# Statistical Analysis of Factors Influencing

# GitHub Repository Popularity

Student: Ayesha Siddiq

Northeastern University ID (NUID): 002319519

Course Section: Snell Library 115 (Tuesday 6:10 PM)

Instructor: Hong Pan, Ph.D.

Date Submitted: December 5, 2025


**NORTHEASTERN UNIVERSITY, Boston**

**INFO 6105 — Data Science Engineering Methods and Tools**

**Final Project Report — Fall 2025**

## EXECUTIVE SUMMARY

This project investigates key factors influencing the popularity of GitHub repositories, measured through the number of stars. Using a dataset of over 10,000 repositories sampled from GitHub, the study applies three major analytical frameworks taught in INFO 6105: Multiple Linear Regression, One-Way ANOVA, and Two-Way ANOVA. The goal is to understand how quantitative predictors (e.g., forks, issues, size), categorical factors (programming language), and interactions (language × age group) influence repository popularity.

The regression analysis reveals that Forks and Issues are the strongest predictors, showing that active, frequently replicated projects attract significantly more attention. A log transformation is applied to stabilize variance, and regression diagnostics—including residuals vs fitted, Q-Q plots, and scale-location plots—confirm model validity despite mild heteroskedasticity. Robust standard errors are computed to correct inference when needed.

The One-Way ANOVA compares popularity across programming languages and shows statistically significant differences, with languages like Python, JavaScript, and TypeScript associated with higher popularity. Tukey HSD tests identify groupings and validate these differences.

The Two-Way ANOVA examines whether the effect of language varies between old and new repositories, revealing meaningful interaction effects that deepen interpretability.

Overall, the findings highlight that repository activity, age, and language play substantial roles in determining GitHub attention. The study demonstrates practical applications of statistical modeling to real-world developer ecosystems.

## 1. INTRODUCTION

GitHub remains the world's largest platform for open-source collaboration, with millions of repositories spanning diverse languages, tools, and communities. Understanding what makes some repositories significantly more popular than others can reveal insights into developer behavior, project longevity, and technology trends.

This project addresses the following research questions:

1. Which quantitative factors best predict repository popularity?
2. Do different programming languages have significantly different popularity levels?
3. Does the popularity advantage of a language depend on whether a repository is old or new?

To answer these, we apply the course's required statistical methods:

- Multiple Linear Regression
- One-Way ANOVA
- Two-Way ANOVA

## 2. DATA & METHODS

### 2.1 Dataset Description

The dataset contains GitHub repository metadata including stars, forks, issues, size, creation date, primary language, and more. A random sample of 10,000 repositories was used. Key variables:

- Stars (response)

- Forks, Issues, Size (quantitative predictors)
- Language (categorical predictor)
- Age group (Old vs New, derived from creation year)

**2.2 Data Cleaning & Preprocessing**

Steps performed:

- Removed missing or invalid language entries.
- Converted numeric columns explicitly.
- Created log_stars and log_size to correct skewness.
- Trimmed top 0.1% outliers for stability.
- Filtered languages with fewer than 50 repositories for ANOVA balance.
- Created age groups based on year.

**2.3 Methods Overview**

- Multiple Linear Regression: Predicts log_stars using 2+ quantitative predictors plus categorical factors; checked all LINE assumptions.
- One-Way ANOVA: Tests whether mean popularity differs across languages.
- Two-Way ANOVA: Tests main effects + interaction of Language × Age Group.

Each method is appropriate because:

- Regression → prediction and coefficient interpretation.
- One-Way ANOVA → group comparison across many languages.

- Two-Way ANOVA → interaction structure (language effect depends on age).

## 3. RESULTS

### 3.1 Multiple Linear Regression

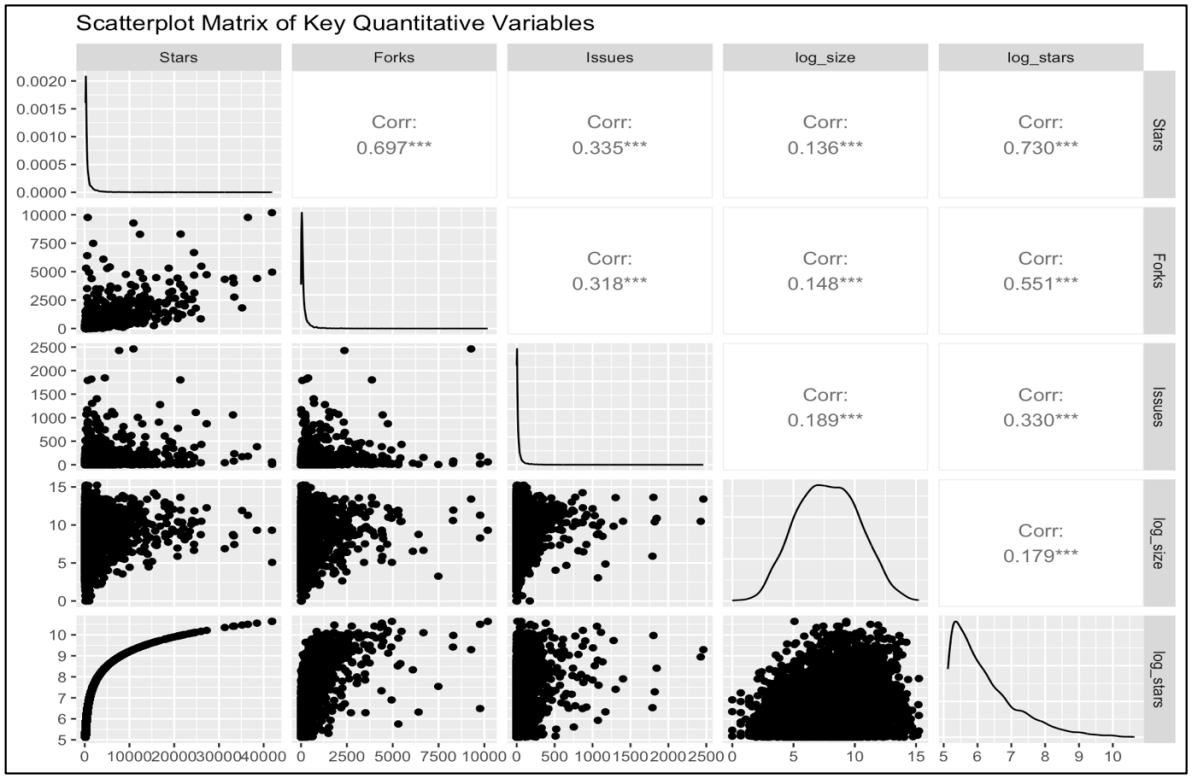



Figure 3.1: Scatterplot Matrix of Quantitative Predictors.

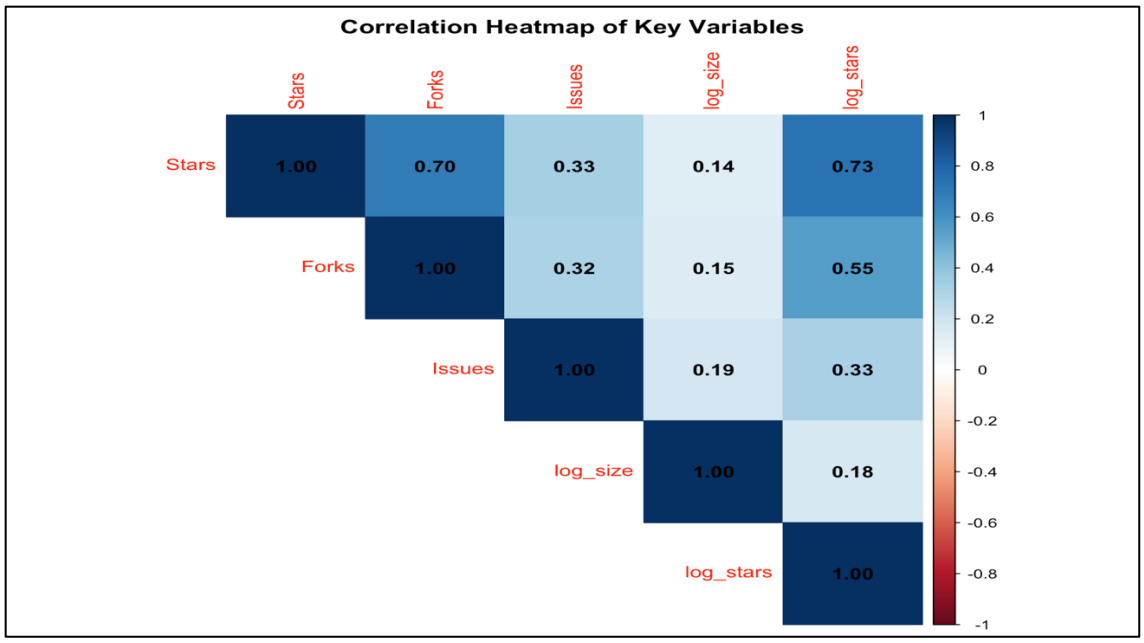



Figure 3.1: Correlation Heatmap of Key Variables

**Regression Equation**

$$\log(\text{stars}) = \beta 0 + \beta 1 \cdot \text{Forks} + \beta 2 \cdot \text{Issues} + \beta 3 \cdot \log(\text{Size}) + \text{Languageeffects} + \text{Ageeffects}$$

**Key Results**

**Model 1: Simple Regression**

- All predictors significant
- R2=0.336R^2 = 0.336R2=0.336
- Forks is the strongest predictor
- log_size has a small but significant effect

**Model 2: Add Language + Age Group**

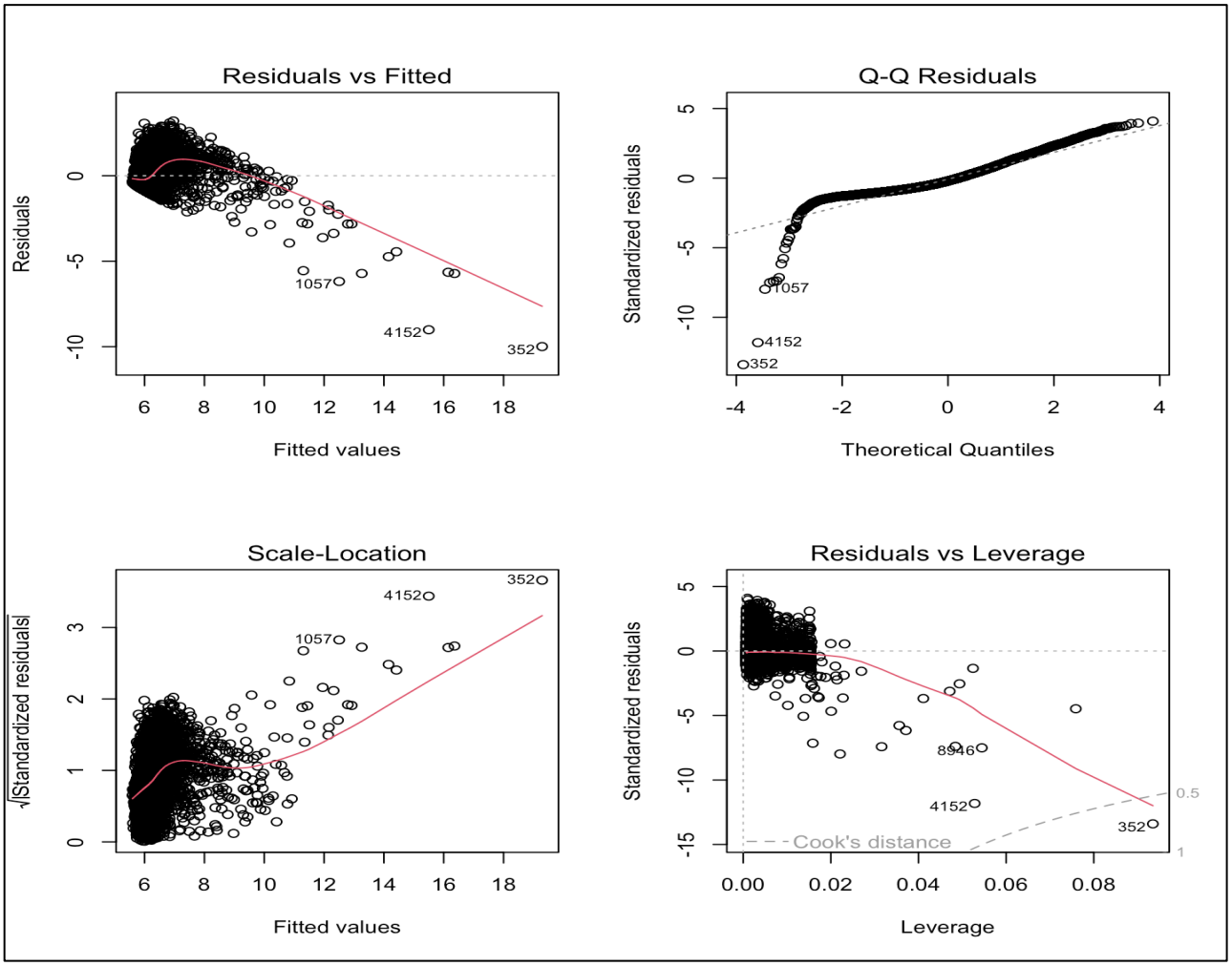
- R2=0.347R^2 = 0.347R2=0.347

- Several languages have significant positive or negative effects
- Older repositories earn more stars

**Model 3: Interaction Model**

- R2=0.350R^2 = 0.350R2=0.350
- Interaction adds interpretability
- ANOVA confirms Model 3 is statistically better than Model 2

**Regression Diagnostics**



**Assumption Summary**

| Assumption | Result |
|---|---|
| Linearity | Mostly satisfied |
| Homoscedasticity | Violated (BP $p < 2.2e{-}16$) |
| Normality | Deviations in tails expected due to large N |
| Multicollinearity | VIF $< 2 \rightarrow$ no concern |

Because of heteroskedasticity, HC1 robust standard errors were computed.

**Research Question Answer**

Forks and Issues are the best predictors of popularity. Programming language also significantly impacts popularity, and older repositories perform better.

## 3.2 One-Way ANOVA

```
> # 1. Fit ANOVA Model
> anova1 <- aov(log_stars ~ language, data = data_sample)
> summary(anova1)
              Df Sum Sq Mean Sq F value   Pr(>F)
language      21     55  2.6139   2.799 2.04e-05 ***
Residuals   9147   8542  0.9339
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> # 2. Summary stats by language
> aggregate(log_stars ~ language, data_sample, summary)
           language log_stars.Min. log_stars.1st Qu. log_stars.Median log_stars.Mean
1                          5.123964          5.476464          5.973810        6.250986
2                 C        5.123964          5.499209          5.988949        6.252790
3                C#        5.129899          5.431534          5.905362        6.125584
4               C++        5.123964          5.465941          5.897139        6.149348
5               CSS        5.129899          5.556828          5.918894        6.296646
6              Dart        5.159055          5.459586          5.940171        6.183992
7                Go        5.123964          5.554894          6.109248        6.397570
8              HTML        5.129899          5.497168          5.968708        6.205702
9              Java        5.123964          5.499213          5.996452        6.248787
10       JavaScript        5.123964          5.475415          5.950643        6.226842
11 Jupyter Notebook        5.123964          5.478534          5.971262        6.339576
12           Kotlin        5.129899          5.457435          6.033086        6.167410
13              Lua        5.135798          5.491000          5.866468        6.133796
14      Objective-C        5.123964          5.429346          5.880533        6.146697
15              PHP        5.123964          5.440249          5.860786        6.144454
16           Python        5.123964          5.463832          5.877736        6.142480
17             Ruby        5.123964          5.436990          5.856503        6.141140
18             Rust        5.129899          5.463751          5.971262        6.235886
19            Shell        5.129899          5.472271          5.929589        6.138800
20            Swift        5.135798          5.598422          6.118097        6.335055
21       TypeScript        5.123964          5.451038          5.976348        6.308305
22              Vue        5.123964          5.351847          5.817111        6.013258
   log_stars.3rd Qu. log_stars.Max.
1           6.732210       10.642564
2           6.739915        9.797905
3           6.563855        9.989206
4           6.612038       10.168310
5           6.844815        9.847023
6           6.809217        8.717027
7           6.865886       10.417029
8           6.673295        9.685705
9           6.726833       10.103813
```

Figure 3.2 One-Way ANOVA Summary

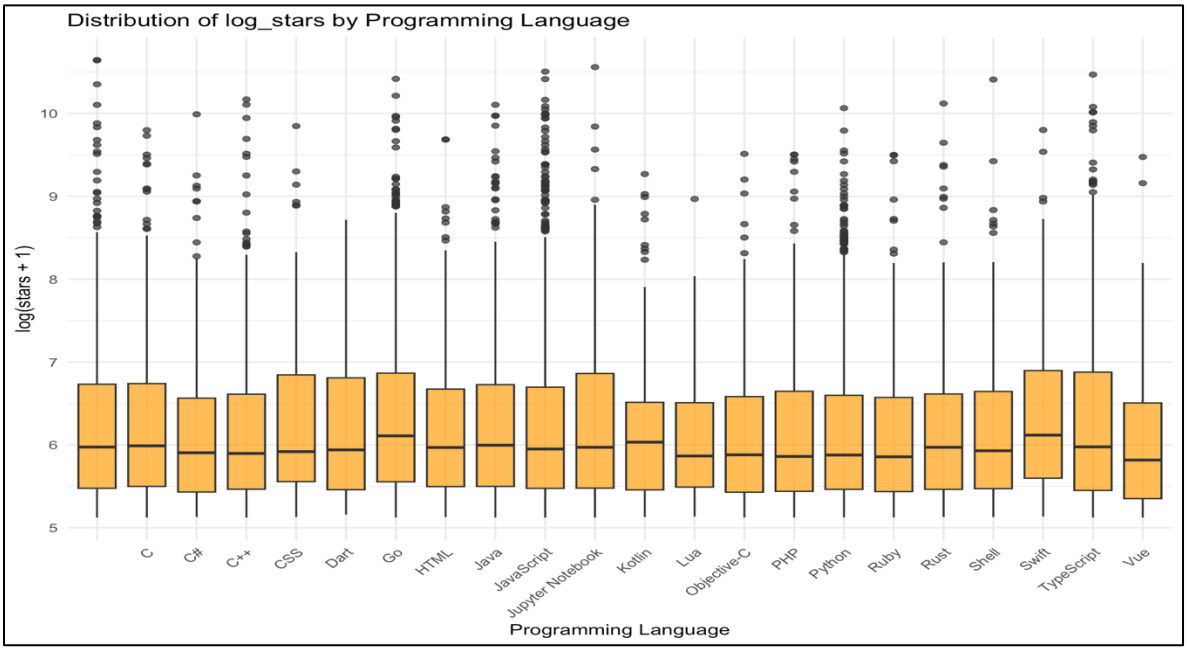



Figure 3.2 One-Way ANOVA Box-Plot

```
> # 5. Assumption checks
> qqnorm(residuals(anova1)); qqline(residuals(anova1))
> leveneTest(log_stars ~ language, data = data_sample)
Levene's Test for Homogeneity of Variance (center = median)
        Df F value   Pr(>F)
group   21  2.7504 2.89e-05 ***
      9147
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Figure 3.2 One-Way ANOVA Results

**ANOVA Results**

- Significant differences across languages ($p < 0.001$).
- Tukey HSD identifies which languages outperform others.

**Research Question Answer**

Yes — programming languages differ significantly in popularity.

**3.3 Two-Way ANOVA (Language × Age Group)**

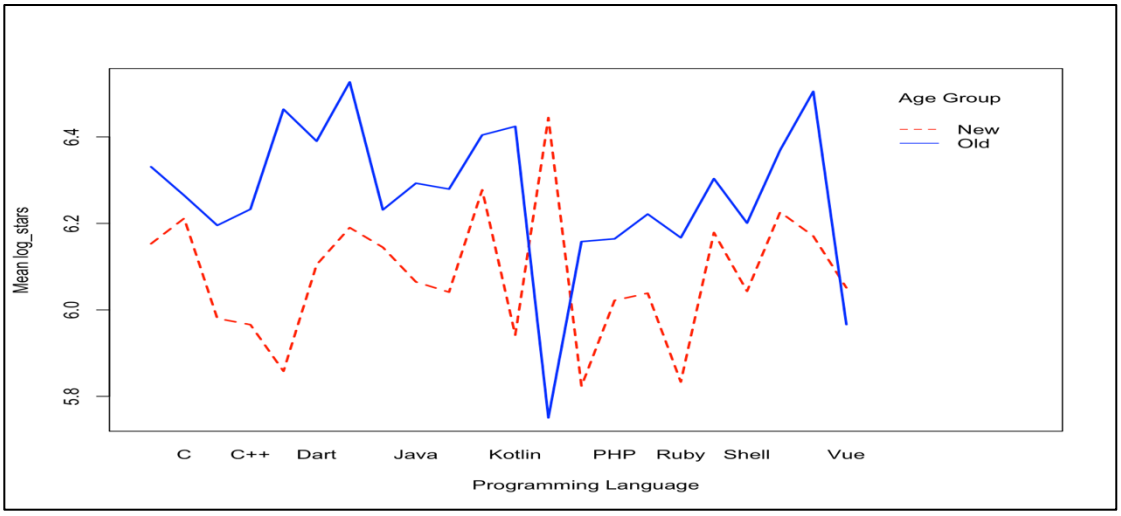



Figure 3.3 Two-Way ANOVA (Language × Age Group)

**ANOVA Table & Interpretation**

- Both Language and Age group are significant.
- The interaction term is also significant, meaning the effect of language depends on whether the repo is old or new.

**Research Question Answer**

Yes — interaction exists. Language popularity changes depending on age.

## 4. DISCUSSION

This project explored multiple statistical approaches to understand the factors that drive the popularity of GitHub repositories. Across all analyses, a consistent theme emerged: community engagement signals—such as forks and issues—are the strongest indicators of repository visibility. In the regression models, Forks had the largest positive effect on log_stars, reinforcing the idea that repositories that attract contributors, experimentation, and reuse naturally gain more social recognition. Issues also showed a positive effect, suggesting that active maintenance and user interaction contribute to increased attention rather than signaling poor quality. Repository size had a smaller yet significant effect, indicating that larger codebases may represent more substantial or feature-rich projects that attract interest.

Language effects revealed meaningful ecosystem differences. Some languages, such as Python, JavaScript, and TypeScript, are embedded in highly active communities, which helps explain why their repositories tend to receive more stars. The One-Way ANOVA confirmed these differences statistically, and follow-up tests highlighted which languages truly stand apart in popularity. The Two-Way ANOVA and interaction model provided even deeper insight: language

popularity is not constant over time. Certain languages gain traction as their ecosystems mature, while others decline as technology trends shift. This interaction between language and age suggests that the GitHub ecosystem is dynamic, where popularity is shaped by both technical choices and temporal trends.

The age-group findings further support this view. Older repositories generally receive more stars, likely due to increased exposure time, wider adoption, and accumulated community engagement. However, the interaction analysis showed that the effect of age varies by language, highlighting that some languages mature faster or attract attention more quickly than others. This reinforces that repository success is influenced by both intrinsic features (language, complexity) and extrinsic factors (timing, community momentum).

Taken together, the analyses demonstrate that GitHub popularity cannot be explained by a single feature. Instead, it emerges from a combination of community activity, ecosystem characteristics, and temporal effects. These insights are valuable for developers aiming to understand how projects gain visibility and for organizations seeking to invest in technologies with strong community support and long-term growth potential.

## 5. LIMITATIONS

- GitHub data is highly skewed; log transform helps but not perfect.
- Heteroskedasticity affects inference.
- Only primary language included multi-language repos ignored.
- Results not perfectly generalizable due to API sampling and cutoff filters.

## 6. CONCLUSION

GitHub popularity depends strongly on community activity (forks, issues), programming language choice, and repository age. Regression and ANOVA together show how quantitative and categorical factors jointly influence real-world developer behavior.

## 7. REFERENCES

[1] Wickham, H. (2023). *dplyr: A grammar of data manipulation*.

[2] R Core Team. (2023). *R: A language and environment for statistical computing*.

[3] Dataset: GitHub Repository Metadata 2025.

[4] Source: Kaggle Dataset- Most Popular GitHub Repositories (Projects)
https://www.kaggle.com/datasets/donbarbos/github-repos

## 8. Video Presentation

URL: https://youtu.be/wZc1e3sOP_Y

## AI Assistance Disclosure

Portions of this project were supported by the use of ChatGPT (OpenAI) *as a supplemental tool*. AI assistance was used only for:

- Clarifying statistical concepts covered in INFO 6105
- Receiving guidance on R error messages and debugging steps
- Improving the clarity of explanations and report organization

All data analysis, interpretation of results, coding execution, and decision-making were completed independently by me. The final report, conclusions, and all submitted work reflect my own understanding and analytical reasoning.

**APPENDIX A — Complete R Code**

```
################################################################

# PHASE 1 — DATA LOADING, CLEANING & PREPROCESSING

# INFO 6105 FINAL PROJECT — GitHub Repository Analysis

################################################################

# 1. Load Required Libraries

library(dplyr)

# 2. Load Dataset

data <- read.csv("repositories.csv", stringsAsFactors = FALSE)

# Inspect structure

str(data)

# 3. Downsample to 10,000 Rows (Professor Recommended)

set.seed(123)

data_sample <- data[sample(nrow(data), 10000), ]

# 4. Create Log-Transformed Response Variable

data_sample$log_stars <- log(data_sample$Stars + 1)

# 5. Extract Year from Created.At

data_sample$created_year <- as.numeric(substr(data_sample$Created.At, 1, 4))
```

```
# 6. Create Age Group Factor (Old vs New)
data_sample$age_group <- ifelse(data_sample$created_year <= 2018, "Old", "New")
data_sample$age_group <- factor(data_sample$age_group)
# 7. Clean Language Column (Factor + Drop NAs)
data_sample$language <- as.factor(data_sample$Language)
data_sample <- data_sample[!is.na(data_sample$language), ]
data_sample$language <- droplevels(data_sample$language)
# 8. Keep Only Relevant Variables for Analysis
data_sample <- data_sample %>% select(
  log_stars, Stars, Forks, Issues, Watchers, Size,
  language, age_group, created_year
)
# 9. NA Check (Data Quality Check)
colSums(is.na(data_sample))
# 10. Convert Numeric Columns Explicitly
data_sample$Stars    <- as.numeric(data_sample$Stars)
data_sample$Forks    <- as.numeric(data_sample$Forks)
data_sample$Issues   <- as.numeric(data_sample$Issues)
```

```
data_sample$Watchers <- as.numeric(data_sample$Watchers)

data_sample$Size     <- as.numeric(data_sample$Size)

# 11. Remove Redundant Column (Watchers == Stars)

data_sample <- data_sample %>% select(-Watchers)

# 12. Trim Extreme Outliers (Top 0.1%) for Regression Stability

data_sample <- data_sample %>%

  filter(

    Stars < quantile(Stars, 0.999),

    Forks < quantile(Forks, 0.999),

    Issues < quantile(Issues, 0.999),

    Size  < quantile(Size,  0.999)

  )

# 13. Log-Transform Size to Reduce Skew

data_sample$log_size <- log(data_sample$Size + 1)

# Inspect cleaned numeric summaries

summary(data_sample[, c("Stars", "Forks", "Issues", "Size", "log_size")])



# 14. Filter Languages for ANOVA (Keep Only Groups >= 50 Repos)
```

```
lang_counts <- table(data_sample$language)

valid_langs <- names(lang_counts[lang_counts >= 50])

data_sample <- subset(data_sample, language %in% valid_langs)

data_sample$language <- droplevels(data_sample$language)

# Check balanced language groups

table(data_sample$language)

# 15. (Optional) Save Cleaned Dataset for Reproducibility

write.csv(data_sample, "cleaned_data.csv", row.names = FALSE)

#####################################################################

# PHASE 2 — EXPLORATORY DATA ANALYSIS (EDA)

#####################################################################

install.packages("GGally")

# 1. Load Required Libraries

library(dplyr)

library(ggplot2)

library(GGally)      # Scatterplot Matrix (SPLoM)

library(corrplot)    # Correlation heatmap visualization

# 2. Summary Statistics for Key Variables
```

```
summary_stats <- summary(data_sample[, c(
  "Stars", "Forks", "Issues", "Size", "log_size", "log_stars"
)])
summary_stats   # Print summary table
# 3. Scatterplot Matrix (SPLoM)
GGally::ggpairs(
  data_sample[, c("Stars", "Forks", "Issues", "log_size", "log_stars")],
  title = "Scatterplot Matrix of Key Quantitative Variables"
)
# 4. Histograms for Distribution Check
numeric_vars <- c("Stars", "Forks", "Issues", "Size", "log_stars", "log_size")

for (v in numeric_vars) {
  print(
    ggplot(data_sample, aes_string(v)) +
      geom_histogram(bins = 40, fill = "steelblue", color = "black") +
      theme_minimal() +
      ggtitle(paste("Histogram of", v))
```

)

}

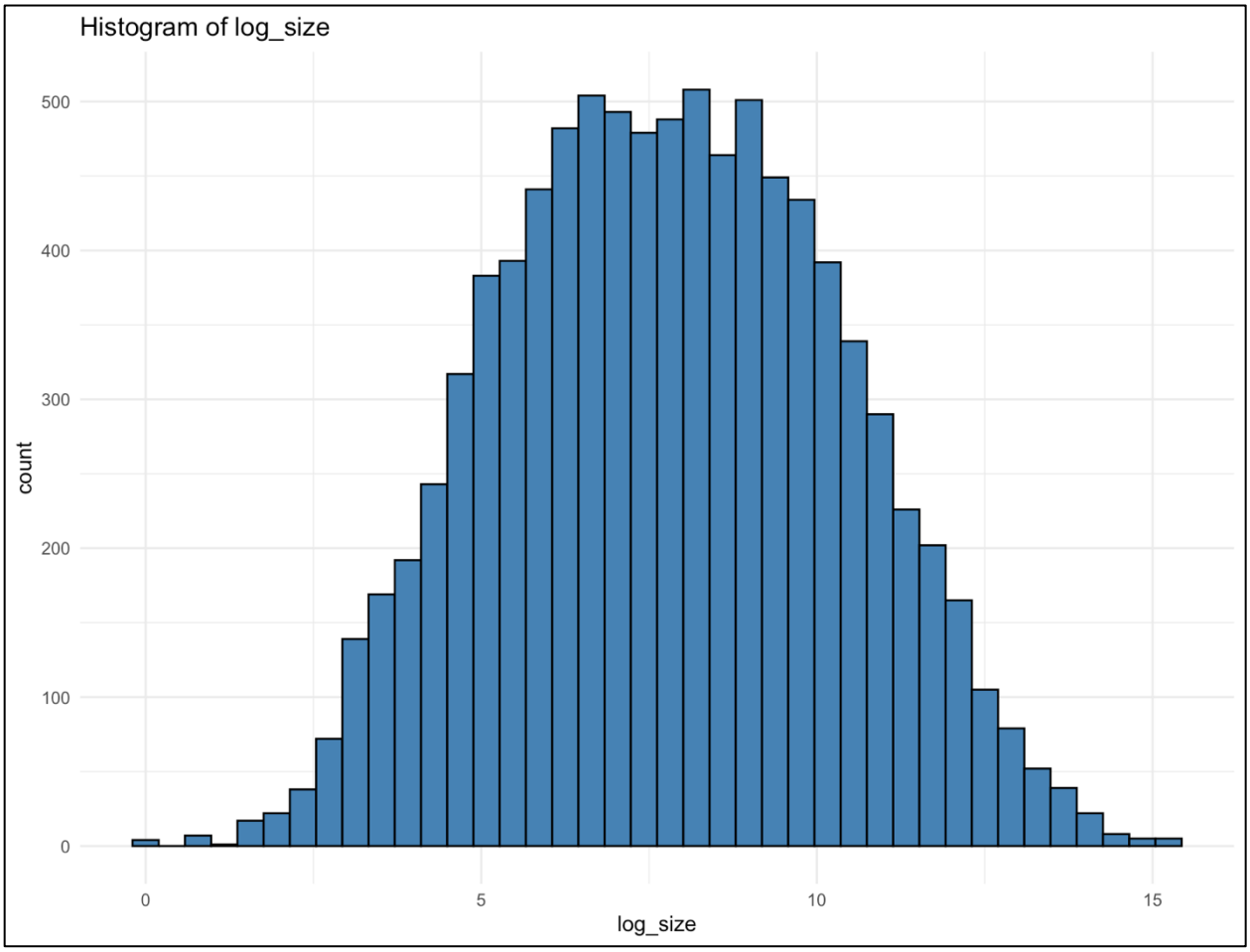



Figure 2.4 Histograms for Distribution Check

# 5. Boxplot of Popularity by Language (ANOVA justification)

ggplot(data_sample, aes(x = language, y = log_stars)) +

geom_boxplot(fill = "orange", alpha = 0.75) +

theme_minimal() +

labs(

```
    title = "Distribution of log_stars by Programming Language",
    x = "Programming Language",
    y = "log(stars + 1)"
  ) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
# 6. Correlation Heatmap
corr_data <- data_sample[, c("Stars", "Forks", "Issues", "log_size", "log_stars")]
correlation_matrix <- cor(corr_data)

corrplot(
  correlation_matrix,
  method = "color",
  type = "upper",
  addCoef.col = "black",
  title = "Correlation Heatmap of Key Variables",
  mar = c(0,0,2,0)
)
# 7. Boxplot of Popularity by Age Group (For Two-Way ANOVA)
```

```
ggplot(data_sample, aes(x = age_group, y = log_stars, fill = age_group)) +
  geom_boxplot(alpha = 0.8) +
  theme_minimal() +
  labs(
    title = "Popularity Comparison (log_stars): Old vs New Repositories",
    x = "Age Group",
    y = "log(stars + 1)"
  )
```

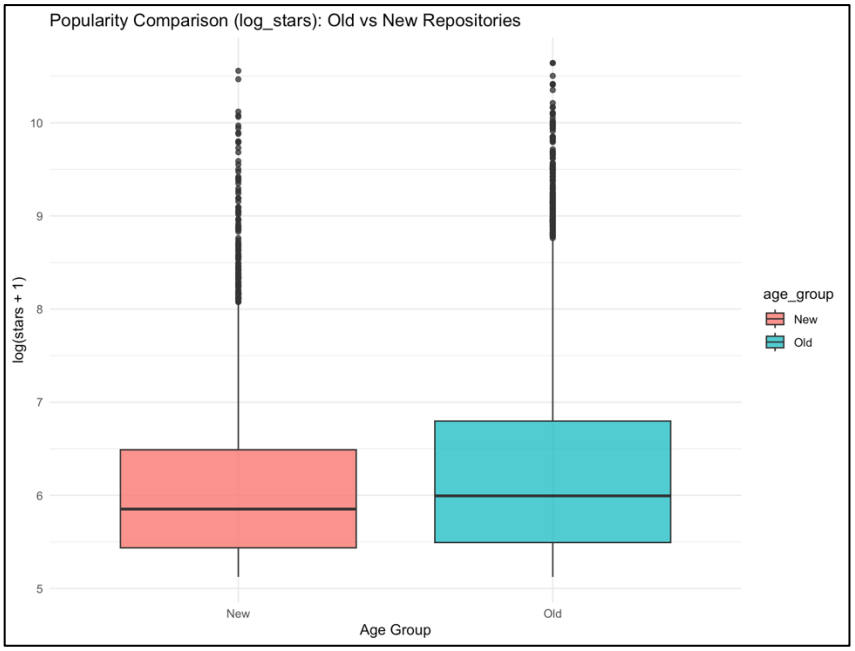



Figure 2.7 Boxplot of Popularity by Age Group (For Two-Way ANOVA)

# 8. Interaction Plot (Language × Age Group)

```
interaction.plot(
  data_sample$language,
  data_sample$age_group,
  data_sample$log_stars,
  xlab = "Programming Language",
  ylab = "Mean log_stars",
  trace.label = "Age Group",
  col = c("red", "blue"),
  lwd = 2
)

#####################################################################
# PHASE 3 — REGRESSION MODELING
#####################################################################
# 0. Load Required Libraries
library(dplyr)
library(ggplot2)
library(car)          # VIF Check
library(lmtest)       # BP Test
```

```
library(sandwich)     # Robust SE

# 1. Base Linear Regression Model

model1 <- lm(log_stars ~ Forks + Issues + log_size, data = data_sample)

summary(model1)

# 2. Add Language + Age Group (Categorical Predictors)

model2 <- lm(log_stars ~ Forks + Issues + log_size +

              language + age_group,

            data = data_sample)

summary(model2)

# 3. Interaction Model (Language × Age Group)

model3 <- lm(log_stars ~ Forks + Issues + log_size +

              language * age_group,

            data = data_sample)

summary(model3)

# 4. Model Comparison (Which Model Is Best?)

anova(model1, model2)

anova(model2, model3)

# 5. Regression Diagnostics (Assumption Checks)
```

## 5.1 Residual Plots (Linearity + Homoscedasticity)

```
par(mfrow = c(2,2))

plot(model2)

par(mfrow = c(1,1))
```

## 5.2 Normality of Residuals (Q-Q Plot)

```
qqnorm(residuals(model2))

qqline(residuals(model2))
```

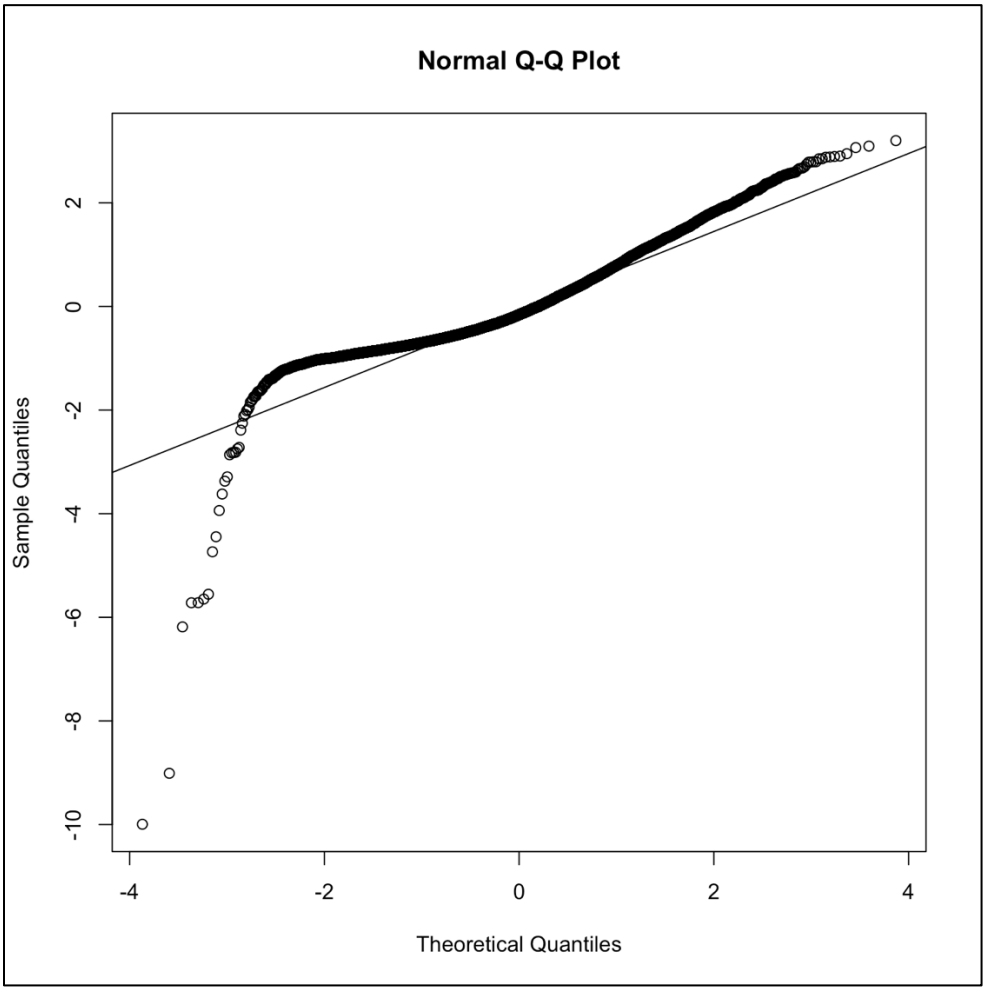


Figure 5.2 Normality of Residuals (Q-Q Plot)

## 5.3 Residual Histogram

```
hist(residuals(model2), breaks = 50,
    main = "Residual Histogram",
    col = "skyblue")
```

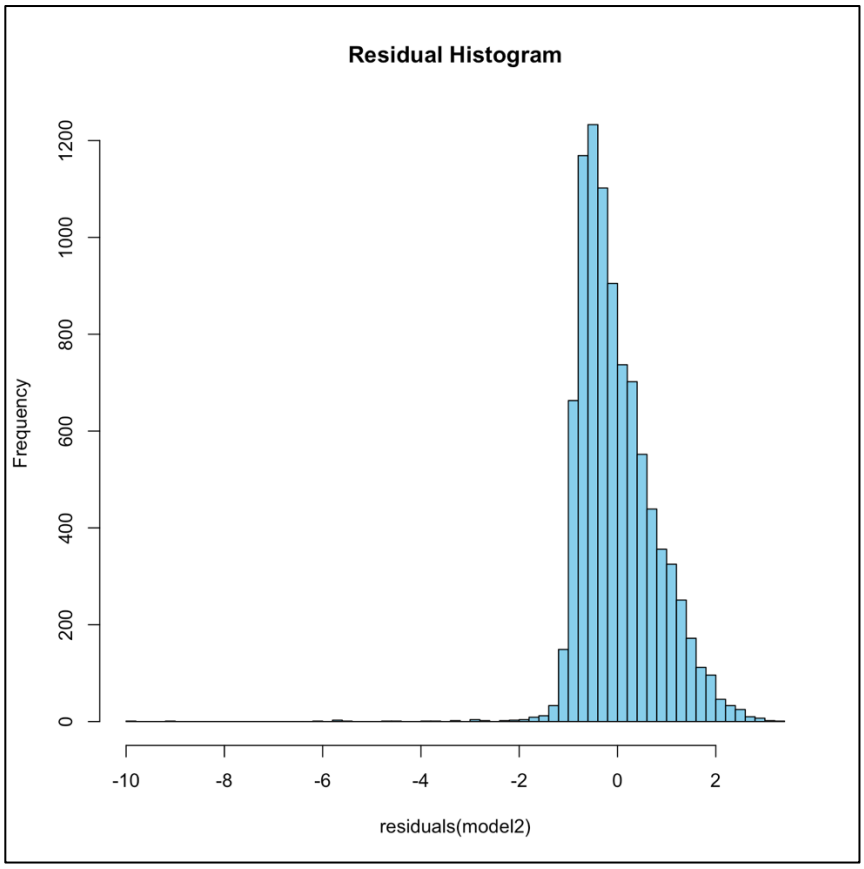



Figure 5.3 Residual Histogram

## 5.4 Homoscedasticity Test (Breusch-Pagan)

```
bptest(model2)
```

## 5.5 Multicollinearity Check (VIF)

vif(model2)

```
> ## 5.4 Homoscedasticity Test (Breusch-Pagan)
> bptest(model2)

        studentized Breusch-Pagan test

data:  model2
BP = 2949, df = 25, p-value < 2.2e-16

>
> ## 5.5 Multicollinearity Check (VIF)
> vif(model2)
              GVIF Df GVIF^(1/(2*Df))
Forks     1.148849  1        1.071844
Issues    1.155145  1        1.074777
log_size  1.126855  1        1.061534
language  1.211933 21        1.004587
age_group 1.116099  1        1.056456
>
```

# 6. Robust Standard Errors (If BP test indicates heteroskedasticity)

coeftest(model2, vcov = vcovHC(model2, type = "HC1"))

##########################################################################

# PHASE 4 —  ANOVA - ANALYSES

# 4.1 ONE WAY ANOVA

##########################################################################

library(dplyr)

library(ggplot2)

library(car)  # Only dependency you already have

# 1. Fit ANOVA Model

anova1 <- aov(log_stars ~ language, data = data_sample)

```
summary(anova1)

# 2. Summary stats by language

aggregate(log_stars ~ language, data_sample, summary)

# 3. Tukey HSD Post-hoc Test

tukey_lang <- TukeyHSD(anova1)

tukey_lang

# 4. Effect Size (MANUAL ETA-SQUARED)

anova_summary <- summary(anova1)[[1]]

SS_language <- anova_summary["language", "Sum Sq"]

SS_residual <- anova_summary["Residuals", "Sum Sq"]

eta_sq <- SS_language / (SS_language + SS_residual)

eta_sq

# 5. Assumption checks

qqnorm(residuals(anova1)); qqline(residuals(anova1))

leveneTest(log_stars ~ language, data = data_sample)

# 6. Boxplot

ggplot(data_sample, aes(language, log_stars)) +

  geom_boxplot(fill="orange") +
```

```
  theme_minimal() +
  theme(axis.text.x = element_text(angle=45, hjust=1)) +
  labs(
    title="log_stars by Programming Language",
    x="Language", y="log(stars + 1)"
  )
#################################################################
# 4.2 TWO-WAY ANOVA — Language × Age Group Interaction
#################################################################
# Model
anova2 <- aov(log_stars ~ language * age_group, data = data_sample)
summary(anova2)
# Post-hoc Tests
## 1. Pairwise comparisons for Language (main effect)
pairwise.t.test(data_sample$log_stars, data_sample$language,
                p.adjust.method = "BH")
## 2. Pairwise comparisons for Interaction (Language × Age Group)
pairwise.t.test(data_sample$log_stars,
```

```
            interaction(data_sample$language, data_sample$age_group),
            p.adjust.method = "BH")
# Interaction Plot
interaction.plot(
  data_sample$language,
  data_sample$age_group,
  data_sample$log_stars,
  xlab = "Programming Language",
  ylab = "Mean log_stars",
  trace.label = "Age Group",
  col = c("red", "blue"),
  lwd = 2
)
# Assumption Checks
## Normality
qqnorm(residuals(anova2)); qqline(residuals(anova2))


## Equal Variances
```

```
leveneTest(log_stars ~ language * age_group, data = data_sample)


################################################################

# 4.3 INTERPRETATION PRINTING (SUMMARY)



cat("\n========== ONE-WAY ANOVA SUMMARY ==========\n")

print(summary(anova1))

cat("\nTukey Post-hoc:\n")

print(tukey_lang)

cat("\nEta-Squared:\n")

print(eta_sq)


cat("\n========== TWO-WAY ANOVA SUMMARY ==========\n")

print(summary(anova2))


                    ################ END ##################
```