RESEARCH PLAN
DA1: RSTUDIO & GGPLOT

Group name: KHDL-02

List of members:
- Đoàn Ngọc Mai - 21127104
- Lê Nguyễn Kiều Oanh - 21127129
- Dương Trường Bình - 21127229
- Lê Phước Quang Huy - 21127616

Keywords: RStudio, ggplot2, Visualization, Data Analysis, Statistical Graphics

Description: In the context of the rapid development of data science and artificial intelligence, understanding and effectively using data analysis tools is crucial. RStudio is a powerful integrated development environment (IDE) for the R programming language, specifically designed to support data analysis and visualization. Among the robust libraries in R, ggplot2 stands out as a leading tool for creating high-quality statistical graphics. With ggplot2, users can easily create customized, intuitive, and interactive plots, providing in-depth and detailed insights into the data

List of references:

- "Introduction to R," RPubs. Available at: https://rpubs.com/ngocnv/Intro_R
- Dhiraj Kumar. "Kaggle Survey 2022 Data Analysis," Kaggle. Available at: https://www.kaggle.com/code/dhirajkumar612/kaggle-survey-2022-data-analysis
- Wickham, H., & Grolemund, G. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Available at: https://ggplot2-book.org
- "ggplot2," Tidyverse. Available at: https://ggplot2.tidyverse.org/articles/ggplot2.html
- "Corona Virus Report Dataset," Kaggle. Available at: https://www.kaggle.com/datasets/imdevskp/corona-virus-report

Presentation outline

1. **Introduction:** an overview of the selected solution (i.e., a tool, a project or a product that provides great helps to Big Data research)

    - R is a programming language and software environment developed primarily for statistical computing and graphics. R was derived from the S language, which John Chambers and colleagues at Bell Laboratories originally created in the 1970s. Ross Ihaka and Robert Gentleman created the R language at the University of Auckland, New Zealand, and was first released in 1995. R is notable for its powerful data processing and analysis capabilities, offering thousands of extension packages, and is an essential tool in the fields of data science and statistical analysis.
    - RStudio is an integrated development environment (IDE) for R. It provides a user-friendly interface and numerous features to support programming, data analysis, and data visualization. The main features of RStudio include:
        - Intuitive user interface
        - Integration with R
        - Project management
        - Debugging support
        - Integration with Git and SVN
        - Support for Markdown and R Markdown
    - ggplot2 is a powerful and flexible graphics library in R, developed by Hadley Wickham. ggplot2 is based on the "Grammar of Graphics," a theory on systematically building plots. The main characteristics of ggplot2 include:
        - Layered structure
        - High flexibility
        - Easy customization
        - Support for large datasets
        - Integration with dplyr and tidyr
    - Which problems could be solved by using this solution?

    RStudio combined with ggplot2 is a powerful solution for addressing many important issues in Big Data research.

    - **Statistical Data Analysis:**
        - Calculate descriptive statistics such as mean, standard deviation, and percentiles.
        - Perform hypothesis tests such as t-tests, ANOVA, and chi-square tests.
    - **Data Visualization:**
        - Create basic charts such as bar charts, line charts, and scatter plots.
        - Create complex charts such as box plots, heat maps, and grid plots.

- **Time Series Data Analysis:**
  - Analyze time series data, forecast trends, and seasonality.
  - Create time series plots to visualize data over time.
- **Multivariate Data Analysis:**
  - Perform linear and nonlinear regression analysis.
  - Conduct principal component analysis (PCA) and clustering.
- **Big Data Processing:**
  - Use libraries such as dplyr and data.table to process and transform large datasets.
  - Efficiently visualize large datasets with ggplot2.
- **Reporting and Presenting Results:**
  - Use R Markdown to create dynamic reports combining R code and descriptive text.
  - Create interactive dashboards with Shiny.

- **A brief history**: motivations and evolution through years

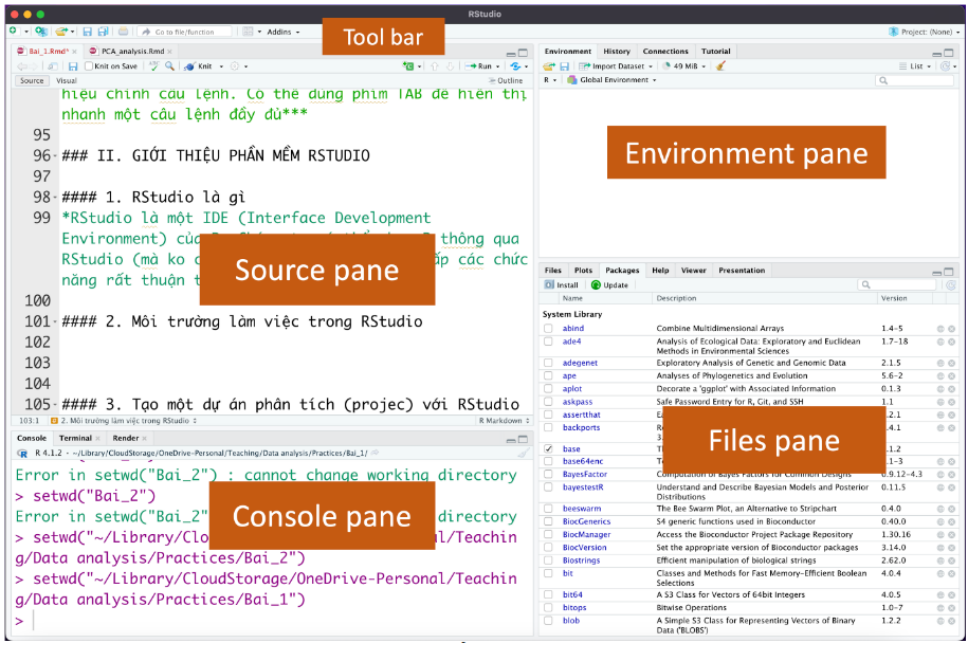
- **RStudio**
  - **Launch:** RStudio was first released in 2011 by J.J. Allaire, the founder of RStudio, PBC (formerly RStudio, Inc.).
  - **Motivation:** The goal of RStudio is to provide a powerful and user-friendly integrated development environment (IDE) for R, making it easier for users to write code, run code, and visualize data.
  - **Development:** Since its launch, RStudio has been continuously updated and improved with many new features such as support for R Markdown, integration with Git and SVN, and support for other data analysis tools like Python and SQL.
- **ggplot2**
  - **Launch:** ggplot2 was developed by Hadley Wickham and first released in 2005.
  - **Motivation:** ggplot2 is based on Leland Wilkinson's "Grammar of Graphics" theory, aiming to provide a flexible and powerful system for creating high-quality charts.
  - **Development:** ggplot2 has become one of the most popular graphics libraries in R and has been integrated into many other tools and libraries within the R ecosystem. Hadley Wickham and the R community have continuously improved and expanded ggplot2 with new features and performance enhancements.

2. A deeper insight to the selected solution

2.1 Major components and main functionalities: consider what are worth to be presented and support them with demonstrative figures and examples

RStudio: RStudio is a powerful integrated development environment (IDE) for the R language. It provides a range of tools to support programming, data analysis, and data visualization.

Key Components of RStudio:

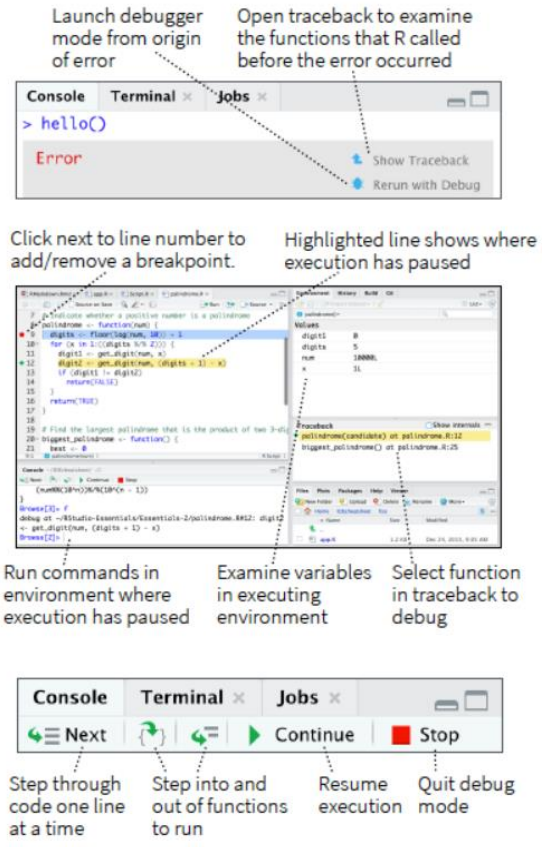


1. **Source Pane:**
    - Allows users to write, edit, and save R scripts.
    - Supports code suggestions, auto-completion, and syntax highlighting.
2. **Console Pane:**
    - Executes R code directly and displays results.
    - Supports interactive commands and line-by-line code testing.
3. **Environment Pane:**
    - Displays current variables and data in the session.
    - Manages data and various objects in the R environment.
4. **Plots Pane:**
    - Shows plots and charts created by ggplot2 or other tools.
    - Supports saving and exporting plots in different formats.
5. **Files, Packages, Help, Viewer Panes:**
    - **Files:** Manages project files and directories.
    - **Packages:** Manages installed R packages.
    - **Help:** Accesses documentation and help for functions and packages.
    - **Viewer:** Previews HTML, PDF, and other document formats.

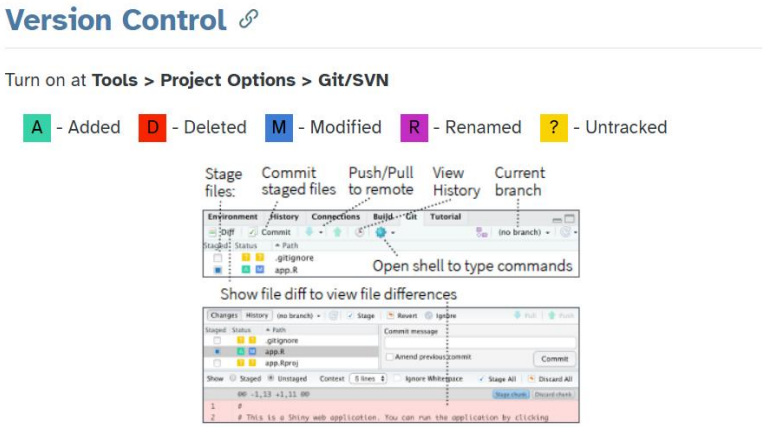
Main Functions of RStudio:

- **Code Editing:**
    - Provides code suggestions and auto-completion, with multi-language support.

- Powerful debugging tools for easy code inspection and error fixing.

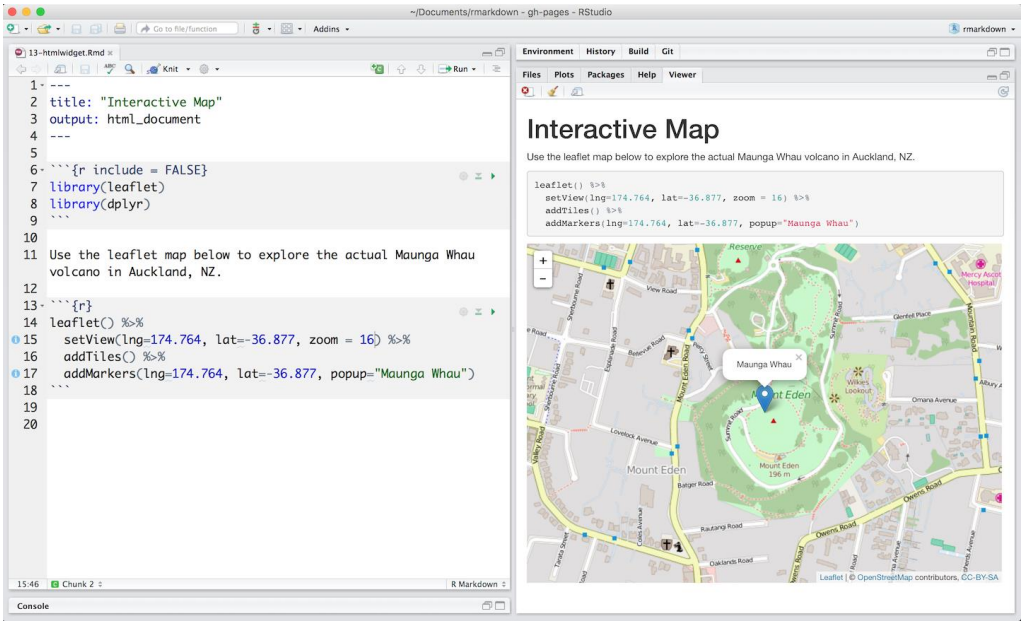


- **Project Management:**
  - Supports managing files and directories within projects, helping users organize and access them quickly.
  - Integration with Git and SVN for source code version control.

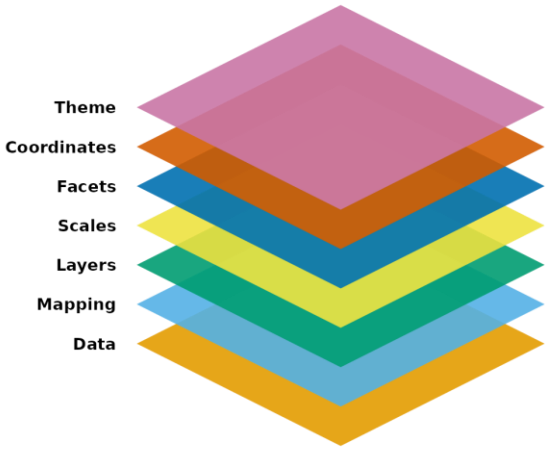


- **Interactive Notebooks:**

- Uses R Markdown to create dynamic documents that combine code, text, and visual results.



ggplot2



ggplot2 is a powerful graphics package in the R programming language, based on the "Grammar of Graphics" developed by Leland Wilkinson. This package provides a structured and flexible system for constructing graphs. It utilizes seven composable parts that come together to form instructions on how to draw a chart. Out of these components, ggplot2 needs at least the following three to produce a chart: data, a mapping, and a layer. The scales, facets, coordinates, and themes have sensible defaults that simplify much of the detailed work.

All plots are composed of:

- **Data:** The information you want to visualize.
- **Mapping:** The description of how the data's variables are mapped to aesthetic attributes.

Key Components of ggplot2:

1. **Data:** The foundation of every graphic. ggplot2 uses data to construct a plot, which works best if provided in a tidy format—a rectangular data frame structure where rows are observations and columns are variables.
    - ***Example*: ggplot(data)**
2. **Mapping:** A set of instructions on how parts of the data are mapped onto aesthetic attributes of geometric objects. It acts as a 'dictionary' to translate tidy data to the graphics system.
    - ***Example*: aes()**
3. **Layer:** The heart of any graphic. Layers take the mapped data and display it in a form understandable to humans, consisting of:
    - **Geometry:** Determines how data are displayed, such as points, lines, or rectangles.
    - **Statistical Transformation:** Computes new variables from the data and affects what of the data is displayed.
    - **Position Adjustment:** Determines where a piece of data is displayed.
    - ***Example*: geom_*(), stat_*()**
4. **Scale:** Translates what is shown on the graph back to an understanding of the data. Scales form pairs with aesthetic attributes and are represented in plots by guides like axes or legends. They update plot limits, set breaks, format labels, and possibly apply transformations.
    - ***Example*: scale_x_continuous(), scale_y_discrete(), scale_color_manual()**
5. **Facets:** Separate small multiples or different subsets of the data into smaller panels, based on one or more variables, to display patterns or trends within the subsets.
    - ***Example*: facet_wrap(), facet_grid()**
6. **Coordinates:** Interpreters of position aesthetics. Typically, Cartesian coordinates are used, but the system also powers map projections and polar plots.
    - ***Example*: coord_cartesian(), coord_flip(), coord_polar()**

7. **Theme:** Controls almost any visual aspect of the plot not controlled by the data. Customizations range from changing legend locations to setting background colors. Themes are hierarchical, meaning that setting the look of the general axis line affects both x and y axes simultaneously.
    - ***Example*: theme(), theme_minimal(), theme_bw()**

2.2 Its applications in academic and/or industry activities

RStudio and ggplot2 are powerful tools widely used in both academia and industry due to their strong data analysis and visualization capabilities. Here are some of their main applications in these fields:

Academic:

- **Data Research:** RStudio is used for data analysis and hypothesis testing in research projects, thanks to its ability to handle large datasets and provide advanced statistical analysis tools.
- **Teaching:** RStudio and ggplot2 are essential tools in courses on statistics, data analysis, and machine learning, due to their user-friendly interface and comprehensive documentation.
- **Scientific Writing:** RStudio supports creating reports and scientific papers through R Markdown and LaTeX, making it easier for researchers to write and format their documents.
- **Reproducibility and Research Sharing:** The ability to reproduce research and share source code allows other researchers to verify and expand upon existing studies.

Industry:

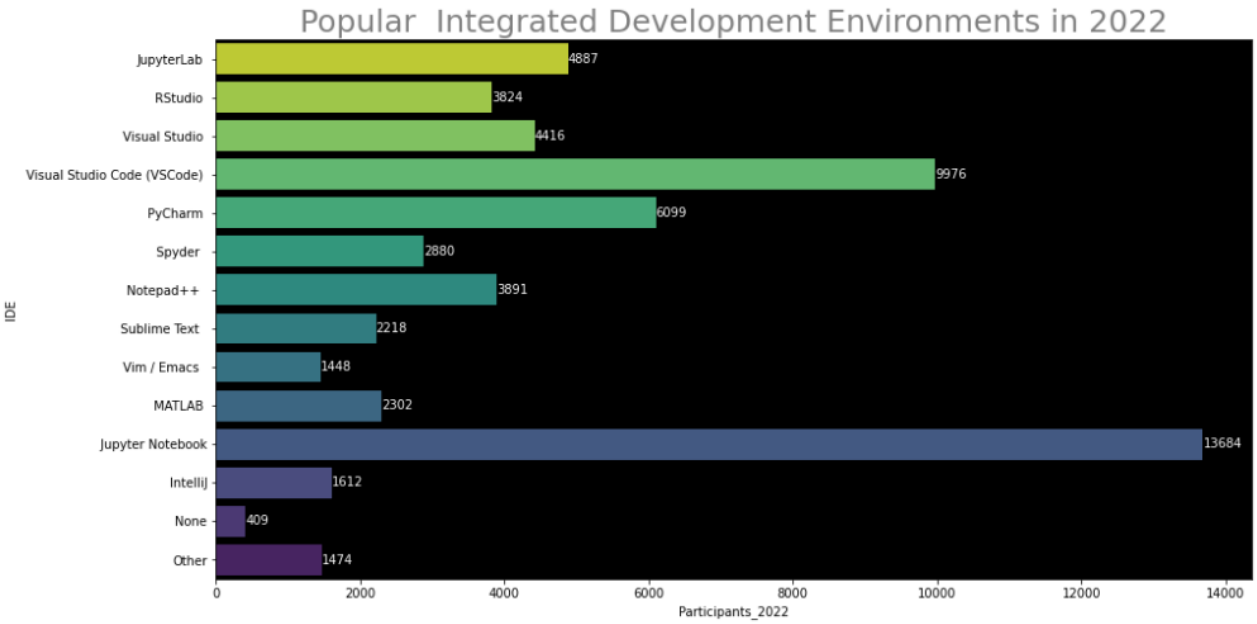
- **Business Data Analysis:** RStudio helps analyze trends, forecast, and extract insights from customer, market, and financial data.
- **Financial Analysis:** Used in risk analysis, financial modeling, and forecasting with complex statistical models.
- **Data Science:** RStudio and ggplot2 are popular in tech companies for building machine learning models and analyzing big data.
- **Healthcare and Pharmaceuticals:** Analyzing medical data, conducting clinical research, and developing pharmaceuticals using RStudio to process and analyze data from trials and medical studies.

RStudio and ggplot2 are not only critical tools in academia but also essential in the industry, enhancing performance and efficiency across various fields.

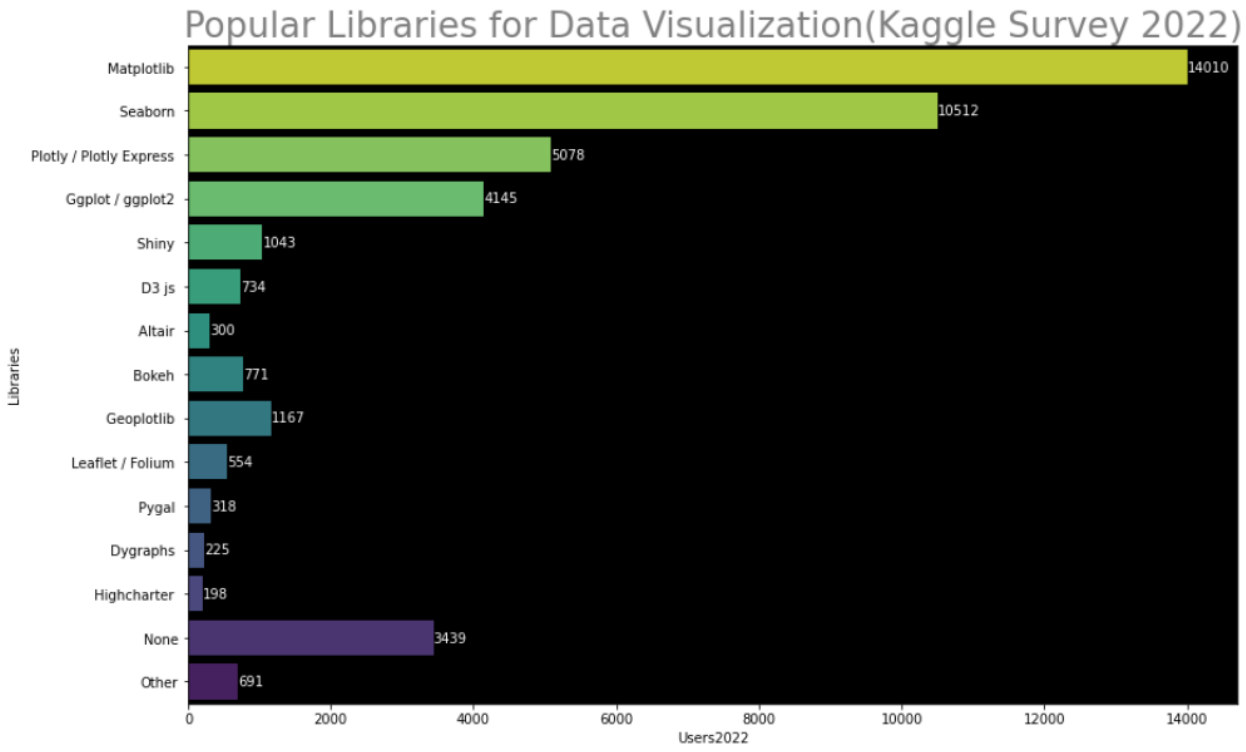
2.3 Its popularity (i.e., how many users choose this solution)

According to Kaggle's 2022 survey on popular IDEs among data scientists, RStudio ranks 5th in popularity with 3,824 participants using it. Compared to other IDEs, RStudio is more popular than Spyder (2,880 users) and Sublime Text (2,218 users), but less popular than Visual Studio Code

(9,976 users) and Jupyter Notebook (13,684 users). RStudio is an IDE specifically designed for the R language and is very popular in the data science and statistics community. Although it is not the most popular IDE, RStudio still holds a significant position in this community.



Based on a chart showing the popularity of data visualization libraries from Kaggle's 2022 survey, ggplot (or ggplot2) ranks 4th in popularity with 4,145 users. Compared to other libraries, ggplot is more popular than Shiny (1,043 users) and D3.js (734 users), but less popular than Plotly/Plotly Express (5,078 users), Seaborn (10,512 users), and Matplotlib (14,010 users). The ggplot library remains an important and popular choice in the data science community. Although it is not the most widely used library, ggplot is still a top choice for many analysts and data scientists when working with the R language. With 4,145 users in 2022, ggplot demonstrates its appeal and value in creating complex and highly customizable data visualizations.



2.4 Identify other solutions that have similar functionalities and compare them on different aspects

RStudio and Other Development Tools (IDEs) for R (ESS, Eclipse, SciViews, JGR, Tinn-R, Notepad++, và RGui): Similarities and Differences

Similarities:

- **Support for R:** All these tools provide support for working with the R programming language.
- **Basic Functionality:** All offer basic features such as code editing, running R code, and displaying results.
- **Integration with R:** They all allow deep integration with R to execute R code easily.

Differences:

➔ **Interface and User Experience:**

- **RStudio:** Has a modern, user-friendly interface, providing comprehensive support for R and Python development.
- **Jupyter Notebook:** Simple web interface that allows writing and executing code one cell at a time, supports Markdown for writing reports and annotations, suitable for experimentation and scientific research.

- **Eclipse:** A general-purpose IDE with plugins for R (like StatET), but the interface and features might differ from RStudio.
- **VSCode:** A versatile IDE with a flexible interface that can be customized and extended with many extensions. Supports debugging, Git, and multiple programming languages, suitable for diverse software development projects.
- **Tinn-R:** Simple interface, providing basic support for R programming.
- **RGui:** Default interface for R, offering basic features but lacking many conveniences of other IDEs.

➔ Features and Utilities:

- **RStudio:** Stands out with many features like integrated debugging, package management, Git integration, report generation, and statistics, with support for multiple programming languages (including Python). Can be extended with plugins and additional packages.
- **Jupyter Notebook:** An interactive environment with cells allowing flexible code writing and execution, supporting multiple languages and Markdown.
- **Eclipse (StatET):** Built on the Eclipse platform, providing general IDE features for R such as code editing, debugging, and project management.
- **VSCode:** A versatile IDE, integrating Git, supporting multiple languages, and highly extensible, suitable for diverse software development projects..
- **Tinn-R:** Simple, providing basic support for R programming, usually used for basic purposes.
- **RGui:** Default interface for R, providing basic features but lacking many conveniences of other IDEs.

➔ Popularity and Community Support:

- **RStudio:** The most popular within the R community, with a large user base, many plugins and extensions, and frequent updates.
- **Eclipse (StatET), JGR, Tinn-R:** Each tool has its own user community and plugins, but the popularity and support may not be as extensive as RStudio.
- **Jupyter Notebook:** Popular in the scientific research and data analysis community, with a wide user base and many supporting documents.
- **VSCode:** Widely used in the general programming community, with strong support from Microsoft and the development community, highly extensible with many extensions.
- **RGui:** Default interface for R, providing basic support.

➔ Choosing an IDE:

- **RStudio:** Often the top choice for many R users due to its diverse features and user-friendly interface.
- **Eclipse (StatET):** Suitable for those familiar with Emacs or Eclipse and want to integrate R into this environment.

- **Jupyter Notebook:** Suitable for experimentation, scientific research, and easy sharing of results.
- **JGR, Tinn-R:** For more complex statistical analysis and calculations.
- **VSCode:** Ideal for developers needing a versatile, highly customizable IDE that supports many programming languages.
- **RGui:** For users using R in a basic manner without needing advanced IDE features.Depending on individual needs and preferences, users can select a suitable tool for developing and working with the R programming language.

Depending on individual needs and preferences, each person can choose a suitable tool to develop and work with the R programming language.

The following is a comparison table between Ggplot and other solutions that have similar functionalities with different aspects

Aspect	**ggplot2**	**Lattice**	**plotly**	**highcharter**	**leaflet**	**rgl**
Programming Language	R	R	R (support Python, Julia, JavaScript)	R	R	R
Design Philosophy	Grammar of Graphics	Trellis Graphics	Create interactive charts and data visualizations easily	Based on the Highcharts JS library	Based on the viện Leaflet JS library	Provides 3D graphics capabilities
Capability	Complex, highly customizable statistical charts	Grouping and multivariate plots	Interactive charts, support 2D and 3D plotting	Rich and beautiful interactive charts	Beautiful interactive map	Powerful interactive 3D graphs
Features	Supports custom classes and themes, integrates many extension packs	Create complex graphs with concise syntax	Visual interaction, zoom, pan, hover, online chart publishing	Supports many chart types, especially financial charts	Supports map layers, markers, popups	Supports rotation, zoom, and other interactive operations on 3D graphs
Usability	It takes time to get used to the syntax	Easy to use for clustering and multivariate charts	Easy to use with Plotly Express, highly interactive	Create beautiful interactive charts with simple syntax	Easy to use for creating interactive maps	Need to configure the graphics system properly, may not be easy to use for beginners
Community	Huge in the R	Big in the R community	Growing in the data science and	Widely used, especially in	Huge in the web	Widely used in the R

	community	but not as popular as ggplot2	machine learning communities	web applications	programming and GIS community	community
Documentation	In-depth books, many articles and blogs, rich official documents	Official documentation and many examples online	Rich official documentation, many examples and tutorials online	Official documentation and many examples	Rich official documentation, many examples and extensive plugins	Official documentation and many examples
Interoperability	Limited, mostly static	Limited, mostly static	High, interactive visualization support	High, rich interactive chart generation	Very high, especially with maps and spatial data	High, strong interactive support with 3D graphs
Summary	Ideal for complex statistical graphs in R with a clear structure according to the Grammar of Graphics.	Good for clustering and multivariate plots in R, but the syntax can be confusing.	Excellent for interactive charts in R, easy to use with online sharing capabilities.	Create beautiful interactive charts with simple syntax, based on Highcharts JS.	Create easy and beautiful interactive maps, suitable for spatial data.	Create powerful interactive 3D graphs in R, suitable for applications requiring 3D graphics.

3. Demonstration

No.	Topic	Description
1	Introduction to RStudio and ggplot2	**Overview**: ● **RStudio**: A comprehensive IDE for R, offering features for coding, debugging, and data visualization. ● **ggplot2**: A powerful and flexible visualization package in R, designed for creating complex and multi-layered graphics. **Key Features**: ● **Data Handling**: Efficient management of large datasets with packages like **dplyr** and **data.table**. ● **Visualization**: Creating a wide range of static and interactive plots. **Scalability and Flexibility**: ● **RStudio**: Supports integration with big data technologies like Spark through sparklyr.

		• **ggplot2**: Allows dynamic and customizable visualizations, adapting to diverse data needs.
2	Installation and Setup	**R & RStudio Installation**: A step-by-step guide for installing R & RStudio on various operating systems. **Installing ggplot2**: Instructions to install ggplot2 using `install.packages()`.
3	Basic Syntax and First Plot	• **Introduction to R Syntax**: Basic R syntax and data structures. • **RStudio Interface**: Overview of the RStudio interface and its main components. • **Basic Functions**: Using RStudio for coding, debugging, and viewing plots.

4	ggplot2 Basics	• **Data and Aesthetics:**
		  ○ Data: Using tidy data format.
		  ○ Aesthetics: Mapping data to visual properties.
		  ○ *Example*: Creating a simple scatter plot.
		• **Layers:**
		  ○ Geometries: Different types of geoms like `geom_point()`, `geom_line()`.
		  ○ Statistics: Statistical transformations.
		  ○ Positions: Position adjustments.
		• **Scales:**
		  ○ Scales: Customizing scales, breaks, and labels.
		• **Facets:**
		  ○ Faceting: Creating subplots using `facet_wrap()` and `facet_grid()`.
		• **Coordinates:**
		  ○ Coordinate Systems: Cartesian, polar, and other coordinate systems.
		• **Themes:**
		  ○ Themes: Customizing the overall appearance of plots using themes.

5	Advanced Plotting Techniques	• **Multi-layered Plots**: Combining multiple types of plots in a single graphic. • **Faceting**: Splitting data into subplots for comparative analysis. • **Customization**: Detailed customization options for themes, colors, and annotations. • **Interactive Plots**: Integrating ggplot2 with interactive visualization packages like plotly

6	Solving Data Growth Problems	• **Data Import and Export:** Reading data from various formats (CSV, Excel, databases) and writing outputs. • **Data Manipulation:** Using **dplyr** for data wrangling tasks like filtering, summarizing, and joining datasets. • **Big Data Handling:** Strategies for managing and analyzing large datasets using **data.table** and **sparklyr**. • **Performance Optimization:** Tips for improving the performance of R scripts and visualizations. • **Scalability:** Demonstrating how RStudio's integration with big data tools ensures scalability and efficiency.
6	Case Study: Real-world Application	• **Case Study**: Analyzing the Corona Virus Report dataset to identify trends. • **Data Import**: Importing the dataset into RStudio. • **Data Cleaning**: Using basic R functions to clean and prepare the data. • **Visualization**: Creating a series of plots with ggplot2 to visualize trends over time. • **Analysis**: Interpreting the visualizations to draw meaningful insights. • **Performance Considerations**: Ensuring the analysis and visualizations remain performant as data volume grows.
7	Conclusion	• **Recap**: Summarize the key points covered in the demonstration, including the installation, basic syntax, advanced techniques, and case study. • **Benefits**: Highlight the benefits of using RStudio and ggplot2 for handling and visualizing large datasets. • **Future Directions**: Discuss future developments and enhancements in RStudio and ggplot2 for big data applications. • **Final Thoughts**: Conclude with thoughts on the importance of effective data management and visualization in today's data-driven world.

4. Discussions and Conclusion

Weekly schedule:

	Week 01	Week 02	Week 03	Week 04

<table>
<tr><td></td><td>24/06 - 30/06</td><td>01/07 - 07/07</td><td>08/07 - 14/07</td><td>15/07 - 21/07</td></tr>
<tr><td></td><td>Getting familiar with RStudio & ggplot2</td><td>Study in depth the topics in the ggplot2-book</td><td>Report findings and combine research into the demo</td><td>Final Report & Prepare presentation</td></tr>
<tr><td>Bình</td><td rowspan="4">Install RStudio
Explore and study how to use the RStudio interface
Study and get familiar with R language, ggplot2</td><td>Research about topic Layers</td><td rowspan="4">Report on the researched topic and the difficulties encountered
Incorporate into the demo each topic on the selected dataset</td><td rowspan="4">Resolve difficulties (if any)
Final report
Create slides
Practice presentation</td></tr>
<tr><td>Huy</td><td>Research about R language, RStudio interface and Advanced topics of ggplot2</td></tr>
<tr><td>Mai</td><td>Research about topic Scales</td></tr>
<tr><td>Oanh</td><td>Research about topic Grammar</td></tr>
</table>

Group self-evaluation: State the objective and subjective reasons that may affect your performance in completing this research; however, it is not about evaluating the contribution of team members.

- **Advantages:**
  - All members are students majoring in data science, so they have certain advantages when working with data, performing operations, and analyzing data. They are already familiar with some types of visual charts and graphs.
  - **R Programming Language**:
    - **Comprehensive Statistical Analysis**: R provides a wide range of statistical and graphical techniques, making it ideal for data analysis.
    - **Open Source**: R is free to use, with a large, active community contributing packages and tools.

        - **Data Handling and Storage**: Efficiently handles and processes large datasets.
    - **RStudio**:
        - **User-Friendly Interface**: RStudio offers an integrated development environment (IDE) with helpful features like syntax highlighting, code completion, and debugging tools.
        - **Project Management**: Facilitates easy organization and management of multiple files and projects.
        - **Integration with RMarkdown**: Simplifies the creation of dynamic reports and reproducible research documents.
    - **ggplot2**:
        - **Powerful Data Visualization**: Based on the Grammar of Graphics, ggplot2 allows for the creation of complex, customizable, and high-quality visualizations.
        - **Consistent and Logical Syntax**: Placing components helps users easily produce a wide range of plots.
        - **Comprehensive Documentation**: Extensive guides and examples available online to assist in learning and troubleshooting.
- **Disadvantages:**

    - Our group has learned and used Python quite a lot and has never learned and used RStudio or ggplot2, so it will take time to learn how to use it.

    - **R Programming Language**:
        - **Performance Issues**: R can be slower compared to other programming languages like Python when handling extremely large datasets.
        - **Learning Curve**: R has a steep learning curve for beginners, especially for those without a programming background.
        - **Memory Management**: Less efficient memory usage, which can be a limitation with very large datasets.
    - **RStudio**:
        - **Resource Intensive**: RStudio can be demanding on system resources, potentially slowing down performance on less powerful machines.

- **Complexity for New Users**: While feature-rich, RStudio's extensive options and settings may overwhelm new users.

- **ggplot2**:
  - **Initial Complexity**: Understanding the Grammar of Graphics and effectively using ggplot2 can be challenging for new users.
  - **Customization Limitations**: Some very specific customizations may require advanced knowledge or additional packages.
  - **Dependence on Tidy Data**: ggplot2 works best with tidy data, requiring preprocessing and data cleaning, which can be time-consuming.