Experiment No.:- 1

Aim:- Basic Statistics and Visualizations in R

Source Code:-

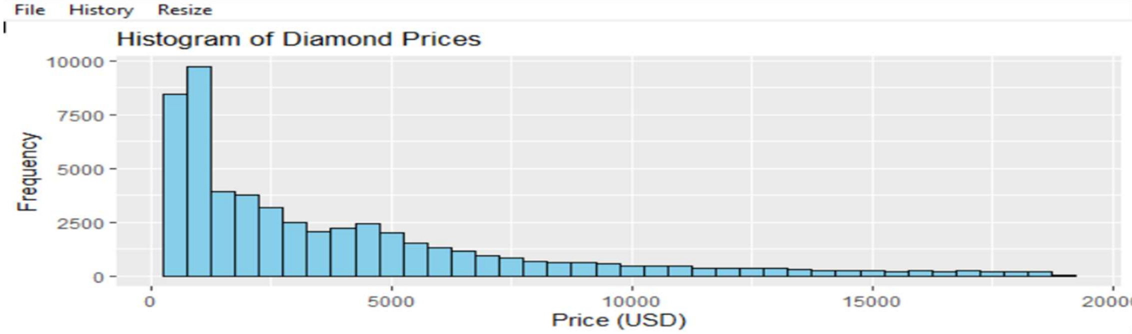
```
# Load libraries
library(ggplot2)
library(dplyr)

# Load dataset
data("diamonds")

cat("=== Summary Statistics of Entire Dataset ===\n")

mean_price <- mean(diamonds$price)
median_price <- median(diamonds$price)
sd_price <- sd(diamonds$price)
var_price <- var(diamonds$price)

cat("\n=== Price Statistics ===\n")
cat("Mean: ", mean_price, "\n")
cat("Median: ", median_price, "\n")
cat("Standard Deviation: ", sd_price, "\n")
cat("Variance: ", var_price, "\n")

mean_carat <- mean(diamonds$carat)
median_carat <- median(diamonds$carat)
sd_carat <- sd(diamonds$carat)
var_carat <- var(diamonds$carat)

cat("\n=== Carat Statistics ===\n")
cat("Mean: ", mean_carat, "\n")
cat("Median: ", median_carat, "\n")
cat("Standard Deviation: ", sd_carat, "\n")
cat("Variance: ", var_carat, "\n")

# Correlation
correlation <- cor(diamonds$carat, diamonds$price)
cat("\n=== Correlation ===\n")
cat("Correlation between carat and price: ", correlation, "\n")

# 1. Histogram
dev.new()  # opens new plot window
print(ggplot(diamonds, aes(x = price)) +
      geom_histogram(binwidth = 500, fill = "skyblue", color = "black") +
      labs(title = "Histogram of Diamond Prices", x = "Price (USD)", y = "Frequency"))
```

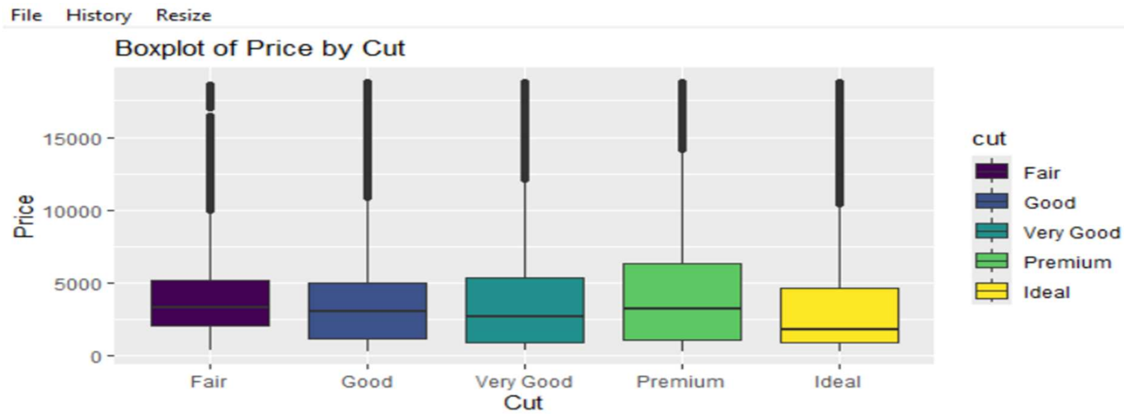
```
# 2. Boxplot
dev.new()
print(ggplot(diamonds, aes(x = cut, y = price, fill = cut)) +
      geom_boxplot() +
      labs(title = "Boxplot of Price by Cut", x = "Cut", y = "Price"))

# 3. Scatter Plot
dev.new()
print(ggplot(diamonds, aes(x = carat, y = price, color = cut)) +
      geom_point(alpha = 0.4, size = 1) +
      labs(title = "Scatter Plot of Carat vs Price", x = "Carat", y = "Price") +
      theme_minimal())

# 4. Bar Plot
dev.new()
print(ggplot(diamonds, aes(x = clarity, fill = clarity)) +
      geom_bar() +
      labs(title = "Bar Plot for Number of Diamonds by Clarity", x = "Clarity", y = "Count") +
      theme_minimal())

# 5. Density Plot
dev.new()
print(ggplot(diamonds, aes(x = price, fill = color)) +
      geom_density(alpha = 0.5) +
      labs(title = "Density Plot of Price by Color", x = "Price", y = "Density") +
      theme_minimal())
```

Output:-

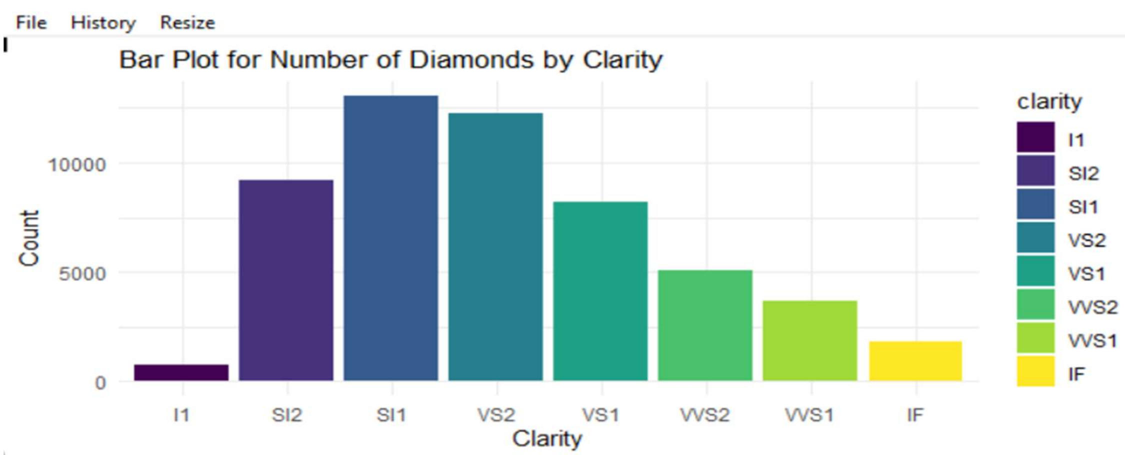
```
R 4.3.2 · ~/
> source("C:/Users/ansu2/Downloads/dv.R")
=== Summary Statistics of Entire Dataset ===

=== Price Statistics ===
Mean:   3932.8
Median:   2401
Standard Deviation:   3989.44
Variance:   15915629

=== Carat Statistics ===
Mean:   0.7979397
Median:   0.7
Standard Deviation:   0.4740112
Variance:   0.2246867

=== Correlation ===
Correlation between carat and price:   0.9215913
```

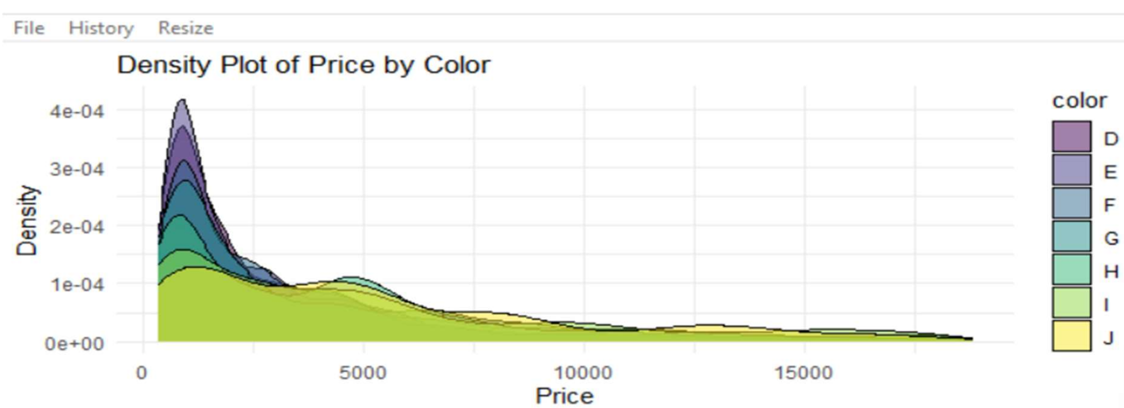
File History Resize




File History Resize
Boxplot of Price by Cut
Price
15000
10000
5000
0
Fair
Good
Very Good
Premium
Ideal
Cut
cut
Fair
Good
Very Good
Premium
Ideal


File History Resize
Scatter Plot of Carat vs Price
Price
15000
10000
5000
0
0
1
2
3
4
5
Carat
cut
Fair
Good
Very Good
Premium
Ideal


File History Resize
Bar Plot for Number of Diamonds by Clarity
Count
10000
5000
0
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF
Clarity
clarity
I1
SI2
SI1
VS2
VS1
VVS2
VVS1
IF


File History Resize
Density Plot of Price by Color
Density
4e-04
3e-04
2e-04
1e-04
0e+00
0
5000
10000
15000
Price
color
D
E
F
G
H
I
J

Experiment No.:- 2

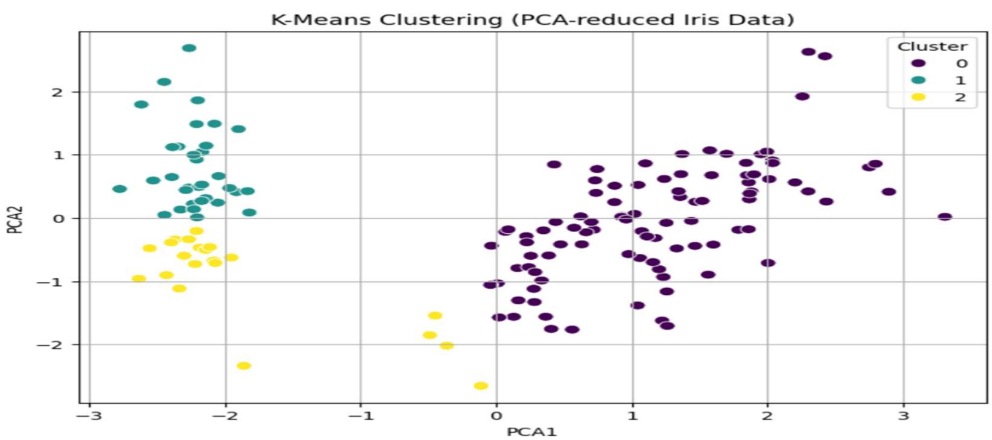
Aim: K- Means clustering in Python

Source Code:-

```
# Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import confusion_matrix
import seaborn as sns

# Load Iris dataset
iris = load_iris()
X = iris.data
y = iris.target
target_names = iris.target_names

# Standardize features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# K-Means clustering
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

# Reduce dimensions for visualization
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Create DataFrame with PCA and cluster info
df = pd.DataFrame(X_pca, columns=['PCA1', 'PCA2'])
df['Cluster'] = clusters
df['True Label'] = y

# Plot clusters
plt.figure(figsize=(8, 6))
sns.scatterplot(data=df, x='PCA1', y='PCA2', hue='Cluster', palette='viridis', s=60)
plt.title("K-Means Clustering (PCA-reduced Iris Data)")
plt.grid(True)
plt.show()

# Confusion Matrix
print("Confusion Matrix (Cluster vs True Label):")
cm = confusion_matrix(y, clusters)
```

```
from scipy.stats import mode

labels = np.zeros_like(clusters)
for i in range(3):
    mask = (clusters == i)
    labels[mask] = mode(y[mask], keepdims=False).mode

# Recomputed confusion matrix
cm_corrected = confusion_matrix(y, labels)

# Display the corrected confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm_corrected, annot=True, fmt='d',
            xticklabels=target_names, yticklabels=target_names, cmap='Blues')
plt.xlabel("Predicted Label (after mapping)")
plt.ylabel("True Label")
plt.title("Corrected Confusion Matrix")
plt.tight_layout()
plt.show()
```

Output:-



Confusion Matrix (Cluster vs True Label):