UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Automating exploratory spatial data analysis (ESDA) for vector and raster data: development and evaluation of the autoESDA Python library

by

Nicholas De Kock

Submitted in partial fulfilment of the requirements of the degree:

Magister Scientiae (Geoinformatics)

in the Department of Geography, Geoinformatics, and Meteorology,

Faculty of Natural and Agricultural Sciences,

University of Pretoria,

Pretoria

20 November 2023

ABSTRACT

Title: Automating exploratory spatial data analysis (ESDA) for vector and raster data: development and evaluation of the autoESDA Python library

Student: Mr. Nicholas De Kock, Department of Geography, Geoinformatics, and Meteorology, University of Pretoria, South Africa

Supervisor: Dr. Victoria Rautenbach, Department of Geography, Geoinformatics, and Meteorology, University of Pretoria, South Africa

Co-supervisor: Prof. Inger Fabris-Rotelli, Department of Statistics, University of Pretoria, South Africa

Degree: MSc Geoinformatics, Department of Geography, Geoinformatics, and Meteorology, University of Pretoria, South Africa

autoESDA is a Python library developed with the aim of automating the Exploratory Spatial Data Analysis (ESDA) process. This is done by generating a HTML report made up of various ESDA graphs and statistics calculated according to the input dataset, requiring no other inputs from the user. ESDA (local spatial autocorrelation specifically) in Python has been a challenge for raster datasets, with software support lagging behind alternative platforms such as R. This dissertation documents the improvements made to the original library. These improvements include the support for raster datasets, an updated architectural design, and other minor, cosmetic improvements. The performance of the updated version of autoESDA is evaluated by investigating how its processing time varies according to vector and raster datasets that differ in size and complexity. These results are then discussed as a measure of how well the library has achieved its goal of automating the ESDA process. Finally, a roadmap for further improvements to the library is discussed.

ACKNOWLEDGEMENTS

First and foremost – to my amazing parents – Chris and Helen, whose wisdom, support, and love have no bounds. Thank you for all your encouragement over the years. Without a doubt – you have made all of this possible!

To my supervisors – Dr. Rautenbach and Prof. Fabris-Rotelli, for your patience and guidance over the course of this project. Thank you for all the time you have put aside in your busy schedules to ensure that I never wandered too far off track.

Finally, to all my friends. Thank you for always being on standby to offer some encouragement, distractions, and laughter – be it on early morning runs, late nights out, or random calls/messages throughout the day.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: INTRODUCTION

1.1 Background

Spatial data is currently generated at an unprecedented rate which is only expected to increase. The United Nations Initiative on Global Geospatial Information Management(UN-GGIM) estimates that around 2.5 quintillion bytes of data each day, with a large portion of itexpected to have a spatial component (UN-GGIM, 2020).

There are two main driving forces behind the magnitude of geospatial big data (UN-GGIM, 2020). Firstly, the rise of new data sources, such as crowdsourced data or volunteered geographic information (VGI), Internet of Things (IoT) devices, self-driving cars, and satellites. The wealth of data sources allows for large **v**olumes, **v**arieties, and **v**elocities of data to be generated – these *three V's* are fundamental to defining geospatial big data (Gandomi and Haider, 2015; Li et al., 2016; Robinson et al., 2017). Secondly, there have been multiple technological advances which enable geospatial big data, such as cloud-computing, digital twins, machine learning, and artificial intelligence (UN-GGIM, 2020). While these technological advancements have been instrumental in handling geospatial big data, there have still been multiple calls for new tools that take advantage of the value of geospatial big data (Mennis and Guo, 2009; Vatsavai et al., 2012; Lee and Kang, 2015; Li et al., 2016).

Exploratory Spatial Data Analysis (ESDA) is an extension of Exploratory Data Analysis (EDA). ESDA functions seek to describe and visualise spatial data. It achieves this through the identification of trends, patterns, and outliers, and displaying these results on a variety of graphs, maps, or other visual displays (Anselin, 1999). Often carried out under the umbrella of spatial data mining (Anselin, 1999), ESDA allows one to form hypotheses and suggest associations within a dataset. This is a particularly useful process when one may not have a firm theoretical understanding of the data being used. This is common due to the diverse nature of spatial data. The growth in size of spatial datasets has led to conceptual and computational challenges within the ESDA workflow (Anselin, 1999).

1.2 Research Context

autoESDA is a Python library and the proof of concept was first developed as part of my BScHons project, with the aim of automating the ESDA workflow. By doing so it removed the repetitive and time consuming process one would otherwise face when analysing large datasets (Higgins and Ray, 2022). The library was, however, very limited and it had not undergone extensive testing.

As part of the earlier project, various interviews were carried out which resulted in a wealth of feedback and potential improvements that could be added to the library and its output. One of the notable suggestions was the need for functionality to process raster datasets. The earlier project is summarised in an article that was published in 2022, and is included as Chapter 3 of this dissertation.

This project will expand on the autoESDA library by implementing improvements that have resulted from the earlier interviews, new technological advancements, and an improved understanding of the ESDA process and the tools on which autoESDA is built.

Both projects have been carried out with the approval of the ethics committee of the Faculty of Natural and Agricultural Sciences at the University of Pretoria. The reference number is NAS229/2021, and the full approval letter can be viewed in Appendix A.

1.3 Problem Statement

ESDA is a repetitive and time-consuming process. It requires constant human input which means there is a large margin for human-induced errors. autoESDA is a Python library that was developed to address these issues. This proof of concept shows great potential; however it does not support raster datasets and its performance has not yet been evaluated.

1.4 Research Aim and Objectives

The aim of this research is to advance the automation of ESDA by implementing improvements to the autoESDA library and evaluating its performance.

This will be achieved through successful attainment of the following objectives:

1. Review related literature in conjunction with previously suggested improvements to the autoESDA Python library.
2. Define requirements based on suggested improvements to the autoESDA Python library.
3. Design and implement solutions that address the identified requirements.
4. Evaluate the autoESDA library in terms of the defined requirements.
5. Based on the results, draw conclusions regarding the success of autoESDA as a means of automating the ESDA workflow.

1.5 Research Methodology

Design science methodology refers to research which leads to the creation of successful artifacts, which is achieved through problem identification, definition of a solution, design and development, demonstration, evaluation, and communication (Peffers et al., 2007). This

project can be broken down into each of these stages, and its methodology is therefore considered to be of design science nature.

The first stage of the project was to review existing literature relating to the project. This laid the foundation on which the motivation for a Python library to automate ESDA for vector and raster datasets is based. The existing autoESDA library was then described, along with some feedback from interviews that were previously conducted as part of existing research. This feedback led to the design of solutions to address multiple (mainly minor, cosmetic) improvements that were identified by interview participants. The most notable improvement is the support for raster datasets. Numerous strategies for automating ESDA (specifically local indicators of spatial autocorrelation - LISA) on raster datasets are discussed and compared. The optimal strategy is then included in the autoESDA library. In order to easily implement the upgrades, the existing autoESDA code was first refactored into a new architecture. The performance of the vector and raster modules of autoESDA are then evaluated with datasets of varying sizes and complexities. These results were used to evaluate the success of implementing a library to automate the ESDA process. Finally, limitations of this research were described, as well as opportunities for future work.

1.6 Significance of Research

Just as with EDA, ESDA is a repetitive and time consuming process (Higgins and Ray, 2022). Its results need to be generated through user interaction which opens up the risk of human-induced errors (Borlongan et al., 2016; Murtiyoso et al., 2020). The autoESDA library aims to address these concerns and as such, any improvements to the library will have the same effect. Additionally, the added support for raster datasets is unique in that we are not aware of a tool that automates ESDA for this data format. These improvements and the design decisions behind them are documented and discussed as part of this dissertation which could be used as a guide for researchers conducting other projects – whether they are similar in nature to this one, or whether they are based on expanding this research and/or the autoESDA library.

1.7 Overview of Chapters

The remaining chapters in this dissertation are structured in the following manner:

Chapter 2: Literature Review

This chapter addresses objective one by presenting a body of literature that summarises and links together fundamental concepts related to automating the ESDA process. Spatial data is described, and an overview of geospatial big data is given. EDA and ESDA are identified as

strategies that address some issues brought about by geospatial big data. Use-cases for raster ESDA is discussed, as well as the motivation to automate ESDA. Similar automation projects are discussed to identify principles that could be adapted to automate ESDA. This is rounded off by a discussion on how easily some popular ESDA functions could be automated.

Chapter 3: Towards an Open Source Library for Automated Exploratory Spatial Data Analysis (ESDA)

This chapter was published as an article in 2022 and documents the initial design of the autoESDA library and the feedback and suggested improvements that resulted from a series of interviews. These suggestions motivate for the updates that form the second iteration of autoESDA which is described in Chapter 4.

Chapter 4: Second Iteration of autoESDA: Redesign and Expanding its Capabilities

This chapter begins by documenting some high-level requirements for the second iteration of autoESDA. The remainder of this chapter documents and discusses how these requirements are met through the development of the updated library. The chapter concludes by highlighting some limitations, and paving the way for future improvements that could be implemented.

Chapter 5: Performance Evaluation

This chapter details the process taken to evaluate the performance of the autoESDA library, along with the implemented improvements. Vector and raster datasets of varying sizes and complexities were used the test the performance of autoESDA. The framework for how these tests were run is described, and the results obtained from this process are discussed.

Chapter 6: Conclusion

This chapter summarises the preceding chapters by discussing the extent to which each of the objectives were achieved. Final conclusions are drawn based on the work done, and further work is proposed.

CHAPTER 2: LITERATURE REVIEW

2.1 Chapter Overview

The literature review is intended to highlight the fundamental components on which this dissertation is based. Spatial data is described to introduce the concept and why it is different from *regular* data. This provides the foundation on which the concept of geospatial data is built, which highlights the need for automation and new strategies to be developed so that valuable insight can be efficiently extracted from spatial data. EDA and ESDA, along with their components are discussed as a means of addressing this challenge. Finally, similar work within the realms of dealing with geospatial big data, and the use of automation of other tasks related to the data life cycle, is discussed.

2.2 Spatial Data

While the terms spatial and geospatial data are often used interchangeably, they refer to slightly different concepts. Spatial data refers to any data linked to a point in space, this could include a Cartesian plane or an alternate or fictional universe, whereas geospatial data refers to data that relates to a point on or near the earth's surface (Longley et al., 2015).

Tobler's (1970) First Law of Geography states that *"everything is related to everything else, but near things are more related than distant things"*. While not all spatial phenomena may adhere to this law, it does encourage one to exercise caution when working with spatial data. The law implies that spatial data is dependent on its surroundings, which means that it cannot be processed under the assumption of independence, as is the case for other statistical datasets (Anselin, 1989).

As a subset of *ordinary* data, geospatial data is not exempt from the growing popularity of big data and it has not taken long for researchers to coin the term *geospatial big data* (Lee and Kang, 2015; Li et al., 2016; Robinson et al., 2017). Just as with big data, geospatial big data can also be described using the 3Vs – having large **v**olume, high **v**elocity, and great **v**ariety (Jin et al., 2015). The challenges of dealing with geospatial big data have already been identified (Anselin, 2010; Tsou, 2015), the rest of this section will discuss the 3Vs and their associated challenges within the context of geospatial big data.

Volume

Singleton and Arribas-Bel (2021) found that there has never been access to a greater amount of geospatial data than we have today. This is partly due to the velocity (and variety) at which we capture data, as well as the added temporal dimension due to the historical data we have

accumulated over time (Robinson et al., 2017). These factors both highlight the magnitude of the volume of geospatial data we have at our disposal.

Before one can use raw geospatial big data to assist with decision making, it needs to be processed into information. This is achieved through the use of descriptive statistics (Li et al., 2016) and visualisations which aid the end user in detecting various patterns and trends within their dataset (Coetzee and Rautenbach, 2017). Visualising geospatial big data is not a simple task as data often overlaps due to the various scales and levels of detail in which it is captured (Jern et al., 2008; Li et al., 2016). This makes it a challenge to develop effective solutions to visualise results, as the required visualisation can vary for per usage. Once computed, statistics and visualisations should be structured in an appropriate manner – such as a report. This will allow for additional insight into the data by the end user (Li et al., 2015; Robinson et al., 2017). Anselin (2010) highlights the need for software to be developed that is capable of handling the large volumes of spatial data.

Velocity

Internet and GPS enabled sensors are more accessible than ever. This is due to declining costs as well as integration into everyday devices such as smartphones (Armstrong et al., 2019). Due to the widespread use of these sensors, geospatial data is increasingly available in real time. At the same time, multiple satellites consistently stream other geospatial data such as reflectance values, temperatures, and air quality measurements (Armstrong et al., 2019.

The challenge no longer lies in collecting geospatial data, but rather in how timeously it can be processed. Lee and Kang (2015) argue that data should be processed as soon as it is collected. This massive influx of data changes frequently and the results need to be viewed almost immediately so that decisions can be made in real time (Dangermond and Goodchild, 2020). This argument is certainly applicable in certain situations such as dealing with immediate natural disasters. Despite the magnitude of big data, not all big data is immediately useful (Jin et al., 2015). Li et al. (2016) cautions that sensor data especially is largely irrelevant due to repetition and over sampling. Just as data may need to be processed immediately for real time decisions, it may be more beneficial to analyse temporal trends and investigate outliers. Regardless of whether data should be processed immediately at the time of collection or if trends are analysed at set time intervals, it is evident that the workflow followed will be repeated numerous times.

Variety

Cressie (1993) outlines three spatial data models, namely: point pattern data, geostatistical data, and lattice data. Point patterns are used to represent point events. Consequently, the value of these datasets lie in the location/pattern of the point(s), rather than the value of a variable at these points. As such, the calculation of spatial autocorrelation of point patterns is of little value (Cressie, 1993). The primary objective of geostatistics is to predict the value of a variable at unknown locations. Conversely, the goal of a lattice-based approach is to detect spatial patterns and find an explanation for these patterns (Saveliev et al., 2007).

A lattice dataset is created by dividing up a larger study area into a collection of smaller areas in a neighbourhood structure, each with an associated observation (Kaluzny et al., 1998; Saveliev et al., 2007). These smaller areas (also known as cells, units, or locations) do not overlap and need to share a common boundary. A regular lattice is made up of cells that have the same shape and size. Satellite data such as temperature or reflectance values are an example of data gridded in a regular lattice. Irregular lattices, on the other hand are formed by irregular shaped units such as political/administrative boundaries or Voronoi polygons (Kaluzny et al., 1998; Saveliev et al., 2007).

Statistical analysis of lattice datasets are carried out to detect and explain spatial patterns (Saveliev et al., 2007). These objectives are by definition part of an ESDA workflow. Just as with the popular ESDA software, GeoDa[1] (Anselin et al., 2006), this dissertation will primarily focus on lattice data. This means that any outcomes or statements may not hold the same truth when working with point pattern or geostatistical datasets.

Most exploratory tools for geospatial big data are designed to handle specific (mainly vector) formats (Robinson et al., 2017). In order to capture the full value of the large variety of formats, software needs to support a variety of data formats.

2.3 Exploratory Data Analysis (EDA)

Tukey (1977) coined the term Exploratory Data Analysis and likened it to "*quantitative detective work*", emphasising that it is purely exploratory rather than confirmatory (Fotheringham, 1992). Peng et al. (2021) describes EDA as *"the process of understanding data through data manipulation and visualisation"*, emphasising that it plays a vital role in every data science project. While it is clear that EDA can never tell the whole story of a dataset, it is a valuable tool for gathering initial insights to the dataset at hand. Anselin (1989) describes EDA as a data-driven approach, in which one must allow the data to speak for itself. One

[1] http://geodacenter.github.io/

should be allowed to freely choose which functions are used so that the data may be viewed from different perspectives (Morgenthaler, 2009). This means that there is no generic algorithm of EDA functions that can be applied to all datasets, as the results will be largely dependent on the individual dataset. Functions need to be executed without any underlying assumptions of the dataset, and the resulting visualisations, indicators and statistics should guide how the user further explores and analyses the data. Unlike with the more specialised ESDA, there is a great variety of proprietary and open source software platforms supporting EDA. There are also multiple libraries within the R and Python ecosystems that support EDA. Due to there being no clear boundary between EDA and other statistical research, there exists no authoritative list defining what functions fall within the EDA toolbox. There are, however, several popular techniques associated with EDA. These will be discussed in the remainder of this section.

2.3.1 Descriptive Statistics

Descriptive statistics are used to outline important features of a dataset (Devore and Berk, 2012). These measures can provide important initial conclusions related to the dataset and are fundamental to any EDA workflow. There are four categories of descriptive statistics, namely: number of observations, measures of central tendency, measures of variation, and measures of shape (Myatt and Johnson, 2014).

A dataset is made up of a collection of **observations**. The observations for random variable X can be defined as $x_1, x_2, x_3 \dots x_{n-1}, x_n$. The **sample size ($n$)** of a dataset is the count of how many observations are in the dataset, thus indicating the size of the dataset.

2.3.1.1 Measures of Central Tendency

Mode

The mode is the most frequently occurring value in a dataset. It can highlight errors or potential trends within a dataset to be further investigated.

Median

The median is the middle value of a dataset that has been arranged in ascending order. It is used as a measure of central tendency as it is not skewed by outliers.

Sample Mean ($\bar{x}$)

The sample mean gives an indication of what an average/expected value of the dataset would be. It is greatly influenced by outliers. The formula for the sample mean is given as:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$

where x_i is an observation of random variable X.

2.3.1.2 Measures of Variation

Minimum and Maximum

The minimum and maximum are the extreme values in the dataset. They refer to the smallest and largest value respectively.

Range

The range gives an indication of the spread of values in a dataset. It is calculated by subtracting the minimum value from the maximum value in the dataset. A larger range is indicative of a greater spread of values.

Quartiles

$$Q1 = 0.25(n+1) \qquad Q2 = 0.5(n+1) \qquad Q3 = 0.75(n+1)$$

$Q1$, $Q2$, and $Q3$ refer to the lower, middle, and upper quartiles respectively. $Q1$, $Q2$, and $Q3$ indicate the position of the values in the dataset (when arranged in ascending order) that divide the dataset into four quartiles.

Sample Variance (s^2)

$$s^2 = \frac{\sum(x_i - \bar{x})}{n-1}$$

The variance is a measure of the spread of the dataset by measuring how values differ from the mean. The greater the value for variance, the greater the spread of the dataset. Datasets with a greater variance are considered to have more noise as the values are further from the mean than datasets with a lower variance.

Sample Standard Deviation (s)

$$s = \sqrt{s^2}$$

Standard deviation is the square root of variance. The greater the value for standard deviation, the greater the spread of the dataset. The statistic has a similar interpretation to that of variance, in that a greater standard deviation is indicative of a dataset with more noise as the

values are, on average, further away from the mean than datasets with a lower standard deviation.

2.3.1.3 Measures of Shape

Skewness ($\tilde{\mu}_3$)

The skewness is determined as

$$\tilde{\mu}_3 = \frac{\sum(x_i - \bar{x})^3}{s^3(n-1)}$$

where s is the sample standard deviation.

A skewness value ranges from -1 to +1. Its value corresponds to how symmetric the dataset's distribution is. A value of 0 indicates a symmetric distribution, while a positive and negative value represents a positive and negative skewed dataset respectively.

Kurtosis ($\tilde{\mu}_4$)

$$\tilde{\mu}_4 = \frac{n-1}{(n-2)(n-3)}\left((n+1)\left(\frac{(\sum_{i=1}^{n}(x_i - \bar{x})^4)^{n^{-1}}}{((\sum_{i=1}^{n}(x_i - \bar{x})^2)^{n^{-1}})^2}\right) - 3\right) + 6$$

A kurtosis value represents the shape of the peak of a distribution. Values close to zero indicate an approximate normal distribution shape, while positive and negative values represent sharper and flatter peaks respectively.

2.3.1.4 Measures of Association

Pearson Correlation (r_{xy})

Pearson's correlation coefficient is calculated as

$$r_{xy} = \frac{\sum\big((x_i - \bar{x})(y_i - \bar{y})\big)}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}$$

Where x_i and y_i refer to an observation, and $\bar{x}$ and $\bar{y}$ refer to the sample mean of random variable X or Y respectively.

Pearson is the most popular correlation coefficient. It measures the linear relationship between two variables. The variables must be quantitative (interval or ratio measurements) and have a normal distribution. Values range from -1 to +1. A value of 0 represents no relationship, a

value of +1 represents a perfect positive linear relationship and a value of -1 indicates a perfect negative linear relationship.

Spearman Correlation (ρ)

Spearman's ρ is also referred to as rank correlation. It does not require variables to follow a normal distribution. Variables can be ordinal, interval or ratio measurements. Values range from -1 to +1. A value of 0 represents no relationship, a value of +1 represents a perfect positive relationship and a value of -1 indicates a perfect negative relationship. In order to calculate Spearman's ρ, the observations are first ranked from the smallest value to the largest value (Zuur et al., 2007). The rank of each observation is then compared to the rank value of the corresponding observation from the other variable. These distances are then used in the formula below:

$$\rho = 1 - \frac{6 \sum d_i^2}{(n^3 - n)}$$

where d_i is the difference between the x_i and y_i variable ranks, and n refers to the sample size.

Kendall Correlation (τ)

Kendall's τ is used for measuring association of the ranks between pairs of variables. Values range from -1 to +1. A value of 0 represents no relationship, a value of +1 represents a perfect positive relationship (the ranking of observations is the same for both datasets). and a value of -1 indicates a perfect negative relationship (perfect disagreement of rankings). Values are first ordered and then ranked from 1 to n. A pair of observations is considered a concordant pair if the differences of their value and the value of another observation for that variable are in the same direction as the difference of the corresponding observations of the other variable. Conversely, a discordant pair occurs when the difference is in the opposite direction. If the difference between the observations are the same for both variables, it is considered a tie. The formula below is used to calculate Kendall's τ:

$$\tau = \frac{n_c - n_d}{\sqrt{(n_c + n_d + t_x)(n_c + n_d + t_y)}}$$

where n_c is the number of concordant pairs, n_d is the number of discordant pairs, t_x is the ties in variable X and t_y is the ties in variable Y.

2.3.2 Visualisation Techniques

The use of visualisation techniques is an effective method of aiding ones understanding of the distribution of a dataset (Myatt and Johnson, 2014). This section will describe some popular visualisation techniques that one may come across when conducting EDA.

2.3.2.1 Brushing

Although not strictly an EDA function, brushing, or subset selection, is considered an important part of any EDA process as it aids the user in identifying spatial and statistical relationships in a dataset (Ward, 1994). Brushing, often accompanied by multiple linked windows (Haslett et al., 1990; Jern et al., 2007; Roberts, 2005), refers to the ability of the user to select only a subset of observations of a dataset for consideration in other statistics or visualisations that form a part of EDA (Ward, 1994). This allows the user to interact with the dataset and observe the effect of excluding certain observations, or only considering a subset of the dataset (Rey and Janikas, 2006).

2.3.2.2 Box Plots

Box plots, also referred to as box and whisker diagrams, are used to visualise the shape of a dataset (Morgenthaler, 2009). The box (illustrated in Figure 1) usually spans the interquartile range (IQR) which is the range between the first quartile (25th percentile) and third quartile (75th percentile). The median (50th percentile) is displayed in the box and its position will indicate the extent to which the dataset is skewed. The IQR is the length of the box and can be calculated by subtracting the 25th percentile (Q1) from the 75th percentile (Q3). Two points are plotted at the upper and lower limits. A line (whisker) is then drawn connecting these points to the edge of the box. Any values falling outside the limits are also plotted, these are referred to as outliers.

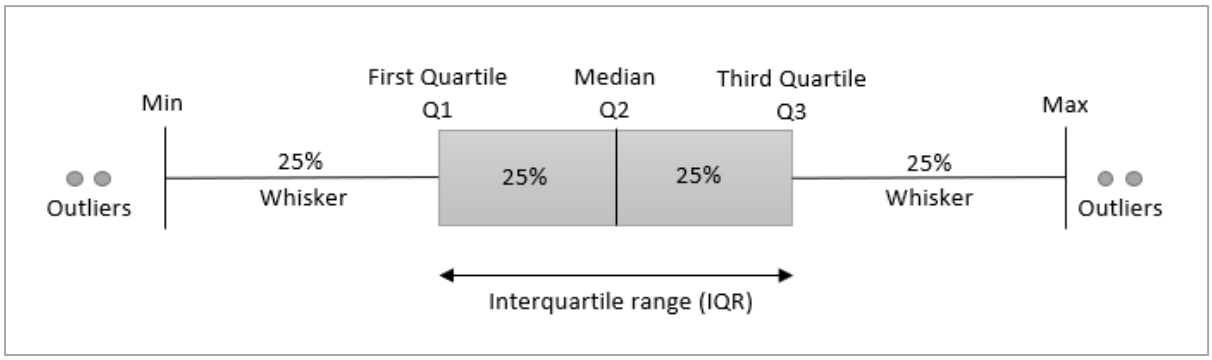


Figure 1: Box plot (also known as a box and whisker diagram)

2.3.2.3 Frequency and Probability Histograms

The purpose of a histogram is to visualise the distribution of a dataset. This is done by arranging values into bins which are represented as bars on the histogram. The user determines the number and range of the bins. Bars on the histogram would represent each of the bins. A frequency histogram will use count to represent the height of the bars, whereas a probability histogram (also called a relative frequency histogram) will use probability to

represent the height of the bars (Myatt and Johnson, 2014). Figure 2 illustrates a frequency and probability histogram respectively.

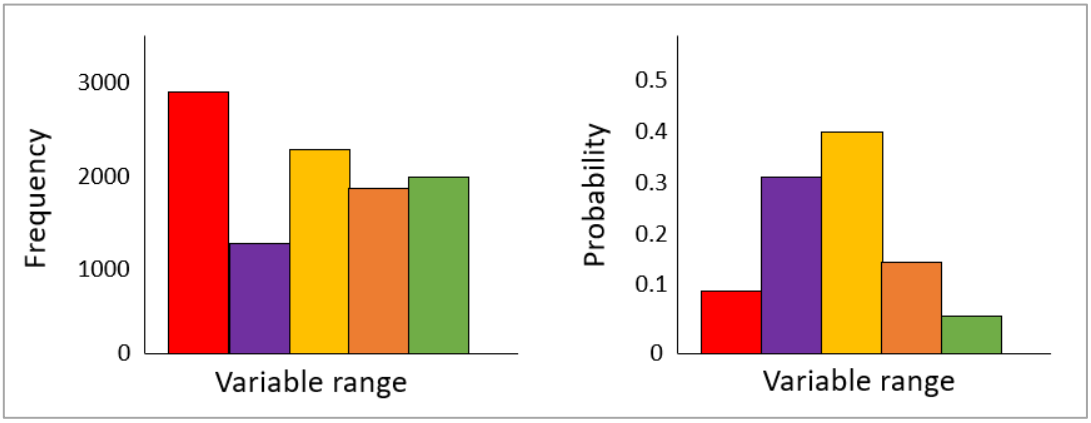


Figure 2: Frequency (left) and probability (right) histogram.

2.3.2.4 Scatter Plots

A scatter plot is used to visualise the relationship between variables. The most common form of scatter plot is a 2D scatter plot, which plots two variables against each other on a Cartesian plane. One variable is plotted on the x-axis and another variable on the y-axis. Additional variables and axes may be added to form 3D or multidimensional scatter plots. It is a common practice to create a scatter plot matrix (Figure 3) in which a scatter plot is generated between each variable in a dataset. The diagonal is then filled with the frequency histogram corresponding to each variable.

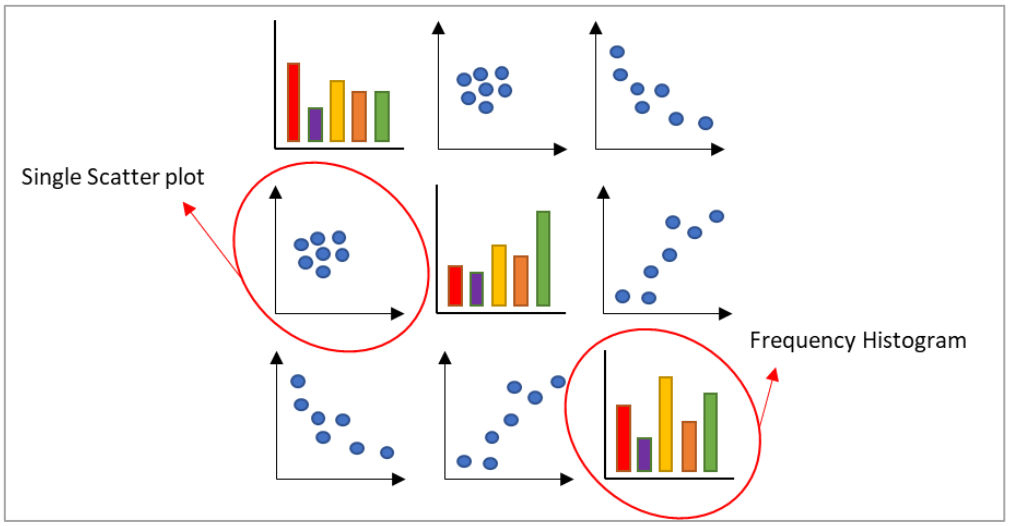


Figure 3: Pairwise plot made up of scatter plots and frequency histograms

2.4 Exploratory Spatial Data Analysis (ESDA)

Most EDA functions assume independence which means that they are not suited for spatial datasets (Anselin, 1989). This creates a gap which is addressed by Exploratory Spatial Data Analysis (ESDA), which is an extension of EDA. This means that EDA can be considered as part of ESDA (Anselin and Rey, 2012), however the converse is not necessarily true. Anselin (1996) defines ESDA as a collection of tools used to describe and visualise spatial distributions, and any patterns that they may contain. This is achieved through the identification of spatial outliers, clusters, and other forms of spatial instability or spatial non-stationarity.

While objectives of ESDA remain the same as that of EDA, Goodchild (1992) argues that the value obtained from ESDA may even be greater than that of regular EDA. Just as with EDA, ESDA takes place after data cleaning in what is known as the exploratory phase (Anselin and Getis, 1992; Anselin and Rey, 2012). It aims to use all available data with minimum previous filters, selection criteria, and hypotheses and the outcomes of ESDA will guide how the dataset is further processed (Goodchild, 1997; Openshaw, 1995). As a data-driven approach, ESDA should impose as little structure as possible and use simple indicators to identify patterns and clusters (Anselin 1996; Anselin et al., 2007; Steiniger and Hunter, 2013).

There are two spatial effects which ESDA should highlight, namely spatial heterogeneity and spatial autocorrelation (also known as spatial dependence) (Anselin 1998; Goodchild, 2000; Dall'erba, 2009). These two effects and their associated functions will be discussed in Section 2.4.1 and 2.4.2 respectively. Section 0 will discuss some of the popular software platforms used for ESDA.

2.4.1 Spatial Heterogeneity

It is widely accepted that phenomenon are not homogeneous (similar) across space, which supports Tobler's First Law in that near things are more related than distant things (Anselin, 1988). This is known as spatial heterogeneity and is what makes each point in space unique, due to the fact that it's attributes may differ from those at other locations (Goodchild and Longley, 2013). Various types of trends/patterns can occur, such as regions of similar values, or regions with great variance. Trends that occur in a dataset could be because the dataset is an accurate depiction of reality, however one should exercise caution as a trend could also be incorporated by imposing a spatial structure not suitable to that phenomenon (Anselin, 1988). It is therefore important that these patterns are investigated, as these findings could guide the user on how to further process a spatial dataset. Box plots, choropleth maps, and cartograms are all techniques used to visualise spatial heterogeneity (Dall'erba, 2009).

2.4.1.1 Choropleth Maps

A choropleth map is a thematic map which represents the distribution of a variable by shading/colouring non-overlapping areas (such as provinces, wards, or raster cells) with different intensities according to the value of the attribute for that area (Longley et al., 2015; Slocum et al., 2014). Choropleth maps are vital tools in ESDA as they give an elementary visual understanding of the spatial distribution of a variable. There are numerous classification methods used to divide a dataset into classes, with each method having different advantages and disadvantages and their effectiveness being dependent on the distribution of the values in the dataset. Slocum et al. (2014) identified some popular classification schemes which are described in Table 1. These are also visualised in Figure 4.

Table 1: Common classification schemes for choropleth maps

Classification	Description (Slocum et al., 2014)
Boxmap	Useful to highlight extreme values. Data is divided into six categories, namely the four quartiles, and then one each for extreme low and extreme high outliers respectively.
Equal Intervals (Equal Steps)	This method divides the observations in the dataset into a user-specified number of classes, with each class having the same interval.
Quantiles	In this method observations are arranged in ascending order and divided so that the same number of observations are present in each user-specified class.
Mean-standard deviation	This method forms classes by incrementally adding or subtracting the standard deviation from the mean of the data in accordance with the number of classes the user has specified.
Maximum Breaks	In this method data is ordered in ascending order and the difference between each observation is calculated. The greatest differences are then used as breaks between a user-specified number of classes.
Fisher-Jenks	This method uses statistical optimisation to minimise the sum of absolute differences between class medians. The number of classes are user-specified.

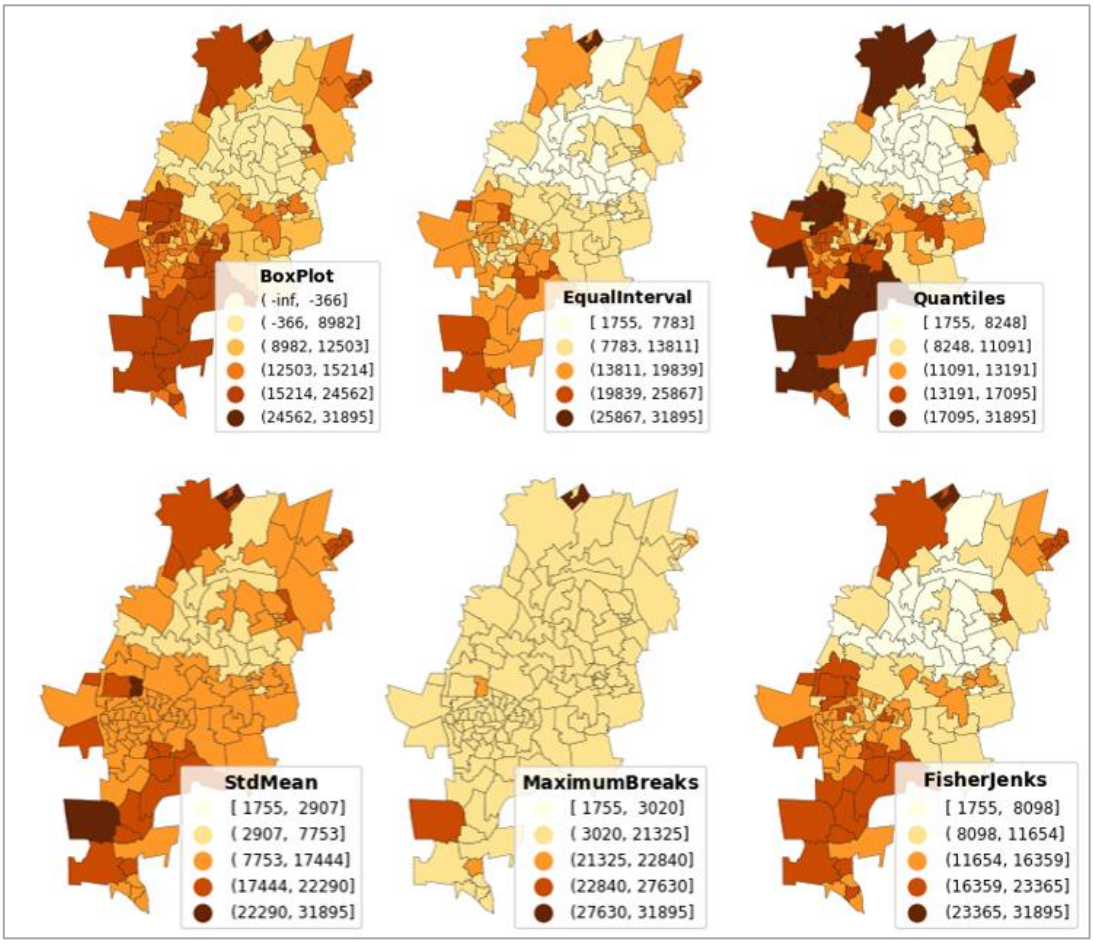


Figure 4: Choropleth maps with different classification schemes

2.4.1.2 Cartograms

A cartogram is used to visualise the magnitude of the variable of interest. It achieves this by distorting the size of the area in a way that is proportional to the target variable (Anselin et al., 2007). This makes it easier to emphasise smaller enumeration areas that have large values and larger enumeration areas that have small values. Traditionally, the shape of the enumeration areas are maintained, however simple identical shapes (circles are used for Dorling cartograms) have been used instead so to allow for more detail to be displayed (Slocum et al., 2014). In addition to using size as a visual element, one can also colour the enumeration areas according to an attribute value. Figure 5 illustrates a regular cartogram with a single colour on the left, whereas the Dorling cartogram on the right has a colour scheme applied based on attribute values.

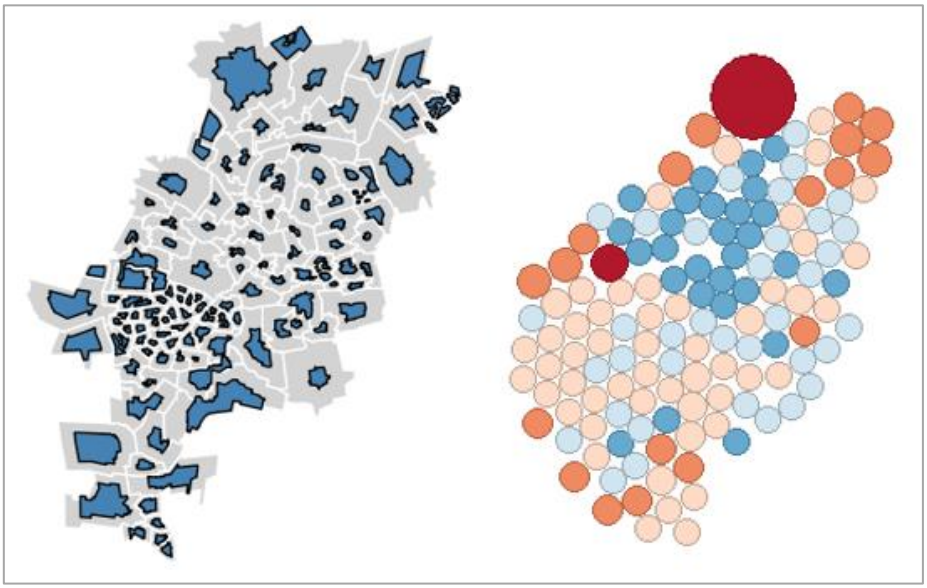


Figure 5: Regular cartogram (left) and a Dorling cartogram (right)

2.4.2 Spatial Autocorrelation

Slocum et al. (2014) define spatial autocorrelation as "*the tendency for like things to occur near one another in geographic space*". One may even argue that spatial autocorrelation measures are used to quantify Tobler's First Law. Positive spatial autocorrelation refers to clusters of observations where the observations share similar values (perfect clustering). This could be concentrations of high values (also known as hotspots) or low values (also known as coldspots). Negative spatial autocorrelation refers to the dispersion of variables in which there are no clusters of high or low values (perfect dispersion). Datasets may also exhibit no spatial autocorrelation, this is referred to as complete spatial randomness (Dall'erba, 2009).

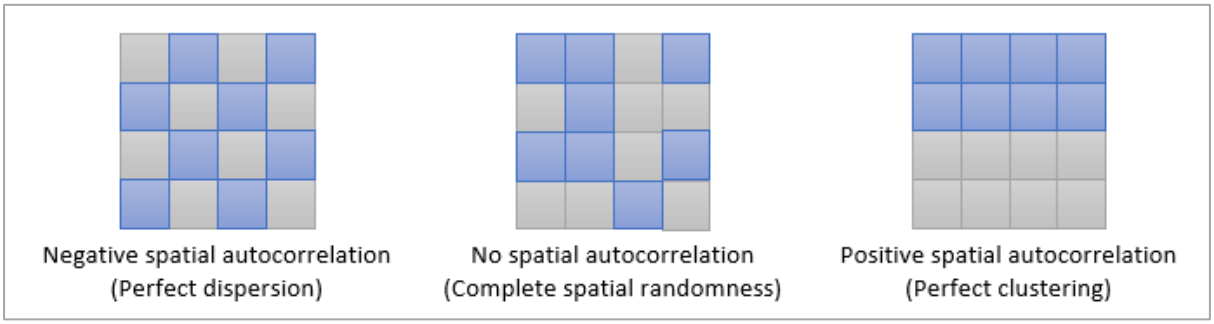


Figure 6: Examples of patterns of spatial autocorrelation

Global spatial autocorrelation refers to spatial autocorrelation in the dataset as a whole. Global spatial autocorrelation statistics only indicate overall clustering or dispersion, but not where these hotspots/coldspots occur (Anselin et al., 2007). Local indicator(s) of spatial

autocorrelation (LISA) are therefore used to address this gap as they identify significant local clusters or outliers (Anselin and Bao, 1997).

There are various measures of spatial autocorrelation. Join count statistics are used for binary data, which are usually nominal values (De Smith et al., 2018). Moran's I and Geary's C are global autocorrelation measures used for numeric data in lattice format (Anselin, 1988). Each measure also has an equivalent LISA – local Moran's I and local Geary's C (Anselin, 1995). The Getis and Ord G statistics may also be used to quantify spatial autocorrelation on a global and local level (Dall'erba, 2009).

Univariate Moran's I and Geary's C are the original and most popular global autocorrelation statistics (Anselin, 1988; Dall'erba, 2009). These statistics, along with their LISA will be discussed in the following sections.

2.4.2.1 Spatial Weights

A spatial weights matrix is used to determine the neighbourhood of an observation. The identified neighbours are included in the calculation of the spatial lag of a variable, and by extension, spatial autocorrelation statistics such as Moran's I or Geary's C (Anselin, 1998). Spatial weights can be defined by whether the observations share boundaries (contiguity-based) or whether they are within a certain distance of each other (distance-based) (Anselin, 1998). Rook and queen are both cases of contiguity strategies, whereas kernel, distance-band and k-nearest neighbours are examples of distance based strategies (Anselin et al., 2006; Dall'erba, 2009). By definition, lattice data share common boundaries, making contiguity-based approaches the most commonly used method to define neighbours (Anselin, 1998). Their popularity may be because they are less complex when compared to their distance-based counterparts due to the subjectivity associated with the distance parameter. A single distance metric cannot be used generically as density of a phenomena is highly dependent on the nature of that phenomena. Depending on the projection of the dataset, the units used to represent distance in each dataset could differ. For this reason, this section will only discuss contiguity-based weights as they are simpler than their distance-based counterparts, whose creation is more of a challenge to automate.

A rook's case approach defines a neighbour as an observation that shares a border with the target observation (Anselin and Rey, 2014). A queen's case, however, would consider all observations that share a vertex and/or border with the target observation as a neighbour (Anselin and Rey, 2014). In addition to the type of contiguity weights, one may also specify the order of a spatial weights matrix. The order refers to the number of steps of adjacency that are included in the identification of neighbours (Anselin and Rey, 2014). An example of

queen's case and rook's case contiguity is illustrated in Figure 7, where first order neighbours are shown in dark purple, and pink is used to represent cells that would only be considered if the order of contiguity is two.

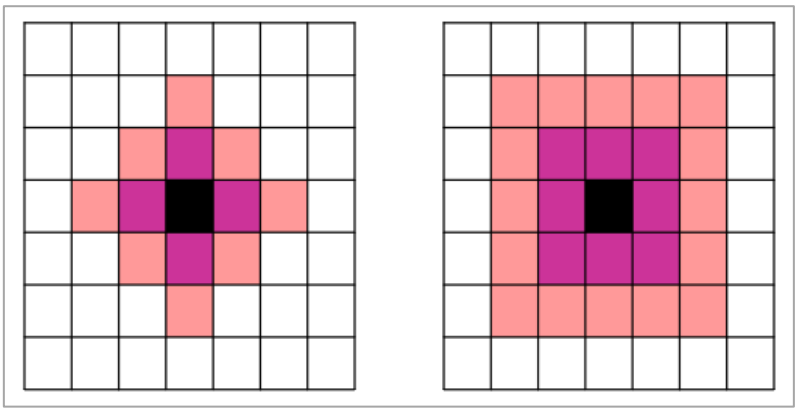


Figure 7: Rook (left) and queen (right) case contiguity.

A spatial weights matrix can be represented as matrix $W = [w_{ij}]$, where w_{ij} represents the relationship between locations i and j. Values in a contiguity-based matrix can either be 0 or 1, with 0 indication i and j are not neighbours, while a value of 1 would indicate that i and j are neighbours (Rogerson and Kedron, 2012).

There is no agreement on which spatial weight format is the best to use, and Anselin (1988) argues that the structure of the spatial weights should be chosen based on the nature of the phenomenon represented in the dataset. Rey et al. (2023), however, state that the distinction between queen's and rook's case neighbours is negligible when working with an irregular lattice rather than a grid. The reason for this is because the shape of the cells in the dataset would have more of an impact on the defined neighbours than the strategy used.

2.4.2.2 Moran's I

The Moran's I statistic is the most popular method to represent spatial autocorrelation (Anselin et al., 2007; De Smith et al., 2018). Its formula is shown in Table 2. Moran's I values range from -1 to +1, with -1 indicating negative spatial autocorrelation (perfect dispersion) and +1 representing positive spatial autocorrelation (perfect clustering). Values close to zero are indicative of no clustering (complete spatial randomness). A Moran's I statistic can be visualised using a Moran's I scatter plot (see Figure 8), in which the spatial lag (y-axis) is plotted against the standardised value (z-score) of an observation (x-axis). The gradient of this scatter plot will be equivalent to the value of Moran's I, meaning that a steeper the line, will be associated with a greater degree of spatial autocorrelation (Anselin, 1996; 1998).

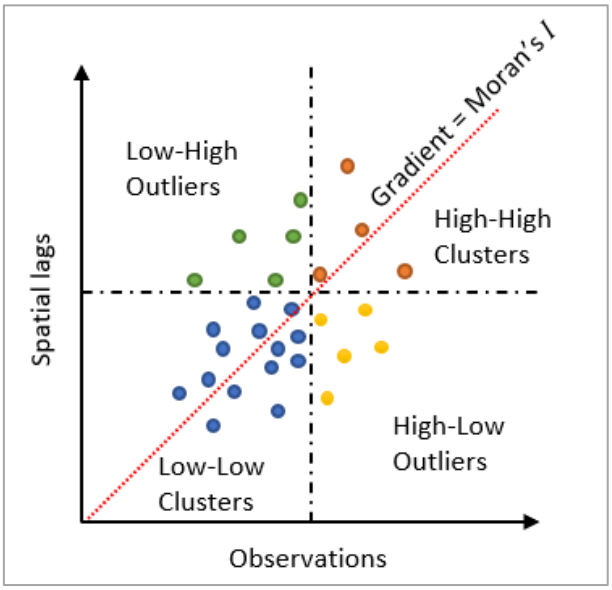


Figure 8: Moran's I scatter plot

The Moran's I scatter plot (Figure 8) is a useful tool for identifying outliers and can be decomposed into four quadrants. These quadrants form the foundation on which the local Moran's I categories are based (Anselin, 1996; Anselin and Bao, 1997). Two of these quadrants represent local clusters (high-high and low-low), while two quadrants represent local outliers (high-low and low-high). These quadrants form the four classes that are present in a LISA cluster map. While the scatter plot quadrants do provide a means for classifications, one needs to be cautioned that this does not indicate significance (Anselin et al., 2004). The local Moran's I value (I_i) can also be calculated using a formula derived from the global statistic. This formula is also shown in Table 2.

2.4.2.3 Geary's C

The Geary's C statistic follows a similar approach to the Moran's I formula, however instead of computing the cross product of standardised values, it uses the sum of squared differences instead (Geary, 1954). Warner and Shank (1997) describe global Geary's C as the average dissimilarity between points. Due to the fact that it uses the sum of squared distances, Geary's C is less sensitive to linear relationships and could potentially identify spatial autocorrelation where Moran's I may not (Anselin, 2019). The interpretation of Geary's C values are different to those of Moran's I. Values range from 0 to +2, those less than +1 are indicative of positive spatial autocorrelation, while values larger than +1 indicate negative spatial autocorrelation.

Table 2: Comparison of Moran's I and Geary's C

	Moran's I	**Geary's C**
Global Formula	$I = \frac{n}{S_0}\frac{\sum_i \sum_j w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_i (x_i - \bar{x})^2}$	$C = \frac{(n-1)}{2S_0}\frac{\sum_i \sum_j w_{ij}(x_i - x_j)^2}{\sum_i (x_i - \bar{x})^2}$
	where: • n = sample size, • w_{ij} = connectivity spatial weight between units i and j, • x_i = observation in region i • x_j = observation in region j • $S_0 = \sum_i \sum_j w_{ij}$	
Range	(-1, +1)	(0, +2)
Interpretation	I < 0 → Dispersion I = 0 → Randomness I > 0 → Clustering	I < 1 → Clustering I = 1 → Randomness I > 1 → Dispersion
Local Formula	$I_i = \left(\frac{x_i - \bar{x}}{\frac{\sum_i (x_i - \bar{x})^2}{n-1}}\right)\sum_j w_{ij}(x_j - \bar{x})$	$C_i = \sum_j w_{ij}(x_i - x_j)^2$
	where: • x_i and x_j are the observations in region i and j respectively, and • w_{ij} is the connectivity spatial weight between regions i and j	

2.4.2.4 Significance Tests

The global and local versions of Moran's I and Geary's C all indicate the presence of clustering, however they yield no measure of their significance. A permutation test is often used to address this shortfall. These tests are based on standardised z-values calculated from a permutation approach. The permutation approach (Anselin, 1995) calculates the statistic for each of a number of randomly simulated datasets. The number of simulations is dependent on the number of permutations (usually 999). The value calculated from the actual dataset is then compared to those obtained from the simulated datasets. This comparison allows for a significance value to be calculated, which usually accompanies the final statistic (Dall'erba, 2009). The permutation approach brings with it a computational challenge due to the volume of calculations involved, which is notable in large datasets (Anselin, 1999).

2.4.3 ESDA Software

This section will describe some of the popular software platforms that support ESDA on lattice data. A suitable ESDA tool is described as being capable of detecting recurring and isolated patterns without knowing in advance what to expect (Openshaw, 1995). There are multiple platforms that support ESDA (to varying extents), and this section will discuss GeoDa, PySAL[2] and spdep[3] – each of which are popular open source platforms. These are not the only platforms that support ESDA, however only these three will be discussed. Table 3 is found at the end of this section and summarises the ESDA functions that are supported by the discussed platforms.

2.4.3.1 GeoDa

GeoDa is an upgrade of the SpaceStat[4] platform brought about through the need for linked views, improved visualisations, and the popularity of larger datasets (Bivand, 1998; Anselin et al., 2006; Rey and Anselin, 2006). A beta version of GeoDa was made available in 2002, however it was only released as a complete open-source and cross-platform software tool in 2011 (Anselin, 2000; Anselin et al., 2006; 2022). It was developed to address the lack of an easy to use, visual, and interactive platform targeted at non-GIS users. GeoDa's design is centred around multiple, linked views that allow the use of brushing to interact with the data. This makes it quite a valuable tool for teaching ESDA. The interface is *point and click* based, requiring the user to have no programming knowledge. One of the major factors contributing to GeoDa's success is that it has no dependency libraries. It can be downloaded as a standalone tool and is licensed under the open source GPL 3.0 license (Anselin, 2012).

When describing its functionality, Anselin et al. (2006) breaks it down into six categories – spatial data manipulation and utilities, data transformation, mapping, EDA, spatial autocorrelation, and spatial regression. This illustrates that GeoDa's functionality goes much further than that of only ESDA.

The rgeoda[5] R package and pygeoda[6] Python library were both developed to allow users of each of these platforms to interface with the libgeoda library which is written in C++ (Anselin et al., 2022). While the libgeoda library is based on the original GeoDa platform, it does not currently have all the functionality of GeoDa as it unfortunately does not support global autocorrelation measures. As with GeoDa, pygeoda, rgeoda, and libgeoda do not support

[2] https://pysal.org/
[3] https://cran.r-project.org/web/packages/spdep/index.html
[4] https://biomedware.com/products/spacestat/spacestat-details/
[5] https://geodacenter.github.io/rgeoda/
[6] https://geodacenter.github.io/pygeoda/

raster datasets. This means that any raster dataset would first have to be vectorised before it can be processed using these platforms. Anselin et al. (2022) compared the performance of these tools to that of spdep and PySAL by calculating a local Moran's I statistic, and found that the multithreading, use of the GPU processor, and C++ codebase all enabled the GeoDa-based platforms to outperform spdep and PySAL.

2.4.3.2 PySAL

PySAL was developed as an open source Python library that supports various spatial analytical functions, among these are numerous ESDA functions (Anselin, 2012; Rey and Anselin, 2007). Its design is modular, and all its dependencies are Python based, meaning that it is compatible with most operating systems. PySAL is the foundation on which many desktop software systems have been built, as well as a toolbox for ArcGIS[7], with plans for a QGIS[8] plugin on the way (Rey et al., 2015). Various developments are in the works to ensure that PySAL can efficiently process geospatial big data. These include integration with desktop GIS platforms such as ArcGIS or QGIS, as well as cloud and parallel processing (Rey et al., 2015; Rey et al., 2022).

The design philosophy behind PySAL is that it is lightweight, resulting in its original, monolithic package being divided into separate Python packages that can be downloaded individually (Rey and Anselin, 2007; Rey et al., 2022). The esda package supports the exploratory analysis of spatial data, however, it depends on other packages in the PySAL ecosystem such as the libpysal package for constructing spatial weights matrices, and the mapclassify package for generating choropleth map classes (Rey et al., 2022).

2.4.3.3 spdep

There are numerous R packages available on the CRAN repository which can be freely downloaded if you work with spatial data (Bivand, 2006). The spdep and DCluster[9] packages are both vital for tools for ESDA within the R ecosystem (Anselin et al., 2006). When comparing the documentation of the two packages, it is evident that spdep is the more comprehensive of the two, and as such, it will be the only one discussed in detail.

spdep is similar to the esda PySAL sub-package in its functionality and is thus quite modular in its functionality (Bivand, 2002). One would need to rely on other R packages for certain ESDA functions, such as maptools[10] to generate choropleth maps (Bivand, 2002; 2006).

[7] https://pro.arcgis.com/en/pro-app/latest/get-started/get-started.htm
[8] https://www.qgis.org/en/site/
[9] https://cran.r-project.org/web/packages/DCluster/index.html
[10] https://cran.r-project.org/web/packages/maptools/index.html

Table 3 is a comparison of the ESDA functionality of GeoDa, pygeoda, PySAL, and spdep. It is worth noting that while each platform supports the majority of the functions discussed, GeoDa is the only platform that supports regular EDA, while pygeoda is limited in that it does not support global spatial autocorrelation statistics. Both PySAL and spdep are extendible to include other functionality, however for the purpose of an *as is* comparison, potential support through external libraries has not been included in Table 3.

Table 3: Comparing the functionality of various ESDA platforms.

	GeoDa	pygeoda	PySAL	spdep
EDA				
Brushing	●	○	○	○
Descriptive Statistics	●	○	○	○
Box Plot	●	○	○	○
Frequency Histogram	●	○	○	○
Scatter Plot	●	○	○	○
ESDA				
Choropleth Map				
Boxmap	●	●	● [1]	○
Equal Intervals	●	○	● [1]	○
Quantiles	●	●	● [1]	○
Mean-Standard Deviation	●	●	● [1]	○
Maximum Breaks	●	○	● [1]	○
Fisher-Jenks	●	○	● [1]	○
Cartogram	●	○	○	○
Spatial Weights				
Contiguity (Queen)	●	●	● [2]	●
Contiguity (Rook)	●	●	● [2]	●
Contiguity Order	●	●	● [2]	●
Distance (k-nearest neighbour)	●	●	● [2]	●
Distance (kernel)	●	●	● [2]	●
Distance (distance-band)	●	●	● [2]	●
Global Spatial Autocorrelation				
Moran's I	●	○	● [3]	●
Moran's I Scatter Plot	●	○	● [4]	●
Geary's C	○	○	● [3]	●
Getis-Ord G	○	○	● [3]	●
Local Spatial Autocorrelation				
Local Moran's I	●	●	● [3]	●
Local Geary's C	●	●	● [3]	●
Getis-Ord G_i Statistics	●	●	● [3]	●

*Available through the mapclassify[1], libpysal[2], esda[3], and splot[4] PySAL submodules respectively

2.5 ESDA with Raster Datasets

Section 2.2 discussed how lattice data can be regular or irregular in its structure. Raster datasets are by definition a regular lattice and the way an ESDA workflow is carried out is identical (Shortridge, 2007). There are, however, major technical differences brought about by the differing data structures (Rey et al., 2023). Although raster datasets have not traditionally been used for ESDA procedures, their use has become increasingly popular (Rey et al., 2023).

Local Moran's I has been calculated across raster surfaces to quantify concentration of vegetation index values (Zhou et al., 2021), changes in sand dunes brought about by the removal of vegetation (Walker et al., 2013), as well as the residuals of a regression model (Li et al., 2019). These studies would have either used GeoDa, presumably after first vectorising the raster dataset, or GeoRasters[11] to carry out the calculations.

2.5.1 Supporting Raster ESDA

Descriptive statistics can be computed for raster datasets with relative ease. ENVI[12], ArcGIS Pro, and QGIS each display various statistics relating to a raster surface without the need for the user to specify anything as the values are either stored in the metadata or calculated immediately upon loading the dataset. The values stored in the raster arrays could also be extracted for EDA visualisations quite easily. The challenge, however, emerges when calculating ESDA statistics.

Rey et al. (2023) explain that conceptually, the calculation of spatial autocorrelation statistics on raster surfaces is very similar to that of an irregular polygon-based lattice, however from a technical perspective, the approach for each format would differ significantly. The main challenge is the creation of the spatial weights matrix to use for spatial autocorrelation.

Currently GeoDa only supports discrete lattice data, and while planned developments aim to include support for flow data (Anselin et al., 2006), as of October 2023, there is no mention of planned support for raster file types in the GeoDa developer notes. The spdep R package allows one to create neighbours from grid cells using the `cell2nb` function and the raster R package allows the user to manually define a weights matrix. According to the spdep GitHub repository, this functionality was created in 2017, however similar functionality has only recently been developed in Python.

As part of the Google Summer of Code[13] (GSOC), (Sheckhar et al., 2020) developed an API for the PySAL library which allowed for the creation of PySAL spatial weights objects from raster datasets stored in a xarray[14] DataArray object. This was a monumental development which allowed for the calculation of global and local spatial autocorrelation statistics in PySAL for raster datasets in addition to its existing vector support.

Less popular Python platforms that support spatial autocorrelation do exist, however they are not optimal solutions for incorporating into autoESDA. MuseoToolBox[15] only supports global

[11] https://georasters.readthedocs.io/en/latest/
[12] https://www.nv5geospatialsoftware.com/Products/ENVI
[13] https://gist.github.com/MgeeeeK/15426217eb5f368ca0ff12f66c2b5823
[14] https://docs.xarray.dev/en/stable/
[15] https://museotoolbox.readthedocs.io/en/latest/index.html

Moran's I, requiring the user to manually create a weights matrix, while GeoRasters is simply a wrapper for various PySAL functions.

2.6 Related Work

Section 2.2 highlighted the challenges brought about by the 3 Vs of geospatial big data, while Section 2.3 and 2.4 introduced EDA and ESDA as potential solutions to some of these challenges. This section is comprised of two parts – the first part outlining the need for new tools to address geospatial big data, and the second part discussing some of the tools available, their design philosophy, and identifying principles that could aid the design of a Python library that automates the ESDA workflow.

2.6.1 Call For New Developments

Throughout the years, Openshaw (1995), Anselin (2010), Dangermond and Goodchild (2020), and Singleton and Arribas-Bel (2021) have all highlighted the need for new tools that address challenges brought about by geospatial big data. The automation of ESDA could play a role in addressing some of these challenges as insights could quickly be obtained and used to guide further analysis (Anselin, 1996). Jern et al. (2008) argue for the use of statistical methods as a strategy to understand geospatial big data. Statistics are a large part of the ESDA toolbox and, as such, an appropriate use of ESDA could address the need to understand and create value from voluminous datasets.

There are two possible advantages to automating the ESDA workflow. The first is the increased efficiency as the time spent generating results could be minimised (Dangermond and Goodchild, 2020). This is beneficial when trying to keep up with the high velocity of big geospatial data. The second potential advantage is that the need for human input could be eliminated and by extension, any human-induced errors would be removed (Borlongan et al., 2016; Armstrong et al., 2019).

It is unclear whether removing human input from an ESDA workflow is an advantage. Anselin (1998) pointed out that ESDA is a data driven approach and thus the results need to guide further analysis. What is unclear is whether this feedback loop could also be automated, or if it can only proceed with human involvement. Datasets should also be processed without preconceived ideas, so that bias is not introduced into the results. Conversely, prior knowledge of the dataset could be advantageous as this may aid in the formulation of hypotheses. While Moncrieff et al. (2016) are of the opinion that data exploration is a user-driven approach, Goodchild et al. (1992) and Openshaw (1995) warn that the user can no longer be expected to be a trained statistician. This is increasingly likely due to the diverse nature of spatial data, meaning that end users are not always experts. For this reason, it would be beneficial to

automate the ESDA process as it eliminates the required input from potentially untrained users.

Openshaw (1995) highlighted the weaknesses of the traditional ESDA process and provided some guidelines for new tools. The first is that there should be extensive automation. The tool should have error checking functionality and its performance should constantly be optimised. The tool must be able to operate in real time, handle unexpected scenarios, and, although it could be considered as a black box, the user should have a means of understanding its design. Dangermond and Goodchild (2020) add to this by explaining that new tools should be able to repeatedly produce reliable results and that emphasis should be placed on efficiency. Each of these principles need not be present in the first few steps towards automating ESDA, as they have various levels of complexity. These principles do, however, paint a good picture of what an optimal tool could look like, and any progress in automating ESDA should follow in the direction of these guiding principles.

2.6.2 Similar Automation Projects

There are numerous automated tools that have been developed to deal with geospatial big data at various stages of the spatial data lifecycle. Batcheller (2008) automated the generation of metadata for spatial datasets, Borlongan et al. (2016) automated feature extraction from LiDAR datasets, Coetzee and Rautenbach (2017) designed a method to automate the creation of thematic maps, which was further adapted for large point datasets by Pillay et al. (2019). The motivation for each of these projects was that the output artifact made some contribution towards the automation of the task it aimed to address. This not only streamlined the data workflows, but it minimised the need for humans to carry out repetitive tasks. In doing so they broadened the abilities of available tools to deal with the large amounts of spatial data, while eliminating opportunities for human error to be incorporated into the workflow.

While there is no known tool that automates the ESDA workflow, there have been numerous advances in the automation of the EDA workflow. These projects can be discussed, and any learnings made can contribute towards to task of automating the ESDA workflow, which is merely an extension of EDA. Python libraries such as pandas-profiling (now officially known as ydata-profiling[16]), sweetviz[17], and autovis[18] allow a data analyst to easily carry out the EDA process by executing one line of code, which generates a HTML report that neatly summarises the results.

[16] https://ydata-profiling.ydata.ai/docs/master/index.html
[17] https://github.com/fbdesignpro/sweetviz
[18] https://github.com/AutoViML/AutoViz

DataPrep.EDA[19] is a Python library that has been designed to automate the EDA workflow (Peng et al., 2021). The philosophy behind its design is that it is: easy to use, has interactive speed, and is easy to customise. Ease of use is an important quality for a tool to have, and DataPrep.EDA has addressed this by only requiring one-line, easy to execute functions. The output results also have "*auto-insight*" functionality which highlights any significant trends that are identified in the dataset. Interactive speed means that the processing time is dependent on the outcome required. This means that time is not unnecessarily spent generating results in which the user has no interest. The design of DataPrep.EDA means that smaller EDA tasks can be run by selecting specific variables, or only calling for specific functions, such as correlation analysis or missing-value analysis. This means that a full report does not need to be generated each time the library is used. Finally, DataPrep.EDA is easy to customise. A full EDA report will require hundreds of parameters to customise the report so that it is "perfect" for its use-case. This makes easy customisation a major challenge – and the only way this has been addressed is by developing a detailed help guide, which the user would need to familiarise themselves with should they want a completely customised report.

Feature extraction from LiDAR datasets is another repetitive and time consuming spatial data workflow which has been successfully automated (Borlongan et al., 2016). The motivation behind its design was also to minimise the opportunities for human error. Just as with ESDA, it is not easy to completely eliminate human involvement from the workflow. Human input was instead handled by grouping similar tasks that did not require a decision to be made, thereby minimising the need for the user to be continuously involved. This reduced the likelihood of the user to introduce errors, regardless of their training (Borlongan et al., 2016). While the process of feature extraction from LiDAR has not been fully automated, the current work towards its automation has proven that task can be completed in less time, with results that are less prone to human-induced errors.

While there are no known attempts to automate ESDA, this section has outlined the need for software to do so, along with some guiding principles and form similar automation projects. This should pave the way for any new developments related to the automation of ESDA.

2.6.3 Similar Evaluation Methodologies

There is limited literature available that document the performance evaluation of platforms that support ESDA. Anselin et al. (2022) outline their approach to evaluating the performance of the rgeoda, pygeoda, GeoDa desktop, spdep and PySAL in order to benchmark the performance of each platform. The time to generate a queen's case matrix with an order of

[19] https://docs.dataprep.ai/user_guide/eda/introduction.html

one is recorded for each of the platforms. This is done with datasets of increasing magnitude of features (and by extension file size). A Local Moran's I permutation test is then calculated for each platform and dataset. The number of permutations are also varied to investigate their effect on the processing time. The results of this study form the basis of multiple assumptions that are highlighted the sections to follow. This methodology, while simple has also been adapted to evaluate the performance of autoESDA and is discussed in more detail in Chapter 4.

2.7 Discussion on Automating ESDA Functions

Section 2.3 and 2.4 described some functions that form part of EDA and ESDA, while Section 2.6 outlined some considerations in the automation of data-related tasks. This section aims to evaluate how easily, and the extent to which, EDA and ESDA functions can be automated. Table 4 summarises the findings that will be discussed in the remainder of this section.

One of the major contributors to how easily a function can be automated is what parameters are required to be specified by the user. The user may be required to specify multiple parameters that are specific to the selected platform. This section will only discuss the vital parameters that are relevant to the function, regardless of the software being used.

The calculation of descriptive statistics can easily be automated as the user is not required to specify any parameters. The sample size, mode, median, mean, minimum and maximum values, range, quartiles, variance, standard deviation, skewness, and kurtosis are all formulas that only require values for their calculation. The same is true for the automation of a box plot as the user is not required to specify any additional parameters aside from the dataset, meaning that the same process can be used without being altered to have a different output.

The generation of a histogram can also be easily automated. Apart from specifying whether a probability or frequency histogram is required, the only parameter that could potentially be specified is the number of bins. Matplotlib[20] uses ten as a default number of bins, while seaborne[21] automatically determines the number of bins to be used if this is not specified by the user.

The generation of a single scatter plot requires the specification of two variables in a dataset to plot against one another. This makes it a challenge to automate, however the solution is to generate a scatter plot matrix, with scatter plots between each combination of all the variables. Fortunately the generation of a scatter plot matrix is not very computationally intensive;

[20] https://matplotlib.org/stable/
[21] https://seaborn.pydata.org/

however it could create large plots that are difficult to view when used for datasets with a lot of variables.

The approach to automating correlation statistics is similar to that of scatter plots. Calculating a correlation value requires the specification of two variables, however the correlation coefficient may be calculated for every variable combination in a dataset, resulting in a correlation matrix. The other parameter a user may need to specify is the correlation type – this could be Pearson's correlation coefficient, Spearman's ρ, or Kendall's τ. Just as with scatter plots, the computation of correlation matrices is not intensive and thus a viable solution would be to automate this process by generating a correlation matrix for each correlation type – removing the need for the user to specify a correlation type.

The generation of a choropleth map requires the user to specify the variable to be mapped, as well as the classification scheme and number of classes. PySAL uses a default number of five classes, which means that the user would only need to specify the variable and classification scheme. The process of generating choropleth maps could be automated by plotting numerous choropleth maps for each variable, using a variety of popular classification schemes. Ordinary cartograms can easily be automated as they only require the variable on which they are based to be specified.

Creating a spatial weights matrix is arguably the most challenging part of an ESDA workflow to automate. This is due to the parameters that are required to generate the matrix, and their selection, which has the ability to skew the result of the function that uses the output weights matrix. The first decision required from the user is to specify the type of spatial weights matrix – this could be distance-based, contiguity-based, or a k-nearest-neighbour approach. If a distance-based matrix is chosen, the user would be required to specify the distance threshold as a parameter. Should a contiguity-based matrix be chosen, the user will have to specify the contiguity type and order. Finally, if a k-nearest-neighbour approach is selected, the user will have to specify k – the number of neighbours to be considered. GeoDa's tutorials recommend a queen's case contiguity matrix with an order of one as a default. Anselin et al. (2022) also use this combination in their experiment. Should the same approach be applied, one would be able to automate the generation of spatial weights matrix.

Global Moran's I and Geary's C are identical in the parameters that they require. These are: target variable, a spatial weights matrix, and the number of permutations. If a spatial weights matrix has already been constructed, its specification can be automated. The selection of a target variable can be avoided by calculating these global statistics for each variable in the input dataset. A default of 999 permutations is used in PySAL and GeoDa and if this approach is used, the need to specify the number of permutations is unnecessary. Finally, the LISA

calculations can also be automated with moderate difficulty and a similar approach to the global autocorrelation measures.

2.8 Conclusion

This chapter has outlined the use of geospatial big data and the challenges that arise from it. EDA and ESDA are potential solutions to some of these challenges, however ESDA is currently a time consuming and repetitive process. There are, however, some software tools that have ESDA functionality, and they could be extended to automate the ESDA workflow. This automation could lead to a saving in time spent on the ESDA workflow, while simultaneously reducing opportunities for human-induced errors. Section 2.7 discusses how easily some ESDA functions could be automated, and how one could go about doing so. The next chapter will summarise the earlier research that has taken place and outline the current status of autoESDA and what improvements can be made.

Table 4: Potential automation of various EDA and ESDA functions

Function	Input	Parameters	Output	Ease of Automation
EDA				
Descriptive Statistics	Dataset	None	Calculated statistics	Easy
Box Plot	Dataset	None	Box plot figure	Easy
Histogram	Dataset	Number of bins	Histogram figure	Easy
Scatter Plot Matrix	Dataset	None	Scatter plot matrix	Easy
Correlation Matrix	Dataset	Correlation type	Correlation matrix	Easy
ESDA				
Choropleth Map	Spatial dataset	Target variable Classification scheme Number of classes	Choropleth map	Moderate
Cartogram (Ordinary)	Spatial dataset	Target variable	Ordinary cartogram	Easy
Spatial Weights Matrix	Spatial dataset	Weights type (distance, knn, contiguity) Magnitude of weights (order, distance, n)	Spatial weights matrix	Difficult
Global Moran's I	Spatial dataset Spatial weights matrix	Target variable Number of permutations	Moran's I statistic Moran's I scatter plot	Moderate
Global Geary's C	Spatial dataset Spatial weights matrix	Target variable Number of permutations	Geary's C statistic	Moderate
Local Moran's I **Local Geary's C**	Spatial dataset	Target variable Spatial weights matrix	LISA cluster map LISA significance map	Moderate

CHAPTER 3: TOWARDS AN OPEN SOURCE LIBRARY FOR AUTOMATED EXPLORATORY SPATIAL DATA ANALYSIS (ESDA)

This chapter was presented in a poster session [22] at the International Society for Photogrammetry and Remote Sensing (ISPRS) congress 3 – 11 June 2022 in Nice, France, and published as an article[23]. It can be referenced using the following citation:

De Kock, N., Rautenbach, V., Fabris-Rotelli, I., 2022. Towards An Open Source Library For Automated Exploratory Spatial Data Analysis. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XLIII-B4-2022, 91–98. https://doi.org/10.5194/isprs-archives-XLIII-B4-2022-91-2022

This chapter was taken "as-is" from the publicly available version which means there may be some repetition from other chapters in this dissertation.

3.1 Abstract

The exploratory spatial data analysis (ESDA) process refers to the use of various functions to gain an initial understanding of a spatial dataset. These include measures of spatial heterogeneity and spatial autocorrelation. Currently, the ESDA process is repetitive and time-consuming. Additionally, while different results arise for different datasets, how these results are generated does not change significantly. Results are also generated individually for each variable which means that they cannot be easily compared or shared.

The automation of the ESDA process would therefore have multiple benefits as it would not only save time, but it would also allow the data analyst to keep up with the rapid rate at which we generate data. This paper aims to introduce the first iteration of autoESDA – a Python library capable of automating the ESDA process by summarising the results into a single report.

In this paper, we present the defined high-level requirements for the implementation of autoESDA. Various dependency libraries are discussed and a high-level overview of the workflow of autoESDA is described. The library is then evaluated against the requirements laid out earlier in the study. Semi-structured interviews were carried out, which yielded a wealth of feedback and suggestions from the participants, describing how the output report

[22] https://isprs2022.stream-up.tv/media-272-towards-an-open-source-python-library-for-automated-exploratory-spatial-data-analysis-esda?&fartype=cat&farval=138
[23] https://doi.org/10.5194/isprs-archives-XLIII-B4-2022-91-2022

could be improved. Finally, a roadmap of proposed further developments and improvements is discussed.

The first version demonstrates that the automation of ESDA is possible and lays the foundation for further development in this regard. This is an important contribution to understanding spatial data as it enables the data analyst to keep up with the magnitude of data that is generated on a daily basis.

3.2 Introduction

In recent years there has been a largescale increase in the volume of spatial data generated. This exponential increase in both the volume and velocity of spatial data is attributed partly to the decreasing price of sensors, along with a world where topics such as the *Internet of Things* (IOT) and *big data analysis* have experienced dramatically increased popularity (Armstrong et al., 2019). Spatial data is rapidly created through methods such as the geotagging of images on social media and traffic data from users of navigation software such as Google Maps. While there is great benefit in the availability of datasets, true value can only be obtained once this data is processed into useful information.

The data lifecycle refers to numerous stages that result in the transition of raw data into information. The lifespan of a data lifecycle varies according to the dataset (Raju and Nathan, 2018). Exploratory data analysis (EDA) is a process carried out near the beginning of a data lifecycle. Its purpose is to gain a basic understanding of the dataset. For spatial datasets, this process is known as exploratory spatial data analysis (ESDA) (Dall'erba, 2009).

ESDA is made up of various functions that aid the exploration of spatial datasets and identification of patterns that may otherwise go unnoticed (Murray and Estivill-Castro, 1998). Results arising from the ESDA process often dictate how the data is further utilised. ESDA functions can be carried out on both vector and raster based spatial data (Moura and Fonseca, 2020). The current iteration of autoESDA only supports data in vector polygon format, however work is currently underway to extend this functionality to support raster and other vector formats.

Two important components of the ESDA process are spatial autocorrelation, and spatial heterogeneity. Spatial heterogeneity is investigated using choropleth maps, box plots, scatter plots, and histograms, for example, which allow one to identify trends or patterns that could have otherwise gone unnoticed. The results of spatial autocorrelation are vital, as they dictate whether or not the recorded instances of a phenomenon are spatially dependant on each other (Dall'erba, 2009).

The ESDA process is both repetitive and time-consuming and the automation thereof would allow the data analyst more time to be able to focus on more important aspects of the data lifecycle. In practice, ESDA functions are run individually, meaning that results are also displayed individually, which does not allow for comparisons to be made.

There are numerous open source technologies with ESDA capabilities; the three most popular ones are Python libraries, R libraries, and GeoDa. Python has numerous advantages when working with spatial data, chief among which is its ability to handle large datasets. Various libraries also allow Python to easily integrate with geoportals, spatial database management systems, and other GIS technologies (Cura, 2019). Python libraries such as pandas[24], geopandas[25], PySAL, plotly and matplotlib are often used for executing ESDA functions and displaying these results. The seamless integration and wealth of available libraries make Python an ideal choice for automating the ESDA process.

Automation of similar processes using Python are not unheard of. The EDA process has been automated using Python libraries such as pandas-profiling (now officially known as ydata-profiling[26]), sweetviz[27], and autovis[28]. These libraries allow a data analyst to easily carry out the EDA process by executing one line of code, which generates a HTML report that neatly summarises the results.

Automation of various processes within the spatial data lifecycle are not uncommon. The curation of spatial metadata falls under the transformation stage of the lifecycle (Ciceli, 2015); it has been automated by Batcheller (2008). The generation of thematic maps falls under the distribution stage of the spatial data lifecycle and various efforts have been made to automate this process (Coetzee and Rautenbach, 2017; Pillay et al., 2019). While these examples are not entirely related to ESDA, they do, however, illustrate that there is a benefit to automating repetitive processes within the spatial data lifecycle.

The aim of this paper is to present our first iteration of autoESDA, a library that automates the ESDA process in Python. The paper discusses the design and implementation of the library by describing the high-level requirements, dependency libraries, and workflow of the library itself. The library is then evaluated according to the defined requirements, and numerous

[24] https://pandas.pydata.org/
[25] https://geopandas.org/en/stable/index.html
[26] https://ydata-profiling.ydata.ai/docs/master/index.html
[27] https://github.com/fbdesignpro/sweetviz
[28] https://github.com/AutoViML/AutoViz

interviews are conducted in order to gain feedback on the first iteration. Finally, a roadmap for further development is discussed.

3.3 Requirements and Implementation

3.3.1 Requirements

Multiple high-level functional and non-functional requirements were defined during the planning phase of this project. These requirements were decided on by identifying solutions to the major issues encountered by carrying out an ESDA process and what a potential solution would look like. Once the library was developed, the specified requirements were revisited to ensure that the high-priority requirements were satisfied. As this was an iterative process, if the functional requirements were not satisfied, the development phase was revisited to ensure that all the high-priority requirements were met.

Functional requirements refer to functions that a system is required to perform (Young, 2003). Conversely, non-functional requirements refer to properties of a system that do not dictate what needs to be done, but rather how well it should be done. Table 5 summarises the high-level functional and non-functional requirements that were defined for the development of the autoESDA library.

Table 5: High-level functional requirements

Functional Requirement	Description
Report Output (High Priority)	The library should generate a HTML report that can be saved to the local computer.
Spatial Heterogeneity (High Priority)	The generated report should include a box plot, histogram, descriptive statistics, and correlation statistics.
Spatial Autocorrelation (High Priority)	The generated report should include a Moran's I simulation, the associated statistics, and a LISA cluster map.
Data Type Detection (High Priority)	The library should be able to distinguish between columns that can be plotted and have statistics calculated on them. It is assumed that data is already converted into the correct types and unsupported data types (such as strings and characters) should be ignored.
Non-Functional Requirement	**Description**

Simple Execution (High Priority)	The library should be simple to use, meaning that only one parameter (the GeoDataFrame) is required to generate the report.
Offline Availability (Low Priority)	The generated report should not reference any external sources, this would mean that the report does not require an internet connection to display correctly.
Colour Use (Medium Priority)	A suitable colour scheme/theme for the report should be selected that is both appealing and free from any alternate connotations.
About Page (Medium Priority)	The report should include an about page which tells the user what defaults have been set for the generated figures and statistics.
Data Sample (High Priority)	A subset of the dataset should be displayed in the report.
Performance (Low Priority)	The library should generate a report timeously once the function has been executed.

3.3.2 Design

The first aspect of the design stages focused on deciding which ESDA functions to include in the library. These decisions were made according to how popular a certain ESDA function was, as well as how easily it could be automated. Functions that require the data analyst to specify numerous parameters that could not easily be set to a default setting, are assumed to be more difficult to automate.

Most ESDA functions are simple enough to automate; however, parallel coordinate plots (PCPs), measures of autocorrelation, and choropleth maps are all seen to be complex to automate. This is due to the fact that they have numerous parameters that need to be specified. The combination of variables to include in the PCP will depend greatly on the dataset used and a generic solution for automation cannot easily be implemented (Zhou et al., 2018). The same argument can be made for the automation of spatial autocorrelation and choropleth maps, as the input parameters could have a huge effect on the outputted results.

The functionality to allow the user to specify their own spatial weights matrix has not been included in the current version of autoESDA. Spatial autocorrelation, however, is seen as a high priority requirement, meaning that there needs to be some degree of spatial autocorrelation in the output report, and the utility thereof can be evaluated in the interviews. For this version of autoESDA, it was decided that a Moran's I simulation with a queen's case

first-order matrix along with a LISA map would be included in the report as measures of global and local spatial autocorrelation. Moran's I was the chosen function as it is the most commonly used measure of spatial autocorrelation (Jackson et al., 2010). A first order queen's case matrix was used as the default spatial weights matrix.

Due to their importance in visualising the spatial distribution of different variables, the decision was made to include choropleth maps. To overcome the need for the user to specify which classification scheme to use for these maps, the decision was made to include four choropleth maps, each with a different classification scheme, for each variable. It was also decided to use the geopandas default number of intervals, which is five.

With the exception of those mentioned above, the majority of ESDA functions do not require input parameters and could therefore be automated with relative ease. These include generic five number summaries (minimum, mean, median, maximum, and standard deviation), box plots, histograms, scatter plots, and correlation matrices.

There are numerous Python libraries that exist with the intention of solving various problems or addressing different needs within the Python development community. In the development of this library, existing functions from other libraries were used. Table 6 describes the libraries referenced in autoESDA, which are known as dependencies. Each of these libraries have been included as they serve a specific purpose in the autoESDA library. These libraries have been chosen according to the functions which they provide, and their relative popularity. Choosing libraries according to popularity has two major advantages, namely: readily available support, and a community of contributors who help to ensure updates and bug fixes are routinely rolled out.

Table 6: Dependencies of the autoESDA library

Dependency (Version)	Description
geopandas **(0.8.1)**	The geopandas library is an extension of the popular pandas library which defines DataFrames as a way to structure data. geopandas adapts this as a way to store spatial data such that this GeoDataFrame is the attribute table, where there are additional columns for geometry or coordinates.
libpysal **(4.4.0)**	This is the core library that PySAL is based on. It is used in this project to create the spatial weights matrix which is used in the autocorrelation calculations.
PySAL **(2.3.0)**	This library, along with its dependencies, allow for the plotting of the choropleth maps as well as the Moran's scatter plot and LISA cluster map.
matplotlib **(3.4.2)**	matplotlib is a popular library for creating graphs and other visual aids. This library enables the use of grids and annotations to combine the numerous figures together.

seaborn **(0.11.2)**	seaborn is similar to matplotlib, however it has extra functions such as the heatmap and pairplot function which was used in this project.
io **(3.8.10)**	This library converts images into objects made up of bytes. In conjunction with the base64 library, it allows for images to be embedded/stored directly in the HTML file.
base64 **(3.8.10)**	The base64 library was used to encode images as objects and works in conjunction with the io library to enable storage and embedding of images within the HTML file. This avoids the need for external files.

3.3.3 Implementation

Once the decisions were made regarding which ESDA functions and dependencies to include, it was time to design how the functions would work together to generate an appropriate report. This workflow is visualised in Figure 9.

To begin with, the library will accept a GeoDataFrame, from which it will determine which columns have a numeric data type and which do not. The ESDA functions for autoESDA are calculated from numeric data, which is why this differentiation needs to take place.

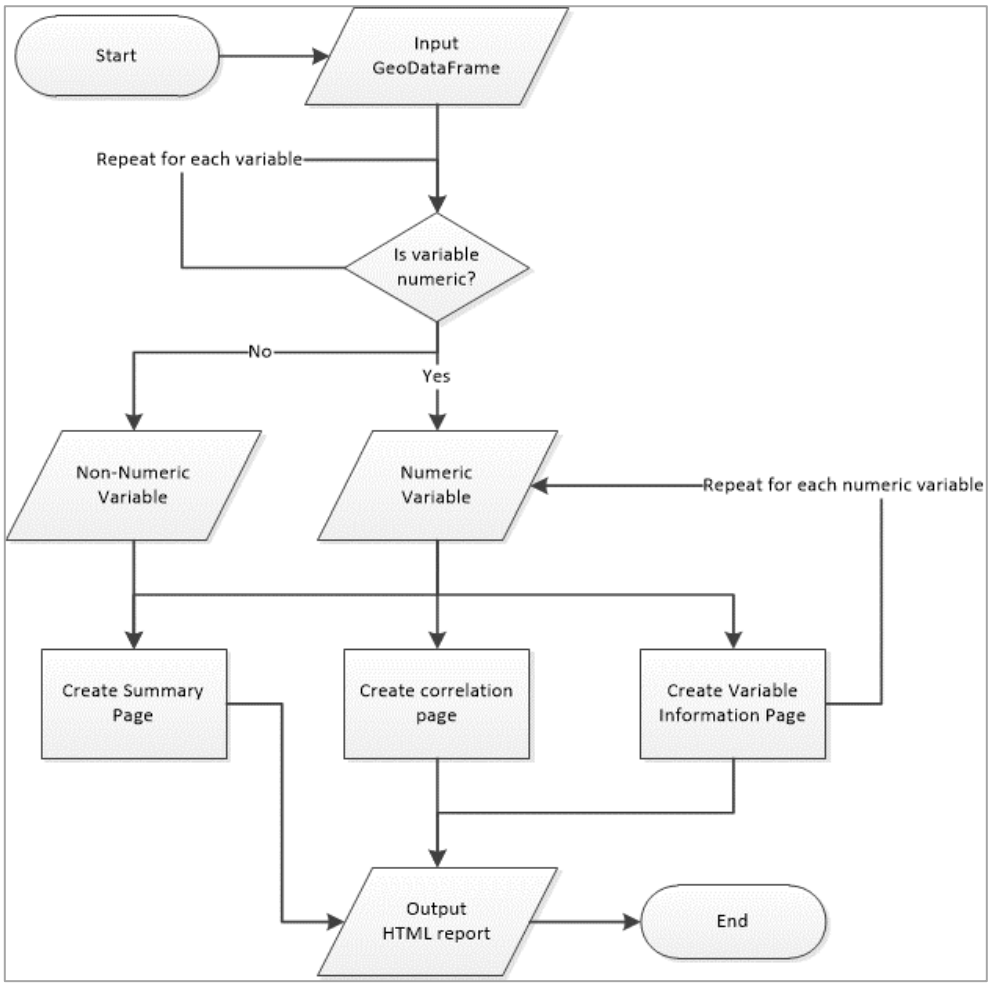


Figure 9: Workflow of the autoESDA library

Both non-numeric and numeric variables are required for the Summary Page, an example of which is shown in Figure 10. This is because the Summary Page displays a sample of the dataset, as well as a description of which datasets were included in the report (numeric variables) and which variables were not included (non-numeric variables). In addition to the dataset sample, the Summary Page also includes a basic outline of the study area, descriptive statistics, and a data overview which includes basic metadata, such as the projection used.

The next block of code entails a loop through the variables in order to create a Variable Information Page for each numeric column in the GeoDataFrame. Each iteration of the loop will create a box plot, histogram, various choropleth maps and a Moran's I and LISA simulation. An example of a Variable Information Page is shown in Figure 11.

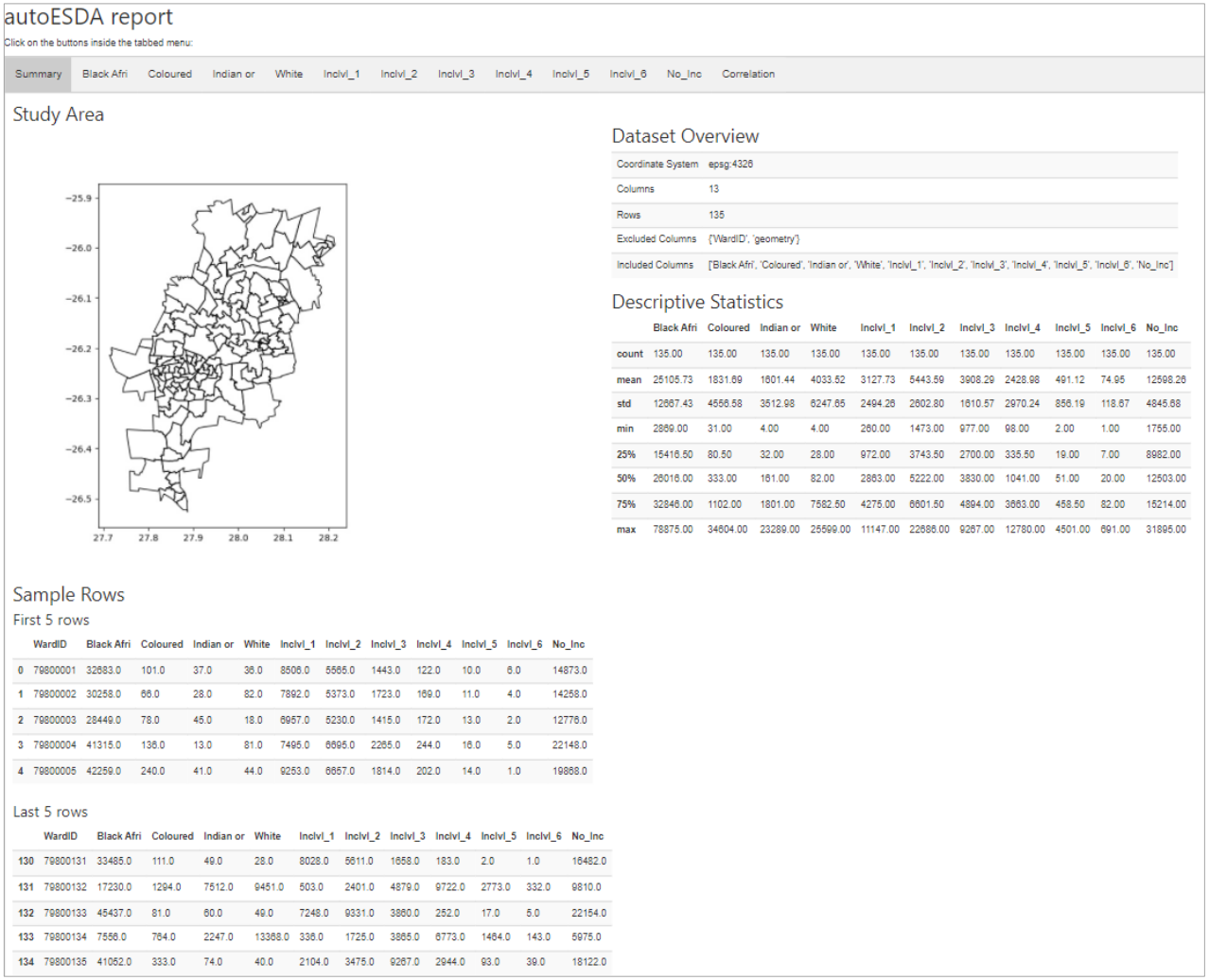
Finally, the Correlation Page (shown in Figure 12), composed of a heatmap and pairwise plot, was created using the numeric variables. The Correlation Page, along with the Summary Page

and all the Variable Information Pages is combined into a HTML report, which is then saved to the working file directory.

3.3.4 Availability and Usage

The source code for autoESDA is available in a GitHub repository[29] under the BSD 3-Clause license. An example report generated by autoESDA can also be viewed online[30].

[29] https://github.com/NicholasDeKock/autoESDA
[30] https://autoesda.github.io/autoESDA-static/



autoESDA report

Click on the buttons inside the tabbed menu:

Summary | Black Afri | Coloured | Indian or | White | Incivl_1 | Incivl_2 | Incivl_3 | Incivl_4 | Incivl_5 | Incivl_6 | No_Inc | Correlation

Study Area

Dataset Overview

Coordinate System	epsg:4326
Columns	13
Rows	135
Excluded Columns	['WardID', 'geometry']
Included Columns	['Black Afri', 'Coloured', 'Indian or', 'White', 'Incivl_1', 'Incivl_2', 'Incivl_3', 'Incivl_4', 'Incivl_5', 'Incivl_6', 'No_Inc']

Descriptive Statistics

	Black Afri	Coloured	Indian or	White	Incivl_1	Incivl_2	Incivl_3	Incivl_4	Incivl_5	Incivl_6	No_Inc
count	135.00	135.00	135.00	135.00	135.00	135.00	135.00	135.00	135.00	135.00	135.00
mean	25105.73	1831.09	1601.44	4033.52	3127.73	5443.59	3908.29	2428.98	491.12	74.95	12598.26
std	12667.43	4556.58	3512.98	6247.85	2494.26	2602.80	1610.57	2970.24	856.19	118.67	4845.88
min	2869.00	31.00	4.00	4.00	260.00	1473.00	977.00	98.00	2.00	1.00	1755.00
25%	15416.50	80.50	32.00	28.00	972.00	3743.50	2700.00	335.50	19.00	7.00	8982.00
50%	26016.00	333.00	161.00	82.00	2863.00	5222.00	3830.00	1041.00	51.00	20.00	12503.00
75%	32846.00	1102.00	1801.00	7582.50	4275.00	6601.50	4894.00	3663.00	458.50	82.00	15214.00
max	78875.00	34604.00	23289.00	25599.00	11147.00	22688.00	9267.00	12780.00	4501.00	691.00	31895.00

Sample Rows

First 5 rows

	WardID	Black Afri	Coloured	Indian or	White	Incivl_1	Incivl_2	Incivl_3	Incivl_4	Incivl_5	Incivl_6	No_Inc
0	79800001	32683.0	101.0	37.0	36.0	8506.0	5565.0	1443.0	122.0	10.0	6.0	14873.0
1	79800002	30258.0	66.0	28.0	82.0	7892.0	5373.0	1723.0	169.0	11.0	4.0	14258.0
2	79800003	28449.0	78.0	45.0	18.0	6957.0	5230.0	1415.0	172.0	13.0	2.0	12776.0
3	79800004	41315.0	136.0	13.0	81.0	7495.0	6695.0	2265.0	244.0	16.0	5.0	22148.0
4	79800005	42259.0	240.0	41.0	44.0	9253.0	6657.0	1814.0	202.0	14.0	1.0	19868.0

Last 5 rows

	WardID	Black Afri	Coloured	Indian or	White	Incivl_1	Incivl_2	Incivl_3	Incivl_4	Incivl_5	Incivl_6	No_Inc
130	79800131	33485.0	111.0	49.0	28.0	8028.0	5611.0	1658.0	183.0	2.0	1.0	16482.0
131	79800132	17230.0	1294.0	7512.0	9451.0	503.0	2401.0	4879.0	9722.0	2773.0	332.0	9810.0
132	79800133	45437.0	81.0	60.0	49.0	7248.0	9331.0	3860.0	252.0	17.0	5.0	22154.0
133	79800134	7566.0	764.0	2247.0	13368.0	336.0	1725.0	3865.0	6773.0	1464.0	143.0	5975.0
134	79800135	41052.0	333.0	74.0	40.0	2104.0	3475.0	9267.0	2944.0	93.0	39.0	18122.0

Figure 10: Summary Page

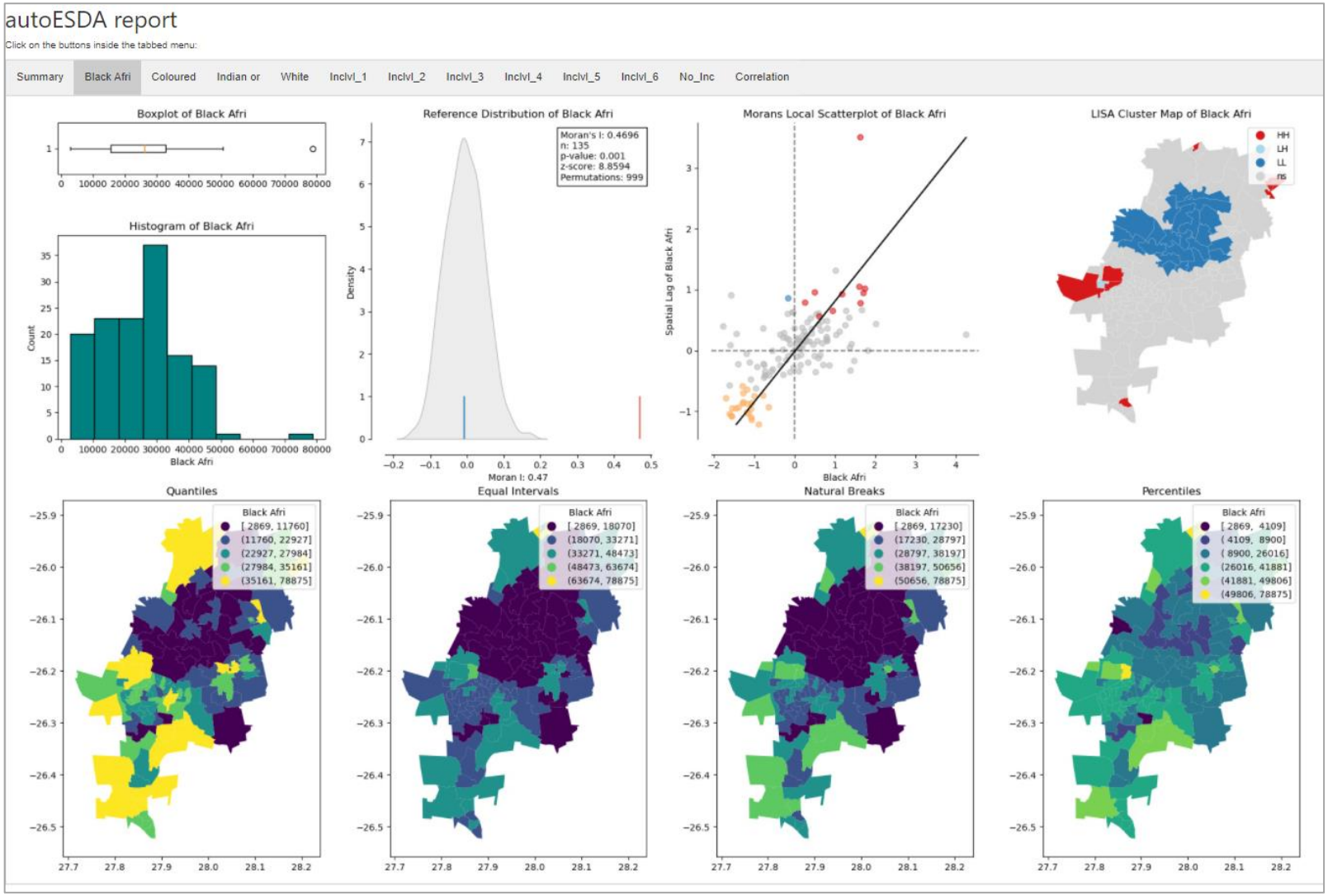


Figure 11: Variable Information Page

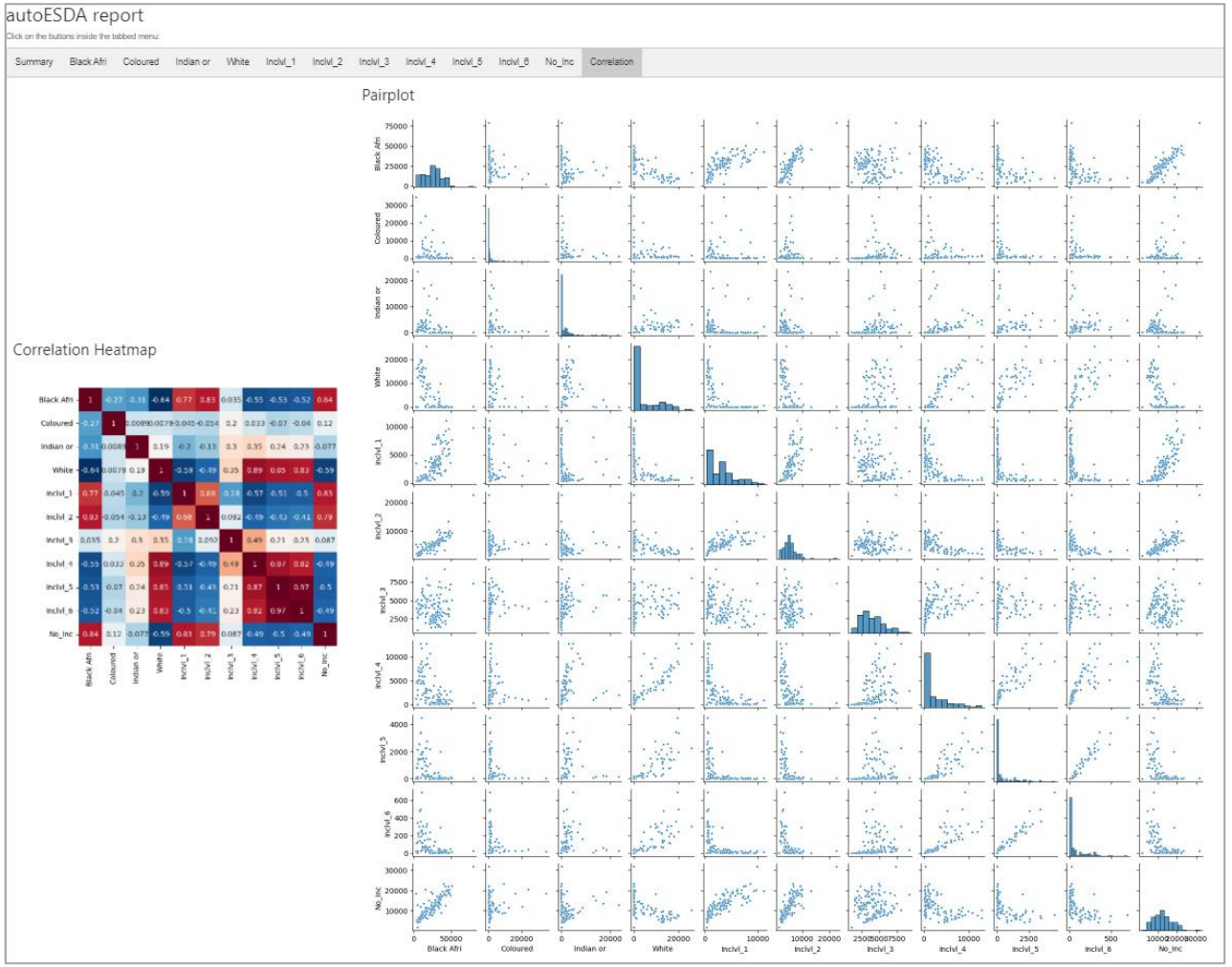


Figure 12: Correlation Page

3.4 Evaluation

3.4.1 Evaluation Against Requirements

In Table 5, four functional requirements and six non-functional requirements were defined. This was used as a guide for the researcher to gauge how much development still needed to take place for the first iteration of the library. This section discusses how each of the requirements were met, whereas the interviews and discussion investigate how well these requirements were met.

3.4.1.1 Functional Requirements

The library generates a HTML report which is saved to the working directory, thus satisfying the first functional requirement.

Measures of spatial heterogeneity include a descriptive statistics table which has a count, mean, standard deviation, minimum, 25th percentile, median, 75th percentile, as well as a maximum value for each variable. Furthermore, there are box plots, histograms, and numerous choropleth maps for each variable, as well as a correlation matrix.

Spatial autocorrelation has been addressed through the inclusion of a reference distribution (to evaluate the statistical significance of the calculated values), Moran's I scatter plot, and LISA cluster map. The reference distribution plot also displays the Moran's I value, sample size, p-value, z-score, and number of permutations.

The final functional requirement refers to the library's ability to discern numerical variables from the rest. This is because mathematical plots and statistics can only be generated from data that is numeric in nature. This is done in the first few lines of code of the library.

3.4.1.2 Non-functional Requirements

The first non-functional requirement which has a high priority is that the library runs using only one line of code, making it simple to use. While the library can be called using one line of code, it is currently not a published library which means that this requirement is not entirely satisfied. The library currently accepts no parameters except for the GeoDataFrame itself.

Offline functionality was a low priority non-functional requirement, which was not satisfied through the development of this library. The requirement aims to ensure that the report can be viewed without an internet connection, however this is not the case as it references two external style sheets. While the report will still display when offline, the experience for the user may be different.

The use of appropriate colour schemes that are neither conflicting, nor misleading, was listed as a medium priority non-functional requirement. This is a challenging requirement to meet as it is very subjective in nature. The researcher made all possible efforts to choose suitable colour schemes and the chosen colours were put through a colourblind simulator[31]. The colours used deemed to not be misleading or conflicting and are appropriate for people with colour vision impairment. For this reason, it is argued that the requirement for suitable colours has been met, and the extent to which this requirement has been satisfied will be determined through the interview process.

Other non-functional requirements include the presence of an About Page, describing decisions and default values made in order to generate the report, as well as a sample of the original dataset. Due to time constraints, the current iteration of autoESDA does not have an About Page, however there are plans to include this in future iterations. The library does, however, show the first and last five rows as a sample of the dataset.

Performance is the final non-functional requirement. It was listed as low priority and was not tested. In order to test performance, a benchmark needed to be identified – this benchmark has not yet been decided on. As such, it cannot be said if or how well this requirement was satisfied. There are, however, plans to test the performance of autoESDA in the future.

3.4.2 Interview Process

Numerous interview participants with varying experience, careers, and frequency of using ESDA were used in the interview process in order to generate a variety of feedback. These interviews took place as part of an earlier project in 2021. Table 7 summarises the demographic information of these participants. The interviews that took place were semi-structured in format. Table 8 shows the predefined questions which were used as a guide for the interview process. This was seen as the most effective strategy to adopt as it allowed the researcher to gain a further understanding regarding some statements that were made by the participants.

[31] https://www.color-blindness.com/

Table 7: Demographics of the interview participants

Participant	Years of experience	Sex	Industry	Job title	How often do you use ESDA functions?
1	2	M	Software Engineering	Software Engineer	Never
2	20	M	GIS education	Associate Professor	Monthly
3	4	F	Commercial GIS	Geospatial Consultant	Monthly
4	1	M	Commercial GIS	Data Scientist	Monthly
5	6	M	Commercial GIS	Geospatial Developer	Every two months
6	1	M	GIS	Geoinformation Specialist	Never
7	2	F	GIS	Student Assistant	Monthly
8	24	M	GIS/Cartography	Senior Cartographer	Weekly
9	25	M	Education	Freelance Data Analyst	Monthly
10	8	M	GIS & research	GIS Analyst, Lecturer	Weekly
11	4	M	IT/ Data science	Data Scientist	Monthly
12	17	F	Research	Associate Professor	Never
13	1	F	Research	Lecturer	Never

The interviews were carried out on the Zoom video conferencing platform as this allowed for the researcher to share their screen and eliminated the need for any travel or physical meetings between the researcher and the participants. This also allowed for a wider variety of participants as travel was not necessary.

The participants were sent an example report beforehand, so that they had time to look at it and consider some feedback before they were interviewed.

Table 8: Interview questions

Interview questions
1. What position do you hold, and how does it require you to make use of ESDA functions? 2. What challenges do you currently have when conducting an ESDA process? 3. Could you tell us about the process you follow when you are performing ESDA process? *[Show prototype]* 4. General **a.** How comfortable are you using Python?

   b. Can you think of any improvements that could be made to the structure/layout of the report?
   c. Are the titles of each section clear or are they misleading i.e. do you get the information you expect when selecting them?
   d. For each page, could you say whether this section is useful to you? What improvements would you recommend?
   e. Now that you have seen what the library can do, would you use it where necessary?
   f. What hesitations do you have about the use of this library?
5. Summary page
   a. Are there any other features/statistics you would like or expect to be on the summary page?
   b. Are the statistics on the summary page useful to you?
6. Choropleth maps
   a. Are there extra classification schemes for choropleth maps that you would like to be included in this library?
   b. Would you prefer there to be more/less classes for the choropleth maps?
   c. Do you feel that the colour scheme is suitable? If not, do you have a recommendation as to what it should be?
   d. Would you recommend any other improvements to be made to the choropleth maps section?
7. Autocorrelation
   a. Are there any extra statistics you would expect to find in a report like this?
   b. Do you feel that it is necessary to include the probability distribution and scatter plot?
   c. A queen's case contiguity matrix with an order of one has been set as the default, do you feel that this is a good idea? Is there another strategy which you would prefer?
   d. Are there other important autocorrelation measures that you would prefer to Moran's I?
8. Correlation
   a. How easy is it for you to interpret the correlation matrix/heatmap?
   b. Do you think it is necessary to include the scatter plots for each relationship?
   c. Do you think the colour scheme is suitable? If not, do you have a recommendation as to what colour scheme should be used?
9. Pairwise plot
   a. Do you like the layout of the pairwise plot or do you find it confusing to understand?
   b. What colour scheme do you feel should be used for the pairwise plot?
   c. How many bars do you think a histogram should have?

3.4.3 Interview Feedback

There were thirteen participants who gave feedback, each with different academic backgrounds, work experience and experience levels. This variety lead to a huge amount of varied and sometimes contradictory feedback.

The feedback is divided into four sections, namely: the Summary Page, the Variable Information Page, the Correlation Page, and the About Page. All of the participants were impressed with the library prototype and agree that the progress has been in the right direction.

Participant 2 said that the report was "*great, and very useful*" which was supported by Participant 3 who said that the tool filled a "*definite need in the GIS industry*". Participant 5 stated that the report was "*useful and well implemented*" which backs up the opinion of Participant 10 in that "*everything I looked for was here*".

3.4.3.1 Summary Page

The first major element on the Summary Page (as shown in Figure 10) is the map of the study area. In general, the participants were glad that it was present and provided the user with some insight about the shape of the area that is described in the report. Participants 2 – 6 all indicated that they see value in this map being interactive, with popups providing them with the relevant information for each of the polygons when hovered over. Participant 7 also recommended the use of colour in the study area map to make it more appealing. While this would improve the library's appearance, the issue would be selecting an appropriate colour scheme that does not have any potential connotations depending on the datasets used in the report. It was also suggested that there should be a name of the study area above the map. This may be challenging due to the versatility of the library being able to generate generic reports, however it was suggested that the call function of the library should have a parameter where the user could specify a name.

Participant 2 who comes from a GIS education background, mentioned that students may be confused by the use of the terms rows and columns as it is too similar to raster data, and that the terms attributes and fields should be used instead.

There was not much feedback given from the participants relating to the dataset overview table with the exception of participants 5 and 11 who mentioned that they would like to see some spatial statistics included in it. Examples they gave included average area of the polygons and average number of neighbours.

The other major element on the Summary Page was the descriptive statistics table. In general, the participants were satisfied that most statistics that they would look for were present, with the exception of the skewness, kurtosis, as well as the number of null or unique values in each column. The majority of participants made this comment. Additionally, Participant 2 also suggested that the descriptive statistics table include a Moran's I value.

The final element on the Summary Page is the dataset sample which consisted of the first and last ten rows to give the user an idea of what the original dataset looked like. There were contrasting views amongst the participants regarding what constitutes a suitable sample. Participants 1, 9, and 11 were of the opinion that showing 20 rows was excessive and that

only the first and last five rows were necessary. Participant 2, however felt that all the rows in the dataset should be included and that the user should be allowed to query these in order to aid their understanding of the dataset. Strategies such as only showing the first ten rows, or a random ten rows were also suggested by Participants 4 and 5.

3.4.3.2 Variable Information Page

The first two elements on the Variable Information Page (as shown in Figure 11) were the box plot and the histogram. No major comments were received from any of the participants; however, each of them emphasised the importance of having these present. Once asked about the number of bins recommended for the histogram, the participants seemed to be happy with the default value of ten bins and did not see the need for this to change.

The reference distribution drew quite a lot of feedback from the participants. While it is a good inclusion in the report, the lack of a key for the red and blue lines on the diagram, coupled with the non-descriptive title, gave some of the participants the impression that it could be improved. The x-axis and y-axis could be more descriptive such as indicating what they represent. The values in the textbox could also be coloured red or blue to link them to the line on the reference distribution that they relate to. One of the participants also suggested that a "*clustered/not clustered*" label should be included on the reference distribution. Some of these changes would be challenging to implement as it would involve the modification of code in the existing PySAL library.

One the major issues identified with the Moran's I scatter plot was that the visual gradient of the line of best fit does not match the Moran's I value (this should not be the case). This was brought about by the stretching of the scatter plot to match the size of the other subplots; however, it is misleading. One of the participants also commented on the colours used in the scatter plot, citing the fact that the user is not told what these colours represent, and therefore unsure whether they relate to the LISA scatter plot or not.

The participants also indicated that they valued the inclusion of the LISA cluster map as part of the report. There was, however, a comment on the colour scheme chosen with one of the participants having the opinion that a single, graduated colour scheme would be more suitable than the Red-Blue colour scheme currently being used. Some of the participants also found the labels in the legend to be difficult to understand, and that inexperienced users may not understand that HH refers to features that have High-High autocorrelation or ns which represents polygons that do not have significant spatial autocorrelation with its neighbours.

All of the participants were of the opinion that Moran's I was an appropriate measure to be used as a measure of autocorrelation, rather than another measure such as Geary's C. While the participants did not express strong opinions regarding what spatial weights format was the most appropriate, they indicated that the default of a queen's case contiguity with an order of one was acceptable, provided that this was indicated somewhere. Participants 9, 10 and 11 all indicated that they would like the functionality that would allow them to specify their own spatial weights matrix as a parameter of the call function for the report.

The choropleth maps generated a lot of discussion, with the majority of the feedback being directed towards the legend placement, that covered a large portion of the map. Although the matplotlib parameter of "*best position*" is used, it is evident that the placement is not always optimal. Some suggestions to overcome this from the participants included placing the legends outside the map, removing the decimals (which are unnecessary) from the legend and making it a horizontal rather than a vertical legend. It was also mentioned that the variable name should be included in the title of the map and not in the legend. Regarding the classification schemes chosen, the participants were in general, happy with those that were present, however some participants did suggest a box map and standard deviation classification scheme to also be included. When questioned about the number of classes for maps (currently the default of 5 is used), none of the participants considered this to be a problem. The colour scheme was also mentioned in the interviews, with the majority of the participants happy with the current one being used. One comment regarding the colour scheme which arose from two of the participants was that it should be inverted, so that values with a greater magnitude are assigned the darker, more intense colours.

Participant 10 questioned if the report was suitable for those who are colourblind. This was not a consideration in the lifecycle of the project, and it was decided to test the report using an online colourblind simulator. Red and green colour-blindness are the most common types, which is what was simulated. The results show that there is an effect of these types of colour-blindness, however all features still vary enough to the colourblind eye to be differentiated from each other.

Concluding remarks relating to the Variable Information Page were that it feels very congested, and that this could be avoided by increasing the spacing between plots and removing the borders from the choropleth maps.

3.4.3.3 Correlation Page

The Correlation Page (as show in Figure 12) was made up of a correlation heatmap and a pairplot. The majority of the participants found benefit in there being both a correlation

heatmap as well as a pairwise plot. Participants 3, 6, 8, and 9 who are all very experienced in the GIS industry, suggested that the values in the correlation heatmap should be rounded up to two decimal values. The colour scheme of the heatmap was also discussed, with some participants of the opinion that it is too closely related to the colours used in the autocorrelation subplots, and therefore misleading. Participants 12 and 13 stated that a colour ramp outside of the correlation matrix would improve their understanding. Importantly, Participants 10-13 also questioned which type of correlation was used as it was not stated anywhere, and that a user should be able to choose which correlation measure they would like to be present in the report.

While captioned as a pairplot, it was brought to the attention of the researcher that the diagram should more appropriately be called a pairwise plot. The pairwise plot could be made more user friendly through the use of more labels, and red borders for the subplots with significant relationships (correlation values above |0.7|). Statistics such as coefficient of determination, trendlines and adjusted R^2 values would also be of value to the user. One of the participants also stated that a correlation value is not suitable if the data is not linear, and for this reason it may be beneficial to include a warning for relationships that are non-linear yet are found to have a significant correlation.

Finally, the placement and layout of the Correlation Page drew a reasonable amount of discussion. Some of the participants preferred both the pairwise plot and the heatmap to be square in shape and rather placed under each other for more space. Other participants, however, were of the opinion that only the upper or lower triangle were necessary and that instead these two elements should be combined so as to maximise the use of space, while minimising the duplication of information on the page. This would be a valuable improvement, however due to the pairwise plot being a function from the seaborne library, these suggestions would be difficult to implement.

3.4.3.4 About Page

While this page was not in the prototype shown to the participants, there was a lot of discussion around the necessity of an About Page and, therefore, it has been given its own subheading. The About Page should act as a manual for the generated report that users could navigate to so that they may improve their understanding of the report. Some elements that were suggested to be included here were the number of histogram bins, significance values and parameters used for the construction of the spatial weights matrix used for the Moran's I simulation, description of each of the subplots, number of default classes for the choropleth maps and correlation type used in the correlation heatmap. Additionally, Participant 9

suggested that the date and time that the report was generated be included, as well as a disclaimer relating to how the report should be used.

3.5 Roadmap of Further Developments

The development of autoESDA is an ongoing process, meaning that the current version has laid the foundation for more features to be included in the future.

A major improvement to autoESDA will be the ability to accept multiple data formats. Currently, the library only works with vector polygon geometries, however there is scope for this to be improved to support vector line and point data, as well as data in raster format.

Results from the interviews are discussed in Section 3 and highlight multiple opportunities for further developments. An example of this is the ability of the user to specify their own spatial weights matrix. This means that instead of using the current default of a queen's case first-order matrix, the user could specify as a parameter the shape and order of their preferred spatial weights matrix.

Interview Participants also mentioned that including additional ESDA functions such as Geary's C, would add to the wealth of information in the generated report.

Participants had numerous suggestions that would improve the layout of the report. These suggestions included the repositioning of some of the elements, as well as increasing the spacing between figures so that the report does not feel so congested.

A popular suggestion amongst the interview participants was the inclusion of an About Page in the report. This would provide the user with information relating to the autoESDA library, as well as the report metadata such as the date generated, the default values for choropleth maps, or the type of spatial weights matrix used in the spatial autocorrelation calculations.

Testing the scalability and performance of the library is another aspect of autoESDA that could be addressed through future work. This includes investigating how efficient the script is in processing datasets, as well as if it has the capability to handle large volumes of data with the same efficiency.

One important milestone planned for the autoESDA library is the refactoring of the code so that it may be used in a QGIS plugin. This will eliminate the need for a user to have a knowledge of Python and will allow the user to generate an autoESDA report through a graphical user interface on the popular GIS platform.

3.6 Conclusion

The aim of this research was to present the first iteration of autoESDA. This was achieved by describing the process of defining requirements and designing the library's workflow. autoESDA was then evaluated against the predefined requirements, as well as through the use of interviews to solicit feedback. While the first iteration of autoESDA is functional, there are planned improvements and additional functionality. Aspects of the library such as scalability and performance could also be investigated to ensure that the library is capable of handling the large datasets that are common in today's data-driven world. This article presented the first iteration of autoESDA and in doing so, has laid the foundation for more work to be carried out in the automation of the ESDA workflow. The next chapter will define a new set of requirements on which the development of the next iteration of autoESDA will be based.

CHAPTER 4: SECOND ITERATION OF AUTOESDA: REDESIGN AND EXPANDING ITS CAPABILITIES

4.1 Chapter Overview

Chapter 4 describes the updates and improvements that have been incorporated into the second iteration of autoESDA. This includes the architectural and cosmetic design decisions and their motivation thereof. The majority of these improvements have been recommended in the interviews that are described in Chapter 3, however some improvements have also been brought about by an increased understanding of software design. The feedback from the interviews have been converted into user stories which can be viewed in Appendix B. This chapter has been divided into three sections, each one addressing a different high-level requirement (Chapter 5 discusses the performance requirement). These requirements are summarised in Table 9 below.

Table 9: High-level requirements for the second iteration of autoESDA

Requirement	Description
Raster Functionality	The updated library should be capable of accepting raster datasets and processing them to generate an autoESDA report. The raster report should generate efficiently and can largely be based off the vector report.
Updated Architecture	The architectural design of autoESDA should be updated so that it is more modular. This will allow for changes to be implemented more efficiently.
Other Minor Improvements	This includes a variety of cosmetic and other minor improvements resulting from the interviews described in Chapter 3.
Performance	The library should be capable of generating a report *timeously.*

Section 4.2 describes the new raster functionality and the comparison of different strategies in order to identify the optimal strategy for ESDA calculations with raster datasets. Section 4.3 describes the new architectural design of the library to allow for the raster functionality to be incorporated into autoESDA. This modular architecture streamlined the process of implementing a variety of other minor improvements – these are described in Section 4.4. Finally, Section 4.5 will discuss some limitations of the second iteration of autoESDA and outline potential future improvements.

4.2 Raster Functionality

One novel suggestion that arose from the earlier interviews was that the autoESDA library should have the functionality to process raster datasets. The first iteration of autoESDA was only compatible with a GeoDataFrame of vector polygons. Raster grids and vector polygons both conform to the definition of a lattice dataset in that they cover an area and are divided into smaller, discrete areas known as cells or units (Cressie, 1993). These units do not overlap; however they share common boundaries. A raster grid is an example of a regular lattice as each cell has the same shape, size, and orientation, whereas a lattice of vector polygons is referred to as an irregular lattice.

Lattice datasets are commonly used to detect spatial patterns and find a suitable explanation for their occurrence (Saveliev et al., 2007). As forms of lattice datasets, albeit with different data models, one could reasonably expect software platforms capable of ESDA to support both raster grids and vector polygons. Regular EDA functions (descriptive statistics, box plots, histograms, and scatter plots) could easily handle raster data as they only rely on the numeric values, and not the spatial component and definition of neighbours. The challenge with ESDA on raster datasets is the calculation of local indicators of spatial autocorrelation (LISA), which are the most computationally intensive part of the ESDA workflow due to the large number of permutations (Anselin et al., 2022).

The xarray[32], rasterio[33], and rioxarray[34] Python packages are used to work with the raster data structures. Just as with pandas, xarray is geared towards non-spatial datasets that could have multiple dimensions and variables. Geopandas is an extension of the pandas library. Similarly, rasterio and rioxarray are extensions of the xarray library, catering for spatial datasets by importing common spatial raster formats such as GeoTIFF or netCDF files.

4.2.1 Strategies for LISA Calculations

LISA are regarded as the most computationally intensive part of an ESDA workflow (Amgalan et al., 2022; Anselin et al., 2022; Paudel and Puri, 2022). The optimisation of these calculations is thus vital in minimising the overall time to generate the autoESDA report. The vector component of autoESDA supports local Geary's C and local Moran's I. Similarly, this section will only discuss these two measures with the intention of the raster functionality of autoESDA mirroring that of vector datasets.

[32] https://docs.xarray.dev/en/stable/
[33] https://rasterio.readthedocs.io/en/stable/intro.html
[34] https://corteva.github.io/rioxarray/html/index.html

Three strategies for calculating LISA on raster datasets have been identified. These will be described along with their advantages and disadvantages. The first strategy processes the raster datasets in its original array format, with the second and third strategies requiring the raster dataset to first be converted into a GeoDataFrame comprised of a regular lattice of polygons (cells). This process is known as vectorisation, and while is does require additional processing, the trade-off could be worth the use of vector-based processing.

4.2.1.1 Strategy 1: Raster + PySAL

The first strategy relies purely on the newly developed methods to create spatial weights matrix using PySAL. The weights matrix is then used with the spatial autocorrelation function (`Moran`, `local_moran`, `Geary`, `local_geary`) in the PySAL library. One could reasonably assume that the calculation of LISA using this approach would be more efficient than with vector polygons due to the simplicity of the raster data model.

Unfortunately, the functionality of generating queen's first order spatial weights matrix from raster datasets and LISA calculations is relatively new to PySAL, which means that there is still room for improvement. A queen's case weight matrix with a first order contiguity will be used for these experiments. Sheckhar et al. (2020) explained that the generation of weights has the potential of being optimised using dask (a Python library for parallel computing)[35], however this has not yet been done. Additionally, the lack of documentation meant that although some PySAL LISA functions have the potential to be parallelised using joblib's loky[36] (a library that allows for Python functions to be parallelised) – this could not be achieved for this comparison. It would be interesting to revise this approach in the future once the functionality has been given the opportunity to mature.

4.2.1.2 Strategy 2: Vectorise + PySAL

The second strategy is to use the PySAL functionality (`Moran`, `local_moran`, `Geary`, `local_geary`) on the vectorised data. This increases the processing required before being able to create the spatial weights matrix or calculate LISA. The PySAL functions used for strategy 1 will be identical to those used in this strategy, making this strategy unlikely to be faster due to the additional time required to vectorise the dataset.

4.2.1.3 Strategy 3: Vectorise + pygeoda

The final strategy is to use the pygeoda LISA functionality. Currently, pygeoda only supports vector datasets, meaning that the raster dataset would first need to be vectorised. Just as with

[35] https://www.dask.org/
[36] https://github.com/joblib/loky

strategy 2, this additional step would increase the processing required, however this trade-off could be worth the ability to then calculate LISA in pygeoda. This library functions as an API to the libgeoda library, which is written in C++, which is known to be faster than Python. pygeoda outperformed PySAL when calculating LISA for the vector functionality of autoESDA, and there is reason to believe this approach may be the most efficient for raster datasets too.

4.2.2 Comparing LISA Strategies

In order to identify the optimal strategy for LISA calculations with raster datasets, the strategies needed to be compared in terms of both performance and scalability. To evaluate this, datasets of different sizes and levels of noise (variance) were used, and the time it took to perform different tasks were recorded. Additionally, each strategy was broken down into six tasks to assist in the identification of which tasks require the most amount of time to complete.

4.2.2.1 Datasets

Each strategy was tested with three datasets[37], each with three bands and each of a different size. The variance (σ^2) of the bands were inflated such that band one had a small variance, band two a medium variance, and band three having the greatest variance. This was achieved by generating a variance surface of the same shape as the original raster that followed a normal distribution, but with increasing standard deviation. The simulated surfaces of low, medium, and high variance were each added to the original raster to create bands 1, 2, and 3 respectively. The code used to simulate these surfaces can be viewed in Appendix C. Table 10 gives the statistical content summary of each band in each dataset. One can observe the increasing minimum/maximum values and standard deviation of the bands in each dataset as the band number increases.

[37] All three datasets are actually clips of different sizes areas from the same CHIRPS dataset which was obtained from https://www.chc.ucsb.edu/data/chirps

Table 10: Statistical summary of datasets used to test raster LISA calculations

	Dataset 1			Dataset 2			Dataset 3		
Size	231 x 180			575 x 409			903 x 593		
Valid %	70.90			88.56			87.88		
Valid Cells	≈ 29 481			≈ 208 271			≈ 470 579		
Band	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
Min	0.49	3.09	1.92	0.72	3.00	3.36	0.00	0.05	0.17
Max	320.89	429.81	690.59	732.79	1 340.99	2 281.75	449.01	732.03	1 103.95
Mean	48.03	106.88	165.00	142.64	316.11	488.65	49.58	157.79	267.70
Std	35.01	56.21	95.74	104.63	167.46	281.47	69.27	107.54	178.58

4.2.2.2 Tasks

The LISA calculations that will be included in the autoESDA report can be broken down into six tasks, these are described below.

Task 1: Preprocessing

This involves the vectorisation of the raster dataset and is therefore only relevant in strategy 2 and 3. When vectorisation occurs, the raster grid is only vectorised once and the values from each band are transferred to the newly created GeoDataFrame. This means that the vectorisation process that makes up task 1 is only carried out once per dataset, instead of once per band. Tasks 2 – 6, however, will be repeated for each band that exists within a dataset.

Task 2: Spatial Weights Creation

This task involves the creation of the spatial weights object that will be used in the LISA calculations. For strategy 1 this requires the use of the `Queen.from_xarray` function to create the sparse queen's case weight matrix and then calls the `WSP2W` function to convert the sparse weights matrix to a full one. For strategy 2 and 3, the NoData values are first removed from the GeoDataFrame before the `Queen.from_dataframe` function is called. The weights matrix is then transformed using the `transform` function. All functions used in task 2 are available through the libpysal weights module.

Task 3: Local Moran Calculation

Task 3 and task 5 are the most computationally intensive functions of the tasks and therefore require the most time to complete. In task 3, the local Moran object is created by calling the `moran.Moran_Local` function in the esda PySAL module.

Task 4: Local Moran Plotting

Once created, the attributes of the local Moran object can be called to populate the rows of the GeoDataFrame with their labels and significance values. The values that have a significance of 0.05 or lower are then plotted.

Task 5: Local Geary Calculation

In task 5, the local Geary object is created by calling the `geary.Geary_Local` function in the esda PySAL module.

Task 6: Local Geary Plotting

Once created, the attributes of the local Geary object can be called to populate the rows of the GeoDataFrame with their labels and significance values. The values that have a significance of 0.05 or lower are then plotted.

4.2.2.3 Methodology

All experiments were conducted on a desktop computer that has a 64-bit operating system with Windows 10 Enterprise installed. It has an Intel Xeon CPU E3-1270 v6 processor runs with a clock speed of 3.80 GHz. There is also 64 GB of RAM installed as well as 32 GB graphics card.The code was run in an Anaconda environment with Python 3.9 installed, including any dependencies required. All data was saved onto the local SSD storage to ensure efficient data retrieval and writing.

The code for each test simulation was run at least three times and the average of these runs was used in the results section for this comparison. If the values in the first three runs differed noticeably, then the process was rerun until three sets of times that were similar to each other were produced. These were used to calculate an average and the irregular results were discarded. To allow for a consistent comparison, the computer was not used for anything else while the code was running.

Each of the three datasets had three bands (low, medium, and large variance), which meant that 16 timestamps were recorded for each dataset (one for task 1, and one for tasks 2 – 6 for each band). The simulation was run three times for each dataset, and the average values of the three runs was used for the results and discussion in the following section.

4.2.2.4 Results

The average time to generate the spatial weights matrix, local Moran object, and local Geary object were each recorded and can be viewed in Table 11. As expected, the size of the dataset is a contributing factor to the LISA processing time. Results were generated significantly faster for Dataset 1, regardless of the strategy used, followed by Dataset 2, and Dataset 3 which had the longest processing time. In all cases, the total processing time for the first band in each dataset is the greatest. This is expected as the first time a spatial weights matrix is created, PySAL indexes the dataset which makes the processing of bands 2 and 3 more efficient. Likewise, the time required to vectorise (task 1) – required in strategy 2 and 3 - is only allocated to band 1 due to the fact that this process only needs to be carried out for a dataset and it will be the same regardless of how many bands there are. This is because the *cost* of vectorisation is assigned only to the first band.

It was expected that tasks 3 and 5 (LISA calculations) would require the most amount of time to run out of all the tasks. However, not all values in Table 11 support this hypothesis, which only holds true for Dataset 2 and 3 for the PySAL Vector and PySAL Raster strategies. In the case of the pygeoda Vector strategy, the plotting times were quite often greater than that of the LISA calculation time.

Figure 13, Figure 14, and Figure 15 illustrate the cumulative processing time (for tasks 1 – 6) for all bands in Dataset 1, 2, and 3 respectively. The y-axis scale differs for each plot, which means that it is not immediately evident that the datasets have different processing times; but upon investigation of the values, it is clear that Dataset 3 (Figure 15) required significantly more processing time as evident in the large range of values on the y-axis. It is interesting to note that the time required to plot (tasks 4 and 6) the vectorised raster (as vectorised cells) in strategy 2 and 3, is significantly more than plotting the results as a raster surface as in strategy 1 – this is most likely due to the fact that the cells are plotted as individual polygons, rather than as a surface.

Table 11: Timing results (seconds) for different raster LISA strategies

		PySAL Raster			PySAL Vector			pygeoda Vector		
	Task	Band 1 Low σ^2	Band 2 Med σ^2	Band 3 High σ^2	Band 1 Low σ^2	Band 2 Med σ^2	Band 3 High σ^2	Band 1 Low σ^2	Band 2 Med σ^2	Band 3 High σ^2
Dataset 1	1	-	-	-	2.64	-	-	3.55	-	-
	2	3.86	0.30	0.30	1.91	1.97	2.05	0.14	0.15	0.15
	3	14.10	1.53	1.58	1.36	1.62	1.42	1.35	1.36	1.36
	4	0.16	0.21	0.12	2.46	1.67	1.16	2.54	2.88	3.12
	5	3.46	1.56	1.51	1.64	1.63	1.75	1.45	1.42	1.43
	6	0.13	0.12	0.11	4.56	1.51	1.04	4.94	4.93	5.10
	Total	**21.71**	**3.71**	**3.62**	**14.57**	**8.38**	**7.42**	**13.97**	**10.73**	**11.15**
	Total	**29.04**			**30.37**			**35.85**		
Dataset 2	1	-	-	-	14.15	-	-	20.60	-	-
	2	6.10	1.97	1.86	13.17	13.69	14.28	1.15	1.26	1.24
	3	102.80	91.75	91.55	89.75	90.42	88.81	24.83	25.02	25.47
	4	0.46	0.22	0.19	21.69	14.77	8.84	22.09	23.63	25.70
	5	91.83	91.76	92.93	91.71	89.89	90.07	28.11	29.22	29.45
	6	0.46	0.24	0.15	33.83	12.30	6.84	33.90	34.61	35.39
	Total	**201.64**	**185.94**	**186.69**	**264.29**	**221.06**	**208.84**	**130.68**	**113.73**	**117.24**
	Total	**574.27**			**694.19**			**361.65**		
Dataset 3	1	-	-	-	30.94	-	-	46.33	-	-
	2	7.72	4.28	3.94	29.06	29.93	31.75	2.69	3.05	3.13
	3	465.09	453.28	453.61	446.85	450.28	450.24	69.26	69.72	70.16
	4	1.01	0.32	0.27	53.45	30.86	21.82	55.11	58.78	62.10
	5	456.56	455.61	455.01	451.94	453.28	453.44	77.77	78.19	78.29
	6	1.00	0.26	0.21	73.02	26.58	15.39	74.56	75.93	77.76
	Total	**931.37**	**913.74**	**913.04**	**1 085.26**	**990.93**	**972.65**	**325.73**	**285.67**	**291.44**
	Total	**2 758.15**			**3 048.84**			**902.84**		

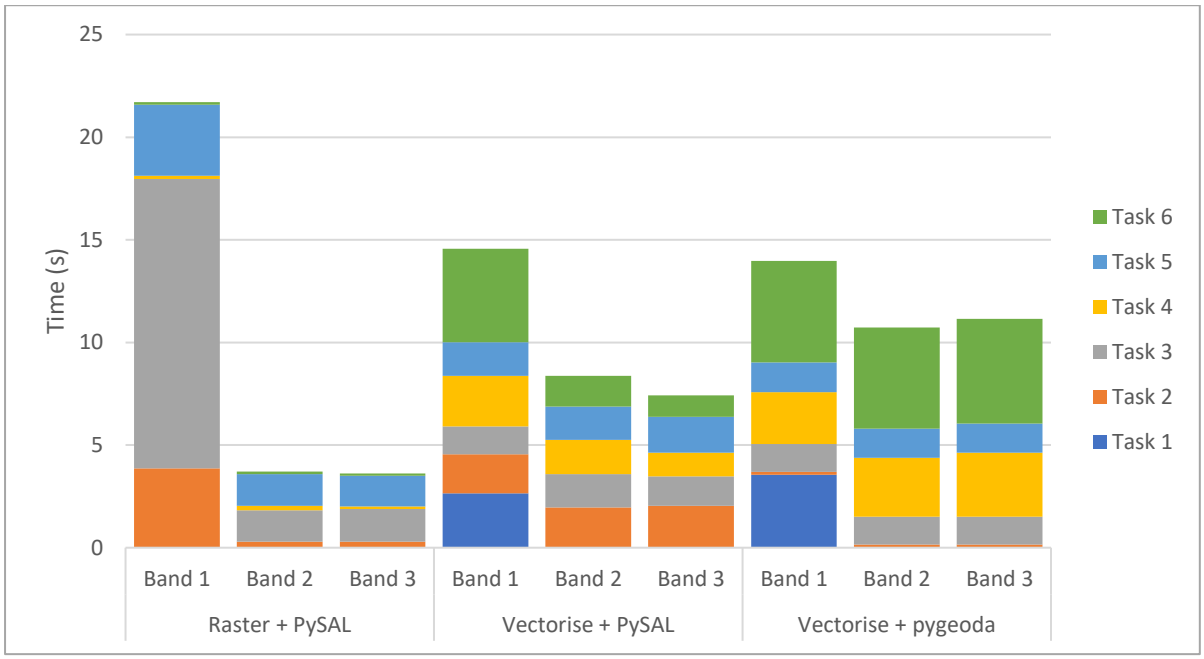


Figure 13: Stacked bar chart of average times for LISA calculations on Dataset 1

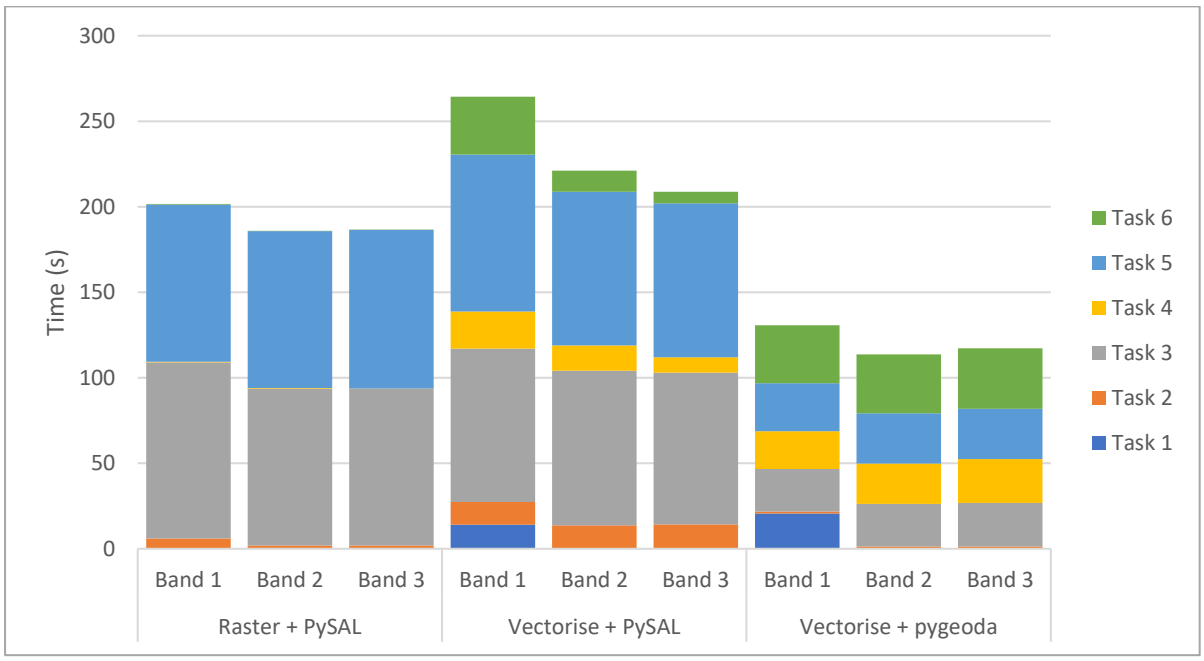


Figure 14: Stacked bar chart of average times for LISA calculations on Dataset 2

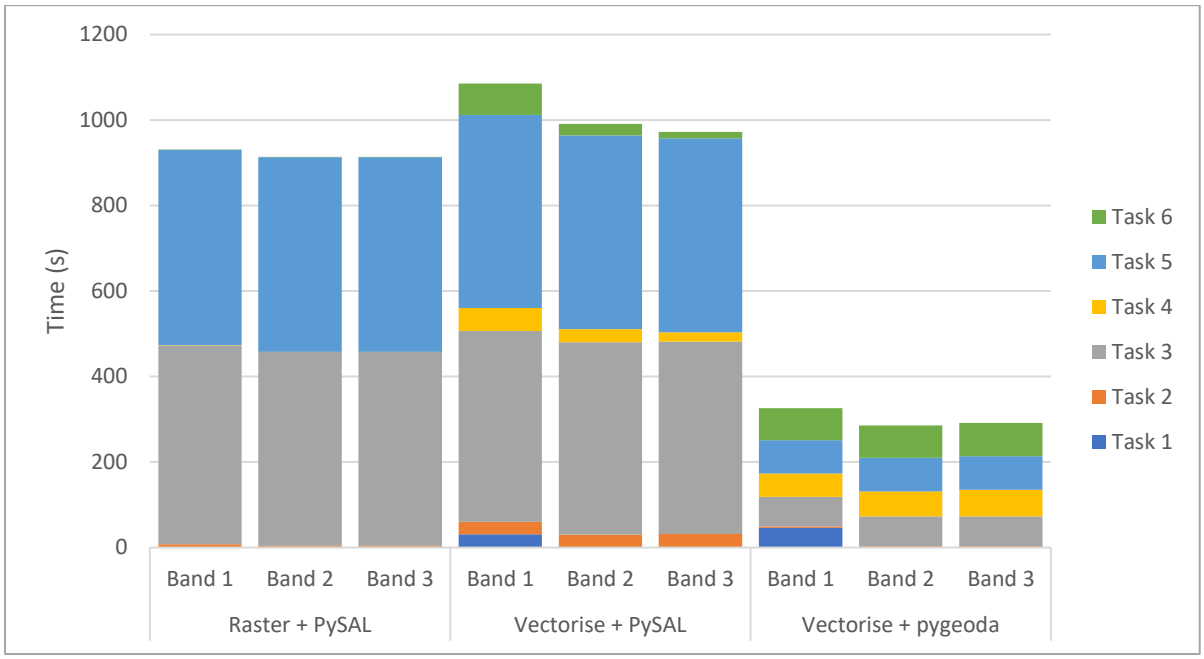


Figure 15: Stacked bar chart of average times for LISA calculations on Dataset 3

The other trend that is noticeable in Figure 13, Figure 14, and Figure 15 is that the first band of each dataset has the longest processing time. This is expected as strategy 1 will cache the raster array when processing the first band, which will optimise the processing of band 2 and 3. Similarly, the vectorisation time (task 1) that is required for strategy 2 and 3, is allocated only to the first band in the dataset. This would address the reason why band 1 requires the most processing time, regardless of the strategy or the size of the dataset.

If one were to ignore the cumulative time for band 1 and compare only the cumulative processing times for bands 2 and 3, the comparison becomes quite interesting. The time to process each of the two bands appear identical in strategy 1. Their processing time decreases, however, as the band number increases for strategy 2, while the processing time increases as the band number increases for strategy 3. This trend holds true regardless of the dataset size. One would expect that the time to process the bands would increase as the band number increases, as the test datasets were altered such that the noise increased for each band. This could indicate that the effect of noise in a dataset (band) on processing time is greater for pygeoda (strategy 3) than PySAL, as increasing processing times are not evident for strategy 1 and 2 which make use of the PySAL library.

4.2.2.5 Discussion

The comparison of times for each of the strategies for LISA with raster data illustrates that pygeoda is more efficient when processing large datasets. The smallest test dataset used was 231 by 180 cells and it is expected that the majority of the datasets that autoESDA would be

used for would be larger than that. This makes the pygeoda strategy (strategy 3) the preferred one to incorporate into the autoESDA library in order to calculate LISA. This comes as no surprise as Anselin et al. (2022) came to the same conclusion when comparing pygeoda to PySAL. pygeoda's underlying C++ codebase allows its processing to be faster than that of Python-based PySAL.

Although this is currently regarded as the most efficient strategy, it would be interesting to revisit these comparisons in the future. One of the advantages of the raster storage format is that their grid-like nature, which is made up of multiple numpy arrays, should enable it to handle calculations such as LISA more efficiently than vector data GeoDataFrames (Sapre and Vartak, 2020). The PySAL weights functionality is still relatively new, and one could expect performance improvements that may enable it to rival the processing time of pygeoda.

For this reason, pygeoda will be incorporated into the raster module of autoESDA. While it is near identical to the pygeoda code used in the vector module, the decision has been made to keep the code separate, to allow for updates and/or changes to be easily made in the future once the raster functionality has been improved.

While Moran's I and Geary's C and their local counterparts are regarded as the most popular spatial autocorrelation statistics, they are not suitable for use on large datasets (Amgalan et al., 2022). While Moran's I and Geary's C may be suited for vector datasets, the grid-like structure of a raster very quickly translates to a large dataset of vector cells. Although these statistics will be incorporated into the autoESDA raster module due to their popularity, it may be beneficial in the future to include other measures of spatial autocorrelation that are more suited to large datasets (Amgalan et al., 2022). Another approach could be the incorporation of parallelization and other optimisation strategies (Paudel and Puri, 2022).

4.3 Updated Architecture

The first iteration of autoESDA was not designed with future improvements in mind. It consisted of a single script file that accepted a GeoDataFrame as an input and saved an output HTML file to the working file directory. This monolithic script was made up of multiple functions being called and numerous HTML strings being passed between them. This design was inefficient for multiple reasons. Firstly, there was an unnecessary number of variables stored that were only called once, which meant that memory was not conserved. Secondly, there was no clear structure to the code, which, combined with the various HTML strings being concatenated at various parts of the scripts' runtime, meant that any change was a challenging task.

The original architectural design was based solely on support for vector data, making the decision to include raster functionality the primary motivator to restructure the autoESDA architecture. The newer, more modular design made it easier to address any existing errors in the code, while also being able to make minor improvements (discussed in Section 4.4) and incorporating the raster functionality discussed in Section 4.2. These tasks have all been easier to implement due to the updated architectural design, as the components are no longer as tightly coupled as they were in the first iteration. Figure 16 illustrates the architectural design differences through the use of a package diagram for each iteration.

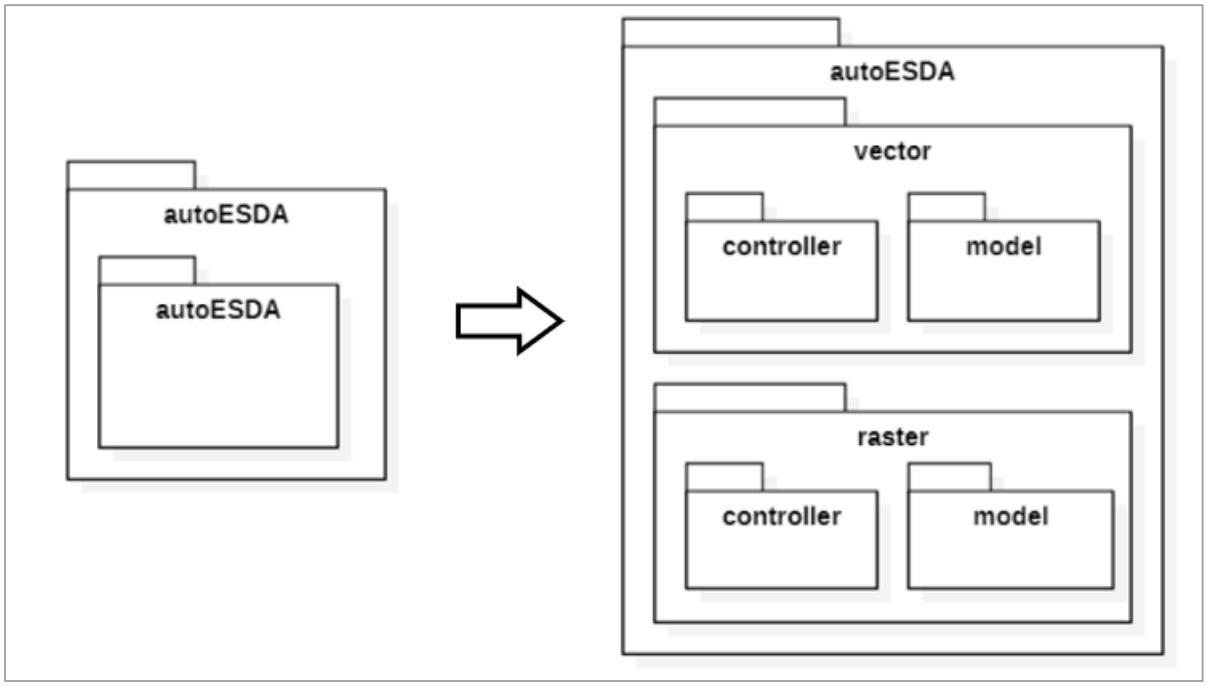


Figure 16: Package diagrams illustrating the architectural design of the first (left) and second (right) iterations of autoESDA

The Model-View-Controller (MVC) architecture is an approach that splits the design of a software platform into three components (Syromiatnikov and Weyns, 2014). The Model consists of a set of classes that make up the structure of the data in the system, the View is what the user engages with, while the Controller handles the input and output of the software by functioning as the link between the Model and the View (Syromiatnikov and Weyns, 2014).

The design of the second iteration autoESDA is loosely based on the MVC approach. The library is split into two modules – vector and raster. Each module works with their namesakes' data format - the vector module accepts a GeoDataFrame as its input, and the raster module accepts a xarray DataArray as its input. These are referred to as the *input vector* and *input raster* respectively. Each module is made up of two components – a model and controller. The model defines the classes that are relevant for that module (depending on whether vector or

raster data is being used). The class functions are called to generate the components that make up the output report. The controller currently acts as the view and controller as these two components are still tightly coupled. The controller makes numerous calls to the model and then combines the results into the HTML report which is the final output of autoESDA. Section 4.3.1 and Section 4.3.2 will discuss the model and controller respectively for both the vector and raster modules.

4.3.1 Model

One of the major changes that has been brought about by the redesign of the autoESDA library is the use of classes to structure the various ESDA outputs that can be generated from a dataset. This enables one to implement new functionality more easily than on the previous design. The raster and vector models differ slightly due to the different input formats; however they are designed to be as similar to each other as possible. Each model is based on a parent-child class schema allowing for only one instance of parent to be created, which have at least one instance of the child class relating to it. For the vector model the parent class is referred to as a `Dataset`, while the child class is referred to as a `Variable`. In the raster model the parent class is referred to as a `Raster`, while its child class is referred to as a `Band`.

Each model is built on the premise that the parent class only needs to be created once for the input dataset, and that its creation would automatically generate the required number of child classes based upon the number of variables (vector datasets) or bands (raster datasets). When the autoESDA report is generated, the relevant function or attribute from the classes are called, which then returns an object to include in the report. Depending on the output, this object may take on a variety of forms, such as a figure or DataFrame.

4.3.1.1 Vector Module

The vector model is made up of a `Dataset` parent class and a `Variable` child class, as illustrated in Figure 17. The `Dataset` class is created from the input vector, and its attributes and functions relate to the entire dataset. It's first attribute, `gdf`, is the input vector which is stored as a GeoDataFrame. The `numeric_columns` attribute is a list of names of the numeric columns in the dataset – these are the only columns that are included in the calculations as other data types are not yet supported. The final two attributes are `pygeoda_weights` and `pysal_weights` – each are spatial weights objects for their respective library. The pygeoda weights object is used for the LISA calculations, while the PySAL weights object is used for the calculation of global spatial autocorrelation statistics. Although it is more efficient at calculating LISA, pygeoda, at the time of writing unfortunately does not support global spatial

autocorrelation statistics. This necessitated the creation of a PySAL weights object which enable the calculation of global measures of spatial autocorrelation using PySAL. Fortunately, these global functions are less computationally intensive than their local counterparts.

The `init` function forms part of any Python class and is used to create a new instance of a class. In this case, a new `Dataset` is created. This function requires the specification of a GeoDataFrame as its only parameter. Each of the attributes described in the paragraph above (`gdf`, `numeric_variables`, `pygeoda_weights`, and `pysal_weights`) will be derived once the `init` function is called. These attributes can then be accessed by the functions in the `Dataset` class and do not need to be derived each time they are required, thus avoiding unnecessary extra processing.

The `overview_statistics` function returns a DataFrame of information relating to the dataset. This includes the coordinate system, number of features and attributes, and which are the numeric attributes which have been included in the report, or attributes of other data types that have been excluded. The purpose of the overview statistics table is to provide the user with quick information relating to the makeup of the dataset and what information has been included or excluded in the autoESDA report.

The `study_area_figure` function returns a basic plot of the dataset made up of black outlines and hollow polygons. The purpose of this map is not to be visually appealing, but to illustrate the shape of the dataset to the user so that any irregularities can immediately be identified. The coordinates (based on the Coordinate Reference System (CRS) of the input vector) are also included on the axes so that the user can assess whether they are consistent with what would be expected.

The `numeric_variables` function initialises a new `Variable` class for each `numeric_column` in the `Dataset` and returns these as a list of `Variable` objects. To analyse each `Variable` in the `Dataset`, this list can be iterated through, or one could access each `Variable` individually by calling the list and specifying the respective index relating to that `Variable`.

The `dataset_statistics` function returns a DataFrame containing the descriptive statistics for each `Variable` in the `Dataset`. The output DataFrame is created by combining the DataFrames returned from the `variable_statistics` function for each `Variable`.

The `dataset_sample` function returns a DataFrame consisting of ten randomly selected features and their attributes from the input vector. The geometry column is excluded as its

contents are often lengthy and are difficult to display in its entirety in the output report. The sample is selected using the pandas `sample` function which returns the user specified number of randomly selected rows from a DataFrame (GeoDataFrame) – in this case, the input vector.

The `correlation_figure` function returns a figure made up of three correlation heatmaps. Three correlation matrices (Pearson, Kendall, and Spearman) are calculated using the pandas `corr` function and are then coloured as a heatmap using seaborne's `heatmap` function.

Finally, the `pairplot_figure` function returns a pairwise plot of each `Variable` in the `Dataset`. The plot is generated using the `pairplot` function from the seaborne library.

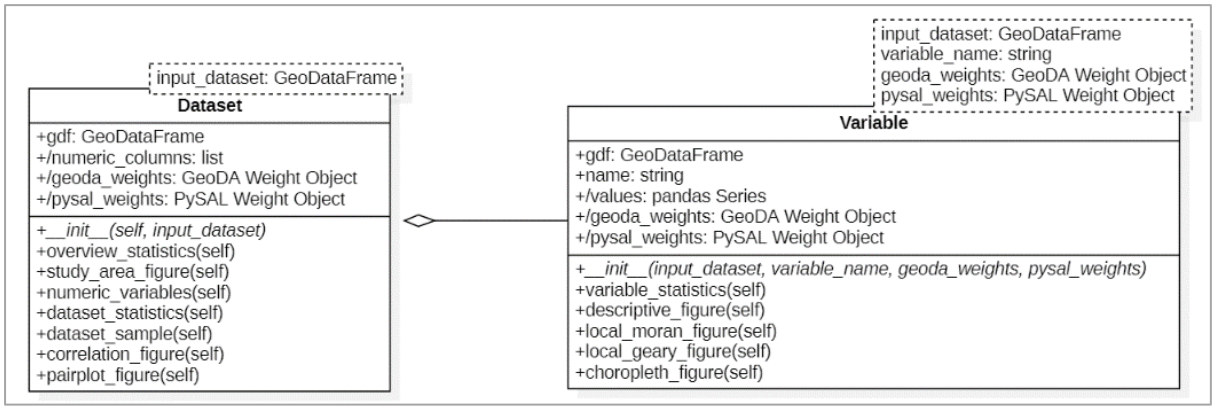


Figure 17: UML class diagram for the vector model

A `Dataset` will have a `Variable` class for each numeric column in the input vector. A `Variable` helps encapsulate the various components related to that specific column, and in doing so it enables the same code to be reused regardless of how many instances of the `Variable` class are created for the `Dataset`.

A `Variable` is instantiated by calling the `init` function, which requires the specification of the parent GeoDataFrame, column name (string), and the parent PySAL and pygeoda weights objects. Each of these parameters are stored as attributes of that instance of the `Variable` class. Additionally, the values (observations) relating to that variable are extracted from the original GeoDataFrame. These are stored as a pandas series and can be accessed by calling the `values` class attribute.

The first function in the Variable class is `variable_statistics`. This function uses the pandas `describe`, `skew`, `kurt`, `nunique` and `isna` functions along with the global `Moran` and `Geary` functions from PySAL. These functions yield various descriptive statistics which are combined into a DataFrame that is returned when the function is called.

The `descriptive_figure` function returns a figure made up of three axes. The first is a box plot which is generated using the matplotlib `boxplot` function; the second is a probability histogram which is generated using seaborne's `histplot` function, and the third is a cartogram which is generated using geoplot's[38] `cartogram` function.

The `local_moran_figure` and `local_geary_figure` operate almost identically. Each function uses the respective pygeoda `local_moran` or `local_geary` function to create a LISA object.

When creating a LISA object for local Moran or local Geary, pygeoda automatically assigns a value of zero to fields that have been left empty or contain a *NaN*. This is misleading as, if no error is given, one can still plot the results for which all of the polygon features will display as High-High clusters (local Moran) or Negative clusters (local Geary) – both with very high significance. To address this issue, the following line of code has been added:

```
undefined_values = values.isna().tolist()
```

This line creates an array of Boolean (True/False) values which will then be used when the local Moran or local Geary function is called. The array acts like a mask and ensures that missing values are not considered in the construction of the LISA object.

Once created, the LISA objects are then passed to the `_plot_lisa` function. This global function in the autoESDA vector model was created to avoid the duplication of code for each statistic and will accept a LISA object and return a figure containing both a cluster map and a significance map.

Finally, the `choropleth_figure` function returns a figure made up of six choropleth maps – each with a different classification scheme. The maps are plotted using matplotlib using classification schemes provided by the mapclassify library.

4.3.1.2 Raster Module

Just as with the vector model, the raster model is also made up of parent and child classes, known respectively as `Raster` and `Band`, as illustrated in Figure 18. The only parameter required to create a `Raster` class is the input raster (in the form of a rioxarray DataArray object). The `init` function is called to create an instance of the `Raster` class. When a `Raster` is created, three attributes are derived, namely: `vectorised`, `geoda_object`, and `geoda_weights`. The `vectorised` attribute is the output GeoDataFrame once the input

[38] https://residentmario.github.io/geoplot/index.html

raster has been vectorised. The `geoda_object` attribute is then created from the vectorised GeoDataFrame and is used to create the geoda weights object, which is stored under the `geoda_weights` attribute. These attributes are be used by the functions within the `Raster` class.

The `overview_statistics` function returns a DataFrame containing information relating to the raster dataset. This includes the CRS, number of bands/rows/columns, resolution, extent, and other metadata that could be extracted from the input raster. These are obtained by calling the `crs`, `shape`, `resolution`, `bounds`, and `attrs` methods which are defined in the xarray DataArray object or it's rioxarray extension.

The `bands` function is similar to the `numeric_variables` function in the `Dataset` class of the autoESDA vector model. This creates a new instance of the `Band` child class in the raster autoESDA model for each band in the dataset. The `bands` function returns a list of `Band` objects, its length being dependent upon the number of bands in the raster dataset.

The `correlation_figure` and `pairplot_figure` functions are identical to the functions of the same name in the vector model, as they use the GeoDataFrame created by vectorisation. The code has been purposely duplicated to allow for one to easily refactor it in the future so that it is built on the input raster rather than on its vectorised form. This would allow it to be altered more easily in the future, should there be a need to use a different, more efficient strategy.

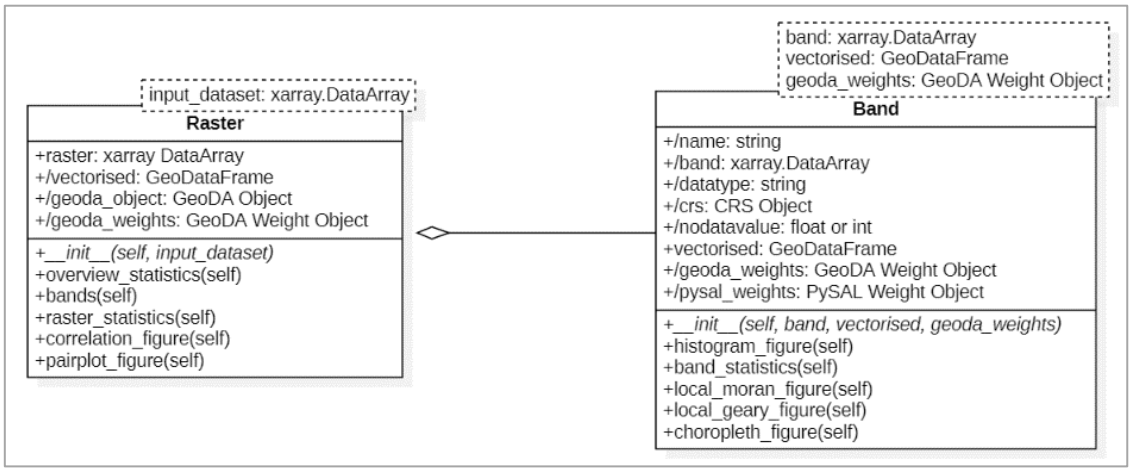


Figure 18: UML class diagram for the raster model

Each `Raster` class will have at least one `Band` relating to it. The `Band` class helps encapsulate various components that relate to that specific band, avoiding the need to duplicate code. To create an instance of a `Band` class, three parameters are required. These

are: a band xarray DataArray object, the vectorised GeoDataFrame, and the pygeoda weights object. These parameters are either assigned to or used to derive the ten attributes associated with a `Band` class, namely: `name`, `band`, `datatype`, `crs`, `nodatavalue`, `bandseries`, `bandnodataremoved`, `vectorised`, `geoda_weights`, and `pysal_weights`.

A `Band` is created by calling its `init` function which requires the specification of the band GeoDataFrame and a geoda weights object. When a `Band` is created, various attributes are automatically derived for use in the class functions. The `name` attribute is extracted from the band parameter required to create the `Band` class.

The `band` attribute stores a xarray DataArray object identical to the parameter, however with the addition of an attribute called `nodatavals` which is added using this line of code:

```
band = band.rio.update_attrs(new_attrs={"nodatavals": [band.rio.nodata]})
```

This is a necessary step as PySAL uses the hard coded `nodatavals` name for this attribute and, if it cannot be found, it will generate a spatial weights matrix for the entire grid extent rather than just the cells that have values. Consequently, when a local Geary or local Moran object was constructed, the result was an error due to the dimension mismatch of the spatial weights matrix and the already masked array of data values. The specification of an attribute with the name `nodatavals` rather than `nodata` or `_FillValue` which is used in rioxarray, means that a spatial weight matrix can still be created. This shortfall has been logged as an issue on the PySAL repository, and once it is addressed, the workaround described here will no longer be necessary.

The `datatype`, `crs`, and `nodatavalue` attributes are extracted from the band parameter which has stored attributes called `dtype`, `crs`, `nodata` respectively. The `vectorised` attribute stores the GeoDataFrame of the vectorised raster, with `geoda_weights` and `pysal_weights` referring to their respective spatial weights object for each library.

The `histogram_figure` function returns a figure in the form of a probability histogram of the values in the band. This is created using the `histplot` function from the seaborne library.

The `band_statistics` function returns a DataFrame of various descriptive statistics of the values in the band. The values are arranged in a pandas series and the pandas `describe` function is used for the aspatial statistics, while PySAL's `Moran` and `Geary` functions are used to calculate the global spatial autocorrelation statistics. All of these statistics are collated into the DataFrame before it is returned.

The `local_moran_figure` and `local_geary_figure` each return a figure with two plots – the respective LISA cluster map, as well as a map illustrating its significance. The code is practically identical to the functions of the same names in the `Variable` class. The code has been duplicated on purpose as the functionality works for the vectorised input raster, however it is expected that in the future there will be a more efficient strategy to compute these statistics that does not require vectorisation. This motivation is discussed in greater detail in Section 4.2.

The output of the `choropleth_figure` function is a figure of six choropleth maps with different classification schemes, just as is the case for the function with the same name in the `Variable` class. The technical component however differs slightly as the raster array is plotted, rather than a GeoDataFrame. Consequently, the classes need to be calculated separately and the cells with no values must be masked.

4.3.2 Controller

Just as with the design of the model for autoESDA, the workflow for the raster and vector modules are very similar, with slight differences due to each data structure and how each is processed. The workflow is carried out in the controller in each module, and it begins when the user calls the `generate_report` function. For both the raster and the vector modules, the user is required to specify an input dataset (GeoDataFrame or xarrayDataArray). The user can optionally specify a string to be used as the name for the output HTML report file, and in the case of the vector model, the user can specify a report name which will be displayed on the Summary Page of the report. Once the `generate_report` function is called, a timer is initialised (to time the report generation process) and the input dataset is used to create a `Dataset` or `Raster` object in the modules' model. These classes are created by calling their `__init__` constructor function.

Various calls are made to the `Dataset`/`Variable` and `Raster`/`Band` classes (located in the model) throughout the workflow and the response will take the form of either a pandas DataFrame or a matplotlib figure. The pandas `to_html` function is used to convert the DataFrames into an HTML compatible string, and the matplotlib figures are converted into a string which makes up a HTML image tag. The latter is done by calling the internal `_encode_image` function of the controller and specifying the figure that is to be encoded as a parameter.

For each controller there are four internal functions that are called which are responsible for creating the Summary, Variable/Band Information, Correlation, and About Pages respectively.

Each of these functions make numerous calls to the respective model to generate the various components of their namesake's page. These internal functions are used to group the calls made to the model. All the interactions between functions and different parts of the module are illustrated and numbered in the UML sequence diagrams for the vector module (Figure 19) and the raster module (Figure 20).

The generation of the Summary Page, and the Variable/Band Information Page differs between the two modules and will be discussed separately. The latter half of the workflow – generating the Correlation and About Pages, along with the final report output, will then be discussed in terms of both the vector and raster modules as this process is identical for each module.

4.3.2.1 Vector Module

The UML sequence diagram in Figure 19 illustrates the interactions between the different components in the vector module.

The `_summary_page` function is an internal function of the controller and requires the created `Dataset`, and optionally, the report title (string) as parameters. This function creates the Summary Page and groups together sequences 3 to 20 in Figure 19. When called, this function will call the `overview_statistics`, `study_area_figure`, `dataset_statistics` and `dataset_sample` functions of the `Dataset` class in the vector model. These functions will return a DataFrame of overview statistics, study area figure, DataFrame of dataset statistics and DataFrame sample respectively. The generation of the DataFrame of dataset statistics is made up from the individual DataFrames of the variable statistics. As a result, executing the `dataset_statistics` function necessitates the call of the `numeric_variables` function in the `Dataset` class which initialises a `Variable` class for each `Variable` in the `Dataset` class. The `variable_statistics` function is then called for each `Variable`, and the returned DataFrames are combined to create the DataFrame of dataset statistics.

The internal `_variable_information` function of the controller is slightly different to the other functions as it loops through each `Variable` in the model rather than dealing with the parent `Dataset` object. The other difference is that it returns two strings – a *tab string* and a *page string*. The *tab string* is the HTML string required to create the buttons along the top of the report to toggle between pages and needs to be created in the `_variable_information` as it requires the iteration through each of the `Band` objects to obtain their name to use as a label for these buttons. The *page string* is the HTML string that

represents the Variable Information Page for each `Variable`. Sequence numbers 21 to 31 in Figure 19 represent the workflow in this function. Each Variable Information page is made up of a table of variable statistics, descriptive figure, local Moran figure, local Geary figure and the choropleth figure. Each of these components are created by calling the `variable_statistics`, `descriptive_figure`, `local_moran_figure`, `local_geary_figure`, and `choropleth_figure` functions of the relevant `Variable` instance. A HTML string representing each Variable Information Page is returned which is combined into one long HTML string which is finally returned by the internal `_variable_information` function within the controller.

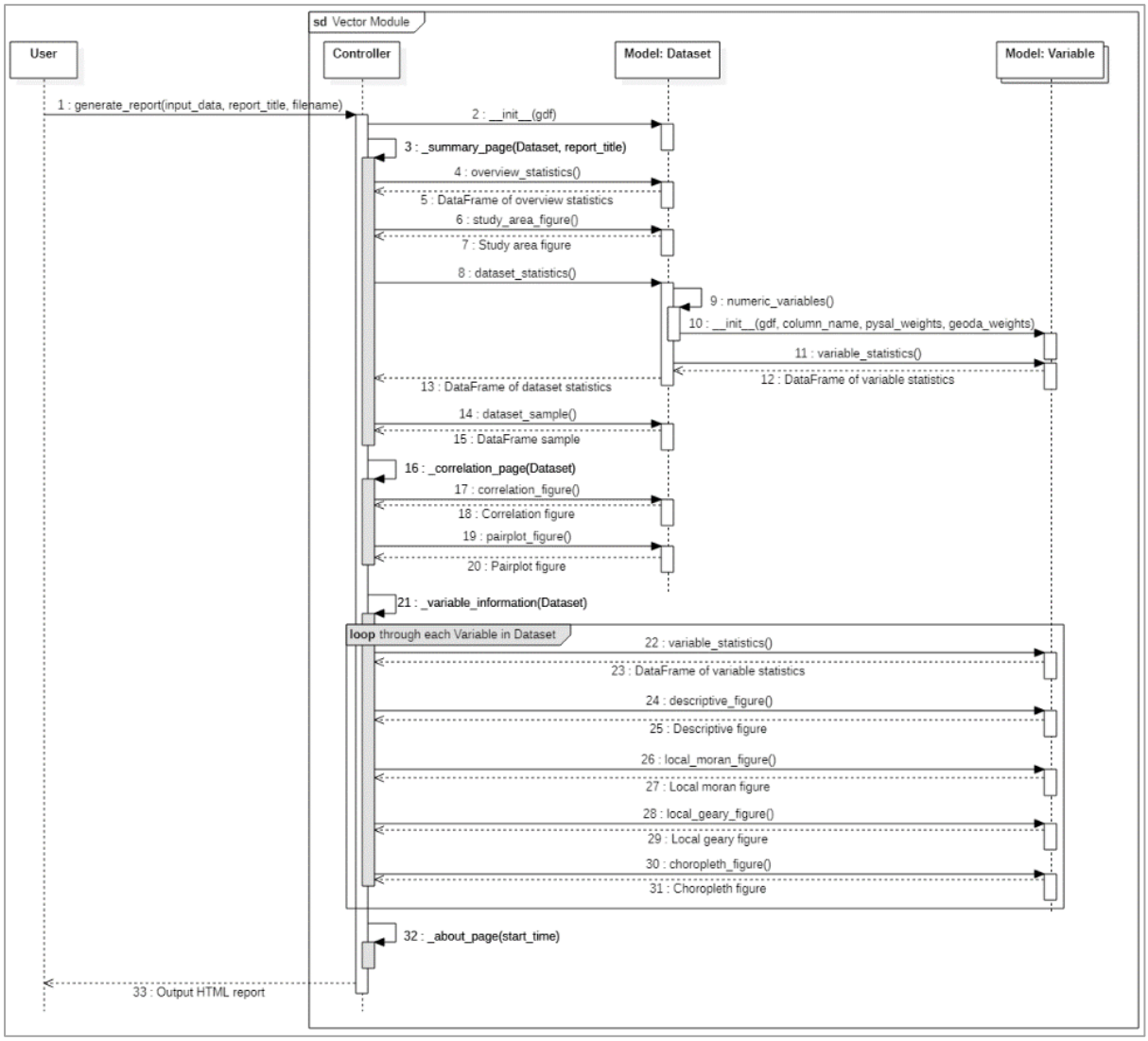


Figure 19: UML sequence diagram of the vector module workflow

4.3.2.2 Raster Module

The UML sequence diagram in Figure 20 shows the interaction between the different components of the raster module. These processes are all part of the raster module workflow, which is responsible for transforming the input raster (xarray.DataArray) into an autoESDA report which is then returned to the user.

The `_summary_page` function groups together the various interactions (sequences 5 to 15) between the model and controller that are required to create the Summary Page and requires the specification of a `Raster` object as its only parameter. The Summary Page is made up of a table of overview statistics and a table of raster statistics. These are returned as DataFrames when the `overview_statistics` and `raster_statistics` class functions of `Raster` are called. In order to generate the DataFrame of raster statistics, the `numeric_variables` function of `Raster` is called which creates an instance of `Variable` for each `numeric_column` in `Dataset`. To create an instance of `Band`, the `Raster` class in the model makes a call to the `__init__` (constructor) function of `Band` which requires the specification of the vectorised GeoDataFrame, column name (string), PySAL weights object and GeoDa weights object. This enables the code in the `raster_statistics` function of `Raster` to call the `band_statistics` function for each `Band`. The returned DataFrame from each of these calls are then combined to form one DataFrame that is returned when the `raster_statistics` function is called.

The internal `_band_information` function groups together the interactions (sequences 21 to 31) between the controller and instances of `Band` in the model to allow for the creation of the Band Information Page(s). A Band Information Page is made up of a table of band statistics, a histogram figure, local Moran and local Geary maps (each with their associated significance map) and a choropleth figure. These are generated by calling the `band_statistics`, `histogram_figure`, `local_moran_figure`, `local_geary_figure`, and `choropleth_figure` functions of the respective `Band`. A HTML string representing each Band Information Page is returned which is combined into one long HTML string which is finally returned by the internal `_band_information` function within the controller.

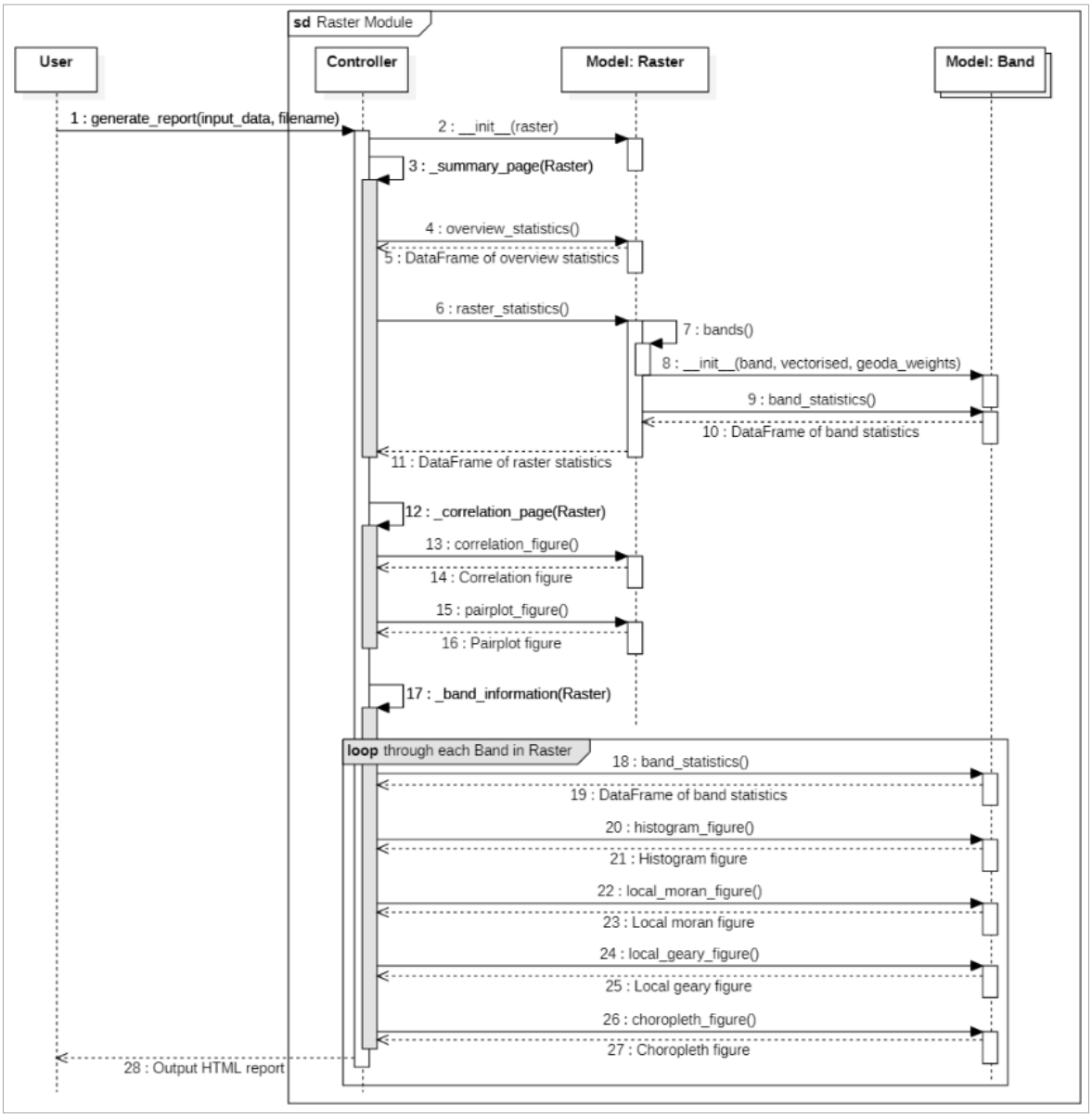


Figure 20: UML sequence diagram of the raster module workflow

4.3.2.3 Both Modules (Correlation and About Pages, Report Output)

The `_correlation_page` and `_about_page` functions are currently identical to each other and have been duplicated on purpose as a placeholder so that changes can easily be implemented for one module without affecting the other. The `_correlation_page` function accepts the relevant `Dataset/Raster` object and calls the respective `correlation_figure` and `pairplot_figure` functions. The output of each of these functions is a figure which will be added to the Correlation Page string once encoded. The `_about_page` function accepts the start time (derived when the `generate_report` is called) as a parameter which it uses to calculate the elapsed time to generate the report. The

remainder of the About Page is a static HTML string with the exception of a data and time text, which is calculated and then included in the about page string which is the output of the `_about_page` function.

The strings resulting from the functions described above are all concatenated with other HTML strings that are hard coded – these include the various elements that make up the structure and functionality of the report, such as inline CSS and JavaScript. The final HTML string is then written to a newly created file. If the user specified a name, the output file will be assigned that name with a *.html* extension, alternatively, if no name was supplied, a default of "*autoESDA-vector-report.html*" or "*autoESDA-raster-report.html*" will be used.

4.4 Other Minor Improvements

Just as is the case for every software project, autoESDA requires continuous improvements. These improvements are discussed in the remainder of this section and are grouped according to the element of the report to which they relate. While these can immediately be seen in the report, improvements are not just cosmetic and have been applied to the core workflow of the library, not just changes to the HTML layout.

4.4.1 General

The general improvements refer to the library as a whole, rather than one of the components of the report. This includes the addition of an About Page in the generated reports, as well as an updated layout of the HTML pages. The library is also available for download from PyPI[39] – the official Python package index, as well as on conda-forge[40] under the BSD 3-Clause license. As of October 2023, the library has been downloaded over 1200 times from PyPI and its GitHub repository[41] has been visited over 1900 times.

The About Page (pictured in Figure 21) appears in both the vector and raster versions of the generated HTML report. Its purpose is similar to that of a metadata file, in that it provides the user with miscellaneous information relating to the report as well as how it is generated and how it could be interpreted. This is achieved by informing the user of any default parameters used to generate the report – allowing the user to identify potential limitations. This is done by including a link to a User Guide which is stored in the autoESDA GitHub repository. The user will, however, require an internet connection to access the User Guide. In addition to a link to the User Guide, the About Page also contains a link to the source code of the autoESDA

[39] https://pypi.org/project/autoesda/
[40] https://anaconda.org/conda-forge/autoesda
[41] https://github.com/NicholasDeKock/autoESDA

library as well as the data and time that the report was generated, and how long it took to generate the report.

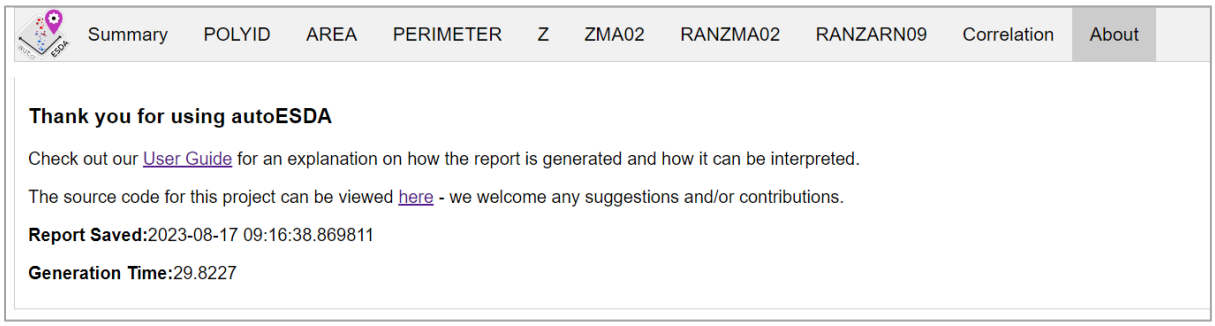


Figure 21: The About Page that is present in both the vector and raster autoESDA reports

The HTML layout has been improved so that it is more modular and can be customised more easily. Each page has adopted a CSS grid structure, made up of various grid elements. Each grid element can be positioned on the grid by specifying its row and column index, and how far it can stretch in each dimension. This also ensures that the position of each of the elements in relation to each other are fixed and should exceed the page size. The overflow for each grid element is set to "*auto*" which means that should the contents of the grid item exceed that which is allocated to the grid item, a scroll bar will appear. This will enable the user to toggle the view so that they can see everything in that grid element, without the overflow having an effect on any of the other elements on the page. Previously, the Variable Information Page was one matplotlib figure with multiple axes. The revamped design makes use of numerous figures which allow for each one to be assigned a different grid element and heading.

4.4.2 Summary Page

As a way of customising the report, the user may now optionally specify a name that will be displayed on the Summary Page above the study area plot. Previously this heading just read *Study Area*, and this has been left as the default text should a user not specify a name to use.

Within the dataset overview table, the names *rows* and *columns* have been replaced by *features* and *attributes* respectively. This was changed to avoid confusion, as one of the interview participants pointed out that the row/column terminology, although applicable when using a GeoDataFrame, could confuse the user into thinking they were working with a raster dataset. The updated names are consistent with the ESRI's definition[42] of an attribute table.

Several statistics have been added to the descriptive statistics table. These include skewness, kurtosis, and a count of null values. Skewness and kurtosis are calculated using the pandas

[42] https://support.esri.com/en-us/gis-dictionary/attribute-table

`skew` and `kurt` functions respectively, which by default skip null and non-numeric values. The null (NaN) values are counted by using the pandas `isna` function in conjunction with the `sum` function. Each null value returns a value of one which in turn is counted to determine the total number of null values. Global Moran's I and Geary's C and their respective p-values have also been included to allow the user to compare measures of spatial autocorrelation across variables.

The last improvement made to the Summary Page is the way in which the dataset sample is composed. Originally it was made up of the first and last five features in the GeoDataFrame, but this has now changed to include ten randomly selected features. This was changed to allow the user to identify faults in the dataset that could systematically be missing from the first and/or last features. All the changes described in this section can be viewed in Figure 22.

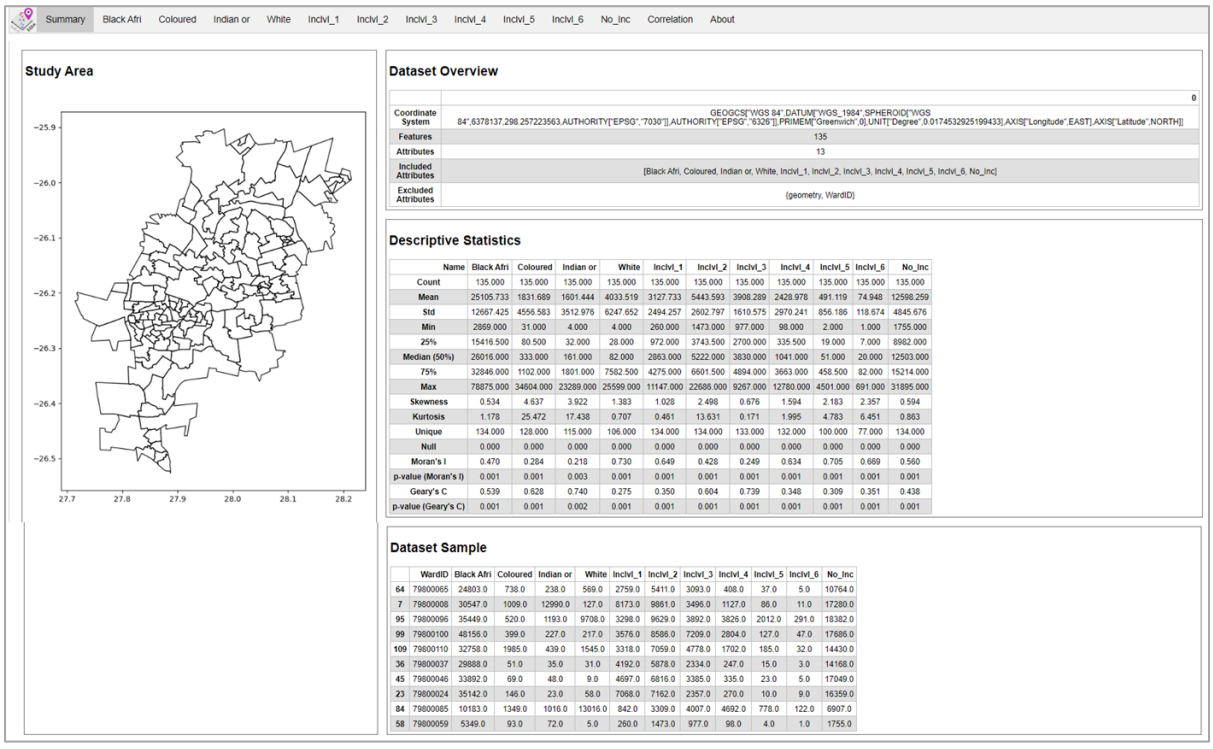


Figure 22: Updated Summary Page for autoESDA vector report

4.4.3 Variable Information Page

The most obvious change to the Variable Information Page (pictured in Figure 23) is that the layout is more complex. Previously the page was made up of one figure with multiple axes, however the updated layout is comprised of multiple figures and a table of descriptive statistics specifically related to the respective variable (a subset of the full descriptive statistics table that forms part of the Summary Page). Previously a frequency histogram was used, however this has been changed to be a probability histogram. This was done as the probability

histogram can be compared with that of other variables. It also avoids duplication as the main diagonal of the Pairwise Plot on the Correlation Page is comprised of a frequency histogram for each variable. A final method of visualising spatial heterogeneity is the newly incorporated cartogram which visualises the magnitude of the variable for each feature. This is created using the cartogram feature from the geoplot library.

The LISA section of the Variable Information Page has been completely revamped. The reference distribution with various statistics has been removed, as numerous interview participants were confused by this and did not see its relevance. The Moran's scatter plot has also been removed. In order to visualise the local Moran and local Geary calculations, a cluster map and significance map are used for each statistic. The calculations and classes are computed using the `local_moran` and `local_geary` function of the pygeoda library. pygeoda was chosen as it calculates faster than PySAL and its creation of a LISA object allows one to extract the cluster classes, labels, colours, and p-values more efficiently. Once extracted, each of these components are used in conjunction with geopandas to plot the cluster and significance maps. This new approach allows for the plotting of each of the results, and the figures can be more easily customised as they no longer form part of the output from the `plot_local_autocorrelation` function form the splot library, which was used previously.

The choropleth maps in the Variable Information Page also received a facelift. There are now six maps with the Boxmap, Equal Interval, Quantiles, Mean-Standard Deviation, Maximum Breaks, and Fisher-Jenks classification schemes. Each of these are plotted using a combination of geopandas and, by extension, mapclassify. The colour ramp has been reversed such that larger values are now visualised using darker colours - as is the norm in cartography (Slocum et al., 2014). The colour scheme has been changed from *viridis* to *YlOrBr* as the yellow-orange-brown palette is considered to be more neutral than the green-yellow-purple colours it used previously. This is important as the colour choice needs to be generic enough such that it avoids potential connotations when used for various types of data. For both the choropleth and LISA maps, the decision was made to place the legends below the map. Previously the legend placement was set to *best*, however it was evident that this has some limitations and an optimal legend location inside the map could not be chosen as the optimal placement would depend upon the shape of the dataset.

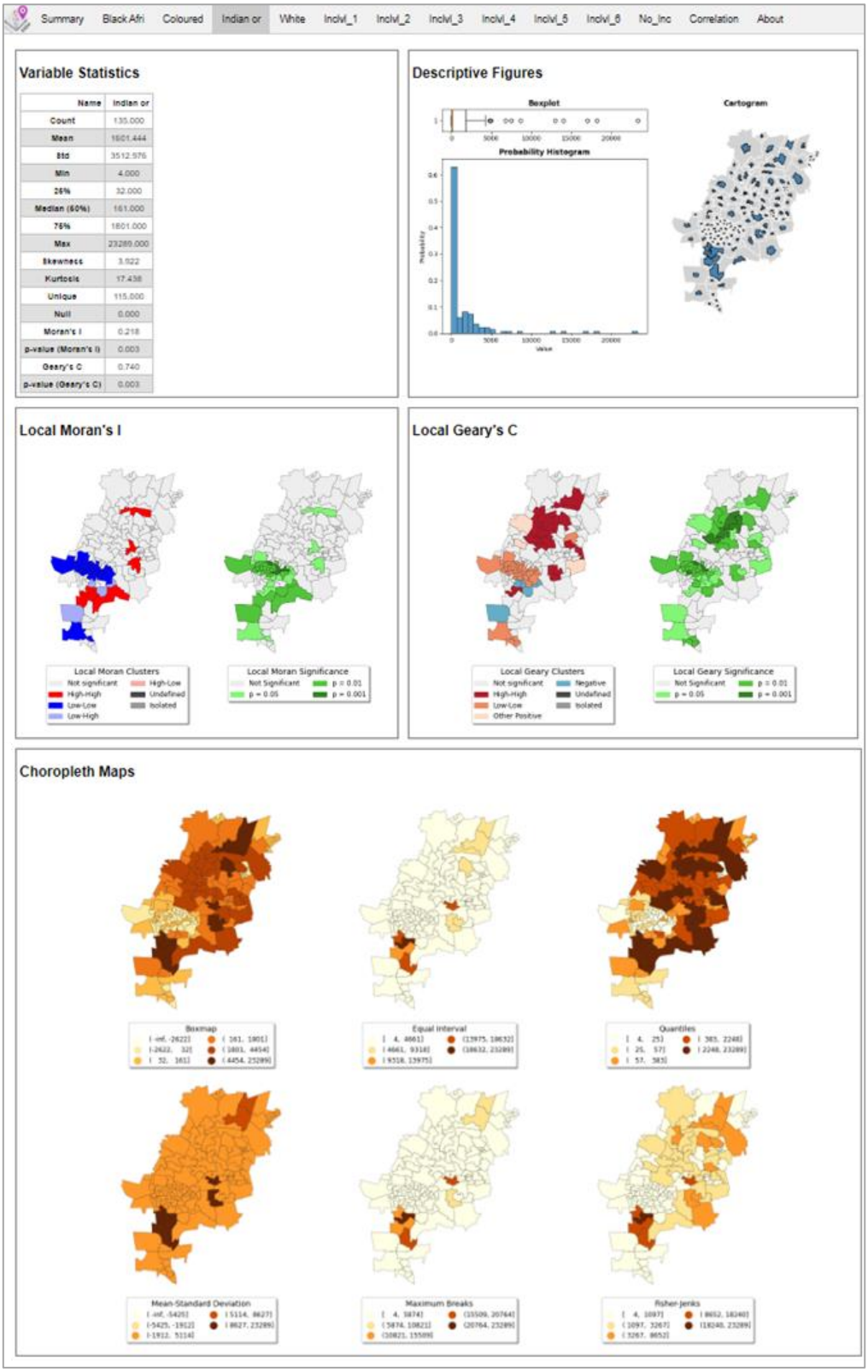


Figure 23: Updated Variable Information Page for autoESDA vector report

4.4.4 Correlation Page

Originally the correlation heatmap was constructed using Pearson correlation, however this is limited in that it is only suitable for linear variables (Devore and Berk, 2012). Some of the interview participants suggested that the user should be allowed to specify their preferred correlation type should they desire. The decision was made to include a heatmap matrix for Pearson, Kendall, and Spearman correlation. This is the same approach used in the pandas-profiling library and does not affect performance as the calculation is not computationally intensive. This leaves the user to interpret the results as they see fit. All values for the correlation heatmaps are now rounded off to two decimal places to avoid the values extending beyond their respective cells. The colour bar has also been included as per numerous suggestions; this allows the user to more easily comprehend the scale of the various colours used.

The pairwise plot had been updated to include a regression line. The use of a regression line is only suitable for linear relationships; however the responsibility has once again been left to the user to interpret it as they see fit.

The addition of extra correlation heatmaps have the added benefit of improving the layout of the Correlation Page. The heatmaps are stacked under each other to the left of the page with the pairwise plot to their right. The figures will all rescale according to the size of the page and, as a result, the viewing experience will have slightly improved from what it was previously, as less scrolling is required. The updated Correlation Page is shown in Figure 24.

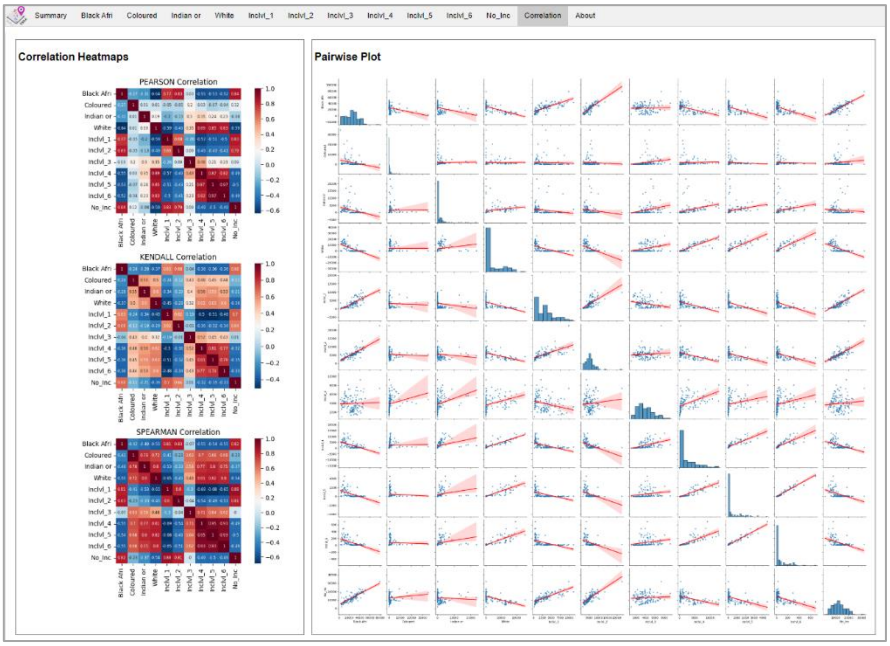


Figure 24: Updated Correlation Page for the autoESDA vector report

4.5 Limitations and Future Developments

The updates to autoESDA architecture and output reports are considered to be significant improvements over earlier versions. There are, however, further improvements that could be made to either improve the utility of the library, or to address current limitations.

File formats

Currently the raster module is only functional with GeoTIFF files. A user will be able to specify another file format (such as PNG, JPEG, or netCDF), however it has not been extensively tested to see how autoESDA will handle these. The structure of a GeoTIFF file ensures that arrays are referred to as bands, and as such, when opened with rioxarray, the terms band, and *x* and *y* coordinates are used. This is, however, different for other formats such as netCDF files where bands are named rather by the attribute they represent, and *lat* and *long* are used as names for the values that represent cell coordinates. Once the input of the raster dataset is addressed, the rest of the library should be capable of handling the data from other file formats.

Spatial weights

autoESDA currently uses queen's case contiguity weights with an order of one by default. Unfortunately, this is a limitation as the user has no way of using a different type of weights matrix. It could be beneficial in the future to include functionality for the user to specify the type of weights matrix (contiguity/distance-based/knn) and to specify the associated parameters (shape/order/distance/number of neighbours). The functionality could even be expanded to allow the user to specify their own, previously created weights file as a parameter.

Additionally, the output report could display the number of islands, as is done when creating the weights matrix in GeoDa. This should be a straightforward task as a warning message is already displayed in the command line when neighbour-less observations (islands) are detected. Other informative measures such as the minimum, maximum, and mean number of neighbours could also be displayed.

Report structure

The structure of the current HTML reports (vector and raster) are functional, however there is still excessive whitespace, and the design often requires the user to scroll more than what should be necessary. Currently the same report structure is used each time the report is generated, and it could be beneficial for the user to customise which elements are included in the report. This could include specifying what data to include in the report, which would mean

that time is not spent processing variables or bands that are of no interest to the user. The same could be true for the various plots, allowing the user to specify when they may only be interested in generating a Correlation Page, or seeing only the choropleth maps and histograms – speeding up processing time as intensive LISA calculations may not be required. This same philosophy is behind the design of DataPrep.EDA where Peng et al. (2021) allowed the user to run certain ad-hoc EDA tasks, rather than a colossal report. This allowed for an *interactive processing speed* that is dependent on the required task.

Improved error handling

The updates to autoESDA has begun incorporating some error handling, however there is still a large room for improvement as it is not uncommon to come across an unexpected error. Currently, if an error is encountered in the generation of a plot, the error message is displayed in the area where the plot would be. This allows the report generation to continue even if an error is encountered, however, this solution could be improved. The displayed error message is a direct copy of the exemption thrown from the source code. This means that the user may not easily understand what has caused the error– especially if they have no programming experience.

Another problem is that there also seem to be frequent geometry errors that yield exceptions. If one tries to generate a vector report with a GeoDataFrame that has multipolygons, the following error message appears:

```
TypeError: 'MultiPolygon' object is not iterable
```

If the multipolygon features are simplified to form multiple individual polygons, the following error message appears instead:

```
TypeError: 'MultiLineString' object is not iterable
```

This seems to be a problem that is dependent on which version of shapely is installed, however more research into the problem needs to take place before a suitable solution can be developed.

Additional EDA and ESDA functionality

The functions included in autoESDA are not meant to be an exhaustive list of all the tools available under the EDA/ESDA umbrella. There are endless possibilities regarding what functionality could be incorporated into the output report. Increased functionality could greatly increase the utility of the library; however it comes at the cost of increased complexity and processing time.

Interactive Reports

The ability to interact with ESDA outputs is extremely effective, especially when working with large datasets (Anselin, 1996). The autoESDA report is static, meaning that no animations or brushing have been incorporated. Brushing, interactive maps, popups and other features could be incorporated into the report; however this would increase the file size of the report. Larger file sizes mean that the reports cannot easily be shared with other parties.

4.6 Conclusion

This chapter has documented the design and implementation of numerous improvements to the autoESDA library. Four high-level requirements were identified, namely: raster functionality, updated architecture, various minor improvements, and performance. The first three of these requirements have been addressed through the discussions within this chapter. In the next chapter, the final requirement – performance, will be addressed. This is done by testing how autoESDA handles vector and rater datasets that vary in their size and complexity.

CHAPTER 5: PERFORMANCE EVALUATION

5.1 Chapter Overview

This chapter aims to evaluate the performance of the updated autoESDA library in how it processes datasets of varying complexity and size. This addresses the final high-level requirement defined at the beginning of Chapter 4. The measure of performance used will be the time taken to generate the autoESDA report for each dataset. The results and their implications will be discussed. The output reports can be viewed online, their links are provided in Appendix E.

5.2 Method

All experiments were conducted on a desktop computer that has a 64-bit operating system with Windows 10 Enterprise installed. It has an Intel Xeon CPU E3-1270 v6 processor runs with a clock speed of 3.80 GHz. There is also 64 GB of RAM installed as well as 32 GB graphics card.

The code was run in an Anaconda environment with Python 3.9 installed, including any dependencies required. All data was saved onto the local SSD storage to ensure efficient data retrieval and writing.

For both the vector and raster tests, the respective `generate_report` function was called from the controller of each module. The appropriate test dataset was given as a parameter, and the report name was specified. The report name was made up from the dataset, the version, and the number of the run simulation (e.g. Dataset1_LowRange_1).

The code for each test simulation was run at least three times and the average computation time for the runs is used in the results section for this comparison. If the values in the first three runs differed significantly then the process was rerun until three sets of times that were similar to each other were achieved. These were then used to calculate an average and the *irregular* results were discarded. To allow for consistent results, the computer was not used for anything else while the code was running.

5.3 Description of Test Datasets

A variety of raster and vector datasets were sourced to evaluate the performance of the updated autoESDA library. The remainder of this section will describe these datasets qualitatively and statistically. The source of these datasets, and any license or restrictions that they may carry, will also be highlighted. The principle on which the autoESDA library is built is

that it is *data agnostic,* and that the same workflow is applied to each dataset regardless of the thematic nature of the dataset. The assumption is that the user will interpret the report to ensure that the results are understood correctly. The thematic nature of a dataset should also not have any effect on the library performance. As such, the thematic nature of the test datasets are described in this section, however the purpose of this is only to be of interest to the reader. Any interpretation of the output autoESDA reports is beyond the scope of this project.

5.3.1 Vector Datasets

These datasets vary in file size, numbers of features and number of attributes. The library only processes numeric attributes, so it is expected that only their values will impact the processing time. The theoretical numeric values are calculated as the product of the number of features and numeric attributes. As there may be some missing values in the datasets, the actual number of values may differ from the theorical one.

The complexity of the polygons in the lattice of the different datasets may have an impact on the computation time. For this reason, a queen's first order spatial weights matrix has been created and the number of geometries, vertices, and average number of neighbours has been recorded. This impact of geometry complexity is not expected to be significant, however, as the average time taken to generate the spatial weights matrices for the tests described in Section 3.3 were found to be negligible when compared to the LISA calculations. The test datasets are qualitatively described below, and their statistics are compared in Table 12.

Dataset 1 – AirBnB Chicago 2015

This dataset has been sourced from the GeoDa data portal[43, 44], although it is curated from data that has been sourced from AirBnB and the Chicago data portal. The dataset is made up of 77 community area polygons, and 20 attributes ranging from AirBnB response/acceptance rates to other crime, population, and socioeconomic data. This dataset is available under the Creative Commons 4.0 license.

Dataset 2 – Grid 100

Sourced from the GeoDa data portal, this simulated dataset is made up of a 10 by 10 grid of cells, making 100 polygons. There are 36 attributes which are all randomly simulated,

[43] https://geodacenter.github.io/data-and-lab/

[44] The limitation of using this portal is that data may have been cleaned to make it easier for students to work with.

autocorrelated observations. This dataset has been created purely for testing purposes and is not based on anything in reality. It is available for download "*as is*".

Dataset 3 – South African 2011 Census

This dataset is the result of combining various demographic metrics from the 2011 census conducted by Statistics South Africa[45], with local municipal boundaries sourced from the Municipal Demarcation Board[46]. The demographic data represents population groups, age, gender, and primary languages – this makes up 33 total attributes for each of the 234 local municipal polygons.

Dataset 4 – Natural Earth Country Boundaries

Obtained directly from the Natural Earth Website[47], this dataset represents the 1:110m country boundaries. There are 177 polygons and 167 attributes, however most of the attributes are text labels, meaning there are only 31 numeric attributes. The numeric attributes are not all statistics or measurements, some are simply zoom levels or label sizes, which may not be of any relevance for this study. This dataset is available with no license or restrictions.

Dataset 5 – Malaria in Colombia

This dataset, also downloaded from the GeoDa data portal, represents malaria incidence in Colombia in 1998, as well as various census figures and annual population projections. Each of the 50 attributes are observed in each of the 1 068 municipal polygons that cover the entire administrative area of Colombia. This dataset is available "*as is*".

Dataset 6 – USA Election Results

The final dataset has been downloaded from the GeoDa data portal. It is composed of 2012 and 2016 election results, along with other socioeconomic and demographic data for 3 108 county polygons in the United States of America. There are 74 attributes, of which 68 are numeric.

[45] https://superweb.statssa.gov.za/webapi/jsf/login.xhtml
[46] https://www.demarcation.org.za/
[47] https://www.naturalearthdata.com/downloads/110m-cultural-vectors/

Table 12: Comparison of vector test datasets

Dataset	1	2	3	4	5	6
File size (MB)	0.57	0.06	18.28	0.70	1.99	19.8
Features (Polygons)	77	100	234	177	1 068	3 108
Attributes	20	36	33	167	50	74
Numeric attributes	17	35	22	31	45	68
Numeric values	1 309	3 500	5 148	5 487	48 060	211 344
No. geometries	79	100	244	288	1 118	3 946
No. vertices	52 641	500	1 103 235	10 654	87 955	885 633
Mean no. neighbours	5.12	6.84	5.32	3.55	5.99	5.84

5.3.2 Raster Datasets

Three single band raster datasets were used to evaluate the performance of the raster module of autoESDA and how it is impacted by the size and complexity of the dataset. autoESDA only works with the valid cells (i.e. cells with values other than the defined NoData values) of a raster grid which makes the number of valid cells the logical statistic to quantify the size of a dataset. Each of the test datasets have a `STATISTICS_VALID_PERCENT` attribute in their metadata which represents the proportion of valid cells in the grid. This attribute could be used to calculate the total number of valid cells for each test dataset. The dataset will still be processed, regardless of the number of NoData values, however these values will just be ignored. Each of the datasets are briefly described below, and their numeric summaries can be viewed in Table 13.

Dataset 1 – Global Terrestrial Precipitation

This raster surface represents the total terrestrial precipitation in 2017. This is measured in millimetres and values range from 0 to 10 765.90. Data is only present for terrestrial areas and as such, cells over the ocean are assigned a NoData value of -9 999. The dataset has been sourced from a GitHub repository[48] under an open data license for educational and non-commercial use. While this dataset has the largest extent (global), it also has the largest cell resolution (0.5 degrees). The 720 by 347 grid has a valid cell percentage of 34.34%, meaning

[48]Obtained from https://github.com/andrea-ballatore/open-geo-data-education

that there are approximately 85 796 valid cells to be analysed. This makes it the smallest raster test dataset.

Dataset 2 – EU NOx concentration

This dataset represents the average concentration of Nitrate and Nitrite (NOx) in the atmosphere for 2016. It covers the European region with 2 577 rows and 1 058 columns with cells of 3km resolution. This dataset has a valid cell percentage of 17.91%, meaning that there are approximately 488 311 valid cells to be analysed. The non-valid cells represent areas over large waterbodies and have been assigned a NoData value of -9 999. The dataset represents percentage of NOx concentration in the atmosphere, and its values range from 0.05% to 56.90%. This dataset has been sourced from the same repository as Dataset 1 and thus also available under an open data license which allows for educational and non-commercial use.

Dataset 3 – South African Population

Dataset 3 is obtained from WorldPop[49] and is available under the Creative Commons license. It represents the unconstrained population count on 1km cells across South Africa in 2020. The cells are arranged into a 2 176 by 1 591 grid, and with a valid percentage of 46.72%, this makes it the largest test dataset with approximately 1 617 454 valid cells. This dataset has a NoData value of -99 999 which is used for cells that do not fall within the borders of South Africa.

In addition to size, the complexity (variance) of a dataset may also have an impact on the time required for it to be processed. This is the same assumption as used in Section 4.2.2 where the performance of different strategies for calculating LISA on raster datasets was evaluated in order to identify the most optimal strategy. In that case, noise was added to the test datasets at a global level by multiplying the variance by different magnitudes to create datasets with low, medium, and high variances. A similar philosophy is used to test the autoESDA library, however instead of inflating the variance globally, spatially autocorrelated noise was added. This resulted in the presence of hotspots and coldspots in the simulated test datasets – which provide varying levels of complexity for testing the library.

[49]Obtained from https://hub.worldpop.org/geodata/summary?id=33892

Table 13: Comparison of original raster test datasets

	Dataset 1 (Global Precipitation)	Dataset 2 (EU NOx Concentration)	Dataset 3 (RSA Population)
Dataset Units	Millimetres (mm)	Concentration (%)	Count
Dimension	720 x 347	2 577 x 1 058	2 176 x 1 591
Valid Percentage	34.34	17.91	46.72
Total Valid Cells	≈ 85 796	≈ 488 311	≈ 1 617 454
Cell Resolution	0.5 degrees	3km	1km
DataType	Float32	Float32	Float32
NoData Value	-9 999	-9 999	-99 999
Minimum	0	0.05	0
Maximum	10 765.90	56.90	15 675.81
Mean	537.11	7.60	37.67
Standard Deviation	640.84	5.22	342.29

In order to incorporate random spatial clusters in the test datasets, 50 points were randomly generated within the area of valid cells of each dataset. A random number in the original pixel value range of the test dataset was assigned to each point. Three experimental variograms were plotted by visually fitting a model with user defined values for the nugget, sill, and range. The nugget and range values were kept constant for the three variograms, while the range was adjusted to ensure variograms with a low, medium, and high range. The R script used to carry out these steps can be viewed in Appendix D. The defined values and the experimental variograms for each dataset can be viewed in Table 14 below.

Table 14: Values used in the three Kriging models (low, medium, and high range) for each dataset.

	Dataset 1	Dataset 2	Dataset 3
Nugget	0	0	0
Sill	10 000 000	600	20 000 000
Low Range	300	50	10
Medium Range	2 000	200	100
High Range	5 000	600	300
Model	Spherical	Spherical	Spherical

The low, medium, and high ranged models were then used in a separate Kriging function to simulate a surface of random low, medium, and high spatial autocorrelation. A mask was used to ensure that the simulated surfaces matched the shape of the valid cells in the original

dataset. The simulated surfaces were then each added to the original dataset to create three additional versions of each dataset. This means that each dataset size has four versions: the original raster, original + low variance (low noise), original + medium variance (medium noise), and original + high variance (high noise). Additionally, these four versions will be stacked to form one raster with four bands.

Table 15 summarises the descriptive statistics for the surfaces created for the raster datasets as well as their simulated surfaces of low, medium, and high noise.

Table 15: Descriptive statistics for each dataset and their simulated surfaces

	Dataset 1			
	Band 1	**Band 2**	**Band 3**	**Band 4**
Description	Original	Low Noise	Medium Noise	High Noise
Minimum	0	230.00	199.17	198.75
Maximum	10 765.90	16 462.48	16 558.76	16 206.87
Mean	537.11	6 165.61	6 169.76	6 300.57
Standard Deviation	640.84	921.56	1 967.75	2 384.88
	Dataset 2			
	Band 1	**Band 2**	**Band 3**	**Band 4**
Description	Original	Low Noise	Medium Noise	High Noise
Minimum	0.05	24.64	16.44	9.20
Maximum	56.90	97.44	106.74	110.27
Mean	7.60	48.20	47.80	47.60
Standard Deviation	5.22	5.36	7.52	14.28
	Dataset 3			
	Band 1	**Band 2**	**Band 3**	**Band 4**
Description	Original	Low Noise	Medium Noise	High Noise
Minimum	0	999.22	189.50	131.03
Maximum	15 675.81	22 468.05	22 339.49	25 308.72
Mean	37.67	6 826.31	6 881.69	6 803.96
Standard Deviation	342.29	385.97	1477.10	2887.50

5.4 Results and Discussion

5.4.1 Vector Module

The average time taken to generate the autoESDA vector report for each dataset has been recorded in Table 16. The general trend was that as the dataset number increased, so did the mean processing time. The processing time has been divided amongst the features, numeric attributes, and numeric values in an effort to standardise the results to aid comparison across datasets. The output report size was also recorded, as the complexity of the plots may have an impact on the processing time.

Table 16: Results for autoESDA vector report generation

Dataset	1	2	3	4	5	6
Mean processing time (min)	2.96	14.33	13.74	26.33	35.53	256.73
Time / feature (s)	2.31	8.60	3.52	8.93	2.00	4.96
Time / numeric attribute (s)	10.45	24.56	37.48	50.96	47.38	226.52
Time / numeric value (s)	0.14	0.25	0.16	0.29	0.04	0.07
Output report size (MB)	11.40	16.20	24.50	24.20	61.50	118.00

There is no obvious trend upon investigating the time spend per polygon feature in each dataset. Dataset 5 has the lowest value with 2s per feature, while Dataset 4 has the greatest value of 8.6s per feature. This is unexpected as Dataset 4 has the least neighbours on average (3.55), while Dataset 5 has the second-highest average number of neighbours (5.99).

The average processing time per numeric attribute in general increased as the dataset number increased, with the exception of Dataset 5. It was expected that the time required to create the autoESDA vector report would be largely dependent on the number of theoretical numerical values. This expected trend holds true for the most part, but with the exception of Dataset 2 (3 500 theoretical numerical values), which has a mean processing time of 14.33 minutes which greater than that of Dataset 3 (5 148 theoretical numerical values), which is 13.74 minutes. Dataset 2 did have a greater average number of neighbours (6.84) when compared to that of Dataset 3 (5.32). This could explain the greater processing time as a more neighbours meant that more values needed to be considered in the LISA calculations. Interestingly, the larger datasets (Dataset 5 and 6) have the lowest time spent per numeric value, even though their total processing times were still the longest of the six datasets.

Other factors that may have an impact on processing time could be the data types in the datasets. The numeric data type (int or float) could impact the processing time, as well as the precision (number of decimal places) of the value. Dataset 2's numeric attributes are all of the float data type with a precision of eight, whereas Dataset 3 only has integer data types with no decimal places. This could possibly explain why Dataset 3 had a shorter processing time, as irrespective of the greater number of numerical values, their complexity was less due to the lack of decimal points.

5.4.2 Raster Module

The timing results for the raster module performance testing are displayed in Table 17. The original file of each dataset was used as the input file, after which low, medium, and high noise was added. Each of these inputs was processed individually, and then they were stacked into one four-band stacked dataset which was then used to generate an autoESDA raster report.

Table 17: Average time (minutes) for autoESDA raster report generation

Values represent time in minutes	**Band 1**	**Band 2**	**Band 3**	**Band 4**	**Total**	**Stacked**
	Original	**+ Low Noise**	**+ Med Noise**	**+ High Noise**		
Dataset 1	3.91	3.50	3.60	3.65	14.66	15.33
Dataset 2	34.10	34.43	36.51	37.38	142.42	152.48
Dataset 3	344.02	370.14	282.74	274.30	1 271.19	Missing*

*The stacked bands for Dataset 3 were large enough to crash the computer each time the process was run, as a result there is no data available.

It was expected that the greater the variance (bands 1 – 4), the longer the processing time would be. This hypothesis however only holds true in the results for Dataset 2. Each of the bands with simulated variance (bands 2 – 4) for dataset took shorter to process than the original dataset. For Dataset 3, band 2 (low noise) took longer to process than the original dataset (band 1); however band 3 (medium noise) and band 4 (high noise) took less time to process than bands 1 and 2.

Although it was expected that the increase in noise would lead to an increase in the time taken to generate the report, this assumed that an increase in mathematical variance would increase the noise. The mathematical variance added was spatially autocorrelated, which could explain why an increase in processing time was not always the case. The spatially autocorrelated variance may have increased the mathematical noise of the dataset, while subsequently

making the raster surface more homogeneous with regards to the spatial distribution of its high and low values.

The *Total* column in Table 17 is the sum of the processing time for bands 1 to 4 when processed individually to generate four separate reports, while the *Stacked* column represents the time to generate a single report from the same four bands stacked into one file. It was expected that the sum of the individual times taken would be greater than the time taken to generate the stacked report, as the four individual reports would require the carrying out tasks four times whereas this would only be done once for the stacked report – such as the creation of an HTML file, or more significantly, the vectorisation of the dataset. This discrepancy would be interesting to investigate further, by recording the time for each component of the autoESDA report. This would allow one to identify where bottlenecks in the processing occurred and whether there were any potential ways to further optimise the process.

5.5 Discussion

The major limitation in the methodology used for this performance test of the updated autoESDA library, is that only a single metric – total processing time is used. While this is useful in comparing the overall process, it cannot be used to determine how the processing time is divided amongst the various components of the output report. This could be investigated further through code profiling, which analyses the time taken to execute each line of code. This had not been done as part of this research project due to time limitations, however, its results could assist in further optimising the autoESDA library, as time-consuming tasks could be re-evaluated, and more efficient approaches could be incorporated. Additionally, the effect of different file types and data types on processing time could add an interesting perspective, which was not adequately been addressed in this project.

The simulations executed for the purpose of understanding the performance of the updated autoESDA library do, however, prove that the ESDA workflow can be automated at least to some degree. It would be an interesting project to compare the processing times recorded here with the total time a user would require to generate the same results manually using other platforms, such as GeoDa or ArcGIS. The comparison between the timed manual process and the autoESDA report generation times would easily quantify whether autoESDA is a suitable alternative to the traditional, manual approach to ESDA.

CHAPTER 6: CONCLUSION

6.1 Chapter Overview

Chapter 6 concludes this dissertation by summarising the work that has been done and assessing the extent to which the aim and each of the objectives were achieved. Finally, future work with regards to the autoESDA library and automating ESDA as a whole is discussed.

6.2 Main Results

The aim of this research was to advance the automation of ESDA by implementing improvements to the autoESDA library and evaluating their performance. The aim is broken down into five objectives that are defined in Section 1.4. The main results from each of these objectives should lead to the attainment of the aim of this study.

Objective 1: Review related literature in conjunction with previously suggested improvements to the autoESDA Python library.

This objective was addressed in Chapter 2 where a variety of sources were consulted in the formation of the literature review. The concept of spatial and big spatial data was discussed, alongside the challenges that they bring about. EDA and ESDA were identified as potential solutions to some of these challenges, as they can summarise large datasets and guide further processing and analysis. One of the novel suggestions from the previously conducted interviews was the support for ESDA specifically LISA on raster datasets. This was discussed in Section 2.5. The literature review finished off with a summary of the ease with which some popular ESDA functions can be automated. This was guided by a discussion of various calls for new tools to deal with spatial data, as well as similar projects that have automated other workflows related to spatial data. In order to automate the ESDA process, the new tool would need to be largely automated, have error checking functionality, operate in real time, handle unexpected scenarios, and efficiently (and repeatedly) generate reliable results. Automating ESDA requires a fine balance between reducing human involvement for the purpose of eliminating human-induced errors, while still including human input that is fundamental to the exploratory, data-driven ESDA process.

Objective 2: Define requirements based on suggested improvements to the autoESDA Python library.

Section 3.4.3 documents the feedback from the previously conducted interviews. We acknowledge that thirteen participants constitute only a small sample size, however the purposes of the interviews were to solicit feedback and not to make empirical conclusions. We

would recommend future studies to incorporate a larger sample size to allow for more accurate inferences to be made. This formed a large component of the user stories which can be found in Appendix B. Four high-level requirements were identified: these were the need for raster functionality, an updated architecture of the library, some other minor improvements and finally, adequate performance. These four requirements are introduced in Section 4.1, and are discussed in greater detail in Sections 4.2, 4.3, 4.4, and Chapter 5 respectively.

Objective 3: Design and implement solutions that address the identified requirements.

Chapter 4 discusses the design and implementation of the improvements made in the second major iteration of the autoESDA library. First, the inclusion of raster functionality was described (Section 4.2). Different strategies for raster LISA were identified and compared and ultimately it was decided that the optimal strategy was to first vectorise the raster dataset and then to calculate the LISA using the pygeoda library. This strategy outperformed the other strategies, which is most likely due to its C++ implementation.

The inclusion of raster functionality necessitated an overhaul of the architectural design of the library so as to minimise code duplication, while allowing for improvements to be implemented with greater ease. This is discussed in Section 4.3. The solution to this is an MVC-type design, which splits the autoESDA library into two modules – raster and vector. Each module consists of a controller and a model. The controller contains the functions with which the user interacts, by calling different functions. The controller also makes numerous functions to the model and combines all the created components into an output HTML report. The model contains all the calculations and functions that are required in order to generate the results which are to be included in the output report.

In addition to the architectural and raster functionality improvements, various other minor improvements were made to improve the utility of the library. These improvements are largely cosmetic and are described in detail in Section 4.4.

All of the updates discussed were implemented with the performance of the library in mind and were designed to be as efficient as possible. Any decisions and limitations are discussed throughout Chapter 4.

Objective 4: Evaluate the autoESDA library in terms of the defined requirements.

The updated architecture described in Chapter 4 meant that the various minor improvements and raster functionality and other minor improvements could be added to the library with relative ease. This process would be a significantly more complicated and would require more

time to implement, should the architecture not have been updated. This is due to it more modular structure, brought about by the use of classes and an MVC-type design.

The raster functionality and other minor improvements were implemented in the most efficient way possible, given the scope of this project. Various datasets have been used to test these improvements and errors have been identified and corrected as part of the development stage. The output reports from each of the datasets used in the performance evaluation were also reviewed to identify potential shortfalls in the design of the library. These have been addressed and/or discussed in Chapter 4.

The evaluation of the performance of the autoESDA library is necessary as it gives a quantitative representation of how well the library achieves its goal. These results also set a benchmark, from which any further improvements could take place.

To evaluate the library's performance, numerous vector and raster datasets were sourced. The vector datasets had varying amounts of features, attributes (numeric and non-numeric), file sizes and geometries. Three different raster datasets of differing sizes were used, and varying degrees of spatially autocorrelated noise were added to each dataset. This allowed for the investigation of the effect of noise on the performance of the library. The generation time for an autoESDA report varies according to the size and complexity of the input dataset. This means that there is no easy way to determine whether autoESDA requires less time to output results than a traditional, manual approach. The output times recorded and discussed in Chapter 5 seem reasonable, however they have the potential to be optimised further.

Objective 5: Based on the results, draw conclusions regarding the success of autoESDA as a means of automating the ESDA workflow.

Although we now have a measure of the time required by the library to generate the autoESDA report, there is no comparative metric representing the time required to perform this process manually. This means that we cannot compare the results obtained in Chapter 5 to a traditional, manual ESDA workflow and comment on the time saved through the use of autoESDA. As a result, the relative effectiveness of autoESDA with regards to time saved still remains unknown. While the time taken to generate the report may be significant (depending on the dataset size), the automation means that the user can focus on other tasks which the report is generated.

Another advantage brought about by autoESDA, is that it has removed the need for constant human input, which thereby eliminates the risk of human-induced errors. The user is required to specify only the input dataset, and autoESDA will output an HTML summary report. This

means that the user is not required to specify any additional parameters, and the dataset can be processed free from any bias or interactions.

6.3 Future Work

While autoESDA is a step in the right direction, these is still a lot more that could be done to enable a fully autonomous ESDA process. A challenge faced quite often when working with data is the need for data cleaning. This is an important part of the data lifecycle and can have a great effect on the results, however it is expected to take place before ESDA occurs.

At this stage, autoESDA still requires a human to interpret the report results and make any decisions regarding further use of the dataset. Another limitation of the autoESDA report is that it is not interactive, as is the case for other popular EDA/ESDA platforms. While this brings about some advantages, such as ease of distribution of the results, it does mean that the insights that can be extracted from the report are somewhat limited.

Although autoESDA is easy to use, it still requires the appropriate setup of a Python environment and the knowledge about how to perform basic tasks using Python and geopandas. This excludes a significant number of potential users who may lack these skills. A potential solution to this would be the development of a QGIS plugin, for which a prototype has already been designed; however this proof-of-concept still requires some work before it could be released. This development would provide a graphical interface as part of a popular GIS, making the autoESDA library accessible to a much larger number of people.

The performance testing of the autoESDA library has found that the processing time is largely dependent on the size, and to an extent, the complexity of the input datasets. These trends could be further investigated by using a greater number of test datasets, and varying the filetype, test computers, datatypes, and any other parameters to observe their effect on the library's performance.

The report generation times should also be compared to the time required to carry out the same process manually, which would allow one to draw conclusions about its role in saving time in the greater data lifecycle. Currently, the only time benefit demonstrated is that the automated process does not require user supervision once running, allowing the user to focus on other, more important tasks. The use of code profiling would also enhance one's understanding of the library. This would allow one to see the time spent on each line of code in the library, which would aid in the identification of bottlenecks and/or other time consuming functions. These could then be addressed in further developments that could further optimise the performance of the autoESDA library.

The possibilities surrounding autoESDA and the automation of the ESDA process are endless, and, as with most software lifecycles, there is always more that can be done. The second iteration of autoESDA has brought about important improvements that benefit the library and aid in the understanding of automation of the ESDA process as a whole. It would be interesting to revisit this project in the near future once the available tools and understandings have matured, in order to identify further potential improvements or dependencies that could further optimise the automation of ESDA.

REFERENCES

Amgalan, A., Mujica-Parodi, L., Skiena, S., 2022. Fast spatial autocorrelation. Knowledge and Information Systems 64, 919–941. https://doi.org/10.1007/s10115-021-01640-x

Anselin, L., 2019. A local indicator of multivariate spatial association: Extending Geary's C. Geographical Analysis 51, 133–150. https://doi.org/10.1111/gean.12164

Anselin, L., 2012. From SpaceStat to cyberGIS: Twenty years of spatial data analysis software. International Regional Science Review 35, 131–157. https://doi.org/10.1177/0160017612438615

Anselin, L., 2010. Thirty years of spatial econometrics. Papers in Regional Science 89, 3–25. https://doi.org/10.1111/j.1435-5957.2010.00279.x

Anselin, L., 2000. Computing environments for spatial data analysis. Journal of Geographical Systems 2, 201–220. https://doi.org/10.1007/PL00011455

Anselin, L., 1999. The future of spatial analysis in the social sciences. Geographic Information Sciences 5, 67–76. https://doi.org/10.1080/10824009909480516

Anselin, L., 1998. Exploratory spatial data analysis in a geocomputational environment, in: Geocomputation: A Primer. Wiley, Chichester, England, pp. 77–94.

Anselin, L., 1996. Interactive techniques and exploratory spatial data analysis. Regional Research Institute Working Papers.

Anselin, L., 1995. Local indicators of spatial association - LISA. Geographical Analysis 27, 93–115. https://doi.org/10.1111/j.1538-4632.1995.tb00338.x

Anselin, L., 1989. What is special about spatial data? Alternative perspectives on spatial data analysis, in: NCGIA Technical Reports. Presented at the UC Santa Barbara: National Center for Geographic Information and Analysis, Syracuse University. https://escholarship.org/uc/item/3ph5k0d4

Anselin, L., 1988. Spatial Econometrics: Methods and Models, Studies in Operational Regional Science. Springer Netherlands, Dordrecht. https://doi.org/10.1007/978-94-015-7799-1

Anselin, L., Bao, S., 1997. Exploratory spatial data analysis linking SpaceStat and ArcView, in: Fischer, M., Getis, A. (Eds.), Recent Developments in Spatial Analysis, Advances in Spatial Science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 35–59. https://doi.org/10.1007/978-3-662-03499-6_3

Anselin, L., Getis, A., 1992. Spatial statistical analysis and geographic information systems. Annals of Regional Science 26, 19. https://doi.org/10.1007/BF01581478

Anselin, L., Kim, Y., Syabri, I., 2004. Web-based analytical tools for the exploration of spatial data. Journal of Geographical Systems 6, 197–218. https://doi.org/10.1007/s10109-004-0132-5

Anselin, L., Li, X., Koschinsky, J., 2022. GeoDa, from the desktop to an ecosystem for exploring spatial data. Geographical Analysis 54, 439–466. https://doi.org/10.1111/gean.12311

Anselin, L., Rey, S., 2022. Open source software for spatial data science. Geographical Analysis 54, 429–438. https://doi.org/10.1111/gean.12339

Anselin, L., Rey, S., 2014. Modern spatial econometrics in practice: A guide to GeoDa, GeoDaSpace and PySAL. GeoDa Press LLC.

Anselin, L., Rey, S., 2012. Spatial econometrics in an age of cyberGIScience. International Journal of Geographical Information Science 26, 2211–2226. https://doi.org/10.1080/13658816.2012.664276

Anselin, L., Sridharan, S., Gholston, S., 2007. Using exploratory spatial data analysis to leverage social indicator databases: the discovery of interesting patterns. Social Indicators Research 82, 287–309. https://doi.org/10.1007/s11205-006-9034-x

Anselin, L., Syabri, I., Kho, Y., 2006. GeoDa: An introduction to spatial data analysis. Geographical Analysis 38, 5–22. https://doi.org/10.1111/j.0016-7363.2005.00671.x

Armstrong, M., Wang, S., Zhang, Z., 2019. The internet of things and fast data streams: prospects for geospatial data science in emerging information ecosystems. Cartography and Geographic Information Science 46, 39–56. https://doi.org/10.1080/15230406.2018.1503973

Batcheller, J., 2008. Automating geospatial metadata generation – an integrated data management and documentation approach. Computers & Geosciences 34, 387–398. https://doi.org/10.1016/j.cageo.2007.04.001

Bivand, R., 2006. Implementing spatial data analysis software tools in R. Geographical Analysis 38, 23–40. https://doi.org/10.1111/j.0016-7363.2005.00672.x

Bivand, R., 2002. Spatial econometrics functions in R: classes and methods. Journal of Geographical Systems 4, 405–421. https://doi.org/10.1007/s101090300096

Bivand, R., 1998. Software and software design issues in the exploration of local dependence. Journal of the Royal Statistical Society. Series D (The Statistician) 47, 499–508.

Borlongan, N., Cruz, R., Olfindo, N., Perez, A., 2016. Automation of lidar-based hydrologic feature extraction workflows using GIS, in: Earth Resources and Environmental Remote Sensing/GIS Applications VII. Presented at the Earth Resources and Environmental Remote Sensing/GIS Applications VII, SPIE, pp. 203–218. https://doi.org/10.1117/12.2241972

Ciceli, T., 2015. Stakeholder profile in spatial data lifecycle defined from SDI perspective. Presented at the State Geodetic Administration, State Geodetic Administration, Zagreb. https://doi.org/DOI: 10.13140/RG.2.1.1529.2648

Coetzee, S., Rautenbach, V., 2017. A design pattern approach to cartography with big geospatial data. Cartographic Journal 54, 301–312. https://doi.org/10.1080/00087041.2017.1400199

Cressie, N., 1993. Statistics for Spatial Data, Revised edition. ed. John Wiley & Sons, Inc, Hoboken, NJ.

Cura, R., 2019. Enriching exploratory spatial data analysis with modern computer tools, in: European Colloquium of Theoretical and Quantitative Geography - ECTQG 2019. Mondorf-les-Bains, Luxembourg.

Dall'erba, S., 2009. Exploratory spatial data analysis, in: Kitchin, R., Thrift, N. (Eds.), International Encyclopedia of Human Geography. Elsevier, Oxford, pp. 683–690. https://doi.org/10.1016/B978-008044910-4.00433-8

Dangermond, J., Goodchild, M., 2020. Building geospatial infrastructure. Geo-spatial Information Science 23, 1–9. https://doi.org/10.1080/10095020.2019.1698274

De Smith, M., Goodchild, M., Longley, P., 2018. Geospatial analysis: A Comprehensive Guide to Principles, Techniques And Software Tools, 6th ed. Winchelsea Press.

Devore, J., Berk, K., 2012. Modern Mathematical Statistics with Applications, Springer Texts in Statistics. Springer New York, New York, NY. https://doi.org/10.1007/978-1-4614-0391-3

Fotheringham, A., 1992. Exploratory spatial data analysis and GIS. Environment and Planning A 24, 1675–1678.

Gandomi, A., Haider, M., 2015. Beyond the hype: big data concepts, methods, and analytics. International Journal of Information Management 35, 137–144. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

Geary, R., 1954. The contiguity ratio and statistical mapping. The Incorporated Statistician 5, 115. https://doi.org/10.2307/2986645

Goodchild, M., 2000. Spatial analysis: methods and problems in land use management, in: Spatial Information for Land Use Management. Gordon and Breach Science Publishers, Amsterdam.

Goodchild, M., 1997. Geographic information systems, in: Ten Geographic Ideas That Changed the World. Rutgers University Press, New Brunswick, NJ, pp. 60–86.

Goodchild, M., 1992. Analysis, in: Geography's Inner Worlds: Pervasive Themes in Contemporary American Geography, Occasional Publications of the Association of American Geographers. Rutgers University Press, New York, pp. 138–162.

Goodchild, M., Haining, R., Wise, S., 1992. Integrating GIS and spatial data analysis: problems and possibilities. International Journal of Geographical Information Systems 6, 407–423. https://doi.org/10.1080/02693799208901923

Goodchild, M., Longley, P., 2013. Geocomputation and GIScience, in: Handbook of Regional Science. Springer, Dordrecht.

Haslett, J., Wills, G., Unwin, A., 1990. SPIDER – an interactive statistical tool for the analysis of spatially distributed data. International Journal of Geographical Information Systems 4, 285–296. https://doi.org/10.1080/02693799008941547

Higgins, C., Ray, S., 2022. Fast exploratory analysis with spatio-temporal aggregation over polygonal regions, in: 2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT). Presented at the 2022 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT), IEEE, Vancouver, WA, USA, pp. 30–39. https://doi.org/10.1109/BDCAT56447.2022.00012

Jackson, M.C., Huang, L., Xie, Q., Tiwari, R.C., 2010. A modified version of Moran's I. International Journal of Health Geographics 9, 33. https://doi.org/10.1186/1476-072X-9-33

Jern, M., Åström, T., Johansson, S., 2008. GeoAnalytics tools applied to large geospatial datasets, in: 2008 12th International Conference Information Visualisation. Presented at the 2008 12th International Conference Information Visualisation, pp. 362–372. https://doi.org/10.1109/IV.2008.27

Jern, M., Johansson, S., Johansson, J., Franzen, J., 2007. The GAV toolkit for multiple linked views. Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization (CMV 2007).

Jin, X., Wah, B., Cheng, X., Wang, Y., 2015. Significance and challenges of big data research. Big Data Research, Visions on Big Data 2, 59–64. https://doi.org/10.1016/j.bdr.2015.01.006

Kaluzny, S., Vega, S., Cardoso, T., Shelly, A., 1998. Analyzing lattice data, in: Kaluzny, S., Vega, S., Cardoso, T., Shelly, A. (Eds.), S+SpatialStats: User's Manual for Windows® and UNIX®. Springer, New York, NY, pp. 110–145. https://doi.org/10.1007/978-1-4615-7826-0_5

Lee, J., Kang, M., 2015. Geospatial big data: challenges and opportunities. Big Data Research 2, 74–81. https://doi.org/10.1016/j.bdr.2015.01.003

Li, J., He, J., Liu, Y., Wang, D., Rafay, L., Chen, C., Hong, T., Fan, H., Lin, Y., 2019. Spatial autocorrelation analysis of multi-scale damaged vegetation in the Wenchuan earthquake-affected area, southwest China. Forests 10, 195. https://doi.org/10.3390/f10020195

Li, S., Dragicevic, S., Castro, F., Sester, M., Winter, S., Coltekin, A., Pettit, C., Jiang, B., Haworth, J., Stein, A., Cheng, T., 2016. Geospatial big data handling theory and methods: a review and research challenges. ISPRS Journal of Photogrammetry and Remote Sensing, Theme issue "State-of-the-art in photogrammetry, remote sensing and spatial information science" 115, 119–133. https://doi.org/10.1016/j.isprsjprs.2015.10.012

Li, X., Anselin, L., Koschinsky, J., 2015. GeoDa web: enhancing web-based mapping with spatial analytics, in: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL '15. Association for Computing Machinery, New York, NY, USA, pp. 1–4. https://doi.org/10.1145/2820783.2820792

Longley, P., Goodchild, M., Maguire, D., Rhind, D., 2015. Geographic Information Science & Systems, Fourth edition. ed. Wiley, Hoboken, NJ.

Mennis, J., Guo, D., 2009. Spatial data mining and geographic knowledge discovery – an introduction. Computers, Environment and Urban Systems 33, 403–408. https://doi.org/10.1016/j.compenvurbsys.2009.11.001

Moncrieff, S., Turdukulov, U., Gulland, E., 2016. Integrating geo web services for a user driven exploratory analysis. ISPRS Journal of Photogrammetry and Remote Sensing 114, 294–305. https://doi.org/10.1016/j.isprsjprs.2016.01.015

Moura, A., Fonseca, B., 2020. ESDA (exploratory spatial data analysis) of vegetation cover in urban areas—recognition of vulnerabilities for the management of resources in urban green infrastructure. Sustainability 12, 1933. https://doi.org/10.3390/su12051933

Morgenthaler, S., 2009. Exploratory data analysis. WIREs Computational Statistics 1, 33–44. https://doi.org/10.1002/wics.2

Murray, A., Estivill-Castro, V., 1998. Cluster discovery techniques for exploratory spatial data analysis. International Journal of Geographical Information Science 12, 431–443. https://doi.org/10.1080/136588198241734

Murtiyoso, A., Veriandi, M., Suwardhi, D., Soeksmantono, B., Harto, A., 2020. Automatic workflow for roof extraction and generation of 3d CITYGML models from low-cost UAV image-derived point clouds. ISPRS International Journal of Geo-Information 9, 743. https://doi.org/10.3390/ijgi9120743

Myatt, G., Johnson, W., 2014. Making Sense of Data I: A Practical Guide to Exploratory Data Analysis and Data Mining, Second edition. ed. John Wiley & Sons, Inc, Hoboken, New Jersey.

Myint, S., Wentz, E., Purkis, S., 2007. Employing spatial metrics in urban land-use/land-cover mapping: comparing the Getis and Geary indices. Photogrammetric Engineering and Remote Sensing 73, 1403–1415. https://doi.org/10.14358/PERS.73.12.1403

Openshaw, S., 1995. Developing automated and smart spatial pattern exploration tools for geographical information systems applications. Journal of the Royal Statistical Society. Series D (The Statistician) 44, 3–16. https://doi.org/10.2307/2348611

Paudel, A., Puri, S., 2022. Accelerating spatial autocorrelation computation with parallelization, vectorization and memory access optimization: with a focus on rapid recalculation of covid related spatial statistics for faster geospatial analysis and response, in: 2022 22nd IEEE International Symposium on Cluster, Cloud and Internet Computing (CCGrid). Presented at the 2022 22nd International Symposium on Cluster, Cloud and Internet Computing (CCGrid), IEEE, Taormina, Italy, pp. 544–554. https://doi.org/10.1109/CCGrid54584.2022.00064

Peffers, K., Tuunanen, T., Rothenberger, M., Chatterjee, S., 2007. A design science research methodology for information systems research. Journal of Management Information Systems 24, 45–77. https://doi.org/10.2753/MIS0742-1222240302

Peng, J., Wu, W., Lockhart, B., Bian, S., Yan, J., Xu, L., Chi, Z., Rzeszotarski, J., Wang, J., 2021. DataPrep.EDA: task-centric exploratory data analysis for statistical modelling in Python, in: Proceedings of the 2021 International Conference on Management of Data. Presented at the SIGMOD/PODS '21: International Conference on Management of Data, ACM, Virtual Event China, pp. 2271–2280. https://doi.org/10.1145/3448016.3457330

Pillay, L., Schaab, G., Coetzee, S., Rautenbach, V., 2019. A comprehensive workflow for automating thematic map geovisualization from univariate big geospatial point data. International Cartographic Association 2, 8. https://doi.org/10.5194/ica-proc-2-100-2019

Raju, P., Nathan, B., 2018. The data life cycle. Strategic Finance 100, 62–63.

Rey, S., Anselin, L., 2007. PySAL: a python library of spatial analytical methods. Review of Regional Studies 37, 5–27.

Rey, S., Anselin, L., Li, X., Pahle, R., Laura, J., Li, W., Koschinsky, J., 2015. Open geospatial analytics with PySAL. ISPRS International Journal of Geo-Information 4, 815–836. https://doi.org/10.3390/ijgi4020815

Rey, S., Arribas-Bel, D., Wolf, L.J., 2023. Geographic Data Science with Python.

Rey, S.J., Anselin, L., 2006. Recent advances in software for spatial analysis in the social sciences. Geographical Analysis 38, 1–4. https://doi.org/10.1111/j.0016-7363.2005.00670.x

Rey, S.J., Anselin, L., Amaral, P., Arribas-Bel, D., Cortes, R., Gaboardi, J., Kang, W., Knaap, E., Li, Z., Lumnitz, S., Oshan, T., Shao, H., Wolf, L., 2022. The PySAL ecosystem: philosophy and implementation. Geographical Analysis 54, 467–487. https://doi.org/10.1111/gean.12276

Rey, S.J., Janikas, M., 2006. STARS: Space–time analysis of regional systems. Geographical Analysis 38, 67–86. https://doi.org/10.1111/j.0016-7363.2005.00675.x

Roberts, J., 2005. Exploratory visualization with multiple linked views, in: Dykes, J., MacEachren, A., Kraak, M. (Eds.), Exploring Geovisualization, International Cartographic Association. Elsevier, Oxford, pp. 159–180. https://doi.org/10.1016/B978-008044531-1/50426-7

Robinson, A., Demšar, U., Moore, A., Buckley, A., Jiang, B., Field, K., Kraak, M., Camboim, S., Sluter, C., 2017. Geospatial big data and cartography: research challenges and opportunities for making maps that matter. International Journal of Cartography 3, 32–60. https://doi.org/10.1080/23729333.2016.1278151

Rogerson, P., Kedron, P., 2012. Optimal weights for focused tests of clustering using the local Moran statistic. Geographical Analysis 44, 121–133. https://doi.org/10.1111/j.1538-4632.2012.00840.x

Sapre, A., Vartak, S., 2020. Scientific computing and data analysis using numpy and pandas. International Research Journal of Engineering and Technology 7, 1334-1346.

Saveliev, A., Mukharamova, S., Zuur, A., 2007. Analysis and modelling of lattice data, in: Analysing Ecological Data. Springer, New York, pp. 321–339.

Sheckhar, M., Lumnitz, S., Arribas-Bel, D., 2020. GSOC 2020 project report: Raster Awareness in PySAL.

Shortridge, A., 2007. Practical limits of Moran's autocorrelation index for raster class maps. Computers, Environment and Urban Systems, GeoComputation 2005 31, 362–371. https://doi.org/10.1016/j.compenvurbsys.2006.07.001

Singleton, A., Arribas-Bel, D., 2021. Geographic data science. Geographical Analysis 53, 61–75. https://doi.org/10.1111/gean.12194

Slocum, T., McMaster, R., Kessler, F., Koward, H., 2014. Thematic cartography and geovisualization, 3. ed., Pearson new international. ed, Always learning. Pearson education limited, Harlw.

Steiniger, S., Hunter, A., 2013. The 2012 free and open source GIS software map – a guide to facilitate research, development, and adoption. Computers, Environment and Urban Systems 39, 136–150. https://doi.org/10.1016/j.compenvurbsys.2012.10.003

Syromiatnikov, A., Weyns, D., 2014. A Journey through the land of Model-View-Design patterns, in: 2014 IEEE/IFIP Conference on Software Architecture. Presented at the 2014 IEEE/IFIP Conference on Software Architecture (WICSA), IEEE, Sydney, Australia, pp. 21–30. https://doi.org/10.1109/WICSA.2014.13

Tobler, W., 1970. A computer movie simulating urban growth in the Detroit region. Economic Geography 46, 234. https://doi.org/10.2307/143141

Tsou, M., 2015. Research challenges and opportunities in mapping social media and big data. Cartography and Geographic Information Science 42, 70–74. https://doi.org/10.1080/15230406.2015.1059251

Tukey, J.W., 1977. Exploratory data analysis, Addison-Wesley series in behavioural science. Addison-Wesley Pub. Co, Reading, Mass.

UN-GGIM, 2020. Future trends in geospatial information management: the five to ten year vision. https://ggim.un.org/meetings/GGIM-committee/10th-Session/documents/Future_Trends_Report_THIRD_EDITION_digital_accessible.pdf

Vatsavai, R., Ganguly, A., Chandola, V., Stefanidis, A., Klasky, S., Shekhar, S., 2012. Spatiotemporal data mining in the era of big spatial data: algorithms and applications, in: Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, BigSpatial '12. Association for Computing Machinery, New York, NY, USA, pp. 1–10. https://doi.org/10.1145/2447481.2447482

Walker, I., Eamer, J., Darke, I., 2013. Assessing significant geomorphic changes and effectiveness of dynamic restoration in a coastal dune ecosystem. Geomorphology, Coastal

Geomorphology and Restoration 44th Binghamton Geomorphology Symposium 199, 192–204. https://doi.org/10.1016/j.geomorph.2013.04.023

Ward, M., 1994. XmdvTool: integrating multiple methods for visualizing multivariate data, in: Proceedings Visualization '94. Presented at the Proceedings Visualization '94, pp. 326–333. https://doi.org/10.1109/VISUAL.1994.346302

Warner, T., Shank, M., 1997. Spatial autocorrelation analysis of hyperspectral imagery for feature selection. Remote Sensing of Environment 60, 58–70. https://doi.org/10.1016/S0034-4257(96)00138-1

Young, R., 2003. Requirements Engineering Handbook. Artech House, Norwood, United States.

Zhou, S., Zhang, W., Wang, S., Zhang, B., Xu, Q., 2021. Spatial–temporal vegetation dynamics and their relationships with climatic, anthropogenic, and hydrological factors in the Amur River basin. Remote Sensing 13, 684. https://doi.org/10.3390/rs13040684

Zhou, Z., Ye, Z., Yu, J., Chen, W., 2018. Cluster-aware arrangement of the parallel coordinate plots. Journal of Visual Languages & Computing 46, 43–52. https://doi.org/10.1016/j.jvlc.2017.10.003

Zuur, A., Ieno, E., Smith, G., 2007. Analysing ecological data, Statistics for Biology and Health. Springer, New York, NY.

REFERENCED DATASETS AND SOFTWARE

Datasets

2012 and 2016 Presidential Elections, available online at https://geodacenter.github.io/data-and-lab/county_election_2012_2016-variables/, downloaded 3 October 2023

Admin 0 – Countries without boundary lakes, available online at https://www.naturalearthdata.com/downloads/110m-cultural-vectors/, downloaded 13 August 2023

AirBnB Chicage 2015, available online at https://geodacenter.github.io/data-and-lab/airbnb/, downloaded 5 October 2023

Census 2011 Descriptive Results, available online at http://superweb.statssa.gov.za/webapi/jsf/login.xhtml, downloaded 4 October 2023

CHIRPS Africa Monthly Rainfall [2023.03], available online at https://data.chc.ucsb.edu/products/CHIRPS-2.0/africa_monthly/tifs/, downloaded 9 May 2023

EU Air Quality 2016, available online at https://github.com/andrea-ballatore/open-geo-data-education, downloaded 1 August 2023

Global Precipitation 1950-2017, available online at https://github.com/andrea-ballatore/open-geo-data-education, downloaded 1 August 2023

Local Municipality Boundaries 2011, available online at https://dataportal-mdb-sa.opendata.arcgis.com/datasets/8466ca680a3e4f35b69dddca4f11e357, downloaded 4 October 2023

Malaria in Colombia (1998), available online at https://geodacenter.github.io/data-and-lab/colomb_malaria/, downloaded 5 October 2023

Simulated Spatial Autocorrelation, available online at https://geodacenter.github.io/data-and-lab/grid100/, downloaded 16 August 2023

South African Population Counts, available online at https://hub.worldpop.org/geodata/summary?id=33892, downloaded 17 August 2023

Software

ArcGIS, available online at https://pro.arcgis.com/en/pro-app/latest/get-started/get-started.htm, accessed 16 September 2023

AutoVis, available online at https://github.com/AutoViML/AutoViz, accessed 16 September 2023

Colblindor, available online at https://www.color-blindness.com/, accessed 16 September 2023

Dask, available online at https://www.dask.org/, accessed 16 September 2023

DataPrep.EDA, available online at https://docs.dataprep.ai/user_guide/eda/introduction.html, accessed 16 September 2023

DCluster, available online at https://cran.r-project.org/web/packages/DCluster/index.html, accessed 16 September 2023

ENVI, available online at https://www.nv5geospatialsoftware.com/Products/ENVI, accessed 24 October 2023

GeoDa, available online at http://geodacenter.github.io/, accessed 16 September 2023

Geopandas, available online at https://geopandas.org/en/stable/index.html, accessed 16 September 2023

Geoplot, available online at https://residentmario.github.io/geoplot/index.html, accessed 16 September 2023

GeoRasters, available online at https://georasters.readthedocs.io/en/latest/, accessed 16 September 2023

Joblib, available online at https://joblib.readthedocs.io/en/stable/, accessed 16 September 2023

MapTools, available online at https://cran.r-project.org/web/packages/maptools/index.html, accessed 16 September 2023

Matplotlib, available online at https://matplotlib.org/stable/, accessed 16 September 2023

MuseoToolBox, available online at https://museotoolbox.readthedocs.io/en/latest/index.html, accessed 16 September 2023

Numpy, available online at https://numpy.org/, accessed 16 September 2023

Pandas, available online at https://pandas.pydata.org/, accessed 16 September 2023

pygeoda, available online at https://geodacenter.github.io/pygeoda/, accessed 16 September 2023

PySAL, available online at https://pysal.org/, accessed 16 September 2023

Python, available online at https://www.python.org/, accessed 16 September 2023

QGIS, available online at https://www.qgis.org/en/site/, accessed 16 September 2023

R, available online at https://www.r-project.org/, accessed 16 September 2023

Rasterio, available online at https://rasterio.readthedocs.io/en/stable/intro.html, accessed 16 September 2023

rioxarray, available online at https://corteva.github.io/rioxarray/html/index.html, accessed 16 September 2023

seaborne, available online at https://seaborn.pydata.org/, accessed 16 September 2023

SpaceStat, available online at https://biomedware.com/products/spacestat/spacestat-details/, accessed 16 September 2023

spdep, available online at https://cran.r-project.org/web/packages/spdep/index.html, accessed 16 September 2023

Sweetviz, available online at https://github.com/fbdesignpro/sweetviz, accessed 16 September 2023

ydata-profiling, available online at https://ydata-profiling.ydata.ai/docs/master/index.html, accessed 16 September 2023

xarray, available online at https://docs.xarray.dev/en/stable/, accessed 16 September 2023

APPENDIX A: ETHICAL CLEARANCE

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Faculty of Natural and Agricultural Sciences
Ethics Committee

E-mail: ethics.nas@up.ac.za

15 November 2022

ETHICS SUBMISSION: EXTENSION LETTER

Dr V Rautenbach
Department of Geography Geoinformatics and Meteorology
Faculty of Natural and Agricultural Science
University of Pretoria

Reference number: NAS279/2021 Line 1
Project title: The development of a python library for automated exploratory spatial data analysis (ESDA)

Dear Dr V Rautenbach,

We are pleased to inform your application for ethics extension conforms to the requirements of the Faculty of Natural and Agricultural Sciences Research Ethics Committee.

Please note the following about your ethics approval:

- Please use your reference number (NAS279/2021) on any documents or correspondence with the Research Ethics Committee regarding your research.
- Please note that the Research Ethics Committee may ask further questions, seek additional information, require further modification, monitor the conduct of your research, or suspend or withdraw ethics approval.
- Please note that ethical approval is granted for the duration of the research (e.g. Honours studies: 1 year, Masters studies: two years, and PhD studies: three years) and should be extended when the approval period lapses.
- The digital archiving of data is a requirement of the University of Pretoria. The data should be accessible in the event of an enquiry or further analysis of the data.

Ethics approval is subject to the following:

- The ethics approval is conditional on the research being conducted as stipulated by the details of all documents submitted to the Committee. In the event that a further need arises to change who the investigators are, the methods or any other aspect, such changes must be submitted as an Amendment for approval by the Committee.
- **If Applications using GM permits:** If the GM permit expires before the end of the study, please make an amendment to the application with the new GM permit before the old one expires.
- **If Applications using Animals:** NAS ethics recommendation does not imply that Animal Ethics Committee (AEC) approval is granted. The application has been pre-screened and recommended for review by the AEC. Research may not proceed until AEC approval is granted.

Post approval submissions including application for ethics extension and amendments to the approved application should be submitted online via the ethics work centre.

We wish you the best with your research.

Yours sincerely,

Prof VJ Maharaj
Chairperson: NAS Ethics Committee

APPENDIX B: USER STORIES

Chapter 3 describes numerous interviews were conducted in order to generate feedback and identify potential improvements that could be made to the autoESDA ecosystem. Before conducting the interviews, users were sent an example report to investigate. The interviews were unstructured with numerous leading questions posed to the participants by the researchers. The interview participants had varying levels of experience and came from a variety of industries, ranging from software development to education. This allowed for a wealth of feedback as each interview participant brought with them a different perspective. This feedback has been translated into user stories which have been categorised into four themes, namely: General Functionality, Summary Page, Variable Information Page, and Correlation Page.

General Functionality

1. As a user I would like the library to have the functionality to accept and process raster datasets as this will allow me to work with this data format which is not supported by other popular platforms.
2. As a user I would like there to be an “About” page which describes the default settings for the various charts as well as the metadata such as the spatial weight matrix used and the date and time the report was generated. This would improve my understanding and therefore the value of the information I can extract from the report.
3. As a user I would like the library to be available on a popular platform such as PyPi or conda-forge as this will allow me to install it more easily.
4. As a developer I would like the code of the library to be refactored so that it is more modular and does not rely on the transfer of HTML strings between functions. This will improve how quickly new contributors can understand and improve the code or add new functionality.

Summary Page

1. As a user I would like the study area map to be interactive and allow popups so that I can further interact with and understand the dataset and its attributes.
2. As a user I would like to have the ability to specify the name of the study area as a parameter so that it is displayed above the study area map.
3. As a user I would like the terms “rows” and “columns” to be replaced by attributes and fields so that I do not get confused by thinking it refers to raster cells.

4. As a user I would like various spatial statistics such as the average area of polygons and average number of neighbours to be shown in the dataset overview table so that I can have a better understanding of the dataset I am working with.
5. As a user I would like to see additional descriptive statistics such as skewness, kurtosis, and the number of null and unique values as this will aid my understanding of the dataset.
6. As a user I would like to see the global Moran's I statistics in the descriptive statistics table as this will give me a quick overview and allow me to easily compare this statistic between all the attributes.
7. As a user I would like the dataset sample to consist of ten randomly selected rows as this will allow a greater chance of detecting irregularities in the dataset that may not be in the first or last five rows.

Variable Information Page

1. As a user I would like to see a key that shows what the red and blue lines of the reference distribution mean as this will improve my understanding of the chart.
2. As a user I would like to see a clustering/no clustering label on the reference distribution so as to easily understand what the interpretation of the Moran's I statistic is.
3. As a user I would like the axes to have the same scaling so that the line is not distorted as the gradient of this is usually equivalent to the Moran's I value.
4. As a user I would like a legend to illustrate what the colours on the Moran's I scatter plot mean as this will increase my understanding of the chart.
5. As a user I would like the abbreviations in the LISA scatter plot to be written out in full as this will allow me to understand what they refer to.
6. As a user I would like to specify my own spatial weights matrix as a parameter so that I have control over how my results are calculated.
7. As a user I would like the type of spatial weights used to be specified somewhere so that I know how results were calculated and I can gauge the effect the weights have on the results.
8. As a user I would like the legend of the choropleth maps to be moved outside of the map so that I can see all the features on the map.
9. As a user I would like to see additional classification schemes such as boxmap and mean-standard deviation as this will allow me to see how these schemes visualise the dataset.

10. As a user I would like the colour scheme on the choropleth maps to be reversed such that the darker colours are used to represent greater values as this is in line with conventional cartographic principles.

Correlation Page

1. As a user I would like the values in the correlation heatmap to be rounded off to two decimal places as this is more visually appealing.
2. As a user I would like to see a colour ramp outside the correlation heatmap as this will enable me to understand what the colours are indicative of.
3. As a user I would like to specify the correlation type (Pearson, Spearman, Kendall) as this will allow me to ensure that the most suitable one is used for the data that I am working with.
4. As a user I would like the name “Pairplot” to be changed to “Pairwise plot” as this is a more descriptive title.
5. As a user I would like there to be red borders around scatter plots with significant relationships (correlation > 0.7) as this will allow me to quickly identify strong relationships.
6. As a user I would like the scatter plots to have trendlines, coefficients of variation and adjusted R^2 values as this will improve the wealth of information given by the plot.
7. As a user I would like the layout of the correlation page to be improved so that I do not have to scroll so much and can spend more time understanding the data.

APPENDIX C: R SCRIPT (ORDINARY VARIANCE)

The R script below was used to inflate the variance of datasets for testing raster LISA calculations:

```
library(tiff)
library(raster)

#Read in data
small <- raster('chirps-small-231x180.tif')
medium <- raster('chirps-medium-575x409.tif')
large<-raster('chirps-large-903x593.tif')

#Store raster to matrix
Smatrix <- values(small)
Mmatrix <- values(medium)
Lmatrix <- values(large)

#Set all potential NoData values to 0
smallempty <- small@file@nodatavalue
Smatrix[Smatrix == smallempty] <- 0
Smatrix[is.na(Smatrix)] <- 0
Smatrix[Lmatrix == -9999] <- 0

mediumempty <- medium@file@nodatavalue
Mmatrix[Mmatrix == mediumempty] <- 0
Mmatrix[is.na(Mmatrix)] <- 0
Mmatrix[Lmatrix == -9999] <- 0

largeempty <- large@file@nodatavalue
Lmatrix[Lmatrix == largeempty] <- 0
Lmatrix[is.na(Lmatrix)] <- 0
Lmatrix[Lmatrix == -9999] <- 0

#Export rasters with 0 values
values(small) <- Smatrix
writeRaster(small,"small-variance-chirps-small-231x180.tif",
overwrite=TRUE)

values(medium) <- Mmatrix
writeRaster(medium,"small-variance-chirps-medium-575x409.tif",
overwrite=TRUE)

values(large) <- Lmatrix
writeRaster(large,"small-variance-chirps-large-903x593.tif",
overwrite=TRUE)

#Rescale function
rescale <- function(image){
  minn <- min(image)
  maxx <- max(image)
  newimage <- (image-minn)/(maxx-minn)
  return(newimage)
}

#New variance function
newvariance <- function(image,varvolume){
```

```
  newvar <- rnorm(n=length(image),mean=0,sd = varvolume)
  newimage <- image
  for (i in 1:length(image)){
      if (image[i] !=0){
        newimage[i] = image[i]+abs(newvar[i])
      }
    }
  return(newimage)
}

###SMALL IMAGE###
newsmall <- rescale(Smatrix)
sdSmall <- sqrt(var(c(newsmall)))
medvarsmall <- newvariance(newsmall,varvolume = sdSmall*2)
sqrt(var(c(medvarsmall)))
medvarsmallraster <- small
values(medvarsmallraster) <- medvarsmall
plot(small)
plot(medvarsmallraster)
writeRaster(medvarsmallraster,"medium-variance-chirps-small-231x180.tif",
overwrite=TRUE)

largevarsmall <- newvariance(newsmall,varvolume = sdSmall*4)
sqrt(var(c(largevarsmall)))
largevarsmallraster <- small
values(largevarsmallraster) <- largevarsmall
plot(small)
plot(largevarsmallraster)
writeRaster(largevarsmallraster,"large-variance-chirps-small-
231x180.tif",overwrite=TRUE)

###MEDIUM IMAGE###
newmedium <- rescale(Mmatrix)
sdMedium <- sqrt(var(c(newmedium)))
medvarmedium <- newvariance(newmedium,varvolume = sdMedium*2)
sqrt(var(c(medvarmedium)))
medvarmediumraster <- medium
values(medvarmediumraster) <- medvarmedium
plot(medium)
plot(medvarmediumraster)
writeRaster(medvarmediumraster,"medium-variance-chirps-medium-
575x409.tif", overwrite=TRUE)

largevarmedium <- newvariance(newmedium,varvolume = sdMedium*4)
sqrt(var(c(largevarmedium)))
largevarmediumraster <- medium
values(largevarmediumraster) <- largevarmedium
plot(medium)
plot(largevarmediumraster)
writeRaster(largevarmediumraster,"large-variance-chirps-medium-
575x409.tif",overwrite=TRUE)

###SMALL IMAGE###
newlarge <- rescale(Lmatrix)
sdLarge <- sqrt(var(c(newlarge)))
medvarlarge <- newvariance(newlarge,varvolume = sdLarge*2)
sqrt(var(c(medvarlarge)))
medvarlargeraster <- large
values(medvarlargeraster) <- medvarlarge
```

```
plot(large)
plot(medvarlargeraster)
writeRaster(medvarlargeraster,"medium-variance-chirps-large-903x593.tif",
overwrite=TRUE)

largevarlarge <- newvariance(newlarge,varvolume = sdLarge*4)
sqrt(var(c(largevarlarge)))
largevarlargeraster <- large
values(largevarlargeraster) <- largevarlarge
plot(large)
plot(largevarlargeraster)
writeRaster(largevarlargeraster,"large-variance-chirps-large-
903x593.tif",overwrite=TRUE)
```

APPENDIX D: R SCRIPT (SPATIALLY AUTOCORRELATED VARIANCE)

The R script below was used to simulate spatially autocorrelated variance in raster datasets to test the performance of the autoESDA library.

```
library(viridis)
library(ggplot2)
library(sf)
library(gstat)
library(gridExtra)
library(stars)

setwd()

mycrs <- st_crs(4326)

input_points <- read.table("50-random-points.csv", sep=",", header=T)
input_points  <-  st_as_sf(input_points,  coords=c("x_coord","y_coord"),
crs=mycrs)

mask <- read_stars("INPUT_DATASET")
st_crs(mask) <- mycrs

options(scipen=999)

g <- gstat(id=c("random_num"), formula=random_num~1, data=input_points)

vg <- variogram(g)

#CHANGE PSILL AND RANGE VALES DEPENDING ON EXPERIMENTAL VARIOGRAM
plot(vg, plot.numbers=TRUE)
vgm_low_range <- vgm(nugget=0, psill=20000000, range=10, model="Sph")
plot(vg,  vgm_low_range,  main="Experimental  Variogram  of  random_num  (Low
Range)")

vgm_med_range <- vgm(nugget=0, psill=20000000, range=100, model="Sph")
plot(vg,  vgm_med_range,  main="Experimental  Variogram  of  random_num  (Med
Range)")

vg_high_range <- variogram(g)
vgm_high_range <- vgm(nugget=0, psill=20000000, range=300, model="Sph")
plot(vg, vgm_high_range, main="Experimental Variogram of random_num (High
Range)")

vgm_line_low = variogramLine(vgm_low_range, maxdist = 650)
vgm_line_med = variogramLine(vgm_med_range, maxdist = 650)
vgm_line_high = variogramLine(vgm_high_range, maxdist = 650)

line_types  <-  c("Low  Range"="#66c2a5",  "Medium  Range"="#fc8d62",  "High
Range"="#8da0cb")

ggplot(vg, aes(x = dist, y = gamma)) +
  theme_bw() + theme(plot.title = element_text(hjust = 0.5)) +
  geom_point() +
  geom_line(data = vgm_line_low, colour='#66c2a5', size=1) +
```

```
  geom_line(data = vgm_line_med, colour='#fc8d62', size=1) +
  geom_line(data = vgm_line_high, colour='#8da0cb', size=1) +
  xlab("Distance")  +  ylab("Semivariance")  +  ggtitle("Dataset  3  -
Experimental Variogram") +  scale_color_identity(name = "Model fit", breaks
= c("#66c2a5", "#fc8d62", "#8da0cb"), labels = c("Linear", "Quadratic",
"Cubic"),guide = "legend")

krig_low_res_low_range <- krige(random_num~1, input_points, newdata=mask,
vgm_low_range)
names(krig_low_res_low_range)[1] <- "random_num.pred"
names(krig_low_res_low_range)[2] <- "random_num.var"

krig_low_res_med_range <- krige(random_num~1, input_points, newdata=mask,
vgm_med_range)
names(krig_low_res_med_range)[1] <- "random_num.pred"
names(krig_low_res_med_range)[2] <- "random_num.var"

krig_low_res_high_range <- krige(random_num~1, input_points, newdata=mask,
vgm_high_range)
names(krig_low_res_high_range)[1] <- "random_num.pred"
names(krig_low_res_high_range)[2] <- "random_num.var"

ggplot() +
  geom_stars(data=krig_low_res_low_range["random_num.pred"]) +
  scale_fill_gradient(low="yellow", high="dark blue") +
  geom_sf(data=input_points, shape=1, aes(size=random_num))
ggplot() +
  geom_stars(data=krig_low_res_med_range["random_num.pred"]) +
  scale_fill_gradient(low="yellow", high="dark blue") +
  geom_sf(data=input_points, shape=1, aes(size=random_num))
ggplot() +
  geom_stars(data=krig_low_res_high_range["random_num.pred"]) +
  scale_fill_gradient(low="yellow", high="dark blue") +
  geom_sf(data=input_points, shape=1, aes(size=random_num))

write_stars(krig_low_res_low_range["random_num.pred"],
"output_low_range.tif")
write_stars(krig_low_res_med_range["random_num.pred"],
"output_med_range.tif")
write_stars(krig_low_res_high_range["random_num.pred"],
"output_high_range.tif")
```

APPENDIX E: LINKS TO OUTPUT AUTOESDA REPORTS

Vector Reports	Raster Reports
Dataset 1 AirBnB Chicago 2015	Global Terrestrial Precipitation Band 1 | Band 2 | Band 3 | Band 4 | Stacked
Dataset 2 Grid 100	EU NOx Concentration Band 1 | Band 2 | Band 3 | Band 4 | Stacked
Dataset 3 South African 2011 Census	South African Population Band 1 | Band 2 | Band 3 | Band 4
Dataset 4 Natural Earth Country Boundaries	
Dataset 5 Malaria in Colombia	
Dataset 6 USA Election Results	

** Dissertation Ends **