Program-1.a) Commands for basic user input/output

b)Basic Data Types and Data Manipulation Functions

```
name <- readline(prompt = "Enter your name: ")

print(paste("Hello,", name))

x <- 5

y <- 3.14

text <- "Hello"

my_list <- list(1, 2, 3)

my_df <- data.frame(a = 1:3, b = 4:6)

print(x)

print(y)

print(text)

print(my_list)

print(my_df)
```

Output:

```
> name <- readline(prompt = "Enter your name: ")

Enter your name: SVDC

> print(paste("Hello,", name))

[1] "Hello, SVDC"

> x <- 5

> y <- 3.14

> text <- "Hello"

> my_list <- list(1, 2, 3)

> my_df <- data.frame(a = 1:3, b = 4:6)

> print(x)

[1] 5

> print(y)
```

```
[1] 3.14

> print(text)

[1] "Hello"

> print(my_list)

[[1]]

[1] 1

[[2]]

[1] 2

[[3]]

[1] 3

> print(my_df)

  a b

1 1 4

2 2 5

3 3 6
```

Program-2) Introduction to basic commands Continued: a) Conditional Statements b) Loops

```
x=7

if (x > 3) {

  print("x is greater than 3")

}

for (i in 1:5) {

  print(i)

}
```

Output:

```
> x=7

> if (x > 3) {
```

```
+   print("x is greater than 3")
+ }
[1] "x is greater than 3"
>
> for (i in 1:5) {
+   print(i)
+ }
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Program-3) **Data Manipulation Package installation and different operations using installed package**

```
install.packages("tidyverse")
library(tidyverse)
data <- tibble(A = c(1, 2, 3), B = c(4, 5, 6))
print(data)
```

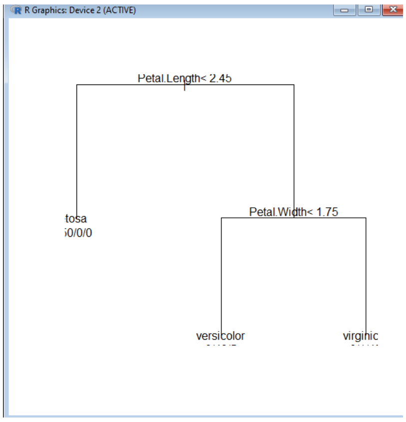
Output:

```
> data <- tibble(A = c(1, 2, 3), B = c(4, 5, 6))
> print(data)
# A tibble: 3 × 2
      A     B
  <dbl> <dbl>
1     1     4
2     2     5
3     3     6
```

Program-4) Standard Library function to plot the Graphs

```
library(rpart)

data(iris)

model <- rpart(Species ~ ., data = iris, method = "class")

print(model)

plot(model)

text(model, use.n=TRUE)
```

```
> library(rpart)

> data(iris)

> model <- rpart(Species ~ ., data = iris, method = "class")

> print(model)

n= 150

node), split, n, loss, yval, (yprob)

      * denotes terminal node

1) root 150 100 setosa (0.33333333 0.33333333 0.33333333)

  2) Petal.Length< 2.45 50   0 setosa (1.00000000 0.00000000 0.00000000) *

  3) Petal.Length>=2.45 100  50 versicolor (0.00000000 0.50000000 0.50000000)

    6) Petal.Width< 1.75 54   5 versicolor (0.00000000 0.90740741 0.09259259) *

    7) Petal.Width>=1.75 46   1 virginica (0.00000000 0.02173913 0.97826087) *

> plot(model)

> text(model, use.n=TRUE)
```



Program-5) Basic Data Exploration on any dataset available publicly

```
install.packages("dplyr")

library(dplyr)

data(iris)

head(iris)

iris %>%

  summarise(across(where(is.numeric), ~mean(., na.rm = TRUE)))
```

Output:

```
> data(iris)

> head(iris)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species

1        5.1        3.5         1.4        0.2  setosa

2        4.9        3.0         1.4        0.2  setosa

3        4.7        3.2         1.3        0.2  setosa

4        4.6        3.1         1.5        0.2  setosa

5        5.0        3.6         1.4        0.2  setosa

6        5.4        3.9         1.7        0.4  setosa

> iris %>%

+   summarise(across(where(is.numeric), ~mean(., na.rm = TRUE)))

  Sepal.Length Sepal.Width Petal.Length Petal.Width

1     5.843333    3.057333       3.758    1.199333
```

Program-6) Learning Algorithms-kNN Linear Regression

```
# Load built-in dataset

data(mtcars)

# Fit linear regression model

model <- lm(mpg ~ wt, data = mtcars)

# View model summary
```

```
summary(model)

# Predict mpg for new weight values

new_weights <- data.frame(wt = c(2.5, 3.0, 3.5))

predict(model, new_weights)
```

Output:

```
> # Load built-in dataset

> data(mtcars)

>

> # Fit linear regression model

> model <- lm(mpg ~ wt, data = mtcars)

>

> # View model summary

> summary(model)

Call:

lm(formula = mpg ~ wt, data = mtcars)

Residuals:

    Min      1Q  Median      3Q     Max

-4.5432 -2.3647 -0.1252  1.4096  6.8727

Coefficients:

            Estimate Std. Error t value Pr(>|t|)

(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***

wt          -5.3445     0.5591  -9.559 1.29e-10 ***

---

Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.046 on 30 degrees of freedom

Multiple R-squared:  0.7528,    Adjusted R-squared:  0.7446

F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```
>
> # Predict mpg for new weight values
> new_weights <- data.frame(wt = c(2.5, 3.0, 3.5))
> predict(model, new_weights)
       1        2        3
23.92395 21.25171 18.57948
```

Program-7) Unsupervised Algorithm- k-means

```
# Load dataset
data(iris)
# Select features
features <- iris[, 1:4]
# Apply k-means clustering
result <- kmeans(features, centers = 3)
# View clustering result
table(result$cluster)
# Add cluster info to iris
iris$Cluster <- result$cluster
# View first few rows
head(iris)
```

Output:

```
> # Load dataset
> data(iris)
> # Select features
> features <- iris[, 1:4]
> # Apply k-means clustering
> result <- kmeans(features, centers = 3)
> # View clustering result
```

```
> table(result$cluster)

 1  2  3

50 38 62

> # Add cluster info to iris

> iris$Cluster <- result$cluster

> # View first few rows

> head(iris)

  Sepal.Length Sepal.Width Petal.Length Petal.Width Species Cluster

1          5.1         3.5          1.4         0.2  setosa       1

2          4.9         3.0          1.4         0.2  setosa       1

3          4.7         3.2          1.3         0.2  setosa       1

4          4.6         3.1          1.5         0.2  setosa       1

5          5.0         3.6          1.4         0.2  setosa       1

6          5.4         3.9
```

Program-8) Implement Decision Tree and Support Vector Machine using Library Functions

```
# Install and load the rpart package (if not already installed)

install.packages("rpart")

library(rpart)

# Load iris data

data(iris)

# Create decision tree model

tree_model <- rpart(Species ~ ., data = iris, method = "class")

# Predict species for first 5 samples

predictions <- predict(tree_model, iris[1:5, ], type = "class")

# Show predictions

print(predictions)
```

Output:

```
> # Create decision tree model
> tree_model <- rpart(Species ~ ., data = iris, method = "class")
> # Predict species for first 5 samples
> predictions <- predict(tree_model, iris[1:5, ], type = "class")
> # Show predictions
> print(predictions)
     1      2      3      4      5
setosa setosa setosa setosa setosa
Levels: setosa versicolor virginica
```

Program-9)Linear Regression in R

```
data(mtcars)
model <- lm(mpg ~ wt, data = mtcars)
summary(model)
```

Output

```
> data(mtcars)
> model <- lm(mpg ~ wt, data = mtcars)
> summary(model)
Call:
lm(formula = mpg ~ wt, data = mtcars)
Residuals:
    Min      1Q  Median      3Q     Max
-4.5432 -2.3647 -0.1252  1.4096  6.8727
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
wt           -5.3445     0.5591  -9.559 1.29e-10 ***
```

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.046 on 30 degrees of freedom

Multiple R-squared: 0.7528,   Adjusted R-squared: 0.7446

F-statistic: 91.38 on 1 and 30 DF, p-value: 1.294e-10

Program-10)K-Means Clustering on the Iris Dataset

```
data(iris)

iris_data <- iris[, 1:4]  # Remove species

km <- kmeans(iris_data, centers = 3)

table(iris$Species, km$cluster)
```

Output

```
> data(iris)

> iris_data <- iris[, 1:4]  # Remove species

> km <- kmeans(iris_data, centers = 3)

> table(iris$Species, km$cluster)

                  1   2   3

  setosa     50  0  0

  versicolor  0 48  2

  virginica   0 14 36
```