Programming Language Popularity Trends

Gulchin Taghizade  
 Computer Science & Networking  
 Wentworth Institute of Technology  
Boston MA, USA  
 [taghizadeg@wit.edu](mailto:taghizadeg@wit.edu)

ABSTRACT

The rapid evolution of technology has led to dynamic shifts in the popularity of programming languages, influencing developer preferences, industry trends, and economic outcomes. This study analyzes historical trends in programming language popularity over the past decade, identifies emerging languages, and explores how these trends vary across industries and regions. Using data from the Stack Overflow Developer Survey 2024 and a Kaggle dataset, and employing linear regression and random forest regression models, the study provides data-driven insights into key questions, including language preferences based on developer experience, regional differences in language usage, and factors contributing to the rise of new languages. The findings offer valuable perspectives for developers, educators, and industry leaders looking to navigate the evolving landscape of software development.

KEYWORDS

Programming Languages, Popularity Trends, Developer Preferences, Data Analysis, Software Development.

1 Introduction

Programming languages are fundamental tools in the software development process, enabling developers to create applications, systems, and technologies that shape the modern world. Over the past decade, the popularity of programming languages has shifted dramatically, influenced by various factors such as technological advancements, industry demands, and developer preferences. This study explores the evolving trends in programming language popularity, specifically analyzing the historical and current usage patterns of major programming languages. The primary focus is on understanding how these languages’ popularity changes over time, identifying emerging languages, and examining the factors driving their adoption across different industries and regions.

The importance of this topic lies in its implications for developers, educators, and organizations. Understanding programming language trends allows developers to make informed decisions about which languages to learn and use, helping them remain competitive in an ever-changing job market. For educators, recognizing the most in-demand languages can inform curriculum development, ensuring that students acquire the skills needed for the workforce. From an industry perspective, organizations can benefit from aligning their technology stack with the most popular and efficient programming languages, optimizing productivity and development costs.

Current research in this field primarily focuses on language popularity through surveys and analyses of developer behavior. For example, the Most Popular Programming Languages 2004-2024 (Kaggle Dataset) offers valuable insights into the most commonly used programming languages, while studies such as those by Tufano et al. [1] and Pizlo et al. [2] have explored the relationship between language usage and factors like developer experience and industry applications. While these studies have made significant contributions, there is still a need for more comprehensive, data-driven insights that not only track language popularity but also explain the reasons behind these trends. Furthermore, understanding the regional and industry-specific adoption of languages can provide a more nuanced view of language trends. This study seeks to fill this gap by integrating various datasets, including the Stack Overflow Developer Survey 2024 and Kaggle Dataset- Most Popular Programming Languages 2004-2024, and applying machine learning techniques to provide deeper insights into programming language usage.

2 Data

This section introduces the datasets used in the analysis, including their sources, characteristics, and any preprocessing steps taken.

2.1 Source of dataset

Two datasets were used for this analysis:

**2.1.1. Most Popular Programming Languages 2004-2024 (Kaggle Dataset):** This dataset was used to answer the first question related to historical trends in programming language popularity. It provides programming language usage trends from 2004 to 2024, including the popularity index and usage percentage over time.

• **Dataset Name:** Most Popular Programming Languages 2004-2024

• **Source:** Kaggle

• **Creator:** Muhammad Roshan Riaz

• **Dataset Generation Date:** 2024

• **Generation Method:** The data was collected from developer surveys and other industry sources, reflecting the trends in programming language popularity from 2004 to 2024.

**2.1.2. Stack Overflow Developer Survey 2024:** This dataset was used to answer the remaining questions, which focus on developer preferences, language experience, and salary trends. The Stack Overflow Developer Survey 2024 contains comprehensive data about developers’ demographics, programming language usage, career experience, education, and salaries.

**• Dataset Name:** Stack Overflow Developer Survey 2024

**• Source:** Stack Overflow Developer Survey 2024

**• Creator:** Stack Overflow

**• Dataset Generation Date:** 2024

**• Generation Method:** The dataset was generated based on a survey conducted by Stack Overflow, which collected responses from developers globally. The survey includes data on developer experience, languages used, job satisfaction, and more.

2.2 Characters of the datasets

**2.2.1 Kaggle Dataset - Most Popular Programming Languages 2004-2024**

**• Format:** CSV

**• Size:** The dataset contains 250 rows and 11 columns.

The following table summarizes the columns in this dataset:

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Description** | **Unit/Format** |
| **Month** | The time period for which the programming language usage is recorded. | Date (YYYY-MM) |
| **Programming Language** | The name of the programming language | Text (e.g., Python Worldwide(%)) |

For the analysis of this dataset I did the following steps:

• **Converted the ‘Month’ column to datetime format:**

To facilitate time-series analysis, the Month column was converted to a datetime object using the following rule:

*data['Month'] = pd.to\_datetime(data['Month'], format='%Y-%m')*

**• Filtered data for the specified years (2014–2024):**

I filtered out data outside the range of 2014 to 2024 to focus on the last decade’s trends:

*data\_filtered = data[(data['Month'].dt.year >= 2014) & (data['Month'].dt.year <= 2024)]*

• **Aggregating monthly data into yearly averages** can be considered a derived metric that facilitates trend analysis over time. This was achieved using:

*heatmap\_data = data\_filtered.resample('Y').mean()*

While no explicit new categories were created, the analysis focused on major programming languages of interest, which were defined as:

*major\_languages = ['Python Worldwide(%)', 'JavaScript Worldwide(%)', 'Java Worldwide(%)', 'C# Worldwide(%)']*

• **Renamed columns to remove unnecessary text:**

Column names were cleaned to make them more readable by removing the “Worldwide(%)” suffix:

*heatmap\_data.columns=heatmap\_data.columns.str.replace(r' Worldwide\(%\)', '', regex=True)*

**2.2.2 Stack Overflow Developer Survey 2024**

**• Format:** CSV

**• Size:** The dataset contains 65437 rows and 114 columns.

Below is a table summarizing key columns, their descriptions, and units:

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Description** | **Unit/Format** |
| **ResponseId** | Unique identifier for each respondent. | Numeric |
| **MainBranch** | Developer’s primary branch (e.g., professional developer, student). | Text |
| **Age** | Age range of the respondent. | Text (e.g., “18-24 years old”) |
| **Employment** | Employment type (e.g., full-time, part-time, student). | Text |
| **EdLevel** | Highest education level attained. | Text |
| **YearsCode** | Total years of coding experience. | Numeric |
| **DevType** | Developer roles (e.g., full-stack developer, data scientist). | Text (comma-separated for multiple roles). |
| **LanguageHaveWorkedWith** | Programming languages the respondent has worked with. | Text (comma-separated for multiple languages). |
| **LanguageWantToWorkWith** | Programming languages the respondent wants to learn/work with. | Text (comma-separated for multiple languages). |
| **ConvertedCompYearly** | Annual total compensation in USD. | Numeric |
| **Country** | The geographic location of the respondent | Text |

**Handling Missing Data:**

**•** The code processes columns with possible missing values. Specifically, it handles the LanguageHaveWorkedWith and LanguageWantToWorkWith columns by dropping any NaN values:

*current\_lang = data['LanguageHaveWorkedWith'].dropna().str.split(';').explode()*

*aspirational\_lang = data['LanguageWantToWorkWith'].dropna().str.split(';').explode()*

• The code drops rows with missing values in both the LanguageHaveWorkedWith and DevType columns to ensure the analysis only includes rows with complete data for both variables:

*exploded\_data = data.dropna(subset=['LanguageHaveWorkedWith', 'DevType'])*

• Rows with missing values in either the ConvertedCompYearly (yearly salary) or LanguageHaveWorkedWith (languages worked with) columns are dropped to ensure the dataset only includes complete data for both variables:

*exploded\_data = stackoverflow\_data.dropna(subset=['ConvertedCompYearly', 'LanguageHaveWorkedWith'])*

**Data Conversion (Splitting Strings and Exploding):**

• The LanguageHaveWorkedWith column contains multiple languages per row, separated by semicolons. To analyze the data at a language level, the str.split(';') method splits each language, and explode() creates separate rows for each language:

*exploded\_data = data.explode('Languages')*

**Data Type Conversion:**

• The YearsCode column is converted to a numeric format, with errors coerced into NaN values:

*exploded\_data['YearsCode'] = pd.to\_numeric(exploded\_data['YearsCode'], errors='coerce')*

• Any rows where YearsCode is NaN (after coercion) are removed:

*exploded\_data = exploded\_data.dropna(subset=['YearsCode'])*

**Combining Metrics:**

• The frequency (popularity) and average years of experience for each language were combined into a single DataFrame:

*language\_data = pd.DataFrame({*

*'Popularity': language\_popularity,*

*'Average\_Years\_Experience': language\_experience*

*})*

**Mapping:**

• A dictionary (industry\_mapping) is used to map specific job titles in the DevType column to broader industry categories. For example, roles like “Developer, full-stack” are mapped to “Web Development”:

*industry\_mapping = {*

*'Developer, full-stack': 'Web Development',*

*'Developer, back-end': 'Web Development',*

*'Developer, front-end': 'Web Development',*

*# More mappings...*

*}*

**New Categories Created:**

• The Industry column is created by mapping job titles from the DevType column to broader industry categories using the industry\_mapping dictionary. This new category groups roles into categories like Web Development, Data Science, Operations, etc.

*exploded\_data['Industry'] = exploded\_data['DevType'].map(industry\_mapping)*

3 Methodology

3.1 Random Forest Regressor

**Method:** The **Random Forest Regressor** is an ensemble learning method used for regression tasks. It combines multiple decision trees to make predictions. Each tree in the forest makes an individual prediction, and the final output is the average of all individual predictions. Random Forest is highly effective for capturing non-linear relationships between the independent and dependent variables.

**Assumptions:**

• Random Forest does not assume any specific form for the relationship between the variables. It can model both linear and non-linear relationships.

• It assumes that the data has a variety of features, allowing it to build multiple decision trees and capture interactions between features.

**Advantages:**

• It handles both linear and non-linear relationships well.

• It is resistant to overfitting, particularly when there is sufficient data and a reasonable number of trees.

• It is capable of handling large datasets with high-dimensional feature spaces.

• It can automatically model interactions between variables, which is helpful when analyzing complex datasets.

**Disadvantages:**

• Computationally intensive, especially as the number of trees (estimators) increases.

• Interpretation of the model is challenging, as it involves numerous decision trees.

• The model may require tuning of hyperparameters, such as the number of trees or depth of each tree, for optimal performance.

**Why Chosen:** The Random Forest Regressor was chosen because of its robustness in handling complex datasets with both linear and non-linear relationships. It does not require strong assumptions about the data, making it a flexible and powerful model for this analysis.

**Optional Adjustments:**

• Hyperparameters such as the number of estimators (n\_estimators) were adjusted to ensure a sufficient number of trees for a robust model. A value of 100 estimators was chosen as a balance between computational efficiency and model accuracy.

• The random state was fixed to ensure reproducibility of results.

A graph with red and blue dots

Description automatically generated

Figure 1: Random Forest Regression Model’s visualization

3.2 Linear Regression

**Method**: **Linear Regression** is a statistical method used to model the relationship between a dependent variable and one or more independent variables. It assumes a linear relationship between the dependent and independent variables and fits a straight line to the data that minimizes the sum of squared errors (residuals).

**Assumptions**:

• The residuals (errors) from the linear model are normally distributed.

• There is no multicollinearity among the independent variables.

• The data is homoscedastic, meaning the variance of the errors is constant across all levels of the independent variable.

**Advantages**:

• Simple, easy to interpret, and computationally efficient.

• Provides a clear insight into the relationship between the dependent and independent variables

• Well-suited for smaller datasets or when there is a clear linear trend.

**Disadvantages**:

• Assumes a linear relationship, which may not be suitable for complex, non-linear datasets.

• Sensitive to outliers, which can significantly affect the model’s predictions and fit.

• May underperform when there are interactions between variables that cannot be captured by a linear model.

**Why Chosen:** Linear Regression was chosen for its simplicity and ease of interpretation. Given the focus on understanding the basic relationship between language rank and salary, Linear Regression provides a clear and interpretable output.

**Optional Adjustments:**

• The dataset was split into training and test sets to evaluate the model’s performance on unseen data.

• Evaluation metrics, such as mean squared error (MSE) and R-squared, were used to assess the model’s accuracy and goodness of fit.

A graph with red and blue dots

Description automatically generated

Figure 2: Linear Regression Model’s visualization

**Python Module/Function:** The Scikit-learn library was used to implement Linear Regression and Random Forest Regressor. Specifically, the LinearRegression and RandomForestRegressor functions were applied, with the dataset being split using train\_test\_split from sklearn.model\_selection. For data manipulation and analysis, Pandas [3] was used, while NumPy [3] was applied for some statistical calculations. Visualizations were created using Seaborn [4] and Matplotlib [5] libraries. Additionally, pycountry\_convert was used for geographic analysis, helping to group countries by continent and visualize regional trends.

4 Results

The analysis reveals interconnected trends in programming language popularity, industry applications, regional preferences, developer experience, and salaries. These aspects not only offer distinct insights but also highlight significant relationships between them.

Over the past decade, the popularity of programming languages has evolved dynamically. Python’s rapid rise and peak in 2022 coincides with its extensive adoption in data science, machine learning, and web development—industries that have seen exponential growth in recent years. Similarly, JavaScript’s stable popularity reflects its indispensable role in front-end and full-stack development, which remain core to web and app development worldwide. Meanwhile, the declining trends in Java and C# suggest a gradual shift from traditional enterprise-level development to newer, more versatile technologies.

Interestingly, current trends demonstrate how developer experience influences language usage. Less experienced developers gravitate toward modern, high-demand languages like Python and TypeScript due to their accessibility and extensive resources. In contrast, experienced developers often continue to use legacy languages such as C and PHP, which, while declining in popularity, remain critical for maintaining older systems.

Regionally, JavaScript emerges as the most popular language across all continents, underscoring its global dominance in web development. However, regional variations are evident in the declining use of C and PHP, likely driven by differing industry focuses and market demands. Despite these variations, the universal appeal of modern languages like JavaScript and Python aligns with the growing need for scalable, flexible solutions in global industries.

The analysis of average developer salaries further interconnects these findings. Languages like Objective-C, Crystal, and Elixir, though less popular, are associated with the highest salaries. This indicates that rare and specialized languages can command a premium in industries such as mobile app development, high-performance computing, and niche frameworks. Conversely, widely used but less specialized languages like PHP and Dart correlate with lower salaries, reflecting their reduced demand and commoditized nature.

When viewed together, these findings demonstrate how language popularity, industry requirements, regional trends, developer experience, and salary potential are deeply intertwined. Modern, versatile languages dominate usage across industries, while niche languages offer lucrative opportunities for developers with specialized expertise. Understanding these relationships provides valuable insights for developers navigating career choices and organizations making technology investments.

5 Discussion

While the project provides meaningful insights into programming language trends, developer preferences, and salary correlations, certain limitations and areas for improvement should be addressed. These shortcomings stem from both the methodology and the scope of the analysis.

One of the primary limitations lies in the dataset itself. The data is self-reported, meaning it may not fully capture global developer trends, especially in regions with lower survey participation. Additionally, while the dataset included education level and learning methods, their impact on programming language popularity was not analyzed in depth. For example, investigating how different education levels (formal education versus self-learning) or specific learning platforms influence language adoption could provide deeper insights. Future work should examine these relationships in detail to better understand how developers’ educational backgrounds and learning resources shape their language preferences.

Regarding the machine learning models used, both the Random Forest Regressor and Linear Regression models were applied to predict average developer salaries based on language rankings. The Random Forest Regressor outperformed Linear Regression, as evidenced by its lower mean squared error (MSE) and higher R-squared value. This suggests that Random Forest is better suited for capturing complex, non-linear relationships between programming language rank and salary. However, even the Random Forest model may benefit from further tuning of hyperparameters to optimize performance.

Despite its advantages, the Random Forest model also has limitations. It lacks interpretability compared to Linear Regression, which makes it harder to explain how specific factors influence the predictions. In future work, incorporating additional models, such as Gradient Boosting or Neural Networks, could enhance both prediction accuracy and insight extraction.

Another notable limitation in this work is the regional grouping of programming language popularity. For simplicity, countries were grouped by continent, and the analysis showcased the top 10 programming languages per continent. While this provided a general view, it may have masked significant variations within individual countries. Future studies could analyze trends on a per-country basis, allowing for a more granular understanding of how cultural, economic, and industry-specific factors shape programming language adoption.

To improve the results in future studies, the inclusion of additional factors—such as deeper analyses of education, learning methods, and industry-specific preferences—would provide a more nuanced understanding of programming language trends. Moreover, analyzing the intersection of developer demographics and industry needs could offer a more targeted perspective on the factors driving programming language adoption. Expanding the dataset to include more recent data and a broader geographic representation would also strengthen the validity of the findings.

In conclusion, while this project provides valuable insights, addressing these limitations and incorporating broader influencing factors will enable a more detailed and accurate understanding of programming language popularity and its relationship to developer preferences and career outcomes.

6 Conclusion

This project examined the trends and factors driving the popularity of programming languages using data from the Stack Overflow Developer Survey and Most Popular Programming Languages 2004-2024 (Kaggle Dataset). Key findings showed that Python has gained significant popularity over the last decade, with JavaScript remaining stable and other languages like Java and C# declining. The analysis also revealed that more experienced developers tend to use less popular languages. In terms of industries, languages like JavaScript, HTML/CSS, and SQL were widely used. Additionally, languages such as Objective-C and Elixir offered higher salaries, while languages like PHP and Dart had lower earnings.

Two models were used to predict the relationship between programming language rank and average salary: Linear Regression and Random Forest Regression. After comparing the performance of both models, it was clear that the Random Forest model provided a better fit, as evidenced by its lower Mean Squared Error and higher R-squared value.

These findings offer valuable insights for developers and organizations, helping them understand language trends, make informed decisions about career paths, and strategically select technologies for their projects. The results also contribute to a better understanding of how various factors, including experience and industry use, influence programming language popularity and salary potential.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my instructor for her guidance and support throughout this project. Her feedback and expertise greatly contributed to shaping the direction and depth of this analysis.

Additionally, I am thankful for the availability of the Stack Overflow Developer Survey and Kaggle dataset by Muhammad Roshan Riaz dataset, which provided a comprehensive foundation for this research.

REFERENCES

[1] M. Tufano, D. Lo, A. Sillitti, and F. S. De Moura, “An empirical study on the impact of programming language on software quality,” ACM SIGSOFT Software Engineering Notes, vol. 42, no. 5, pp. 1-9, 2017.

[2] F. Pizlo, D. N. Smith, and L. A. V. Smith, “Understanding the rise of new programming languages,” Journal of Software Engineering and Applications, vol. 12, no. 4, pp. 235-245, 2019.

[3] McKinney, W., Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython, O’Reilly Media, Inc., 2017.

[4] Waskom, M., seaborn: statistical data visualization, The Journal of Open Source Software, vol. 6, no. 60, p. 3021, 2021. doi: 10.21105/joss.03021.

[5] Hunter, J. D., Matplotlib: A 2D graphics environment, Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007. doi: 10.1109/MCSE.2007.55.

Conference Name:ACM Woodstock conference

Conference Short Name:WOODSTOCK’18

Conference Location:El Paso, Texas USA

ISBN:978-1-4503-0000-0/18/06

Year:2018

Date:June

Copyright Year:2018

Copyright Statement:rightsretained

DOI:10.1145/1234567890

RRH: F. Surname et al.

Price:$15.00