GIS for Economists 2

Giorgio Chiovelli   Sebastian Hohmann

16/03/2020

Table of contents

- Overview
  - The plan for the session
- A first simple geoprocessing example
  - Automation using the Graphical Processing Modeler
- Introduction to Python
  - Python sales pitch
  - Installing Python
  - Language peculiarities
  - Opening and closing python
- From Model Builder to Python
  - Saving the model as python script
  - Cleaning up the python script
  - A second example

Overview

The plan for today

A first simple geoprocessing example

- Automation using the Graphical Processing Modeler
- Example: average agricultural productivity in U.S. counties

Introduction to Python

- Installing python
- Language peculiarities
- Opening and closing python
- Interactive mode and normal mode

Simple GIS automation in python

- Example: average agricultural productivity in U.S. counties
- Example: Area of every country in the world

Graphical Processing Modeler

What is it good for?

Making life easier

- Geoprocessing tools have many options, often need to be executed in succession.
- Clearly, we want to automate Geoprocessing somehow to make it faster.

Making research replicable

- Geoprocessing can become complex quickly.
- To make research replicable (also for ourselves!) we want to automate as much as possible of the GIS workflow.

An improvement but not the last word

- The Graphical Modeler is a first (though not ideal) step towards automation.
- It also functions as a nice bridge towards full automation in Python (see below).

Simple example: build a model to calculate average agricultural suitability of all U.S. counties

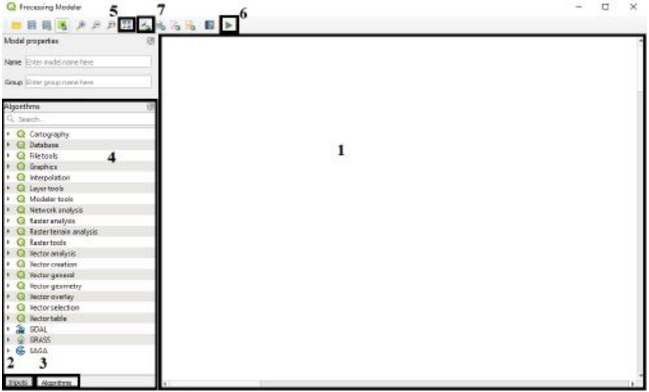
- Download U.S. counties from `http://www.gadm.org/`.
- Download global raster data for agricultural suitability from `https://nelson.wisc.edu/sage/data-and-models/atlas/data.php?incdataset=Suitability%20for%20Agriculture`.
- Both also on the google drive

Graphical Processing Modeler

Getting started

Starting the modeler

- Processing → Graphical Modeler OR
- In the Processing Toolbox panel, click on → Create New Model OR
- `Ctrl+Alt+G`



1 Canvas: showing workflow of processing tools

2 Defining inputs

3 Choosing algorithms (processing tools) to transform inputs; currently algorithm tab selected

4 List of geoprocessing algorithms by category

5 Zoom canvas to full extent

6 Run model

7 Export model to python script

Graphical Processing Modeler

How to build a model

Principle: save only the final output

- Geoprocessing can be complex – many steps executed in sucession
- You don't care about intermediate outputs, just final product (usually a .csv)
- Most tools we use in the graphical modeler (and later python scripts) store the intermediate products in temporary files (under Windows: `/AppData/Local/Temp/processingXYZ`) if you ask it to
- Otherwise, things are done in memory

Chain tools together

- Search for a tool under Algorithms
- Drag the tool onto the canvas
- (double) click on the tool to open its configuration window
- For tools that don't use primary inputs from your disk, use output from a previous tool as an input, `'Output name' from algorithm 'Algo name'`
- Most tools have a field with a grey text *[Enter name if this is a final result]*
    - If you want to load results of some steps into the GUI, enter a layer name, o/w leave blank
    - For the final output producing tool, enter a name

Run the model

- Click ▶ to run the model.
- There is one output parameter for every *[Enter name if this is a final result]* that you replaced with a layer name above
- Uncheck "Open output file after running algorithm" for outputs that you don't want to load as layers after the tool is done
- Enter a full path name for the final output file (usually a .csv)

Graphical Processing Modeler

Example: agricultural suitability for all U.S. counties 1/2

We use the following tools and configurations:

GDAL: Warp (reproject)

- Input layer: /suit/suit/hdr.adf
- Target CRS: EPSG: 4326 – WGS 84
- Resampling method: Nearest neighbor
- Reprojected: agrisuit (we want to inspect this layer)

  Note: this tool can also be used to change raster resolution.

Drop field(s)

- Input layer: /USA_adm_shp/USA_adm2.shp
- Fields to drop: ISO;ID_0;NAME_0;ID_1;ID_2;HASC_2;CCN_2;CCA_2;TYPE_2;ENGTYPE_2;NL_NAME_2;VARNAME_2

  Note: this has to be entered as a list separated by semicolons with no spaces!

Add autoincremental field

- Input Layer: 'Remaining fields' from algorithm 'Drop field(s)'
- Field name: cid
- Start values at: 1
- Incremented: counties

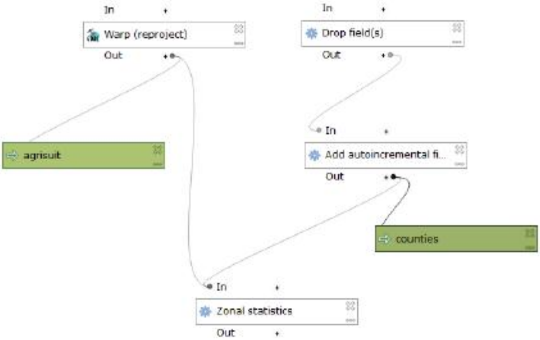
Graphical Processing Modeler

Example: agricultural suitability for all U.S. counties 2/2

Zonal statistics

- Raster layer: 'Reprojected' from algorithm 'Warp (reproject)'
- Raster band: 1
- Vector layer containing zones: 'counties' from algorithm 'Add autoincremental field'
- Output column prefix: _
- Statistics to calculate: Mean

Final workflow



Python

A sales pitch

Even if you don't end up using GIS, python may be for you

- Fully fledged programming language, very easy to learn
- Jupyter for seamlessly combining beautifully formatted text and code: reproducible research! (can do symbolic math, so even for theorists!)

  `http://jupyter.org/`

  - Paul Romer (heard of him?) is a fan: `https://bit.ly/2D82Q16`
- The "scientific stack": *numpy, pandas, scipy, matplotlib, Scikit-learn,...*
- Very promising (still in alpha, Mac users will have easier time trying it out than Win users:) *pydatatable*
  - `https://www.youtube.com/watch_popup?v=1yTHSxJ4KL8`
  - `https://github.com/h2oai/datatable`

Introduction to Python

Installing python

QGIS comes with python

- If you have installed QGIS, you already have python.

The official source

- `https://www.python.org/downloads/`

A nice distribution

- Anaconda (choose the lightweight miniconda)
- `https://docs.conda.io/en/latest/miniconda.html`
- Has most packages you will need if you want to use python for research (with miniconda, type:
  `conda install packagename`
  from the command line)

2.x.y or 3.x.y ?

- The two versions are maintained and widely used in parallel.
- For the basic functionalities we will use, the differences are minimal.
- QGIS 3.12 works in python 3.7 so that is what we will use.

Introduction to Python

Language peculiarities

Case sensitive

- `a = 2` is different from `A = 2`

Indentation is syntactically significant

- You don't need to enclose blocks in `{ }` as in Stata, or terminate blocks with a statement like `end` in MATLAB.
- Indent to start a block, dedent to end it.
- Statements that should be followed by indentation start with a colon ":".

Path names

- Use frontslashes or double backslashes or raw strings (a backslash inside a raw string is just a backslash, otherwise backslash is used to escape special characters).
- `'C:/the/path/to/your/folder'`
- `'C:\\the\\path\\to\\your\\folder'`
- `r'C:\the\path\to\your\folder'`

Opening and closing Python

Windows

- Start → Search → “cmd” → Enter
- Type `python` and hit Enter
- If you followed the optional instructions for installing QGIS in lecture 1, open the OSGeo4W Shell and type `python-qgis` instead

Mac

- Search for “Terminal” and open it
- Type `python` and hit Enter

Closing python

- Type `quit()`

We will now switch over to python. The examples we will run are in the file *python_intro.py* on google drive.

Introduction to Python

Interactive mode and normal mode

Interactive mode

- Open python
- You see a prompt >>>
- Type the examples in section 1) - 7) from *python_intro.py* into the command line / terminal.
- In interactive mode, you get immediate feedback for each statement. Previously run statements (such as variable assignments) are kept in memory.

Normal mode

- Write a python script and save it with the ending *.py* in some directory.
- Inside the command line / terminal, type `pwd` to see your current working directory.
- Change to the directory containing the *.py* file by typing `cd path/to/directory`.
- Run the script by typing `python scriptname.py` (or `python-qgis scriptname.py` if you followed the optional instructions)
- All the commands in the script are executed, just like in a *.do* or *.m* file.
- Alternatively, you can use QGIS's **Python Console**: Plugins → Python Console (`Ctrl+Alt+P`).
- Open the code editor with 📝 and run the script with ▶

Agricultural suitability in U.S. counties example

Saving Model Builder model to python script

Exporting to Python

- Inside the QGIS Graphical Modeler, click Export as Script Algorithm ( )
- This opens a window of neatly formatted code. Copy all of it, in the main QGIS window open a python console via Plugins → Python console, show the editor with , paste the code and save it to some file name ending in *.py*.
- This will give you a raw python script, which you can edit further to make it more readable and make it run smoothly.

Try to run the script

- Clicking ▶ to run our python script produces no output.*
- Let's go into the script to fix that.

*If you run in the command line or in QGIS Python Console. If you run in in the Processing Script Editor, you get the exact same result as the graphical modeller

Agricultural suitability in U.S. counties example

Understanding and cleaning up the script, part 1

What QGIS has produced and what we want

- QGIS gives us the geoprocessing script organized as a **Class**, with all the geoprocessing happening inside the `processAlgorithm` **method**
- It also imports a bunch of modules
- For now, we will make this simpler (forget about OOP and classes) and just keep the core geoprocessing steps and only the module imports necessary

The header

- Remove all the imports, class, and method definitions before the first `alg_params` dictionary
- Before getting to the definition of local variables, enter (adjust the path name to fit your directory structure):

```
maindir = 'C:/Users/se.4537/Dropbox/PoliteconGIS/LBS_2020/PhD/lecture_2/gis_data'
```

The other paths then follow as in the script if you unzipped the GIS data into a folder of the same name.

Optional

- If you want to call pyqgis directly from the command line, you have to uncomment the lines starting with `print('preliminary setup')`

Agricultural suitability in U.S. counties example

Understanding and cleaning up the script, part 2

Intermediate outputs

- `qgis:` and `native:` (latter are written in C++, so faster) algorithms can store outputs in memory if we specify `'OUTPUT': 'memory:'`
- other algorithms, such as those of GDAL, require storage of outputs. Here we create a "junk" folder that we will delete after the program has run
- inside our simple program, both types of output are stored in simple variables (the automatically generated script created a hash table to store the outputs but that is unnecessary complexity)

Algorithm parameters are specified via dictionaries

- The basic syntax for running an algorithm is `processing.run('provider:algoname', parameters)`
- The parameters are specified in dictionaries. Useful, since the order of arguments doesn't matter
- Optional parameters don't have to be specified in the dictionary; default values are then used

Writing the output

- We use our own code to write the output
  - The code writes the attribute table line-by-line with fields separated by commas
  - It first gets the field names and outputs this as the first line of the `.csv`
  - It then iterates over the rows of the attribute table
  - Each row contains the values of each field stored in a dictionary that has the field names as keys
  - Later we will wrap this `csv` writing procedure in a function
  - we will also see a much simpler way!!!

Agricultural suitability in U.S. counties example

Understanding and cleaning up the script, part 3

Processing tools

- It is a good idea to comment out all geoprocessing commands in the beginning, run the script; then (if it works), successively un-comment one geo-processing command after another.
- As geoprocessing scripts can run for a while, it is a good idea to insert messages using `print('xyz')`.
- The parameter-dictionary of geoprocessing tools translate the menu-structure of QGIS into python scripting language. Essentially, every field you fill out in the QGIS GUI corresponds to a key in the dictionary.
- To find out how to specify the parameters for a given tool, type the following:

```
for alg in QgsApplication.processingRegistry().algorithms():
    print(alg.id(), "->", alg.displayName())
```

- This gives the full list of algorithms: their IDs and display names. To get information on a particular one, type

```
processing.algorithmHelp("provider:algoname")
```

For a list of coordinate systems and their codes, see:
http://resources.arcgis.com/en/help/arcgis-rest-api/index.html#/Using_spatial_references/02r3000000qq000000/

A second example

Area for every country in the world

We will calculate the area of every country in the world

- from the google drive or from `http://www.naturalearthdata.com/downloads/10m-cultural-vectors/` download the "Admin 0 – Countries" from and unzip
- create a python script to
  - drop all fields except country-name and ISO_3 code (can we do this more elegantly than just listing all the variables we want to drop?)
  - project to an equal area projection
  - fix geometries (otherwise area calculation will crash)
  - calculate area in 1000s of km^2. Note that `$area` and `area($geometry)` do (almost) the same thing if, before calling the second one, we have brought the data into an appropriate projection before running it
  - output the result as .csv