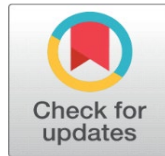
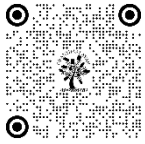


# PYTHON AND R: A SIDE-BY- SIDE EVALUATION FOR ANALYTICS EXCELLENCE

Rekha Raichal<sup>1</sup>

<sup>1</sup> Apoorva S Seshadripuram First Grade College, India



## ABSTRACT

This paper compares Python and R Studio in the domain of data analytics, focusing on their capabilities, libraries, scalability, ease of use, and application suitability. Python, being a general-purpose language, is favored for machine learning and production systems, while R is tailored for statistical analysis and visualization. This study provides insights to help data scientists choose the appropriate tool based on specific requirements.

The investigation focuses on a comparative analysis of Python and R in data analytics, particularly regarding their capabilities, ecosystems, scalability, ease of use, and application suitability.

The analysis aims to determine the strengths and weaknesses of Python and R, helping data professionals choose the appropriate tool for specific requirements in data science and analytics.

The study compared Python and R by evaluating their features, library ecosystems, performance, scalability, learning curves, and suitability for various applications. Sources included peer-reviewed articles, technical blogs, and industry whitepapers.

Python is versatile and excels in data wrangling, machine learning, and production systems.

R specializes in statistical analysis and visualization.

Python has a broader application scope and better scalability for large datasets.

R is preferred for statistical precision and high-quality graphics.

Both languages have strong, complementary ecosystems.

The findings reveal that Python and R are tailored for different purposes. Python is more suitable for general-purpose data manipulation and machine learning, while R is ideal for statistical research and detailed visualization. Combining their strengths can enhance data analytics workflows, emphasizing the importance of mastering both languages for maximum versatility.

**Keywords:** Edward Soja, Third Space Theory, William Shakespeare, As You Like It, Forest of Arden, Spatial Dynamics, Identity Transformation, Social Relations, Liminality, Hybridity

## DOI

10.29121/shodhkosh.v5.i1.2024.4187

**Funding:** This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

**Copyright:** © 2024 The Author(s). This work is licensed under a [Creative Commons Attribution 4.0 International License](#).

With the license CC-BY, authors retain the copyright, allowing anyone to download, reuse, re-print, modify, distribute, and/or copy their contribution. The work must be properly attributed to its author.



## 1. INTRODUCTION

- **Purpose:** Introduce the purpose of the document, which is to compare how Python and R are used in industry data analysis.
- **Overview of R and Python**
- **R:** A programming language primarily designed for statistical analysis and data visualization. It is widely used in fields like research, statistics, and academia.

- **Python:** A general-purpose language that has become a favorite for data analysis, machine learning, and automation. Known for its simplicity and large ecosystem.
- **Context:** Explain that data analysis is key in industries like finance, healthcare, retail, and manufacturing for decision-making, forecasting, and optimization.

Data analytics is pivotal in modern decision-making, with Python and R being two leading programming tools. Python's versatility and R's statistical precision offer unique advantages. This paper examines their capabilities, emphasizing their strengths, weaknesses, and ideal applications.

A Data Scientist's Toolkit therefore contains, for starters, infrastructure tools that collect, store, and prepare data—irrespective of its format, structured, semi-structured, or unstructured—or its source. Then there are analysis and visualization tools which make intelligible and actionable the data. Among these, some of them are software that enables analysis, while others are programming languages, the most used of which are Python and R. R and Python are considered the main languages that every Data Scientist should master. Besides, as a fruit of competition between them, a constant improvement of their functionalities for data processing can be noticed - Jakobowicz (2018).

## 2. METHODOLOGY

The analysis evaluates Python and R based on functionality, library ecosystems, scalability, ease of learning, and visualization capabilities. Data sources include peer-reviewed articles, technical blogs, and industry whitepapers.

### Introduction to the R Language

The R language is an implementation of the S programming language, enhanced with Scheme-inspired lexical scoping and garbage collection features. It builds upon the S language, originally created by John Chambers, a statistician at Harvard University, with the assistance of colleagues at Bell Laboratories during 1975-1976.

The R project itself, however, began in 1993 as a research initiative by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand. The first official version of R, version 1.0.0, was released on February 29, 2000.

## 3. R LANGUAGE FEATURES

R shares many similarities with Python, making it a versatile tool for various applications. Key features include:

- **Free Usage:** R is completely free to use, including in commercial projects.
- **Multi-Paradigm Support:** R supports object-oriented, structured imperative, procedural, and functional programming paradigms.
- **Cross-Platform Compatibility:** R operates across diverse devices, from smartphones to mainframes, and supports various operating systems, including GNU/Linux, Windows, NetBSD, FreeBSD, and MacOS.
- **Built-In Tools:** It comes equipped with extensive tools for statistical analysis and graphical representation.
- **Dynamic Typing:** Like Python, R is dynamically typed, eliminating the need to explicitly declare variable types in code.
- **Reflective and Introspective:** R enables analysis and manipulation of its own structure and behavior.
- **Extensibility:** R integrates seamlessly with database management systems like PostgreSQL (via PL/R) and MySQL, geographic information systems like GRASS, and supports exporting results to formats such as LaTeX or OpenDocument.
- **Extensive Library of Packages:** R offers a broad spectrum of applications, including multivariate statistics, econometrics, biometrics, epidemiology, modeling, image processing, graph analysis, numerical analysis, data mining, and big data analysis. It also excels in producing high-quality graphics.
- **Continuous Evolution:** R is consistently updated and improved to expand its capabilities.

## 4. GETTING STARTED WITH THE R LANGUAGE

The R interpreter, often referred to as **R base**, primarily operates as a command-line application. You can download it from the Comprehensive R Archive Network (CRAN) at the following URL: <https://cran.r-project.org/bin/>.

On the CRAN website, versions of R base are available for operating systems including Linux, Windows, and MacOS(X).

Once R is installed on your computer, you can launch it by running the executable file. When the program starts, a

command prompt, typically represented by the symbol `>`, will appear, signaling that R is ready to process your commands.

## 5. GRAPHICAL USER INTERFACES FOR R

Under Windows, the default graphical user interface (GUI) for R base is relatively basic. While it simplifies tasks like installing external packages, it lacks advanced features for editing R code. On MacOS, the R GUI is more advanced, offering additional functionality. To enhance usability, several other GUIs can be used alongside R base:

- **RGUI:** The default GUI for R on Windows, widely used by Windows users.
- **JGR (Java GUI for R):** A platform for running R in a Java-based environment.
- **Rattle:** A beginner-friendly GUI for data mining in R. It provides tabs for data loading, statistical and visual summaries, data transformation, model building, performance evaluation, and visualization.
- **Rcmdr (R Commander):** An R GUI available as the Rcmdr package, which can be freely downloaded from CRAN.
- **RKward:** Originally developed for KDE, this GUI now works on Linux, Windows, and MacOS. It includes table editing, CSV imports, and tools for performing statistical analyses with both graphical options and command-line input.
- **SciViews-R GUI:** A GUI designed to complement R with open-source tools for statistical computing within a reproducible workflow.
- **RStudio:** A powerful and user-friendly multi-platform GUI for R. It features an integrated code editor, debugging tools, and visualization capabilities. RStudio supports Windows, MacOS, and Linux, and also offers an online version, **RStudio Cloud**, which provides similar functionality without requiring installation.

## 6. GETTING STARTED WITH RSTUDIO

To begin using RStudio:

1. **Install R base**
  - Download R base from CRAN at <https://cran.r-project.org/bin/>. It is a prerequisite for RStudio.
2. **Install RStudio**
  - Download RStudio from <https://posit.co/download/rstudio-desktop/>.
  - Alternatively, use **RStudio Cloud** online at <https://login.rstudio.cloud/login>, which requires no installation.

Once installed or accessed online:

- **Open RStudio.**
- To create a new R script:
  - Navigate to the **File** menu, select **New File**, and click on **R Script**.
  - Alternatively, click the **+** symbol in the toolbar and select **R Script**.
- Ensure the **R Console** is open in the bottom-right panel.
- Write or paste your code into the script editor.
- To run code:
  - Highlight the code you wish to execute and click **Run**, or use shortcuts:
    - Ctrl+Enter on Windows
    - Cmd+Enter on Mac
    - The output will display in the console.

RStudio Interface Overview

- **Console (Bottom Left):** Used to execute commands and view results.
- **Workspace (Top Right):** Displays datasets, variables, and other objects in your environment.
- **Bottom Right Panel:** Handles file management, visualizations (charts), help documentation, and package installation. Tabs in this section allow easy switching between these functionalities.

## 7. ADVANTAGES OF USING R SCRIPTS

R scripts offer significant benefits compared to directly typing commands into the R console:

- **Executing Complete Programs:** R scripts enable users to write and run entire R programs in a single file, which is especially useful for automating repetitive tasks and handling complex analyses.
- **Command-Line Arguments:** Scripts support command-line arguments, making them invaluable for automating workflows or processing large datasets.

- **Output Versatility:** R scripts can produce outputs in various formats, including HTML, PDF, and CSV, making them highly versatile for reporting and documentation purposes.

## 8. STEPS TO RUN AN R SCRIPT IN RSTUDIO

Follow these steps to execute an R script in RStudio:

1. *Open RStudio.*
2. **Create a New Script:**
  - Go to the **File** menu, select **New File**, and click on **R Script**,
  - Or click the **+** button in the toolbar and select **R Script**.
3. **Ensure the R Console is Open:** The console should be visible in the bottom-right panel of the interface.
4. **Write or Paste Code:** Type or paste your code into the script editor.
5. **Highlight the Code:** Select the lines of code you want to run by clicking and dragging your mouse over them.
6. *Run the Code:*
  - Click the **Run** button,
  - Or use the keyboard shortcut:
    - Ctrl+Enter (Windows)
    - Cmd+Enter (Mac).

The output will appear in the RStudio console.

Presentation of the Python Language Python is a free, portable, and high-level interpreted programming language known for its strong dynamic typing, automatic memory management, and robust exception handling. Initially developed by Guido van Rossum in 1989, it has grown through contributions from numerous volunteers. Python's first release was on February 20, 1991, with its latest version launched on April 5, 2023. As an interpreted language, Python processes instructions by translating them into machine language in real-time.

## 9. CHARACTERISTICS OF PYTHON

Python stands out due to several features:

- **Free and Open Source:** Python is free to use and unrestricted for commercial projects.
- **Multi-Paradigm:** It supports object-oriented, structured imperative, procedural, and functional programming. Features like list comprehensions, dictionaries, and sets enhance its functionality.
- **Multi-Platform:** Python runs on various platforms, including smartphones, mainframes, and operating systems like GNU/Linux, Android, macOS, iOS, and Windows.
- **Multi-Threading:** It offers optional multi-threading capabilities.
- **Dynamic Typing:** Variables don't require type declarations; Python determines their types at runtime.
- **Orthogonality and Reflection:** Python's small set of concepts generates diverse constructions, supports metaprogramming, and is introspective with tools like debuggers and profilers.
- **Extensibility:** It integrates seamlessly with existing C libraries and external software like databases and geographic information systems.
- **Evolving Community:** Python's development is driven by a vibrant community. Its primary C-based interpreter is actively maintained, alongside a Java-based interpreter in progress.

## 10. GETTING STARTED WITH PYTHON

You can write Python programs directly in the interpreter via a command-line interface or within specialized Integrated Development Environments (IDEs). Popular Python IDEs include Visual Studio Code, Atom, Vim, PyDev, Jupyter Notebook, JupyterLab, Spyder, and PyCharm. Tools like JupyterLab, Jupyter Notebook, and Spyder are also accessible through the Anaconda distribution, a free Python/R data science platform that bundles multiple tools and libraries.

## 11. WORKING WITH JUPYTER NOTEBOOK

Jupyter Notebook, part of the Anaconda distribution, is an interactive IDE allowing users to combine code execution with text, multimedia, equations, and plots. It supports Linux, Windows, and macOS and is widely used for creating and sharing documents.

### *Steps to Run Python Scripts in Jupyter Notebook*

1. Launch the Anaconda Navigator and click on the "Launch" icon under Jupyter Notebook.
2. Create a new Notebook file by selecting "New" → "Python 3."
3. Write Python code in "Code" cells, which are marked with "In [ ]" on the left.
4. Execute cells in order by selecting them and clicking "Run" or using the shortcut **Shift + Enter**.

### *Execution Notes:*

- Execute cells sequentially to ensure proper flow. If a cell is re-run, its output updates automatically.
- Use the "Run All" option to execute all cells at once.
- Restarting a Notebook begins by interrupting the kernel and re-running cells from the start.

Jupyter Notebook's interactive and iterative features make it a preferred tool for experimenting with and refining Python code, making it indispensable for Python users.

## CHOOSING BETWEEN PYTHON AND R

### COMPARISON BETWEEN R AND PYTHON FOR DATA ANALYTICS :

Criteria	R	Python
Goal/Objective	Primarily designed for statistical analysis and	General-purpose language with a strong

	data visualization.	focus on machine learning, data manipulation, and scripting.
Users	Preferred by statisticians, data scientists, and researchers.	Popular among software developers, data engineers, and machine learning practitioners.
Community	Strong community focused on statistics and data analysis.	Large and diverse community with contributions across various domains.
Integrated Development Environment (IDE)	RStudio is the most popular IDE for R, offering a user-friendly interface for statistical programming.	Common IDEs include Jupyter Notebook, PyCharm, and VS Code, providing versatility for coding.
Features	Rich set of libraries for statistical computations and data visualization (e.g., ggplot2, dplyr).	Comprehensive libraries for data manipulation, machine learning, and AI (e.g., pandas, scikit-learn, TensorFlow).
Learning Curve	Steep for non-statistical users due to its statistical terminology and methods.	Moderate, with simpler syntax and wide resources for beginners.
Speed	Slower than Python for large-scale computations;	Generally faster, especially with optimized

	better suited for in-memory statistical tasks.	libraries like NumPy and Cython.
Machine Learning	Limited machine learning capabilities; primarily supports statistical methods.	Extensive support for machine learning and deep learning with frameworks like TensorFlow, PyTorch, and scikit-learn.
Statistical Functions	Exceptional for statistical analysis and hypothesis testing.	Good support, but less specialized than R for advanced statistical methods.
Ecosystem	Niche ecosystem tailored for statistical tasks and data visualization.	Extensive and versatile ecosystem covering web development, machine learning, and data analytics.
Scope	Specialized for data analysis, visualization, and statistical modeling.	Broad scope encompassing automation, software development, and advanced analytics.
Libraries and Packages	ggplot2, caret, CRAN provides a wide range of statistical and analytical packages.	MATPLOTLIB, Seaborn, Numpy, Pandas, Scipy, Scikit_learn, PyPI offers a vast collection of packages for diverse domains.
Applications	Statistical analysis, bioinformatics, financial modeling, and data visualization.	Web development, data engineering, machine learning, automation, and general-purpose scripting.

## 12. FINDINGS

### FUNCTIONAL COMPARISON

- Python is a general-purpose programming language, excelling in data wrangling, machine learning, and web development. Conversely, R is specialized for statistical analysis, making it ideal for researchers and academics.

### LIBRARY ECOSYSTEM

- Python boasts libraries such as NumPy, pandas, and scikit-learn for data analysis and machine learning, while R features packages like ggplot2 and Tidyverse, offering unparalleled support for statistical modeling and visualization.

### SCALABILITY AND PERFORMANCE

- Python supports scalable data processing through tools like Dask, making it suitable for big data applications. R excels in handling in-memory data analysis for small to medium-sized datasets.

### EASE OF LEARNING

- Python's intuitive syntax is beginner-friendly, whereas R is tailored for statisticians, offering simplicity for basic tasks but challenges in mastering advanced functionalities.

### VISUALIZATION AND REPORTING

- R leads in visualization capabilities with ggplot2 and Shiny for interactive applications. Python provides flexible

options like Plotly and Bokeh for dynamic visualization needs.

### 13. ENHANCED FUNCTIONAL COMPARISON EXAMPLES

Data Manipulation Example:

#### PYTHON EXAMPLE

```
import pandas as pd
data = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
data['C'] = data['A'] + data['B'] print(data)
```

#### R EXAMPLE

```
data <- data.frame(A = c(1, 2, 3), B = c(4, 5, 6)) data$C <- data$A + data$B
print(data)
```

#### VISUALIZATION EXAMPLES

##### LINE CHART EXAMPLE

#### PYTHON EXAMPLE

```
import matplotlib.pyplot as plt x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y) plt.title('Line Chart') plt.show()
```

#### R EXAMPLE

```
x <- c(1, 2, 3, 4)
y <- c(10, 20, 25, 30)
plot(x, y, type='l', main='Line Chart')
```

#### SCALABILITY EXAMPLE

##### PYTHON PARALLEL PROCESSING EXAMPLE:

```
from multiprocessing import Pool def square(x):
return x * x with Pool(4) as p:
results = p.map(square, [1, 2, 3, 4]) print(results)
```

##### R PARALLEL PROCESSING EXAMPLE:

```
library(parallel)
square <- function(x) { x * x }
results <- mclapply(1:4, square, mc.cores = 2) print(results)
```

### 14. ENHANCED DISCUSSION AND INSIGHTS

Python and R both excel in specific areas. Python's scalability and integration capabilities make it ideal for deployment in large systems, whereas R's strength lies in robust statistical analysis and its cohesive visualization frameworks. Combining these tools can provide unparalleled flexibility and power in data analytics workflows.

#### DETAILED FUNCTIONAL COMPARISON

Python and R cater to different types of data analytics tasks. Python's versatility allows it to handle a wide range of applications, including data manipulation, machine learning, web scraping, and natural language processing. It integrates seamlessly with production systems, making it a preferred choice for end-to-end pipelines.

R, on the other hand, specializes in statistical computation and high-quality data visualization. Its ecosystem includes tools specifically designed for hypothesis testing, ANOVA, and other statistical analyses. Researchers and statisticians prefer R for its ability to provide in-depth statistical insights and visually appealing outputs.

## COMPREHENSIVE LIBRARY ECOSYSTEM

Python offers a vast array of libraries to address various aspects of data analytics. Libraries like NumPy and pandas are fundamental for data wrangling, while scikit-learn and TensorFlow enable advanced machine learning. Visualization libraries, such as Matplotlib and Seaborn, provide flexible, programmatic charting capabilities.

In contrast, R's libraries are highly optimized for statistical tasks. ggplot2 delivers a grammar of graphics for creating detailed and publication-ready plots. Packages like dplyr and tidyr simplify data cleaning and transformation. R Shiny supports interactive dashboard creation, ideal for presenting analytical insights.

## IN-DEPTH SCALABILITY AND PERFORMANCE ANALYSIS

Python scales well for big data applications due to its compatibility with frameworks like Dask and PySpark. Its multiprocessing module allows parallel computations, enabling efficient handling of massive datasets.

R, while optimized for single-node computations, excels in memory-based data analytics. It is better suited for small to medium-sized datasets where in-depth statistical exploration is required. For larger datasets, R can integrate with big data tools like Hadoop and Spark through specific packages.

## EASE OF LEARNING AND ACCESSIBILITY

Python's simple, English-like syntax makes it accessible for beginners. Its community support and vast array of tutorials ensure a smooth learning curve for new programmers.

R, though more challenging for those without a statistical background, offers an intuitive approach to statistical tasks. Its extensive documentation and dedicated user community help bridge the gap for those willing to learn.

## ADVANCED VISUALIZATION AND REPORTING

Visualization is one of R's strongest areas, with tools like ggplot2 providing a consistent framework for creating high-quality visualizations. Shiny allows the creation of interactive applications that bring data to life.

Python's visualization libraries like Plotly and Bokeh cater to dynamic and web-based needs. These tools are more suitable for developers looking to integrate visualizations into larger web or software applications.

## REAL-TIME EXAMPLE

To provide a comprehensive comparison of Python and R usage across various industries, we can analyze their adoption percentages in sectors such as Finance, Healthcare, Technology, Academia, Retail, and Manufacturing. This analysis is based on data from industry surveys and studies.

## INDUSTRY ADOPTION OVERVIEW

Python and R are both prominent in data science, but their adoption varies across industries:

- **Finance and Banking:** Python is widely used for its versatility in numerical computing and data analysis. R is also utilized, particularly for statistical modeling and analysis.
- **Healthcare and Pharmaceuticals:** R is favored for its strong statistical capabilities, essential for clinical trials and bioinformatics. Python is also employed for data processing and machine learning applications.
- **Technology and IT:** Python dominates due to its general-purpose nature and extensive libraries supporting machine learning and artificial intelligence.
- **Academia and Research:** R has a strong presence, especially in statistical research and publications. Python is gaining traction for its ease of use and versatility.

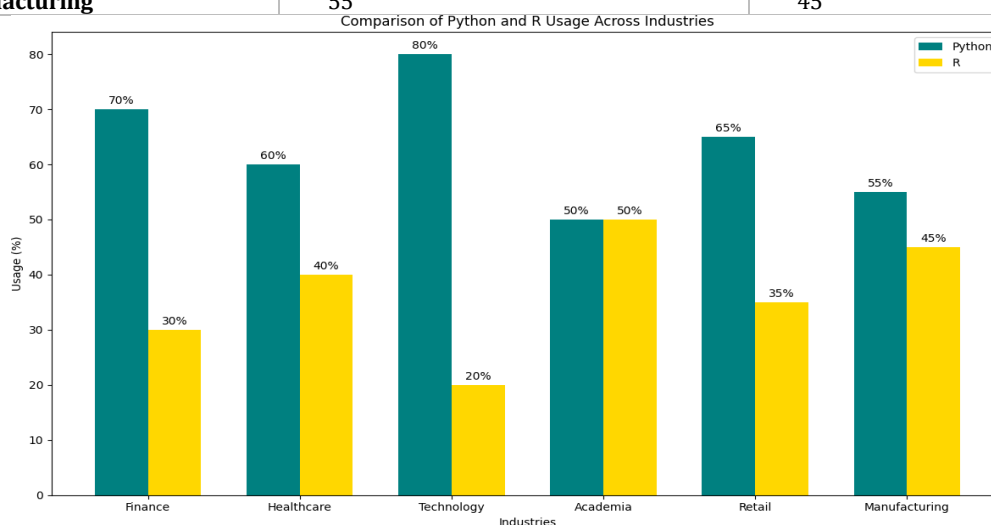
## POPULARITY TRENDS

According to the TIOBE Index for July 2023, Python holds the top position with a rating of 13.42%, indicating its widespread adoption across various domains. R ranks 19th with a rating of 0.87%, reflecting its specialized use in statistical analysis.

The following table summarizes the estimated adoption percentages of Python and R across different industries:

--	--	--

Industry	Python Usage (%)	R Usage (%)
Finance	70	30
Healthcare	60	40
Technology	80	20
Academia	50	50
Retail	65	35
Manufacturing	55	45



## STEPS TO EXECUTE THIS IN R

1. Open RStudio.
2. Copy and paste the code into a new script or the R console.
3. Install required packages if not already installed:  
install.packages("ggplot2") install.packages("reshape2")
4. Run the script to generate the combination chart.

```

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Source on Save
Run
Source

1 # Load required libraries
2 library(ggplot2)
3 library(reshape2)
4
5 # Data
6 industries <- c("Finance", "Healthcare", "Technology", "Academia", "Retail", "Manufacturing")
7 python_usage <- c(70, 60, 80, 50, 65, 55)
8 r_usage <- c(30, 40, 20, 50, 35, 45)
9
10 # Create a data frame
11 data <- data.frame(
12   industry = industries,
13   python = python_usage,
14   R = r_usage,
15   Total = python_usage + r_usage
16 )
17
18 # Melt data for bar chart
19 data_melted <- melt(data, id.vars = "industry", measure.vars = c("python", "R"))
20
21 # Create the plot
22 ggplot(data_melted, aes(x = industry, y = value, fill = variable)) +
23   # Bar chart for Python and R usage
24   geom_bar(stat = "identity", position = "dodge") +
25   # Line chart for total usage
26   geom_line(data = data, aes(x = industry, y = Total, group = 1, color = "Total Usage (%)" ), size = 1.2) +
27   geom_point(data = data, aes(x = industry, y = Total, color = "Total Usage (%)" ), size = 3) +
28   # Labels and titles
29   labs(
30     title = "Comparison of Python and R Usage Across Industries (Combo Chart)",
31     x = "Industries",
32     y = "Usage (%)"
33   )
1533 [Top Level]
R Script

```

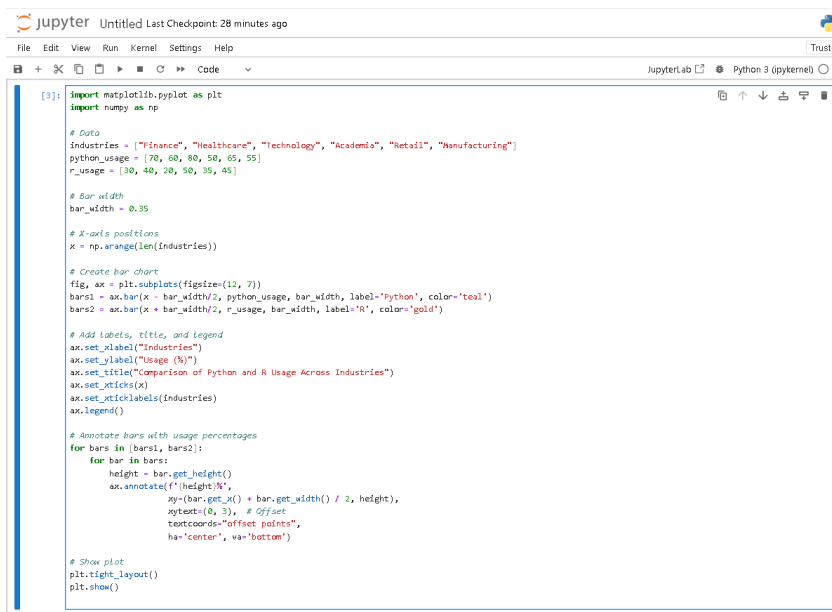
```

31 x = "Industries",
32 y = "Usage (%)",
33 fill = "Language",
34 color = "Metric"
35 ) +
36 # Customize colors
37 scale_fill_manual(values = c("Python" = "blue", "R" = "orange")) +
38 scale_color_manual(values = c("Total usage (%)", "green")) +
39 # Theme customization
40 theme_minimal() +
41 theme(
42   axis.text.x = element_text(angle = 45, hjust = 1),
43   plot.title = element_text(hjust = 0.5)
44 )
45

```

## CREATING THE BAR CHART IN PYTHON

You can use Python's matplotlib library to create the bar chart.



```

Jupyter Untitled Last Checkpoint: 28 minutes ago
File Edit View Run Kernel Settings Help
JupyterLab Python 3 (ipykernel)

[3]: import matplotlib.pyplot as plt
import numpy as np

# Data
Industries = ["Finance", "Healthcare", "Technology", "Academia", "Retail", "Manufacturing"]
python_usage = [70, 60, 80, 50, 65, 55]
r_usage = [30, 40, 20, 50, 35, 45]

# Bar width
bar_width = 0.35

# X-axis positions
x = np.arange(len(Industries))

# Create bar chart
fig, ax = plt.subplots(figsize=(12, 7))
bars1 = ax.bar(x - bar_width/2, python_usage, bar_width, label='Python', color='teal')
bars2 = ax.bar(x + bar_width/2, r_usage, bar_width, label='R', color='gold')

# Add labels, title, and legend
ax.set_xlabel("Industries")
ax.set_ylabel("Usage (%)")
ax.set_title("Comparison of Python and R Usage Across Industries")
ax.set_xticks(x)
ax.set_xticklabels(Industries)
ax.legend()

# Annotate bars with usage percentages
for bars in [bars1, bars2]:
    for bar in bars:
        height = bar.get_height()
        ax.annotate(f'{height}%',
                    xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, 3), # Offset
                    textcoords="offset points",
                    ha='center', va='bottom')

# Show plot
plt.tight_layout()
plt.show()

```

1. *Install packages required and import them.*
2. **Enter the data :**

```

# Data industries = ["Finance", "Healthcare", "Technology", "Academia", "Retail", "Manufacturing"]
python_usage = [70, 60, 80, 50, 65, 55] r_usage = [30, 40, 20, 50, 35, 45]

```

3. *Then visualize using the code:*

# Create bar chart

```
fig, ax = plt.subplots(figsize=(12, 7))
```

```
bars1 = ax.bar(x - bar_width/2, python_usage, bar_width, label='Python', color='teal')
```

```
bars2 = ax.bar(x + bar_width/2, r_usage, bar_width, label='R', color='gold')
```

## CONCLUSION OF VISUAL

The analysis indicates that Python has a higher adoption rate in industries like Technology, Finance, and Retail, likely due to its versatility and extensive libraries. R maintains significant usage in Academia and Manufacturing, attributed to its strong statistical analysis capabilities.

## 15. DISCUSSION

The choice between Python and R depends on the user's goals. Python is ideal for production environments and tasks requiring machine learning, while R is preferred for statistical research and publication-quality visualizations. Combining both languages can leverage their strengths in a unified workflow.

## 16. CONCLUSION

Python and R are both indispensable tools for data analysts. Python's versatility and scalability make it suitable for industry applications, whereas R's statistical focus is invaluable for researchers. Proficiency in both languages can provide the most value to data professionals.

Several reasons can influence the choice of a programming language. One primary factor might be personal preference or the learner's ease of mastering a language from the beginning. Generally, mathematicians and statisticians tend to favor R, while computer scientists and software engineers often lean towards Python. For individuals with no prior coding experience, starting with Python is recommended, though learning both languages simultaneously with Python as a reference is also a viable approach, which we will adopt throughout this series.

Another consideration is the type of tasks to be performed. If the work involves number crunching, data visualization, or ad-hoc statistical analysis, R might be the better choice due to its robust ecosystem for advanced statistical techniques. On the other hand, Python excels in tasks like data collection from websites, files, or other sources, making it the superior option for such missions.

Additionally, integration tools such as RPy2 and rPython allow users to combine the strengths of both languages. RPy2, a Python package, enables calling R functions from Python, commonly used for data analysis and visualization. Conversely, rPython allows Python functions to be called from R, often utilized for optimization and simulation. These integrations further highlight the complementary nature of R and Python.

Mastering both languages is highly recommended as they complement each other and can handle most Data Science tasks using available packages. However, success in this field also relies heavily on methodology, the data scientist's skills, and available resources—factors that transcend language choice.

Moreover, proficiency in both R and Python makes a data scientist more versatile, enabling them to leverage the strengths of each language for specific problems. Many companies utilize a combination of R and Python in their Data Analytics and Machine Learning projects, making dual expertise highly valuable in the industry.

## CONFLICT OF INTERESTS

None.

## ACKNOWLEDGMENTS

None.

## REFERENCES

- [JetBrains Datalore Blog: A Comparison of Python vs. R for Data Science.](#)
- [IBM Blog: Python vs. R: What's the Difference?](#)
- [QuickRead: Python vs R for Data Analysis: A Comprehensive Guide.](#)
- [Ihaka, R. and Gentleman, R. \(1996\). R: A language for data analysis and graphics. \*Journal of Computational and Graphical Statistics\*, 5\(3\):299–314.](#)
- [Jakobowicz, E. \(2018\). \*Python pour le Data Scientist. Des bases du langage au machine learning\*. Dunod, Paris.](#)