

# **JOURNAL**

**Name– Shubham Shah**

**Class – Sybscit**

**Roll No. – 051**

**Div – A**

**Subject– Operation Research (OR)**

**Teacher Incharge – Manisha Ma'am**

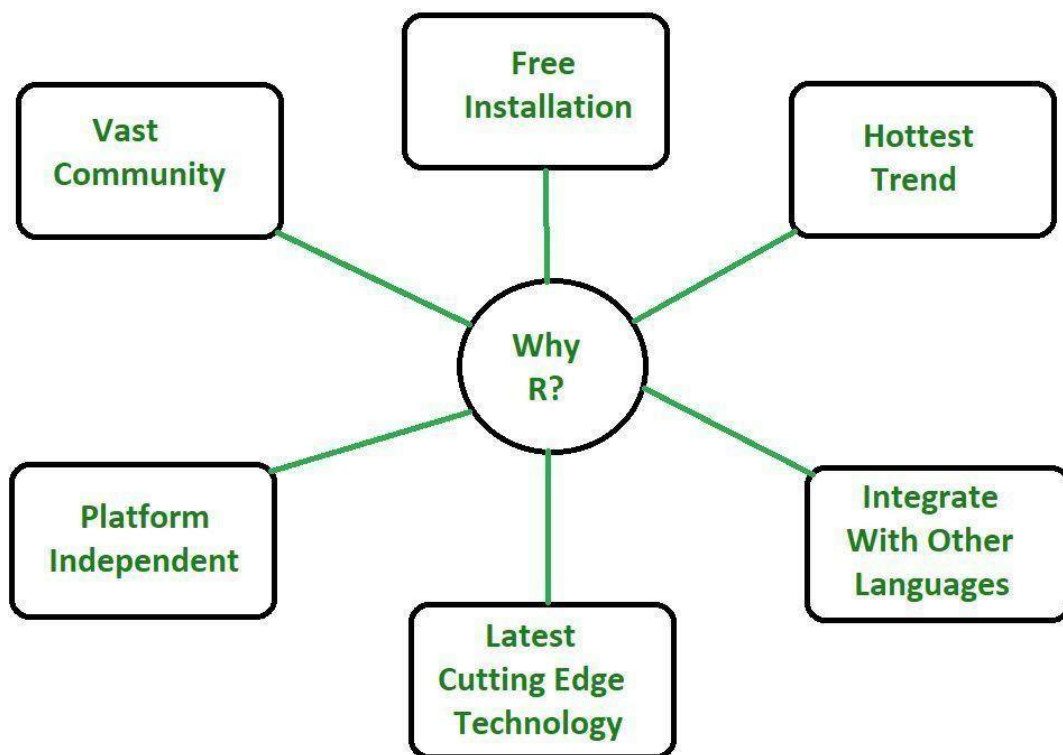
# PRACTICAL-1

## Introduction to R

R is an open-source programming language that is widely used as a statistical software and data analysis tool. R generally comes with the Command-line interface. R is available across widely used platforms like Windows, Linux, and macOS. Also, the R programming language is the latest cuttingedge tool.

It was designed by **Ross Ihaka and Robert Gentleman** at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team. R programming language is an implementation of the S programming language. It also combines with lexical scoping semantics inspired by Scheme. Moreover, the project conceives in 1992, with an initial version released in 1995 and a stable beta version in 2000.

### Why R Programming Language?



- R programming is used as a leading tool for machine learning, statistics, and data analysis. Objects, functions, and packages can easily be created by R.
- It's a platform-independent language. This means it can be applied to all operating system.
- It's an open-source free language. That means anyone can install it in any organization without purchasing a license.

- R programming language is not only a statistic package but also allows us to integrate with other languages (C, C++). Thus, you can easily interact with many data sources and statistical packages.
- The R programming language has a vast community of users and it's growing day by day.
- R is currently one of the most requested programming languages in the Data Science job market that makes it the hottest trend nowadays. **Features of R Programming Language**

#### Statistical Features of R:

- **Basic Statistics:** The most common basic statistics terms are the mean, mode, and median. These are all known as "Measures of Central Tendency." So using the R language we can measure central tendency very easily.
- **Static graphics:** R is rich with facilities for creating and developing interesting static graphics. R contains functionality for many plot types including graphic maps, mosaic plots, biplots, and the list goes on.
- **Probability distributions:** Probability distributions play a vital role in statistics and by using R we can easily handle various types of probability distribution such as Binomial Distribution, Normal Distribution, Chi-squared Distribution and many more.
- **Data analysis:** It provides a large, coherent and integrated collection of tools for data analysis.

What is R Used For?

Although R is a popular language used by many programmers, it is especially effective when used for Data analysis

Statistical inference

Machine learning algorithms

R offers a wide variety of statistics-related libraries and provides a favorable environment for statistical computing and design. In addition, the R programming language gets used by many quantitative analysts as a programming tool since it's useful for data importing and cleaning.

#### Why Do We Use R Programming Language?

There are various reasons why one should learn R programming language. Here are some:

- R is **free** and **open-source**. That means you don't need to pay for licenses.
- It is **platform-independent**. This makes it cost-effective and versatile. You only need to make one program that can run on various platforms.

- Currently, R has over **10,000 packages** available. It is stored in CRAN repositories, and it is continuously updated.
- It's popular for statistics and preferred by programmers for statistical tools.
- It's suitable for machine learning, offering features and packages for tasks like regression and classification.
- R helps with data wrangling. R contains various packages which help transform messy data into structured formats.
- Last but importantly, It has continuously evolved with a supportive community.

However, there are some drawbacks to consider:

- R has a steep learning curve and is better suited for experienced programmers.
- It's not super secure. It lacks basic security measures and can't be used for web-safe applications.
- Compared to other languages like Python or MATLAB, R may be slower.
- Memory management is also not the strong point of R language. It requires more memory as data is stored in physical memory. Although, cloud-based memory service may eliminate this drawback.
- R does not have consistent package quality. Documentation and package quality can vary due to the community-driven nature of R. Docs, and packages may be patchy or inconsistent as it doesn't have official dedicated support.

Despite these drawbacks, R remains a powerful and widely used language for data analysis and statistical computing.

## Installing R

- Can be downloaded for free from  
<http://www.r-project.org/>
- Download the version compatible with your OS

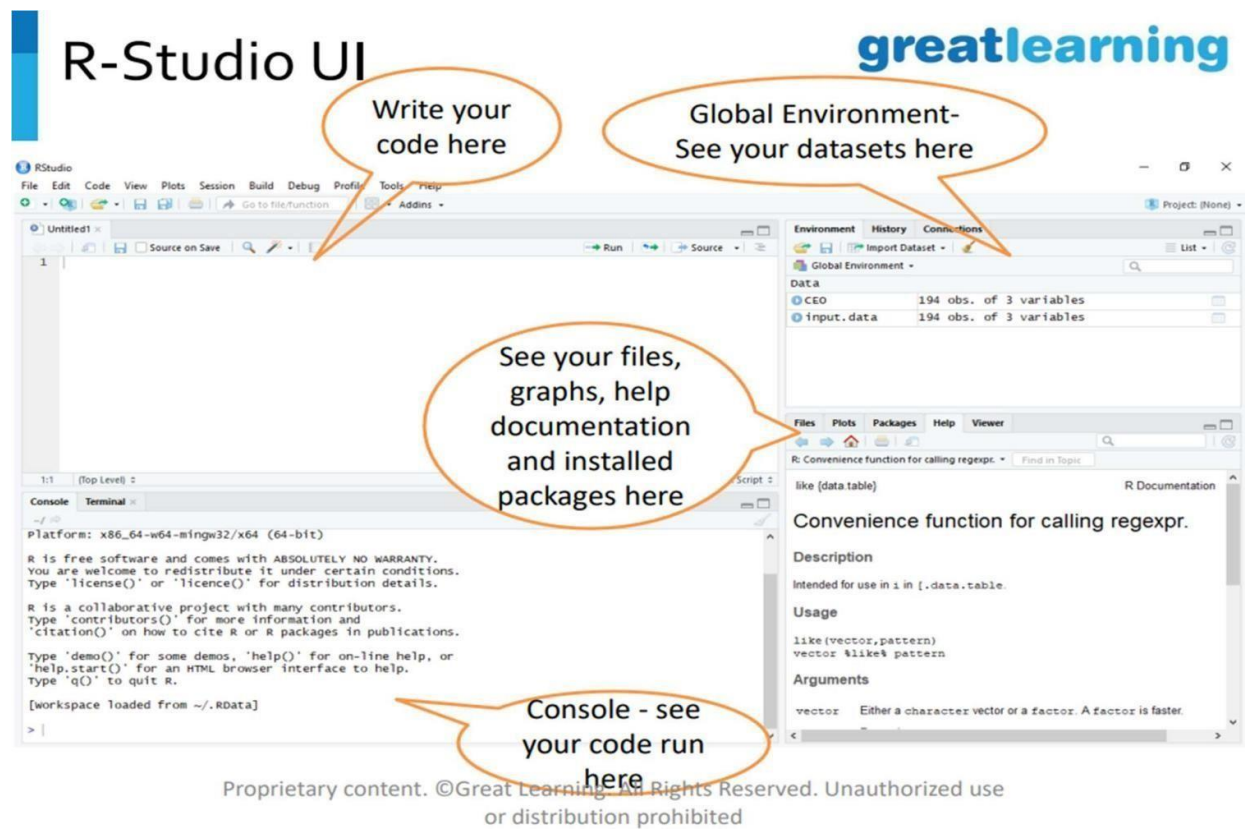
- Simple/Standard installation process

## Installing R -Studio

- Can be downloaded for free from:

<https://www.rstudio.com/products/rstudio/download/>

- Download the free version compatible with your OS
- R needs to be installed before installing R- Studio



### R Script File

Usually, you will do your programming by writing your programs in script files and then you execute those scripts at your command prompt with the help of R interpreter called **Rscript**. So let's start with writing following code in a text file called test. R as under –

```
# My first program in R Programming
myString <- "Hello, World!" print (
myString)
```

```
[1] "Hello, World!"
```

### *R Commands*

- Assignments E.g.:  $x = 1$ , or  $x <- 1$
- Functions E.g.: `print("Hello World")`
- Computations

E.g.:  $17 + 3$  ;  $x + 5$

- Mix E.g.:  $y = \sqrt{16}$ ;  $y = 15 + 5$
- Assignment queries will update objects in your R environment
- Queries without assignment, as well as 'call' of R objects will either generate an output in the console, or in the plot tab

### *R Basics: Data Types*

#### **Variable Assignment in R**

- A basic construct in programming is "variable"
- A variable allows you to store a piece of data ('datum', e.g. 6, 'Hello', etc..) or several pieces of data of a common type, and assign them a unique name
- You can then later 'call' this variable's name to easily access the value(s) that is/are stored within this variable.

**Careful, R is case sensitive: The variables 'x' and 'X' can coexist in R environment and have different values.**

### *Basic data types in R*

- R works with numerous data types. The most common types are:
- Decimals values like 3.5, called 'numeric'
- Natural numbers like 3 are called 'integers'. Integers are also numeric Boolean variables (TRUE or FALSE) are classified as 'logical'
- Text (or string) values are classified as 'character'

```
# Create two vectors. v1 <-  
c(3,8,4,5,0,11) v2 <-  
c(4,11,0,8,1,2)
```

```
# Vector addition. add.result
<- v1+v2 print(add.result)

# Vector subtraction.
sub.result <- v1-v2
print(sub.result)

# Vector multiplication.
multi.result <- v1*v2
print(multi.result)

# Vector division. divi.result
<- v1/v2 print(divi.result)
```

## R Basics: Data Structures

### *R Objects: Vectors*

- To assign multiple values to a variable, we can use an R object called a ‘vector’
- A vector is a sequence/collection of data elements of the same basic type. Members in a vector are officially called components.

For Example: `my_vector = c(14,26,38,30)`

- To access a specific element in the vector, we simply need to call `variable_name[i]`, ‘i’ being the element’s position in the vector.

For example: `vect[3]` would return 38

### **Vector arithmetic**

Two vectors of same length can be added, subtracted, multiplied or divided giving the result as a vector output.

```
[1] 7 19 4 13 1 13
[1] -1 -3 4 -3 -1 9
[1] 12 88 0 40 0 22
[1] 0.7500000 0.7272727 Inf 0.6250000 0.0000000 5.5000000
```

When we execute the above code, it produces the following result –

### *R Objects: Matrices*

- A matrix is a sequence/collection of data elements of the same basic type arranged in a two-dimensional rectangular layout.
- Being a 2-dimensional object, in order to obtain a specific value within the matrix, 2 coordinates needs to be entered. For example:  
my\_matrix[i,j] would return the element on the ith row, in the jth column
- my\_matrix[i,] would return the entire ith row
- my\_matrix[,j] would return the entire jth column

## Matrices

A matrix is a two-dimensional rectangular data set. It can be created using a vector input to the matrix function.

```
# Create a matrix.
```

```
M = matrix(c('a','a','b','c','b','a'), nrow = 2,  
ncol = 3, byrow = TRUE) print(M)
```

When we execute the above code, it produces the following result .

```
[,1] [,2] [,3]
[1,] "a" "a" "b"
[2,] "c" "b" "a"
```



# **PRACTICAL-2**

## **Importing Packages R: Packages**

- R Packages are collections of R functions and data sets
- Some standard ones come with R installation
- Others can be installed in a few clicks in R studio, or using `install.packages("package name")` function. You can choose the CRAN Mirror closest to your location, but the default R studio is consistently good all over the world.
- Some have to be downloaded (from <http://cran.rproject.org/>), or through Google and manually installed
- Once installed we need to call the package in when needed using `library("package name")`

## **Binding**

- Binding columns: If 2 datasets, a dataset and a vector, or 2 vectors have the same number of values (rows in the case of datasets), they can be placed together into one same dataset using `cbind()`
- This is different from « merging » (see later chapter), hence there is no row matching system: rows need to be in the exact same order for the data to make sense.
- See example:- Binding rows: If 2 datasets have the same columns(order, data types, names), one can be appended under the other using `rbind()`

# PRACTICAL – 3

## PROBLMS ON LPP USING R

### PRACTICAL – 3(A)

1. Maximize  $Z = 3X_1 + 4X_2$

SUBJECT TO

$X_1 + 0X_2 \leq 3$

$X_1 + X_2 \leq 4$

$X_1, X_2 \geq 0$

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R script for solving the LPP.

```
1 library(lpSolve)
2 f.obj <- c(3,4)
3 f.con <- matrix(c(1,0,1,1),nrow=2,byrow=TRUE)
4 f.dir <- c("<=", "<=")
5 f.rhs <- c(3,4)
6 lp("max",f.obj,f.con,f.dir,f.rhs)$solution
7 lp("max",f.obj,f.con,f.dir,f.rhs,compute.sens=TRUE)
8
```
- Console:** Shows the execution output.

```
R 4.3.1 ~ ./
> library(lpSolve)
> f.obj <- c(3,4)
> f.con <- matrix(c(1,0,1,1),nrow=2,byrow=TRUE)
> f.dir <- c("<=", "<=")
> f.rhs <- c(3,4)
> lp("max",f.obj,f.con,f.dir,f.rhs)$solution
[1] 0.4
> lp("max",f.obj,f.con,f.dir,f.rhs,compute.sens=TRUE)
Success: the objective function is 16
>
```
- Environment:** Lists the objects created in the global environment.

Object	Class	Attributes
f.con	num	[1:2, 1:2] 1 1 0 1
f.dir	chr	[1:2] "<=" "<="
f.obj	num	[1:2] 3 4
f.rhs	num	[1:2] 3 4
- Files:** Shows the installed and updated packages.

Name	Description	Version
lpSolve	Interface to 'lp_solve' v. 5.5 to Solve Linear/integer Programs	5.6.19
- System Library:** Lists the installed system packages.

Package	Description	Version
base	The R Base Package	4.3.1
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28.1
class	Functions for Classification	7.3-22
cluster	"Finding Groups in Data"; Cluster Analysis Extended	2.1.4
codetools	Code Analysis Tools for R	0.2-19
compiler	The R Compiler Package	4.3.1
datasets	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...	0.8-84
foreign	The R Foreign Package	4.3.1
graphics	The R Graphics Package	4.3.1
grDevices	The R Graphics Devices and Support for Colours and Fonts	4.3.1
grid	The Grid Graphics Package	4.3.1
kernelSmooth	Functions for Kernel Smoothing Supporting Wand & ...	2.23-21

## PRACTICAL – 3(B)

2. Minimize  $Z = 4x + 8y$

Subject to

$16X + y \geq 18$

$X + 4y \geq 12$

$2X + Y \geq 10$

$X, Y \geq 0$

The screenshot displays the RStudio interface with a script editor on the left and the Environment and Packages panels on the right.

**Script Editor:**

```
1 #lpp program
2 library(lpSolve)
3 f.obj <- c(4,8)
4 f.con <- matrix(c(16,1,1,4,2,1),nrow=3,byrow=TRUE)
5 f.dir <- c(">=", ">=", ">=")
6 f.rhs <- c(18,12,10)
7 #ans
8 lp("min",f.obj,f.con,f.dir,f.rhs)$solution
9 #obj function
10 lp("min",f.obj,f.con,f.dir,f.rhs,compute.sens=TRUE)
11
12
```

**Console:**

```
R 4.3.1 ~> #ans
> lp("min",f.obj,f.con,f.dir,f.rhs)$solution
[1] 4 2
> #obj function
> lp("min",f.obj,f.con,f.dir,f.rhs,compute.sens=TRUE)
Success: the objective function is 32
>
```

**Environment Panel:**

Variable	Class	Dimensions	Values
f.con	num	[1:3, 1:2]	16 1 2 1 4 1
f.dir	chr	[1:3]	">=" ">=" ">="
f.obj	num	[1:2]	4 8
f.rhs	num	[1:3]	18 12 10

**Packages Panel:**

Name	Description	Version
MASS	Support Functions and Datasets for Venables and Ripley's MASS	7.3-60
Matrix	Sparse and Dense Matrix Classes and Methods	1.5-4.1
methods	Formal Methods and Classes	4.3.1
mgcv	Mixed GAM Computation Vehicle with Automatic Smoothness Estimation	1.8-42
nime	Linear and Nonlinear Mixed Effects Models	3.1-162
nnet	Feed-Forward Neural Networks and Multinomial Log-Linear Models	7.3-19
parallel	Support for Parallel Computation in R	4.3.1
rpart	Recursive Partitioning and Regression Trees	4.1.19
spatial	Functions for Kriging and Point Pattern Analysis	7.3-16
splines	Regression Spline Functions and Classes	4.3.1
stats	The R Stats Package	4.3.1
stats4	Statistical Functions using S4 Classes	4.3.1
survival	Survival Analysis	3.5-5
tcltk	Tcl/Tk Interface	4.3.1
tools	Tools for Package Development	4.3.1
translations	The R Translations Package	4.3.1

## PRACTICAL – 3(C)

Solve

Minimize  $Z = 4X + 8Y$

Subject to

$2X + 2Y \leq 28$

$3X + 2Y \geq 30$

$4X + 2Y \leq 36, X, Y \geq 0$

The screenshot shows the RStudio environment with a script editor, console, and environment pane. The script defines a linear programming problem and solves it using the `lpSolve` package.

```
1 library(lpSolve)
2 f.obj <- c(4,8)
3 f.con <- matrix(c(2,2,3,2,4,2),nrow=3,byrow=TRUE)
4 f.dir <- c("<=", ">=", "<=")
5 f.rhs <- c(28,30,36)
6 #ans
7 lp("min",f.obj,f.con,f.dir,f.rhs)$solution
8 #obj function
9 lp("min",f.obj,f.con,f.dir,f.rhs,compute.sens=TRUE)
10
```

The console output shows the solution and the objective function value:

```
> #ans
> lp("min",f.obj,f.con,f.dir,f.rhs)$solution
[1] 6 6
> #obj function
> lp("min",f.obj,f.con,f.dir,f.rhs,compute.sens=TRUE)
Success: the objective function is 72
>
```

The Environment pane shows the following variables:

Variable	Class	Value
f.con	num	[1:3, 1:2] 2 3 4 2 2 2
f.dir	chr	[1:3] "<=" ">=" "<="
f.obj	num	[1:2] 4 8
f.rhs	num	[1:3] 28 30 36

The Packages pane shows the installed packages:

Package	Description	Version
lpSolve	Interface to 'lp_solve' v. 5.5 to Solve Linear/Integer Programs	5.6.19
base	The R Base Package	4.3.1
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28.1
class	Functions for Classification	7.3-22
cluster	'Finding Groups in Data': Cluster Analysis Extended	2.1.4
codetools	Code Analysis Tools for R	0.2-19
compiler	The R Compiler Package	4.3.1
datasets	The R Datasets Package	4.3.1
foreign	Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ...	0.8-84
graphics	The R Graphics Package	4.3.1
grDevices	The R Graphics Devices and Support for Colours and Fonts	4.3.1
grid	The Grid Graphics Package	4.3.1
KernSmooth	Functions for Kernel Smoothing Supporting Wand & ...	2.23-21

# PRACTICAL – 4

## SOLVE PROBLEMS ON LPP USING SIMPLEX METHOD USING R

### PRACTICAL – 4(A)

**Maximization**

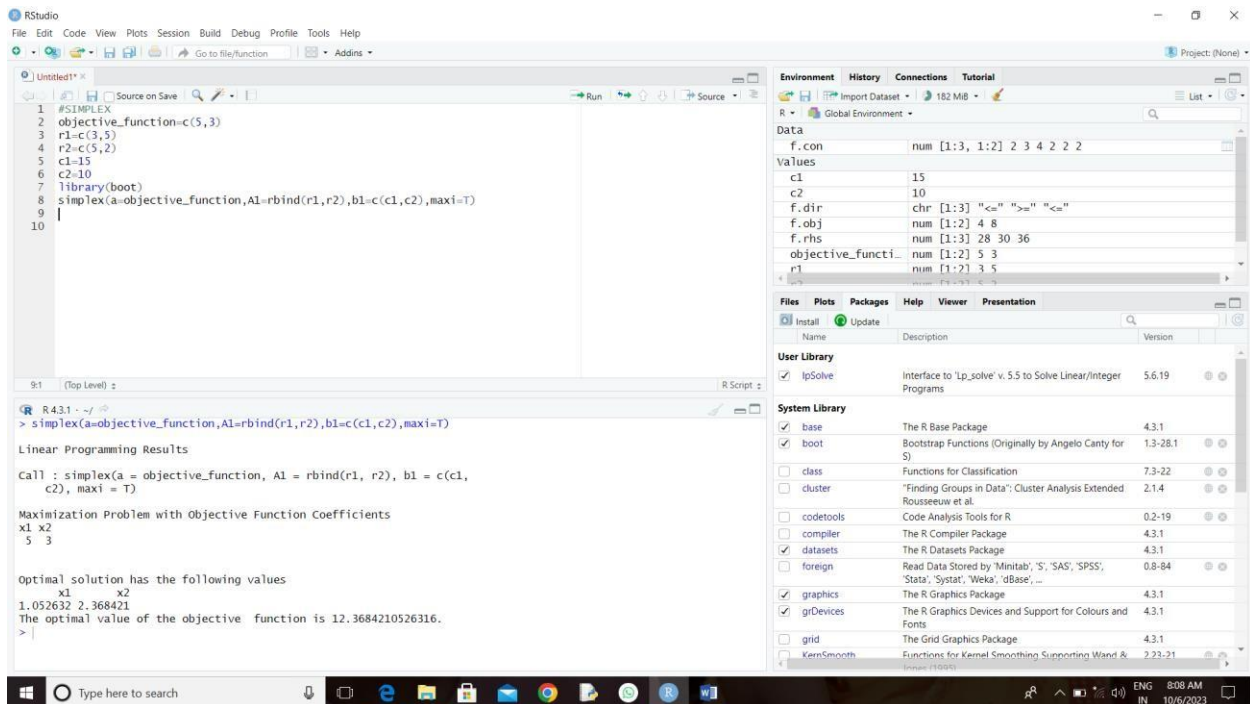
$$Z = 5X_1 + 3X_2$$

**SUBJECT TO**

$$3X_1 + 5X_2 \leq 15$$

$$5X_1 + 2X_2 \leq 10$$

$$X_1, X_2 \geq 0$$



```
1 #SIMPLEX
2 objective_function=c(5,3)
3 r1=c(3,5)
4 r2=c(5,2)
5 c1=15
6 c2=10
7 library(boot)
8 simplex(a=objective_function,A1=rbind(r1,r2),b1=c(c1,c2),maxi=T)
9
10
```

Linear Programming Results

Call : simplex(a = objective\_function, A1 = rbind(r1, r2), b1 = c(c1, c2), maxi = T)

Maximization Problem with Objective Function Coefficients

x1	x2
1.052632	2.368421

The optimal value of the objective function is 12.3684210526316.

Environment

Variable	Value
f.con	num [1:3, 1:2] 2 3 4 2 2 2
c1	15
c2	10
f.dir	chr [1:3] "<=" ">=" "<="
f.obj	num [1:2] 4 8
f.rhs	num [1:3] 28 30 36
objective_functi...	num [1:2] 5 3
r1	num [1:2] 3 5

User Library

Package	Version
lpSolve	5.6.19

System Library

Package	Version
base	4.3.1
boot	1.3-28.1
class	7.3-22
cluster	2.1.4
codetools	0.2-19
compiler	4.3.1
datasets	4.3.1
foreign	0.8-84
graphics	4.3.1
grDevices	4.3.1
grid	4.3.1
KernSmooth	2.23-21

## PRACTICAL – 4(B)

Maximize

$$Z = 40X_1 + 50X_2$$

SUBJECT TO

$$3X_1 + X_2 \leq 9$$

$$X_1 + 2X_2 \leq 8$$

$$X_1, X_2 \geq 0$$

The screenshot shows the RStudio interface with a script editor, console, and environment pane. The script defines a linear programming problem and solves it using the `simplex` function from the `lpSolve` package.

```
1 #SIMPLEX
2 objective_function=c(40,50)
3 r1<- c(9,1)
4 r2<- c(1,2)
5 c1=9
6 c2=8
7 library(boot)
8 simplex(a=objective_function,A1=rbind(r1,r2),b1=c(c1,c2),maxi=TRUE)
9
```

The console output shows the results of the simplex method:

```
R 4.3.1 ~\> simplex(a=objective_function,A1=rbind(r1,r2),b1=c(c1,c2),maxi=TRUE)
Linear Programming Results

Call: simplex(a = objective_function, A1 = rbind(r1, r2), b1 = c(c1,
c2), maxi = TRUE)

Maximization Problem with Objective Function Coefficients
x1 x2
40 50

Optimal solution has the following values
x1 x2
2 3
The optimal value of the objective function is 230.
>
```

The Environment pane shows the following variables:

Variable	Class	Value
f.con	num	[1:3, 1:2] 2 3 4 2 2 2
c1	num	9
c2	num	8
f.dir	chr	[1:3] "c1" "c2" "c3"
f.obj	num	[1:2] 4 8
f.rhs	num	[1:3] 28 30 36
objective_functi_	num	[1:2] 40 50
r1	num	[1:2] 3 1
r2	num	[1:2] 1 2

The Packages pane shows the following installed packages:

Package	Version
lpSolve	5.6.19
base	4.3.1
boot	1.3-28.1
class	7.3-22
cluster	2.1.4
codetools	0.2-19
compiler	4.3.1
datasets	4.3.1
foreign	0.8-84
graphics	4.3.1
grDevices	4.3.1
grid	4.3.1
KernSmooth	2.23-21

## PRACTICAL – 4(C)

Maximize

$$Z = 10X_1 + 6X_2 + 4X_3$$

SUBJECT TO

$$X_1 + X_2 + X_3 \leq 100$$

$$10X_1 + 4X_2 + 5X_3 \leq 600$$

$$2X_1 + 2X_2 + 6X_3 \leq 300$$

$$X_1, X_2, X_3 \geq 0$$

The screenshot shows the RStudio interface with a script editor on the left and the Environment and Packages panels on the right. The script defines an objective function and constraints, then uses the `simplex` function to solve the problem.

```
1 objective_function=c(10,6,4)
2 r1=c(1,1,1)
3 r2=c(10,4,5)
4 r3=c(2,2,6)
5 c1=100
6 c2=600
7 c3=300
8 library(boot)
9 simplex(a=objective_function,A1=rbind(r1,r2,r3),b1=c(c1,c2,c3),maxi=TRUE)
10
```

The console output shows the results of the simplex method:

```
R 4.3.1 ~ ./
> simplex(a=objective_function,A1=rbind(r1,r2,r3),b1=c(c1,c2,c3),maxi=TRUE)
Linear Programming Results

Call : simplex(a = objective_function, A1 = rbind(r1, r2, r3), b1 = c(c1,
      c2, c3), maxi = TRUE)

Maximization Problem with Objective Function Coefficients
x1 x2 x3
10 6 4

optimal solution has the following values
      x1      x2      x3
33.33333 66.66667 0.00000
The optimal value of the objective function is 733.333333333333.
>
```

The Environment panel shows the following variables:

Variable	Value
f.con	num [1:3, 1:2] 2 3 4 2 2 2
c1	100
c2	600
c3	300
f.dir	chr [1:3] "<=" ">=" "<="
f.obj	num [1:2] 4 8
f.rhs	num [1:3] 28 30 36
objective_functi...	num [1:3] 10 6 4
r1	num [1:3] 1 1 1
r2	num [1:3] 10 4 5
r3	num [1:3] 2 2 6

The Packages panel shows the following installed packages:

Package	Version
lpSolve	5.6.19
base	4.3.1
boot	1.3-28.1
class	7.3-22
cluster	2.1.4
codetools	0.2-19
compiler	4.3.1
datasets	4.3.1
foreign	0.8-84
graphics	4.3.1
grDevices	4.3.1
grid	4.3.1
kernSmooth	2.23-21

## PRACTICAL – 5

### SOLVING PROBLEMS ON TRANSPORTATION PROBLEM USING R

#### PRACTICAL - 5(A)

SOLVE THE FOLLOWING TRANSPORTATION PROBLEM FOR MINIMAL COST

		Destination				Supply
		D1	D2	D3	D4	
Source	O1	3	1	7	4	300
	O2	2	6	5	9	400
	O3	8	3	3	2	500
Demand:		250	350	400	200	1200

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for solving the transportation problem using the `lpSolve` package. The code defines costs, supply, demand, and signs, then uses `lp.transport` to find the optimal solution.
- Environment:** Lists the objects created in the session: `costs` (numeric vector), `lptrans` (list), `demand` (numeric vector), `demand.signs` (character vector), `supply` (numeric vector), and `supply.signs` (character vector).
- Console:** Shows the execution of the code and the resulting solution. The optimal objective value is 2850.
- Package Manager:** Lists installed and available packages, including `lpSolve` (5.6.19) and various base R packages.

```
1 library(lpSolve)
2 costs<-matrix(c(3,1,7,4,2,6,5,9,8,3,3,2),ncol=4,byrow=TRUE)
3 supply.signs <- rep("<=",3)
4 supply <- c(300,400,500)
5 demand.signs <- rep(">=",4)
6 demand <- c(250,350,400,200)
7 lptrans <- lp.transport(costs,"min",supply.signs,supply,demand.signs,demand)
8 lptrans$status
9 lptrans$solution
10 lptrans$objval
11
```

Console Output:

```
> lptrans$status
[1] 0
> lptrans$solution
      [,1] [,2] [,3] [,4]
[1,]    0   300    0    0
[2,]   250    0   150    0
[3,]    0    50   250   200
> lptrans$objval
[1] 2850
```



## PRACTICAL – 5(B)

### SOLVED EXAMPLES

#### Example 1 :

Solve the following transportation problem for the optimum cost

	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	Capacity
O <sub>1</sub>	6	5	1	3	100
O <sub>2</sub>	4	8	7	2	125
O <sub>3</sub>	6	3	9	3	75
Demand	70	90	80	60	300

The screenshot displays the RStudio interface with the following components:

- Source Editor:** Contains R code for solving the transportation problem using the `lpSolve` package. The code defines costs, supply, demand, and signs, then uses `lp.transport` to find the optimal solution.
- Environment:** Shows the objects created in the global environment: `costs` (numeric vector), `lptrans` (list), `demand` (numeric vector), `demand.signs` (character vector), `supply` (numeric vector), and `supply.signs` (character vector).
- Console:** Shows the execution of the code, including the output of `lptrans$status` (0), `lptrans$solution` (a 3x4 matrix), and `lptrans$objval` (785).
- Files:** Shows the project files, including the R script and the solution output file.
- System Library:** Lists the installed R packages, including `lpSolve` (version 5.6.19).

```

1 library(lpSolve)
2 costs<-matrix(c(6,5,1,3,4,8,7,2,6,3,9,5),ncol=4,byrow=TRUE)
3 supply.signs <- rep("<=" ,3)
4 supply <- c(100,125,75)
5 demand.signs <- rep(">=" ,4)
6 demand <- c(70,90,80,60)
7 lptrans <- lp.transport(costs,"min",supply.signs,supply,demand.signs,demand)
8 lptrans$status
9 lptrans$solution
10 lptrans$objval
11

```

```

> lptrans$status
[1] 0
> lptrans$solution
      [,1] [,2] [,3] [,4]
[1,]    0    15    80    5
[2,]   70     0     0   55
[3,]    0    75     0     0
> lptrans$objval
[1] 785

```

## PRACTICAL - 5(C)

SOLVE FOR OPTIMUM VALUE

		Destination				Supply(S <sub>i</sub> )
		D1	D2	D3	D4	
Source	O1	3	1	7	4	250
	O2	2	6	5	9	350
	O3	8	3	3	2	400
Demand(D <sub>j</sub> ):		200	300	350	150	

The screenshot shows the RStudio interface with the following components:

- Source Editor:** Contains the R code for solving the transportation problem using the `lpSolve` package.
- Environment:** Lists the objects created in the global environment, including `costs`, `lptrans`, `demand`, `demand.signs`, `supply`, and `supply.signs`.
- Files:** Shows the project files, including `lpSolve` and `lpSolve.R`.
- Console:** Displays the execution of the R code and the resulting output, including the optimal objective value of 2450.

```

1 library(lpSolve)
2 costs <- matrix(c(3,1,7,4,2,6,5,9,8,3,3,2), ncol = 4, byrow = TRUE)
3 supply.signs <- rep("<=", 3)
4 supply <- c(250, 350, 400)
5 demand.signs <- rep(">=", 4)
6 demand <- c(200, 300, 350, 150)
7 lptrans <- lp.transport(costs, "min", supply.signs, supply, demand.signs, demand)
8 lptrans$status
9 lptrans$solution
10 lptrans$objval
11

```

```

> lptrans$status
[1] 0
> lptrans$solution
      [,1] [,2] [,3] [,4]
[1,]    0  250    0    0
[2,]  200    0  150    0
[3,]    0   50  200  150
> lptrans$objval
[1] 2450

```

# PRACTICAL 6

## SOLVE ASSIGNMENT PROBLEM USING R

### PRACTICAL – 6(A)

#### Example 1 :

M/s. Global Pvt. Ltd. has five jobs A, B, C, D & E to be done by five Employees Vishesh, Dhruv, Nilanj, Aditya, and Shivam. Each Employee is assigned one and only one job. The number of hours each Employee would take to complete each job is given by the following table.

Employees	Jobs				
	A	B	C	D	E
Vishesh	28	27	24	35	38
Dhruv	26	24	23	32	39
Nilanj	18	20	22	30	32
Aditya	27	30	25	24	27
Shivam	29	31	40	40	36

Find the assignment of jobs to Employee in such a way that the total time taken to perform 5 jobs is minimum.

The screenshot shows the RStudio interface with the following content:

**Source Editor:**

```
1 library(lpSolve)
2 assign.costs <- matrix(c(28,27,24,35,38,26,24,23,32,27,30,25,24,27,29,31,40,40,36), ncol=5, byrow=TRUE)
3 assign.costs
4 lp.assign(assign.costs)
5 lp.assign(assign.costs)$solution
```

**Console:**

```
> assign.costs
     [,1] [,2] [,3] [,4] [,5]
[1,] 28 27 24 35 38
[2,] 26 24 23 32 39
[3,] 18 20 22 30 32
[4,] 27 30 25 24 27
[5,] 29 31 40 40 36
> lp.assign(assign.costs)
Success: the objective function is 126
> lp.assign(assign.costs)$solution
     [,1] [,2] [,3] [,4] [,5]
[1,] 0 0 1 0 0
[2,] 0 1 0 0 0
[3,] 1 0 0 0 0
[4,] 0 0 0 1 0
[5,] 0 0 0 0 1
>
```

**Environment:**

- assign.costs: num [1:5, 1:5] 28 26 18 27 29 27 24 20 30 31 ...
- costs: num [1:3, 1:4] 3 2 8 1 6 3 7 5 3 4 ...
- lptrans: List of 20
- values:
  - demand: num [1:4] 200 300 350 150
  - demand.signs: chr [1:4] ">=" ">=" ">=" ">="
  - supply: num [1:3] 250 350 400
  - supply.signs: chr [1:3] "<=" "<=" "<="

**Files:** Untitled1.R

**Plots:** (Empty)

**Packages:**

- lpSolve: Interface to 'lp\_solve' v. 5.5 to Solve Linear/Integer Programs (5.6.19)

**User Library:**

- base: The R Base Package (4.3.1)
- boot: Bootstrap Functions (Originally by Angelo Canty for S) (1.3-28.1)
- class: Functions for Classification (7.3-22)
- cluster: "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. (2.14)
- codetools: Code Analysis Tools for R (0.2-19)
- compiler: The R Compiler Package (4.3.1)
- datasets: The R Datasets Package (4.3.1)
- foreign: Read Data Stored by 'Minitab', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ... (0.8-84)
- graphics: The R Graphics Package (4.3.1)
- grDevices: The R Graphics Devices and Support for Colours and Fonts (4.3.1)
- grid: The Grid Graphics Package (4.3.1)
- KernSmooth: Functions for Kernel Smoothing Supporting Wand & ... (2.23-21)

**System Library:**

- base: The R Base Package (4.3.1)
- boot: Bootstrap Functions (Originally by Angelo Canty for S) (1.3-28.1)
- class: Functions for Classification (7.3-22)
- cluster: "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al. (2.14)
- codetools: Code Analysis Tools for R (0.2-19)
- compiler: The R Compiler Package (4.3.1)
- datasets: The R Datasets Package (4.3.1)
- foreign: Read Data Stored by 'Minitab', 'SAS', 'SPSS', 'Stata', 'Systat', 'Weka', 'dBase', ... (0.8-84)
- graphics: The R Graphics Package (4.3.1)
- grDevices: The R Graphics Devices and Support for Colours and Fonts (4.3.1)
- grid: The Grid Graphics Package (4.3.1)
- KernSmooth: Functions for Kernel Smoothing Supporting Wand & ... (2.23-21)

## PRACTICAL - 6(B)

### Example 3 :

The MBI manufacturing company plans to manufacture 4 types of new minicomputers. Each of the plant has manufacturing capacity for one product only. The unit manufacturing cost for producing the different minicomputers at the four plants are shown in table below. What is the lowest total manufacturing cost?

Plants	Minicomputers Type			
	1	2	3	4
1	24	18	22	28
2	28	20	16	24
3	26	28	20	24
4	22	32	24	22

The screenshot shows the RStudio interface with the following components:

- Script Editor:** Contains the R code for solving the linear programming problem.
 

```
1 library(lpSolve)
2 assign.costs <- matrix(c(24,18,22,28,20,16,24,26,28,20,24,22,32,24,22), ncol=4, byrow=TRUE)
3 assign.costs
4 lp.assign(assign.costs)
5 lp.assign(assign.costs)$solution
6
```
- Console:** Shows the output of the R code.
 

```
R 4.3.1 ~ /
> assign.costs
      [,1] [,2] [,3] [,4]
[1,] 24 18 22 28
[2,] 28 20 16 24
[3,] 26 28 20 24
[4,] 22 32 24 22
> lp.assign(assign.costs)
success: the objective function is 80
> lp.assign(assign.costs)$solution
      [,1] [,2] [,3] [,4]
[1,] 0 1 0 0
[2,] 0 0 1 0
[3,] 0 0 0 1
[4,] 1 0 0 0
>
```
- Environment:** Lists the objects in the global environment:
  - `assign.costs`: num [1:4, 1:4] 24 28 26 22 18 20 28 32 22 16 ...
  - `costs`: num [1:3, 1:4] 3 2 8 1 6 3 7 5 3 4 ...
  - `lptrans`: List of 20
  - `values`:
    - `demand`: num [1:4] 200 300 350 150
    - `demand.signs`: chr [1:4] ">=" ">=" ">=" ">="
    - `supply`: num [1:3] 250 350 400
    - `supply.signs`: chr [1:3] "<=" "<=" "<="
- Files:** Shows the installed and updated packages.
  - User Library:**
    - ☒ `lpSolve`: Interface to 'lp\_solve' v. 5.5 to Solve Linear/Integer Programs (5.6.19)
  - System Library:**
    - ☒ `base`: The R Base Package (4.3.1)
    - ☐ `boot`: Bootstrap Functions (Originally by Angelo Canty for S) (1.3-28.1)
    - ☐ `class`: Functions for Classification (7.3-22)
    - ☐ `cluster`: "Finding Groups in Data"; Cluster Analysis Extended (2.1.4)
    - ☐ `codetools`: Code Analysis Tools for R (0.2-19)
    - ☐ `compiler`: The R Compiler Package (4.3.1)
    - ☒ `datasets`: The R Datasets Package (4.3.1)
    - ☐ `foreign`: Read Data Stored by 'Minitab', 'S', 'SAS', 'SPSS', 'Stata', 'Syntac', 'Weka', 'dBase', ... (0.8-84)
    - ☒ `graphics`: The R Graphics Package (4.3.1)
    - ☒ `grDevices`: The R Graphics Devices and Support for Colours and Fonts (4.3.1)
    - ☐ `grid`: The Grid Graphics Package (4.3.1)
    - ☐ `kernelSmooth`: Functions for Kernel Smoothing Supporting Wand & ... (2.23-21)

## PRACTICAL – 6(C)

**Example 4 :**  
**Case Study**

1) A departmental store agency runs five stores located at different parts of greater Bombay. Each store is administered by the manager appointed by the agency. The manager resides in the different parts of the city. The agency reimburses the car travel expenses incurred by the managers in commuting of the work from their residence to the stores to which they are assigned. The basis of reimbursements is

A fixed sum of ₹ 300/- per month for repair and maintenance. A variable amount at the rate of ₹ 1.60 per Km. of travel incurred during the month. All stores work for 25 days in a month.

The distance in kilometers from a manager's residence to stores is displayed in the following table. Find the optimal assignment of managers to stores so that the monthly expenditure to be incurred by the agency on car travel of managers is minimum.

2) After having operated optimal assignment for some time, Mr. Vansh request the agency to assign him store S1, which is the closest to his residence, as he has been advised by to take long journeys. If the agency agrees to his request, how should the present assignment be changed and how much extra will it cost to the agency?

	Stores (Distance in Km.)				
Managers	S <sub>1</sub>	S <sub>2</sub>	S <sub>3</sub>	S <sub>4</sub>	S <sub>5</sub>
Vansh	4	10	12	18	17
Aayushi	7	16	16	22	18
Dhvani	8	6	9	19	21
Arth	11	12	15	12	13
Kripesh	9	14	19	18	14

The screenshot shows the RStudio environment with the following components:

- Source Editor:** Contains the R script for solving the assignment problem.
 

```
1 library(lpSolve)
2 assign.costs <- matrix(c(4,10,12,18,17,7,16,16,22,18,8,6,9,19,21,11,12,15,12,13,9,14,19,18,14), ncol=5, byrow=TRUE)
3 assign.costs
4 lp.assign(assign.costs)
5 lp.assign(assign.costs)$solution
6
```
- Console:** Shows the execution of the script.
 

```
R 4.3.1 ~ ./
> assign.costs
      [,2] [,3] [,4] [,5]
[1,]  4  10  12  18  17
[2,]  7  16  16  22  18
[3,]  8   6   9  19  21
[4,] 11  12  15  12  13
[5,]  9  14  19  18  14
> lp.assign(assign.costs)
Success: the objective function is 51
> lp.assign(assign.costs)$solution
      [,2] [,3] [,4] [,5]
[1,]  0  0  1  0  0
[2,]  1  0  0  0  0
[3,]  0  1  0  0  0
[4,]  0  0  0  1  0
[5,]  0  0  0  0  1
>
```
- Environment:** Lists the objects created: assign.costs (matrix), costs (matrix), lptrans (list), demand (matrix), demand.sigs (matrix), supply (matrix), and supply.sigs (matrix).
- Files:** Shows the project files.
- Plots:** Empty.
- Packages:** Shows the installed and updated packages.
- Help:** Provides documentation for the lpSolve package.
- Viewer:** Displays the output of the lpSolve package.
- Presentation:** Shows the presentation slides.