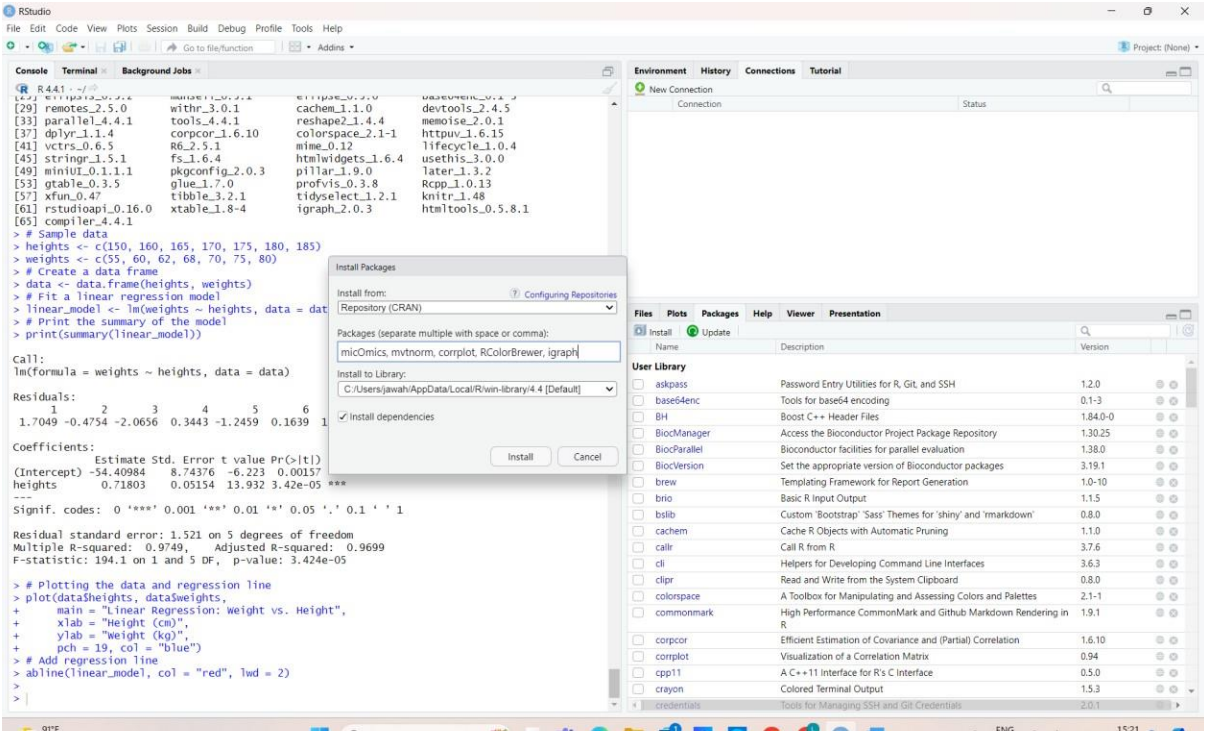
Ex 7 IMPLEMENT LINEAR AND LOGISTIC REGRESSION

AIM:

To implement linear and logistic regression using R Programming.

PROCEDURE:

1. Install R for windows.
2. Install R Studio.
3. Open R Studio and install packages



RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Project: (None)
Console Terminal Background Jobs
R 4.4.1 · ~/
Environment History Connections Tutorial
New Connection
Connection Status
Files Plots Packages Help Viewer Presentation
Install Update

```
[29] remotes_2.5.0       withr_3.0.1       cachem_1.1.0        devtools_2.4.5
[33] parallel_4.4.1      tools_4.4.1       reshape2_1.4.4      memoise_2.0.1
[37] dplyr_1.1.4         corpcor_1.6.10    colorspace_2.1-1    httpuv_1.6.15
[41] vctrs_0.6.5         R6_2.5.1          mime_0.12           lifecycle_1.0.4
[45] stringr_1.5.1       fs_1.6.4          htmlwidgets_1.6.4   usethis_3.0.0
[49] miniUI_0.1.1.1      pkgconfig_2.0.3   pillar_1.9.0        later_1.3.2
[53] gtable_0.3.5        glue_1.7.0        profvis_0.3.8       Rcpp_1.0.13
[57] xfun_0.47           tibble_3.2.1      tidyselect_1.2.1    knitr_1.48
[61] rstudioapi_0.16.0   xtable_1.8-4      igraph_2.0.3        htmltools_0.5.8.1
[65] compiler_4.4.1
> # Sample data
> heights <- c(150, 160, 165, 170, 175, 180, 185)
> weights <- c(55, 60, 62, 68, 70, 75, 80)
> # Create a data frame
> data <- data.frame(heights, weights)
> # Fit a linear regression model
> linear_model <- lm(weights ~ heights, data = dat
> # Print the summary of the model
> print(summary(linear_model))

Call:
lm(formula = weights ~ heights, data = data)

Residuals:
      1       2       3       4       5       6
 1.7049 -0.4754 -2.0656  0.3443 -1.2459  0.1639  1

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -54.40984    8.74376  -6.223  0.00157
heights       0.71803    0.05154  13.932 3.42e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.521 on 5 degrees of freedom
Multiple R-squared:  0.9749,    Adjusted R-squared:  0.9699
F-statistic: 194.1 on 1 and 5 DF,  p-value: 3.424e-05

> # Plotting the data and regression line
> plot(data$heights, data$weights,
+      main = "Linear Regression: Weight vs. Height",
+      xlab = "Height (cm)",
+      ylab = "Weight (kg)",
+      pch = 19, col = "blue")
> # Add regression line
> abline(linear_model, col = "red", lwd = 2)
>
> |
```

Install Packages

Install from: Repository (CRAN)

Configuring Repositories

Packages (separate multiple with space or comma): micOmics, mvtnorm, corrplot, RColorBrewer, igraph

Install to Library: C:/Users/jawah/AppData/Local/R/win-library/4.4 [Default]

☑ Install dependencies

Install Cancel

	Name	Description	Version
	User Library		
□	askpass	Password Entry Utilities for R, Git, and SSH	1.2.0
□	base64enc	Tools for base64 encoding	0.1-3
□	BH	Boost C++ Header Files	1.84.0-0
□	BiocManager	Access the Bioconductor Project Package Repository	1.30.25
□	BiocParallel	Bioconductor facilities for parallel evaluation	1.38.0
□	BiocVersion	Set the appropriate version of Bioconductor packages	3.19.1
□	brew	Templating Framework for Report Generation	1.0-10
□	brio	Basic R Input Output	1.1.5
□	bslib	Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'	0.8.0
□	cachem	Cache R Objects with Automatic Pruning	1.1.0
□	callr	Call R from R	3.7.6
□	cli	Helpers for Developing Command Line Interfaces	3.6.3
□	clipr	Read and Write from the System Clipboard	0.8.0
□	colorspace	A Toolbox for Manipulating and Assessing Colors and Palettes	2.1-1
□	commonmark	High Performance CommonMark and Github Markdown Rendering in R	1.9.1
□	corpcor	Efficient Estimation of Covariance and (Partial) Correlation	1.6.10
□	corrplot	Visualization of a Correlation Matrix	0.94
□	cpp11	A C++11 Interface for R's C Interface	0.5.0
□	crayon	Colored Terminal Output	1.5.3
	credentials	Tools for Managing SSH and Git Credentials	2.0.1

Image 7.1

4. Write the program for Linear Regression

# Sample data
heights<- c(150, 160, 165, 170, 175, 180, 185)
weights <- c(55, 60, 62, 68, 70, 75, 80)
# Create a data frame data <- data.frame(heights, weights)
# Fit a linear regression model
linear_model <- lm(weights ~ heights, data = data)
# Print the summary of the model
print(summary(linear_model))
# Plotting the data and regression line
plot(data$heights, data$weights,
main = "Linear Regression: Weight vs. Height",
xlab = "Height (cm)",
ylab = "Weight (kg)",
pch = 19, col = "blue")
# Add regression line
abline(linear_model, col = "red", lwd = 2)

Output: Refer image 7.2

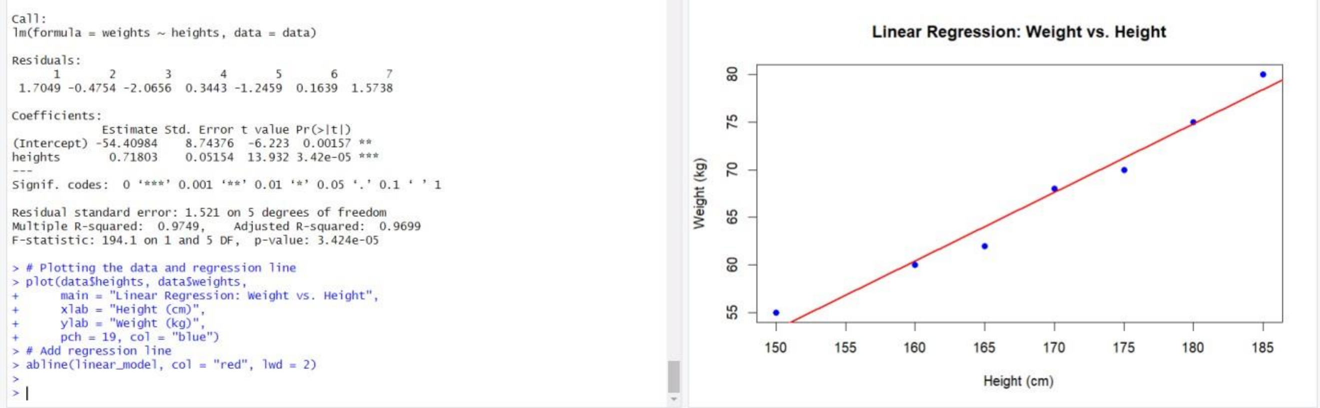
5. Write the program for Logistic Regression

```
# Load the dataset
data(mtcars)

# Convert 'am' to a factor (categorical variable)
mtcars$am <- factor(mtcars$am, levels = c(0, 1), labels = c("Automatic", "Manual"))

# Fit a logistic regression model
logistic_model <- glm(am ~ mpg, data = mtcars, family = binomial)

# Print the summary of the model
print(summary(logistic_model))

# Predict probabilities for the logistic model
predicted_probs <- predict(logistic_model, type = "response")

# Display the predicted probabilities
print(predicted_probs)

# Plotting the data and logistic regression curve
plot(mtcars$mpg, as.numeric(mtcars$am) - 1,
    main = "Logistic Regression: Transmission vs. MPG",
    xlab = "Miles Per Gallon (mpg)",
    ylab = "Probability of Manual Transmission",
    pch = 19, col = "blue")

# Add the logistic regression curve
curve(predict(logistic_model, newdata = data.frame(mpg = x), type = "response"),
     add = TRUE, col = "red", lwd = 2)
```

Output refer Image 7.2

Output:

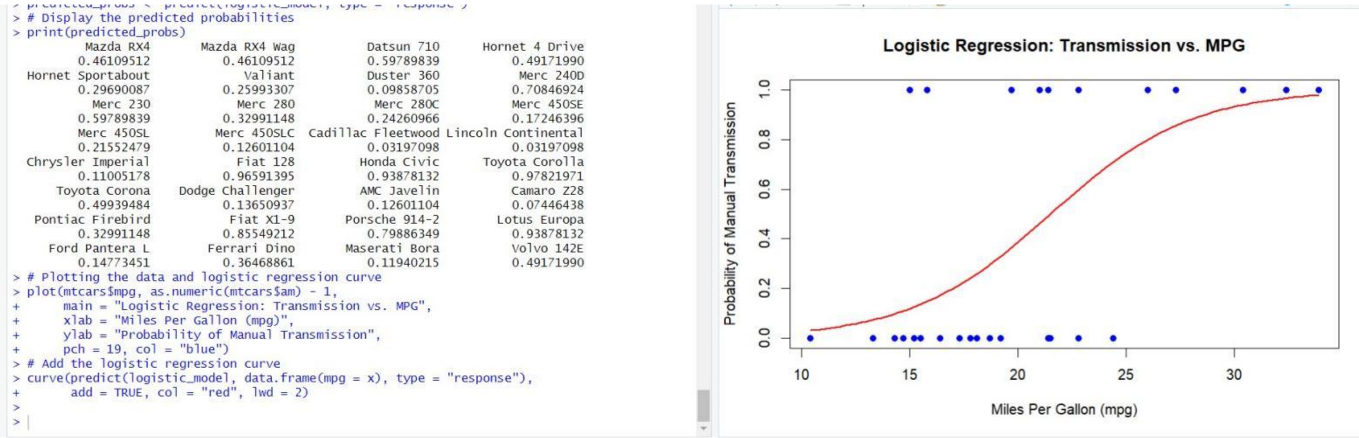
Output for Linear Regression



```
Call:
lm(formula = weights ~ heights, data = data)

Residuals:
      1       2       3       4       5       6       7 
 1.7049 -0.4754 -2.0656  0.3443 -1.2459  0.1639  1.5738 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -54.40984    8.74376  -6.223  0.00157 ** 
heights       0.71803    0.05154  13.932 3.42e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.521 on 5 degrees of freedom
Multiple R-squared:  0.9749,	Adjusted R-squared:  0.9699 
F-statistic: 194.1 on 1 and 5 DF,  p-value: 3.424e-05

> # Plotting the data and regression line
> plot(data$heights, data$weights,
+      main = "Linear Regression: Weight vs. Height",
+      xlab = "Height (cm)",
+      ylab = "Weight (kg)",
+      pch = 19, col = "blue")
> # Add regression line
> abline(linear_model, col = "red", lwd = 2)
> 
> |
```

Image 7.2

Output for Logistic Regression



```
> predicted_probs <- predict(logistic_model, type = "response")
> # Display the predicted probabilities
> print(predicted_probs)
          Mazda RX4       Mazda RX4 Wag          Datsun 710      Hornet 4 Drive 
         0.46109512          0.46109512          0.59789839          0.49171990 
  Hornet Sportabout             Valiant          Duster 360           Merc 240D 
         0.29690087          0.25993307          0.09858705          0.70846924 
           Merc 230            Merc 280           Merc 280C          Merc 450SE 
         0.59789839          0.32991148          0.24260966          0.17246396 
         Merc 450SL         Merc 450SLC  Cadillac Fleetwood Lincoln Continental 
         0.21552479          0.12601104          0.03197098          0.03197098 
  Chrysler Imperial            Fiat 128         Honda Civic      Toyota Corolla 
         0.11005178          0.96591395          0.93878132          0.97821971 
      Toyota Corona    Dodge Challenger         AMC Javelin          Camaro Z28 
         0.49939484          0.13650937          0.12601104          0.07446438 
   Pontiac Firebird           Fiat X1-9       Porsche 914-2        Lotus Europa 
         0.32991148          0.85549212          0.79886349          0.93878132 
     Ford Pantera L        Ferrari Dino       Maserati Bora          Volvo 142E 
         0.14773451          0.36468861          0.11940215          0.49171990 
> # Plotting the data and logistic regression curve
> plot(mtcars$mpg, as.numeric(mtcars$am) - 1,
+      main = "Logistic Regression: Transmission vs. MPG",
+      xlab = "Miles Per Gallon (mpg)",
+      ylab = "Probability of Manual Transmission",
+      pch = 19, col = "blue")
> # Add the logistic regression curve
> curve(predict(logistic_model, data.frame(mpg = x), type = "response"),
+       add = TRUE, col = "red", lwd = 2)
> 
> |
```

Logistic Regression: Transmission vs. MPG
Probability of Manual Transmission
0.0
0.2
0.4
0.6
0.8
1.0
10
15
20
25
30
Miles Per Gallon (mpg)

Image 7.2

Result :

Thus the linear and logistic regression are implemented successfully using R Programming.