Q1:

We Utilize “cor” function in R to find the correlation where we indicated method “spearman”.

For PDE dataset,

Correlation Coefficients:

The correlation coefficient between "bug" and itself is always 1, as it represents the perfect correlation of a variable with itself.
The correlation between "bug" and "loc" is approximately 0.2664.
The correlation between "bug" and "bf" is approximately 0.2896.
The correlation between "bug" and "hcm" is approximately 0.2617.

Interpretation:

The positive correlation coefficients between "bug" and the other variables (loc, bf, hcm) indicate a positive relationship. This means that as the values of "bug" increase, the values of these variables tend to increase as well.
The correlation coefficients are all below 1, suggesting that these variables are not perfectly correlated with "bug." Instead, they exhibit moderate positive correlations.

Strength of Correlations:

A correlation coefficient of 0.2664 to 0.2896 suggests a moderate positive relationship, indicating that as one variable increases, the other tends to increase, but the relationship is not extremely strong.

Variable with the Strongest Correlation to Bug:

The correlation coefficient of approximately 0.2896 indicates that "bf" has the highest positive association with "bug" in the PDE dataset.

For JDT dataset

Correlation Coefficients:

The correlation coefficient between "bug" and itself is always 1, indicating a perfect correlation, as it represents the variable's relationship with itself.
The correlation between "bug" and "bf" is approximately 0.4493.
The correlation between "bug" and "loc" is approximately 0.4090.
The correlation between "bug" and "hcm" is approximately 0.4532.

Interpretation:

The positive correlation coefficients between "bug" and the other variables (bf, loc, hcm) indicate a positive relationship. This means that as the values of "bug" increase, the values of these variables tend to increase as well.
These correlation coefficients are notably higher compared to those in the PDE dataset, indicating stronger positive correlations.

Strength of Correlations:

In terms of the strength of the correlations, all three variables ("bf," "loc," and "hcm") exhibit relatively strong positive correlations with "bug."
The correlation coefficients of approximately 0.4090 to 0.4532 suggest that these variables have strong positive associations with "bug."

Variable with the Strongest Correlation to Bug:

The correlation coefficient of approximately 0.4532 indicates that "hcm" has the highest positive association with "bug" in the JDT dataset.

```
> library(readr)
> pde_data <- read.csv("C:/Users/faiya/OneDrive - Texas Tech University/Texas tech course/fall 23/software analytics/HW6/PDE.csv")
> jdt_data <- read.csv("C:/Users/faiya/OneDrive - Texas Tech University/Texas tech course/fall 23/software analytics/HW6/JDT.csv")
> #pde
> pde_correlation <- cor(pde_data[, "bug"], pde_data[, -1], method = "spearman")
> # Find the variable with the strongest absolute correlation to bug
> abs_correlation <- abs(pde_correlation)
> pde_max_corr_var <- names(pde_correlation)[which(abs_correlation == max(abs_correlation[-1]))]
> pde_max_corr <- max(abs_correlation[-1])  # Excluding self-correlation with "bug"
> cat("PDE Dataset - Spearman Correlation Matrix:\n")
PDE Dataset - Spearman Correlation Matrix:
> print(pde_correlation)
     bug       loc        bf       hcm
[1,]   1 0.2663918 0.2895665 0.2617094
> cat("\nThe variable with the strongest correlation to 'bug' in the PDE dataset is:", pde_max_corr_var, "with a correlation of", p
de_max_corr)

The variable with the strongest correlation to 'bug' in the PDE dataset is: with a correlation of 0.2895665
>
```

```
The variable with the strongest correlation to 'bug' in the PDE dataset is: with a correlation of 0.2895665
> jdt_correlation <- cor(jdt_data[, "bug"], jdt_data[, -1], method = "spearman")
> # Find the variable with the strongest absolute correlation to bug in the JDT dataset
> abs_jdt_correlation <- abs(jdt_correlation)
> jdt_max_corr_var <- names(jdt_correlation)[which(abs_jdt_correlation == max(abs_jdt_correlation[-1]))]
> jdt_max_corr <- max(abs_jdt_correlation[-1])  # Excluding self-correlation with "bug"
> # Print the results for the JDT dataset
> cat("\nJDT Dataset - Spearman Correlation Matrix:\n")

JDT Dataset - Spearman Correlation Matrix:
> print(jdt_correlation)
     bug        bf       loc       hcm
[1,]   1 0.4492973 0.4089799 0.4531625
> cat("\nThe variable with the strongest correlation to 'bug' in the JDT dataset is:", jdt_max_corr_var, "with a correlation of", j
dt_max_corr)

The variable with the strongest correlation to 'bug' in the JDT dataset is: with a correlation of 0.4531625
>
```

Q2:

The provided code serves two primary purposes: calculating the percentage of defect-free classes and determining the percentage of bugs found in the top 10% and top 20% most defect-prone classes for both the JDT and PDE datasets.

For the percentage of defect-free classes, the code begins by identifying defect-free classes where the "bug" variable equals zero, signifying the absence of defects. Subsequently, it calculates the total number of classes in each dataset and derives the percentage of defect-free classes by dividing the count of such classes by the total class count, then multiplying by 100. This provides valuable insights into the data quality and reliability of both datasets.

Concerning the percentage of bugs found in the most defect-prone classes, the code follows a multi-step process. It first sorts the datasets in descending order based on the number of bugs, thus pinpointing the most defect-prone classes. After calculating the total number of bugs in each dataset, the code determines the number of classes constituting the top 10% and top 20% most defect-prone categories, scaling the thresholds based on dataset size. It then selects the classes within these thresholds and calculates the total number of bugs found in these high-defect classes. Finally, it computes the percentage of bugs present in the top classes by dividing the sum of bugs in these classes by the overall bug count and multiplying by 100.

Yes, based on the analysis of both the JDT and PDE datasets, it can be stated that "most post-release bugs occur in a small portion of code." This observation is supported by the significantly higher

percentages of bugs found in the top 10% and top 20% most defect-prone classes compared to the overall class count.

For the JDT dataset:

- Approximately 71.66% of the bugs are found in the top 10% most defect-prone classes.
- Approximately 98.13% of the bugs are found in the top 20% most defect-prone classes.

For the PDE dataset:

- Approximately 82.70% of the bugs are found in the top 10% most defect-prone classes.
- All bugs (100%) are found in the top 20% most defect-prone classes.

These percentages indicate that a relatively small portion of code, represented by the top classes in terms of defects, is responsible for most post-release bugs. This skew in the distribution of post-release bugs suggests that these specific classes might be critical areas where improvements or bug-fixing efforts should be prioritized.

```
> #Q2
> # Count the number of defect-free classes
> defect_free_classes <- sum(jdt_data$bug == 0)
> # Calculate the total number of classes
> total_classes <- nrow(jdt_data)
> # Calculate the percentage of defect-free classes
> percentage_defect_free <- (defect_free_classes / total_classes) * 100
> # Print the result
> cat("Percentage of defect-free classes in the JDT dataset:", percentage_defect_free, "%\n")
Percentage of defect-free classes in the JDT dataset: 79.33801 %
> # Sort the dataset by the number of bugs in descending order
> sorted_jdt_data <- jdt_data[order(-jdt_data$bug), ]
> # Calculate the number of classes in the top 10% and top 20%
> top_10_percent <- round(0.10 * total_classes)
> top_20_percent <- round(0.20 * total_classes)
> # Select the top 10% and top 20% of classes
> top_10_classes <- sorted_jdt_data[1:top_10_percent, ]
> top_20_classes <- sorted_jdt_data[1:top_20_percent, ]
> # Calculate the total number of bugs in all classes
> total_bugs <- sum(jdt_data$bug)
> # Calculate the total number of bugs in the top 10% and top 20% most defect-prone classes
> bugs_in_top_10_percent <- sum(top_10_classes$bug)
> bugs_in_top_20_percent <- sum(top_20_classes$bug)
> # Calculate the percentages
> percentage_bugs_in_top_10_percent <- (bugs_in_top_10_percent / total_bugs) * 100
> percentage_bugs_in_top_20_percent <- (bugs_in_top_20_percent / total_bugs) * 100
> # Print the results
> cat("Percentage of bugs found in the top 10% most defect-prone classes:", percentage_bugs_in_top_10_percent, "%\n")
Percentage of bugs found in the top 10% most defect-prone classes: 71.65775 %
> cat("Percentage of bugs found in the top 20% most defect-prone classes:", percentage_bugs_in_top_20_percent, "%\n")
Percentage of bugs found in the top 20% most defect-prone classes: 98.12834 %
>
```

```
> #pde
> # Percentage of Defect-Free Classes in the PDE Dataset
> defect_free_classes_pde <- sum(pde_data$bug == 0)
> total_classes_pde <- nrow(pde_data)
> percentage_defect_free_pde <- (defect_free_classes_pde / total_classes_pde) * 100
> cat("Percentage of defect-free classes in the PDE dataset:", percentage_defect_free_pde, "%\n")
Percentage of defect-free classes in the PDE dataset: 86.03874 %
> # Percentage of Bugs in Top 10% and Top 20% Most Defect-Prone Classes in the PDE Dataset
> sorted_pde_data <- pde_data[order(-pde_data$bug), ]
> total_bugs_pde <- sum(pde_data$bug)
> top_10_percent_pde <- round(0.10 * total_classes_pde)
> top_20_percent_pde <- round(0.20 * total_classes_pde)
> top_10_classes_pde <- sorted_pde_data[1:top_10_percent_pde, ]
> top_20_classes_pde <- sorted_pde_data[1:top_20_percent_pde, ]
> bugs_in_top_10_percent_pde <- sum(top_10_classes_pde$bug)
> bugs_in_top_20_percent_pde <- sum(top_20_classes_pde$bug)
> percentage_bugs_in_top_10_percent_pde <- (bugs_in_top_10_percent_pde / total_bugs_pde) * 100
> percentage_bugs_in_top_20_percent_pde <- (bugs_in_top_20_percent_pde / total_bugs_pde) * 100
> cat("Percentage of bugs found in the top 10% most defect-prone classes in the PDE dataset:", percentage_bugs_in_top_10_percent_pde, "%\n")
Percentage of bugs found in the top 10% most defect-prone classes in the PDE dataset: 82.69795 %
> cat("Percentage of bugs found in the top 20% most defect-prone classes in the PDE dataset:", percentage_bugs_in_top_20_percent_pde, "%\n")
Percentage of bugs found in the top 20% most defect-prone classes in the PDE dataset: 100 %
>
```

Q3:

We create separate boxplots for the JDT and PDE datasets to visualize the distribution of bugs in each project. We use boxplot function, this will display the central tendency, spread, and potential outliers in the data.

No, they have different distributions.

Based on the Welch Two Sample t-test results, it can be concluded that there exists a statistically significant difference in the bug distributions between the JDT (JDE) and PDE datasets. The obtained p-value of 0.0003576 is substantially smaller than the commonly used significance level of 0.05, providing robust statistical evidence to reject the null hypothesis. This rejection implies that the bug distributions in the two projects are not equal.

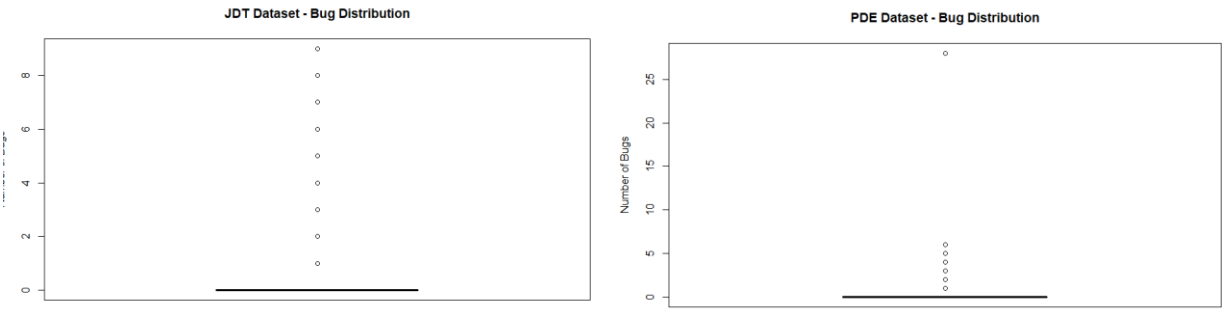
Furthermore, a comparison of the means reveals that the JDT dataset exhibits a higher mean number of bugs (approximately 0.3751) compared to the PDE dataset, where the mean number of bugs is approximately 0.2278. This disparity in means signifies that the JDT project tends to have classes with a higher average number of post-release bugs than the PDE project.

The 95% confidence interval for the true difference in means, ranging from 0.0665 to 0.2281, reinforces the conclusion of a substantial distinction in bug distributions.

So, it can be inferred that the JDT project is more defect-prone, as its classes often exhibit a higher number of post-release bugs compared to the PDE project.

```
#q3
# Create a boxplot for the JDT dataset
boxplot(jdt_data$bug, col = "lightblue", main = "JDT Dataset - Bug Distribution", ylab = "Number of Bugs")

# Create a boxplot for the PDE dataset
boxplot(pde_data$bug, col = "lightgreen", main = "PDE Dataset - Bug Distribution", ylab = "Number of Bugs")
```




```
> # Perform a t-test to compare the bug distributions between JDT and PDE
> t_test_result <- t.test(jdt_data$bug, pde_data$bug)
> # Print the t-test result
> cat("T-test Result:\n")
T-test Result:
> print(t_test_result)

        Welch Two Sample t-test

data:  jdt_data$bug and pde_data$bug
t = 3.5757, df = 2007.2, p-value = 0.0003576
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.06652755 0.22814538
sample estimates:
mean of x mean of y
0.3751254 0.2277889
```

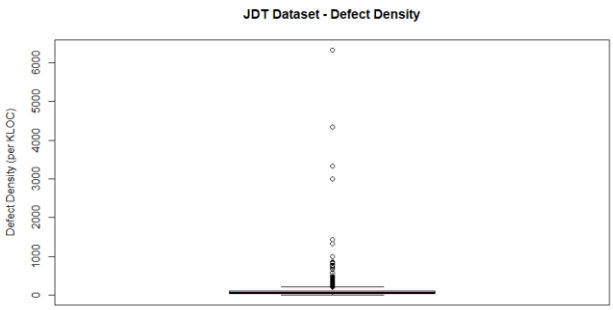
Q4:

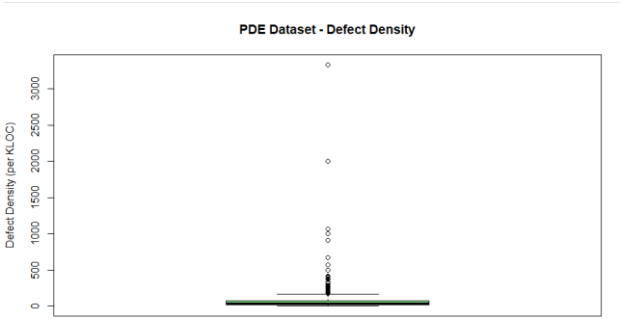
Based on the results of the Welch Two Sample t-test, a comprehensive analysis of the defect density distributions in the JDT (JDE) and PDE datasets reveals significant distinctions. The obtained p-value of 1.561e-05 is markedly smaller than the commonly accepted significance level of 0.05, providing compelling statistical evidence to reject the null hypothesis. This rejection signifies a pronounced and statistically significant difference in the defect density distributions between the two datasets.

Furthermore, a comparison of the means of defect density is illuminating. The JDT dataset exhibits a notably higher mean defect density, measuring approximately 111.39293 defects per thousand lines of code (KLOC), whereas the PDE dataset has a mean defect density of approximately 65.67926 defects per KLOC. This discrepancy in means is substantial and implies that classes within the JDT dataset are prone to a significantly higher average number of defects per KLOC when contrasted with their counterparts in the PDE dataset.

The 95% confidence interval for the true difference in means, ranging from 25.03610 to 66.39125 defects per KLOC, further underscores the statistically significant difference in defect density.

So, JDT dataset exhibits a significantly higher defect density than the PDE dataset. Consequently, it can be inferred that the JDT dataset contains classes that are more defect-prone, with a higher average number of defects per KLOC.






```
> #q4
> jdt_data$defect_density <- (jdt_data$bug + jdt_data$bf) / (jdt_data$loc / 1000)
> View(jdt_data)
> pde_data$defect_density <- (pde_data$bug + pde_data$bf) / (pde_data$loc / 1000)
```

```
> # Identify rows with infinite defect density in the JDT dataset
> jdt_outliers <- which(!is.finite(jdt_data$defect_density))
> # Identify rows with infinite defect density in the PDE dataset
> pde_outliers <- which(!is.finite(pde_data$defect_density))
> # Remove rows with infinite defect density in the JDT dataset
> jdt_data <- jdt_data[-jdt_outliers, ]
> # Remove rows with infinite defect density in the PDE dataset
> pde_data <- pde_data[-pde_outliers, ]
> # Create a boxplot for the JDT dataset's defect density
> boxplot(jdt_data$defect_density, col = "lightblue", main = "JDT Dataset - Defect Density", ylab = "Defect Density (per KLOC)")
> # Create a boxplot for the PDE dataset's defect density
> boxplot(pde_data$defect_density, col = "lightgreen", main = "PDE Dataset - Defect Density", ylab = "Defect Density (per KLOC)")
> # Perform a t-test to compare the defect density distributions between JDT and PDE
> t_test_resultpo <- t.test(jdt_data$defect_density, pde_data$defect_density)
> # Print the t-test result
> cat("T-test Result:\n")
T-test Result:
> print(t_test_resultpo)

        Welch Two Sample t-test

data:  jdt_data$defect_density and pde_data$defect_density
t = 4.3374, df = 1220.1, p-value = 1.561e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 25.03610 66.39125
sample estimates:
mean of x mean of y
111.39293  65.67926
```

Q5:

In defect prediction, the priority is to identify the most defect-prone code rather than achieving the smallest average prediction errors. If a defect prediction model has a low average error, but it is not very good at identifying the most defect-prone code, then it will not be very useful for improving software quality. On the other hand, if a defect prediction model is very good at identifying the most defect-prone code, even if it has a higher average error, it will be more useful for improving software quality. This focus is driven by the need to allocate limited resources effectively, mitigate risks, reduce costs, and enhance the customer experience.

When using a linear regression model for defect prediction, it's crucial to measure predictive power accurately. Common measurements include R-squared (R^2) for explaining variance, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) for prediction accuracy, and classification metrics like precision, recall, and F1-score to assess the model's ability to identify defect-prone code. The choice of measurement depends on project goals, with a balance between prediction accuracy and defect identification being key.