**Operations Research (Paper III)**

**MSc. (Computer Science) Semester III 2022-23**

# Index

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr No.** | **Date** | **Title** | **Page No.** | **Sign** |
| 1 |  | Graphical Method | 2 |  |
| 2 |  | Simplex Method for 2 variables | 5 |  |
| 3 |  | Simplex Method for 3 variables | 7 |  |
| 4 |  | Simplex Method for equality constraint | 9 |  |
| 5 |  | Big M Method | 11 |  |
| 6 |  | Resource Allocation problem | 13 |  |
| 7 |  | Infeasible solution | 15 |  |
| 8 |  | Dual Simplex Method | 17 |  |
| 9 |  | Transportation Method | 19 |  |
| 10 |  | Assignment Method | 21 |  |

## Practical 1

**Aim:** Use graphical method to solve the following LPP:

Max Z = 3x + 5y

w.r.t.

x + 2y ≤ 2000, x + y ≤ 1500, y ≤ 600, x, y ≥ 0

**Source Code:** require(lpSolve) C <- c(3, 5)

A <- matrix(c(1, 2,

1, 1,

0, 1), nrow = 3, byrow = T) B <- c(2000, 1500, 600) constraint\_direction <- c("<=", "<=", "<=")

plot.new()

plot.window(xlim = c(0, 2000), ylim = c(0, 2000)) axis(1) axis(2)

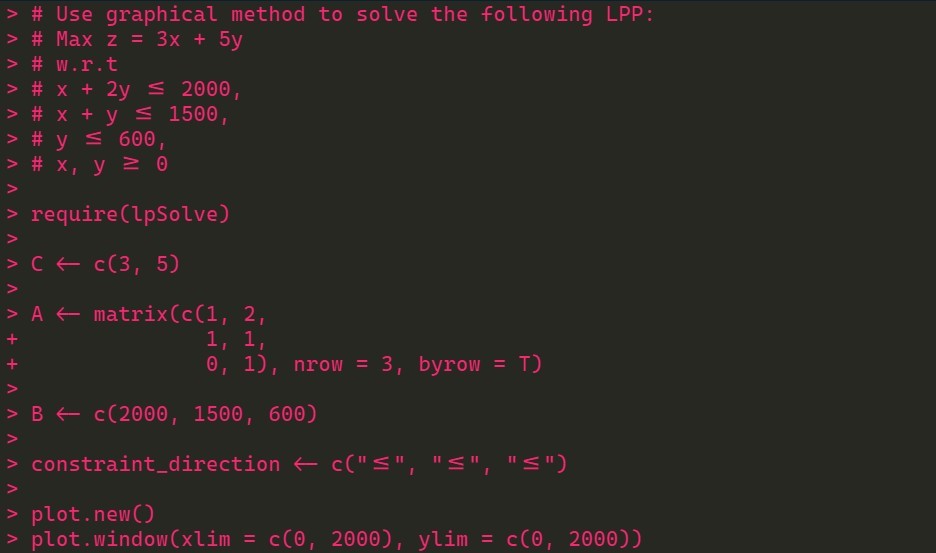
title(main = "LPP using graphical method", xlab = "X-axis", ylab = "Yaxis") box()

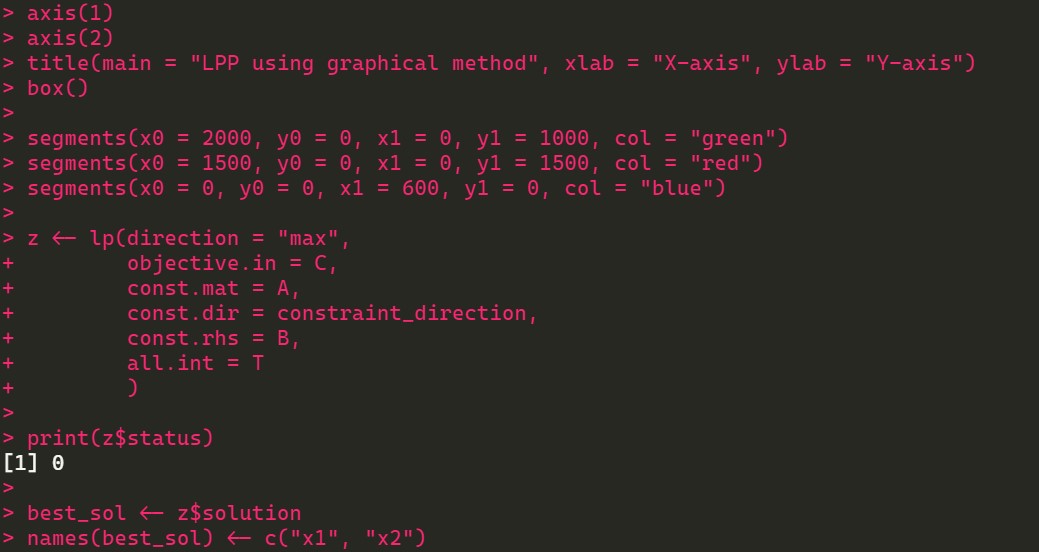
segments(x0 = 2000, y0 = 0, x1 = 0, y1 = 1000, col = "green") segments(x0 = 1500, y0 = 0, x1 = 0, y1 = 1500, col = "red") segments(x0 = 0, y0 = 0, x1 = 600, y1 = 0, col = "blue") z <- lp(direction = "max",

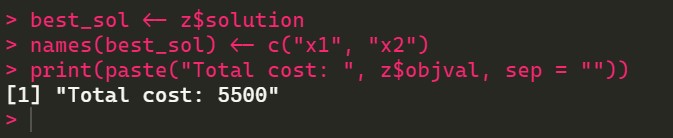
objective.in = C, const.mat = A, const.dir = constraint\_direction, const.rhs = B, all.int = T ) print(z$status)

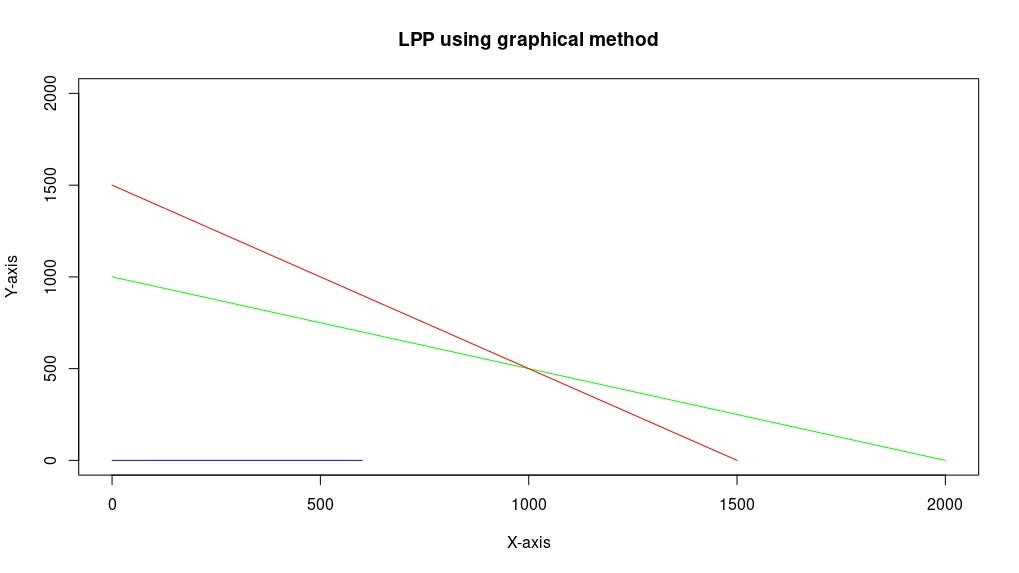
best\_sol <- z$solution names(best\_sol) <- c("x1", "x2") print(paste("Total cost: ", z$objval, sep = ""))

**Output:**









## Practical 2

Max Z = 3x + 2y

w.r.t.

x + y ≤ 4, x – y ≤ 2, x, y ≥ 0

**Source Code:**

from scipy.optimize import linprog

obj = [-3, -2] lhs\_ineq = [[1, 1], [1, -1]] In[1]:

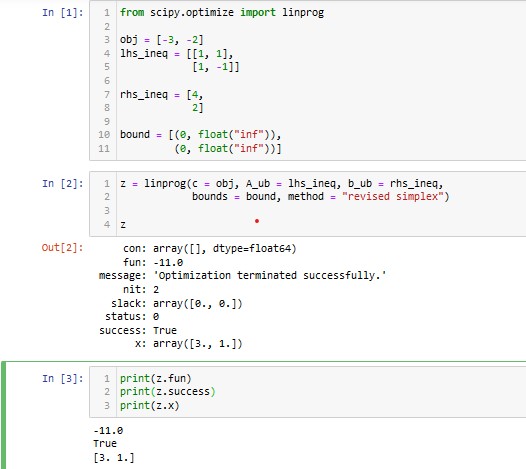
rhs\_ineq = [4, 2] bound = [(0, float("inf")), (0, float("inf"))]

z = linprog(c = obj, A\_ub = lhs\_ineq, b\_ub = rhs\_ineq, bounds = bound, method = "revised simplex") In[2]:

z

print(z.fun) In[3]:

print(z.success) print(z.x)



## Practical 3

Min Z = x1 – 3x2 + 2x3 w.r.t

3x1 – x2 + 3x3 ≤ 7,

-2x1 + 4x2 ≤ 12, -4x1 + 3x2 + 8x3 ≤ 10, x1, x2, x3 ≥ 0

**Source Code:**

from scipy.optimize import linprog obj = [1, -3, 2]

lhs\_ineq = [[3, -1, 3],

[-2, 4, 0],

[-4, 3, 8]]

In [1]:

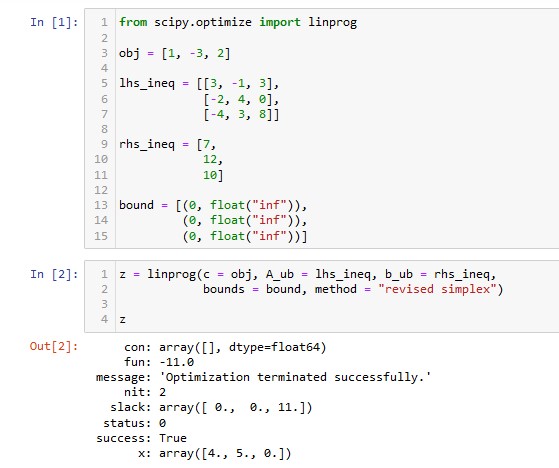
rhs\_ineq = [7,

12, 10]

bound = [(0, float("inf")),

(0, float("inf")), (0, float("inf"))]

z = linprog(c = obj, A\_ub = lhs\_ineq, b\_ub = rhs\_ineq, In [2]: bounds = bound, method = "revised simplex") z



## Practical 4

Max Z = x + 2y

w.r.t.

2x + y ≤ 20,

* 4x + 5y ≤ 10,
* x + 2y ≥ – 2,– x + 5y = 15, x, y ≥ 0

**Source code:**

from scipy.optimize import linprog obj = [-1, -2]

lhs\_ineq = [[2, 1],

[-4, 5], [1, -2]]

rhs\_ineq = [20, In [1]:

10,

2]

lhs\_eq = [[-1, 5]] rhs\_eq = [15]

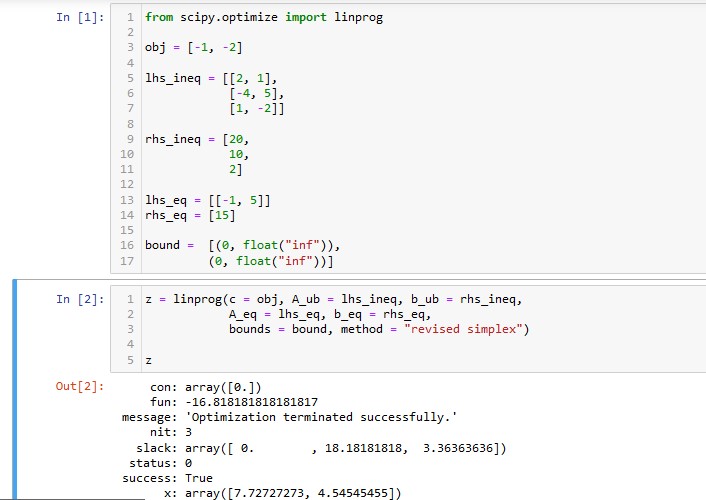
bound = [(0, float("inf")),

(0, float("inf"))]

z = linprog(c = obj, A\_ub = lhs\_ineq, b\_ub = rhs\_ineq,

A\_eq = lhs\_eq, b\_eq = rhs\_eq, In [2]:

bounds = bound, method = "revised simplex") z



## Practical 5

Use Big M method to solve the following LPP:

Min Z = 4x1 + x2

w.r.t.

3x1 + 4x2 ≥ 12, x1 + 5x2 ≥ 15, x1, x2 ≥ 0

**Source code:**

from scipy.optimize import linprog

obj = [4, 1] lhs\_ineq = [[-3, -4],

[-1, -5]]

In [1]:

rhs\_ineq = [-20,

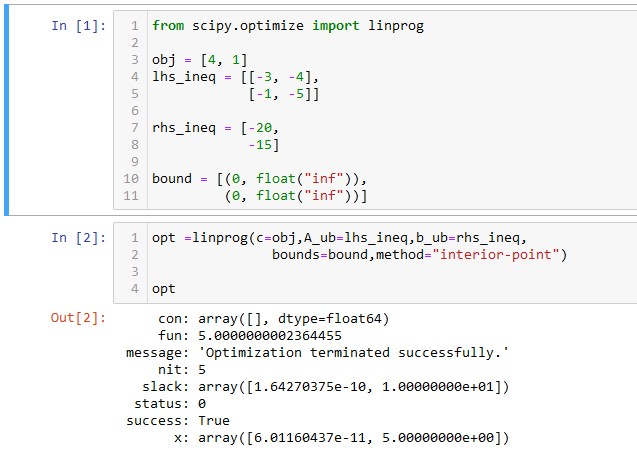
-15]

bound = [(0, float("inf")),

(0, float("inf"))]

opt = linprog(c=obj,A\_ub=lhs\_ineq,b\_ub=rhs\_ineq, bounds=bound,method="interior-point") In [2]:

opt



## Practical 6

Use any method to solve the following resource allocation problem:

Max Z = 20x1 + 12x2 + 50x3 + 25x4 ..................(profit) w.r.t. x1 + x2 + x3 + x4 ≤ 50 ........................................(manpower) 3x1 + 2x2 + x3 ≤ 100 ........................................(material A) x2 + 2x3 ≤ 90, ........................................(material B) x1, x2, x3 ≥ 0

**Source code:**

from scipy.optimize import linprog obj = [-20, -12, -40, -25]

lhs\_ineq = [[1, 1, 1, 1],

[3, 2, 1, 0], In [1]:

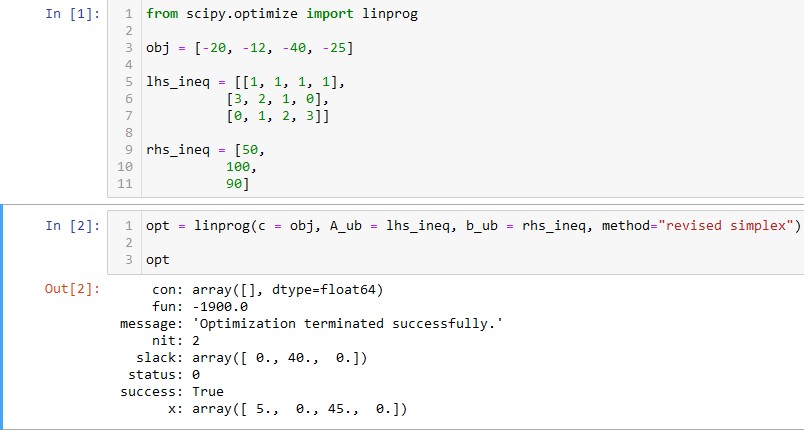
[0, 1, 2, 3]]

rhs\_ineq = [50,

100,

90]

opt = linprog(c = obj, A\_ub = lhs\_ineq, b\_ub = rhs\_ineq, In [2]: method="revised simplex") opt



## Practical 7

Use simplex method to solve the following LPP:

Max Z = 200x + 300y

w.r.t.

2x + 3y ≥ 1200, x + y ≤ 400, 2x + 1.5y ≥ 900, x, y ≥ 0

**Source code:**

from scipy.optimize import linprog obj = [-200, 300]

lhs\_ineq = [[-2, -3],

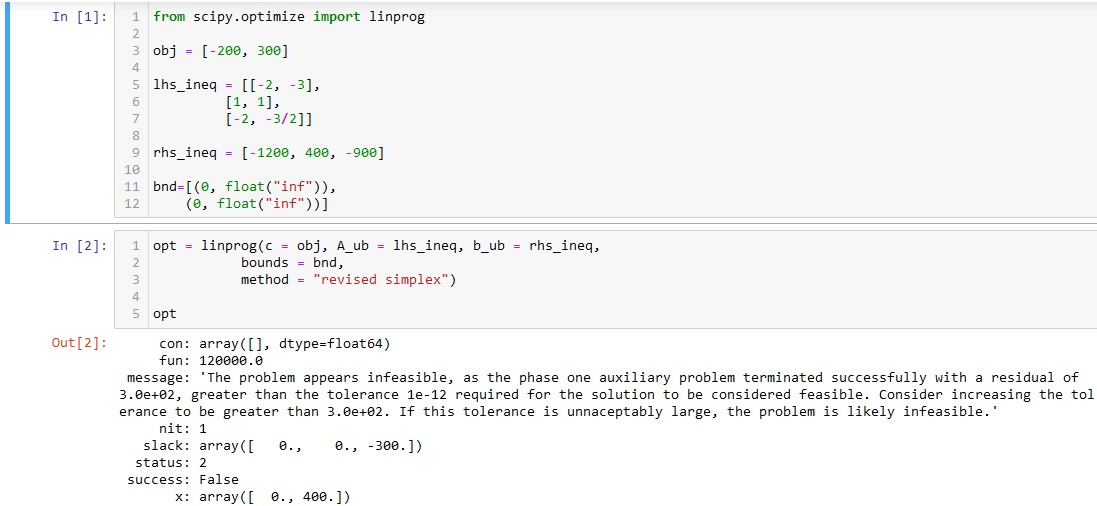
[1, 1],

In [1]: [-2, -3/2]] rhs\_ineq = [-1200, 400, -900]

bnd=[(0, float("inf")), (0, float("inf"))]

opt = linprog(c = obj, A\_ub = lhs\_ineq, b\_ub = rhs\_ineq, bounds = bnd, In [2]:

method = "revised simplex") opt



## Practical 8

Use dual simplex method to solve the following LPP:

Max Z = 40x1 + 50x2

w.r.t.

2x1 + 3x2 ≤ 3, 8x1 + 4x2 ≤ 5, x1, x2 ≥ 0

**Source code:**

require(lpSolve)

f.obj <- c(40, 50)

f.con <- matrix(c(2, 3,

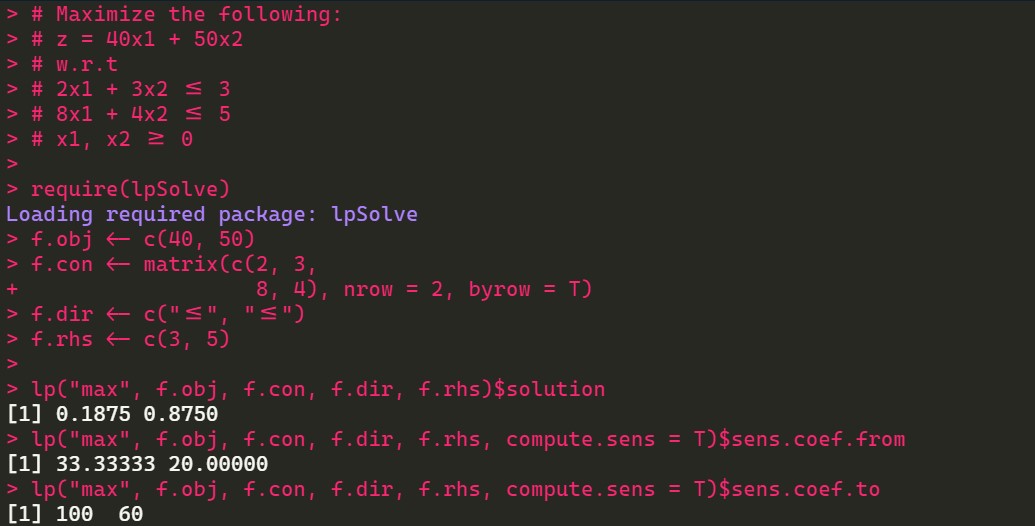
8, 4), nrow = 2, byrow = T)

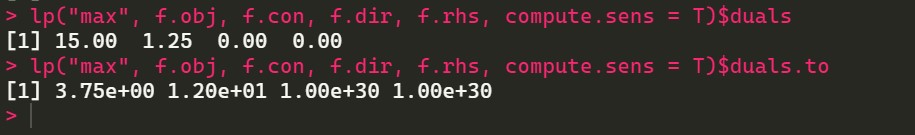
f.dir <- c("<=", "<=")

f.rhs <- c(3, 5)

lp("max", f.obj, f.con, f.dir, f.rhs)$solution

lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens = T)$sens.coef.from lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens = T)$sens.coef.to lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens = T)$duals lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens = T)$duals.to





## Practical 9

Solve following transportation problem in which each cell represents

unit costs:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | **Customers** | |  | **Supply** |
| **1** | **2** | **3** | **4** |
| **Suppliers** | 10 | 2 | 20 | 11 | 15 |
| 12 | 7 | 9 | 20 | 25 |
| 4 | 14 | 16 | 18 | 10 |
| **Demand** | 5 | 15 | 15 | 15 |  |

**Source code:**

library(lpSolve)

cost <- matrix(c(10, 2, 20, 11,

12, 7, 9, 20,

4, 14, 16, 18), nrow = 3, byrow = T)

colnames(cost) <- c("Customer 1", "Customer 2", "Customer 3", "Customer

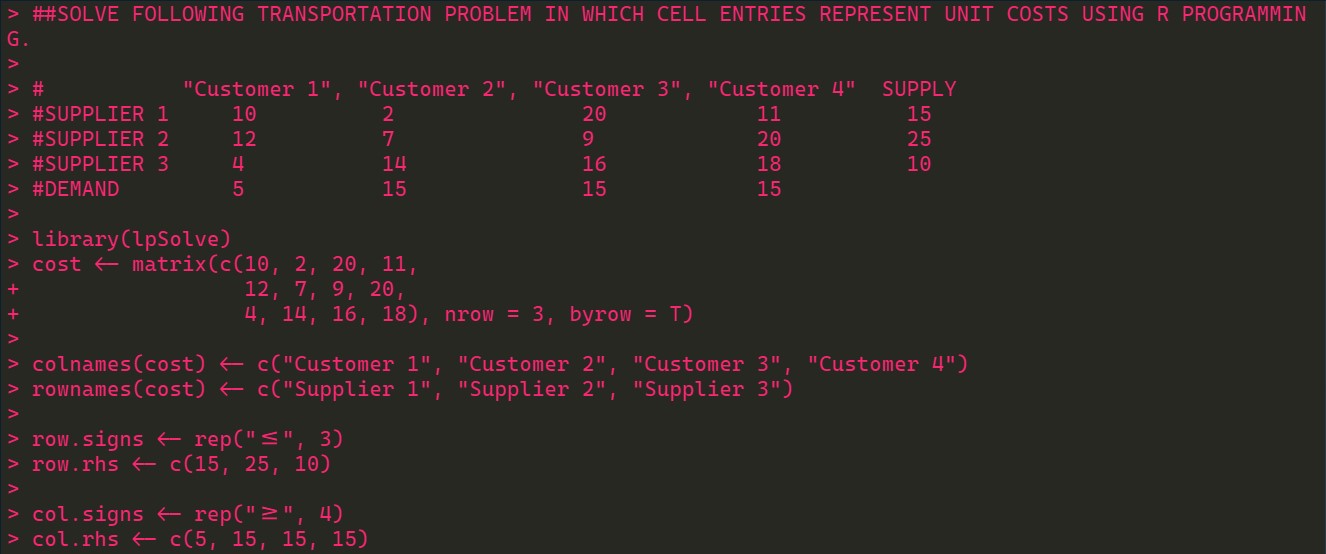
4")

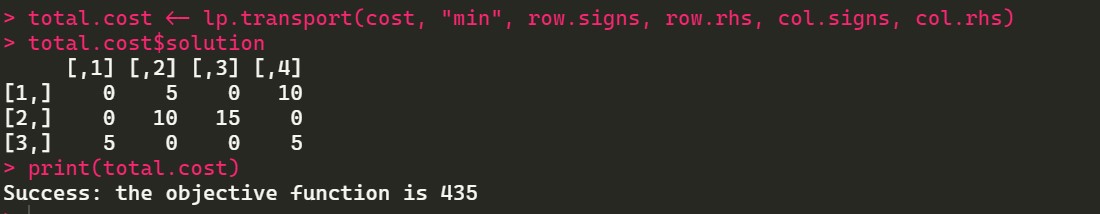
rownames(cost) <- c("Supplier 1", "Supplier 2", "Supplier 3")

row.signs <- rep("<=", 3) row.rhs <- c(15, 25, 10)

col.signs <- rep("<=", 4) col.rhs <- c(5, 15, 15, 15)

total.cost <- lp.transport(cost, "min", row.signs, row.rhs, col.signs, col.rhs) total.cost$solution print(total.cost)





## Practical 10

Solve following assignment problem represented in this matrix:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | **Jobs** |  |
| **1** | **2** | **3** |
| **Workers** | **1** | 15 | 10 | 9 |
| **2** | 9 | 15 | 10 |
| **3** | 10 | 12 | 8 |

**Source Code:** library(lpSolve)

cost <- matrix(c(15, 10, 9, 9, 15, 10,

10, 12, 8), nrow = 3, byrow = T)

cost

answer <- lp.assign(cost) answer$solution

**Output:**

