# Visualization of Cyber Attacks

## Data Visualization - Final Project

Dániel Cser
Héctor Flores

# Table of Contents

**Table of Contents** **2**
**Disclaimer** **3**
**Problem Characterization** **4**
**Abstractions** **5**
- Data Abstraction 5
  - Attributes and Characteristics 5
  - Cardinality and Ranges 6
  - Data Transformations 6
- Task Abstraction 6
  - Why? (Actions and Goals of Visualization) 6
  - What? (Targets of Analysis) 7
  - How? (Design and Interactions) 7
  - Mapping to Analyst Questions 7

**Visualization and Interaction** **9**
- Main Page 9
- Geographical Patterns 9
- Temporal Trends 10
- Payload Size and Confidence Levels 11

**Implementation** **14**
**Instructions** **16**
- Method 1: Running Locally on Your System 16
- Method 2: Accessing the Hosted Application 17
- Development Environment Compatibility 17
- Using the Application 17

**References** **17**

## Disclaimer

Throughout the semester, we faced significant challenges due to the lack of participation from certain team members. Midway through the semester, one member announced their decision to leave the group after contributing nothing to the project. Following this, we redistributed the workload among the remaining three members to allow for flexible scheduling and individual progress.

However, another member continued to assure us of their participation without making any contributions. Just a week before the final deadline, this individual also announced their departure, leaving critical tasks incomplete. This situation was further compounded by the overlap of project deadlines with exam preparations.

Despite these obstacles, we managed to deliver a cohesive project. While the outcome could not meet its full potential under these circumstances, we are confident in the quality of our work. With a fully committed team from the outset, we believe this project could have been elevated to the next level.

## Problem Characterization

Cybersecurity has become a critical challenge in the digital age, with organizations and individuals increasingly vulnerable to cyberattacks. Detecting and analyzing these attacks efficiently is vital to mitigating damage, safeguarding sensitive information, and ensuring system resilience. With the proliferation of cyber threats and the diversity of attack vectors, visual analytics can play a pivotal role in uncovering hidden patterns, trends, and anomalies in data.

The **Cyberattacks Detection dataset** serves as a comprehensive resource for exploring this domain. It includes 100,000 rows, each representing a unique cyberattack event. The dataset captures a variety of attributes, such as attack types, protocols, affected systems, and other key factors, making it ideal for developing machine learning models and visualization tools to study cyberattacks.

To effectively analyze this dataset, a series of questions can guide the design of a visual analytics tool. These questions are tailored to leverage visualization techniques, which are more effective than automatic queries for identifying complex patterns, correlations, and temporal trends:

1. **Geographical Patterns**: Are there distinct geographical patterns in the occurrence of cyberattacks based on attributes such as attack type, protocol, or targeted system? This analysis can help identify regional vulnerabilities and inform targeted defense strategies.

2. **Temporal Trends**: Are there specific days of the week or hours when certain types of attacks or targeted systems exhibit notable trends? Temporal visualizations can highlight periods of heightened activity, enabling proactive measures.

3. **Payload Size and Confidence Levels**: How does the payload size influence the distribution of confidence levels in machine learning models when classifying attacks or identifying affected systems? Visualizing these relationships can aid in refining detection algorithms and understanding critical thresholds.

By answering these questions through visual analytics, analysts can uncover actionable insights, identify vulnerabilities, and develop more effective detection and prevention mechanisms. This approach underscores the critical role of visualization in making sense of complex and high-dimensional data in the domain of cybersecurity.

# Abstractions

The **Data and Task Abstraction** chapter lays the foundation for understanding the data's structure and the specific analytical tasks that the visual analytics tool aims to address. This process involves systematically identifying the characteristics of the dataset and aligning them with the objectives and methods of analysis.

## Data Abstraction

The dataset, structured as a **table**, contains diverse attributes that capture essential aspects of cyberattacks. Each row represents a unique attack event, and the attributes include both categorical and quantitative data types, as well as temporal information.

### Attributes and Characteristics

The table below summarizes the key attributes:

| **Attribute** | **Type** | **Number of categories / range** |
|---|---|---|
| Attack ID | categorical | 100 000 (= number of items) |
| Timestamp | ordered: ordinal | from 1/1/2024 0:22 to 9/9/2024 9:18 |
| Source IP | categorical | 99 951 |
| Destination IP | categorical | 99 946 |
| Source Country | categorical | 11 |
| Destination Country | categorical | 11 |
| Protocol | categorical | 4 |
| Source Port | categorical | 51 222 |
| Destination Port | categorical | 51 359 |
| Port Type | categorical | 11 |
| Attack Type | categorical | 21 |
| Payload Size (bytes) | ordered: quantitative | from 1 to 4999 |
| Detection Label | categorical | 3 |

| | | |
|---|---|---|
| Confidence Score | ordered: quantitative | from 0 to 1 |
| ML Model | categorical | 6 |
| Affected System | categorical | 10 |

Table 1 - Attribute characteristics

This dataset encompasses a rich variety of dimensions, such as geographical locations, temporal information, network-level details (e.g., ports and protocols), and attack-specific metrics (e.g., payload size and confidence score).

### Cardinality and Ranges

The dataset features high cardinality in attributes like IP addresses and ports, while attributes like countries, protocols, and detection labels have relatively low cardinality, facilitating straightforward categorization. Quantitative attributes, such as payload size and confidence score, have defined ranges that allow for effective scaling and normalization if required.

### Data Transformations

At this stage, no significant data transformations appear necessary, apart from the removal of na and null values has been conducted as a necessary step of preparation. The dataset's structure and attributes align well with the analytical goals. However, future refinements, such as binning payload sizes or discretizing confidence scores, may enhance specific visualizations.

## Task Abstraction

The task abstraction process identifies the analytical goals and defines the methods for achieving them through visualization. This involves specifying the "Why?" (actions and goals) and "What?" (targets of analysis) for the visual analytics tool, as well as determining the "How?" (design and interactions).

### Why? (Actions and Goals of Visualization)

The primary purpose of visualization is to discover new knowledge by identifying patterns, trends, and dependencies within datasets. Analysts use visualization tools to explore the geographical, temporal, and network-level characteristics of cyberattacks, enabling the generation and validation of hypotheses through interactive exploration. Additionally, visualization aims to present information effectively, highlighting key findings such as trends, clusters, and anomalies to support informed decision-making. To achieve this, clear and intuitive visual encodings are essential for conveying complex relationships in a comprehensible manner.

### What? (Targets of Analysis)

The analysis focuses on several key targets to better understand cyberattacks and their implications. Trends and patterns are examined by analyzing geographical distributions based on attributes such as attack type, protocol, and targeted systems. This helps identify regional vulnerabilities and potential hotspots. Outliers and anomalies, such as unusual payload sizes or confidence levels, are also critical to detect, as they may reveal unique attack vectors or atypical behaviors. Temporal attributes are another area of interest, with analysts studying attack patterns tied to specific days or times to inform proactive defensive strategies.

### How? (Design and Interactions)

Effective visualization relies on thoughtful design and interactions to facilitate analysis and decision-making. **Visual Encoding** employs maps with heatmaps or cluster visualizations to represent geographical distributions, while boxplots are used to depict payload sizes and confidence scores.

To focus analysis, **Filtering and Aggregation** enable users to refine datasets by timeframes, protocols, or targeted systems, while aggregating payload and confidence data provides high-level summaries. **Interactivity** is crucial, allowing users to drill down into specific events or anomalies for detailed investigations and to select regions, time intervals, or system types to access details-on-demand.

Finally, **Context and Detail Combination** ensures a comprehensive view by embedding global trends, such as geographical or temporal patterns, alongside focused views of specific attack types. This combination enhances understanding by providing both macro and micro perspectives in the same interface.

### Mapping to Analyst Questions

The design of the visualization tool directly addresses the core questions posed by analysts, ensuring alignment with their investigative needs. **Geographical patterns** are represented using maps with overlays for attack types, protocols, and targeted systems, while clustering techniques highlight regional hotspots and vulnerabilities. **Payload versus confidence levels** are visualized through boxplots, with color-coded markers representing different machine learning algorithms, enabling analysts to uncover trends.

**Temporal patterns** are explored using tile-based temporal diagrams, filtered by weekdays and hours, to reveal time-based trends and inform proactive defense strategies.

By abstracting data and tasks into meaningful visual representations, the tool is designed to uncover actionable insights, equipping cybersecurity analysts with the information needed for effective decision-making and improved threat mitigation.

# Visualization and Interaction

This section provides an overview of the application's structure, focusing on the chosen visualizations and their relevance to addressing the analysis questions.

## Main Page

The main page of the application serves as an introduction to the project, outlining the primary analytical questions that the visualization tool is designed to address. This introductory section offers users a clear understanding of the tool's objectives and the broader context of the project. Once familiarized with the purpose and goals of the application, analysts can seamlessly transition to exploring the dataset through the thoughtfully designed visualizations, enabling them to effectively carry out their analysis with ease and efficiency.

## Geographical Patterns

The primary analytical question concerns identifying geographical patterns based on attack type, protocol, or targeted system. A choropleth map was determined to be the most effective visualization for this purpose, given the nature of the data.

This visualization offers an intuitive starting point for exploration, enabling users to quickly gain a geographical perspective. It clearly distinguishes countries included in the dataset and highlights their roles as either the origins or destinations of cyberattacks.

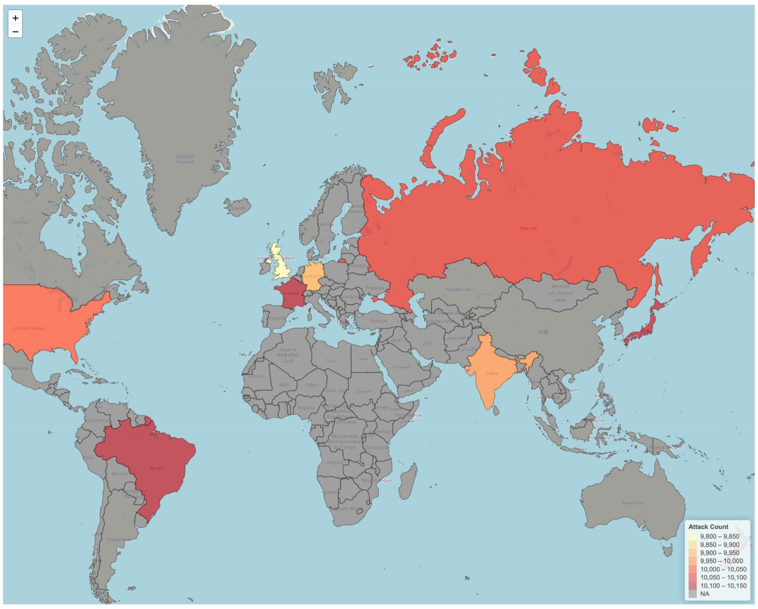


Figure 1 - choropleth map

To enhance usability, this visualization includes advanced filtering options. Users can filter by attack type, protocol, and targeted systems, allowing for both broad overviews and specific analyses. For example, users can investigate which countries experienced DDoS attacks targeting IoT devices via UDP. In such a scenario, the United Kingdom emerges as a key affected region.

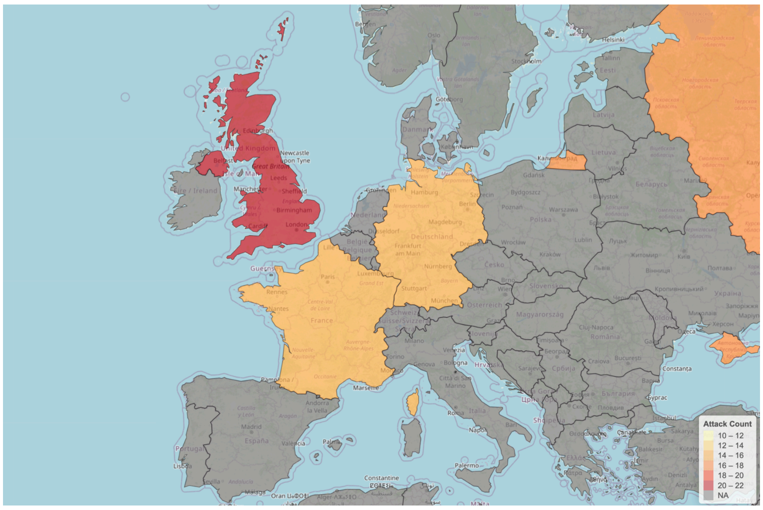
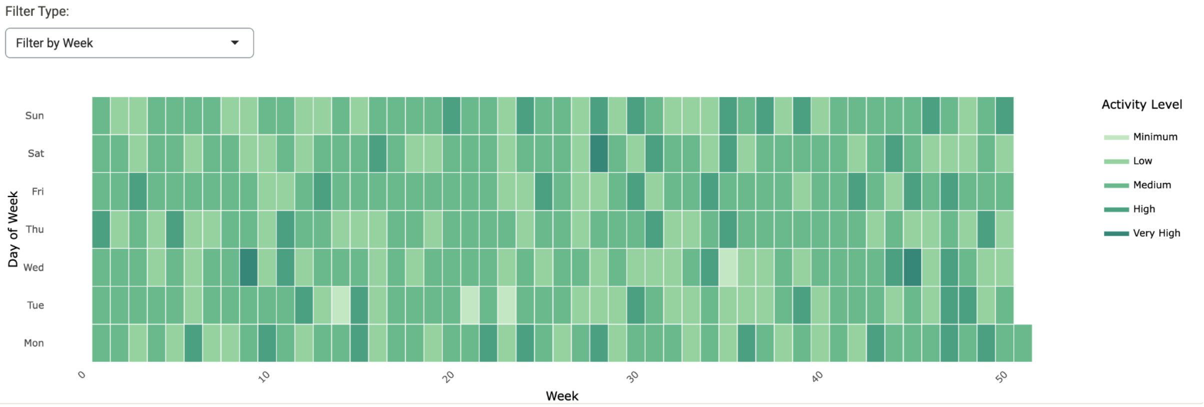


Figure 2 - IoT Devices by DDoS by UDP

As illustrated in *Figure 2*, the choropleth map effectively conveys categorical attributes of the dataset, providing a clear and detailed understanding of the data's geographical distribution and associated patterns.

## Temporal Trends

Understanding temporal patterns in cyberattacks can provide valuable insights into recurring trends, such as specific days or hours when certain types of attacks or targeted systems are more prevalent. To address this analytical question, we implemented a tile-based temporal diagram, a visualization technique commonly used on platforms like GitHub to display activity frequency over time. This approach effectively highlights trends in cyberattack occurrences.

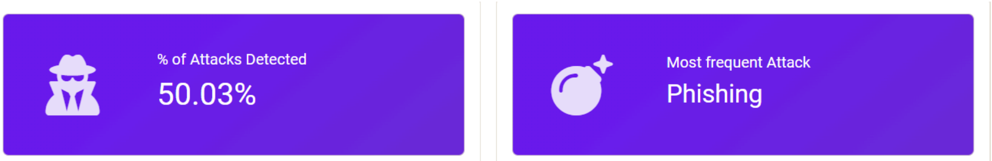


Filter Type: Filter by Week

Figure 3 - Tile-Based Temporal Diagram

The diagram enables users to explore temporal trends by adjusting the time interval and filtering data to focus on specific periods. For example, analysts can examine patterns such as the peak occurrence of cyberattacks in 2024 between 4 and 5 a.m., providing valuable insights into critical times for implementing mitigation strategies.

In addition to the tile diagram, this section of the analytical tool integrates statistical enhancements that offer further context and deeper insights into the data. One of these enhancements is the detection rate, which calculates in real time the percentage of cyberattacks detected within the selected time interval. Another feature is the analysis of attack type frequency, which summarizes the most commonly used attack methods, allowing analysts to quickly identify trends in attack techniques.



% of Attacks Detected
50.03%

Most frequent Attack
Phishing

Figure 4 - Supporting Statistical Depictions

These supplementary features enable users to delve deeper into the data, correlating temporal patterns with specific attack types or protocols. For example, if DDoS attacks show a significant spike during a specific interval, analysts can investigate related system vulnerabilities or mitigation strategies tailored to that timeframe.

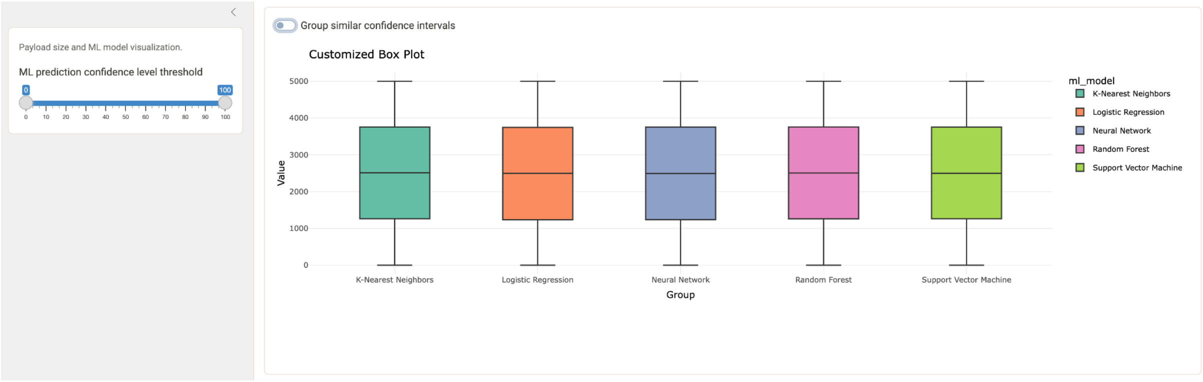
By combining the visual clarity of the tile-based temporal diagram with statistical summaries, this section of the tool equips analysts with a comprehensive view of time-based cyberattack trends, enhancing their ability to draw actionable conclusions.

## Payload Size and Confidence Levels

Our final visualization focuses on examining whether the payload size plays a significant role in the distribution of confidence levels when classifying cyberattacks using machine learning (ML) models. This question is particularly relevant for understanding the behavior of classification models in identifying attacks and for optimizing detection strategies. To address this, we designed a visualization that combines statistical insights with advanced clustering techniques.

To begin, we constructed a boxplot to depict the distribution of payload sizes for each ML algorithm across specified confidence level intervals. This visualization provides a clear representation of the variability and central tendency of payload sizes associated with varying confidence levels for different algorithms.

By presenting the payload size distribution in this manner, we allow analysts to observe whether specific algorithms produce distinct patterns in their classification performance related to payload size. For instance, it becomes immediately apparent if larger payloads are consistently associated with higher confidence levels or if certain algorithms perform better with specific payload sizes.



Payload size and ML model visualization.

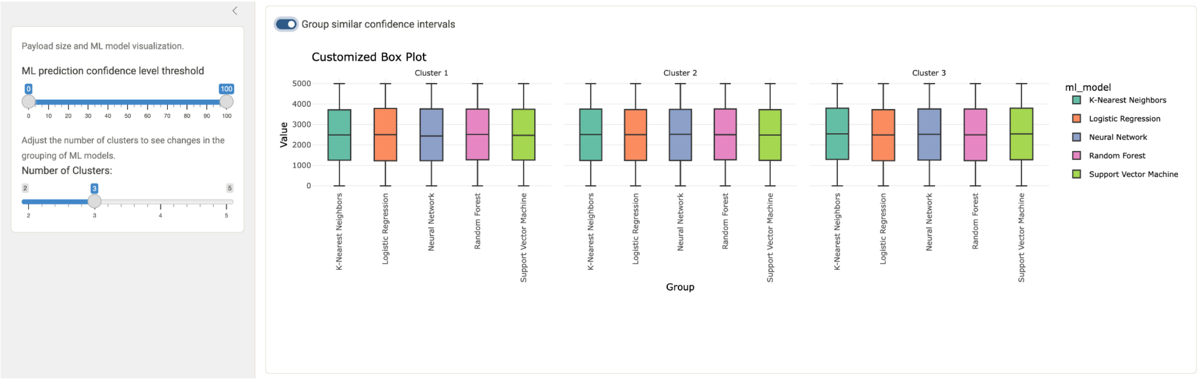
ML prediction confidence level threshold: 0 – 100

○ Group similar confidence intervals

Figure 5 - Boxplot of payload sizes

To further enrich the analysis, we implemented a clustering algorithm to group data points based on their confidence intervals and payload sizes. By applying clustering with a varying number of clusters, analysts can explore how the payload sizes are distributed within distinct confidence level groups. This additional layer of analysis provides a more granular understanding of the relationship between payload size and model confidence.

For illustration, we present a second boxplot where the data is clustered into three groups based on confidence levels. This visualization enables a deeper examination of how payload sizes are distributed within low, medium, and high-confidence classifications. It also highlights patterns that may not be immediately apparent in the unclustered data, such as whether certain confidence intervals correspond to payloads of a particular size range.



Payload size and ML model visualization.

ML prediction confidence level threshold: 0 – 100

Adjust the number of clusters to see changes in the grouping of ML models.

Number of Clusters: 3

● Group similar confidence intervals

Figure 6 - Boxplot of payload sizes clustered into three groups

By combining boxplots with clustering, our visualization equips analysts with both high-level overviews and detailed breakdowns of the data. This dual perspective can inform decisions on selecting the most suitable ML models for cyberattack detection based on payload size characteristics and confidence level performance.

In conclusion, this final visualization not only deepens our understanding of how payload size impacts ML classification confidence but also exemplifies how integrating statistical and clustering approaches can enhance analytical capabilities. It demonstrates a practical and effective method for visualizing complex relationships in cybersecurity datasets, providing valuable tools for model evaluation and refinement.

# Implementation

The implementation of this project was carried out using **RStudio** in combination with the **Shiny package**, which provides an interactive platform for developing web applications. The project structure is modular and consists of three primary files: *global.R*, *server.R*, and *ui.R*. Additionally, a subdirectory contains supplementary files that handle specific visualizations. This structure enhances the clarity, reusability, and scalability of the codebase.

The *global.R* file serves as the foundational layer of the project. It initializes the global environment, loads essential libraries, imports datasets, and defines utility functions. This file establishes a centralized setup for resources that are shared across the application.

The file imports various libraries, including *shiny*, *dplyr*, *ggplot2*, and *cluster* among others, which provide the necessary tools for data manipulation, visualization, and geospatial data handling. It also loads multiple datasets that are critical to the project. Geographic data is imported from a GeoJSON file (*50m.geojson*) to define polygon shapes for the map. A supplementary dataset (*countries_codes_and_coordinates.csv*) is used to ensure consistency between country names and the geographic dataset by mapping different naming conventions. The primary dataset (*cyberattacks_detection.csv*), which contains details of cyberattacks, is also imported. The column names in this dataset are standardized to snake_case to ensure uniform referencing.

A customized UI theme is defined using the *bs_theme* function, which enhances the aesthetic appeal of the application. External links to resources, such as Shiny's GitHub page, are also included to provide additional reference material for users.

The file defines several utility functions. The *calculate_geojson_data* function enriches the geographic data by aggregating attack counts and merging this information with the GeoJSON dataset. The *render_heatmap* function generates heatmaps based on input data, ensuring standardized visualization styling and behavior. Additionally, helper functions like *determine_filter_expr* streamline the filtering process by interpreting user selections and dynamically updating filters in the application.

The *server.R* file manages the backend logic of the Shiny application. It processes user inputs, manipulates data in real-time, and dynamically renders visualizations.

The server logic initializes a Leaflet map, which serves as a base layer for interactive geographic visualizations. The map allows users to zoom, pan, and focus on relevant areas using a bounding box. User-defined filters for attack types, protocols, and targeted systems are managed using reactive variables and observers. These filters dynamically update the displayed data as user selections change.

The server logic aggregates attack data, applies user-defined filters, and updates the map visualization accordingly. A color-coded legend is included to highlight the severity of attacks across different regions. For temporal analysis, the server processes timestamped attack data to generate dynamic heatmaps. These heatmaps reveal patterns based on the day of the week or the hour of the day, and tooltips provide detailed insights into the data.

The *ui.R* file defines the layout and user interface of the application. It organizes visual components and ensures a seamless user experience.

The global layout includes a navigation panel that allows users to switch between different visualizations and a well-panel that displays the application title. Three distinct panels are included, each dedicated to a specific visualization. The first panel analyzes geographic patterns in attacks, enabling filtering by type, protocol, or targeted system. The second panel focuses on temporal trends, offering insights into the timing of attacks. The third panel explores the relationship between payload size and the confidence levels of machine learning models in classifying attacks.

Sidebars are used to present filtering options and contextual information for each visualization. These sidebars provide users with intuitive controls for interaction. Additionally, external links and resources are integrated into the UI, allowing users to access further information about Shiny and related tools.

This modular design, combined with the interactive capabilities of Shiny, ensures that the application is user-friendly and robust. It effectively addresses the project’s objectives while providing a clear and intuitive interface for end users.

# Instructions

The application offers two primary methods for usage, each catering to different preferences and technical setups. In this updated version, additional tools and enhancements have been introduced to simplify dependency management and development environment setup.

## Method 1: Running Locally on Your System

To run the application locally, you need to set up the necessary environment and install the required dependencies. The project leverages the *renv* package for dependency management, which streamlines the process of ensuring all required libraries are installed. Follow these steps in your **R-console** to get started:

1. Inside the **final_project** folder, install the **renv** package:

```
install.packages("renv")
```

2. Initialize the **renv** environment:

```
renv::init()
```

3. Restore and install any missing dependencies:

```
renv::restore()
```

The **renv** package will automatically identify and install all the necessary dependencies for the project.

Once the required dependencies are installed, you can launch the application directly from an R console using the following command:

```
shiny::runApp('final_delivery')
```

This method provides the flexibility to work with the full codebase locally, enabling customization, debugging, and deeper exploration of the application.

### Method 2: Accessing the Hosted Application

If you prefer not to set up the application locally, you can access the hosted version online. This method eliminates the need for installation or configuration. Simply visit the following link to use the application:

https://frhec.shinyapps.io/final_delivery/

The hosted version offers the same functionalities as the local version, ensuring a seamless experience for users who prioritize convenience.

### Development Environment Compatibility

The project is also compatible with containerized development environments, making it easier to set up and manage. Specifically, the project is **Devcontainer friendly**. This means you can use tools like **Devpod** (https://devpod.sh/docs/what-is-devpod) or refer to **Containers.dev** (https://containers.dev/) to set up a standardized and portable development environment. These tools streamline the process, allowing you to start working on the project without worrying about discrepancies in your local setup.

### Using the Application

Once the application is up and running—whether locally or online—you can explore its rich set of visualizations and interactive features. The app allows you to experiment with various filters and delve into the data to address the analytical questions posed within the project. This hands-on approach provides an engaging way to uncover insights and fully utilize the capabilities of the application.

## References

| | |
|---|---|
| App | https://frhec.shinyapps.io/final_delivery/ |
| Main dataset | https://www.kaggle.com/datasets/lastman0800/cyberattacks-detection |
| 50m.geojson | https://github.com/eparker12/nCoV_tracker/blob/master/input_data/50m.geojson |
| countries_codes_and_coordinates.csv | https://gist.github.com/tadast/8827699 |