EXPERIMENT NO 6

Aim- Write a program to draw a scatter diagram, residual plots, outliers leverage and influential data points in R.

Theory:

Scatter Diagram: This plot visually depicts the relationship between the independent (x) and dependent (y) variables. Outliers, which deviate significantly from the overall trend, can be readily identified.

Residual Plot: The residuals represent the difference between the observed y-values and the fitted y-values from the regression line. A well-fitted model should have residuals randomly scattered around the horizontal line (y=0), indicating no systematic pattern.

Outliers: Outliers are data points that fall far away from the majority of the data in a scatter plot. They can potentially distort the regression line and affect the model's accuracy. Studentized residuals (obtained by dividing residuals by their standard error) are commonly used to detect outliers, with absolute values exceeding 2.5 being considered potential outliers. However, the threshold may vary depending on the specific context and distribution of your data.

Leverage: Leverage measures how far a data point is from the average of the independent variables (x-values). High leverage points have a greater influence on the regression line. While not necessarily outliers themselves, they can become influential if they also have large residuals.

Influential Points: These are data points that have a substantial impact on the slope and intercept of the regression line. Cook's distance is a commonly used diagnostic tool to identify influential points. Values exceeding 0.5 (or one times the number of predictors) are often considered influential.

Cook's distance:cooks_distance <- cooks.distance(model)
Visualizing Outliers, Leverage, and Influential Points:Overlay the identified outliers, leverage points, and influential points on the scatter plot for visualization.

Overlay outliers, leverage points, and influential points on the scatter plot:points(X[outliers], Y[outliers], col = "red", pch = 19),points(X[leverage > threshold], Y[leverage > threshold], col = "blue", pch = 17) points(X[cooks_distance > threshold],
Y[cooks_distance > threshold], col = "green", pch = 15)

Source Code:

R-script

```
# Print the mtcars dataset
print(mtcars)

# Fit a regression model
model <- lm(mpg ~ disp + hp, data = mtcars)
summary(model)  # View model summary

# Calculate leverage for each observation in the model
hats <- as.data.frame(hatvalues(model))
hats  # Display leverage stats for each observation

# Sort observations by leverage, descending
sorted_hats <- hats[order(hats[['hatvalues(model)']], decreasing = TRUE), ]
print(sorted_hats)

# Plot leverage values for each observation
plot(hatvalues(model), type = 'h')  # Generates vertical lines in the graph

# Residual vs. Fitted Plot
res <- resid(model)  # Get list of residuals
standard_res <- rstandard(model)  # Standardized residuals
library(MASS)  # For studentized residuals, load library(MASS)
student_res <- studres(model)  # Studentized residuals
plot(fitted(model), res)  # Generate residual vs. fitted plot
abline(0, 0)  # Add a horizontal line at 0

# Q-Q Plot
qqnorm(res)  # Create Q-Q plot for residuals
qqline(res)  # Add a straight diagonal line to the plot

# Density plot of residuals
plot(density(res))

# Influential Data-points
# Create data frame with outliers
outliers <- data.frame(
  x = c(1, 2, 2, 3, 4, 5, 7, 3, 2, 12, 11, 15, 14, 17, 22),
  y = c(190, 23, 24, 23, 19, 34, 35, 36, 36, 34, 32, 38, 41, 42, 180)
)

# Fit the linear regression model to the dataset with outliers
```

```
model <- lm(y ~ x, data = outliers)

# Find Cook's distance for each observation in the dataset
cooksD <- cooks.distance(model)

# Plot Cook's Distance
n <- nrow(outliers)
plot(cooksD, main = "Cook's Distance for Influential Observations")
abline(h = 4 / n, lty = 2, col = "steelblue")  # Add cutoff line

# Identify influential points
influential_obs <- as.numeric(names(cooksD)[(cooksD > (4 / n))])

# Define new data frame with influential points removed
outliers_removed <- outliers[-influential_obs, ]
print(outliers_removed)

# Using mtcars dataset
model2 <- lm(mpg ~ cyl + wt, data = mtcars)
cooks.distance(model2)
cooks.distance(model2)[which.max(cooks.distance(model2))]
plot(model2, which = 4)
plot(cooks.distance(model2), type = "b", pch = 18, col = "red")

# Calculate cutoff value
N <- 32
k <- 2
cutoff <- 4 / (N - k - 1)
abline(h = cutoff, lty = 2)
```

Output:

Dataset(mt cars)-

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2

Model summary-

```
Call:
lm(formula = mpg ~ disp + hp, data = mtcars)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7945 -2.3036 -0.8246  1.8582  6.9363

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 30.735904   1.331566  23.083  < 2e-16 ***
disp        -0.030346   0.007405  -4.098 0.000306 ***
hp          -0.024840   0.013385  -1.856 0.073679 .
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.127 on 29 degrees of freedom
Multiple R-squared:  0.7482,  Adjusted R-squared:  0.7309
F-statistic: 43.09 on 2 and 29 DF,  p-value: 2.062e-09
```

Leverage Stats

	hatvalues(model)
Mazda RX4	0.04235795
Mazda RX4 Wag	0.04235795
Datsun 710	0.06287776
Hornet 4 Drive	0.07614472
Hornet Sportabout	0.08097817
Valiant	0.05945972
Duster 360	0.09828955
Merc 240D	0.08816960
Merc 230	0.05102253
Merc 280	0.03990060
Merc 280C	0.03990060
Merc 450SE	0.03890159
Merc 450SL	0.03890159
Merc 450SLC	0.03890159
Cadillac Fleetwood	0.19443875
Lincoln Continental	0.16042361
Chrysler Imperial	0.12447530
Fiat 128	0.08346304
Honda Civic	0.09493784
Toyota Corolla	0.08732818
Toyota Corona	0.05697867
Dodge Challenger	0.06954069
AMC Javelin	0.05767659
Camaro Z28	0.10011654
Pontiac Firebird	0.12979822
Fiat X1-9	0.08334018
Porsche 914-2	0.05785170
Lotus Europa	0.08193899

```
 [1] 0.49663919 0.19443875 0.16042361 0.13831817 0.12979822 0.12608583
 [7] 0.12447530 0.10011654 0.09828955 0.09493784 0.08816960 0.08732818
[13] 0.08346304 0.08334018 0.08193899 0.08097817 0.07614472 0.06954069
[19] 0.06287776 0.05945972 0.05848459 0.05785170 0.05767659 0.05697867
[25] 0.05102253 0.04235795 0.04235795 0.03990060 0.03990060 0.03890159
[31] 0.03890159 0.03890159
```

Outliers-

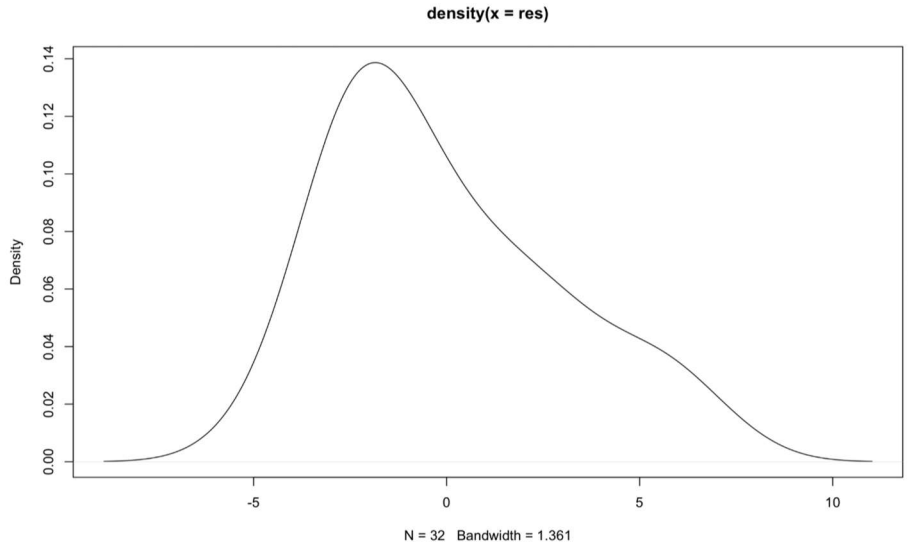
	x	y
2	2	23
3	2	24
4	3	23
5	4	19
6	5	34
7	7	35
8	3	36
9	2	36
10	12	34
11	11	32
12	15	38
13	14	41
14	17	42

Cook's Distance-

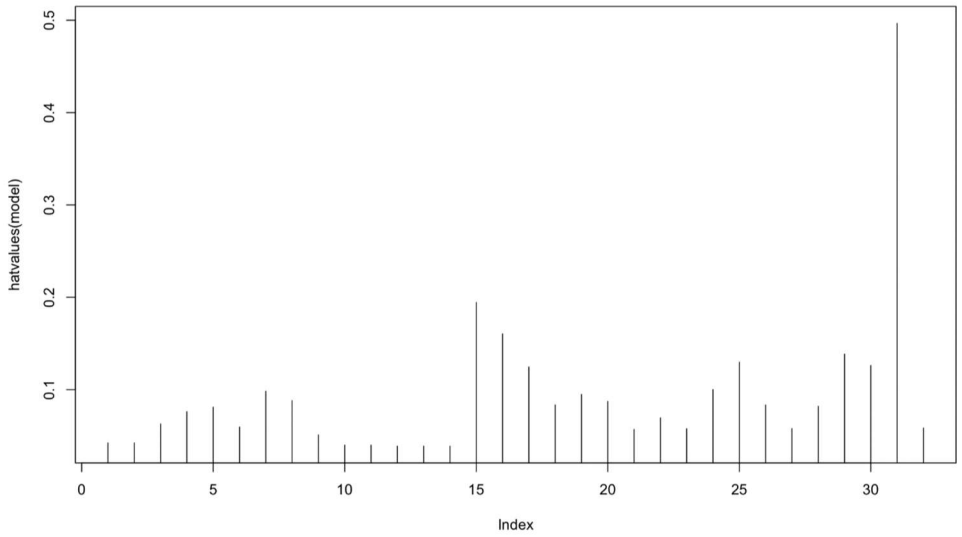
```
          Mazda RX4       Mazda RX4 Wag          Datsun 710
       0.0050772590        0.0004442585        0.0567764620
     Hornet 4 Drive   Hornet Sportabout             Valiant
       0.0018029260        0.0235271472        0.0050205614
        Duster 360           Merc 240D            Merc 230
       0.0178733213        0.0091033181        0.0065061176
         Merc 280           Merc 280C           Merc 450SE
       0.0004643600        0.0075293380        0.0116847953
        Merc 450SL          Merc 450SLC   Cadillac Fleetwood
       0.0102875723        0.0005228914        0.0035498738
Lincoln Continental   Chrysler Imperial            Fiat 128
       0.0001501537        0.3189363624        0.1592990291
         Honda Civic      Toyota Corolla       Toyota Corona
       0.0276449872        0.2233281268        0.0913548207
   Dodge Challenger         AMC Javelin          Camaro Z28
       0.0040263378        0.0120218543        0.0165559199
   Pontiac Firebird           Fiat X1-9       Porsche 914-2
       0.0569730451        0.0001790454        0.0033281614
        Lotus Europa     Ford Pantera L         Ferrari Dino
       0.0216355209        0.0237336584        0.0105550987
      Maserati Bora          Volvo 142E
       0.0072685192        0.0727399065
```

```
Chrysler Imperial
        0.3189364
```

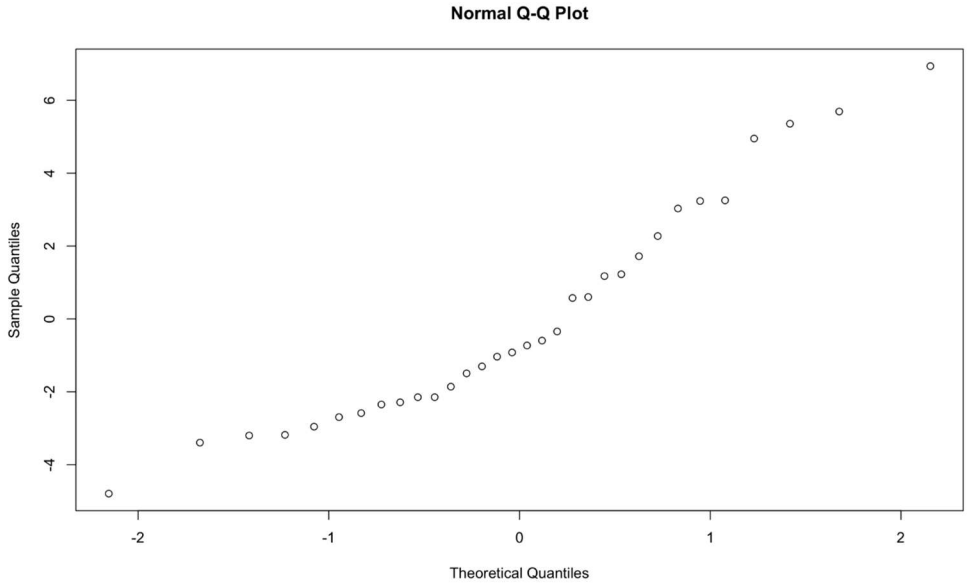
Residual Density Graph-



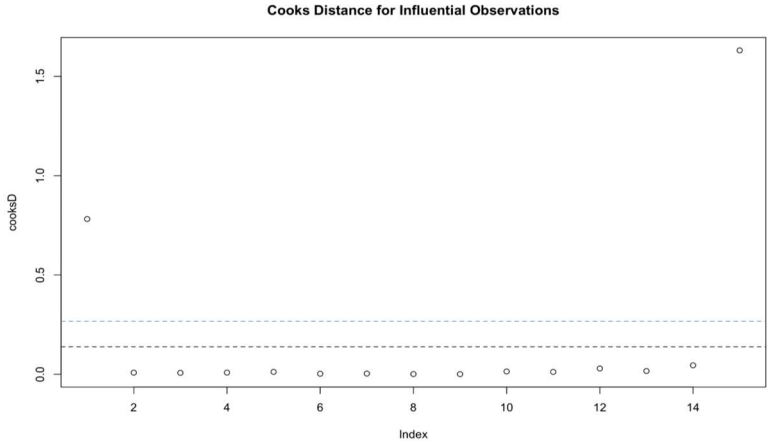
H-Values plot-



Normal Q-Q plot-



Cook's Distance plot-



VIVA QUESTIONS-

Q1.What are outliers in regression analysis?

Answer: Outliers are data points that significantly deviate from the overall trend of the dataset. They can impact the regression model's performance.

Q2.How do you detect outliers in a scatterplot?

Answer: Look for data points that fall far away from the main cluster in the scatterplot. High residuals indicate potential outliers.

Q3.What is leverage in regression?

Answer: Leverage measures how much an observation's x-value deviates from the mean of x. High leverage points can influence the regression line.

Q4.How do you calculate Cook's distance?

Answer: Cook's distance quantifies the influence of each observation on the regression model. It considers both leverage and residuals.

Q5.Why are influential data points important?

Answer: Influential data points can significantly impact the regression results. Detecting them helps ensure robust model estimation.

EXPERIMENT NO 7

Aim-Write a program to implement Time series Analysis using R.

Theory:

Understanding Time Series Analysis - Time series analysis can be used to forecast future events based on known past data. Examples of time series data include the number of users signed up for a service every day, the daily closing price of a company stock, or the number of page views each minute.

Implementing Time Series Analysis in R - Let's now look at how we can implement time series analysis in R. For this tutorial, we will use the Air Passengers dataset, which is the monthly totals of international airline passengers from 1949 to 1960.

Step 1: Loading the Dataset

Load the dataset data("AirPassengers")

Print the first few rows print(head(AirPassengers))

Step 2: Checking the Structure of the Dataset

Check the structure- str(AirPassengers)
The str() function will reveal that AirPassengers is a 'ts' (Time Series) object. The start attribute shows the first year and month of the data, and the end attribute shows the last year and month.

Step 3: Plotting the Dataset

Plot the dataset-plot(AirPassengers)
This will create a time series plot. From this plot, we can observe a trend in the data that suggests that the number of air passengers was increasing over time.

Step 4: Decomposing the Time Series Data

Decompose the time series data decomposed <- decompose(AirPassengers)

Plot the decomposed data- plot(decomposed)-The decompose() function breaks down the time series data into three components: trend, sea- sonal, and random.

Step 5: Forecasting Future Data

To forecast future data, we can use the forecast() function from the 'forecast' package in R. If this package is not yet installed, you can install it using the install.packages() function.

Install the 'forecast' package install.packages('forecast')

Load the 'forecast' package library(forecast)
Forecast future data- forecast_data <- forecast(AirPassengers, h = 24)

Plot the forecasted -dataplot(forecast_data),The h parameter determines the number of periods for forecasting. In this case, we are forecast- ing the number of air passengers for the next 24 months.

River Nile annual streamflow data -

In this exercise, you will plot the River Nile annual stream- flow data using the plot() function. For time series data objects such as Nile, a Time index for the horizontal axis is typically included. From the previous exercise, you know that this data span.from 1871 to 1970, and horizontal tick marks are label not very informative. Since these data are annual measurements, you should use the label,Additionally, it helps to have an informative title, which can be set using the argument main.

Source Code:

R-script

Air Passengers dataset

 # Load the dataset data("AirPassengers")

 # Print the first few rows

print(head(AirPassengers))

 # Check the structure

str(AirPassengers)

 # Plot the dataset

plot(AirPassengers)

 # Decompose the time series data decomposed

<- decompose(AirPassengers)

 # Plot the decomposed data plot(decomposed)

 # Load the 'forecast' package

library(forecast)

 # Forecast future data
forecast_data <- forecast(AirPassengers, h = 24)
 # Plot the forecasted data

plot(forecast_data)

print(Nile)
length(Nile)
 # Display the first 10 elements of the Nile dataset head(Nile, n = 10)
 # Display the last 12 elements of the Nile dataset tail(Nile, n = 12)

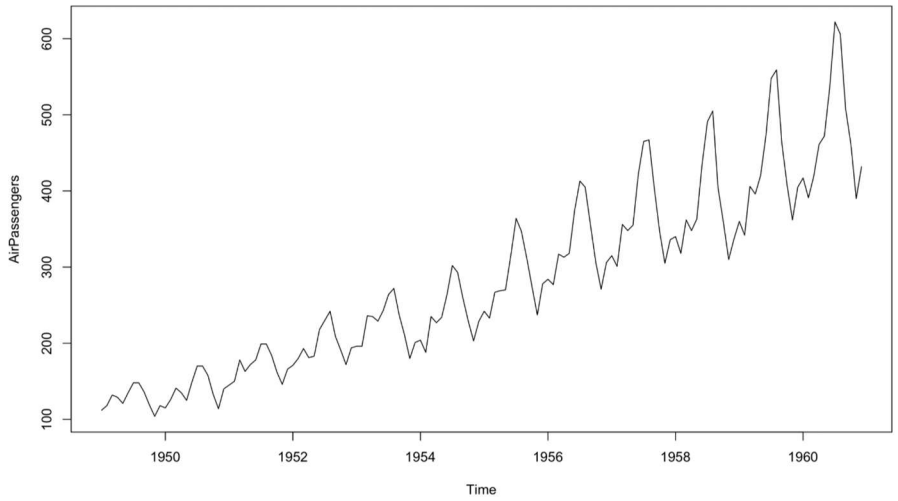
#plot(Nile, col = "blue") plot(Nile,col = "blue", xlab = "Year", ylab = "River Volume (1e9 m^{3})")

Output:

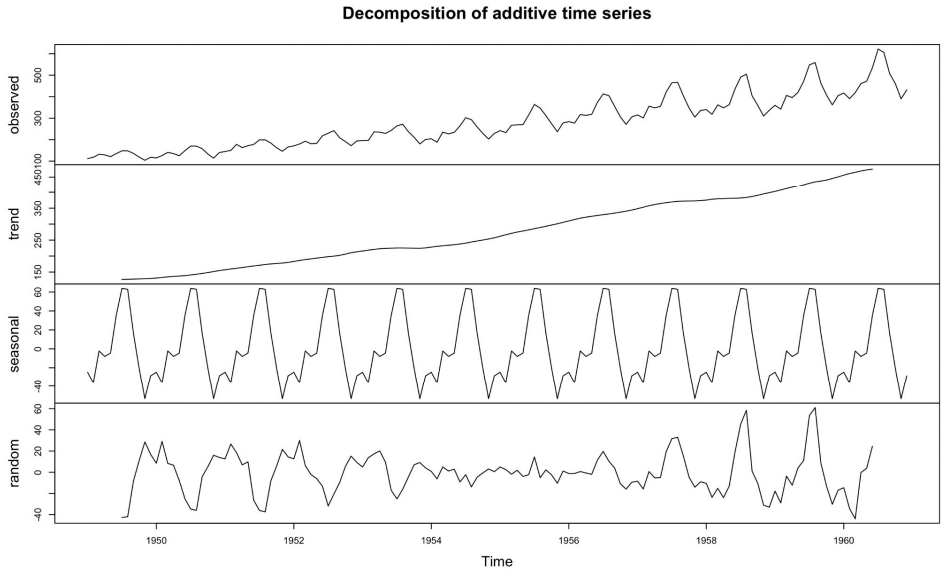
Air passengers data head and structure-

```
> print(head(AirPassengers))
[1] 112 118 132 129 121 135
> str(AirPassengers)
 Time-Series [1:144] from 1949 to 1961: 112 118 132 129 121 135 148 148 136
119 ...
```

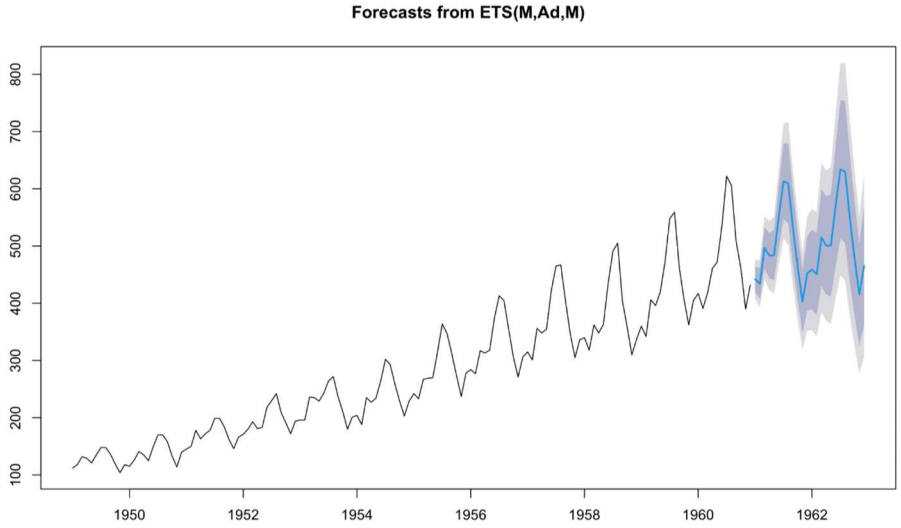
DATASET PLOT-



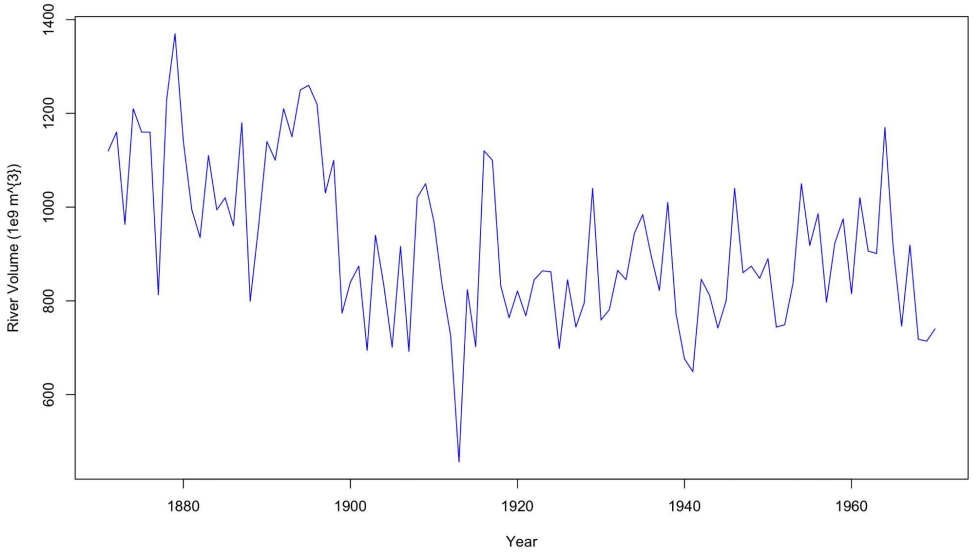
DECOMPOSED PLOT-



FORECAST PLOT-



RIVER NILE COLUMN PLOT-



VIVA QUESTIONS-

Q1.What is time series analysis, and how is it different from cross-sectional data analysis?

Answer: Time series analysis deals with data collected over time intervals (e.g., daily, monthly, yearly), where observations are dependent on previous values. Cross-sectional data analysis, on the other hand, examines data at a single point in time, without considering temporal relationships.

Q2.How do you load time series data into R?

Answer: You can load time series data into R using functions like read.csv(), read.table(), or specialized time series functions like ts() or xts().

Q3.What is the difference between trend, seasonality, and noise in a time series?

Answer:Trend: The long-term movement or direction in a time series. It can be upward, downward, or flat.

Seasonality: Regular patterns that repeat at fixed intervals (e.g., daily, weekly, yearly).

Noise (Random Fluctuations): Irregular variations that cannot be attributed to trend or seasonality.

Q4.How do you check for stationarity in a time series?

To check stationarity:

- Plot the time series data and look for trends or seasonality.
- Perform the Augmented Dickey-Fuller (ADF) test to assess stationarity statistically.

Q5.What are some common time series models in R?

- ARIMA (AutoRegressive Integrated Moving Average): Combines autoregressive (AR) and moving average (MA) components.
- Exponential Smoothing (ETS): Models with exponential decay of past observations.
- Seasonal Decomposition of Time Series (STL): Separates trend, seasonality, and remainder components.

EXPERIMENT NO 8

Aim-Write a program to implement linear regression using R .

Theory:

Linear regression is a statistical method used to model the relationship between a dependent variable and one or more independent variables. The main idea is to find the best-fitting straight line (or hyperplane in higher dimensions) that describes the relationship between the variables.

Here's a brief overview of linear regression:

Dependent Variable (Response Variable): This is the variable we want to predict or explain.

Independent Variables (Predictors): These are the variables used to predict or explain the dependent variable.

Linear Relationship: Linear regression assumes a linear relationship between the independent variables and the dependent variable. It means that the change in the dependent variable is proportional to the change in the independent vari- able(s).

Simple Linear Regression: In simple linear regression, there's only one inde- pendent variable. The relationship between the independent and dependent vari- ables is described by a straight line.

Multiple Linear Regression: In multiple linear regression, there are multiple in- dependent variables. The relationship is described by a hyperplane in higher di- mensions.

Source Code:

R-script

 # Step 1: Generate synthetic data
set.seed(123) # For reproducibility n
<- 100 # Number of observations x
<- rnorm(n) # Independent variable
y <- 2*x + 3 + rnorm(n) # Dependent variable (with noise)

 # Combine data into a data frame -data <- data.frame(x, y)

 # Step 2: Fit the linear regression model

lm_model <- lm(y ~ x, data = data) # Step

3: View summary of the model

summary(lm_model)

 # Step 4: Make predictions
 # For demonstration purpose, let's make predictions on the same data predictions <- predict(lm_model)

 # Step 5: Evaluate the model
 # For simplicity, let's just print the mean squared error mse <-

mean((predictions - data$y)^2) print(paste("Mean Squared Error:", mse))

OUTPUT-

Model summary:

```
> summary(lm_model)

Call:
lm(formula = y ~ x, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-1.9073 -0.6835 -0.0875  0.5806  3.2904

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.89720    0.09755   29.70   <2e-16 ***
x            1.94753    0.10688   18.22   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9707 on 98 degrees of freedom
Multiple R-squared:  0.7721,  Adjusted R-squared:  0.7698
F-statistic:   332 on 1 and 98 DF,  p-value: < 2.2e-16
```

Mean squared Error-

```
> mse <- mean((predictions - data$y)^2)
> print(paste("Mean Squared Error:", mse))
[1] "Mean Squared Error: 0.923441292618146"
```

VIVA QUESTIONS-

Q1.What is linear regression?

Answer-Linear regression is a statistical method used to model the relationship between a dependent variable (response) and one or more independent variables (predictors). It assumes a linear relationship, aiming to find the best-fitting straight line (regression line) that minimizes the sum of squared differences between observed and predicted values.

Q2.What are the steps involved in performing linear regression?

Answer-

The steps include:

- **Data Collection:** Gather relevant data on the dependent and independent variables.
- **Data Preprocessing:** Clean, transform, and handle missing values.
- **Model Selection:** Choose between simple linear regression (one predictor) or multiple linear regression (multiple predictors).
- **Model Fitting:** Estimate coefficients using least squares optimization.
- **Assumptions Check:** Validate assumptions (linearity, independence, homoscedasticity, normality).
- **Interpretation:** Interpret coefficients and assess model performance. ○ **Prediction:** Use the model for predictions.

Q3.How do you interpret the coefficients in a linear regression model?

Answer-

Coefficients represent the change in the dependent variable associated with a one-unit change in the corresponding predictor.

For example:

- **A positive coefficient means an increase in the dependent variable when the predictor increases.**
- **A negative coefficient means a decrease in the dependent variable when the predictor increases.**
- **The intercept represents the value of the dependent variable when all predictors are zero.**

Q4.What are some assumptions of linear regression?

Answer-

- **Linearity:** The relationship between predictors and the response is linear.
- **Independence:** Residuals are independent of each other**.**
- **Homoscedasticity:** Residuals have constant variance**.**
- **Normality:** Residuals follow a normal distribution.
- **No Multicollinearity:** Predictors are not highly correlated**.**

Q5.How do you assess the goodness of fit of a linear regression model?

Answer-

Common methods include:

- **R-squared (Coefficient of Determination):** Measures the proportion of variance explained by the model.
- **Adjusted R-squared:** Adjusts for the number of predictors.
- **Residual Analysis:** Check residuals for patterns (should be random).
- **F-test:** Compare the model to a null model.
- **AIC/BIC:** Evaluate model complexity.

EXPERIMENT NO 9

Aim-Write a program to find the Eigenvalues and eigenvectors in R.

Theory:

Eigenvalues - Eigenvalues are scalars associated with a square matrix.

For a given square matrix A, an eigenvalue is a scalar such that there exists a non-zero,vector v (the eigenvector) satisfying the equation Av= v.

Geometrically, multiplying a vector by a matrix transforms the vector, and an eigenvector remains in the same direction after the transformation, only scaled by its corresponding eigenvalue.

Eigenvectors - Eigenvectors are non-zero vectors corresponding to eigenvalues. An eigenvector v of a matrix A is a vector that remains in the same direction after multiplication by A. Eigenvectors are not unique; any scalar multiple of an eigenvector is also an eigenvector. The set of all eigenvectors corresponding to a particular eigenvalue forms an eigenspace.

Applications Eigenvalues and eigenvectors are extensively used in solving systems of differential equations, stability analysis of dynamical systems, and analyzing linear transformations in computer graphics and image processing.

In data analysis, they are used in principal component analysis (PCA) for dimensionality reduction and feature extraction.

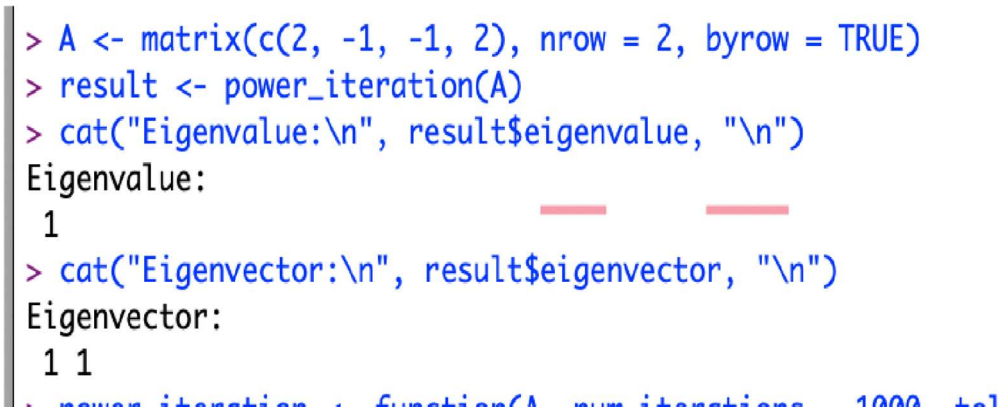
Calculation Eigenvalues and eigenvectors can be calculated using various methods, including power iteration, QR algorithm, characteristic polynomial method, and singular value decomposition (SVD). The choice of method depends on the size and properties of the matrix.

Source Code:

```
power_iteration <- function(A, num_iterations = 1000, tol = 1e-6) {
  n <- nrow(A)
  x <- rep(1, n)  # Initial guess for eigenvector
  lambda <- 0     # Initial guess for eigenvalue
```

```
  for (i in 1:num_iterations) {
    x_old <- x
    x <- A %*% x
    lambda_old <- lambda
    lambda <- max(x)
    x <- x / lambda
    # Check for convergence
    if (abs(lambda - lambda_old) < tol) {
      break
    }
  }
  return(list(eigenvalue = lambda, eigenvector = x))
}
# Example usage
A <- matrix(c(2, -1, -1, 2), nrow = 2, byrow = TRUE)
result <- power_iteration(A)
cat("Eigenvalue:\n", result$eigenvalue, "\n")
```

OUTPUT-



VIVA QUESTIONS-

Q.1. What are eigenvalues and eigenvectors?

Answer-

- Eigenvalues are scalar values associated with eigenvectors in linear transformations. The term "eigen" originates from German, meaning
  "characteristic." Eigenvalues indicate the factor by which eigenvectors stretch (or reverse direction if negative) under the transformation.
- Eigenvectors are non-zero vectors that, when multiplied by a square matrix, yield a scalar multiple of the vector. An eigenvector for matrix A satisfies the equation: $Av = \lambda v$, where λ is the eigenvalue.

Q.2. How do eigenvalues and eigenvectors relate to matrix operations?

Answer-

- **Eigenvalues and eigenvectors play a crucial role in matrix operations:**
  - Diagonalization: Diagonalizing a matrix involves expressing it as a product of three matrices: P (containing eigenvectors), D (a diagonal matrix with eigenvalues), and P^{-1} (the inverse of P).
  - Matrix Powers: Matrix powers (A^n) can be computed using eigenvalues and eigenvectors.
  - Matrix Exponential: The exponential of a matrix (e^A) can be expressed using its eigenvalues and eigenvectors.

Q3.Why is finding eigenvalues and eigenvectors important in data analysis?

Answer-

- **Eigenvalues and eigenvectors are fundamental in data science:**
  - Feature Extraction: Principal Component Analysis (PCA) uses eigenvectors to extract essential features from high-dimensional data.
  - Dimensionality Reduction: By selecting top eigenvectors, we reduce data dimensionality while preserving information.
  - Understanding Variability: Eigenvectors reveal the inherent structure and variability within datasets.
  - Machine Learning: Eigenvalues guide model selection and regularization. **Q.4. Can**

a matrix have multiple sets of eigenvalues and eigenvectors?

Answer-Yes,a matrix can have multiple sets of eigenvalues and eigenvectors

Q.5. What is the relationship between eigenvalues, eigenvectors, and diagonalization of a matrix?

Answer-Diagonalization connects eigenvalues, eigenvectors, and matrices:

- A matrix is diagonalizable if it has a complete set of linearly independent eigenvectors.
- Diagonalization transforms the original matrix into a diagonal matrix using eigenvectors and eigenvalues.

EXPERIMENT NO 10

Aim-Write a program to implement the associative, commutative and distributive property in a matrix in R

Theory: Matrices in R follow specific rules for addition and scalar multiplication. These rules are essential for performing accurate matrix calculations.

Here, we'll explore three fundamental properties:

Associative Property (Addition): (A + B) + C = A + (B + C). This means the order of adding three matrices doesn't affect the final result**.**

Commutative Property (Addition): A + B = B + A. The order of adding two matrices doesn't change the sum. (Note: This applies to addition only, not multiplication).

Distributive Property (Scalar Multiplication): k(A + B) = kA + kB. Multiplying a scalar (number) by the sum of two matrices is the same as multiplying the scalar by each matrix individually and then adding the products .

Source Code:

R-script

```
# Create sample matrices A <-
 matrix(c(1, 2, 3, 4), nrow = 2)
B <- matrix(c(5, 6, 7, 8), nrow = 2)
C <- matrix(c(9, 10, 11, 12), nrow = 2) k <- 2

# Verify properties

cat("Associative (Addition):\n") print(A + (B + C)) print(A
+ B + C)

cat("\nCommutative (Addition):\n") print(A + B) print(B
+ A)

cat("\nDistributive (Scalar Multiplication):\n") print(k * (A + B)) print(k
* A + k * B)
```

OUTPUT-

```
> cat("Associative (Addition):\n")
Associative (Addition):
> print(A + (B + C))
     [,1] [,2]
[1,]   15   21
[2,]   18   24
> print(A + B + C)
     [,1] [,2]
[1,]   15   21
[2,]   18   24
> cat("\nCommutative (Addition):\n")

Commutative (Addition):
> print(A + B)
     [,1] [,2]
[1,]    6   10
[2,]    8   12
> print(B + A)
     [,1] [,2]
[1,]    6   10
[2,]    8   12
> cat("\nDistributive (Scalar Multiplication):\n")

Distributive (Scalar Multiplication):
> print(k * (A + B))
     [,1] [,2]
[1,]   12   20
[2,]   16   24
```

```
> print(k * A + k * B)
     [,1] [,2]
[1,]   12   20
[2,]   16   24
```

VIVA QUESTIONS-

Q.1. Explain the associative property of addition for matrices. Can this property be applied to matrix multiplication as well?

Answer-The associative property states that the grouping of matrix additions does not affect the result.

- For any matrices A, B, and C of compatible dimensions:
- [(A + B) + C = A + (B + C)]
- This property holds true for matrix addition because adding matrices is analogous to adding corresponding elements.

Q.2. Why is the commutative property only valid for matrix addition and not multiplication?

Answer-

- **Matrix Addition:**
  - ○ The commutative property holds: [A + B = B + A] ○ The order of addition does not matter.
- **Matrix Multiplication:**
  - ○ The commutative property does not hold: [AB \neq BA]
  - ○ The order of multiplication matters; matrix multiplication is not commutative.

Q.3. Demonstrate the distributive property for scalar multiplication with a different scalar value.

For any scalar k and matrices A and B: ○ [k(A + B) = kA + kB]

○ Distributing the scalar k to each element of the sum A + B.

Q.4. How do these properties simplify matrix calculations?

Answer-

- These properties simplify matrix calculations by allowing us to rearrange expressions without changing the result.
- Associative property helps group additions efficiently.
- Distributive property simplifies scalar multiplication.

Q.5. In what scenarios might these properties not hold true (e.g., non-square matrices, singular matrices)?

Answer-

- **Non-Square Matrices:**
    - ○ Matrix addition and multiplication require compatible dimensions (e.g., m × n and n × p).
    - ○ Non-square matrices may not satisfy these properties due to dimension constraints.
- **Singular Matrices:**
    - ○ Singular matrices (with determinant 0) may not have inverses.
    - ○ Inverse matrices are essential for some properties (e.g., commutativity in multiplication).