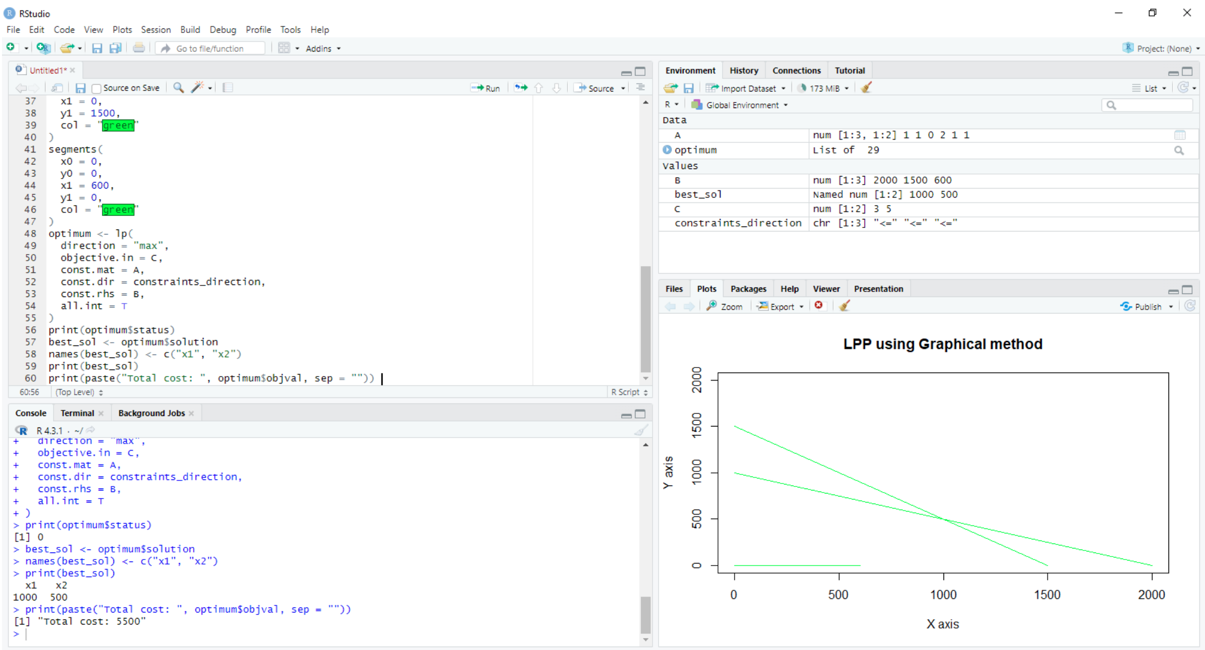
Practical 1: GRAPHICAL METHOD USING R PROGRAMMING

Code:

```
#PRACTICAL 1: GRAPHICAL METHOD USING R PROGRAMMING
#Find a geometrical interpretation and solution as well for the following LP
problem
#Max z= 3x1 + 5x2
#subject to constraints:
#x1+2x2<=2000
#x1+x2<=1500
#x2<=600
#x1,x2>=0
#To solve linear programming using R studio, we need to install lpsolve
package

install.packages("lpSolve")
require(lpSolve)
C <- c(3, 5)
A <- matrix(c(1, 2,
              1, 1,
              0, 1), nrow = 3, byrow = TRUE)
B <- c(2000, 1500, 600)
constraints_direction <- c("<=", "<=", "<=")
plot.new()
plot.window(xlim = c(0, 2000), ylim = c(0, 2000))
axis(1)
axis(2)
title(main = "LPP using Graphical method")
title(xlab = "X axis")
title(ylab = "Y axis")
box()
segments(
  x0 = 2000,
  y0 = 0,
  x1 = 0,
  y1 = 1000,
  col = "green"
)
segments(
  x0 = 1500,
  y0 = 0,
  x1 = 0,
  y1 = 1500,
  col = "green"
)
segments(
  x0 = 0,
```

```
  y0 = 0,
  x1 = 600,
  y1 = 0,
  col = "green"
)
optimum <- lp(
  direction = "max",
  objective.in = C,
  const.mat = A,
  const.dir = constraints_direction,
  const.rhs = B,
  all.int = T
)
print(optimum$status)
best_sol <- optimum$solution
names(best_sol) <- c("x1", "x2")
print(best_sol)
print(paste("Total cost: ", optimum$objval, sep = ""))
```

OUTPUT



Practical 2: Simplex Method (2 Variables)

Max z = 3x1 + 2x2
Subject to:
x1 + x2 <= 4
x1 - x2 <= 2
x1, x2 >= 0

In [1]:
```
from scipy.optimize import linprog
```

In [2]:
```
obj  = [-3, -2]
```

In [3]:
```
lhs_ineq = [[1,  1], #Red constraint left side
            [1, -1]]  #Blue constraint left side
```

In [4]:
```
rhs_ineq = [4, #Red constraint right side
             2] #Blue constraint right side
```

In [5]:
```
bnd = [(0, float("inf")), #Bounds of x
      (0, float("inf"))]  #Bounds of y
```

In [6]:
```
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
              bounds=bnd, method="revised simplex")
opt
```

```
C:\Users\asif0\AppData\Local\Temp\ipykernel_9212\3460113553.py:1: DeprecationWarning: `
method='revised simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use
one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq,
```

Out[6]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -11.0
       x: [ 3.000e+00  1.000e+00]
     nit: 2
```

In [7]:
```
opt.fun
```

Out[7]:
```
-11.0
```

In [8]:
```
opt.success
```

Out[8]:
```
True
```

In [9]:
```
opt.x
```

Out[9]:
```
array([3., 1.])
```

Practical 3: Simplex Method (3 Variables)

Min z = x1 - 3x2 + 2x3
Subject to:
3z1 - x2 + 3x3 <= 7
-2x1 + 4x2 <= 12
-4x1 + 3x2 + 8x3 <= 10
x1,x2,x3 >= 0

In [ ]:
```
from scipy.optimize import linprog
```

In [ ]:
```
obj = [1,-3,2]
```

In [ ]:
```
lhs_ineq = [[3, -1, 3], #Red constraint left side
            [-2, 4, 0],  #Blue constraint left side
            [-4, 3, 8]]  #Yellow constraint left side
```

In [ ]:
```
rhs_ineq = [7, #Red constraint right side
            12, #Blue constraint right side
            10] #Yellow constraint right side
```

In [ ]:
```
bnd = [(0, float("inf")), #Bounds of x
      (0, float("inf")),
      (0, float("inf"))] #Bounds of y
```

In [ ]:
```
opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq, bounds = bnd,
              method = "revised simplex")
opt
```

```
<ipython-input-6-b4277e38080e>:1: DeprecationWarning: `method='revised simplex'` is depre
cated and will be removed in SciPy 1.11.0. Please use one of the HiGHS solvers (e.g. `met
hod='highs'`) in new code.
  opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq, bounds = bnd, method = "revise
d simplex")
```

Out[6]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -11.0
       x: [ 4.000e+00  5.000e+00  0.000e+00]
     nit: 2
```

Practical 4: Simplex Method with Equality Constraints

Max z = x + 2y

Subject to:

2x + y <= 20

-4x + 5y <= 10

-x + 2y >= -2

-x + 5y = 15

x,y >= 0

In [1]:
```
from scipy.optimize import linprog
```

In [2]:
```
obj = [-1, -2]
```

In [3]:
```
lhs_ineq = [[2, 1], #Red constraint left side
            [-4, 5], #Blue constraint left side
            [1 ,-2]] #Yellow constraint left side
```

In [4]:
```
rhs_ineq = [20, #Red constraint right side
            10,  #Blue constraint right side
            2]   #Yellow constraint right side
```

In [5]:
```
lhs_eq = [[-1, 5]] #Green constraint left side
rhs_eq = [15]      #Green constraint right side
```

In [6]:
```
bnd = [(0, float("inf")),# Bounds of x
       (0, float("inf"))] # Bounds of y
```

In [7]:
```
opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
             A_eq = lhs_eq, b_eq = rhs_eq, bounds = bnd,
             method = "revised simplex")
opt
```

```
C:\Users\asif0\AppData\Local\Temp\ipykernel_9416\4111130241.py:1: DeprecationWarning: `
method='revised simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use
one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
```

Out[7]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -16.818181818181817
       x: [ 7.727e+00  4.545e+00]
     nit: 3
```

In [ ]:

Practical 5: Big M Simplex Method

Min z = 4x1 + x2
Subject to:
3x1 + 4x2 >= 20
x1 + 5x2 >= 15
x1,x2 >= 0

In [ ]:
```
from scipy.optimize import linprog
```

In [ ]:
```
obj = [4,1]
```

In [ ]:
```
lhs_ineq = [[ -3, -4], #Left side of first constraint
            [-1, -5]]  #Right side of first constraint
```

In [ ]:
```
rhs_ineq = [-20, #Right side of first constraint
            -15]  #Right side of second constraint
```

In [ ]:
```
bnd = [(0, float("inf")), # Bounds of x1
       (0, float("inf"))] # Bounds of x2
```

In [ ]:
```
opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
              bounds = bnd, method = "interior-point")
opt #method = "interior-point" solves linear programming problem using default
#simplex method.
```

```
<ipython-input-6-46aec19352fa>:1: DeprecationWarning: `method='interior-point'` is deprec
ated and will be removed in SciPy 1.11.0. Please use one of the HiGHS solvers (e.g. `meth
od='highs'`) in new code.
  opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
```

Out[6]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: 5.000000000236444
       x: [ 6.012e-11  5.000e+00]
     nit: 5
```

PRACTICAL 6: RESOURCE ALLOCATION PROBLEM BY SIMPLEX METHOD

Use SciPy to solve the resource allocation problem stated as follows:

Max z= 20x1 + 12x2 +40x3 + 25x4 (profit)
subjected to:
x1 + x2 + x3 + x4 <= 50 (manpower)
3x1 + 2x2 + x3 <= 100 (material A)
x2 + 2x3 <= 90 (material B)
x1, x2, x3, x4 >= 0

In [ ]:
```
from scipy.optimize import linprog
obj = [-20, -12, -40, -25] #profit objective function
```

In [ ]:
```
lhs_ineq = [[1, 1, 1, 1,], #Manpower
           [3, 2, 1, 0],   #Material A
           [0, 1, 2, 3]]   #Material B
```

In [ ]:
```
rhs_ineq = [[50, #Manpower
            100, #Material A
            90]] #Material B
```

In [ ]:
```
opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
              method = "revised simplex")
```

```
C:\Users\asif0\AppData\Local\Temp\ipykernel_248\677215074.py:1: DeprecationWarning: `meth
od='revised simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use one o
f the HiGHS solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
```

In [ ]:
```
opt
```

Out[8]:
```
 message: Optimization terminated successfully.
 success: True
  status: 0
     fun: -1900.0
       x: [ 5.000e+00  0.000e+00  4.500e+01  0.000e+00]
     nit: 2
```

Practical 7: Infeasibility in the Simplex Method

Solve the following linear programming problem using Simplex Method:

WHILE SOLVING LINEAR PROGRAMMING PROBLEM USING SIMPLEX METHOD, IF ONE OR MORE ARTIFICIAL VARIABLES REMAIN IN THE BASIS AT POSITIVE LEVEL AT THE END OF PHASE 1 COMPUTATION , THE PROBLEM HAS NO FEASIBLE SOLUTION(INFEASIBLE SOLUTION).

Max z = 200x - 300y
subject to constraints
2x + 3y >= 1200
x + y <= 400
2x+ 3/2y >= 900
x, y >= 0

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

In [ ]:
```
lhs_ineq = [[-2, -3],  #Red constraint left side
            [1, 1],     #Blue constraint left side
            [-2, -1.5]] #Yellow constraint left side
```

In [ ]:
```
rhs_ineq = [-1200, #Red constraint right side
            400,    #Blue constraint right side
            -900,]  #Yellow constraint right side
```

In [ ]:
```
bnd = [(0, float("inf")), #Bounds of x
      (0, float("inf"))]  #Bounds of y
```

In [ ]:
```
opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
              bounds = bnd,
              method = "revised simplex")
```

```
C:\Users\asif0\AppData\Local\Temp\ipykernel_6952\1416332328.py:1: DeprecationWarning: `
method='revised simplex'` is deprecated and will be removed in SciPy 1.11.0. Please use
one of the HiGHS solvers (e.g. `method='highs'`) in new code.
  opt = linprog(c = obj, A_ub = lhs_ineq, b_ub = rhs_ineq,
```

In [ ]:
```
opt
```

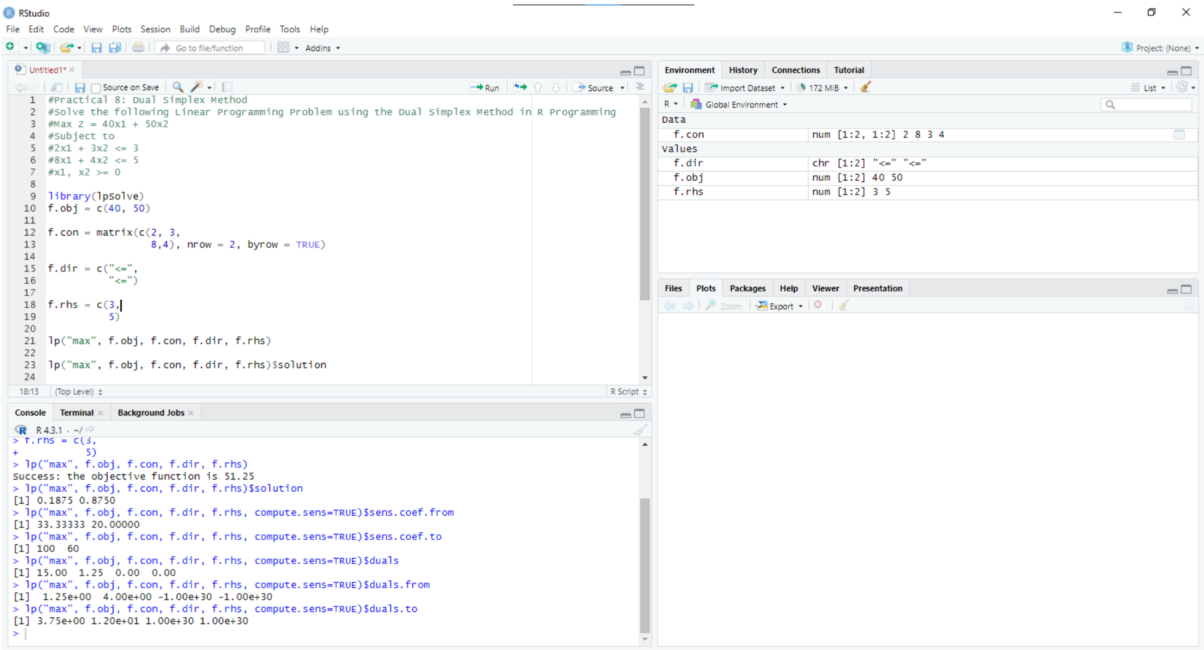
Out[30]:
```
 message: The problem appears infeasible, as the phase one auxiliary problem terminated
successfully with a residual of 3.0e+02, greater than the tolerance 1e-12 required for
the solution to be considered feasible. Consider increasing the tolerance to be greater
than 3.0e+02. If this tolerance is unnaceptably large, the problem is likely infeasibl
e.
 success: False
  status: 2
     fun: 120000.0
       x: [ 0.000e+00  4.000e+02]
     nit: 1
```

Practical 8: Dual Simplex Method

Code:

```
Practical 8: Dual Simplex Method

#Solve the following Linear Programming Problem using the Dual Simplex Method
in R Programming
#Max Z = 40x1 + 50x2
#Subject to
#2x1 + 3x2 <= 3
#8x1 + 4x2 <= 5
#x1, x2 >= 0

library(lpSolve)
f.obj = c(40, 50)

f.con = matrix(c(2, 3,
                 8,4), nrow = 2, byrow = TRUE)

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

f.rhs = c(3,
          5)

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

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

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

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

lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals

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

lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals.to
```

OUTPUT

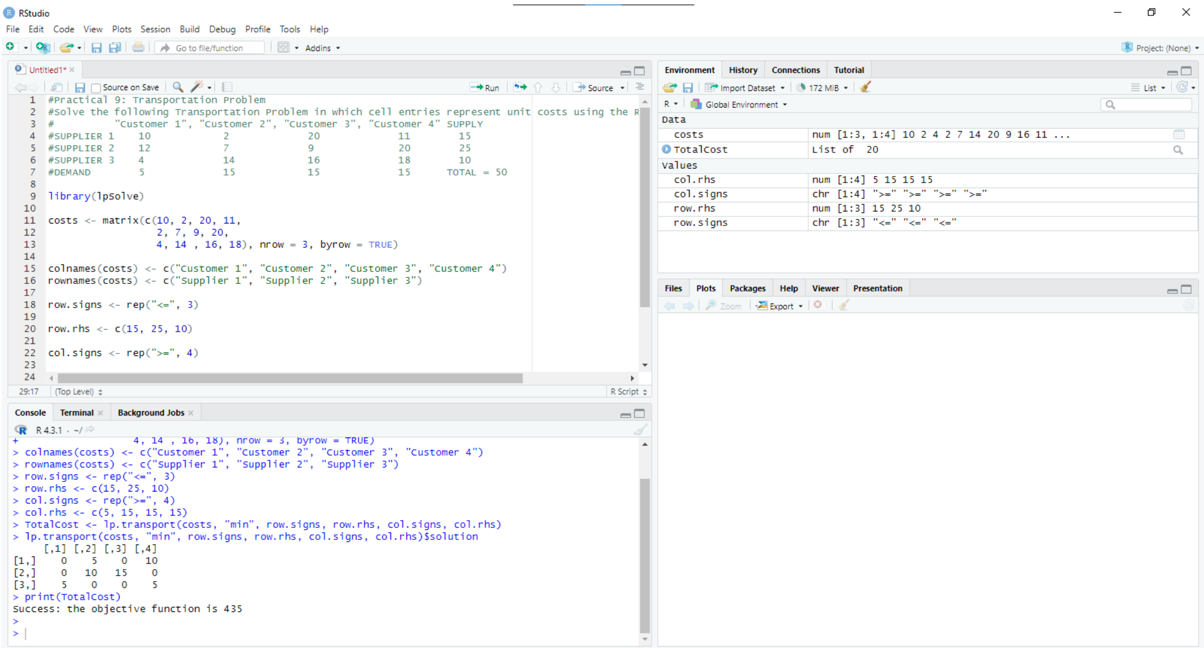

RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1*
#Practical 8: Dual Simplex Method
#Solve the following Linear Programming Problem using the Dual Simplex Method in R Programming
#Max Z = 40x1 + 50x2
#Subject to
#2x1 + 3x2 <= 3
#8x1 + 4x2 <= 5
#x1, x2 >= 0
library(lpSolve)
f.obj = c(40, 50)
f.con = matrix(c(2, 3,
8,4), nrow = 2, byrow = TRUE)
f.dir = c("<=",
"<=")
f.rhs = c(3,
5)
lp("max", f.obj, f.con, f.dir, f.rhs)
lp("max", f.obj, f.con, f.dir, f.rhs)$solution
Console Terminal Background Jobs
R 4.3.1 · ~/
> f.rhs = c(3,
+ 5)
> lp("max", f.obj, f.con, f.dir, f.rhs)
Success: the objective function is 51.25
> lp("max", f.obj, f.con, f.dir, f.rhs)$solution
[1] 0.1875 0.8750
> lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$sens.coef.from
[1] 33.33333 20.00000
> lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$sens.coef.to
[1] 100 60
> lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals
[1] 15.00 1.25 0.00 0.00
> lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals.from
[1] 1.25e+00 4.00e+00 -1.00e+30 -1.00e+30
> lp("max", f.obj, f.con, f.dir, f.rhs, compute.sens=TRUE)$duals.to
[1] 3.75e+00 1.20e+01 1.00e+30 1.00e+30
Environment History Connections Tutorial
Global Environment
Data
f.con num [1:2, 1:2] 2 8 3 4
Values
f.dir chr [1:2] "<=" "<="
f.obj num [1:2] 40 50
f.rhs num [1:2] 3 5
Files Plots Packages Help Viewer Presentation

Practical 9: Transportation Problem

Code:

```
#Practical 9: Transportation Problem
#Solve the following Transportation Problem in which cell entries represent
unit costs using the R programming language
#          "Customer 1", "Customer 2", "Customer 3", "Customer 4" SUPPLY
#SUPPLIER 1    10            2             20             11         15
#SUPPLIER 2    12            7             9              20         25
#SUPPLIER 3    4             14            16             18         10
#DEMAND        5             15            15             15      TOTAL = 50

library(lpSolve)

costs <- matrix(c(10, 2, 20, 11,
                  2, 7, 9, 20,
                  4, 14 , 16, 18), nrow = 3, byrow = TRUE)

colnames(costs) <- c("Customer 1", "Customer 2", "Customer 3", "Customer 4")
rownames(costs) <- 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)

TotalCost <- lp.transport(costs, "min", row.signs, row.rhs, col.signs,
col.rhs)

lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)$solution
print(TotalCost)
```

OUTPUT



RStudio
#Practical 9: Transportation Problem
#Solve the following Transportation Problem in which cell entries represent unit costs using the R
"Customer 1", "Customer 2", "Customer 3", "Customer 4" SUPPLY
#SUPPLIER 1   10   2   20   11   15
#SUPPLIER 2   12   7   9   20   25
#SUPPLIER 3   4   14   16   18   10
#DEMAND   5   15   15   15   TOTAL = 50
library(lpSolve)
costs <- matrix(c(10, 2, 20, 11,
2, 7, 9, 20,
4, 14 , 16, 18), nrow = 3, byrow = TRUE)
colnames(costs) <- c("Customer 1", "Customer 2", "Customer 3", "Customer 4")
rownames(costs) <- 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)
> TotalCost <- lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)
> lp.transport(costs, "min", row.signs, row.rhs, col.signs, col.rhs)$solution
[,1] [,2] [,3] [,4]
[1,] 0 5 0 10
[2,] 0 10 15 0
[3,] 5 0 0 5
> print(TotalCost)
Success: the objective function is 435
costs   num [1:3, 1:4] 10 2 4 2 7 14 20 9 16 11 ...
TotalCost   List of 20
col.rhs   num [1:4] 5 15 15 15
col.signs   chr [1:4] ">=" ">=" ">=" ">="
row.rhs   num [1:3] 15 25 10
row.signs   chr [1:3] "<=" "<=" "<="

Practical 10: Assignment Problem

Code:

```
#Practical 10: Assignment Problem
#Solve the following Assignment Problem represented in the following matrix
using R Programming
#   JOB_1 JOB_2 JOB_3
#W1   15    10     9
#W2   9     15     10
#W3   10    12     8

library(lpSolve)

costs <- matrix(c(15, 10, 9,
                  9, 15, 10,
                  10, 12 ,8), nrow = 3, byrow = TRUE)

costs

lp.assign(costs)

lp.assign(costs)$solution
```

OUTPUT