

The RPyGeo Package

May 5, 2008

Type Package

Title ArcGIS Geoprocessing in R via Python

Version 0.9-0

Date 2007-09-18

Author Alexander Brenning

Maintainer Alexander Brenning <brenning@uwaterloo.ca>

Description Provide access to (virtually any) ArcGIS Geoprocessing tool from within R by running Python geoprocessing scripts without writing Python code or touching ArcGIS. Requires ArcGIS >=9.2, a suitable version of Python (currently 2.4), and Windows.

License GPL

SystemRequirements Windows, Python 2.4.0, ArcGIS (>= 9.2)

OS_type windows

R topics documented:

RPyGeo-package	1
ArcGIS Geoprocessing Tools	2
rpygeo.build.env	4
rpygeo.geoprocessor	5
rpygeo.required.extensions	8

Index	10
--------------	-----------

RPyGeo-package *ArcGIS Geoprocessing in R via Python*

Description

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

Details

Package: RPyGeo
 Type: Package
 Version: 0.9-0
 Date: 2007-09-18
 License: GPL

The function `rpygeo.geoprocessor` is the core function of this package. It creates and runs a Python script that executes your ArcGIS/Python geoprocessing command from within R. This function can be used to define more convenient wrappers for frequently used geoprocessing tools. Some are already implemented, for example `rpygeo.Slope.sa` and `rpygeo.EucDistance.sa`, more are to be added in future releases.

Author(s)

Alexander Brenning <brenning@uwaterloo.ca>

Examples

```

## Not run:
rpygeo.geoprocessor("Slope_sa('dem','slope')",
  "Aspect_sa('dem','aspect')",
  "Hillshade_sa('dem','hshd')")
## End(Not run)

## Not run: rpygeo.Slope.sa("dem","slope")

```

ArcGIS Geoprocessing Tools
Wrappers for selected ArcGIS functions

Description

Wrappers for a small selection of ArcGIS geoprocessing functions based on the `rpygeo.geoprocessor`.

Usage

```

rpygeo.EucDistance.sa(in.data, out.raster, maxdist = NULL,
  cellsize = NULL, out.direction.raster = NULL,
  env = rpygeo.env, ...)
rpygeo.Hillshade.sa(in.raster, out.raster, azimuth = 315,
  altitude = 45, model.shadows = c("NO_SHADOWS", "SHADOWS"),
  z.factor = 1, ...)
rpygeo.Slope.sa(in.raster, out.raster,
  unit = c("DEGREE", "PERCENT_RISE"), z.factor = 1, ...)
rpygeo.Aspect.sa(in.raster, out.raster, ...)
rpygeo.Delete.management(in.data, data.type = NULL, ...)

```

Arguments

```

in.raster, in.data, out.raster
    Names of ArcGIS raster or vector datasets or feature classes in a geodatabase
    (relative to the current workspace defined in a rpygeo.env environment).
    Shapefiles must include the extension “.shp”.

env
    A list defining an RPyGeo working environment as built by rpygeo.build.env.

maxdist, cellsize, out.direction.raster,

azimuth, altitude, model.shadows, z.factor,

unit, data.type
    Arguments to be passed to the Python geoprocessing function. See ArcGIS help
    files for information on the usage of scripting commands and their arguments.

...
    Additional arguments to be passed to rpygeo.geoprocessor.

```

Details

These functions simply try to replicate the behaviour of the ArcGIS/Python geoprocessing functions of the same name. See `rpygeo.geoprocessor` for details on what happens behind the scenes.

Value

The function return NULL if no error occurred, otherwise a character vector containing the error message.

Author(s)

Alexander Brenning

See Also

[rpygeo.geoprocessor](#), [rpygeo.build.env](#)

Examples

```

# Allow ArcGIS to overwrite existing datasets:
## Not run: rpygeo.env$overwriteoutput = 1
# Calculate the slope of a DEM raster dataset
# in the current ArcGIS workspace:
## Not run: rpygeo.geoprocessor("Slope_sa",c("dem","slope"))
# Same:
## Not run: rpygeo.geoprocessor("Slope_sa('dem','slope')")
# Same, using the more convenient wrapper:
## Not run: rpygeo.Slope.sa("dem","slope")

# Three at a time or separately:
## Not run: date()
## Not run:
rpygeo.geoprocessor("Slope_sa('dem','slope')",
  "Aspect_sa('dem','aspect')", "Hillshade_sa('dem','hshd')")
## End(Not run)
## Not run: date() # ~20 sec on my computer
## Not run: rpygeo.Slope.sa("dem","slope")
## Not run: rpygeo.Aspect.sa("dem","aspect")

```

```

## Not run: rpygeo.Hillshade.sa("dem", "hshd")
## Not run: date() # ~50 sec
## Not run: rpygeo.Delete.management("slope")
## Not run: rpygeo.Delete.management("aspect")
## Not run: rpygeo.Delete.management("hshd")

# Calculate the Euclidian distance from railway lines
# up to a max. distance of 1000 map units:
## Not run:
rpygeo.geoprocessor("EucDistance_sa",
  args=list("rail.shp", "raildist", 1000))
## End(Not run)
# Same:
## Not run: rpygeo.EucDistance.sa("rail.shp", "raildist", maxdist=1000)

# Use MapAlgebra to calculate a distance-decay function:
## Not run:
rpygeo.geoprocessor("SingleOutputMapAlgebra_sa",
  args=c("exp( raildist / -100 )", "distdecay"))
## End(Not run)

# Or why not in just one step if you like MapAlgebra:
## Not run:
rpygeo.geoprocessor( "SingleOutputMapAlgebra_sa",
  args=c("exp( EucDistance( rail.shp, \#, \#, 1000 ) / -100 )", "distdecay") )
## End(Not run)

```

rpygeo.build.env *RPyGeo Geoprocessing Environments*

Description

Set up a geoprocessing environment for ArcGIS/Python scripting

Usage

```

rpygeo.build.env(modules = "arcgisscripting",
  init = "gp = arcgisscripting.create()",
  workspace = NULL, cellsize = NULL, extent = NULL,
  mask = NULL, overwriteoutput = 0, extensions = NULL,
  python.path = "C:\software\Python24",
  python.command = "python.exe")
rpygeo.env

```

Arguments

modules	(Do not modify!) Name of Python module for ArcGIS geoprocessing.
init	(Do not modify!) Python code for initializing the Python geoprocessor.
workspace	Path of ArcGIS workspace (or name of geodatabase) in which to perform the geoprocessing.
cellsize	Default cellsize (default: maximum(?) of inputs - see ArcGIS documentation).

extent, mask Datasets or character strings defining the analysis extent and mask - see ArcGIS documentation.

overwriteoutput Overwrite existing ArcGIS datasets (=1) or not (=0 - default)?

extensions Names of extensions to be used in geoprocessing; it is usually not necessary to specify this here. Possible values: "Spatial", "3d", "geostats", "network", "datainteroperability".

python.path Where to find the Python interpreter (depends on Python version).

python.command Name of the Python command line interpreter executable.

Details

See ArcGIS documentation. This geoprocessing environment reflects only a small fraction of the ArcGIS environment settings. Future releases of this package may include more than the properties listed above.

Value

A list whose components are exactly the arguments passed to the `rpygeo.build.env` function.

Author(s)

Alexander Brenning

See Also

[rpygeo.geoprocessor](#)

Examples

```
# Everything in this workspace will be masked with DEM extent
# and have a cellsize of 100m:
## Not run: env.lo <- rpygeo.build.env( mask="clip", cellsize=100 )
# and this is for high-resolution output:
## Not run: env.hi <- rpygeo.build.env( mask="clip", cellsize=1 )

# Slope from different DEMs at different target resolutions
# (which may be different from the original DEM resolution):
## Not run: rpygeo.Slope.sa("srtm-dem", "slope-lo", env=env.lo)
## Not run: rpygeo.Slope.sa("laser-dem", "slope-hi", env=env.hi)
```

rpygeo.geoprocessor

ArcGIS Geoprocessor Workhorse

Description

This function creates a Python geoprocessing script file and runs it from the operating system using the ArcGIS Geoprocessor.

Usage

```
rpygeo.geoprocessor(fun, args = NULL, py.file = "rpygeo.py",
  msg.file = "rpygeo.msg", env = rpygeo.env, extensions = NULL,
  working.directory = getwd(), quote.args = TRUE, add.gp = TRUE,
  wait = TRUE, clean.up = wait, detect.required.extensions = TRUE)
```

Arguments

<code>fun</code>	This can be either a complete Python geoprocessing command (see examples), a single geoprocessing function name, or a vector of function or Python expressions to be evaluated by the Python geoprocessor.
<code>args</code>	Vector or list of arguments to be passed to the function listed in <code>fun</code> . The argument <code>quote.args</code> determines whether these arguments will be decorated with quotation marks.
<code>py.file</code>	Name of the temporary Python script file (in the <code>working.directory</code>).
<code>msg.file</code>	Name of the temporary file in which to dump Python/ArcGIS error messages (in the <code>working.directory</code>).
<code>env</code>	A list defining the RPyGeo working environment. Defaults to the standard working environment <code>rpygeo.env</code> , which is created at start-up. See rpygeo.build.env for details.
<code>extensions</code>	Optional character vector listing ArcGIS extension that should be enabled before using the geoprocessing function. This adds to any extensions that are listed in the environment or eventually detected by <code>rpygeo.required.extensions</code> .
<code>working.directory</code>	The working directory for temporary files (i.e. the Python script and error message files); defaults to R's current working directory.
<code>quote.args</code>	Logical value (default: <code>TRUE</code>) or logical vector that determines whether quotation marks have to be added to the <code>args</code> arguments before passing them to Python. If this is a vector, it must have the same length as <code>args</code> . See Details.
<code>add.gp</code>	Logical (default: <code>TRUE</code>). See Details.
<code>wait</code>	Logical (default: <code>TRUE</code>). Experimental(!) option. If <code>FALSE</code> (NOT recommended), do not wait for the operating system / ArcGIS to finish the Python geoprocessing script.
<code>clean.up</code>	Logical (default <code>TRUE</code>) or character vector (" <code>msg</code> ", " <code>py</code> ", or <code>c("msg", "py")</code>). Determines whether the error message file, the Python script file, or both (default) should be deleted after geoprocessing is finished. Ignored if <code>wait</code> is <code>FALSE</code> .
<code>detect.required.extensions</code>	Logical (default: <code>TRUE</code>). Determines whether <code>rpygeo.required.extensions</code> should try to find out which ArcGIS extensions are required to evaluate the function(s).

Details

This function is the R geoprocessing workhorse that creates a Python geoprocessing script, runs it, and returns any error messages.

If `fun` is a ready-to-use Python expression such as `gp`, then `add.gp` only determines whether the `"gp."` has to be added as a prefix to access the Python geoprocessor or not.

In most cases however, `fun` will be a single ArcGIS geoprocessing script function such as `"Slope_sa"`, where `"_sa"` tells us that this function can be found in the Spatial Analyst extension of ArcGIS (`rpygeo.required.extensions` will check this for you if the `detected...` argument is `TRUE`) Now `args` will be a vector or list of arguments to `Slope_sa`, e.g. `c("dem", "slope")` or `list("dem", "slope", "PERCENT_RISE", 2)` (see ArcGIS help files for information on the arguments of `Slope_sa`). These will result in Python expressions `gp.Slope_sa("dem", "slope")` and `gp.Slope_sa("dem", "slope", "PERCENT_RISE", 2)` if `add.gp==TRUE` and if we use the `quote.args` arguments `TRUE` and `c(T, T, T, F)`, respectively.

Dataset names will always be relative to the path or geodatabase defined in the geoprocessing environment settings `env$workspace`. Also, ArcGIS will be allowed to overwrite any existing output files (`env$overwriteoutput==1`) or not (`==0`). See [rpygeo.build.env](#) for details.

Value

The function returns `NULL` if it was successful, or otherwise a character vector with the ArcGIS error message. In addition, the ArcGIS function will generate the output described in the ArcGIS help files etc. Depending on the `clean.up` argument, the Python code may still be available in the `py.file`, and error messages in `msg.file`.

Note

The Python script created by this geoprocessor is loaded with initialization code for setting up the ArcGIS workspace and enabling ArcGIS extensions. This makes this function pretty inefficient, but you save a lot of time because you don't have to switch between three applications and two programming languages...

ArcGIS is pretty flexible with respect to numeric arguments such as the z factor in `Slope_sa` being passed as character string. As a consequence, `quote.args=TRUE` will normally work fine. `wait==FALSE` is experimental and not recommended. Watch for file name conflicts if you really want to try it - competing geoprocessing scripts must use different temporary Python script files etc.

Author(s)

Alexander Brenning

See Also

[rpygeo.build.env](#)

Examples

```
# Allow ArcGIS to overwrite existing datasets:
## Not run: rpygeo.env$overwriteoutput = 1
# Calculate the slope of a DEM raster dataset
# in the current ArcGIS workspace:
## Not run: rpygeo.geoprocessor("Slope_sa", c("dem", "slope"))
# Same:
## Not run: rpygeo.geoprocessor("Slope_sa('dem', 'slope')")
# Same, using the more convenient wrapper:
## Not run: rpygeo.Slope.sa("dem", "slope")

# Three at a time or separately:
## Not run: date()
## Not run:
rpygeo.geoprocessor("Slope_sa('dem', 'slope')",
```

```

    "Aspect_sa('dem','aspect')", "Hillshade_sa('dem','hshd')")
## End(Not run)
## Not run: date() # ~20 sec on my computer
## Not run: rpygeo.Slope.sa("dem","slope")
## Not run: rpygeo.Aspect.sa("dem","aspect")
## Not run: rpygeo.Hillshade.sa("dem","hshd")
## Not run: date() # ~50 sec
## Not run: rpygeo.Delete.management("slope")
## Not run: rpygeo.Delete.management("aspect")
## Not run: rpygeo.Delete.management("hshd")

# Calculate the Euclidian distance from railway lines
# up to a max. distance of 1000 map units:
## Not run:
rpygeo.geoprocessor("EucDistance_sa",
  args=list("rail.shp","raildist",1000))
## End(Not run)
# Same:
## Not run: rpygeo.EucDistance.sa("rail.shp","raildist",maxdist=1000)

# Use MapAlgebra to calculate a distance-decay function:
## Not run:
rpygeo.geoprocessor("SingleOutputMapAlgebra_sa",
  args=c("exp( raildist / -100 )","distdecay"))
## End(Not run)

# Or why not in just one step if you like MapAlgebra:
## Not run:
rpygeo.geoprocessor( "SingleOutputMapAlgebra_sa",
  args=c("exp( EucDistance( rail.shp, \#, \#, 1000 ) / -100 )","distdecay") )
## End(Not run)

```

```
rpygeo.required.extensions
```

Check required ArcGIS extensions

Description

Internal function that checks which ArcGIS extensions have to be enabled to evaluate a Python expression.

Usage

```
rpygeo.required.extensions(expr)
```

Arguments

expr	A vector or list of character strings with Python geoprocessing expressions or function names.
------	--

Value

Returns a character vector with the ArcGIS extension names (currently e.g. "Spatial", "3d", "geostats", "network", and/or "datainteroperability").

Note

This internal function is used by `rpygeo.geoprocessor`.

Author(s)

Alexander Brenning

See Also

[rpygeo.geoprocessor](#)

Index

*Topic **database**

- ArcGIS Geoprocessing Tools, [2](#)
- RPyGeo-package, [1](#)
- rpygeo.build.env, [4](#)
- rpygeo.geoprocessor, [5](#)
- rpygeo.required.extensions, [8](#)

*Topic **interface**

- ArcGIS Geoprocessing Tools, [2](#)
- RPyGeo-package, [1](#)
- rpygeo.build.env, [4](#)
- rpygeo.geoprocessor, [5](#)
- rpygeo.required.extensions, [8](#)

*Topic **package**

- RPyGeo-package, [1](#)

ArcGIS Geoprocessing Tools, [2](#)

RPyGeo (*RPyGeo-package*), [1](#)

RPyGeo-package, [1](#)

rpygeo.Aspect.sa (*ArcGIS Geoprocessing Tools*), [2](#)

rpygeo.build.env, [3](#), [4](#), [6](#), [7](#)

rpygeo.Delete.management (*ArcGIS Geoprocessing Tools*), [2](#)

rpygeo.env (*rpygeo.build.env*), [4](#)

rpygeo.EucDistance.sa (*ArcGIS Geoprocessing Tools*), [2](#)

rpygeo.geoprocessor, [3](#), [5](#), [5](#), [9](#)

rpygeo.Hillshade.sa (*ArcGIS Geoprocessing Tools*), [2](#)

rpygeo.required.extensions, [8](#)

rpygeo.Slope.sa (*ArcGIS Geoprocessing Tools*), [2](#)