ANALYST

Co-funded by the European Union

MODULE 5/R and Data Analysis

Carlo Drago PhD

University Niccolo Cusano, Rome

MASSIVE OPEN ONLINE COURSE (MOOC)

Project N. 2023-1-IT02-KA220-HED-000161770

ANALYST - A New Advanced Level for Your Specialised Training

HOZINT HORIZON INTELLIGENCE

fyg consultores

UNIVERSITY OF WESTERN MACEDONIA

Centro Studi AMIStaDeS APS

UNIVERSITÀ CUSANO

European Institute for Local Development

UNIVERSIDAD DE ALMERÍA

THE R PROGRAMMING LANGUAGE

- The **R programming language** is a powerful tool for data analysis. It offers a comprehensive environment for statistical computing and graphics, making it a popular choice among data analysts and statisticians.
- With its extensive library of packages, R can handle a wide array of data manipulation, calculation, and graphical display tasks, providing users with the flexibility to conduct complex analyses and visualize data in various ways.

ANALYST

Co-funded by the European Union

THE R PROGRAMMING LANGUAGE

R FUNCTIONALITIES

- **Data Analytics** in practice often involves utilizing various tools and techniques to extract insights from data. In R, a powerful programming language used extensively in data analysis, several functionalities can be employed to streamline and enhance this process:
- 1. **Data Manipulation**: With packages like `dplyr` and `tidyr` for instance, R allows users to easily manipulate and tidy datasets, enabling efficient filtering, selection, and transformation of data.
- 2. **Data Visualization**: R excels in creating visual representations of data using libraries such as `ggplot2`, which provides a flexible and systematic approach to plotting data.
- 3. **Statistical Analysis**: R is renowned for its statistical capabilities, offering a wide range of functions for performing complex analyses, such as regression modeling,

R FUNCTIONALITIES

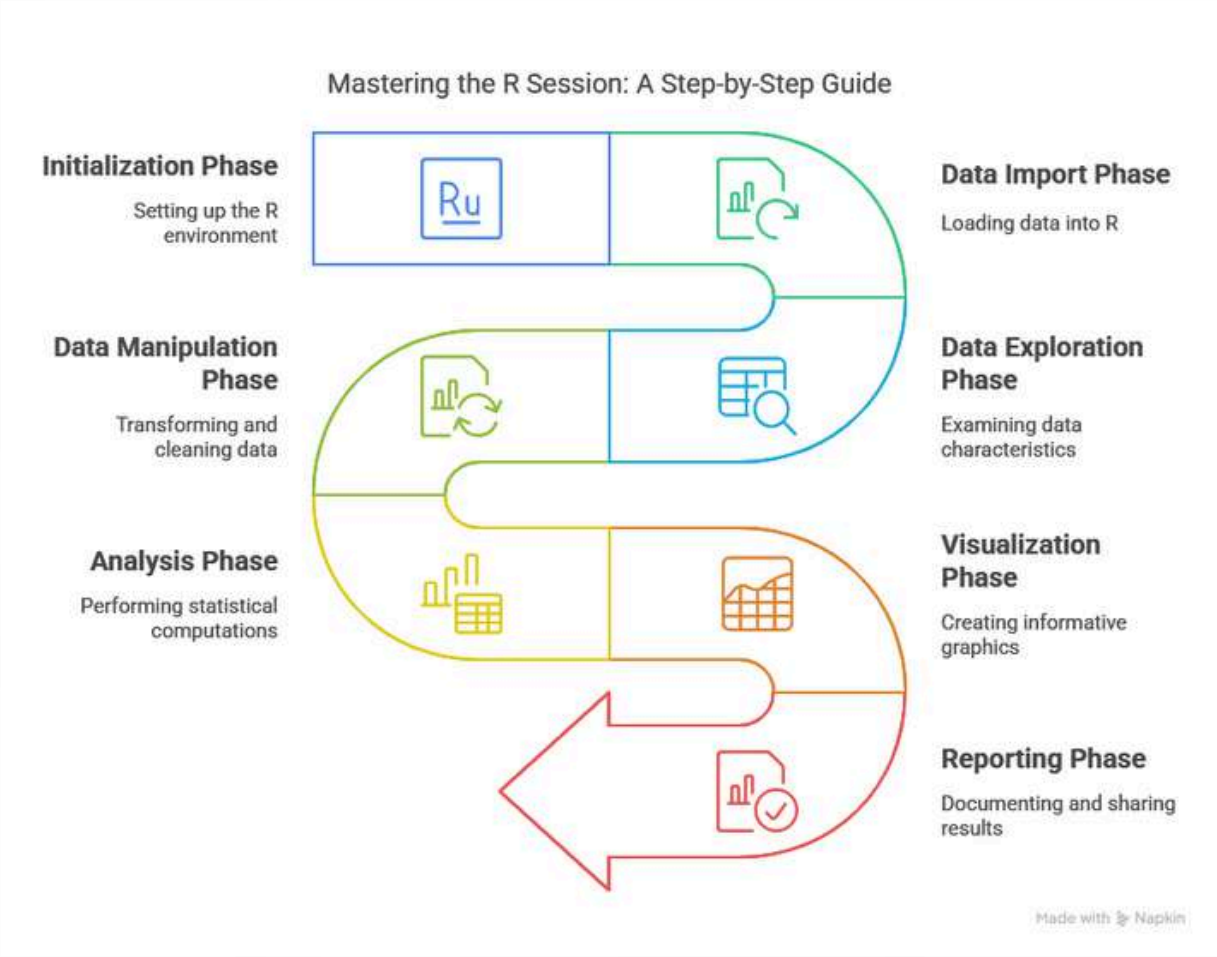
hypothesis testing, and time-series analysis.

- 4. **Machine Learning**: With packages like `caret` and `randomForest`, R supports the implementation of machine learning algorithms, allowing users to develop predictive models and improve decision-making processes.
- 5. **Data Import and Export**: R facilitates easy data import and export from various file formats, including CSV, Excel, and databases, ensuring seamless integration with other systems.
- By leveraging these functionalities, R becomes an indispensable tool for data analysts seeking to uncover valuable insights and drive data-driven decision-making.

ANALYST

Co-funded by the European Union

R SESSIONS



HOZINT HORIZON INTELLIGENCE

fyg consultores

UNIVERSITY OF WESTERN MACEDONIA

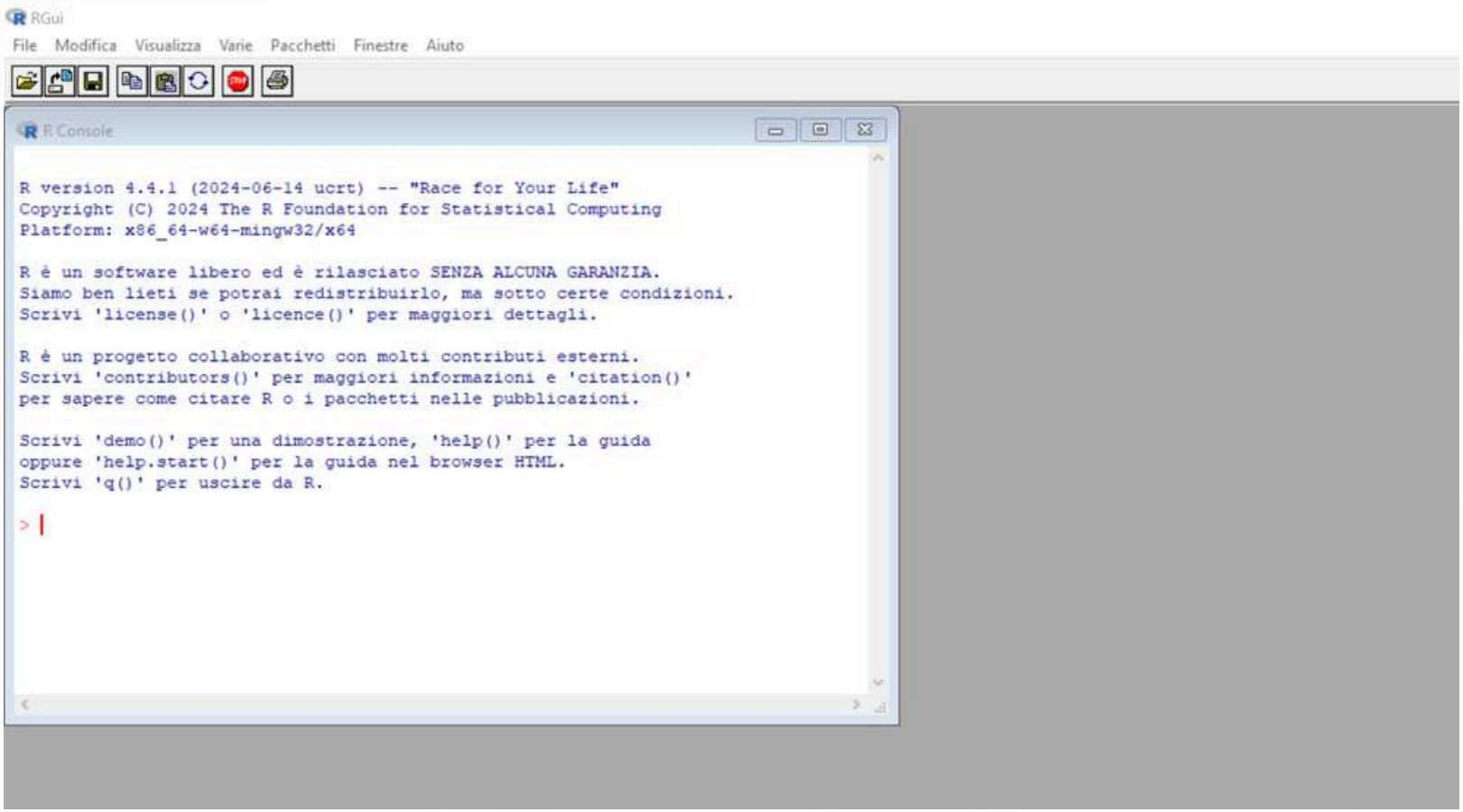
Centro Studi AMIStaDeS APS

UNIVERSITÀ CUSANO

European Institute for Local Development

UNIVERSIDAD DE ALMERÍA

R GUI


RGui
File Modifica Visualizza Varie Pacchetti Finestre Aiuto
R Console
R version 4.4.1 (2024-06-14 ucrt) -- "Race for Your Life"
Copyright (C) 2024 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64
R è un software libero ed è rilasciato SENZA ALCUNA GARANZIA.
Siamo ben lieti se potrai redistribuirlo, ma sotto certe condizioni.
Scrivi 'license()' o 'licence()' per maggiori dettagli.
R è un progetto collaborativo con molti contributi esterni.
Scrivi 'contributors()' per maggiori informazioni e 'citation()'
per sapere come citare R o i pacchetti nelle pubblicazioni.
Scrivi 'demo()' per una dimostrazione, 'help()' per la guida
oppure 'help.start()' per la guida nel browser HTML.
Scrivi 'q()' per uscire da R.
>

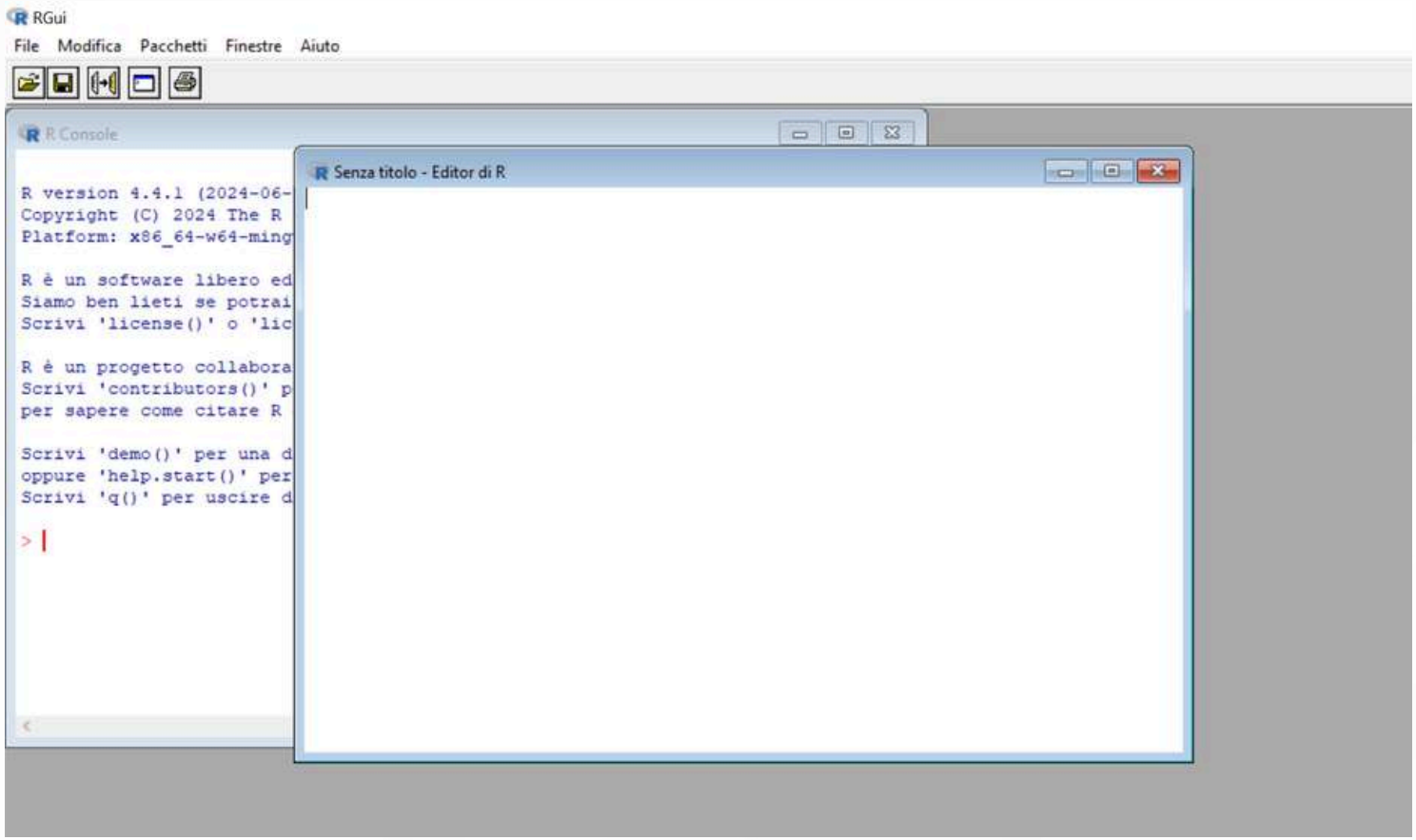
R GUI

The **R GUI, or Graphical User Interface**, is a user-friendly interface for interacting with the R programming language. It allows users to execute R commands, scripts, and visualize data without needing to rely solely on command-line inputs. The R GUI typically includes features like script editors, data viewers, and plotting tools, making it easier for users, especially those less familiar with programming, to engage with R's statistical and graphical capabilities. Popular R GUIs include RStudio and R Commander, each offering additional tools and resources to enhance the R programming experience.

ANALYST

Co-funded by the European Union

R SCRIPTS



HOZINT HORIZON INTELLIGENCE
fyg consultores
UNIVERSITY OF WESTERN MACEDONIA
Centro Studi AMIStaDeS APS
UNIVERSITÀ CUSANO
European Institute for Local Development
UNIVERSIDAD DE ALMERÍA

ANALYST

Co-funded by the European Union

R SCRIPTS

R scripts are essential components in the R programming language, used for performing a wide range of data analysis and statistical tasks. These scripts are essentially text files containing a sequence of R commands and functions, which can be executed to manipulate data, perform calculations, and generate visualizations.

ANALYST

Co-funded by the European Union

R PROGRAMMING

The **importance of programming** with R scripts lies in their ability to automate repetitive tasks, ensuring consistency and efficiency in data processing. By writing scripts, users can document their workflows, making analyses reproducible and easier to share with others. Moreover, R scripts enable the handling of large datasets, complex statistical operations, and the development of sophisticated models, which are crucial for data-driven decision-making in various fields such as finance, healthcare, and scientific research.

Overall, programming in R empowers users to leverage the full capabilities of statistical computing and data analysis, driving insights and innovation.

DATA IMPORT IN R

When working with data in R, **importing data is a crucial step.** One convenient function for importing data directly from the clipboard is `read.delim('clipboard', header=TRUE)`. This function allows you to quickly bring data into R by copying it to the clipboard and then using this command to import it. The `header=TRUE` argument specifies that the first row of your data contains column names, ensuring that your data is structured correctly upon import. This method is particularly useful for small datasets or quick data analyses, as it saves time and effort in specifying file paths.

ANALYST

Co-funded by the European Union

PRE PROCESSING IN R

Pre-processing in R involves preparing raw data for analysis by cleaning and transforming it into a suitable format. This process typically includes handling missing values, normalizing or scaling data, encoding categorical variables, and removing duplicates or irrelevant features.

PRE PROCESSING IN R

Pre-processing in R involves preparing raw data for analysis by cleaning and transforming it into a suitable format. Here are some simple commands it is possible to use:

1. **Handling Missing Values:** It is possible to use `na.omit()` remove rows with missing values or `impute()` from the `Hmisc` package to fill them.

```R
cleaned_data <- na.omit(raw_data)
```

PRE PROCESSING IN R

2. **Normalizing or Scaling Data:** It is possible to use the `scale()` function to normalize or standardize your data.

```R
scaled_data <- scale(raw_data)
```

3. **Encoding Categorical Variables**: Convert categorical variables into factors using `as.factor()`.

APPLY

In R, the `**apply**` function is a powerful tool used to perform operations on rows or columns of a matrix or an array. It allows you to apply a function to each row or column, making it an efficient way to manipulate and analyze data without using loops. The basic syntax of `apply` is `apply(X, MARGIN, FUN, ...)`, where `X` is the data, `MARGIN` specifies whether to apply the function over rows (1) or columns (2), and `FUN` is the function to be applied. This command is particularly useful for data frame operations, enabling concise and readable code.

TAPPLY

The `**tapply**` function in R is a useful command for applying a function over subsets of a vector, allowing you to perform group-wise operations. It is particularly handy when you want to compute summary statistics or perform calculations across different categories of data in a single vector. By specifying a factor or list of factors, `tapply` applies the chosen function to each subset, simplifying data analysis tasks involving grouped data.

LAPPLY

The **lapply** function in R is used to apply a function to each element of a list. The result is a new list where each element is the result of the function applied to the corresponding element of the original list. It is useful for processing data in a list without needing to manually iterate.

DESCRIPTIVE ANALYSIS IN R

Descriptive statistics is a fundamental aspect of data analysis that focuses on summarizing and organizing data to make it easily interpretable. In the R programming language, a powerful tool for statistical computation, descriptive statistics are typically used to provide a clear summary of the key characteristics of a dataset.

These statistics are often conveyed through visual displays like graphs and tables, as well as summary measures including the mean, median, mode, variance, and standard deviation.

DESCRIPTIVE ANALYSIS IN R

Using R, it is possible efficiently compute these statistics to gain insights into data. For instance, the `summary()` function can be used to quickly obtain a basic statistical overview of a dataset, including the minimum, maximum, median, and mean values for each variable. Additionally, packages such as `ggplot2` can be used to create detailed graphical representations of data, enhancing the understanding of its distribution and trends.

Descriptive statistics in R provide an essential foundation for further statistical analysis by allowing to grasp the basic features of your data. Whether it is possible to explore the central tendency or variability, these statistics serve as the first step in the analytical process.

ANALYST

Co-funded by the European Union

EXPLORATORY DATA ANALYSIS IN R

Exploratory Data Analysis (EDA) in R involves using statistical tools and visualization techniques to summarize the main characteristics of a dataset. It's a crucial step in the data analysis process that helps identify patterns, spot anomalies, test hypotheses, and check assumptions with the aid of summary statistics and graphical representations.

In R, EDA typically starts with loading the dataset, often using functions like `read.csv()` or `read.table()`. Once the data is loaded, functions such as `summary()`, `str()`, and `head()` the results provide a quick overview of the data structure and key statistics.

CORRELATION ANALYSIS IN R

Correlation analysis is a statistical method used to evaluate the strength and direction of the linear relationship between two continuous variables. In R, this analysis can be performed using functions like cor() to calculate the correlation coefficient, which ranges from -1 to 1.

A coefficient close to 1 indicates a strong positive correlation, while a coefficient near -1 suggests a strong negative correlation. A value around 0 implies little to no linear relationship. R also provides functions like cor.test() for testing the significance of the correlation

SIMPLE REGRESSION ANALYSIS IN R

Simple regression analysis in R involves examining the relationship between two variables: one independent variable and one dependent variable. To perform simple regression in R, it is possible to use the lm() function. Here's a basic overview of the process:

1. **Preparing the Data:** It is necessary that the data is clean and organized, typically in a data frame where columns represent different variables.
2. **Loading Data:** Importing the dataset into R using functions like read.csv() for CSV files.
3. **Fit the Model:** Using the lm() function to fit a linear model. For example, if it is necessary to predict y based on x, it is necessary to use:
4. model <- lm(y ~ x, data = your_data)

Multiple regression in R involves using multiple predictor variables to predict a single outcome variable. It is an extension of simple regression, which only uses one predictor. In R, it is possible to perform multiple regression using the `lm()` function. Here's a basic example:

```
# Fitting a multiple regression model
model <- lm(outcome ~ predictor1 + predictor2, data = data)

# Summary of the model
summary(model)
```

SENSITIVITY ANALYSIS IN R

2. **Monte Carlo Simulation:** This method involves running simulations to understand the variability of outcomes.

- Use the `set.seed()` function for reproducibility.
- Generate random samples of input parameters using functions like `rnorm()` or `runif()`.
- Evaluate the model for each set of generated inputs and analyze the results.

3. **Custom Implementations:** If it is possible to have a specific model or set of inputs, it is possible to write functions to simulate variations in input and observe changes in output.

- Create a loop or use `apply()` functions to systematically vary inputs.
- Collect and analyze the output data to determine sensitivity.

ANALYST

Co-funded by the European Union

SENSITIVITY ANALYSIS IN R

By applying these methods, it is possible effectively conduct sensitivity analysis in R to better understand the robustness of the models and the influence of uncertain inputs on the results.

REFERENCES

- Anuradha, J. (2015). A brief introduction on Big Data 5Vs characteristics and Hadoop technology. Procedia computer science, 48, 319-324.
- Choi, T. M., & Lambert, J. H. (2017). Advances in risk analysis with big data. Risk Analysis, 37(8), 1435-1442

Testo del paragrafo

- Elgendy, N., & Elragal, A. (2016). Big data analytics in support of the decision making process. Procedia Computer Science, 100, 1071-1084.
- Greene, W. H. (2002) *Econometric analysis.* 5th Ed. Prentice-Hall
- Joshi, A. P., & Patel, B. V. (2021). Data preprocessing: the techniques for preparing clean and quality data for data analytics process. Orient. J. Comput. Sci. Technol, 13(0203), 78-81.

REFERENCES

- Moreira, J., Carvalho, A., & Horvath, T. (2018). A general introduction to data analytics. John Wiley & Sons. University Press.
- Neuhäuser, M., & Ruxton, G. D. (2024). The statistical analysis of small data sets. Oxford
- Piccolo, D. (1998). Statistica. Il mulino.
- Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., ... & Tarantola, S. (2008). Global sensitivity analysis: the primer. John Wiley & Sons.
- Sharma, A. K., Sharma, D. M., Purohit, N., Rout, S. K., & Sharma, S. A. (2022). Analytics techniques: descriptive analytics, predictive analytics, and prescriptive analytics. Decision intelligence analytics and the implementation of strategic business management, 1-14

REFERENCES

- Weisberg, S. (2004). Lost opportunities: Why we need a variety of statistical languages. Journal of Statistical Software, 13, 1-12.