Assessment 4: Spatial Data Science: data input, manipulation, analysis and presentation

1 Introduction

This is a group assessment, for groups of two students, with – in exceptional circumstances – groups of up to three students. Each member of the group will also lodge their own, individual reflection on the assessment and their contribution.

The assessment is worth **50** of your final class mark. The assignment allows for extensive creativity and self-exploration. Creative solutions and approaches will be bonified with bonus marks counting towards the final subject results (max 4 marks).

No late submissions will be accepted.

Problem setting:

Imagine you are a spatial data analyst tasked with investigating the spatio-temporal patterns in traffic accidents. Data about the traffic accidents are collected by the road authority on a regular basis. You have to:

1. Automate the data processing pipeline to reduce your and others' workload with routine data processing;
2. Enable regular generation of statistical overview of the data – updating a report containing tables, graphs and maps giving access to the summary overview about the dynamics of accident occurrences;
3. Enable insights (including statistical) into any significant patterns and/or changes of patterns (especially spatial) in the data, that may require decision-maker attention;
4. Generate appropriately structured, *analysis ready* data products, as report results of your analysis for decision makers and other domain-experts and analysts.

2 Aim

The aim of this assignment is to increase your exposure and experience with the basic elements of Python from Assignments 1 and 2, by incorporating the use of spatial data input/output, the use of advanced data structures (e.g., using Pandas/Geopandas, or Fiona and Shapely), the definition of your own toolbox of applied functions and/or spatial analyses (e.g., clustering or association rule mining, or other), explore the use of general purpose statistical and spatial data analysis libraries, and visualise the results using data visualisation libraries (e.g., matplotlib, seaborn, plotly, cartopy, altair, holoviews/geoviews and others). This assignment consists of the following tasks:

1. Practice the ingestion and manipulation of spatial data from a series of shapefiles, together with other supporting datasets of your choice to produce an analysis-ready dataset;
2. Perform exploratory data analysis, providing overview and insights into your data, supported by tabular data and graphical visualisations, charts and maps;
3. Independently explore and experiment with basic or even advanced spatial analysis concepts and tasks, including through the implementation of your own utilities/algorithms;
4. Demonstrate the ability to collaborate with a group member on sharing the task workload, coordinate efficiently during the project timeline, and help each other assure the quality of the delivered product supported by git-based workflows.

3 Assessment

Data: There are three sets of data as the input of this assignment:

- Six years of Victorian traffic accident data from VicRoads during the years 2013 to 2018. The data provides users with enough information to analyse fatal and injury crash data based on time, location, severity, crash type and count of affected persons. The full metadata is provided in **CrashMetadata.csv**. Each year of data is divided into its own point shapefile.
- The boundaries for the ABS level 2 statistical regions of Australia. These geographically represent population boundaries of the country set up at periodical census dates to allow mostly for demographic statistical analysis on. Official documentation can be found here.
- The local-government area (LGA) boundaries of Victoria, supplied by the ABS.
- You can source and use other data of your choice – for instance, a street network, sourced either from Open Data Vic (`https://discover.data.vic.gov.au/dataset/road-infrastructure-vicmap-transport`) or OpenStreetMap, as well as other data of your choice (e.g, weather conditions). Any additional data should be kept to a minimal size needed, and should not exceed 50Mb – if necessary, you may restrict your area of interest for the demonstration of your approach. These data should be uploaded to your Gitlab data folder, and may be preprocessed (through a separate, documented script).

Your analysis should be structured in three main report sections (a Jupyter notebook with three clearly identified sections, properly documented using markdown) for the following three tasks, respectively. These tasks can be coded independently from one another but should be included in **one** notebook. You should assume that all the input files are unzipped and in the `/data` folder of your project. Any custom functions should be read from a module residing in an appropriate folder in your project (i.e., not be in your notebook). For the output directory, you should create a sub-folder named `output`. We recommend to use the project structure template discussed in the `resources` folder of the GIT repository of this subject, and in the `reproducible_reports` template.

4 Tasks

Provide a report of your analysis, outlining the three main tasks below. Therefore, an **introduction** should be provided, outlining what this analysis presents, and concluded through a **discussion** and **conclusion** section that wraps up the report. The report should use meaningful title and subheading including your names, contact and student number (formatted following your own preferences). We recommend one section per task, with subsections.

Explore all shapefiles and understand the data. Use well-documented function definitions meaningfully organised in modules (imported into your notebook or script, like what we did with rasters.py in the previous assessment), to produce the results as per following specifications. You may call the modules how you wish, and have as many as you see necessary, providing they are created in a logical fashion (ie. one module with one function is excessive, however, a module for data file handling, and another for spatial analysis would make the functions easier to find). You can organise the modules in a library, for ease of handling.

4.1 Task 1: Data extraction and Exploratory Analysis

This first task should be executed within your notebook from functions written in your modules, under the heading of in **Section 1: Data extraction and Exploratory Analysis**.

1. **Text output:** Provide functions that assist in automatically outputting the following information into this section of your report:

   (a) The average number of accidents per year is ____.

(b) The most common type of accident in all the recorded years is ____ (equal to ____% accidents), and the second most common is ____ (____%).

2. **Tabular output:** Enrich your report by adding graphs and data tables to it. You can do it using whatever means you decide such as through Pandas, PrettyTable, HTML tables – as long as they look professional. Remember, as it is a report, the tables should have captions and numbers. Generate two tables:

   (a) **Table 1** Number of accidents by vehicle type (rows) by year (columns). Organise your tables so that it is sorted by the number of accidents in the first reported year. The vehicle types need to be discerned from the meta-data and the definition of a vehicle type must be consistent with the rest of your analysis. For example, if you decide that it is valuable to consider a bicycle as a vehicle type, it must also be noted why and be consistent with the rest of your report.

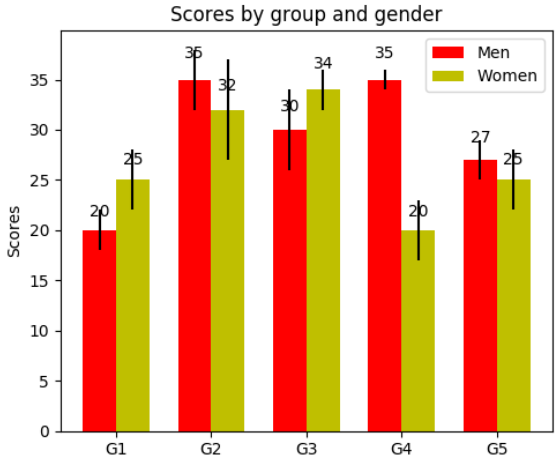
	<td>	<td>	<td>
<tr>	Content	Content	Content
<tr>	Content	Content	Content
<tr>	Content	Content	Content

   (b) **Table 2** A table of the top 10 LGAs (Local government areas) that have the highest number of accidents in 2013, sorted in decreasing order. Then, compute the changes of the numbers of accidents in 2014 compared to the previous year, and so on, up until 2018 for these 10 LGAs. These differences should be computed both in absolute numbers and as percentage change. An example is given below, however you can choose your own name headings and layout. You may decide to insert extra information if you decide, or format your table so that cells highlight significant percentage changes. You have freedom in the table formatting, as long as it has a professional appearance.

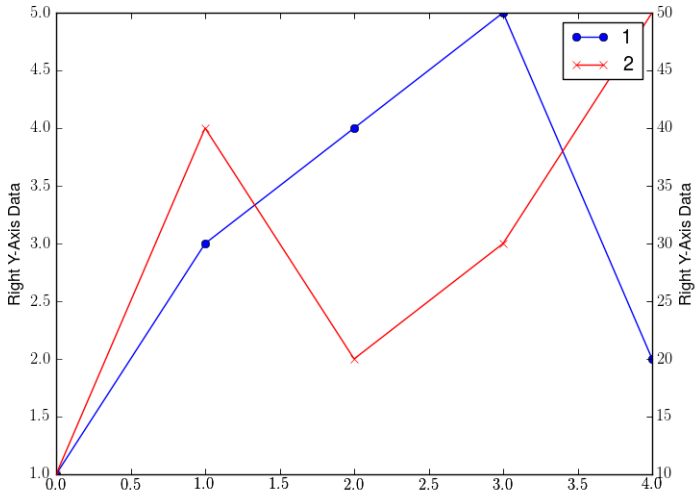
LGA	2013	2014			...			2018		
	No.	No.	Diff.	Change	No.	Diff.	Change	No.	Diff.	Change
Geelong										
Mitchell										
...										

3. **Charts and Maps:** Write functions that produce the three following charts described below for your report. The charts should be exported into a PNG figure with appropriate filenames and also included in the appropriate place inn the report. You may use the **matplotlib** library or any other suitable plotting library.

   (a) **Figure 1:** Produce a **bar chart** of accident numbers by days of the week, comparing the change from 2013 to 2018. The chart should include 7 * 2 bars. The output should look similar to this example:

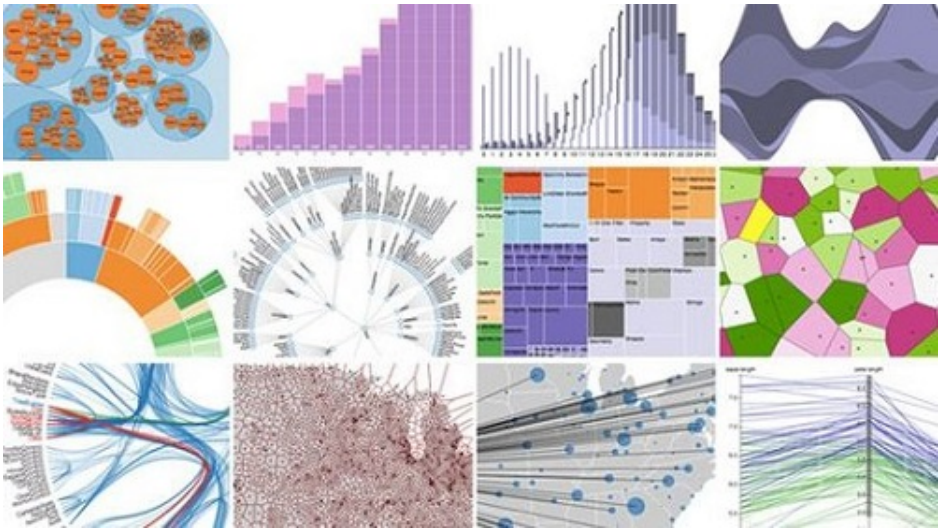


(b) **Figure 2:** Use the same library to produce a **line chart** of the yearly change of the total number of accidents from 2013 to 2018, for each severity category. The chart should include a time series line for each severity category, respectively. An example of such a multi-line chart is given below:



(c) **Figure 3:** Produce a Figure with 2 sub-figures, both **choropleth maps**, showing the fraction of the most common category of accidents per LGA. Display one sub-figure for 2013 and one sub-figure for 2018. Select appropriate colour maps to present these data, and include a legend that is aligned for both sub-figures (the colour ramp splits should be the same).

Note: Statistical and spatial data can be presented nicely and in more advanced ways with the help of third-party libraries (e.g. plotly, seaborn, and several others), which requires deep understanding of the data and careful designing of presentation manner. Some examples are shown below. You are encouraged to explore and play with these libraries and get bonus marks for an additional visualisation (max 1, leave this until you at least finish all the required tasks from above and below). If doing so, you are allowed to create your own task scenarios (but with the same input data). You may display dynamic content in a html version of your report separate to the Jupyter notebook.



4.2 Task 2: Analysis-ready data export

You now need to process the data into a spatial dataset (GeoPackage) that can be used by other analysts in your company (typically using ArcGIS or QGIS). You need to take the textual data provided and export them as a spatial dataset using **Geopandas** and/or **Fiona/Shapely**. This second task should be executed form a function called in your notebook, in **Section 2: Analysis-ready data export**. To begin, look at the geopandas documentation first: http://geopandas.org/index.html. A Geopackage is built on top of the SQLite database, so if you *wanted* to write non-spatial data directly, you should explore this: https://www.dataquest.io/blog/python-pandas-databases/

1. Create functions to export the content captured by the two tables into a single, geopackage dataset, writing a layer of "AccidentsByYear" and "AccidentsByLGA", respectively, holding the data from the two tables above. You will note that you will need to provide a geometry for the "AccidentByYear" – you can, for instance, use the geometry of the whole State of Victoria, or a centroid. The AccidentsByLGA layer should use the LGA geometries - The 2017 boundaries have been included and can be used for all years.

2. Create a new layer with only a subset of the input data for the analysts, called "AccidentLocations". This should contain only data for accidents with at least 3 people involved, and contains the point data with the following attributes: "AccidentNumber", "VehicleType", "DayOfWeek", "NumPeople". Beware of the coordinate system assigned to the generated dataset. Add this to your geopackage.

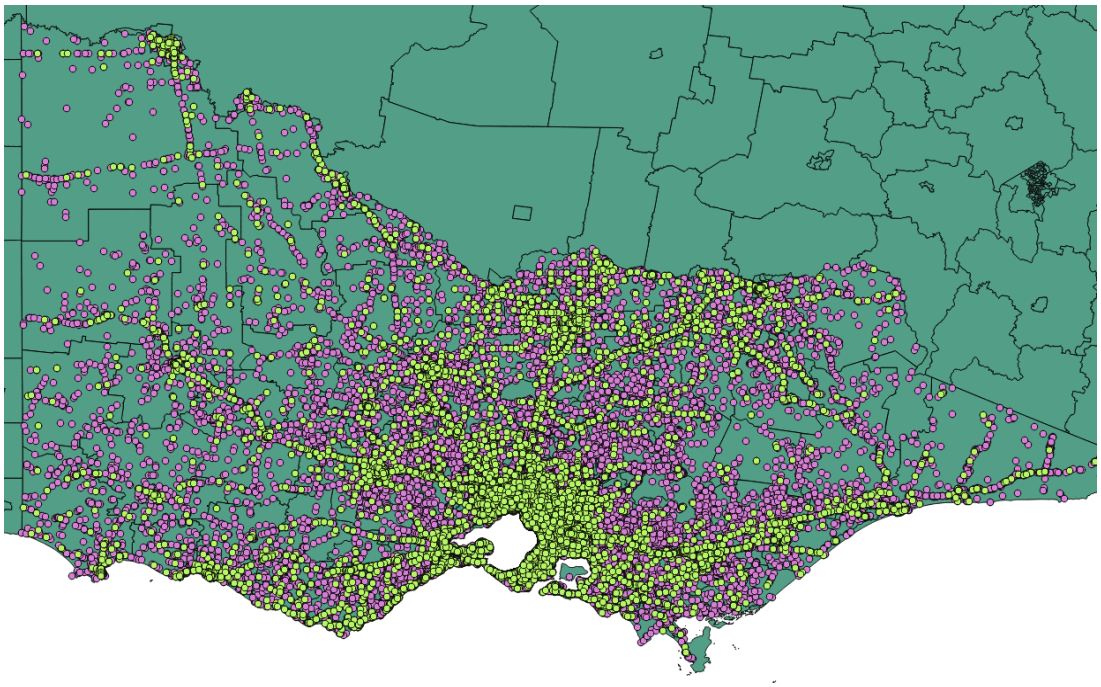
   The first field of the dataset should correspond to the AccidentNumber attribute in the provided shapefiles. Examples of the second field is "PassengerVehicle", "Motorcycle" or similar (a string). The third field corresponds to the day of the week when the accident happened (Monday/Tuesday/...). The last field (NumPeople) will Contain the number of people involved.

3. Add a new field "SA2" to "AccidentLocations". The value of the field will be set to the Statistical Area 2 name the accident happened within. The supplied 2016 SA2 dataset contains the Statistical Area 2 boundaries and should be used here. You should use a spatial join (ie, a geometric point in polygon query as demonstrated in lectures) to to identify the SA2 in which the accident happened. This function can be made more performant with the use of a spatial index. A bonus point will be awarded if you write your code to incorporate and demonstrate the use of a spatial index on top of your spatial data frames. You can include timing statements using "timeit" to explore the relative acceleration. **Remember**, these must be *analysis ready* data. It is useful to create a spatial index, and it is necessary to appropriately set the projection of your dataset.

4. Write a function for the analysts to read the created layer, split the entries in the created layer AccidentLocations into two layers: "SevereAccidentsWeekday" and "SevereAccidentsWeekend". The first resulting dataset should include all entries with fatal accidents that occurred on weekdays (Mon to Fri). Similarly, the second resulting dataset should include entries of fatal accidents that happened on weekends. You should only **provide the**

function, and in the report document it on a small readout. This function will be tested by us, based on your documentation in the report. No additional data other than the output analysis-ready dataset will be used.

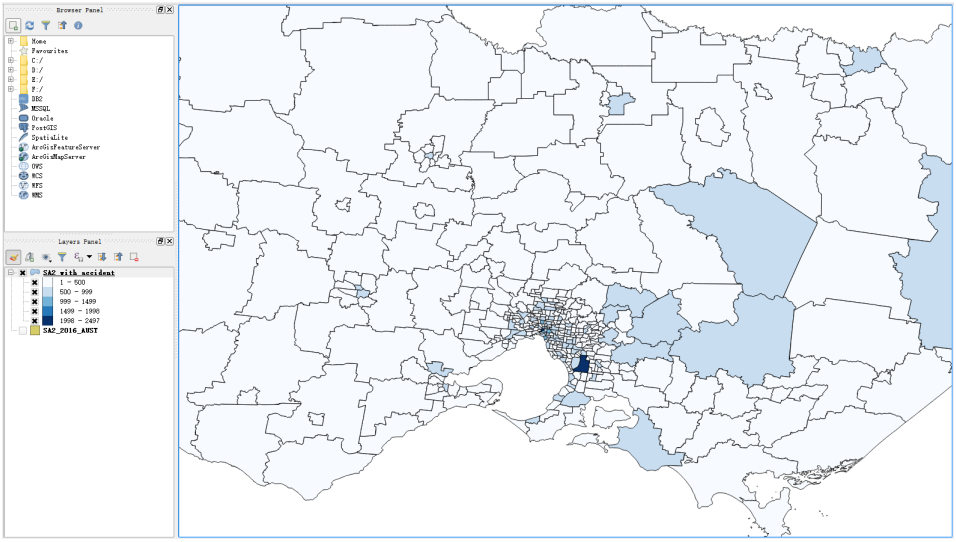
An example of opening the SA2_2016_AUST.shp, AccidentLocations, and SevereAccidentsWeekday files in QGIS is shown below. If your result shapefiles look not similar to the example when opened in QGIS or ArcGIS, you may want to double check your program.



4.3 Task 3: Spatial data insights

In this third task, you will explore, document and discuss the use of spatial analysis (using libraries such as scipy, scikit-learn, pysal, etc.) for analysing the spatial patterns of the distribution of accidents based on your point dataset. Some example analysis tasks are given below. You can either choose from these tasks or develop your own exploratory tasks. Multiple analysis tasks presented as well as in-depth discussions and presentations will receive bonus marks (but keep in mind the extent of the report). Also, should any attempt be made that predicts future occurrences of accidents based off of the analysis, will be considered for bonus marks. This third task should be in **Section 3: Spatial data insights**.

- **Spatio-temporal visual analysis (basic)**
  Create a map showing the number of accidents that occurred in each statistical area. Are there any patterns? How about comparing the result for different years/weekdays and weekends/vehicle types, etc.? Note that all analysis incl. presentation (either in the Jupyter notebook, or exported as a HTML report) must be automated. Also think of a way to present the temporality of the data, generating a GIF is one possible method.



- **Spatial Autocorrelation calculation (intermediate)**
  In order to analysis the spatial autocorrelation of these accidents (roughly speaking, the relationship between accident frequencies and closeness in space), you can use the **Pysal** library, see http://darribas.org/gds_scipy16/ipynb_md/04_esda.html for an example.

- **Clustering analysis (intermediate)**
  Libraries including **Scipy** and **Sklearn** can be used for clustering analysis such as DBSCAN (parameter-dependent) and Kernel Density Estimation (much less parameter-dependent) to generate clustered point clouds or probabilistic surfaces. You can identify spatial concentrations of accidents (using appropriate choice of parameters for your functions, or use the power of batch executions to identify parameters that are meaningful).

- **Predictive modelling (advanced)**
  You can investigate the ability to train a predictive model to infer what will be the numbers of accidents on a given day in the future. You should carefully consider your training and testing data, in a process called cross-validation. This is an advanced task, talk to us if interested.

Functions for this task must be included in its own module. At least one analysis task (either from the examples or defined by yourself) must be completed. You should visualise the results using Python visualisation libraries and include maps into your report, and include any observation you can make about any trends in a per visualisation. Think of possible variations of this task - as an analyst, how would you nuance your analysis, or what improvements can you make? If the analysis can be improved through the addition of an extra data set, this is welcome, but it must be warranted and included in your submission.

Include the results of your analysis in your notebook. Reflect on your reasoning and insights in the data. The entire report should ideally not exceed about 4 pages long (if printed from the notebook), including figures (appendices for Figures and Tables are possible but should be used judiciously). You should carefully reference and cite any external tools and libraries or code, papers, and websites on which you relied. This is a technical report, yet proper attribution is always necessary.

5 Marking scheme

Your assessment will be marked out of **50** according to the criteria correctness, programming style, output, and comments:

- Your Python program is submitted correctly and produces well-formatted output. **(5 marks)**
- Your program performs all required computations and tasks and is well-tested. **(5 marks)**

- Your program is efficient, short, and compact. Code that is modular, and re-useable (for instance, but passing it code with a new set of parameters), will score higher marks. You should import your own functions from your project module or library into the final Jupyter notebook, to avoid unclear analytical flow. Code that is difficult to understand or uses poor programming style (non-pythonic) will lose marks. Your code should always conform with PEP-8, as in previous assignments. **(5 marks)**
- Your code is well documented and understandable in structure, and in operation of the functions. The analytical workflow is clear (apply methods of literate programming). Code that is poorly commented will lose marks. **(5 marks)**
- Group collaboration - you will use your group's GIT repository to coordinate and collaborate on this joint analysis. Collaboration is key to any more extensive analytical task. You should meaningfully divide tasks, and provide quality assurance for each other continuously through the project. You can use more advanced GIT workflows and GITlab features throughout the project. This will be judged based on your GIT repository's history **(5 marks)**
- The analysis is insightful and meaningful. You have applied your spatial knowledge to the problem as a spatial analyst, not just as a software developer. You have reflected on the impact of data and analysis on the results, and are able to point the limitations of both, as well as suggest refinements. **(10 marks)**
- Your report is presented clearly, well structured, interleaving the text through the code to support the transparency and understandability of your analysis. The produced maps, graphs and tables are of professional presentation quality. **(10 marks)**
- You have provided an individual reflection (1 page) on your role in the group, and on the learnings from collaborating collaboratively on a programming task. Your reflection should be honest, individual, and not about your group partner(s). **(5 marks)**
- Creative solutions and approaches beyond the minimal requirements will be bonified with a maximum of **4 marks** towards your final subject's mark.

6 Submission

Submit a single **groupno_A4.zip** file (where **groupno** is your student number) including:

- Your fully commented **Python modules**,
- Your **Jupyter notebook**,
- Your **data directory** including any additional datasets you used,
- Your **output directory** including:
  - All figures,
  - A report named **groupno_A4_report.pdf** of your notebook (Using the nbconvert method from the previous assessment - A browser-printed PDF is unacceptable),
  - Any other outputs, such as if you chose to write a dynamic HTML page in task 2.
- Your Anaconda environment file **environment.yaml**

Each of the group members should submit the assignment via Canvas but must reflect the final version on the group's git repository. At least one of the submissions must contain the complete, structured project folder, excluding the data folder. If any additional data have been used, these will be retrieved from the git repository.

Each team member must also submit their own, 1 page long reflection file, named **studentno_A4_reflection.pdf** in the zip. The main report (**groupno_A4_report.pdf**) must contains the names and the student numbers of the team members. Similarly, the names of all team members should also be included in the actual Python code as a comment. (in each file, if multiple are used).

7 Tips

Make sure you check carefully that your code works correctly, and produces sensible answers (you may want to test answers by opening your outputs using a program that can work with spreadsheets). You should also check that your code works on another computer, and if not, include a set of comments explaining concisely the steps required to install what's required to run it in the project's main directory's `README.md` (and including a Python environment file or a command to install required libraries).