Assessment 3: Working with Rasters and Digital Elevation Models

1 Introduction

This is an individual assessment. It is worth 20% of your final class mark. No late submissions will be accepted.

2 Aim

The aim of this assessment is to:

1. Understand how spatial *raster data* are represented in a raster data format;
2. Apply your understanding of georeferencing and coordinate systems in conjunction with raster data;
3. Understand how you can use Python to efficiently analyse raster data and derive raster products;
4. Explore the impact of data uncertainty on the produced outcomes (and start thinking like data scientists reporting on their work);
5. Practice defensive programming strategies and iteratively develop your code with a remote code repository;
6. Understand how you can create modules that can be easily imported into your projects and to share with others.
7. Use a literate programming (write for human readers first, then code) approach proper to data science to produce a report of your analysis.

3 Fundamentals of Computational Raster processing

3.1 Digital Elevation Models

We have discussed rasters in the lectures: you have an understanding of how rasters decompose space into rectangular cells of equal dimensions. In this assignment, you will get to practice this understanding, and reflect on the issues with georeferencing rasters, their reprojection, and resampling. You will work with rasters with a single set of values - a digital elevation model. This is different to say, multispectral rasters, or rasters capturing visual spectrum information (which would come in bands of Red, Green, and Blue). The rest of the work is, however, identical.
DEMs are critical for applications in hydrology, for the computation of fundamental properties of the terrain used for site selection, or for a variety of applications, from solar panel site selection to vineyard planning.

3.2 Terrain analysis: Products

To analyse terrains captured by DEMs you compute data products, as results of operations:

- *Local* – the simplest raster products are those that results from operations on a single cell: we call operations that produce these products . You only need the value of a cell to compute this operation (such as, the value of the elevation 2m above the current cell's height).
- *Focal* operations compute a value of a cell based on the neighbourhood of the cell. These are very common, and important (and relate to the moving window approaches that you have practised to compute trajectory parameters). Gradient, and aspect (orientation of maximum gradient) maps are products of focal operations, and you will practice these in this assignment. Focal operations are equivalent to convolutions, as they are now known (when applied in a matrix manner). These are now extremely useful for the omnipresent Convolutional Neural Networks (CNN), and you are sure to meet them again. While in

CNNs the machine learns the right filter, in this assignment we design them - we code what the filter looks like. To read more about some useful convolutions, see https://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/convolution-function.htm, and this great visual explanation: https://setosa.io/ev/image-kernels/. You will notice a relationship between Sobel filters and the Slope computation.

- *Zonal* operations require two input rasters, one containing a *mask* of cells defining zones, and an operation that takes the values of the cells incident with the mask to compute a result (such as the average height of the terrain within a given suburb), and,
- *Global* operations, where all the cells in a raster must be evaluated. You execute a global operation when looking e.g., for the maximum height cell in a raster.

3.3 Arrays of numbers and raster-specific Python libraries

You have been introduced to Numpy arrays. In this assignment, you will experience wrappers that enable working with raster data captured in Numpy arrays: rasterio (a Python wrapper over the C library GDAL, (https://rasterio.readthedocs.io/en/latest) and Pysheds (https://github.com/mdbartos/pysheds). YOu may also use the SciPy (a sister project of Numpy) convolve2D operation, documented here: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html.

You will use some of the functions of these packages, but you will also be asked to implement a function on your own, to understand how raster processing is done (without the need to engage in optimisation approaches).

4 Assessment

In this assignment, you will implement two different functions to compute the slope (maximum gradient) of a raster, based on a paper description of the functions [1] https://www.witpress.com/Secure/elibrary/papers/RM11/RM11013FU1.pdf. You will compare their outputs, and understand the impact of raster parameters on the outputs. This is a more complex assignment than the last one – engage with it as soon as possible. The amount of code needed is not extensive, but you are asked to think about the inputs and outputs, and also conduct the analysis in a stringent manner. Give it your best effort!
In this assessment you will be required to write three files:

1. Write a fully documented Python module named **rasters.py**,
2. A test module named **test_rasters.py** that will test the functions written in the above module,
3. A notebook program named **assessment3.ipynb**, that will utilise the functions written in **rasters.py** to document and perform the analysis tasks below.

Additionally, you will attempt to automatically produce the final report of this assignment, organising the structure of your code so that it fits with the template structure in the folder `resources/reproducible_reports`. You will find there a structure that will help you generate this report (more below).

Take note that like in assessment 2, the specified function **names**, **input arguments** and **output data types** are important in order to follow test rules in the final task. Boilerplate code for the required tests can be found in the repository alongside this document.

In this assessment you will also be required to make a minimum of **three** meaningful commits in your personal git repository. They must provide substantial change from your previous commits, with an informative commit message. There is **no maximum** number of commits you can make, so if you choose to frequently commit all of your work, this will illustrate meaningful commits (just be sure to also have unique and relevant commit messages).

Input data and test-file boilerplate code are contained in the git repository for Assessment 3.

5 Questions

5.1 Task 1: Spatial data reading and exploration

Extract the Geoscience Australia sourced **DEM.zip** and read in the spatial data provided in the **CLIP.tif** GeoTIFF file.

1. Write a function in rasters.py named `summary_dem(filename)`, that reads in a raster file and outputs summary information about the input necessary to populate the below table. The output data type must be a dictionary.

Parameter	Value
Filename	name
Coordinate system	value [EPSG]
Min x, Min Lon	value [units], value [units]
Max x, Max Lon	value [units], value [units]
Min y, Min Lat	value [units], value [units]
Max y, Max Lat	value [units], value [units]
Width, Height, Cell size	value [px], value [px], value [unit]
NoData	value
Min value, max value	value [units], value [units]

2. Create a function **display_summary(summary_dict)** that prints the values of the previous subtask's dictionary in a nicely, automatically spaced output (like above), where the longest length of each column is adjusted to the length of the longest string.

   Use and possibly extend the PrettyTable function in the ProjectModules folder of the reproducible report folder in resources as a target for your function. Before you use it, read through the code carefully. It is a nice, simple output function. See how Python's string formatting functions are used here. Try to understand it, and search online if something is unclear. Reading others' code is part of learning to program. Make some visual changes to the output of this library, and document them.

3. Use the bash script (run.sh) (or batch script (run.bat) if you're on Windows) in the `scripts/` folder to generate the resulting report. It should generate a .tex file (a LaTeX file), and then execute it to generate a pdf. LaTEX is a useful way to generate reports in programming, as documents are compiled from source code which can easily be generated from within Python. This assessment document you're reading, for example, is also written in LaTEX.

   In order to generate a PDF from your generated .tex file, you will need a TeX installation. You can install this with Anaconda, however it differs depending on your operating system; for Windows you need to install miktex from the conda-forge repository, for MacOS and Linux, you can install texlive-core also from the conda-forge repository. Both of these install a set of tools to compile LaTEX code. The one we're interested in to compile a PDF document is **pdflatex**, which allows us to compile from the command-prompt a .tex to a .pdf by typing:

   ```
   pdflatex assessment3.tex
   ```

   If you look within the script mentioned above, you will see this happening for you. In your submission, make sure you rename the output PDF file from the default name.

   Note: If you cannot run the script on Linux/MacOS due to a permission denied error, you may need to enable execution privileges from the terminal (chmod +x run.sh). Bash and batch scripts are also just an automated way at issuing commands, so if all else fails you can copy and run these line by line.

4. Write a function that outputs the geographic coordinates of the cell with the highest value in the dataset, and output a visualization of the raster with the cell highlighted (how you do this is up to you, there are multiple ways, using rasterio). Add this visualization into your report above, and nicely add the value of the cell's coordinates in the report too.

5.2 Task 2: Processing a raster DEM

In this task we will process the raster DEM using the `rasterio` package to compute a slope and aspect. You will implement two functions from the paper noted above (2FD and Maximum max methods). You can implement the second method as a moving window function, and the first as a convolution.
Your tasks are:

1. Convert the dataset to a projected coordinate system appropriate for the area (a Victorian projected coordinate system) – find the right EPSG value for yourself on epsg.io. Generate two coarser versions of the dataset with cell sizes of 2x and 4x the cells size, using bilinear interpolation and output the histograms and tables as above.

2. Write a function to implement the slope functions.

3. Compare the distribution of the slope values, and the statistical properties of their distributions for both functions and rasters (4x combinations). A summary of data is best complemented by a visual representation. Write a function that outputs a histogram (use **matplotlib**) of the cell height values in the raster.

4. Write a markdown cell where you interpret the results and discuss the impact of coarsening on the data. Present the results in a nicely formatted table and text. In a text cell, reason about the choice of these functions (you can get inspiration from the paper noted above).

Note: When creating plots, you are encouraged to use the customisations of **matplotlib** to best describe your data. This means using an appropriate number and width of bins in histograms, correct axis labels, and descriptive titles. Also, think deeply into what you want to offer the reader here. Would it be beneficial to plot all iterations of your data on one plot for a side-by-side comparison, or in different plots? Being descriptive with your outputs are more engaging for the reader than raw statistical output.

5.3 Task 3: Test your code

Included in the git repository, you will find **test_rasters.py**, this has been filled with the required boilerplate code for you to insert test cases to test the above functions you have written for the assessment.

In assessment 2, your submission needed to pass all the provided test cases successfully; in this assessment you will be marked on the test cases you create for your functions and their ability to reliably test the outputs of your functions.

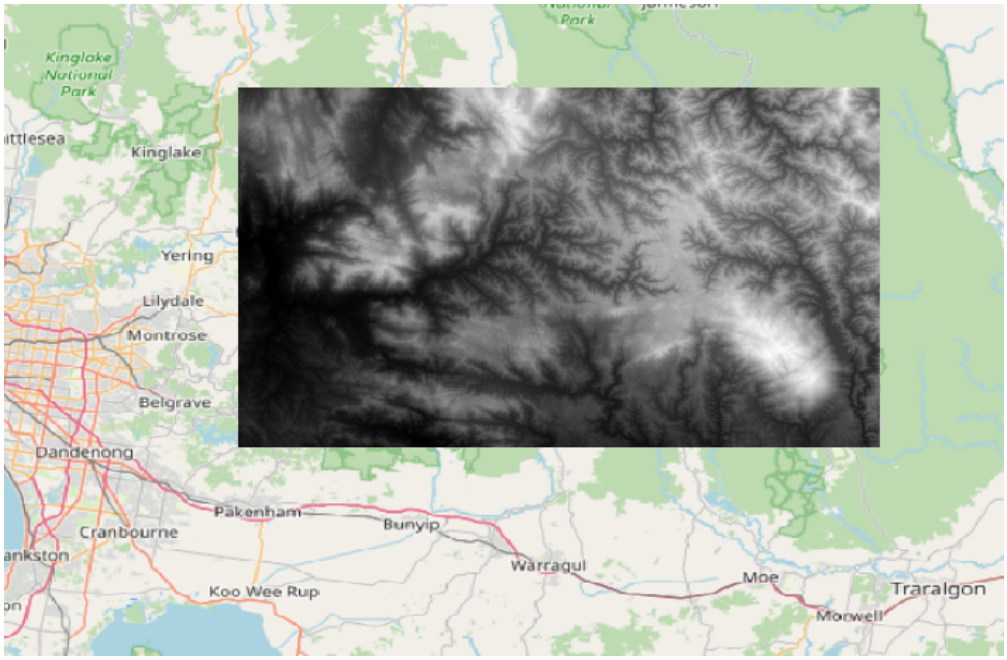
You are required to write **three** test methods for the functions in the first task. Some suggestions for the types of tests that you could use would be to test the validity of your functions:

- If the function calculates and returns a value that should be within a known range, you should try and test this. You could also test the opposite and check if your code handles error cases the way you intended it to.
- Are you able to check validity off of a known value? If you look at the tests included in assessment 2, we can see this being done in **test_project_coordinate()**, where the outputs of a projection are tested against known values that were, in this case, derived from a reprojection in QGIS.

Much like building exception handling in to your code, you should strive to test things you assume could go wrong.

6 Sample data

The included data covers the region below:



7 Marking scheme

Your assessment will be marked out of **20**.

The table below shows detailed marking criteria associated with marks:

Criteria	Sub-criteria	Mark
Correctness	Task completion and output correctness	6
	Interoperability, exception handling and tests cases suitable	4
Design	Abstraction of problem, appropriate use of functions and module design	3
	PEP-8 conformity in tests and module, and elegance in execution	3
Documentation	Documentation appropriate, notebook descriptive and reproducible	2
	Frequent and meaningful commits in git repository	2

8 Tips

Make sure when reading the documentation for the libraries you import, that the documentation version matches the installed version. You can generally find the version of a library by executing a .version() function within it (this will differ depending on the library), or you can check your installation version from an exported Anaconda environment file.

9 Submission

Submit a single **.zip** file (where **studentno** is your student number) containing the following files for submission:

- **rasters.py** (your code)
- **test_rasters.py** (your test scripts)
- **assessment3.ipnyb** (your Jupyter notebook)
- **assessment3.pdf** (the generated PDF file from the LaTEX)
- **environment.yml** (your Anaconda environment file)

studentno_A3.zip

You submit your file by uploading the file to Canvas. Only the last upload before the submission deadline will be marked.

References

[1] J Tang and P Pilesjö. Estimating slope from raster data: a test of eight different algorithms in flat, undulating and steep terrain, 2011.