Fraud detection

Matteo Delle Cave and Tommaso Biganzoli

2025-03-05

FRAUD DETECTION: Anomaly Analysis in Transactional Data

Introduction

This project focuses on analyzing a financial transaction dataset with the goal of identifying anomalies and potential fraud. The dataset, sourced from Kaggle (https://www.kaggle.com/datasets/valakhorasani/bank-transaction-dataset-for-fraud-detection), contains **2,512 transactions** with detailed attributes related to bank accounts, transaction characteristics, and user behavior. The analysis aims to detect unusual patterns and suspicious transactions through **Feature Engineering, Exploratory Data Analysis (EDA), Clustering, and Anomaly Detection** techniques.

Dataset Description

The dataset includes key information on financial transactions, such as:

- **TransactionID**: Unique identifier for each transaction.
- **AccountID**: Unique identifier for the account associated with the transaction.
- **TransactionAmount**: Monetary value of the transaction.
- **TransactionDate**: Timestamp of the transaction.
- **TransactionType**: Type of transaction (*Credit* or *Debit*).
- **Location**: City where the transaction was performed.
- **DeviceID**: Identifier of the device used.
- **IP Address**: IP address associated with the transaction.
- **MerchantID**: Unique identifier for the merchant.
- **AccountBalance**: Account balance after the transaction.
- **PreviousTransactionDate**: Date of the last transaction for the account.
- **Channel**: Channel through which the transaction was conducted (*Online, ATM, Branch*).
- **CustomerAge** and **CustomerOccupation**: Demographic information of the account holder.
- **TransactionDuration**: Duration of the transaction in seconds.
- **LoginAttempts**: Number of login attempts before the transaction.

Analysis Objective

The main goal is to **identify anomalous transactions** that may indicate fraudulent activity. To achieve this, the project employs:

- **Feature Engineering** to extract new informative variables as DaySinceLastTransaction, LoginAttempts,Transaction Ratio, IPChange and DeviceChange.
- **Exploratory Data Analysis (EDA)** to detect patterns and unusual distributions.
- **Clustering (K-Means, Hierarchical, DBSCAN)** to segment transactions into groups and detect outliers.
- **Anomaly Detection (LOF)** to isolate suspicious observations based on their deviation from normal transaction behavior.

- **Principal Component Analysis (PCA)** to try to reduce dimensionality and identify the most influential variables.

The combination of **unsupervised Machine Learning methods** and **statistical analysis** enhances fraud detection and provides valuable insights for financial security.

LOADING DATA AND LIBRARIES

```
bank = read.csv('bank_transactions_data_2.csv')
library(dplyr)
library(ggplot2)
library(gridExtra)
library(ggrepel)
library(dbscan)
library(Rlof)
library(psych)
library(FactoMineR)
library(factoextra)
```

1. FEATURE ENGENEERING

To effectively analyze transactional behavior and detect anomalies, the dataset is first **sorted by AccountID and TransactionDate** to ensure a chronological order for each account's transactions. This allows for calculating derived features based on historical transaction patterns.

```
# Data conversion in the format "%Y-%m-%d"
bank$TransactionDate <- as.Date(bank$TransactionDate, format = "%Y-%m-%d")
bank$PreviousTransactionDate <- as.Date(bank$PreviousTransactionDate, format = "%Y-%m-%d")

# Sorting by AccountID and TransactionDate
bank <- bank %>%
  arrange(AccountID, TransactionDate) %>%
  group_by(AccountID) %>%
  mutate(PreviousTransactionDate = lag(TransactionDate)) %>%
  ungroup()
```

To effectively analyze transactional behavior and detect anomalies, the dataset is first **sorted by AccountID and TransactionDate** to ensure a chronological order for each account's transactions. This allows for calculating derived features based on historical transaction patterns.

```
bank$DaysSinceLastTransaction <- as.numeric(difftime(bank$TransactionDate, bank$PreviousTransactionDate
bank <- bank %>% filter(!is.na(DaysSinceLastTransaction))

bank$TransactionRatio <- bank$TransactionAmount / (bank$AccountBalance + 1)
```

- **DaysSinceLastTransaction**: Computes the time difference (in days) between consecutive transactions for the same account.
- **TransactionRatio**: Measures the proportion of the transaction amount relative to the account balance.

```
# Device and IP change identification (first value per account excluded)
bank <- bank %>%
  arrange(AccountID, TransactionDate) %>%
  group_by(AccountID) %>%
  mutate(
    DeviceChange = ifelse(DeviceID != lag(DeviceID), 1, 0),
    IPChange = ifelse(IP.Address != lag(IP.Address), 1, 0)
  ) %>%
  ungroup()
# Anomalous logins: more than 3 attempts
bank$LoginAnomaly <- ifelse(bank$LoginAttempts > 3, 1, 0)
# Check if there are anomalous values
negative_values <- sum(bank$DaysSinceLastTransaction < 0, na.rm = TRUE)
missing_values <- sum(is.na(bank$DaysSinceLastTransaction))
# Device and IP different from last transaction
bank$DeviceChange <- ifelse(bank$DeviceID != lag(bank$DeviceID), 1, 0)
bank$IPChange <- ifelse(bank$IP.Address != lag(bank$IP.Address), 1, 0)
```

- **DeviceChange & IPChange**: Detects if a transaction was made using a different device or IP address compared to the previous transaction.
- **LoginAnomaly**: Flags transactions where the number of login attempts exceeds 3, indicating potential fraudulent behavior.

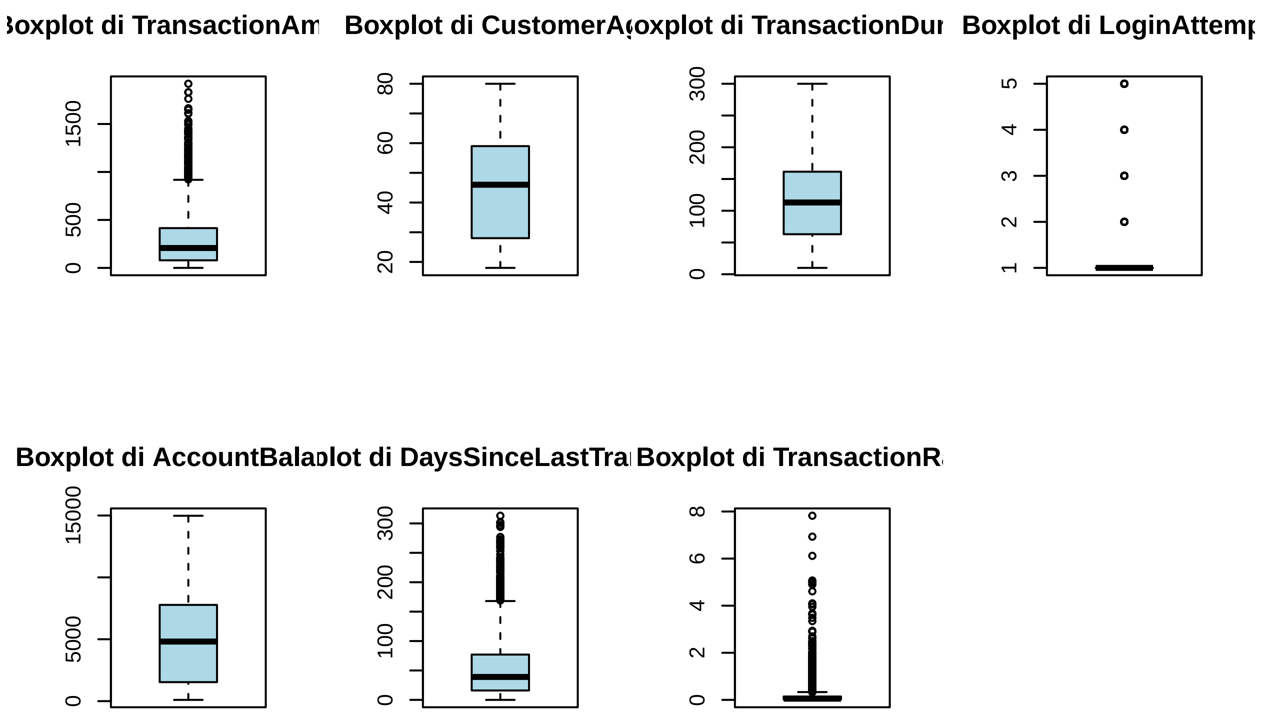
Additional checks ensure data consistency by filtering out missing values and identifying inconsistencies in transaction timing. These steps provide a foundation for further anomaly detection and fraud analysis.

2. DATA EXPLORATORY ANALYSIS

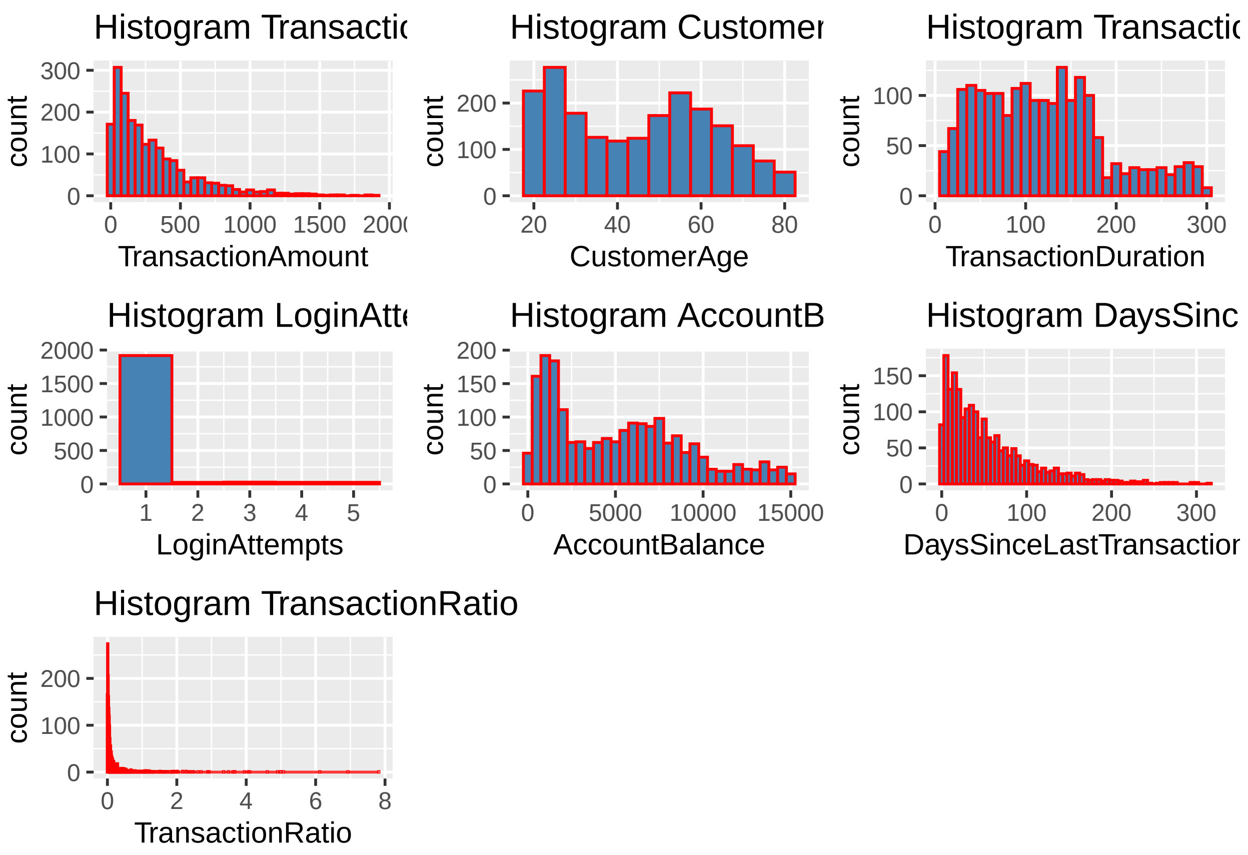
To begin the Exploratory Data Analysis (EDA), we visualized **quantitative variables** using **histograms and box plots** to examine their distributions, detect possible skewness, and identify potential outliers. Understanding the distribution of numerical variables helps assess whether transformations are needed to normalize skewed data.

For **categorical variables**, we plotted **frequency distributions** to check for imbalanced factors, which could affect the performance of machine learning models. Identifying highly unbalanced categories may be crucial for fraud detection.

```
# Categorical variables
categorical_vars <- c("TransactionID", "AccountID", "TransactionType", "Location",
                      "DeviceID", "IP.Address", "MerchantID", "Channel", "CustomerOccupation",
                      "DeviceChange", "IPChange", "LoginAnomaly")

# Numerical variables
numeric_vars <- c("TransactionAmount", "CustomerAge", "TransactionDuration",
                  "LoginAttempts", "AccountBalance", "DaysSinceLastTransaction", "TransactionRatio")

bank[categorical_vars] <- lapply(bank[categorical_vars], as.factor)


par(mfrow = c(2,4))
for (var in numeric_vars) {
  boxplot(bank[[var]], main = paste("Boxplot di", var), col = "lightblue")
}
par(mfrow = c(1,1))
```

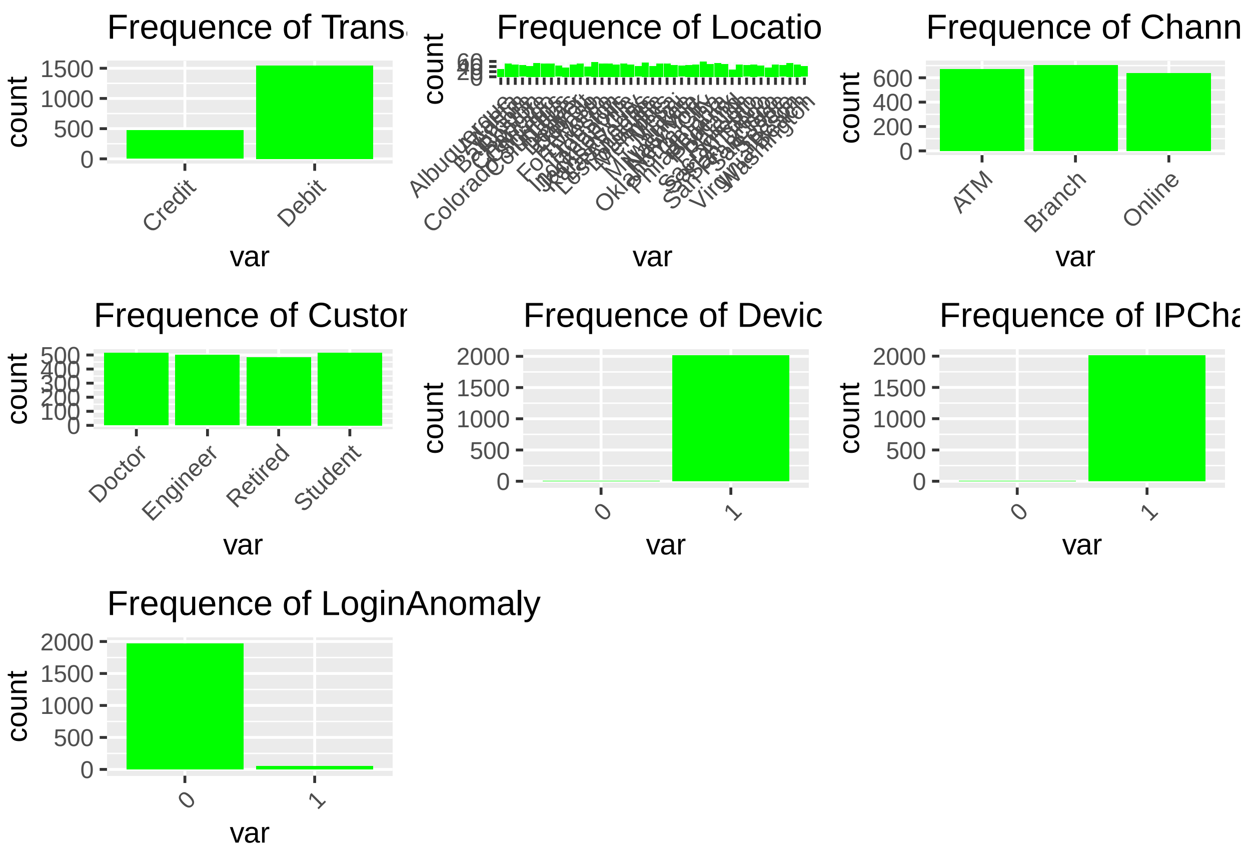


```r
# histograms
p1 <- ggplot(bank, aes(x = TransactionAmount)) +
  geom_histogram(binwidth = 50, fill = "steelblue", color = "red") +
  ggtitle("Histogram TransactionAmount")

p2 <- ggplot(bank, aes(x = CustomerAge)) +
  geom_histogram(binwidth = 5, fill = "steelblue", color = "red") +
  ggtitle("Histogram CustomerAge")

p3 <- ggplot(bank, aes(x = TransactionDuration)) +
  geom_histogram(binwidth = 10, fill = "steelblue", color = "red") +
  ggtitle("Histogram TransactionDuration")

p4 <- ggplot(bank, aes(x = LoginAttempts)) +
  geom_histogram(binwidth = 1, fill = "steelblue", color = "red") +
  ggtitle("Histogram LoginAttempts")

p5 <- ggplot(bank, aes(x = AccountBalance)) +
  geom_histogram(binwidth = 500, fill = "steelblue", color = "red") +
  ggtitle("Histogram AccountBalance")

p6 <- ggplot(bank, aes(x = DaysSinceLastTransaction)) +
  geom_histogram(binwidth = 5, fill = "steelblue", color = 'red') +
  ggtitle("Histogram DaysSinceLastTransaction")

p7 <- ggplot(bank, aes(x = TransactionRatio)) +
  geom_histogram(binwidth = 0.01, fill = "steelblue", color = "red") +
  ggtitle("Histogram TransactionRatio")

# Mostra tutti i grafici
grid.arrange(p1, p2, p3, p4, p5, p6, p7, ncol = 3)
```



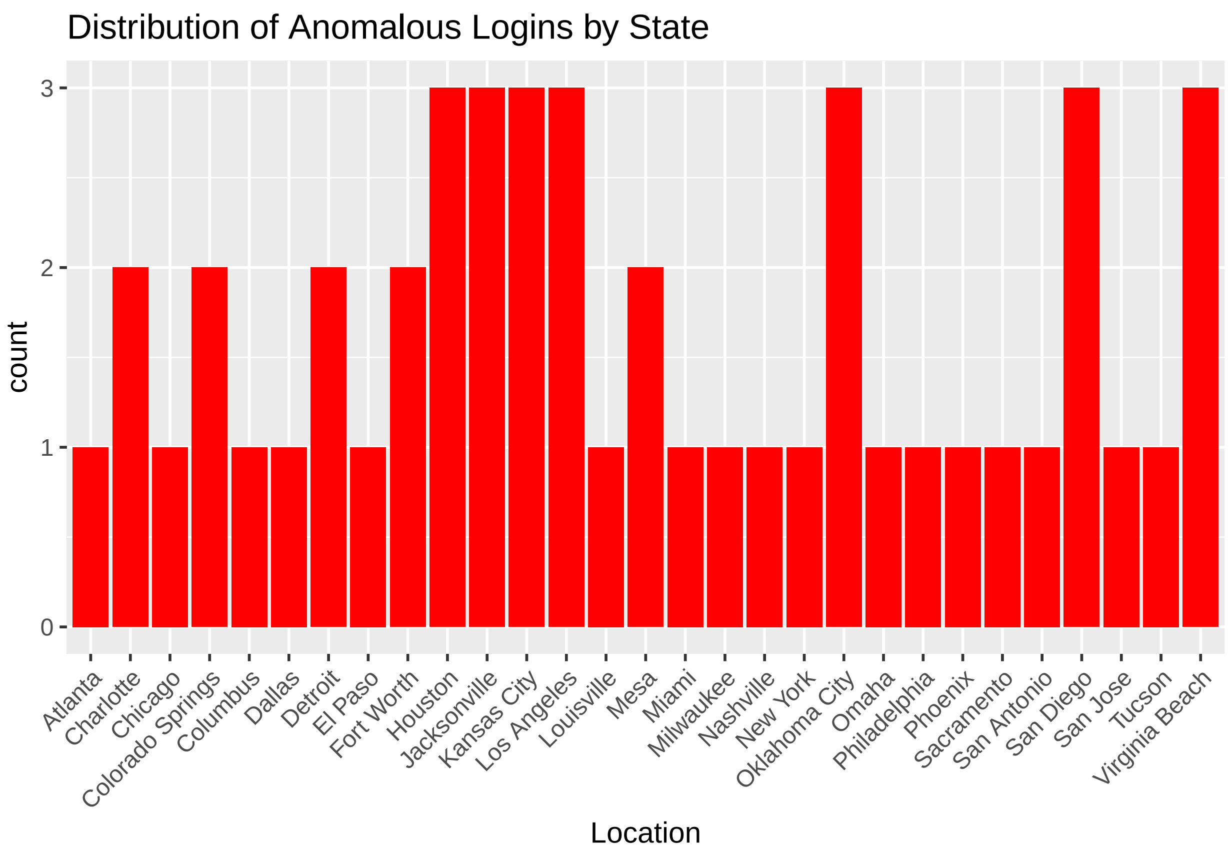
The distributions of the **quantitative variables** appear to be fairly symmetric, except for **TransactionAmount, DaysSinceLastTransaction, and TransactionRatio**, which exhibit strong **right skewness**. Given this skewed distribution, a **logarithmic transformation** could be considered to improve normality and stabilize variance for more effective analysis.

```r
# Frequencies
plot_frequency <- function(var, var_name) {
  ggplot(bank, aes(x = var)) +
    geom_bar(fill = "green") +
    ggtitle(paste("Frequence of", var_name)) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
}

# plots
p8 <- plot_frequency(bank$TransactionType, "TransactionType")
p9 <- plot_frequency(bank$Location, "Location")
p10 <- plot_frequency(bank$Channel, "Channel")
p11 <- plot_frequency(bank$CustomerOccupation, "CustomerOccupation")
p12 <- plot_frequency(bank$DeviceChange, "DeviceChange")
p13 <- plot_frequency(bank$IPChange, "IPChange")
p14 <- plot_frequency(bank$LoginAnomaly, "LoginAnomaly")

grid.arrange(p8, p9, p10, p11, p12, p13, p14, ncol = 3)
```

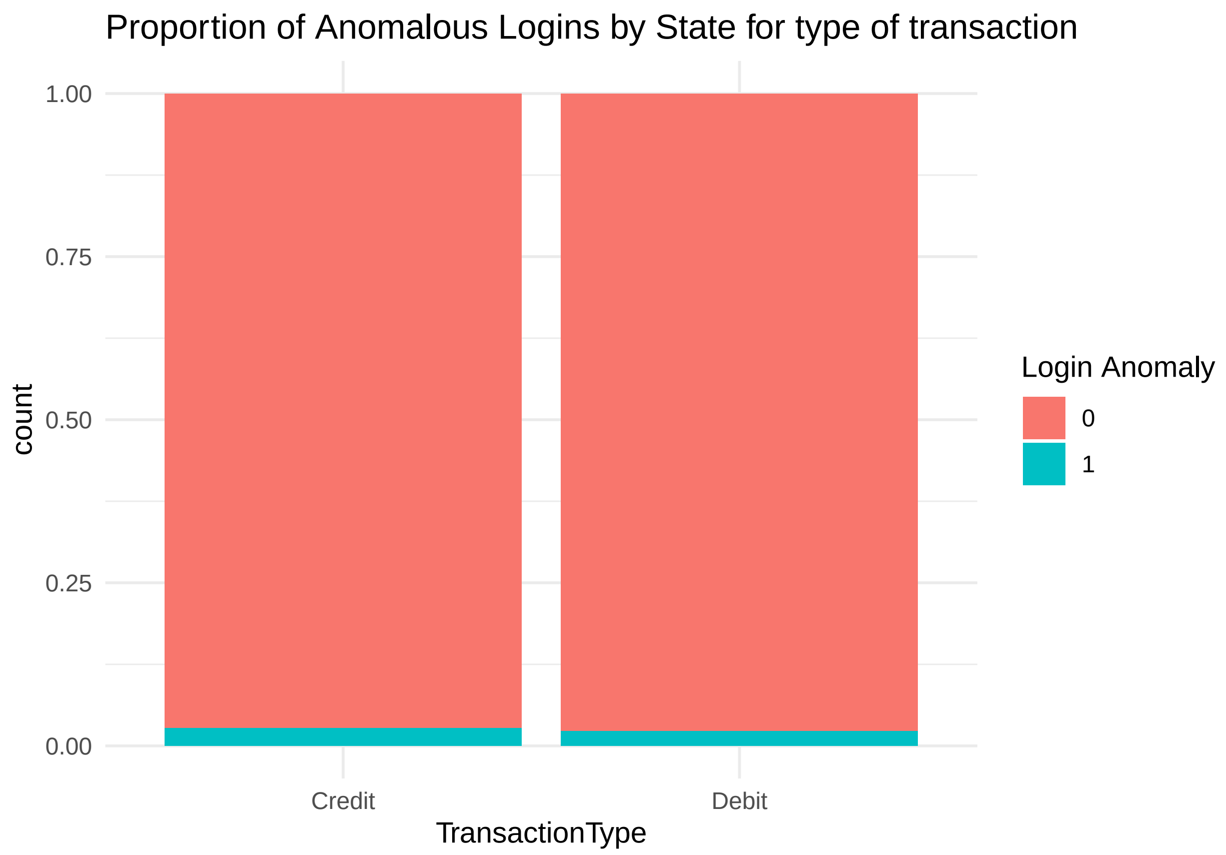


The categorical variables appear to be **mostly well-balanced**, except for the engineered features **IPChange, LoginAnomaly, and DeviceChange**, which have relatively few observations in certain categories. These imbalanced variables are particularly important to monitor in the subsequent analyses, as they may indicate **potential anomalies** and contribute significantly to fraud detection.

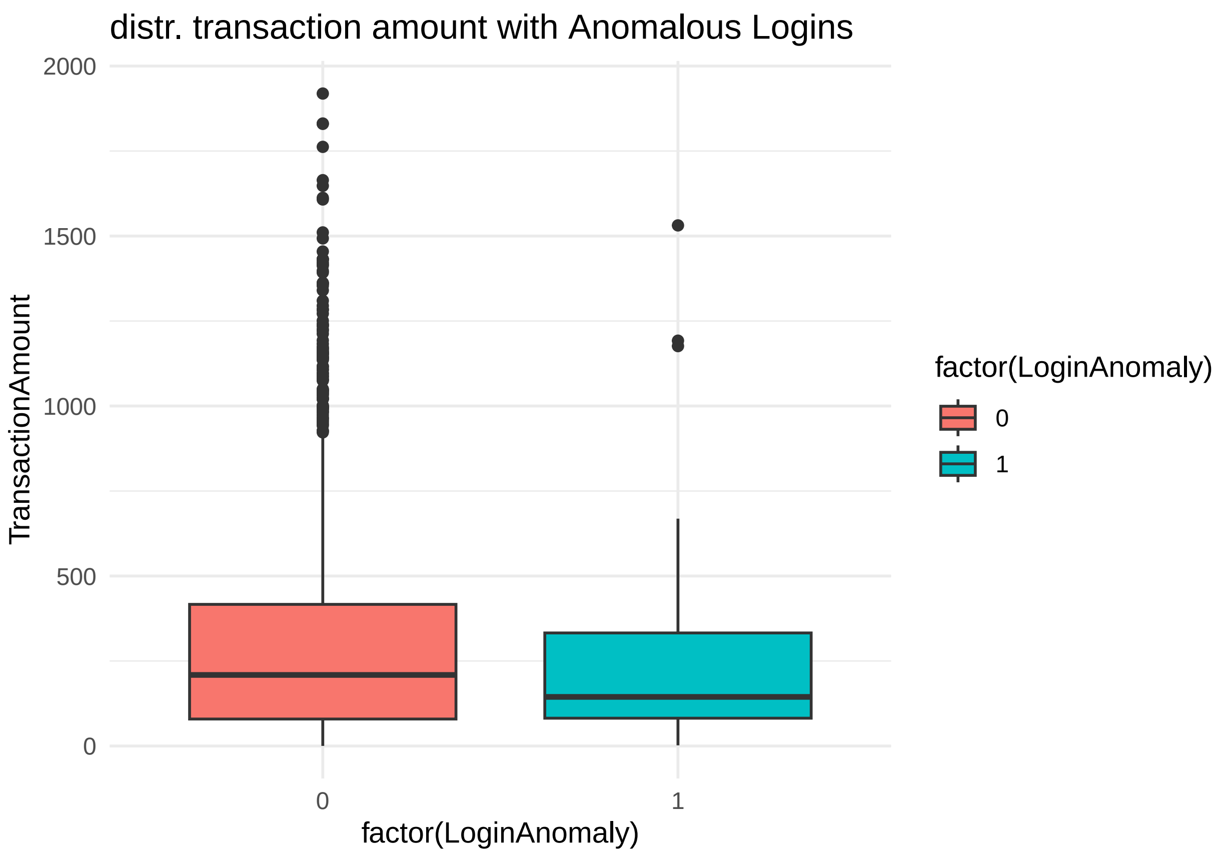
```r
ggplot(bank[bank$LoginAnomaly == 1, ], aes(x = Location, fill = factor(LoginAnomaly))) +
  geom_bar(position = "dodge") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1),
        legend.position = "none") +
  ggtitle("Distribution of Anomalous Logins by State") +
  scale_fill_manual(values = c("red"))
```



```r
#transactions types vs anomalous logins

ggplot(bank, aes(x = TransactionType, fill = factor(LoginAnomaly))) +
  geom_bar(position = "fill") +
  theme_minimal() +
  ggtitle("Proportion of Anomalous Logins by State for type of transaction") +
  labs(fill = "Login Anomaly")
```



```r
# transactions amount

ggplot(bank, aes(x = factor(LoginAnomaly), y = TransactionAmount, fill = factor(LoginAnomaly))) +
  geom_boxplot() +
  theme_minimal() +
  ggtitle("distr. transaction amount with Anomalous Logins")
```



With these visualizations, we analyzed the distribution of **login anomalies** across different dimensions:

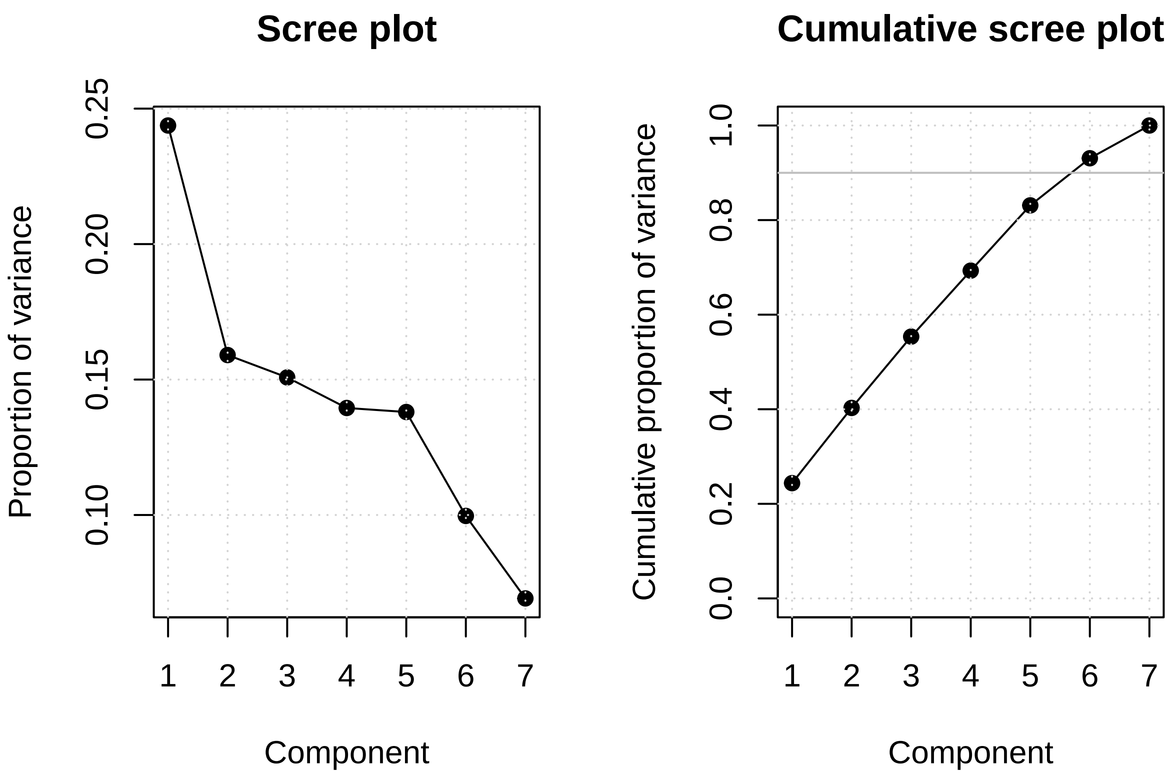
Login Anomalies by Location : - The number of **login anomalies appears to be evenly distributed** across most states, with no particular location showing a significantly higher concentration of anomalies.
- However, some cities—including **Houston, Kansas City, Los Angeles, Jacksonville, Oklahoma, San Diego, and Virginia**—exhibit a slightly **higher frequency of login anomalies**, likely due to an increased number of login attempts compared to other states. Hence, a few states experience a **higher concentration of suspicious login attempts**, which could warrant further investigation.

Transaction Type vs. Login Anomalies : - Login anomalies are **evenly distributed** across **credit and debit transactions**, indicating that fraudulent login attempts are not biased toward a specific transaction type.

Transaction Amount vs. Login Anomalies: - The **distribution of transaction amounts** for login anomalies does not show clear distinctions, suggesting that login anomalies occur across both **high-value and low-value transactions**.

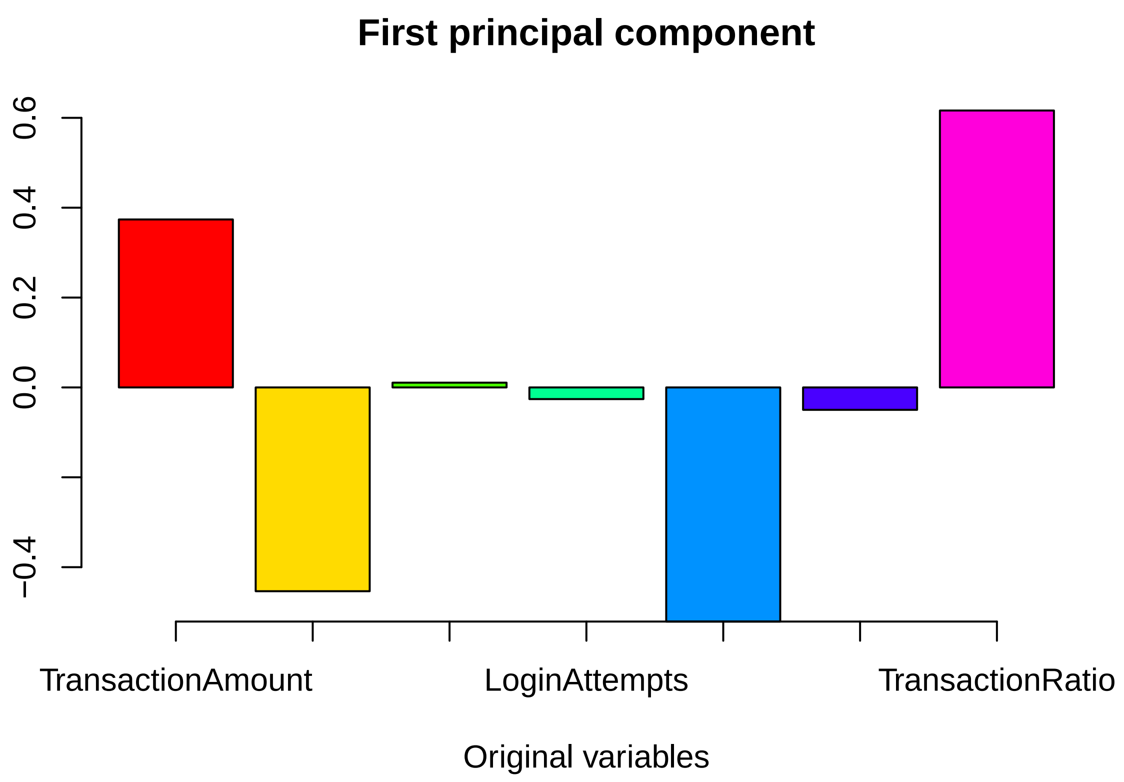
3. PCA

```r
# numeric_data for PCA
num_data <- bank[numeric_vars]

pca_res <- prcomp(num_data, center = TRUE, scale. = TRUE)

# Scree plot
sd <- pca_res$sdev
layout(matrix(1:2, ncol = 2))
plot(sd^2 / sum(sd^2), type = "o", pch = 19, xlab = "Component", main = "Scree plot", ylab = "Proportion
grid()
plot(cumsum(sd^2) / sum(sd^2), type = "o", pch = 19, xlab = "Component", main = "Cumulative scree plot"
grid()
abline(h = 0.9, col = "gray")
```

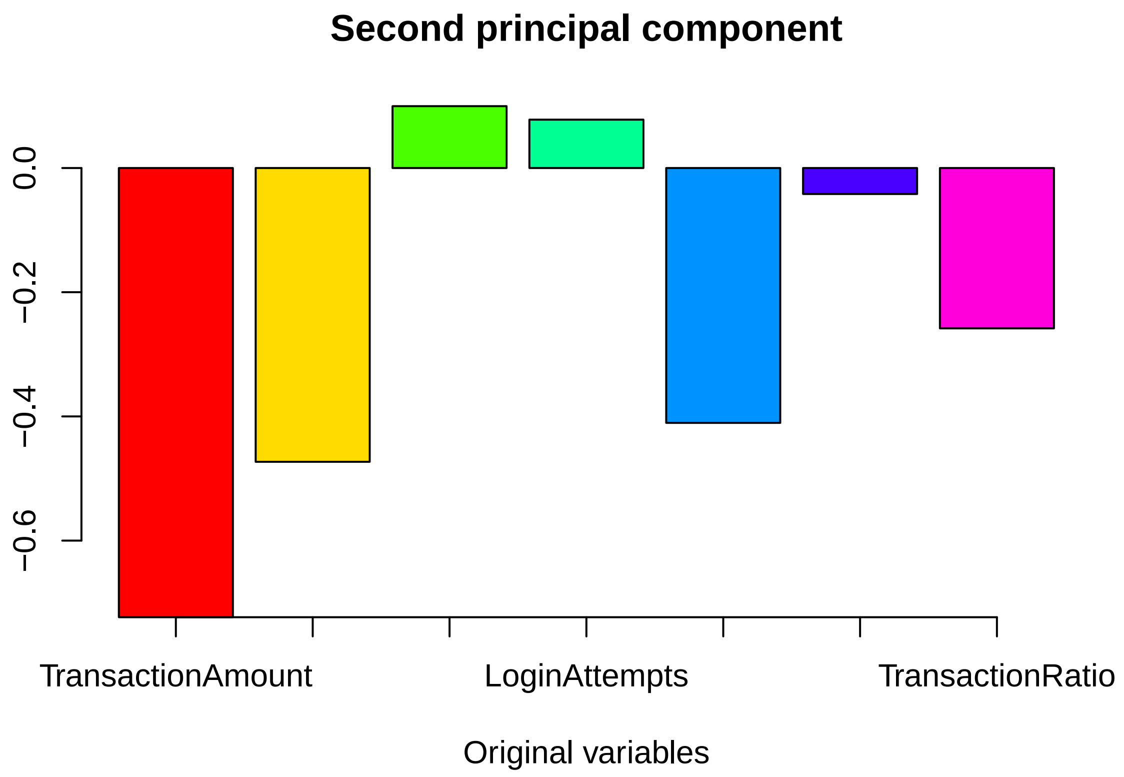


Examining the **scree plot**, there is no clear **elbow point** to decisively determine the optimal number of principal components. However, from the **cumulative variance plot**, we observe that the first **six principal components** account for more than **90% of the explained variance**.
This suggests that we could **reduce the number of variables** while retaining most of the dataset's information, making subsequent analyses more efficient and interpretable.

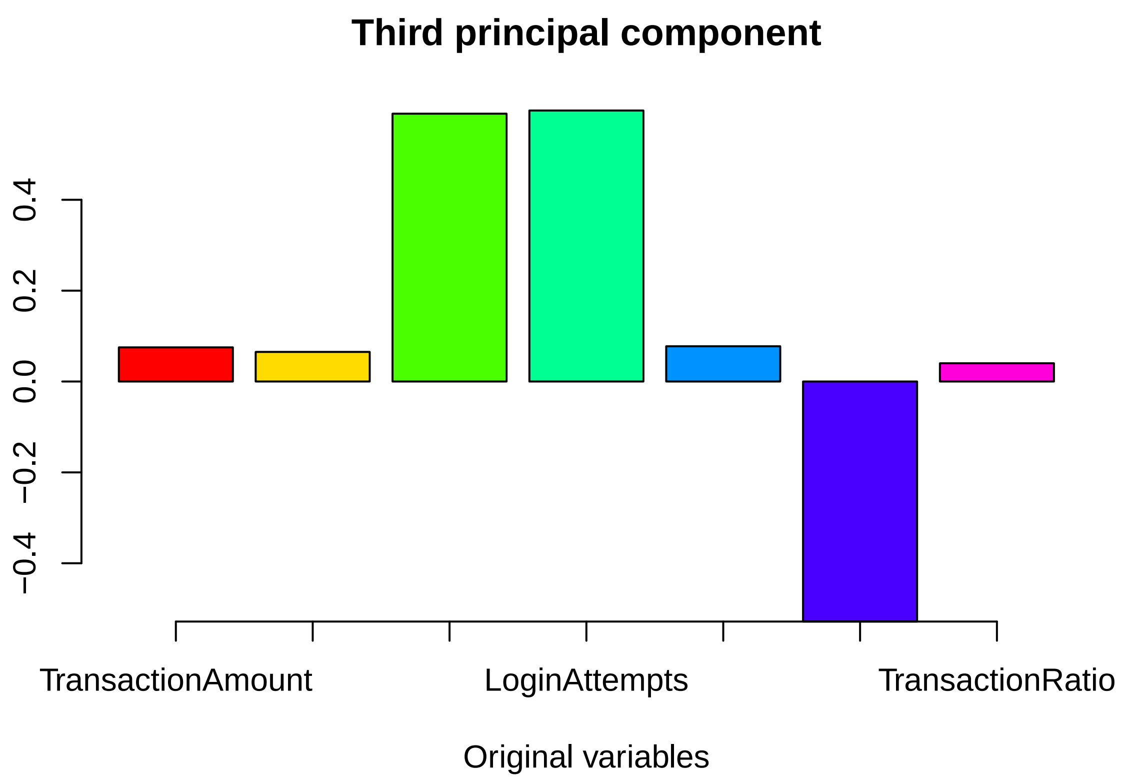
```
# Visualization of loadings
pcs <- pca_res$rotation
p <- dim(num_data)[2]

barplot((pcs[, 1]), beside = TRUE, axis.lty = "solid", col = rainbow(p), main = "First principal compone
```



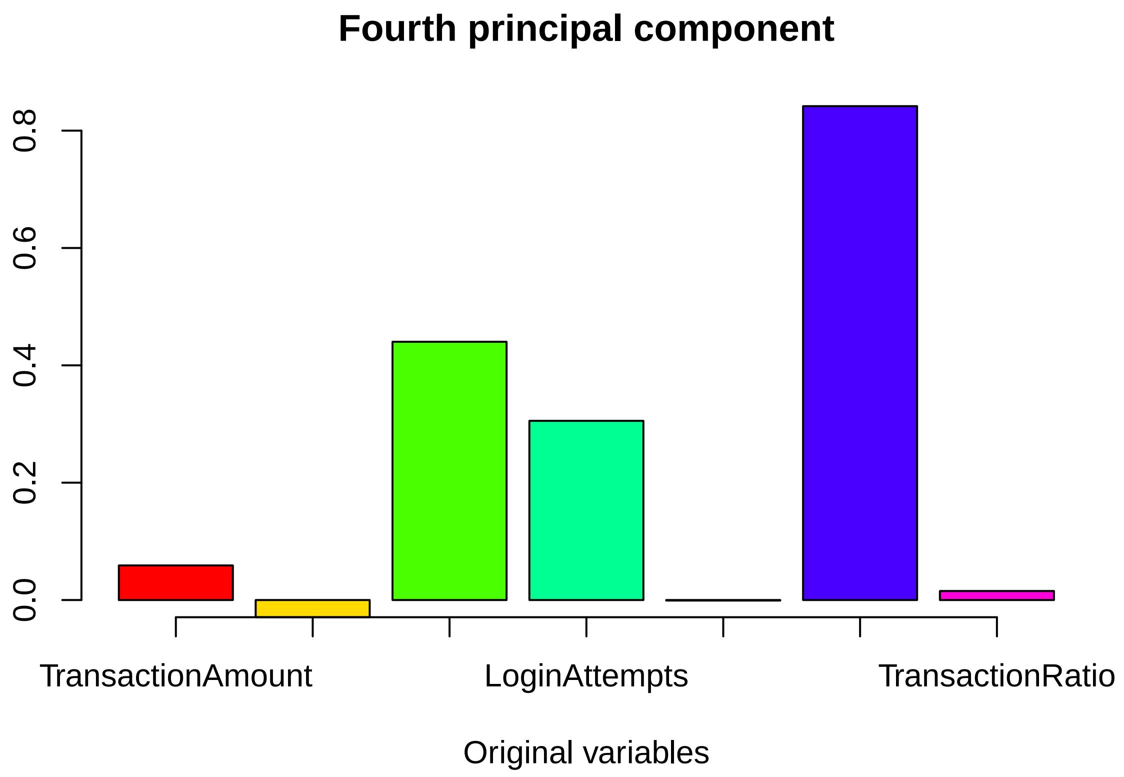
```r
barplot((pcs[, 2]), beside = TRUE, axis.lty = "solid", col = rainbow(p), main = "Second principal compon
```



```r
barplot((pcs[, 3]), beside = TRUE, axis.lty = "solid", col = rainbow(p), main = "Third principal compone
```



```r
barplot((pcs[, 4]), beside = TRUE, axis.lty = "solid", col = rainbow(p), main = "Fourth principal compo
```



Considering the **loadings** and the contribution of each variable to the **first principal components**, **DaysSinceLastTransaction** appears to explain the least variance. However, we have decided to **retain it for further analysis**, as it could serve as a valuable indicator of **potential anomalous transactions**.

Relying solely on **variance explained** as a selection criterion may not be appropriate for our specific analysis, where detecting unusual transaction patterns is a key objective.

4.1 CLUSTERING

After **standardizing the variables** for clustering, we performed **Bartlett's test** to assess the **homogeneity of variances** among them.

Bartlett's test is used to check whether multiple groups (in this case, the selected numerical variables) have **equal variances**—a key assumption for certain clustering algorithms, particularly those relying on distance metrics like **K-Means and Hierarchical Clustering**. When variances differ significantly, distance-based methods may give **undue weight to high-variance variables**, potentially distorting cluster formation.

The test result shows a **very low p-value ($<$ 2.2e-16)**, indicating that the assumption of **homogeneous variances is violated**. This suggests that the variables have **significantly different variances**, which should be carefully considered in the clustering analysis. In cases like this, alternative approaches such as **PCA for dimensionality reduction or scaling transformations** may be useful to mitigate variance-related biases.

```
bartlett.test(num_data)
```

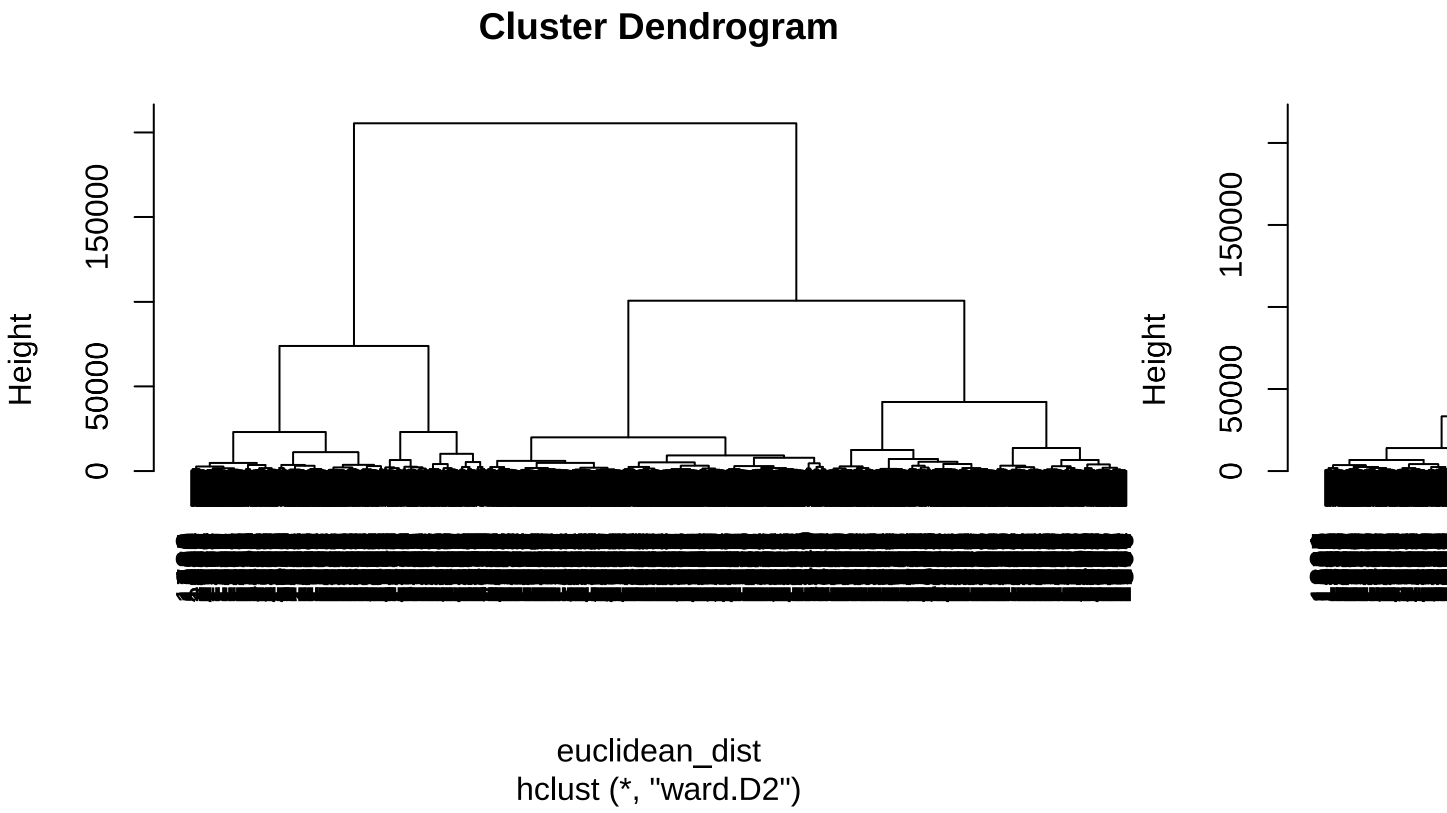
```
##
##  Bartlett test of homogeneity of variances
##
## data:  num_data
## Bartlett's K-squared = 109766, df = 6, p-value < 2.2e-16
```

```
##### data preparation for clustering ########

bank$LogTransactionAmount <- log(bank$TransactionAmount + 1)
bank$LogTransactionRatio <- log(bank$TransactionRatio + 1)
# Standardizzazione delle variabili quantitative
bank$StandardizedTransactionAmount <- scale(bank$LogTransactionAmount)
bank$StandardizedTransactionRatio <- scale(bank$LogTransactionRatio)
bank$StandardizedTransactionDuration <- scale(bank$TransactionDuration)
bank$StandardizedLoginAttempts <- scale(bank$LoginAttempts)
bank$StandardizedAccountBalance <- scale(bank$AccountBalance)

categorical_vars <- c("TransactionType", "Location", "Channel", "CustomerOccupation")
one_hot_encoded_data <- model.matrix(~ TransactionType + Location + Channel + CustomerOccupation - 1, da

bank$StandardizedDaysSinceLastTransaction = scale(bank$DaysSinceLastTransaction)
bank$StandardizedIPChange = scale(as.numeric(bank$IPChange))


final_data <- cbind(bank[, c("StandardizedTransactionAmount", "StandardizedTransactionRatio",
                             "StandardizedTransactionDuration", "StandardizedLoginAttempts", "Standardi


######## Distances ########

euclidean_dist <- dist(num_data, method = "euclidean")
manhattan_dist <- dist(num_data, method = "manhattan")
minkowski_dist <- dist(num_data, method = "minkowski", p = 1)

# 3. Standardized Scores
```

```r
std_scores <- pca_res$x / matrix(pca_res$sdev, nrow = dim(num_data)[1], ncol = dim(num_data)[2], byrow
# 4. Mahalanobis Distance
mahalanobis_dist <- dist(std_scores)
```

We plotted **all types of dendrograms** using various combinations of **distance metrics** and **linkage methods**. Across all cases, the results consistently suggest an optimal division into **two clusters**.

The **best representation** is provided by **hierarchical clustering** using the **Euclidean and Minkowski distances** with the **Ward.D2 linkage method**, as it produces well-separated and interpretable clusters.



4.2 DBSCAN

Density-Based Clustering for Anomaly Detection

What is DBSCAN?

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an **unsupervised clustering algorithm** that groups data points based on **density**. Unlike **K-Means** or **Hierarchical Clustering**, DBSCAN does **not require** specifying the number of clusters beforehand. Instead, it identifies clusters as **regions of high density separated by sparse areas**, making it particularly effective for **detecting anomalies and outliers** that do not belong to any dense region.

DBSCAN relies on two key parameters:
- **eps (epsilon)**: Defines the maximum distance between points to be considered part of the same cluster.
- **`minPts`**: The minimum number of points required to form a dense region (i.e., a cluster).

The algorithm classifies points into three categories:
1. **Core points**: Points with at least `minPts` neighbors within `eps`.
2. **Border points**: Points that do not meet `minPts` but are within `eps` of a core point. 3. **Noise (outliers)**:

Points that do not belong to any cluster, making them potential **anomalies**.

DBSCAN is **well-suited for anomaly detection**, as fraudulent transactions often do not conform to normal patterns and appear as **isolated points**. By identifying transactions that do not belong to any dense cluster (i.e., labeled as **noise points**), DBSCAN helps **detect suspicious activities** without relying on predefined labels.

In this analysis, we applied DBSCAN using the **dbscan** library in R. The dataset was preprocessed by standardizing numerical features to ensure proper distance calculations. The clustering was performed with:
- `eps = 0.6`: Defines the neighborhood radius.
- `minPts = 8`: Ensures a minimum density for cluster formation.

```r
final_data <- as.data.frame(final_data)
set.seed(123)
db <- dbscan((final_data[,-c(6,7)]), eps = 0.6, minPts = 8)
final_data$Cluster <- as.numeric(db$cluster)
db
```

```
## DBSCAN clustering for 2016 objects.
## Parameters: eps = 0.6, minPts = 8
## Using euclidean distances and borderpoints = TRUE
## The clustering contains 1 cluster(s) and 256 noise points.
##
##    0    1
##  256 1760
##
## Available fields: cluster, eps, minPts, metric, borderPoints
```

We can say that we have 256 noise observations, in the next steps we'll focus our attention on understanding the nature of those datapoints.

5. ANOMALY DETECTION (LOF)

The **Local Outlier Factor (LOF)** is an anomaly detection method that measures the degree of isolation of a data point relative to its local neighborhood. Unlike global outlier detection techniques, LOF compares the local density of a point to its neighbors, identifying observations that significantly deviate from their surroundings.

By setting $k = 10$, we achieve a robust detection of local anomalies, ensuring that fraudulent transactions stand out relative to their immediate neighborhood without excessive smoothing of density estimates.

```r
lof_scores <- lof((final_data[,-c(6,7)]), k = 10)
final_data$LOF <- lof_scores
```

We aimed to filter our data by creating a **context flag**, a set of characteristics that could indicate a fraudulent transaction. This flag, used alongside LOF scores, allowed us to isolate a series of **"suspicious"** transactions within the first cluster, which already contained points classified as **"noisy"** by DBSCAN.

Why We Chose the 99th Percentile as a Threshold?

In financial transactions, fraudulent activities often involve extreme values in key variables. To systematically identify unusual patterns, we set **thresholds at the 99th percentile** for several standardized numerical features:

- **Transaction Amount (`StandardizedTransactionAmount`)**: Large transaction amounts are often associated with fraudulent behavior, as fraudsters attempt to withdraw as much as possible in a short period.

- **Transaction Ratio (`StandardizedTransactionRatio`)**: This metric captures the ratio of transaction value to account balance. Anomalous ratios may indicate suspicious transfers, such as account draining.

- **Login Attempts (`StandardizedLoginAttempts`)**: Multiple failed login attempts are a red flag for unauthorized access attempts.

- **Days Since Last Transaction (`StandardizedDaysSinceLastTransaction`)**: Fraudulent transactions often occur after long inactivity periods, exploiting dormant accounts.

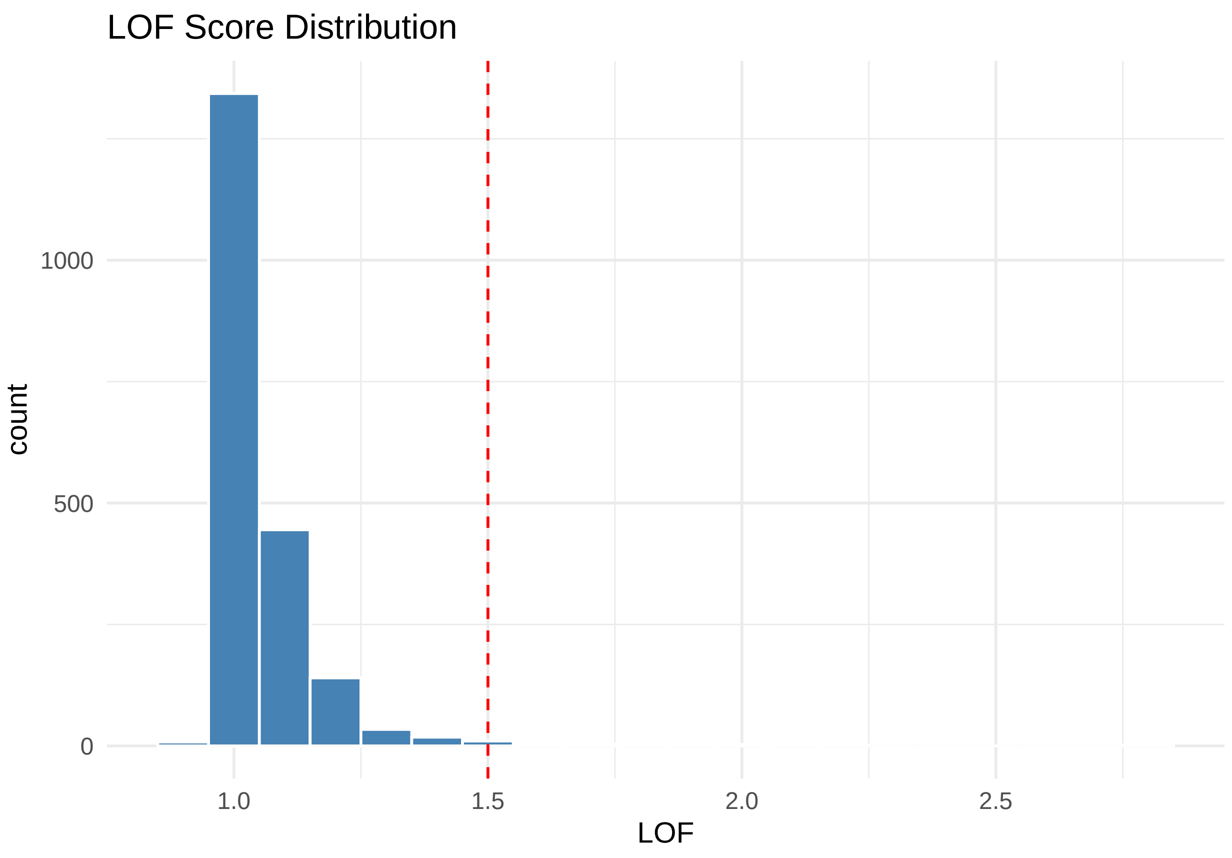
- **IP Changes (`StandardizedIPChange`)**: Frequent IP address changes can indicate attempts to mask fraudulent activity.

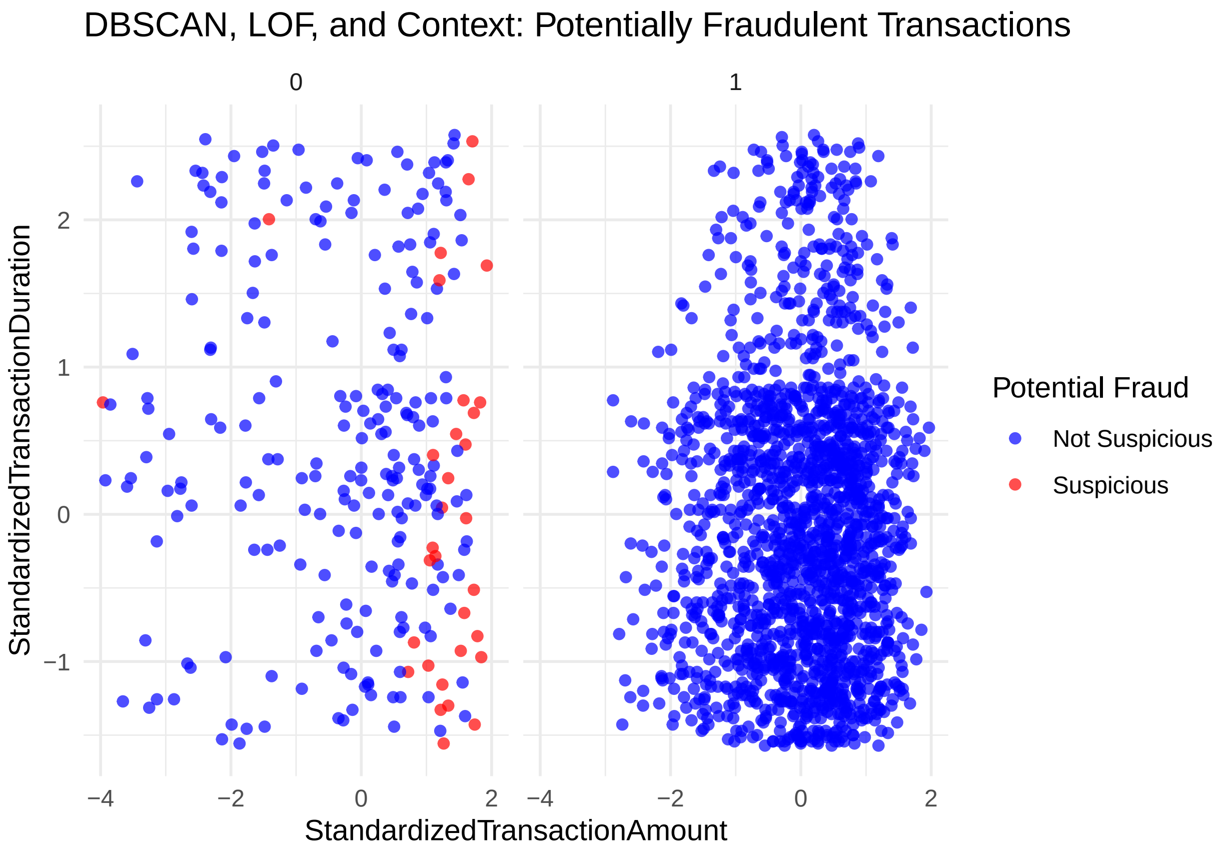
By flagging transactions exceeding these thresholds (**ContextFlag = 1**), we highlight transactions that significantly deviate from normal behavior. Combining **LOF scores (>1.5)** with **DBSCAN noise points (Cluster == 0)** ensures a robust fraud detection framework by detecting anomalies both in density-based clustering and through individual outlier analysis.

We chose **LOF > 1.5** as the threshold for potential fraud because it indicates that a transaction's local density is significantly lower than its neighbors, suggesting it is an anomaly. This value balances **false positives and false negatives**, ensuring that only truly suspicious transactions are flagged while avoiding excessive noise.

LOF scores distribution

```
### Visualization of Results ###

# Distribution of LOF scores with a threshold of 1.5
### Identification of Potential Fraud Cases ###
# A transaction is considered potentially fraudulent if:
# - It has LOF > 1.5 and contextual signals (ContextFlag == 1)
# OR
# - It is labeled as noise by DBSCAN (Cluster == 0) and presents contextual signals.
ggplot(final_data, aes(x = LOF)) +
  geom_histogram(binwidth = 0.1, fill = "steelblue", color = "white") +
  geom_vline(xintercept = 1.5, color = "red", linetype = "dashed") +
  ggtitle("LOF Score Distribution") +
  theme_minimal()
```


LOF Score Distribution
count
1000
500
0
1.0
1.5
2.0
2.5
LOF

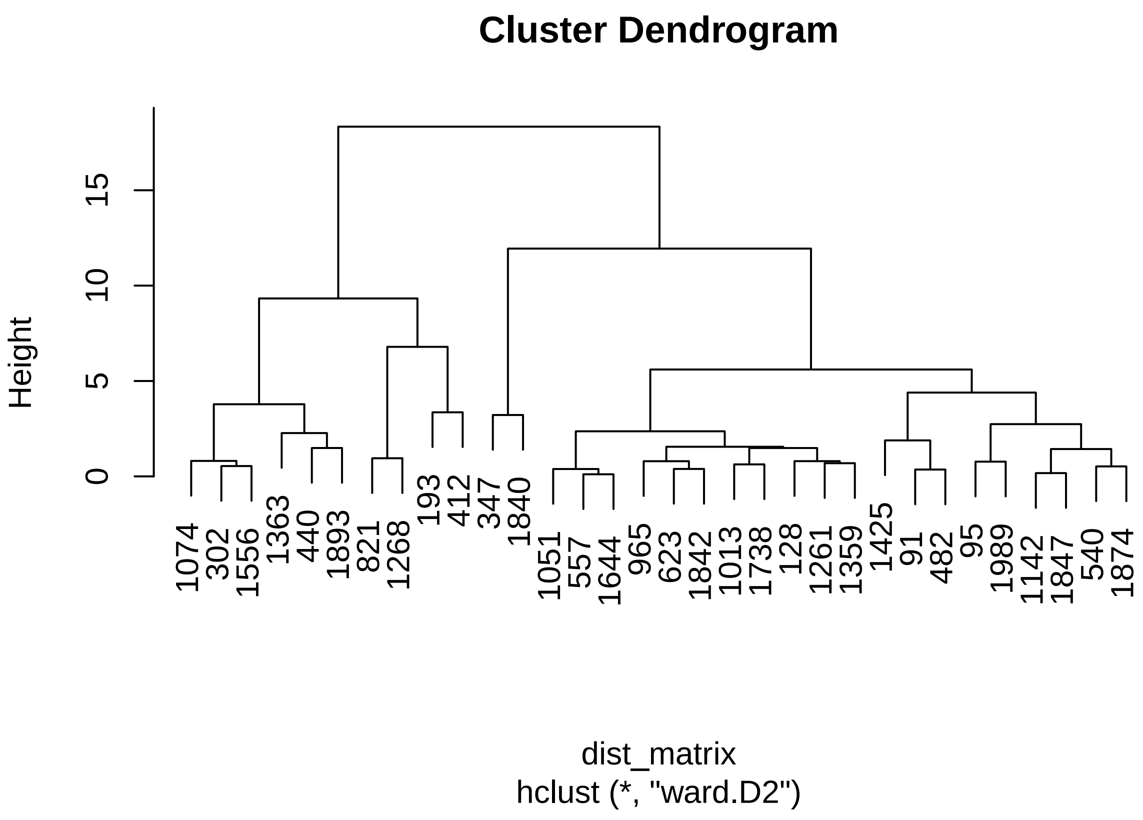

DBSCAN, LOF, and Context: Potentially Fraudulent Transactions
0
1
StandardizedTransactionDuration
2
1
0
−1
−4
−2
0
2
StandardizedTransactionAmount
Potential Fraud
Not Suspicious
Suspicious

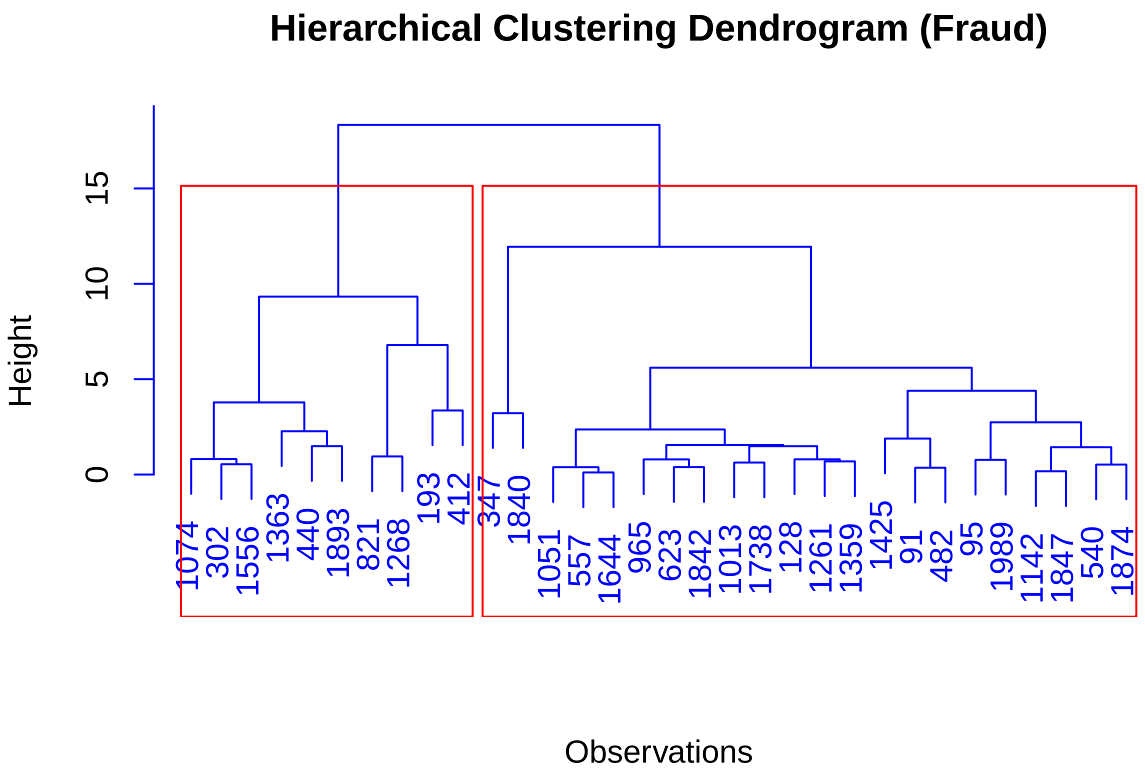
From now on we want to perform a clustering on those suspicious transactions

Hierarchical clustering approach

We choose minPts = log(32) , that is the nrows of outliers data

```
# Filter only suspicious observations
outlier_data <- final_data[final_data$PotentialFraud == 1, -c(6,7)]

# Compute the distance matrix
dist_matrix <- dist(outlier_data, method = "euclidean")

# Hierarchical clustering using Ward's method
hc <- hclust(dist_matrix, method = "ward.D2")
plot(hc)
```



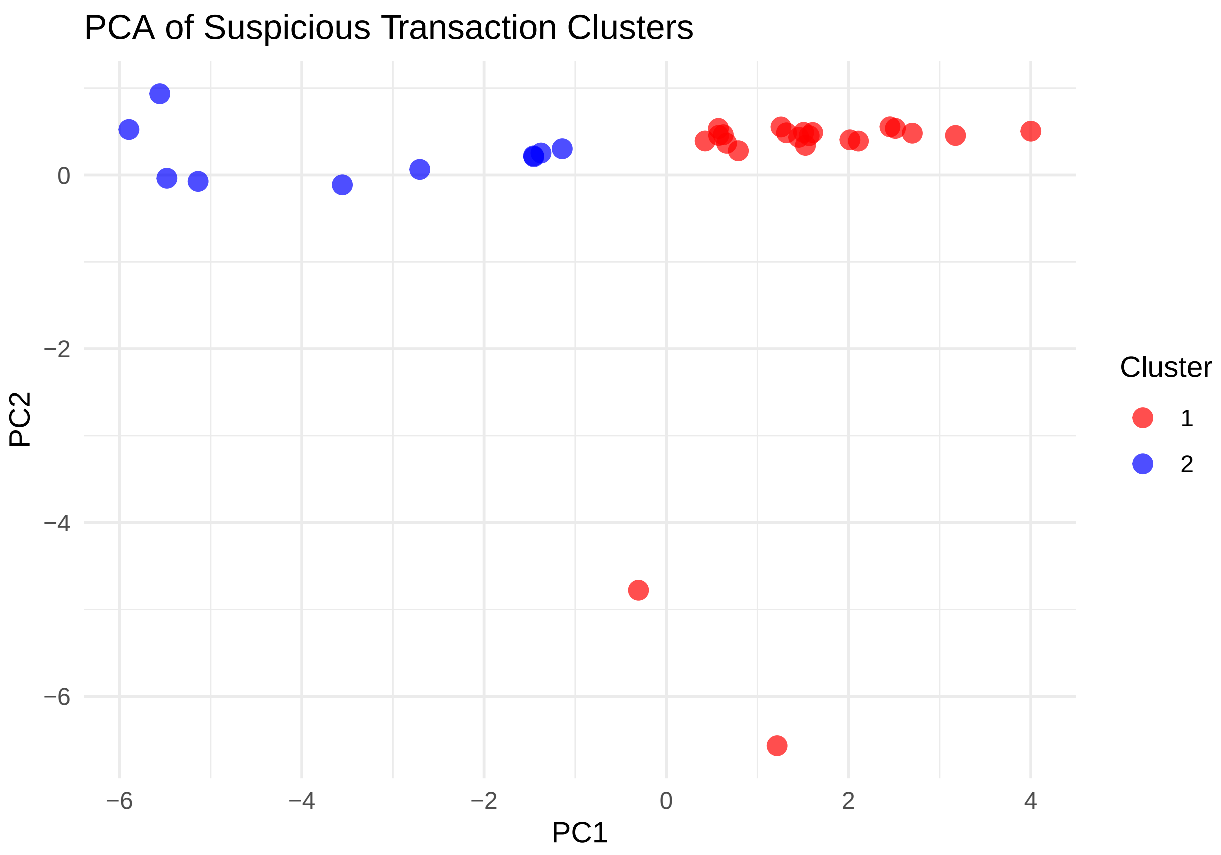


PCA on Suspicious Transactions

This code applies **Principal Component Analysis (PCA)** to visualize suspicious transactions and detect patterns.

- **PCA is performed** on flagged fraudulent transactions to reduce dimensionality.
- **Clusters are assigned** using hierarchical clustering.
- **A scatter plot** of the first two principal components (`PC1` and `PC2`) highlights potential fraud patterns.

This helps in understanding **different fraud behaviors** and grouping similar anomalies.

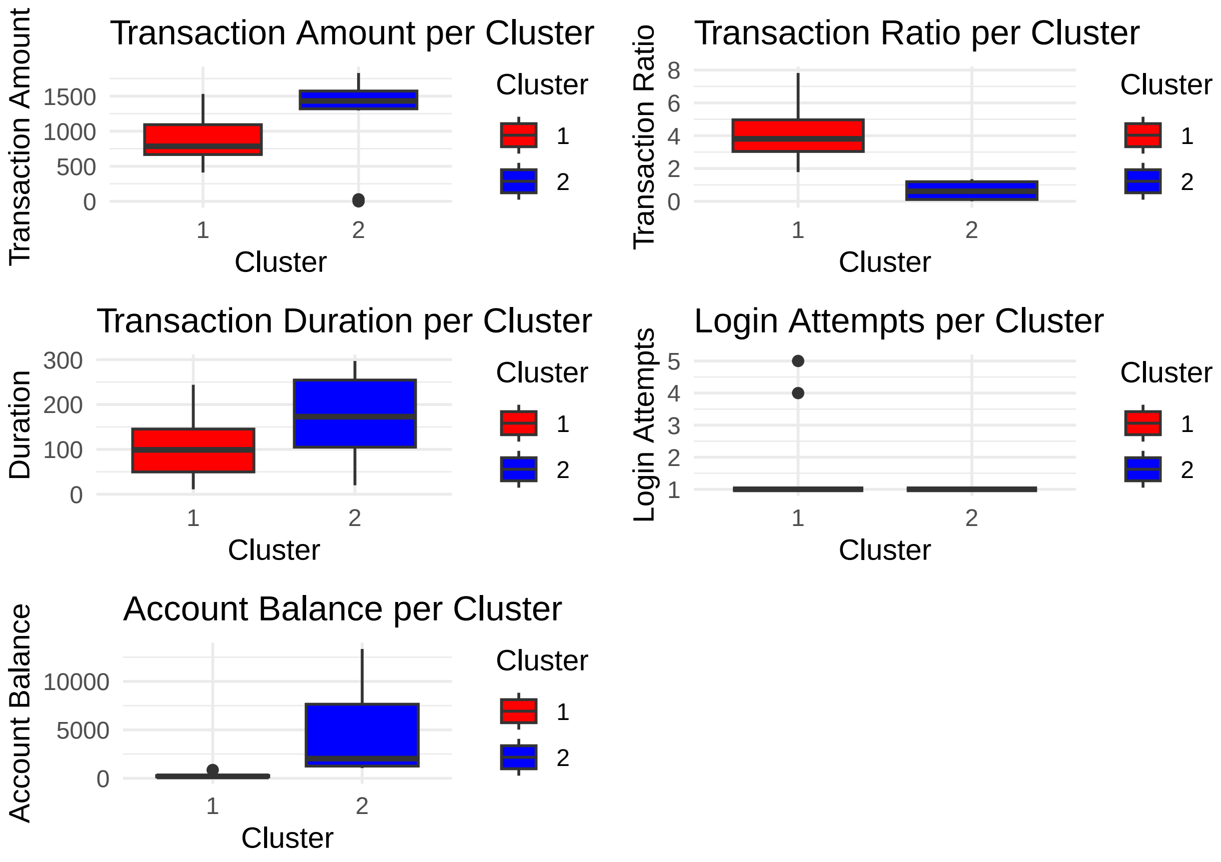


Cluster Comparison of Suspicious Transactions

This code assigns **cluster labels** to suspicious transactions and compares their characteristics with the original dataset.

- **Matching original data**: Using row indices, we extract key variables (`TransactionAmount`, `TransactionRatio`, etc.) from the **bank** dataset.
- **Boxplot analysis**: We visualize differences between clusters based on financial metrics, helping to identify patterns in fraudulent behavior.
- **Multi-variable comparison**: Factors like **transaction amount, login attempts, and account balance** are analyzed to assess how clusters differ.

This approach helps in understanding **distinct fraud profiles** within the detected anomalies.

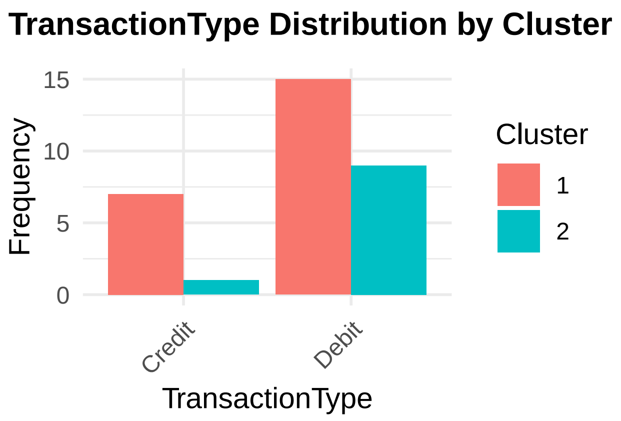


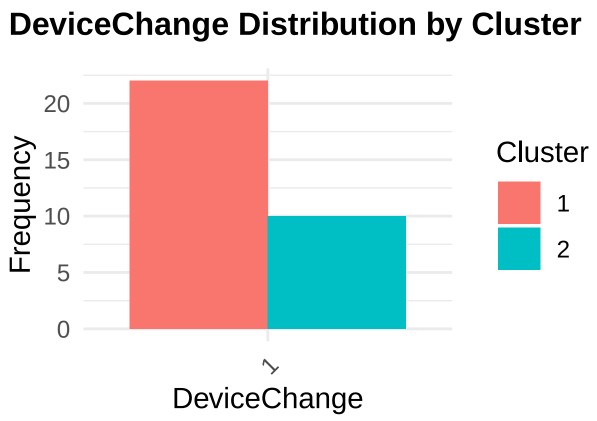
The boxplots highlight differences between **Cluster 1 (red) and Cluster 2 (blue)**:

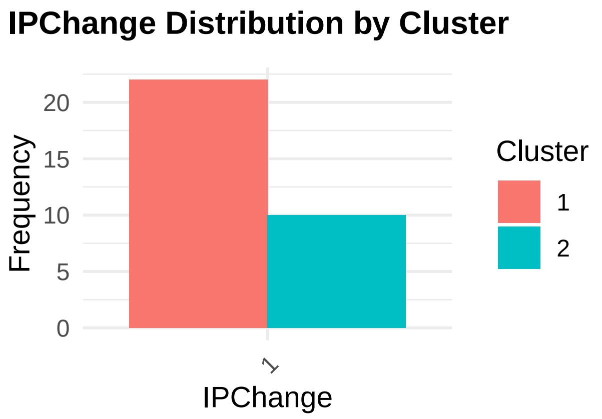
- **Cluster 2**: Higher **transaction amounts** and **account balances**, suggesting large-scale fraud.
- **Cluster 1**: Higher **transaction ratio**, possibly indicating **account draining** tactics.
- **Transaction Duration**: Longer in **Cluster 2**, potentially due to additional security checks.
- **Login Attempts**: Cluster 1 shows **failed login outliers**, suggesting unauthorized access attempts.

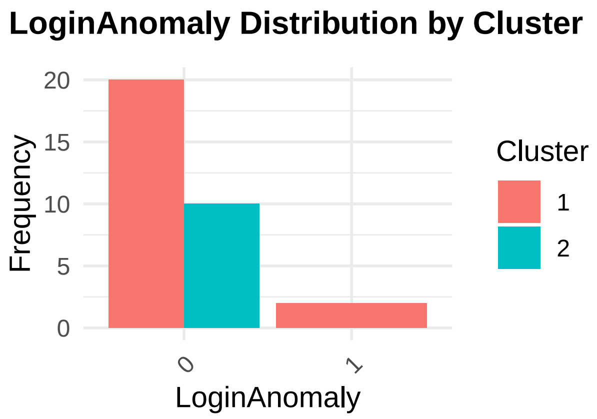
These differences reveal **distinct fraud patterns**, helping refine fraud detection strategies.

Now let's consider categorical variable to figure out who is the dishonest one !









The bar charts compare **Transaction Type, Device Change, IP Change, and Login Anomalies** across the two clusters.

Key Observations:

- **Cluster 1 (red)** has a **higher frequency** of:
  - **Debit transactions**, suggesting potential account draining.
  - **Device and IP changes**, indicating possible attempts to evade detection.
  - **Login anomalies**, suggesting unauthorized access attempts.
- **Cluster 2 (blue)** shows a lower frequency of these risk factors.

Most Suspicious Cluster:

Cluster 1 appears more suspicious, as it exhibits more fraudulent indicators such as frequent **device/IP changes and login anomalies**, which are common in fraud attempts.

Conclusions

After performing two different clustering approaches—one on the dataset's numerical covariates and another on data points that were not assigned to any cluster—we used a set of **indicators**, including qualitative variables, to determine which of the two final clusters exhibited potentially fraudulent characteristics.

We approached this analysis as if it had been commissioned by a **bank**, assuming they provided us with common fraud indicators. These indicators were consistently found in **Cluster 1**, suggesting a higher likelihood of fraudulent behavior.

Rather than assigning explicit **fraud labels (0 = not fraud, 1 = fraud)**, we chose to avoid introducing a **non-deterministic classification** into the dataset. Instead, our goal was to provide the client with **recurring fraud patterns**, highlighting how certain **"outlier"** transactions tend to follow specific behaviors.