



UNIVERSITY
OF WARSAW



Feluda LIDAr

Enhancing Data Exploration Efficiency: Developing an Automated EDA Tool with ShinyApp in R

Presenting By:

Porimol Chandro

p.chandro@student.uw.edu.pl

Shokoufe Naseri

s.naseri@student.uw.edu.pl

Ho Thi Hoang Nhu

t.ho2@student.uw.edu.pl

1st Year Master's Student of Data Science and Business Analytics

Faculty of Economic Sciences, UW

Agenda

- Brief introduction of the project
- Shiny dashboard
- R package development
- ChatBot based on Llama
- OOP in R (R6)
- RCPP implementation
- Exploratory data analysis
- Function vectorisation in R
- Defensive programming in R
- Data visualization
- Conclusion
- Reference
- Appendix

Feluda Analytica

Dashboard

EDA

Project Title: Enhancing Data Exploration Efficiency: Developing an Automated EDA Tool with ShinyApp in R.

Contributors:

Fullname	Student.ID	Email
Porimol Chandro	468264	p.chandro@student.uw.edu.pl
Shokoufe Naseri	466750	s.naseri@student.uw.edu.pl
Ho Thi Hoang Nhu	466503	t.ho2@student.uw.edu.pl

Note: Every contributor makes an equal contribution.

Introduction

Exploratory Data Analysis (EDA) is one of the crucial parts of Data Science and Machine Learning projects for understanding the datasets better to detect anomalies, and guiding feature engineering in data science projects.

Developing an automated EDA tool using ShinyApp in R offers efficiency, interactivity, reproducibility, scalability, and customization benefits. Automating the EDA will speed up the analysis, enhance the developer experience, and ensure EDA consistency across projects. Interactive interfaces facilitate deeper exploration of complex datasets and promote transparency in analyses. Using automated EDA tools can be tailored to meet specific project requirements and developer preferences. By focusing on EDA and automation, data scientists can derive insights more efficiently and make better-informed decisions for business.

Motivation

EDA is the foundation of effective data analysis and communication based on data. However, creating compelling and informative visualizations can be a time-consuming and skill-intensive process.

Our project aims to streamline the visualization creation process by leveraging the power of large language models (LLMs). Drawing inspiration from the research presented in 'LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models' (Dibia, 2023), we aim to develop a project that will transform natural language descriptions into tailored R EDA.

In summary, prioritizing EDA and developing automated tools using ShinyApp in R leads to more effective data analysis and facilitates better decision-making in various domains.

References

[1] Dibia, V. (2023). Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. arXiv preprint arXiv:2303.02927 .

[2] Touvron, Hugo, et al. 'Llama: Open and efficient foundation language models.' arXiv preprint arXiv:2302.13971 (2023) .

[3] 'Ollama (Version v0.1.38)' [Computer software]. <https://github.com/ollama/ollama> .

[4] Maria Kubara and Ewa Weychert (2024). 'Advanced Programming in R'. <https://usosweb.wne.uw.edu.pl/kontroler.php>

Project Title: Enhancing Data Exploration Efficiency: Developing an Automated EDA Tool with ShinyApp in R.

Contributors:

Fullname	Student.ID	Email
Porimol Chandro	468264	p.chandro@student.uw.edu.pl
Shokoufe Naseri	466750	s.naseri@student.uw.edu.pl
Ho Thi Hoang Nhu	466503	t.ho2@student.uw.edu.pl

Note: Every contributor makes an equal contribution.

Introduction

Exploratory Data Analysis (EDA) is one of the crucial parts of Data Science and Machine Learning projects for understanding the datasets better to detect anomalies, and guiding feature engineering in data science projects.

Developing an automated EDA tool using ShinyApp in R offers efficiency, interactivity, reproducibility, scalability, and customization benefits. Automating the EDA will speed up the analysis, enhance the developer experience, and ensure EDA consistency across projects. Interactive interfaces facilitate deeper exploration of complex datasets and promote transparency in analyses. Using automated EDA tools can be tailored to meet specific project requirements and developer preferences. By focusing on EDA and automation, data scientists can derive insights more efficiently and make better-informed decisions for business.

Motivation

EDA is the foundation of effective data analysis and communication based on data. However, creating compelling and informative visualizations can be a time-consuming and skill-intensive process.

Our project aims to streamline the visualization creation process by leveraging the power of large language models (LLMs). Drawing inspiration from the research presented in 'LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models' (Dibia, 2023), we aim to develop a project that will transform natural language descriptions into tailored R EDA.

In summary, prioritizing EDA and developing automated tools using ShinyApp in R leads to more effective data analysis and facilitates better decision-making in various domains.

References

[1] Dibia, V. (2023). Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. arXiv preprint arXiv:2303.02927 .

[2] Touvron, Hugo, et al. 'Llama: Open and efficient foundation language models.' arXiv preprint arXiv:2302.13971 (2023) .

[3] 'Ollama (Version v0.1.38)' [Computer software]. <https://github.com/ollama/ollama> .

[4] Maria Kubara and Ewa Weychert (2024). 'Advanced Programming in R'. <https://usosweb.wne.uw.edu.pl/kontroler.php?>

Brief introduction of the project

- **Efficiency and Automation:** Developing an automated EDA tool using ShinyApp in R enhances efficiency by speeding up the analysis process and ensuring consistency across projects.
- **Interactivity and Exploration:** Interactive interfaces in automated EDA tools facilitate deeper exploration of complex datasets and promote transparency in analyses.
- **Chatbot:** Develop a chatbot based on open source LLM (Llama3) and Ollama.

Shiny dashboard [cont.]



Shiny dashboard

Feluda Analytica

Dashboard

EDA

Data Uploader

EDA Summary

Visualization

Chat with Llama

Uplaod a CSV file!

Please upload a CSV file to perform analysis

Browse...

cps_small.csv

Upload complete

Uploaded dataset as tabular format

Show 10 entries

Search:

	wage	educ	exper	female	black	white	midwest	south	west
1	2.03	13	2	1	0	1	0	1	0
2	2.0699999	12	7	0	0	1	1	0	0
3	2.1199999	12	35	0	0	1	0	1	0
4	2.54	16	20	1	0	1	0	1	0
5	2.6800001	12	24	1	0	1	0	1	0
6	3.0899999	13	4	0	0	1	0	1	0
7	3.1600001	13	1	0	0	1	0	0	1
8	3.1700001	12	22	1	0	1	0	1	0
9	3.2	12	23	0	0	1	0	1	0
10	3.27	12	4	1	0	1	0	0	1

Showing 1 to 10 of 1,000 entries

Previous12345...100Next

R package development

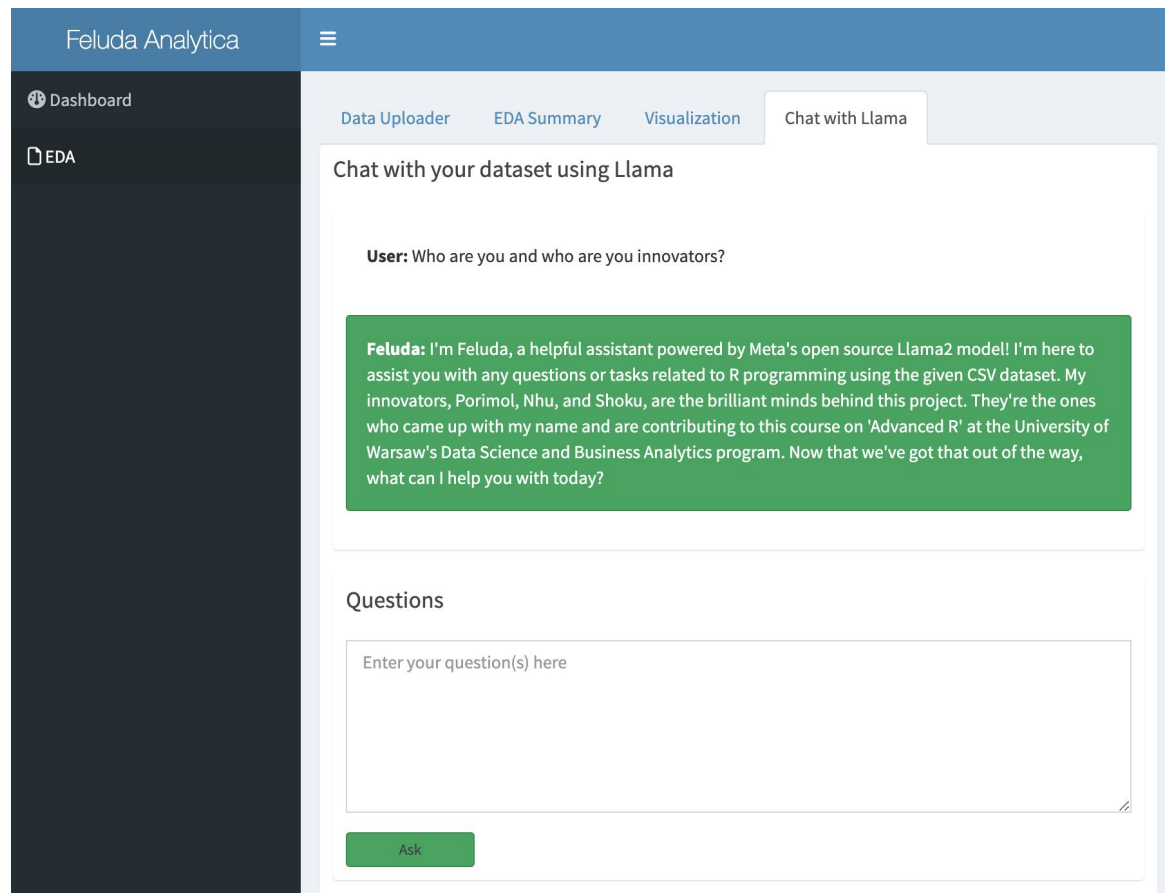
```
.
├── DESCRIPTION
├── LICENSE
├── LidaRollama.Rproj
├── NAMESPACE
├── R
│   └── LidaRollama.R
├── README.md
├── man
│   └── LidaRollama.Rd
├── tests
│   └── testthat
│       └── test-ollama.R
```

```
LidaRollama <- R6Class(
  "LidaRollama",
  private = list(
    api_call = function(api_url, json_payload) {
      if (is.null(api_url)) {
        api_url <- "http://localhost:11434/api/generate"
      }
      handle <- curl::new_handle()
      curl::handle_setopt(
        handle,
        copypostfields = json_payload
      )
      curl::handle_setheaders(handle,
                             "Content-Type" = "application/json",
                             "Accept" = "application/json")
      response <- curl::curl_fetch_memory(
        api_url,
        handle = handle
      )
      # Parse the response
      parsed_response <- jsonlite::fromJSON(rawToChar(response$content))
      return(trimws(parsed_response$response))
    }
  ),
  .....,
  .....,
)
```

ChatBot based on Llama [Cont.]

Prompt:

Who are you and who are your innovators?



The screenshot displays the 'Feluda Analytica' web application. The interface includes a sidebar with 'Dashboard' and 'EDA' options. The main content area has tabs for 'Data Uploader', 'EDA Summary', 'Visualization', and 'Chat with Llama'. The 'Chat with Llama' tab is active, showing a chat window titled 'Chat with your dataset using Llama'. A user prompt is visible: 'User: Who are you and who are you innovators?'. The chatbot's response, in a green box, identifies itself as 'Feluda', a helpful assistant powered by Meta's open source Llama2 model, and mentions its creators: Porimol, Nhu, and Shoku, who are contributing to an 'Advanced R' course at the University of Warsaw's Data Science and Business Analytics program. Below the chat window is a 'Questions' section with a text input field labeled 'Enter your question(s) here' and a green 'Ask' button.

ChatBot based on Llama

User: Could you please write a R6 class in R. Here is the requirements: class name: Student methods name: full_name, address, enrolled_courses

Prompt:

Could you please write a R6 class in R.

Here is the requirements: class name: Student methods name: full_name, address, enrolled_courses

Feluda: I'm afraid I have some bad news... As you can see, there's no data provided for me to work with. The "TRUE" value doesn't contain any relevant information about students or courses. However, I'd be happy to help you create a R6 class in R! Here's an example of how you could define the `Student` class with the required methods: Note that this is a basic example, and you can add more features to the `Student` class as needed.

```
R
library(R6)

Student <- R6::R6Class("Student",
  private = list(
    _enrolled_courses = character()
  ),
  public = list(
    full_name = function(name) {
      paste(name, " Student")
    },
    address = function(address) {
      return(address)
    },
    enrolled_courses = function(courses) {
      _enrolled_courses <- courses
      return(_enrolled_courses)
    }
  )
)

# Create a new student object
student1 <- Student$new(full_name("John"), address("123 Main St"), enrolled_courses=c("Math", "Science"))

# Access the methods
print(student1$full_name()) # Output: John Student
print(student1$address())   # Output: 123 Main St
print(student1$enrolled_courses) # Output: c("Math", "Science")
```

OOP in R (R6)

```
# Define the LIDArApp class
LIDArApp <- R6Class(
  "LIDArApp",
  public = list(
    llm_chat = NULL,
    eda = NULL,

    initialize = function(sys_prompt) {
      # Initialize the LLMChat object
      self$llm_chat <- LLMChat$new(
        model_name = "llama3:latest",
        temperature = 0.7,
        max_length = 512,
        sysprompt = sys_prompt,
        api_url = "http://localhost:11434/api/generate"
      )
      # Initialize the EDA object
      self$eda <- EDA$new()
    },

    # Function about project contributors
    project_contributors = function(output) {
      # Data frame of contributors
      contributors_df <- data.frame(
        Fullname = c("Porimol Chandro", "Shokoufe Naseri", "Ho Thi Hoang Nhu"),
        Student.ID = c(468264, 466750, 466503),
        Email = c("p.chandro@student.uw.edu.pl", "s.naseri@student.uw.edu.pl",
          "t.ho2@student.uw.edu.pl"),
        stringsAsFactors = FALSE
      )
      contributors_df$Student.ID <- format(contributors_df$Student.ID, scientific = FALSE)

      # Render contributors table
      output$contributors_table <- renderTable({
        contributors_df
      },
        rownames = FALSE
      ),
    },

    server = function() {
      .....
    }
  )
}
```

Rcpp implementation

```
// File: corrMatrix.cpp
#include <Rcpp.h>
using namespace Rcpp;

// Function to compute the mean of a vector
double computeMean(NumericVector x) {
    int n = x.size();
    double sum = 0.0;
    for(int i = 0; i < n; ++i) {
        sum += x[i];
    }
    return sum / n;
}

// Function to compute the correlation between two vectors
double computeCorrelation(NumericVector x, NumericVector y) {
    int n = x.size();
    double mean_x = computeMean(x);
    double mean_y = computeMean(y);
    double sum_xy = 0.0;
    double sum_x_squared = 0.0;
    double sum_y_squared = 0.0;
    for(int i = 0; i < n; ++i) {
        sum_xy += (x[i] - mean_x) * (y[i] - mean_y);
        sum_x_squared += pow(x[i] - mean_x, 2);
        sum_y_squared += pow(y[i] - mean_y, 2);
    }
    return sum_xy / sqrt(sum_x_squared * sum_y_squared);
}

// Function to compute the correlation matrix
NumericMatrix computeCorrelationMatrix(NumericMatrix mat) {
    int ncol = mat.ncol();
    NumericMatrix corrMatrix(ncol, ncol);
    for(int i = 0; i < ncol; ++i) {
        for(int j = i; j < ncol; ++j) {
            double corr = computeCorrelation(mat(_, i), mat(_, j));
            corrMatrix(i, j) = corr;
            corrMatrix(j, i) = corr; // because correlation matrix is
            // symmetric
        }
    }
    return corrMatrix;
}
```

Exploratory data analysis [Cont.]

[Data Uploader](#)[EDA Summary](#)[Visualization](#)[Chat with Llama](#)

Summary of Exploratory Data Analysis

wage	educ	exper	female	black	white	midwest
Min. : 2.03	Min. : 1.00	Min. : 0.00	Min. : 0.000	Min. : 0.000	Min. : 0.000	Min. : 0.000
1st Qu.: 5.53	1st Qu.: 12.00	1st Qu.: 10.00	1st Qu.: 0.000	1st Qu.: 0.000	1st Qu.: 1.000	1st Qu.: 0.000
Median : 8.79	Median : 13.00	Median : 18.00	Median : 0.000	Median : 0.000	Median : 1.000	Median : 0.000
Mean : 10.21	Mean : 13.29	Mean : 18.78	Mean : 0.494	Mean : 0.088	Mean : 0.912	Mean : 0.237
3rd Qu.: 12.78	3rd Qu.: 16.00	3rd Qu.: 26.00	3rd Qu.: 1.000	3rd Qu.: 0.000	3rd Qu.: 1.000	3rd Qu.: 0.000
Max. : 60.19	Max. : 18.00	Max. : 52.00	Max. : 1.000	Max. : 1.000	Max. : 1.000	Max. : 1.000

south	west
Min. : 0.000	Min. : 0.000
1st Qu.: 0.000	1st Qu.: 0.000
Median : 0.000	Median : 0.000
Mean : 0.315	Mean : 0.222
3rd Qu.: 1.000	3rd Qu.: 0.000
Max. : 1.000	Max. : 1.000

Data Types

wage	educ	exper	female	black	white	midwest	south	west
"numeric"	"integer"	"integer"	"integer"	"integer"	"integer"	"integer"	"integer"	"integer"

Null Values

[1] 0

Correlation Matrix

	wage	educ	exper	female	black	white	midwest	south
wage	1.000000000	0.44985057	0.149280359	-0.212745399	-0.097224107	0.097224107	0.01616352	-0.11176812
educ	0.449850567	1.000000000	-0.182318532	-0.023342372	-0.050195280	0.050195280	-0.02149096	-0.04605425
exper	0.149280359	-0.18231853	1.000000000	0.008960122	0.001360391	-0.001360391	0.05677602	-0.05113087
female	-0.212745399	-0.02334237	0.008960122	1.000000000	0.031968991	-0.031968991	-0.05680935	0.07057318
black	-0.097224107	-0.05019528	0.001360391	0.031968991	1.000000000	-1.000000000	-0.04030914	0.19970367
white	0.097224107	0.05019528	-0.001360391	-0.031968991	-1.000000000	1.000000000	0.04030914	-0.19970367
midwest	0.016163520	-0.02149096	0.056776016	-0.056809354	-0.040309135	0.040309135	1.000000000	-0.37793916
south	-0.111768120	-0.04605425	-0.051130867	0.070573175	0.199703669	-0.199703669	-0.37793916	1.000000000
west	-0.002686363	-0.03635257	0.012940140	0.011223376	-0.140451190	0.140451190	-0.29771354	-0.36224023

Exploratory data analysis

Correlation Plot with Selected Feature Variables

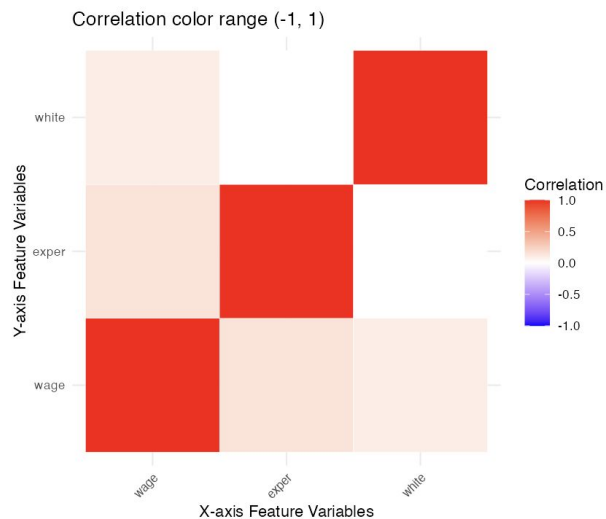
Select columns to plot correlation

wage exper white

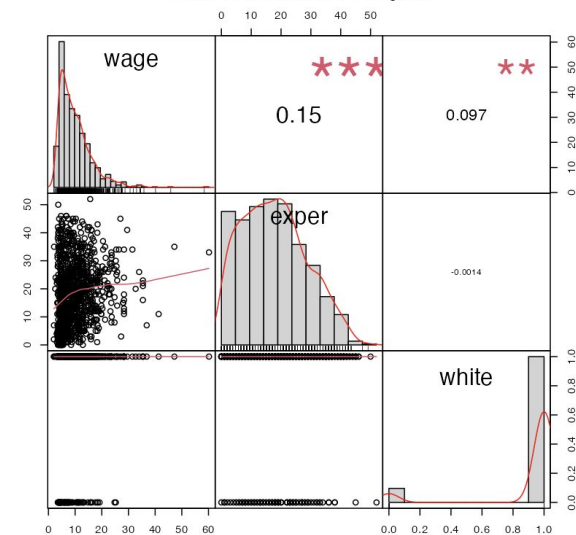
educ
female
black
midwest
south
west

US

white
.097224107
.001360391
.000000000



Correlation Matrix with Histogram



Function vectorisation in R

lapply
and
purrr::map_chr

```
output$chat_response_output <- renderUI({
  chatBox <- lapply(1:nrow(chat_data()), function(i) {
    text_code <- chat_data()[i, "message"]
    if (is.null(text_code) || !nzchar(text_code)) {
      return(NULL)
    }
    # Extract R code
    r_code <- self$llm_chat$extract_code(text_code)

    # Extract surrounding text
    surrounding_text <- self$llm_chat$extract_text(text_code)
    tags$div(
      class = ifelse(
        chat_data()[i, "source"] == "User",
        "alert alert-secondary",
        "alert alert-success"
      ),
      HTML(
        if (length(r_code) > 0) {
          paste0("<b>", chat_data()[i, "source"], "</b> ", text = surrounding_text, r_code)
        } else {
          paste0("<b>", chat_data()[i, "source"], "</b> ", text = surrounding_text)
        }
      )
    )
  })
})
do.call(tagList, chatBox)
})
```

```
# Extract data types using purrr::map_chr
data_types <- map_chr(csv_data, class)

output$data_types_output <- renderPrint({
  data_types})
```

Defensive programming in R [Cont.]

tryCatch and message
for Error

```
csv_to_json_data <- reactiveVal(NULL)
observeEvent(input$file$datapath, {
  observe({
    if (is.null(input$file$datapath)) {
      message("No file uploaded")
    } else {
      tryCatch({
        message("File uploaded")
      })
    }
  })
})

output$expSinglePlot <- renderPlot({
  self$eda$add_ggplot(csv_data, input) + self$eda$add_geom(input)
}, error = function(e) {
  showNotification(paste("Error: ", e$message), type = "error")
})
```

```
gpt_res <- tryCatch({
  self$llm_chat$ollama_api(prompt = question_input)
}, error = function(e) {
  showNotification(paste("Error: ", e$message), type = "error")
  NULL
})
if (!is.null(gpt_res)) {
  gpt_data <- data.frame(source = "Feluda", message = gpt_res,
                        stringsAsFactors = FALSE)
  chat_data(rbind(chat_data(), gpt_data))
}
updateTextInput(session, "question_input", value = "")
})
```

In Llama chatbox

Defensive programming in R

```
LidaRollama <- R6Class(  
  "LidaRollama",  
  private = list(  
    api_call = function(api_url, json_payload) {  
      if (is.null(api_url)) {  
        api_url <- "http://localhost:11434/api/generate"  
      }  
      .....  
    },  
  ),  
  public = list(  
    model_name = NULL,  
    temperature = NULL,  
    max_length = NULL,  
    sysprompt = NULL,  
    api_url = NULL,  
  
    initialize = function(  
      model_name,  
      temperature,  
      max_length,  
      sysprompt,  
      api_url  
    ) {  
      # defensive programming  
      # check if the input is of the correct type  
      stopifnot(  
        is.character(model_name),  
        is.numeric(temperature),  
        is.numeric(max_length),  
        is.character(sysprompt)  
      )  
      self$model_name = model_name  
      self$temperature = temperature  
      self$max_length = max_length  
      self$sysprompt = sysprompt  
      self$api_url = api_url  
  
      invisible(self)  
    }  
  )  
)
```

stopifnot
in Package

Data visualization [Cont.]

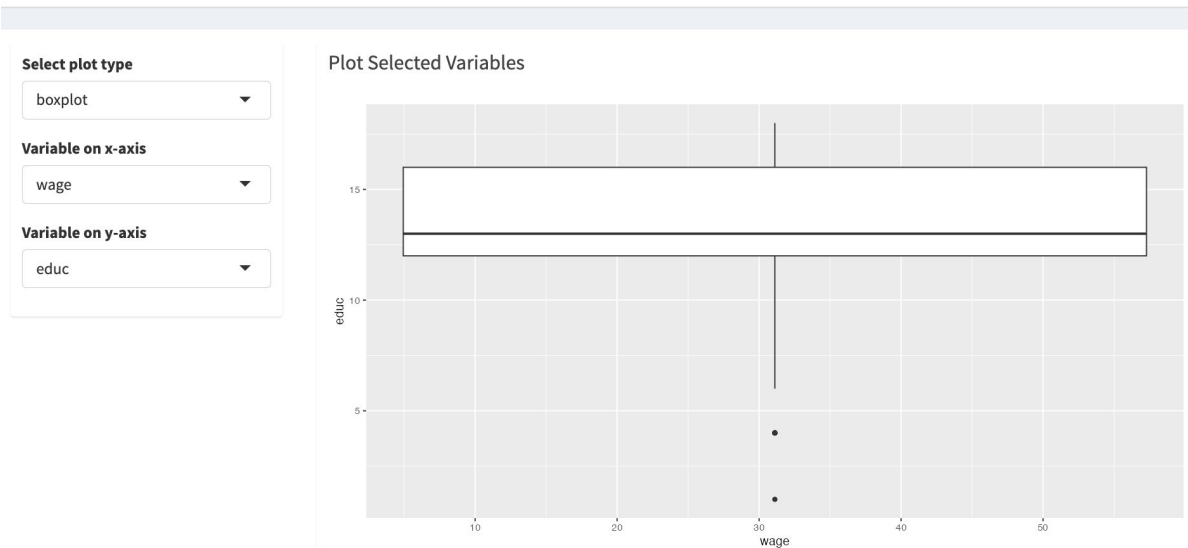
Data Uploader EDA Summary Visualization Chat with Llama

Top 5 rows of the dataset.

Show 10 entries Search:

	wage	educ	exper	female	black	white	midwest	south	west
1	2.03	13	2	1	0	1	0	1	0
2	2.0699999	12	7	0	0	1	1	0	0
3	2.1199999	12	35	0	0	1	0	1	0
4	2.54	16	20	1	0	1	0	1	0
5	2.6800001	12	24	1	0	1	0	1	0

Showing 1 to 5 of 5 entries Previous 1 Next



Data visualization

Select plot type

point ▲

point

boxplot

histogram

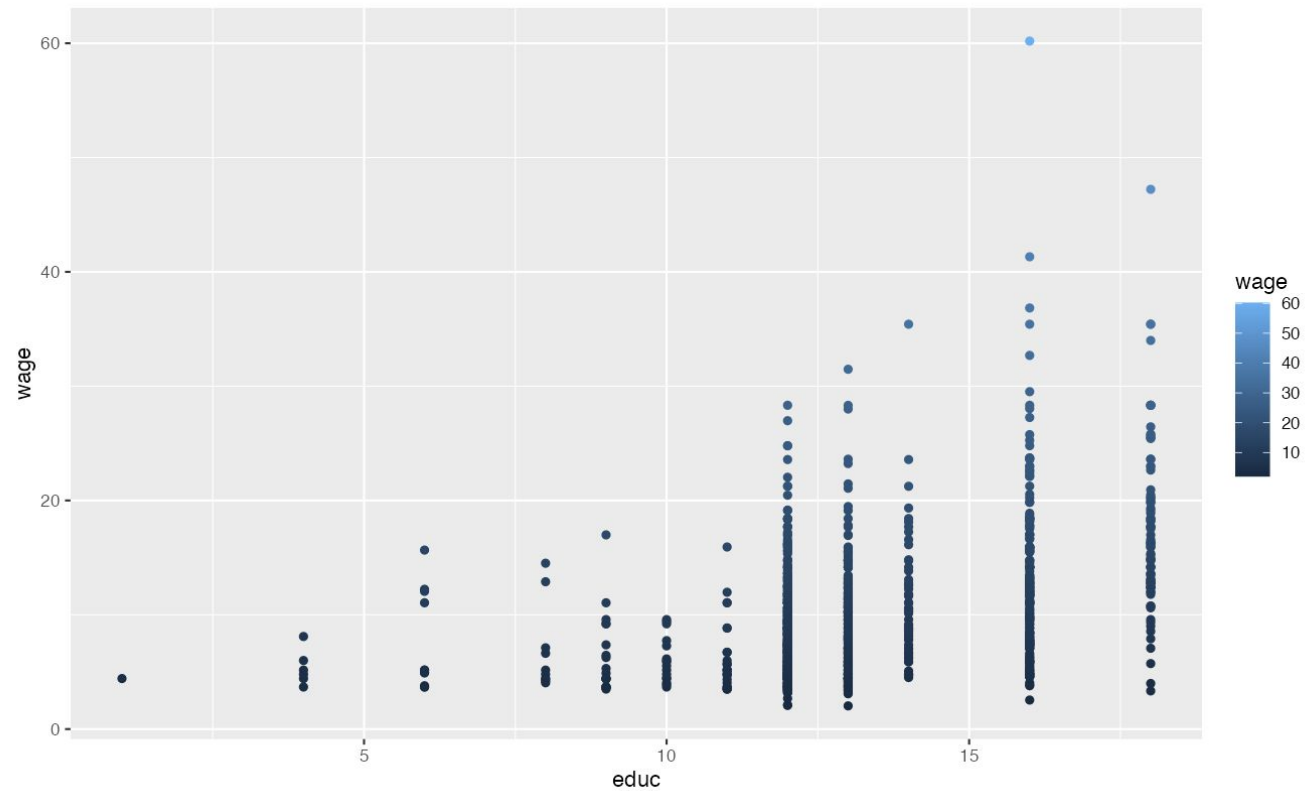
density

wage ▼

Variable to color by

wage ▼

Plot Selected Variables



Conclusion

In conclusion, developing an automated EDA tool using ShinyApp in R offers significant advantages for data science projects.

- The automation of EDA processes ensures efficiency and consistency, greatly speeding up the analysis and enhancing the developer experience.
- The interactive nature of these tools allows for deeper exploration of datasets, with transparency and comprehensive understanding.
- The ability to customize EDA tools to meet specific project requirements and developer preferences provides flexibility and improved usability.
- By leveraging **large language models (LLMs)**, the project aims to streamline the creation of informative visualizations from natural language descriptions, reducing the time and skill required for effective data analysis.

Reference

- [1] Maria Kubara and Ewa Weychert (2024). 'Advanced Programming in R'.
https://usosweb.wne.uw.edu.pl/kontroler.php?_action=katalog2/przedmioty/pokazPrzedmiot&kod=2400-DS1APR .
- [2] Dibia, V. (2023). Lida: A tool for automatic generation of grammar-agnostic visualizations and infographics using large language models. arXiv preprint arXiv:2303.02927 .
- [3] Touvron, Hugo, et al. 'Llama: Open and efficient foundation language models.' arXiv preprint arXiv:2302.13971 (2023) .
- [4] 'Ollama (Version v0.1.38)' [Computer software].
<https://github.com/ollama/ollama> .

Acknowledgement

We would like to thank you **Mgr. Ewa Weychert** and **Mgr. Maria Kubara** for introducing and teaching us all the techniques during the semester on **Advanced Programming in R** course.

Appendix

Demo: short video

The screenshot displays the RStudio environment with the following components:

- Script Editor:** Contains an R script with a function `check_and_install` and a list of packages to be installed. A yellow warning banner at the top states: "Package this.path required but is not installed. Install Don't Show Again".
- Environment:** Shows the current environment with variables like `coef_gen`, `coef_ind`, `full_for`, `i`, `indeksy`, `insignif`, `json_data`, `required`, `rr`, and `x`.
- Console:** Displays the output of the script execution, including warnings about aesthetics and a message about the file upload.

```
1 # Check if a package is installed, and if not, install it
2 check_and_install <- function(pkg) {
3   if (!requireNamespace(pkg, quietly = TRUE)) {
4     install.packages(pkg, dependencies = TRUE)
5   }
6   library(pkg, character.only = TRUE)
7 }
8
9 # List of required packages
10 packages <- c("shiny", "shinydashboard", "shinyjs", "R6", "curl",
11              "jsonlite", "jsonify", "reshape2", "ggplot2", "dplyr",
12              "PerformanceAnalytics", "this.path", "Rcpp")
13
14 # Check and install each package
15 lapply(packages, check_and_install)
16 # Get the path of the currently executing script
17 setwd(this.path::this.dir())
```

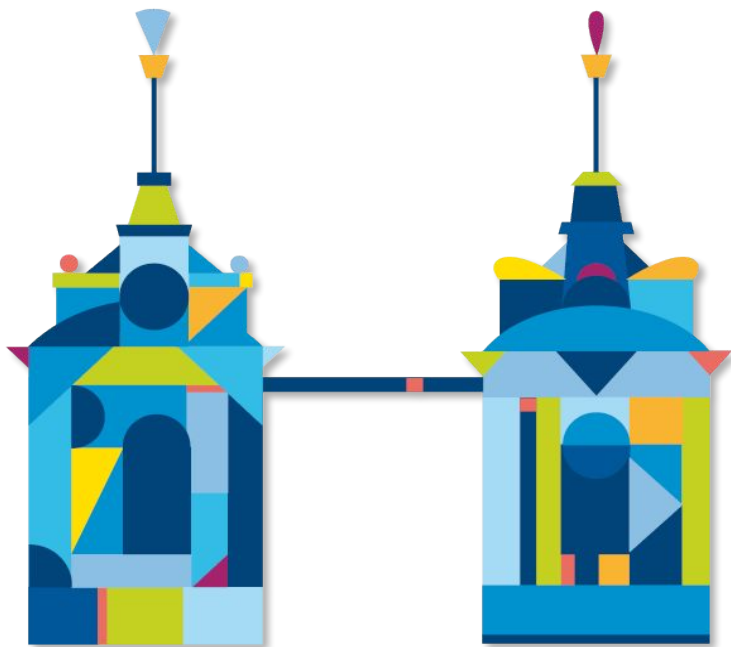
Warning: Error in data.frame: arguments imply differing number of rows: 1, 0

1: shiny::runApp

> runApp('C:/Users/Shokoufeh/Downloads/feluda-lidar-main (3)/feluda-lidar-main')

Listening on http://127.0.0.1:6831
File uploaded
> |

Thank you for your attention!



Faculty of Economic Sciences,
University of warsaw

ul. Długa 44/50
00-241 Warszawa, Poland

<https://www.wne.uw.edu.pl>