TITLE:	DATE: PAGE NO:

Calculator Application:

a) Using with and without R objects on console.
b) Using mathematical functions on console.
c) Write an R script, to create R objects for calculator application and save in a specified location in disk.
d) R Packages: tidyr, ggplot2, ggraph, glue. shiny.

a) Program:

```
# Without R objects 10
+ 5
20 - 7
8 * 3
15 / 3
```

Output:

```
[1] 15
[1] 13
[1] 24
[1] 5
```

```
# Using R objects
x <- 25
y <- 5

sum <- x + y
diff <- x - y
prod <- x * y
div <- x / y

print(sum)
print(diff)
print(prod)
print(div)
```

Output:

```
[1] 30
[1] 20
[1] 125
[1] 5
```

b) Program:

```
sqrt(16)       # Square root
exp(2)         # Exponential
log(10)        # Natural log
log10(1000)  # Log base 10
sin(pi/2)      # Trigonometric function
abs(-45)       # Absolute value
```

Output:

```
[1] 4
[1] 7.389056
```

TITLE:	DATE: PAGE NO:

```
[1] 2.302585
[1] 3
[1] 1
[1] 45
```

c) Program:

```
# Calculator script calculator
<- function(a, b) { result <-
list(
    Addition = a + b,
    Subtraction = a - b,
    Multiplication = a * b,
    Division = a / b
  )
  return(result)
}
res <- calculator(20, 4)
print(res)
saveRDS(res, file = "calculator_output.rds")
read_res <- readRDS("calculator_output.rds")
print(read_res)
```

Output:

```
$Addition
[1] 24

$Subtraction
[1] 16

$Multiplication
[1] 80

$Division
[1] 5

$Addition
[1] 24

$Subtraction
[1] 16

$Multiplication
[1] 80

$Division
[1] 5
```

d) Program:

```
# Load essential packages
library(tidyr)       # Data cleaning & reshaping
library(ggplot2)  # Data visualization
library(ggraph)     # Graph visualization
library(glue)        # String interpolation
library(shiny)       # Web applications in R
```

TITLE:	DATE: PAGE NO:

Descriptive Statistics
a) Write an R script to find basic descriptive statistics using summary().
b) Write an R script to find subset of dataset by using subset().

a) Program:

```
data(iris)
summary(iris)
summary(iris$Sepal.Length)
```
Output:
```
 Sepal.Length     Sepal.Width      Petal.Length     Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa          50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50 Median
 :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean  :5.843  Mean  :3.057  Mean  :3.758  Mean  :1.199 3rd
 Qu.:6.400   3rd  Qu.:3.300   3rd  Qu.:5.100   3rd  Qu.:1.800  Max.
 :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

b) Program:
```
subset_data <- subset(iris, Sepal.Length > 5)
head(subset_data)
```
Output:
```
 Sepal.Length Sepal.Width Petal.Length Petal.Width Species
 1          5.1         3.5           1.4          0.2    setos
                                                           a
 2          4.9         3.0           1.4          0.2    setos
                                                           a
 3          4.7         3.2           1.3          0.2    setos
                                                           a
 5          5.0         3.6           1.4          0.2    setos
                                                           a
 6          5.4         3.9           1.7          0.4    setos
                                                           a
 7          4.6         3.4           1.4          0.3    setos
                                                           a
```

```
# Subset selecting only specific columns
subset_cols <- subset(iris, select = c(Sepal.Length, Species))
head(subset_cols)
```
Output:
```
  Sepal.Length  Specie
```

TITLE:	DATE: PAGE NO:

```
 s
1      5.1    setos
                a
2      4.9    setos
                a
3      4.7    setos
                a
4      4.6    setos
                a
5      5.0    setos
                a
6      5.4    setos
                a
```

# Subset with condition and selected columns
subset_cond <- subset(iris, Species == "setosa", select = c(Sepal.Length, Sepal.Width)) head(subset_cond)

Output:

```
 Sepal.Length  Sepal.Width
1       5.1        3.5
2       4.9        3.0
3       4.7        3.2
4       4.6        3.1
5       5.0        3.6
6       5.4        3.9
```

TITLE:	DATE: PAGE NO:

Reading and Writing Different Types of Datasets

a) Reading different types of datasets (.txt, .csv) from web and disk and writing in file in specific disk location.
b) Reading Excel dataset in R.
c) Reading XML dataset in R.

a) Program:

```
# Writing a sample .txt file
write.table(iris[1:5, ], file = "sample.txt", sep = "\t", row.names = FALSE)

# Reading .txt file
txt_data <- read.table("sample.txt", header = TRUE, sep = "\t")
print(txt_data)
```

Output:

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
 1      5.1        3.5         1.4        0.2     setos
                                                     a
 2      4.9        3.0         1.4        0.2     setos
                                                     a
 3      4.7        3.2         1.3        0.2     setos
                                                     a
 4      4.6        3.1         1.5        0.2     setos
                                                     a
 5      5.0        3.6         1.4        0.2     setos
                                                     a
```

```
# Writing a sample .csv file
write.csv(iris[1:5, ], file = "sample.csv", row.names = FALSE)

# Reading .csv file
csv_data <- read.csv("sample.csv")
print(csv_data)
```

Output:

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
 1      5.1        3.5         1.4        0.2     setos
                                                     a
 2      4.9        3.0         1.4        0.2     setos
                                                     a
 3      4.7        3.2         1.3        0.2     setos
                                                     a
 4      4.6        3.1         1.5        0.2     setos
                                                     a
 5      5.0        3.6         1.4        0.2     setos
                                                     a
```

b) Program:

```
# Install package if not already installed #
install.packages("readxl")
```

l

TITLE:	DATE: PAGE NO:

```
ibrary(readxl)

# Assume an Excel file named "sample.xlsx" with iris data #
For demo, we first create one:
# (Requires openxlsx package to write) #
install.packages("openxlsx")
library(openxlsx)
write.xlsx(iris[1:5, ], "sample.xlsx")

# Read the Excel file
excel_data <- read_excel("sample.xlsx")
```

```
print(excel_data)
```

Output:

```
# First 5 rows of iris data #
A tibble: 5 × 5
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
        <dbl>       <dbl>        <dbl>       <dbl> <chr>
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
```

c) Program:

```
# Install if not installed
# install.packages("xml2")
library(xml2)

# Create sample XML content
xml_content <- '<root>
  <flower>
    <Sepal.Length>5.1</Sepal.Length>
    <Sepal.Width>3.5</Sepal.Width>
    <Species>setosa</Species>
  </flower>
  <flower>
    <Sepal.Length>4.9</Sepal.Length>
    <Sepal.Width>3.0</Sepal.Width>
    <Species>setosa</Species>
  </flower>
</root>'
# Write XML file
writeLines(xml_content, "sample.xml")

# Read XML file
xml_file <- read_xml("sample.xml")
print(xml_file)
# Extract data
flowers <- xml_find_all(xml_file, "//flower")
for (f in flowers) {
  cat("Sepal.Length:", xml_text(xml_find_first(f, "Sepal.Length")),
      " Sepal.Width:", xml_text(xml_find_first(f, "Sepal.Width")),
      " Species:", xml_text(xml_find_first(f, "Species")), "\n")
}
```

Output:

```
{xml_document}
<root>
1.  <flower>\n  <Sepal.Length>5.1</Sepal.Length>\n  <Sepal.Width>3.5</Sepal.Width>\n
<Species>setosa</Species>\n</flower>
2.  <flower>\n  <Sepal.Length>4.9</Sepal.Length>\n  <Sepal.Width>3.0</Sepal.Width>\n
<Species>setosa</Species>\n</flower> Sepal.Length:
5.1 Sepal.Width: 3.5 Species: setosa
Sepal.Length: 4.9 Sepal.Width: 3.0 Species: setosa
```

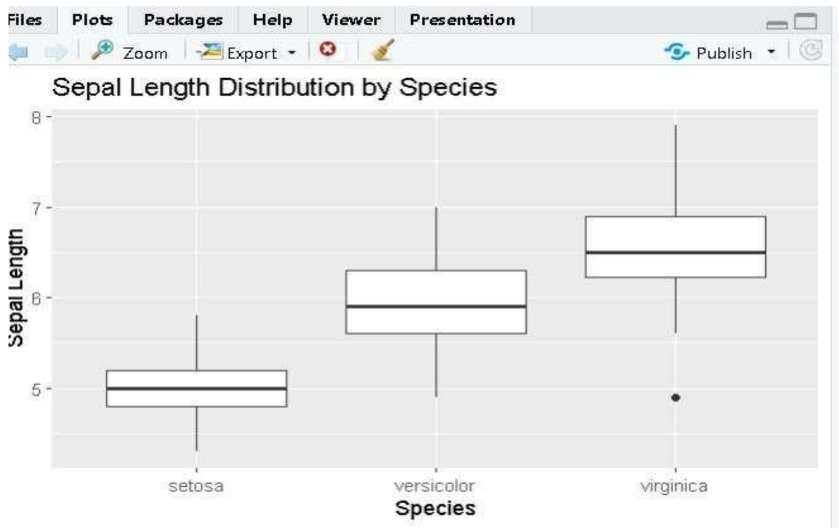
TITLE:	DATE: PAGE NO:

Visualizations
a) Find the data distributions using box plot and scatter plot.
b) Find the outliers using plot().
c) Plot the histogram, bar chart, and pie chart on sample data.

a) Program:

```
#box plot

library(ggplot2)

ggplot(data = iris, aes(x = Species, y = Sepal.Length)) +

  geom_boxplot() +

  labs(title = "Sepal Length Distribution by Species", x

      = "Species",

      y = "Sepal Length")
```

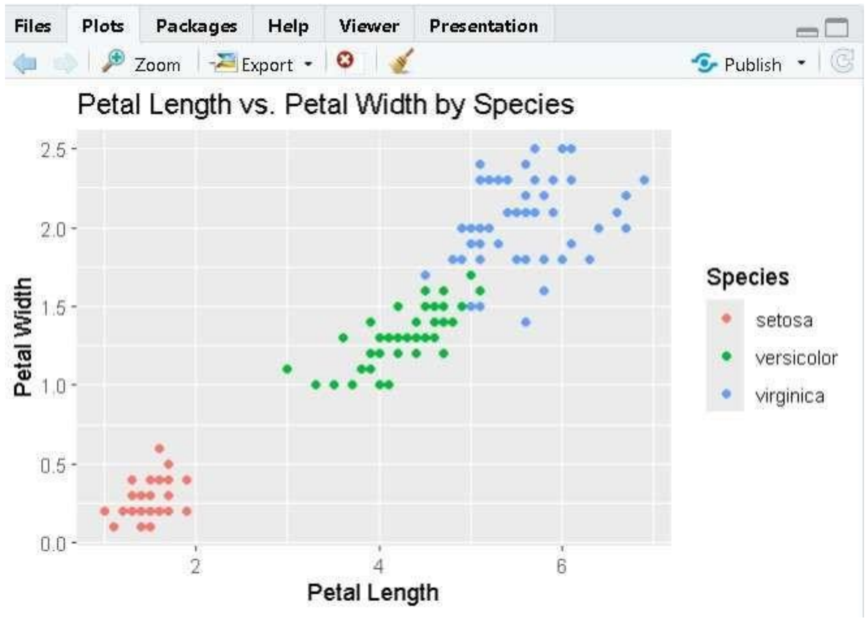
Output:



```
# scatter plot

library(ggplot2)

ggplot(data = iris, aes(x = Petal.Length, y = Petal.Width)) +

  geom_point(aes(color = Species)) +

  labs(title = "Petal Length vs. Petal Width by Species", x

      = "Petal Length",

      y = "Petal Width")
```

Output:

TITLE:	DATE: PAGE NO:



b) Program:

Find the outliers using plot

data <- c(10, 12, 13, 15, 16, 18, 19,33,3,36,63,99,32,15,6, 20, 22, 100, 105)

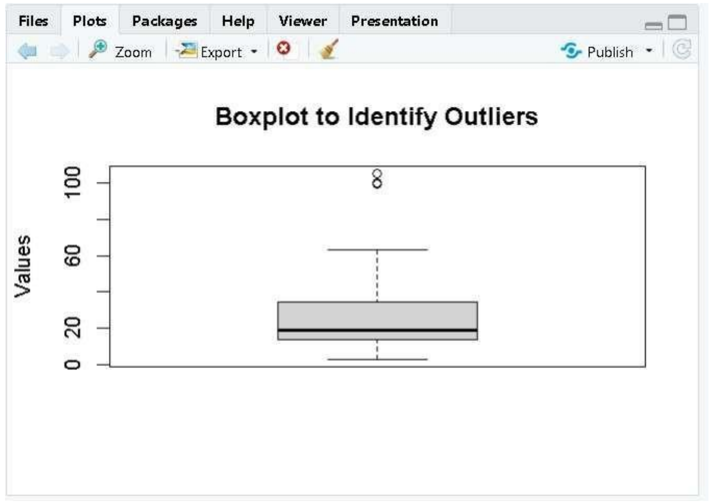
# Create a boxplot

boxplot(data, main = "Boxplot to Identify Outliers", ylab = "Values") #

Optional: print outliers detected by boxplot.stats

outliers <- boxplot.stats(data)$out print(paste("Outliers:",

paste(outliers, collapse = ", "))) **Output:**



[1] "Outliers: 99, 100, 105"

TITLE:	DATE: PAGE NO:

c) Program:

Plot the histogram, bar chart and pie chart on sample data #pie

chart

slices <- c(30, 20, 15, 35) labels

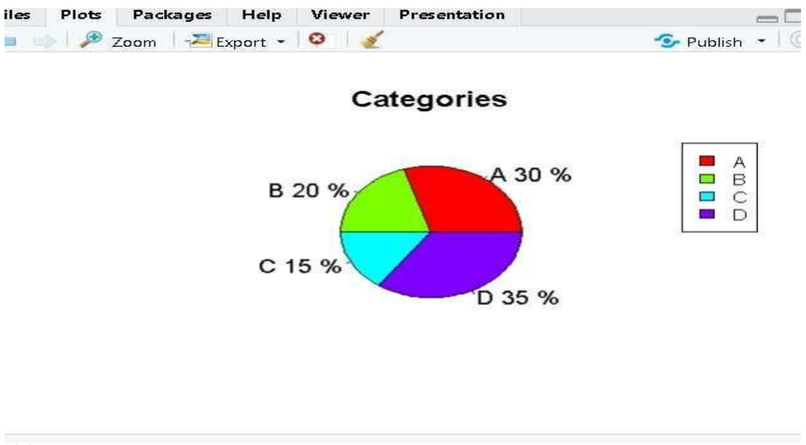
<- c("A", "B", "C", "D")

piepercent<- round(100 * slices / sum(slices), 1)

pie(slices, labels=paste(labels, piepercent, "%"), main="Categories", col=rainbow(length(slices)))

legend("topright", labels, fill=rainbow(length(slices)), cex=0.8)

Output:

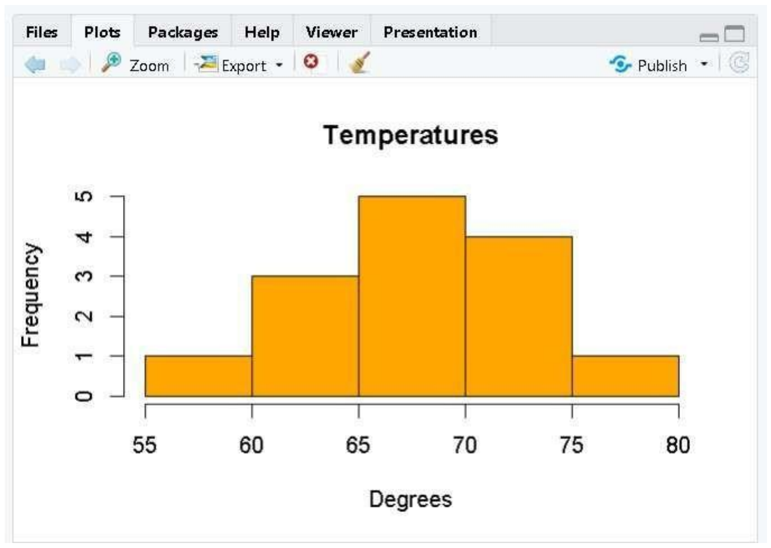


#histogram

temperatures <- c(67, 72, 74, 62, 76, 66, 65, 59, 61, 69, 70, 71, 75, 68)

hist(temperatures, main="Temperatures", xlab="Degrees", breaks=5, col="orange")

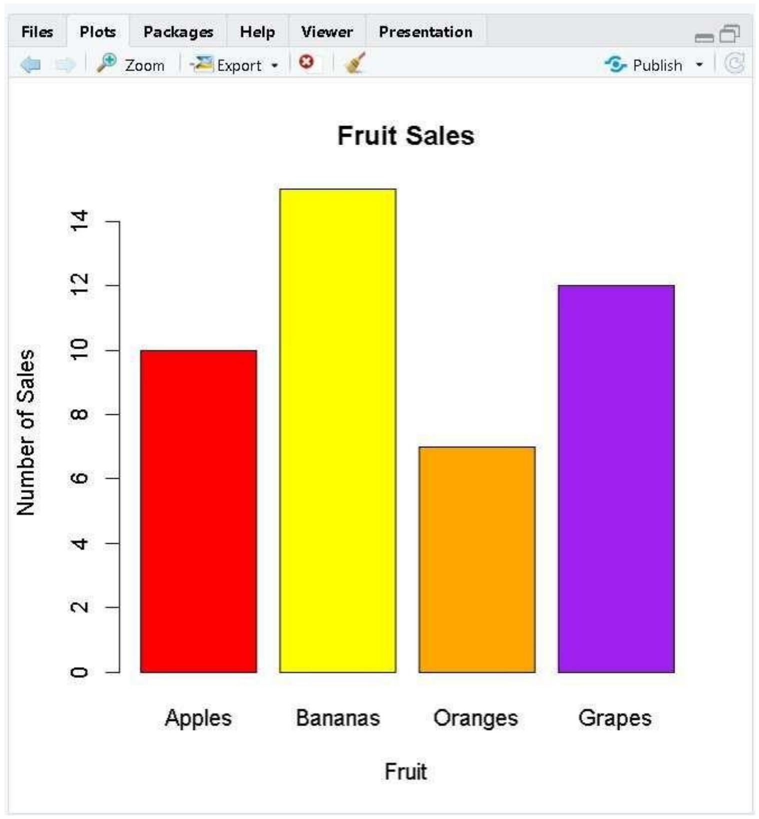
Output:



TITLE:	DATE: PAGE NO:

```
#barchart
categories <- c("Apples", "Bananas", "Oranges", "Grapes")
values <- c(10, 15, 7, 12)
barplot(values, names.arg = categories, main = "Fruit Sales",
        xlab = "Fruit", ylab = "Number of Sales",
        col = c("red", "yellow", "orange", "purple"))
```

Output:



TITLE:	DATE: PAGE NO:

Correlation and Covariance
a) Develop a program to find the correlation matrix on *iris* data.
b) Plot the correlation plot on dataset and visualize, giving an overview of relationships among data on *iris* data.
c) Analysis of covariance: variance (ANOVA) if data has categorical variables on *iris* data.

a) Program:

```
# Take only numeric columns from iris
num_data <- iris[, 1:4]

# Correlation matrix cor_matrix
<- cor(num_data)
print(cor_matrix)
```

Output:

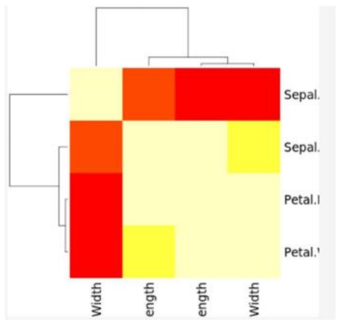
```
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Lengt     1.0000000  -0.1175698    0.871753    0.817941
h                                        8           1
Sepal.Widt     -0.1175698   1.0000000   -0.4284401  -0.366125
h                                                    9
Petal.Lengt     0.8717538  -0.4284401    1.0000000   0.962865
h                                                    4
Petal.Width     0.8179411  -0.3661259    0.9628654   1.000000
                                                     0
```

b) Program:

```
# install.packages("corrplot") # run once if not installed
library(corrplot)

corrplot(cor_matrix, method = "circle", type = "upper",
        tl.col = "black", tl.cex = 0.8)
```

Output:

```
             Sepal.Length Sepal.Width Petal.Length Petal.Width
Sepal.Length    1.0000000  -0.1175698    0.8717538   0.8179411
Sepal.Width    -0.1175698   1.0000000   -0.4284401  -0.3661259
Petal.Length    0.8717538  -0.4284401    1.0000000   0.9628654
Petal.Width     0.8179411  -0.3661259    0.9628654   1.0000000
```



TITLE:	DATE: PAGE NO:

c) Program:

```
# Compare Sepal.Length across species using ANOVA
anova_model <- aov(Sepal.Length ~ Species, data = iris)
summary(anova_model)
```

Output:

```
             Df Sum Sq Mean Sq F value Pr(>F)
Species       2  63.21  31.606   119.3 <2e-16 ***
Residuals   147  38.96   0.265
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05
```

TITLE:	DATE: PAGE NO:

6) Write an R-script that detects file type (.csv, .txt, .xlsx, .xml) and reads it accordingly.

Program:

```
read_file <- function(filename) { #
  Extract file extension
  ext <- tools::file_ext(filename)

  if (ext == "csv") {
    data <- read.csv(filename)
  } else if (ext == "txt") {
    data <- read.table(filename, header = TRUE)
  } else if (ext == "xlsx")
    { library(readxl)
    data <- read_excel(filename)
  } else if (ext == "xml")
    { library(xml2)
    data <- read_xml(filename)
  } else {
    stop("Unsupported file type!")
  }
  return(data)
}

# Example usage
write.csv(iris[1:5, ], "iris_sample.csv", row.names = FALSE)
result <- read_file("iris_sample.csv")
print(result)
```

Output:

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
 1        5.1         3.5          1.4         0.2   setos
                                                       a
 2        4.9         3.0          1.4         0.2   setos
                                                       a
 3        4.7         3.2          1.3         0.2   setos
                                                       a
 4        4.6         3.1          1.5         0.2   setos
                                                       a
 5        5.0         3.6          1.4         0.2   setos
                                                       a
```