optional character vector listing ArcGIS extension that should be enabled (does not work while using ArcGIS API for Python)
x64	Logical (default: FALSE). Determines if path search should look for 64 bit Python ArcPy version in default folder (C:/Python27)
pro	Logical (default: FALSE). If set to TRUE ' rpygeo_build_env tries to find Python version to use in the default ArcGIS Pro location (C:/Program Files/ArcGIS/Pro/bin/Python/envs/a

arcgisAPI	Logical (default: FALSE). Must be set to TRUE in order to use the ArcGIS API. This is the only option to work with the RPyGeo Package under a linux operation system.
workspace	Path of ArcGIS workspace in which to perform the geoprocessing (does not work while using ArcGIS API for Python).
scratch_workspace	Path to ArcGIS scratch workspace in which to store temporary files (does not work while using ArcGIS API for Python). If NULL a folder named scratch is created inside the workspace folder or on the same directory level as the workspace file geodatabase.

**Value**

Returns ArcPy or ArcGIS modules in R

**Author(s)**

Fabian Polakowski, Marc Becker

**Examples**

```
## Not run:
# Load ArcPy side-package of ArcGIS Pro with 3D and Spatial Analysis extension.
# Set environment setting 'overwrite' to TRUE.
# Note that no path parameter is necessary because Python is located in the
# default location.
arcpy <- rpygeo_build_env(overwrite = TRUE,
                        extensions = c("3d", "Spatial"),
                        pro = TRUE)

## End(Not run)

# Load the ArcPy module when your Python version is located in a different
# folder
```

---

rpygeo\_help

*Get help file for ArcPy function*

---

**Description**

This function opens the help file for ArcPy function in viewer panel or if not available in the browser.

**Usage**

```
rpygeo_help(arcpy_function)
```

**Arguments**

arcpy\_function ArcPy module with function or class

**Author(s)**

Marc Becker

**Examples**

```
## Not run:
# Load the ArcPy module and build environment
arcpy <- arcpy_build_env(overwrite = TRUE, workspace = tempdir())

# Open help file
rpygeo_help(arcpy$Slope_3d)

## End(Not run)
```

---

rpygeo\_load

*Load output of ArcPy functions into R session*

---

**Description**

This function loads the output of an ArcPy function into the R session. Raster files are loaded as raster objects and vector files as sf objects.

**Usage**

```
rpygeo_load(data)
```

**Arguments**

data            reticulate object or filename of the ArcPy function output

**Details**

Currently files and datasets stored in file geodatabases are supported.

Supported file formats:

- Tagged Image File Format (.tif)
- Erdas Imagine Images (.img)
- Esri Arc/Info Binary Grid (.adf)
- Esri ASCII Raster (.asc)
- Esri Shapefiles (.shp)

Supported datasets:

- Feature Class
- Raster Dataset

Esri has not released an API for raster datasets in file geodatabases. `rpygeo_load` converts a raster dataset to a temporary ASCII raster first and then loads it into the R session. Be aware that this can take a long time for large raster datasets.

This function can be used with the `%>%` operator from the `dplyr` package. The `%>%` operator forwards the `reticulate` object from the `ArcPy` function to `rpygeo_load` (s. Example 1). If used without the `%>%` operator an `reticulate` object can be specified for the `data` parameter (s. Example 2). It is also possible to use the filename of the `ArcPy` function output (s. Example 3). For Arc/Info Binary Grids the `data` parameter is just the name of the directory, which contains the `adf` files.

### Value

raster or sf object

### Author(s)

Marc Becker

### Examples

```
## Not run:
# Load packages
library(RPyGeo)
library(magrittr)
library(RQGIS)
library(spData)

# Get data
data(dem, package = "RQGIS")
data(nz, package = "spData")

# Write data to disk
writeRaster(dem, file.path(tempdir(), "dem.tif"), format = "GTiff")
st_write(nz, file.path(tempdir(), "nz.shp"))

# Load the ArcPy module and build environment
arcpy <- arcpy_build_env(overwrite = TRUE, workspace = tempdir())

# Create a slope raster and load it into the R session (Example 1)
slope <-
  arcpy$Slope_3d(in_raster = "dem.tif", out_raster = "slope.tif") %>%
  rpygeo_load()

# Create a aspect raster and load it into the R session (Example 2)
ras_aspect <- arcpy$sa$Aspect(in_raster = "dem.tif")
rpygeo_load(ras_aspect)

# Convert elevation raster to polygon shapefile and load it into R session (Example 3)
arcpy$RasterToPolygon_conversion("dem.tif", "elev.shp")
rpygeo_load("elev.shp")

## End(Not run)
```

---

`rpygeo_save`*Save temporary raster to workspace*

---

**Description**

This function saves temporary a raster as permanent raster to the workspace.

**Usage**

```
rpygeo_save(data, filename)
```

**Arguments**

<code>data</code>	reticulate object or full path of the ArcPy function output
<code>filename</code>	Filename with extension or without extension if the workspace is file geodatabase

**Details**

Some ArcPy functions have no parameter to specify an output raster. Instead they return a raster object and a temporary raster is saved to the scratch workspace. This functions writes the temporary raster as a permanent raster to the workspace.

How the file is written depends on the workspace and scratch workspace environment settings.

- Workspace and scratch workspace are directories: Raster is loaded with the raster package and is written to workspace directory. The file format is inferred from the file extension in the filename parameter.
- Workspace and scratch workspace are file geodatabases: Raster is copied to workspace file geodatabase. No file extension necessary for the filename parameter.
- Workspace is file geodatabase and scratch workspace is directory: Raster is copied to workspace file geodatabase. No file extension necessary for the filename parameter.
- Workspace is directory and scratch workspace is file geodatabase: Raster is exported to workspace directory. The filename parameter is ignored due to restrictions in `arcpy.RasterToOtherFormat_conversion` function. If the automatically generated filename already exists, a number is appended to the end of the filename.

**Author(s)**

Marc Becker

**Examples**

```
## Not run:  
# Load packages  
library(RPyGeo)  
library(RQGIS)  
library(magrittr)
```

```
# Get data
data(dem, package = "RQGIS")

# Load the ArcPy module and build environment
arcpy <- arcpy_build_env(overwrite = TRUE, workspace = tempdir())

# Write raster to workspace directory
writeRaster(dem, file.path(tempdir(), "dem.tif"), format = "GTiff")

# Calculate temporary aspect file and save to workspace
arcpy$sa$Aspect(in_raster = "dem.tif") %>%
  rpygeo_save("aspect.tif")

## End(Not run)
```

---

rpygeo\_search

*Search for ArcPy functions and classes*

---

## Description

Search for ArcPy functions and classes with a character string or regular expression.

## Usage

```
rpygeo_search(search_term = NULL)
```

## Arguments

search\_term      Search term. Regular expressions are possible.

## Details

The list members are referenced by the ArcPy module names. Each member contains a character vector of matching ArcPy functions and classes. Except for the main module, functions and classes have to be accessed by their module names (s. examples).

## Value

Named list of character vectors of matching ArcPy functions and classes

## Author(s)

Marc Becker

## Examples

```
## Not run:
# Load packages
library(RPyGeo)
library(magrittr)
library(RQGIS)

# Get data
data(dem, package = "RQGIS")

# Write data to disk
writeRaster(dem, file.path(tempdir(), "dem.tif"), format = "GTiff")

# Load the ArcPy module and build environment
arcpy <- rpygeo_build_env(overwrite = TRUE,
                          workspace = tempdir(),
                          extensions = "Spatial")

# Search for ArcPy functions, which contain the term slope
rpygeo_search("slope")

#> $toolbox
#> [1] "Slope_3d"      "SurfaceSlope_3d"
#>
#> $main
#> [1] "Slope_3d"      "SurfaceSlope_3d"
#>
#> $sa
#> [1] "Slope"
#>
#> $ddd
#> [1] "Slope"        "SurfaceSlope"
```

```
# Run function from sa module
arcpy$sa$Slope(in_raster="dem.tif")

# Run function from main module
arcpy$Slope_3d(in_raster="dem.tif")

## End(Not run)
```

---

`%rpygeo_-%`*Subtraction operator*

---

## Description

Subtraction operator for map algebra. Spatial Analyst extension is required for map algebra.

## Usage

```
raster_1 %rpygeo_-% raster_2
```

**Arguments**

raster_1	raster dataset or numeric
raster_2	raster dataset or numeric

**Value**

reticulate object

**Author(s)**

Marc Becker

**Examples**

```
## Not run:
# Load the ArcPy module and build environment
arcpy <- arcpy_build_env(overwrite = TRUE, workspace = "C:/workspace")

# Write raster to workspace directory
writeRaster(elev, "C:/workspace/elev.tif", extensions = "Spatial")

# Create raster object
ras <- arcpy$sa$Raster("elev.tif")

# Subtract raster from itself
ras %rpygeo_+% ras %>%
  rpygeo_load()

## End(Not run)
```

---

%rpygeo\_+%

*Addition operator*

---

**Description**

Addition operator for map algebra. Spatial Analyst extension is required for map algebra.

**Usage**

```
raster_1 %rpygeo_+% raster_2
```

**Arguments**

raster_1	raster dataset or numeric
raster_2	raster dataset or numeric

**Value**

reticulate object

**Author(s)**

Marc Becker

**Examples**

```
## Not run:  
# Load the ArcPy module and build environment  
arcpy <- arcpy_build_env(overwrite = TRUE, workspace = "C:/workspace")  
  
# Write raster to workspace directory  
writeRaster(elev, "C:/workspace/elev.tif", extensions = "Spatial")  
  
# Create raster object  
ras <- arcpy$sa$Raster("elev.tif")  
  
# Add raster to itself  
ras %rpygeo_+% ras %>%  
  rpygeo_load()  
  
## End(Not run)
```

---

%rpygeo\_/%

*Division operator*

---

**Description**

Division operator for map algebra. Spatial Analyst extension is required for map algebra.

**Usage**

```
raster_1 %rpygeo_/% raster_2
```

**Arguments**

raster_1	raster dataset or numeric
raster_2	raster dataset or numeric

**Value**

reticulate object

**Author(s)**

Marc Becker

**Examples**

```
## Not run:
# Load the ArcPy module and build environment
arcpy <- arcpy_build_env(overwrite = TRUE, workspace = "C:/workspace")

# Write raster to workspace directory
writeRaster(elev, "C:/workspace/elev.tif", extensions = "Spatial")

# Create raster object
ras <- arcpy$sa$Raster("elev.tif")

# Divide raster by itself
ras %rpygeo_+% ras %>%
  rpygeo_load()

## End(Not run)
```

---

*%rpygeo\_\***Multiplication operator*

---

**Description**

Multiplication operator for map algebra. Spatial Analyst extension is required for map algebra.

**Usage**

```
raster_1 %rpygeo_+% raster_2
```

**Arguments**

```
raster_1      raster dataset or numeric
raster_2      raster dataset or numeric
```

**Value**

reticulate object

**Author(s)**

Marc Becker

**Examples**

```
## Not run:
# Load the ArcPy module and build environment
arcpy <- arcpy_build_env(overwrite = TRUE, workspace = "C:/workspace")

# Write raster to workspace directory
writeRaster(elev, "C:/workspace/elev.tif", extensions = "Spatial")
```

`%rpygeo_*`

13

```
# Create raster object
ras <- arcpy$sa$Raster("elev.tif")

# Multiply raster to itself
ras %rpygeo_+% ras %>%
  rpygeo_load()

## End(Not run)
```

# Index

`%rpygeo_*%`, [12](#)

`%rpygeo_+%`, [10](#)

`%rpygeo_-%`, [9](#)

`%rpygeo_/%`, [11](#)

RPyGeo (RPyGeo-package), [2](#)

RPyGeo-package, [2](#)

`rpygeo_build_env`, [2, 3](#)

`rpygeo_help`, [4](#)

`rpygeo_load`, [5](#)

`rpygeo_save`, [7](#)

`rpygeo_search`, [8](#)