IP: Application Development

Steffen Reichel

Introduction

These scripts convert CSV files that contain point data in a given directory to GeoTIFF using GDAL's Grid function. It has been tested with and developed for rain radar data provided by ZAMG. Nonetheless they should also work with other data, as long as they meet certain format requirements. Prerequisite is, that there are three columns containing longitude, latitude and data value (data points with x,y,m value). The format itself can be described in a VRT template file. As second script reads a directory containing GeoTIFF files (preferably produced by the first script) and creates a CSV file containing a time series for a given coordinate.

Structure of implementation

The solution consists of three scripts that can be found in the *src/* folder. In the *utils/* folder there is a scripts *gdalUtils.py* that wraps all the used GDAL functionality into functions.

The scripts *csv2gtiff.py* and *csv4point.py* are scripts that can be run and convert CSV files to GeoTIFF and create a timeseries for a given coordinate out of a GeoTIF file timeseries.

It uses GDAL's Grid function to create a raster dataset out of point data. To get the values for the timeseries CSV it uses GDAL's ReadRaster function. The source itself is documented inside the python files.

Prerequisites for running the scripts: Python 2.7 & GDAL

Python 2.7 and GDAL (including their python bindings) are needed to run these scripts. I recommend using the Anaconda[1] python distribution, as they have a very nice way of managing multiple python environments. Every other Python interpreter should work as well, as long as GDAL and its Python libraries (tested on version 2.1.0) are installed correctly.

Python environment setup

To setup an environment with the name "ip-app-dev" using Python 2.7 and installing the GDAL libraries use the following on the commandline.

```
#> conda create -n ip-app-dev python=2.7 gdal=2.1.0

#> source activate ip-app-dev
```

To activate the environment under Windows use

```
#> activate ip-app-dev
```

Now all environments should be set correctly and you can run the scripts from the commandline. For further information on Anaconda and how to setup and manage Python environments refer to their documentation[2].

[1] https://www.continuum.io/why-anaconda
[2] http://conda.pydata.org/docs/using/envs.html

Running scripts

Once the correct environment is installed and activated the scripts can be started. Some sample data is provided in the *data/* directory.

csv2gtiff.py

The *csv2gtiff.py* script converts all CSV files to GeoTIFFs. To run it on the sample data call

```
#> python csv2gtiff.py ../data ../target
```

This will run the script, convert all CSV files in the *data/* directory to GeoTIFFs and store them in the *target/* directory. *target/* will be created if it does not exist. The script provides more parameters to adapt to different input data. To get help on its parameters call

```
#> python csv2gtiff.py -h

usage: csv2gtiff.py [-h] [--vrt_template VRT_TEMPLATE]

                    [--fill_interval FILL_INTERVAL]

                    [--name_format NAME_FORMAT]

                    input_directory output_directory
```

The paramerters _input_directory_ and _output_directory_ are mandatory. The _--vrt_template_ parameter takes a filename with a VRT template. This can be used to adapt to the data format of th CSV file. There is an example provided in the _data/_ directory. This is also the default template.

The script is meant to convert time series data. In case that there are data files missing it provides the posibility to fill up gaps with files containing NODATA values. The parameter _--fill_interval_ takes an interval in minutes. The sample data provided in _data/_ contains gaps. The data is meant to have an interval of 5 minutes.

```
#> python csv2gtiff.py --fill_interval 5 ../data ../target
```

This will create a complete time series filled up with NODATA GeoTIFFs. The default template for parsing the timestamp out of the file name is `inca_sbgl_%Y%m%d-%H%M+000.csv`. This can be changed using the *--name_format* parameter. The format is parsed by Python's `strptime()` and the format parameters are described in the Python documentation[3].

csv4point.py

The script _csv4point.py_ will extract a time series out of a directory containing GeoTIFFs (e.g. created by _csv2gtiff.py_) for a given point. Help on it's parameters can be displayed calling

```
#> python csv4point.py -h

usage: csv4point.py [-h] [--name_format NAME_FORMAT]

                    [--no_data_as_none NO_DATA_AS_NONE]

                  input_directory output_file latitude longitude
```

The *--name_format* parameter works exactly as in *csv2gtiff.py*. The *--no_data_as_none* parameter is a switch. When present it will write NODATA values as empty cells to the CSV. If not present the NODATA value will be written.

[3] https://docs.python.org/2/library/datetime.html#strftime-and-strptime-behavior

The parameter *input_directory* points to the directory containing the source GeoTIFFs. The *output_file* is the full path of the CSV file to be written. If the directory does not exist, it will be created. If the file exists, it will be overwritten. *Latitude* and *longitude* are the coordinates.

```
    #>    python    csv4point.py    --no_data_as_none    ../target
../target/test.csv 12.998 48.021
```

This will create a *text.csv* containing all values for the coordinate (12.998,46.021).