International Journal of Computer Applications (0975 – 8887)
Volume 185 – No. 10, May 2023

Exploring Visualization and Development: Python vs R in a Comparative Analysis

Deep Karan Singh
India Meteorological Department, MoES
Department of Computer Science & Systems Engineering, Andhra University, Visakhapatnam

Nisha Rawat
Meteorological Office
INS Dega
Visakhapatnam

ABSTRACT

Visualization development has become increasingly important in today's data-driven world. As more and more data are generated across a wide range of industries, effective visualization has become essential for understanding and communicating insights and trends. Python and R are two of the most popular programming languages for data analysis and visualization, each with their own strengths and weaknesses when it comes to developing visualizations. This piece of work focuses on developing map visualizations using both Python and R, with the aim of creating an aesthetically appealing visualization for observations recorded by a Doppler Weather Radar (DWR) and disseminating warnings based on those recorded observations via color-coded maps. DWR provides critical weather information services to stakeholders and plays a crucial role in severe weather events, such as thunderstorms or cyclones. The reflectivity information from a DWR is utilized in this attempt to create color-coded maps, with users able to lock the severity level obtained from the information of the DWR and allocate it to a specific geographical district of the state of Andhra Pradesh in India, through an interactive graphical user interface (GUI). The software developed using Python and R has been executed and tested at the Doppler Weather Radar station at Visakhapatnam, where alerts are being issued for further dissemination of warnings. This paper highlights the differences between using two programming languages for the same development and discusses the features and benefits of each.

Keywords

Visualization development (Viz-Dev), Data Analysis, Doppler Weather Radar (DWR), Python, R, Graphical User Interface (GUI).

1. INTRODUCTION

Visualization development is an essential aspect of modern research, enabling researchers to communicate complex data in an accessible and engaging way. Visualization techniques have been used in a wide range of fields, from biology to physics, and have proven to be an effective means of conveying information. By leveraging the power of visualization, researchers can explore and analyze data in new and innovative ways, allowing them to make meaningful discoveries and gain new insights. Visualizations can take many forms, including graphs, charts, maps, and diagrams, and can be used in a wide range of research fields, including science, social science, and humanities. The ability to create effective visualizations is critical for researchers to effectively communicate their findings and engage their audience. Python and R are two popular programming languages that are widely used in the development of visualizations for data analysis and interpretation. These languages offer a wide range of libraries and tools specifically designed for data visualization, making it easier for researchers to create and customize visualizations that are tailored to their specific research needs.

1.1 Python Programming

Python programming language has gained immense popularity in recent years due to its versatility, flexibility, and ease of use. One of the areas where Python has been particularly successful is in the field of visualization development. Python has become a preferred language for creating data visualizations and interactive dashboards, thanks to its rich ecosystem of libraries and tools. In this article, the role of Python programming language in visualization development and how it has become a go-to language for data scientists, analysts, and developers has been explored. Python has a vast number of libraries that make it easier to create visualizations. Libraries like Matplotlib, Seaborn, Plotly, Bokeh, and Altair provide developers with the tools to create interactive and dynamic visualizations that can be embedded in web applications, desktop applications, or even printed reports. These libraries allow developers to create visualizations that are not only aesthetically pleasing but also highly informative. Python's versatility also makes it ideal for creating different types of visualizations, including static and dynamic visualizations. Python provides several plotting libraries that enable developers to create different types of plots, such as line plots, scatter plots, bar plots, and histograms. Python's visualization tools also allow for the creation of more complex visualizations, such as heatmaps and 3D surface plots. One of the most significant advantages of Python is its ability to handle large datasets. Python provides various data manipulation tools and techniques that make it easier to preprocess data before visualization. Additionally, Python has several tools that can be used for data exploration, such as Jupyter Notebook, which allows developers to explore data and create visualizations interactively. Another advantage of using Python for visualization development is its ability to create interactive visualizations. Python's libraries provide support for creating interactive visualizations that can respond to user inputs, including mouse clicks and hover events. This feature allows users to explore data and gain insights in a more engaging and interactive way.

1.2 R programming

R programming is a powerful tool for developing data visualizations, including map visualizations. With its robust data analysis capabilities and wide range of visualization libraries, R makes it easy to create dynamic and informative maps that can be used for a variety of purposes. One of the key benefits of using R for visualization development is its flexibility. R provides a wide range of packages that enable users to create visualizations in a variety of formats, including bar charts, scatterplots, heatmaps, and more. Additionally, R allows for the creation of custom visualizations that can be tailored to specific needs or datasets. When it comes to map visualizations, R really shines. With packages like ggplot2 and leaflet, users can create dynamic and interactive maps that allow for exploration and analysis of geographic data. These maps can be customized with a variety of features, including markers, labels, and interactive layers that can be toggled on and off. In addition to its flexibility and map visualization capabilities, R also excels in data preparation and analysis. With packages like dplyr and tidyr, users can easily clean, transform, and manipulate their data to prepare it for visualization. This makes it possible to create maps that are not only visually appealing, but also accurate and informative.

1.3 Graphical User Interface

A graphical user interface (GUI) is a type of user interface that allows users to interact with electronic devices through visual elements such as icons, buttons, menus, and windows. GUIs are designed to be easy to use and intuitive, enabling users to perform tasks quickly and efficiently. The use of graphical user interfaces has become widespread in modern computing, from desktop and mobile operating systems to web applications and video games. The design and implementation of effective GUIs require careful consideration of user needs and behavior, as well as technical constraints and capabilities. One of the key benefits of GUIs is their visual appeal and ease of use. Users can quickly understand how to navigate a GUI and perform tasks without needing to learn complex commands or programming languages. GUIs also provide feedback to users through visual cues, such as changing the color of a button when it is clicked or displaying a progress bar during a long-running task. Graphical user interfaces can be designed in many different ways, depending on the device or application they are intended for. For example, a desktop operating system may use a window-based interface, while a mobile app may use a tabbed or swipe-based interface. Web applications may use a combination of menus and forms to allow users to enter and retrieve data. One common feature of GUIs is the use of icons, which represent applications, files, and other objects. Icons can be designed to be recognizable and intuitive, such as using a magnifying glass icon to represent a search function or a trash can icon to represent deleting a file. Icons can also be customized by users, allowing them to personalize their interface and make it easier to find commonly used applications or files. Another important aspect of graphical user interfaces is the use of menus and toolbars, which provide access to a range of functions and features. Menus can be hierarchical, allowing users to drill down into sub-menus to access more specific options. Toolbars can provide quick access to commonly used functions, such as copy, paste, and undo. GUIs can also incorporate input devices such as keyboards, mice, touchscreens, and voice recognition. These devices can be used to enter text, select options, and perform other tasks. For example, a touchscreen interface may allow users to swipe, pinch, or tap to navigate through an application or enter data. The design and implementation of effective graphical user interfaces requires a combination of technical expertise and user-centered design principles.

1.4 Doppler Weather Radar

Doppler weather radar is a technology used to detect and track precipitation and other atmospheric phenomena, such as thunderstorms, hurricanes, and tornadoes. It is a powerful tool that provides meteorologists and other weather professionals with critical information about the movement and intensity of weather patterns. Doppler weather radar works by emitting high-frequency radio waves that bounce off objects in the atmosphere, such as raindrops or hailstones. The radar then analyzes the reflected waves to determine the distance, velocity, and direction of the moving objects. One of the key benefits of Doppler weather radar is its ability to detect and track the movement of storms and other weather patterns in real-time. This allows meteorologists to issue timely warnings and alerts to the public, helping to minimize the risk of property damage and loss of life. Doppler weather radar can also provide detailed information about the size and intensity of precipitation, allowing meteorologists to accurately predict the amount of rainfall or snowfall that a particular area is likely to experience. This information can be used to help local authorities prepare for potential flooding or other weather-related emergencies. In addition to its use in weather forecasting, Doppler weather radar is also used in aviation, military, and scientific applications. For example, it can be used to detect and track aircraft, missiles, and other objects in the sky, as well as to study the movement and behavior of birds and other wildlife.

2. LITERATURE REVIEW

Visualization and development tools play a crucial role in the field of data science, aiding in data exploration, analysis, and presentation. Among the popular programming languages used for these tasks are Python and R. This literature review aims to provide a comparative analysis of Python and R as visualization and development tools based on the findings of several relevant studies. The study conducted by [1] Brittain et al. (2018) focuses on comparing the performance of Python, R, and SAS in the context of data analysis. It provides insights into the strengths and weaknesses of each language, shedding light on their capabilities and efficiency for data-related tasks. [2] Millman and Aivazis (2011) discuss the relevance of Python as a powerful tool for scientists and engineers. The article highlights Python's versatility, ease of use, and extensive libraries for scientific computing, making it an attractive option for data visualization and analysis. In their study, [3] Ozgur et al. (2017) compare MatLab, Python, and R, emphasizing their capabilities in data science applications. The article provides insights into the different features, libraries, and development environments offered by each language, helping researchers and practitioners make informed decisions. [4] Zhang et al. (2011) explore the application of Python and ArcGIS software in remote sensing data management. This study showcases Python's effectiveness in handling geospatial data and demonstrates its potential for visualization and analysis in the field of remote sensing. [5] Singh (2017) presents a compelling argument for learning both R and Python. The article highlights the unique strengths and applications of each language, suggesting that a combination of the two can enhance data analysis capabilities and expand the range of tools available to data scientists. [6] McKinney (2013) provides an introduction to Python for data analysis, emphasizing its role in processing and visualizing data. The book offers practical examples and insights into using Python's data manipulation libraries, such as pandas, making it a valuable resource for understanding Python's capabilities in the context of data analysis. The reviewed literature provides valuable insights into the comparative analysis of Python and R as visualization and development tools. These studies highlight the unique features, performance, and libraries offered by each language, allowing data scientists and researchers to make informed choices based on their specific requirements. While Python stands out for its versatility and extensive libraries for scientific computing, R offers a comprehensive suite.

3. DEVELOPMENT ANALYSIS

The research paper aims to conduct a comprehensive comparison between the programming languages Python and

R, specifically in the context of a practical use-case. The use-case involves the input and analysis of weather observations obtained from a Doppler Weather Radar, with the ultimate goal of disseminating weather warnings to users. In order to achieve this, the authors have developed code in both Python and R, which provide a similar interface and accomplish the same task. The initial step in both code implementations is to read the shapefile for the Andhra Pradesh state in India, as depicted in Figure 1. This shapefile represents the geographic boundaries of the 26 districts within the state. The duty officers responsible for monitoring weather conditions utilize the observations recorded by the Doppler Weather Radar located in Visakhapatnam. These observations are used to allocate weather severity to the various districts within Andhra Pradesh. This allocation is facilitated through the utilization of a graphical user interface (GUI) that employs color-coding.

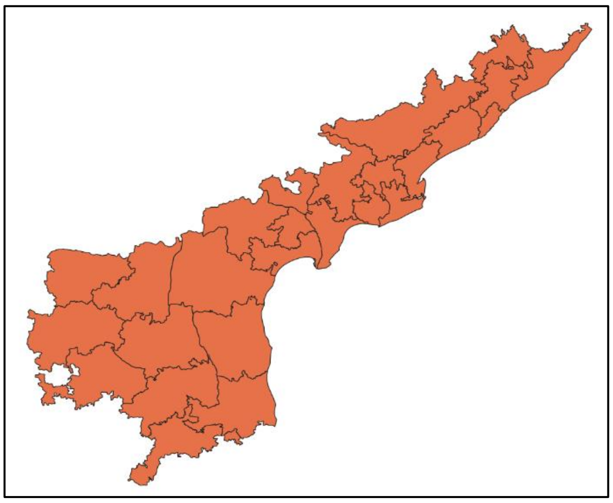

Fig 1: District shapefile for the state of Andhra Pradesh in India

The GUI includes a canvas where a plot is displayed, which is updated with the date and time of the recorded observation. Additionally, an alert warning is generated and disseminated to the relevant parties based on the severity of the weather conditions detected by the radar.

3.1 Use of Doppler Weather Radars

Doppler Weather Radars in India are under the administrative and operational control of India Meteorological Department. These DWRs generate volumetric scan data employing two operational scan strategies: IMD-B and IMD-C. IMD-C is a long-range scan covering 500 km with two elevations, while IMD-B is a short-range scan encompassing 250 km with ten elevation scans ranging from 0.2° to 21.0°. The volumetric raw data obtained from IMD-B scans, represented in antenna coordinates of slant range, elevation angle, and azimuth angle from true North, serves as input for the system.

The most widely-used product from a DWR available is the MAX_Z (Maximum Reflectivity) which takes a polar volume raw data set, converts it to a cartesian volume, generates three partial images and combines them to the displayed image. The MAX_Z image generated by DWR at Visakhapatnam can be seen in Figure 2.

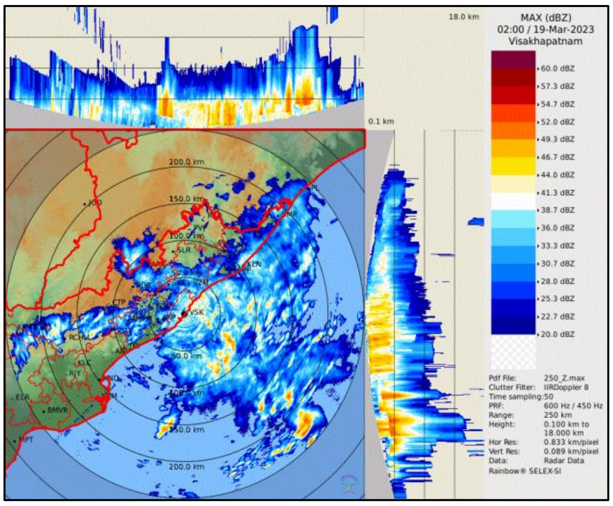

Fig 2: MAX_Z imagery generated by DWR Visakhapatnam (Source: IMD)

The partial images inside the MAX_Z are the three projections or views available to the interpreter giving different information:

• A **top view** of the highest measured values in Z-direction
This image shows the highest measured value for each vertical column.

• A **north-south** view of the highest measured values in Y-direction
This image is appended above the top view and shows the highest measured value for each horizontal line seen from north to south.

• An **east-west** view of the highest measured values in X-direction
This image is appended to the right of the top view and shows the highest measured value for each horizontal line seen from east to west.

Table 1 Rainfall categorization with radar reflectivity

S No	Rainfall category	dBZ	Rain intensity
1	Mist to light	15 - 30	0.2 – 2 mm
2	Moderate	30 - 40	2 – 10 mm
3	Heavy	40 - 46	10 – 30 mm
4	Very heavy	46 - 50	30 – 50 mm
5	Intense	50 - 56	50 – 100 mm
6	Extreme (including hail)	> 56	Over 100 mm

(Table source: Marshall-Palmer relation)

In the present work, the operators need to feed the values manually in the GUI prepared for the software. The operator can hover the cursor over the generated product in the server of the radar and would get the value and location of the reflectivity observed. Based on the feeding of the value of the radar-derived reflectivity in the software GUI, the plot can be updated by the user and dissemination of warnings can take place. The severity values to be assigned to a particular district is as per the Table 1.

3.2 Python use-case

The Python development results in the creation of a graphical user interface (GUI) that features a plot displayed on the canvas on the right-hand side. The left-hand side of the GUI allows users to select multiple options via a sidebar panel. The plot on the right-hand side shows the shapefile of Andhra Pradesh, with acronyms representing the district names located at the geometric centroid of the respective district. The GUI is built using tkinter, and a visual representation of it can be viewed in Fig 3.

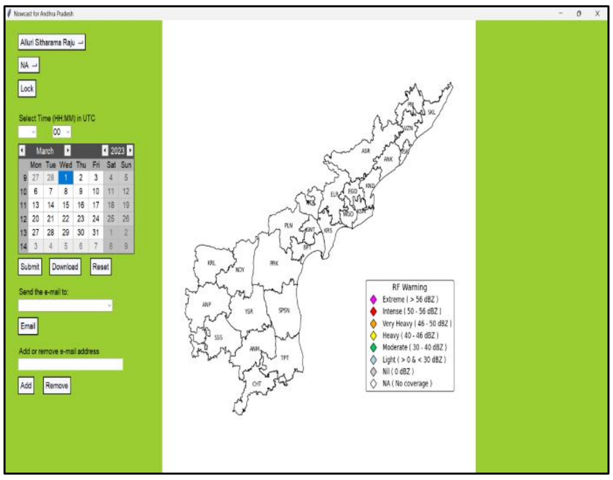

Fig 3: GUI developed using tkinter in python

The left-hand side bar panel of the GUI contains a drop-down menu, which when clicked, displays a list of the 26 districts of Andhra Pradesh based on the contents of the shapefile. The default selected district is displayed on the bar, and users can select any of the 26 districts to which severity allocations need to be made. The user is then required to select the severity level to be allocated to the previously selected district. On clicking the next available Lock button, the selected severity level will be assigned to the district. The user can select multiple severities and lock them as required. Next, the user can select the time in UTC (HH:MM format) and date from the calendar provided on the left-hand side bar panel. On clicking the submit button, the selected time and date are displayed on the plot above the map. Clicking the Download button downloads the last updated plot on the canvas as a png file. The reset button, when clicked, erases all the allocated color-coded severities to the districts and presents the map afresh to the user. The last-updated plot can be sent as an attachment in an e-mail by selecting the target e-mail address from the drop-down menu and clicking on the e-mail button.

At the bottom of the side-bar panel in GUI is a text box that enables users to input any e-mail address. Upon clicking the Add button, the entered e-mail address is added to the drop-down list of e-mail addresses. A visual representation of the randomly allocated severity levels to different districts can be viewed in Fig 4. The allocations depicted in Fig 4 can be downloaded as a png file with the selected date and time from the GUI mentioned in the plot. The plot also features an index with color-coded rainfall warnings, which makes interpretation easier.

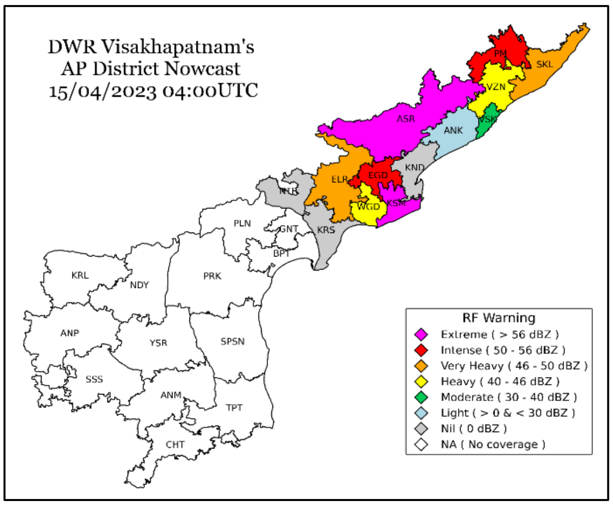

Fig 4: Downloaded PNG file

The breakdown of the Python code is as follows:

1. **Importing Required Libraries**: The script starts by importing the necessary libraries, including geopandas, pandas, matplotlib, smtplib, email-related modules, PIL (Python Imaging Library), io, imageio, tkinter, and other libraries required for the functionality.
2. **Defining Color Scheme**: The script defines color variables for the GUI interface.
3. **Reading Shapefile**: The script reads a shapefile containing the district boundaries of Andhra Pradesh from a specified path using the geopandas library.
4. **Mapping District Names to Acronyms:** A dictionary named "district_codes" is created to map district names to acronyms.
5. **Creating Severity Colors**: A dictionary named "selected_severity_colors" is created to map severity levels to colors.
6. **Initializing GUI**: The script initializes a Tkinter window and sets its properties such as the title and background color.
7. **Creating Dropdown Menus and Calendar Widget:** Dropdown menus for district selection and severity selection are created using the OptionMenu widget. Additionally, a Calendar widget from the tkcalendar library is placed in the GUI for selecting a date.
8. **Selecting Time**: Combobox widgets are created for selecting the hour and minute in UTC time format.
9. **Submit Function**: The submit function is defined to retrieve the selected date and time values and update the plot accordingly.
10. **Lock Button**: A lock button is created, which triggers the update_map function when clicked.
11. **Update Map Function**: The update_map function is responsible for updating the map based on the selected district and severity.
12. **Download Plot Function**: The download_plot function is defined to save the current plot as a PNG file.
13. **Recipient Email Addresses**: A list named "recipients" is created to store recipient email addresses.
14. **Adding and Removing Recipients**: Functions add_recipient and remove_recipient are defined to add or remove email addresses from the recipients list.
15. **Emailing Plot**: The send_email function is defined to send an email with the plot attached as a PDF.
16. **Reset Function**: The reset function is defined to reset the GUI components and plot to their initial state.
17. **Call to Update Map**: The update_map function is called

initially to update the map with the default district and severity.
18. **Embedding Plot in Tkinter**: A Tkinter canvas object is created to embed the plot in the GUI.
19. **Setting GUI Properties**: The background color of the window is set, and the window is maximized.
20. **Tkinter Main Loop**: The script enters the Tkinter main loop to start the GUI application.

3.3 Use-case involving R

The development using R has been designed to resemble the development using Python as closely as possible, as depicted in Fig 5. The user interface in R shares similarities with that in Python, with the plot displayed on the canvas on the right-hand side and various options available to the user on the left-hand side bar panel. The plot on the right displays the 26 districts of Andhra Pradesh in India, accompanied by the acronyms for district names positioned at the centroid of the geographic coordinates. The first option on the left-hand side bar panel enables the user to select a particular district from the drop-down menu, which presents a list of the 26 districts of Andhra Pradesh. Once the district has been selected, the user can assign a severity level to it from the drop-down menu. Upon clicking the Lock button, the assigned severity will be displayed in color-coded format on the district. The user can repeat this process of selecting a district and assigning severity level multiple times as needed.

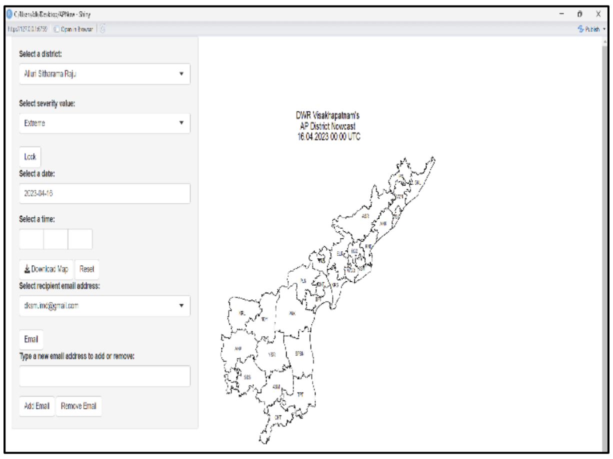

Fig 5: GUI developed using shiny app in R

After locking the severities to the districts, the user can select the date and time, which will be displayed at the top of the plot as the observation time. Clicking on the date field displays a calendar for the user to make a selection. The user can then select the time in HH:MM format. If the time is not selected, the system date will be used, and the time will default to 00:00 UTC. Clicking the Download Map button downloads the last-updated plot as a PNG file to a folder of the user's choice. Clicking the RESET button erases all the user's allocations made till that point. The user can then select a target e-mail address from a drop-down list to which the last-updated plot will be sent as an attachment in PNG format on clicking the EMAIL button. The last widget on the interface allows the user to add or delete an e-mail address from the drop-down list. The user can enter the e-mail address to be added in the last text box and click the ADD button. If any e-mail address needs to be removed from the list, the user can select the address and click the REMOVE EMAIL button.

As shown in Fig 6, as the user assigns severities to districts, a legend appears at the bottom of the map, displaying the interpretation of color-coding for easy understanding.

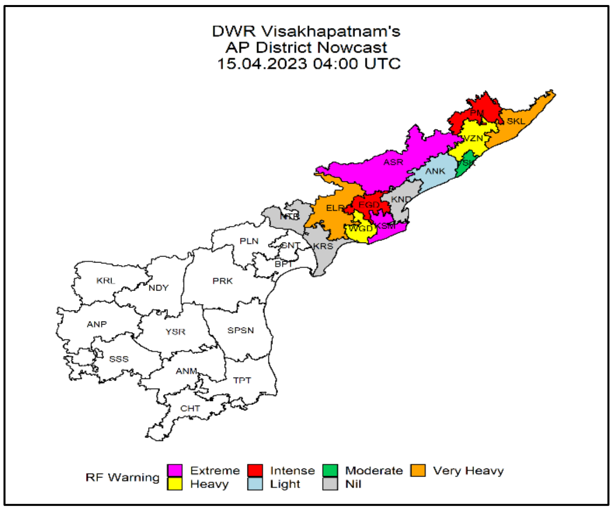

Fig 6: Downloaded PNG file from Shiny App

The following is the breakdown of the R code:

1. The necessary R packages are imported.
2. The shapefile containing the district boundaries is read using the st_read function from the sf package.
3. A named vector of district acronym values is defined.
4. A named vector of severity values and colors is defined.
5. The UI (User Interface) for the Shiny app is defined using the fluidPage function from the shiny package.
6. The server logic is defined using the server function.
    - Reactive values and reactive expressions are used to store and update the user-selected severity allocations and the last generated plot.
    - Various Shiny input controls (select inputs, action buttons, date input, time input) are defined in the sidebar panel of the UI.
    - Event observers are set up to respond to user interactions with the input controls and perform corresponding actions (e.g., updating severity allocations, resetting the allocations, sending an email with the plot).
    - The severity allocations are merged with the district data and used to generate the map plot using ggplot2.
    - The generated plot is displayed in the main panel of the UI.
    - The user can download the plot as a PNG file by clicking the "Download Map" button.
    - The user can add or remove email addresses for recipients of the plot by typing them in the text input fields and clicking the corresponding buttons.
    - The user can send an email with the plot as an attachment to the selected recipient email address by clicking the "Email" button.
7. The Shiny app is run using the shinyApp function, which takes the defined UI and server logic as arguments.

4. RESULT

In this study, the usage of popular programming languages Python and R for map visualization has been compared. Through analysis, several advantages and disadvantages associated with each language have been observed. One notable distinction between Python and R lies in the availability of libraries for working with maps. Python offers a wide range of robust mapping libraries, including Folium, GeoPandas, and Basemap, which provide user-friendly tools for creating maps and manipulating geographic data. R also provides mapping libraries such as ggmap and leaflet, but Python's libraries are generally more versatile and easier to use. Another significant difference is the syntax used to create maps in each language. Python utilizes object-oriented programming techniques, allowing maps to be created as objects with customizable methods and attributes. In contrast, R typically uses functions and parameters to generate maps, resulting in more concise but potentially more complex code.

Python is often paired with libraries like Pandas and NumPy for data manipulation, enabling seamless integration of map visualizations into larger data analysis workflows. On the other hand, R has its own built-in data manipulation functions, making it easier to handle data within the language. Additionally, R has a strong foundation in statistical analysis and modeling, providing numerous built-in functions that are not available in Python.

Table 2 Comparing the application of Python and R

Aspect	Python	R
Availability of Libraries	Folium, GeoPandas, Basemap	ggmap, leaflet
Mapping Syntax	Object-oriented programming techniques	Functions and parameters
Data Manipulatio n	Seamless integration with Pandas and NumPy	Built-in data manipulation functions
Statistical Analysis	Limited built-in functions for statistical analysis	Strong foundation in statistical analysis and modeling
Lines of Code	More lines of code required for the task	Fewer lines of code required for the task
Executable Files	PyInstaller tool for creating bundled executables	Script execution with library loading and .R extension

The study also compared the number of lines of code required to achieve the desired task. Interestingly, the authors found that Python required more lines of code compared to R in this particular development. While Python is generally known for its clean syntax and concise code, the specific requirements of the task resulted in a greater number of lines for Python. This highlights the importance of considering the specific context and requirements when evaluating the code length in different languages. Figure 7 shows the lines of code required by both the languages.

Furthermore, the approach to generating an executable file differs between Python and R. Python offers the PyInstaller tool, which allows bundling a Python application and its dependencies into a single executable file. In contrast, the authors utilized a script in R that loaded the required libraries and executed the desired functions, saving it with a .R extension to generate an executable. Overall, Python provides more convenience in creating and sharing executables without exposing the source code.

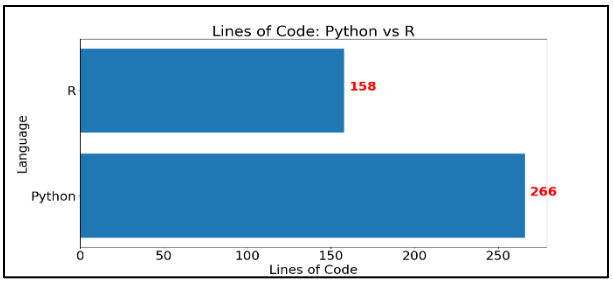

Fig 7 Comparisons of lines of code

5. CONCLUSION

This research provides valuable insights into the usage of Python and R for map visualization, highlighting their respective advantages and disadvantages. The study demonstrates that Python excels in terms of its extensive mapping libraries, versatility, and ease of integration with data manipulation tools. However, R has its own strengths, particularly in statistical analysis and modeling, and can offer more concise code in certain scenarios. Looking ahead, there are several areas of future work to be considered. Firstly, expanding the comparison to include other programming languages commonly used for map visualization, such as JavaScript or Julia, would provide a broader perspective. Additionally, evaluating the performance of Python and R in handling larger datasets and real-time data streams would be beneficial, as this study focused on a specific use-case. Furthermore, exploring the potential for optimizing code length and execution time in both languages could enhance the efficiency of map visualization tasks. This research presented a comprehensive comparison of Python and R for map visualization, highlighting their strengths and weaknesses. It ultimately emphasizes the importance of considering the specific requirements and context of a project when choosing between these two powerful and popular programming languages to achieve greater efficiency and optimality. The future scope of work includes further exploration of alternative programming languages, scalability testing, and code optimization strategies.

6. REFERENCES

[1] Jim Brittain, Mariana Cendon, Jennifer Nizzi, John Pleis. “Data Scientist’s Analysis Toolbox: Comparison of Python, R, and SAS Performance.”SMU Data Science Review. SMU Data Center, 2018.

[2] J.K.Millman & M. Aiyazis, "Python for Scientists and Engineers" Computing in Science & Engineering, vol. 13, no. 12, pp. 9-12, 2011.

[3] Ceyhun Ozgur, Taylor Colliau, Grace Rogers, Zachariah Hughes,Elyse “Bennie” Myer-Tyson.” MatLab vs. Python vs. R.” Journal of Data Science 15,355-372.2017.

[4] Zhang, Jing, Hongxia Luo, and Xueqing Zhang. "Application of python language and arcgis software in RS data management." 2011 International Conference on Remote Sensing, Environment and Transportation Engineering. IEEE, 2011.

[5] Suryansh Singh. “R vs Python, Why you should learn both?”Quadratyx Power of Insight.2017.

[6] W. McKinney, "Chapter 1- Preliminaries," in Python for Data Analysis, Sebastopol, O'Reilly Media, 2013, p. 3.