QMM_Assignment5

Maddukuri Janakisrija

11/08/2021

```
library(lpSolveAPI)  # loading package- lpSolveAPI
library(Benchmarking)  # loading package- Benchmarking
```

```
## Loading required package: ucminf
```

```
## Loading required package: quadprog
```

```
library(cowplot)
```

```
dmu1 <- read.lp("dmu1.lp")
dmu1
```

```
## Model name: 
##                a1     a2     b1     b2          
## Maximize  14000   3500      0      0          
## R1        14000   3500   -150   -0.2  <=  0
## R2        14000  21000   -400   -0.7  <=  0
## R3        42000  10500   -320   -1.2  <=  0
## R4        28000  43000   -520     -2  <=  0
## R5        19000  25000   -350   -1.2  <=  0
## R6        14000  15000   -320   -0.7  <=  0
## R7            0      0    150    0.2   =  1
## Kind        Std    Std    Std    Std          
## Type       Real   Real   Real   Real          
## Upper       Inf    Inf    Inf    Inf          
## Lower         0      0      0      0          
```

```
solve(dmu1)
```

```
## [1] 0
```

```
get.objective(dmu1)
```

```
## [1] 1
```

```
get.variables(dmu1)
```

```
## [1] 7.142857e-05 0.000000e+00 5.172414e-03 1.120690e+00
```

The efficiency of the dmu1 is 1 which is maximum

The outputs of the lp are 5.17241 and 1.12069

The inputs of the lp are 7.14285 and 0.00 for maximum efficiency.

```
dmu2 <- read.lp ("dmu2.lp")
dmu2
```

```
## Model name:
##               a1     a2     b1     b2
## Maximize  14000  21000      0      0
## R1        14000   3500   -150   -0.2  <=  0
## R2        14000  21000   -400   -0.7  <=  0
## R3        42000  10500   -320   -1.2  <=  0
## R4        28000  43000   -520     -2  <=  0
## R5        19000  25000   -350   -1.2  <=  0
## R6        14000  15000   -320   -0.7  <=  0
## R7            0      0    400    0.7   =  1
## Kind        Std    Std    Std    Std
## Type       Real   Real   Real   Real
## Upper       Inf    Inf    Inf    Inf
## Lower         0      0      0      0
```

```
solve(dmu2)
```

```
## [1] 0
```

```
get.objective(dmu2)
```

```
## [1] 1
```

```
get.variables(dmu2)
```

```
## [1] 0.000000e+00 4.761905e-05 1.299694e-03 6.858890e-01
```

The efficiency of the dmu2 is 1 which is maximum

The outputs of the lp are 1.29969 and 6.85889

The inputs of the lp are 0.00 and 4.76190 for maximum efficiency.

```
dmu3 <- read.lp("dmu3.lp")
dmu3
```

```
## Model name:
##               a1     a2     b1     b2
## Maximize  42000  10500      0      0
## R1        14000   3500   -150   -0.2  <=  0
## R2        14000  21000   -400   -0.7  <=  0
## R3        42000  10500   -320   -1.2  <=  0
## R4        28000  43000   -520     -2  <=  0
## R5        19000  25000   -350   -1.2  <=  0
## R6        14000  15000   -320   -0.7  <=  0
## R7            0      0    320    1.2   =  1
## Kind        Std    Std    Std    Std
## Type       Real   Real   Real   Real
## Upper       Inf    Inf    Inf    Inf
## Lower         0      0      0      0
```

```
solve(dmu3)
```

```
## [1] 0
```

```
get.objective(dmu3)
```

```
## [1] 1
```

```
get.variables(dmu3)
```

```
## [1] 2.380952e-05 0.000000e+00 1.724138e-03 3.735632e-01
```

The efficiency of the dmu3 is 1 which is maximum

The outputs of the lp are 1.72413 and 3.73563

The inputs of the lp are 2.38095 and 0.000 for maximum efficiency.

```
dmu4 <- read.lp("dmu4.lp")
dmu4
```

```
## Model name:
##               a1     a2     b1     b2    na2
## Maximize  28000  42000      0      0      0
## R1        14000   3500   -150   -0.2      0  <=  0
## R2        14000  21000   -400   -0.7      0  <=  0
## R3        42000  10500   -320   -1.2      0  <=  0
## R4        28000  43000   -520     -2      0  <=  0
## R5        19000  25000   -350   -1.2      0  <=  0
## R6        14000      0   -320   -0.7  15000  <=  0
## R7            0      0    520      2      0   =  1
## Kind        Std    Std    Std    Std    Std
## Type       Real   Real   Real   Real   Real
## Upper       Inf    Inf    Inf    Inf    Inf
## Lower         0      0      0      0      0
```

```
solve(dmu4)
```

```
## [1] 0
```

```
get.objective(dmu4)
```

```
## [1] 0.9836182
```

```
get.variables(dmu4)
```

```
## [1] 1.055657e-05 1.638177e-05 1.923077e-03 0.000000e+00 0.000000e+00
```

The efficiency of the dmu4 is 1 which is maximum

The outputs of the lp are 1.92307 and 0.000

The inputs of the lp are 1.05565 and

1.63817 for maximum efficiency.

```
dmu5 <- read.lp("dmu5.lp")
dmu5
```

```
## Model name:
##                a1     a2     b1     b2
## Maximize  19000  25000      0      0
## R1        14000   3500   -150   -0.2  <=  0
## R2        14000  21000   -400   -0.7  <=  0
## R3        42000  10500   -320   -1.2  <=  0
## R4        28000  43000   -520     -2  <=  0
## R5        19000  25000   -350   -1.2  <=  0
## R6        14000  15000   -320   -0.7  <=  0
## R7            0      0    320    0.7   =  1
## Kind        Std    Std    Std    Std
## Type       Real   Real   Real   Real
## Upper       Inf    Inf    Inf    Inf
## Lower         0      0      0      0
```

```
solve(dmu5)
```

```
## [1] 0
```

```
get.objective(dmu5)
```

```
## [1] 1.367534
```

```
get.variables(dmu5)
```

```
## [1] 1.590217e-05 4.261572e-05 1.469508e-03 7.567965e-01
```

The efficiency of the dmu5 is 1 which is maximum

The outputs of the lp are 1.46950 and

7.56796

The inputs of the lp are 1.59021 and 4.26157 for maximum efficiency.

```
dmu6 <- read.lp("dmu6.lp")
dmu6
```

```
## Model name:
##               a1     a2     b1     b2
## Maximize  14000  15000      0      0
## R1        14000   3500   -150   -0.2  <=  0
## R2        14000  21000   -400   -0.7  <=  0
## R3        42000  10500   -320   -1.2  <=  0
## R4        28000  43000   -520     -2  <=  0
## R5        19000  25000   -350   -1.2  <=  0
## R6        14000  15000   -320   -0.7  <=  0
## R7            0      0    320    0.7   =  1
## Kind        Std    Std    Std    Std
## Type       Real   Real   Real   Real
## Upper       Inf    Inf    Inf    Inf
## Lower         0      0      0      0
```

```
solve(dmu6)
```

```
## [1] 0
```

```
get.objective(dmu6)
```

```
## [1] 0.8618663
```

```
get.variables(dmu6)
```

```
## [1] 1.590217e-05 4.261572e-05 1.469508e-03 7.567965e-01
```

The efficiency of the dmu6 is 1 which is

maximum

The outputs of the lp are 1.46950 and 7.56796

The inputs of the lp are 1.59021 and 4.26157 for maximum efficiency.

dmu is not efficient even with the greatest weights.

The inputs and outputs are defined as vector and 2 inputs which are staff hours and supplies and 2 outputs which are Reimbursed Patient_days and Privately Paid Patient_Day.

```
a <- matrix(c(150, 400, 320, 520, 350, 320, 0.2, 0.7, 1.2, 2.0, 1.2, 0.7), ncol = 2)
b <- matrix(c(14000,14000,42000,28000,19000,14000,3500,21000,10500,42000,25000,15000)
,ncol = 2)
colnames(a) <- c("Staff_Hours", "Supplies")
colnames(b) <- c("Reimbursed Patient_Days","Privately Paid Patient_Days")
print(a)
```

```
##      Staff_Hours Supplies
## [1,]         150      0.2
## [2,]         400      0.7
## [3,]         320      1.2
## [4,]         520      2.0
## [5,]         350      1.2
## [6,]         320      0.7
```

```
print(b)
```

```
##      Reimbursed Patient_Days Privately Paid Patient_Days
## [1,]                   14000                        3500
## [2,]                   14000                       21000
## [3,]                   42000                       10500
## [4,]                   28000                       42000
## [5,]                   19000                       25000
## [6,]                   14000                       15000
```

```
Matrix<- cbind(a,b)
row.names(Matrix) = c("Facility1", "Facility2", "Facility3", "Facility4", "Facility5"
, "Facility6")
Matrix
```

```
##           Staff_Hours Supplies Reimbursed Patient_Days
## Facility1         150      0.2                   14000
## Facility2         400      0.7                   14000
## Facility3         320      1.2                   42000
## Facility4         520      2.0                   28000
## Facility5         350      1.2                   19000
## Facility6         320      0.7                   14000
##           Privately Paid Patient_Days
## Facility1                        3500
## Facility2                       21000
## Facility3                       10500
## Facility4                       42000
## Facility5                       25000
## Facility6                       15000
```

1) Formulate and perform DEA analysis under all DEA assumptions of FDH, CRS, VRS, IRS, DRS, and FRH.

2) Determine the Peers and Lambdas under each of the above assumptions

```
#Now, we are going to formulate and compute the DEA  analysis using FDH.

 #The free disposability assumption stipulates that we can freely discard unnecessary inputs and unwanted outputs

FDH <- dea(a,b, RTS = "fdh")
FDH
```

```
## [1] 1 1 1 1 1 1
```

```
peers(FDH)
```

```
##      peer1
## [1,]     1
## [2,]     2
## [3,]     3
## [4,]     4
## [5,]     5
## [6,]     6
```

```
FDH_Weights <- lambda(FDH)
```

The value of the peer for each facility is always the same as the peer.

Now, we are going to formulate and compute the DEA analysis using CRS

#Constant returns to scale (CRS) is one of the scaling assumptions. This assumption stipulates that if any possible production combination can arbitrarily be scaled up or down.

```
CRS <- dea(a,b, RTS = "crs")
CRS
```

```
## [1] 1.0000 1.0000 1.0000 1.0000 0.9775 0.8675
```

```
#Identify Peers
peers(CRS)
```

```
##      peer1 peer2 peer3
## [1,]     1    NA    NA
## [2,]     2    NA    NA
## [3,]     3    NA    NA
## [4,]     4    NA    NA
## [5,]     1     2     4
## [6,]     1     2     4
```

```
#Identify lambda
CRS_Weights <- lambda(CRS)
```

The results show DMU has 1 which is maximum efficiency for 1,2,3,4 and DMU 5 is 0.9775, DMU 6 0.8675
The peer for 5 and 6 are 1,2,4

Now, we are going to formulate and compute the DEA analysis using VRS.

VRS is one of the scaling assumptions. This Assumptions represents that variable return to scale

```
VRS <- dea(a,b, RTS = "vrs")
VRS
```

```
## [1] 1.0000 1.0000 1.0000 1.0000 1.0000 0.8963
```

```
peers(VRS)
```

```
##      peer1 peer2 peer3
## [1,]     1    NA    NA
## [2,]     2    NA    NA
## [3,]     3    NA    NA
## [4,]     4    NA    NA
## [5,]     5    NA    NA
## [6,]     1     2     5
```

```
VRS_Weights <- lambda(VRS)
```

All facilities 1,2,3,4,5 are efficient except DMU6 which is 0.8963

#Now, we are going to formulate and compute the DEA analysis using IRS.

#IRS is one of the scaling assumptions It represents that for possible production process we can arbitrarily increase the scale of the operation.

```
IRS <- dea(a,b, RTS = "irs")
IRS
```

```
## [1] 1.0000 1.0000 1.0000 1.0000 1.0000 0.8963
```

```
peers(IRS)
```

```
##      peer1 peer2 peer3
## [1,]     1    NA    NA
## [2,]     2    NA    NA
## [3,]     3    NA    NA
## [4,]     4    NA    NA
## [5,]     5    NA    NA
## [6,]     1     2     5
```

```
IRS_Weights <- lambda(IRS)
```

```
# Now, we are going to formulate and compute the DEA  analysis using DRS.

# DRS is one of the scaling assumptions It represents that for possible production pr
ocess we can arbitrarily decrease the scale of the operation.

DRS <- dea(a,b, RTS = "drs")
DRS
```

```
## [1] 1.0000 1.0000 1.0000 1.0000 0.9775 0.8675
```

```
peers(DRS)
```

```
##      peer1 peer2 peer3
## [1,]     1    NA    NA
## [2,]     2    NA    NA
## [3,]     3    NA    NA
## [4,]     4    NA    NA
## [5,]     1     2     4
## [6,]     1     2     4
```

```
DRS_Weights <- lambda(DRS)
```

```
# Now, we are going to formulate and compute the DEA  analysis using add.

# FRH is a free disposability and replicability hull assumption. It indicates the Add
itivity (scaling up and down, but only with integers) and free disposability.

FRH <- dea(a,b, RTS="add")
FRH
```

```
## [1] 1 1 1 1 1 1
```

```
peers(FRH)
```

```
##      peer1
## [1,]     1
## [2,]     2
## [3,]     3
## [4,]     4
## [5,]     5
## [6,]     6
```

```
FRH_Weights <- lambda(FRH)
```

```
as.data.frame(Matrix)
```

```
##           Staff_Hours Supplies Reimbursed Patient_Days
## Facility1         150      0.2                    14000
## Facility2         400      0.7                    14000
## Facility3         320      1.2                    42000
## Facility4         520      2.0                    28000
## Facility5         350      1.2                    19000
## Facility6         320      0.7                    14000
##           Privately Paid Patient_Days
## Facility1                        3500
## Facility2                       21000
## Facility3                       10500
## Facility4                       42000
## Facility5                       25000
## Facility6                       15000
```

```
DataFrame<- data.frame(CRS = c(1.0000, 1.0000, 1.0000, 1.0000, 0.9775, 0.8675), FDH =
c(1, 1, 1 ,1 ,1 ,1), VRS =c(1.0000, 1.0000, 1.0000, 1.0000 ,1.0000, 0.8963), IRS = c(
1.0000, 1.0000, 1.0000, 1.0000, 1.0000, 0.8963), DRS = c(1.0000, 1.0000, 1.0000 ,1.00
00 ,0.9775, 0.8675), FRH = c(1, 1, 1 ,1, 1, 1))
DataFrame
```

```
##      CRS FDH    VRS    IRS    DRS FRH
## 1 1.0000   1 1.0000 1.0000 1.0000   1
## 2 1.0000   1 1.0000 1.0000 1.0000   1
## 3 1.0000   1 1.0000 1.0000 1.0000   1
## 4 1.0000   1 1.0000 1.0000 1.0000   1
## 5 0.9775   1 1.0000 1.0000 0.9775   1
## 6 0.8675   1 0.8963 0.8963 0.8675   1
```

From the above output the Facilities 1,2,3,4 are fully efficient for all the assumptions and Facilities 5,6 are not efficient.

Facility 5 is fully efficient for FDH, VRS, IRS and FRH assumptions. It is observed that 97.75% efficient for CRS and DRS assumptions. Facility 6 is fully efficient for FDH and FRS assumptions. For Facility 6 CRS and DRS assumptions 86.75% efficient. For Facility 6 IRS and VRS assumptions 89.63% efficient.

Question 2 : GOAL PRORAMMING

Maximize Z = P - 6C - 3D, where P = total (discounted) profit over the life of the new products, C = change (in either direction) in the current level of employment, D = decrease (if any) in next year's earnings from the current year's level.

Profit P is expressed as:

P = 20x1 +15x2 + 25x3

Employment level is expressed as :

6x1 + 4x2 + 5x3 = 50

Next year Earnings goal is expressed as:

8x1 + 7x2 + 5x3 >=75

1. Model_Formulation:

Let us consider y1 - Employment Level minus the target and y2 - Next Year Earnings minus the Target y1+ - Penalty for employment level goal exceeding 50 y1- - Penalty for employment level goal decreasing below 50 y2+ - Exceed the next year earnings y2- - Penalty for not reaching the next year earnings

y1 = 6x1 + 4x2 + 5x3 - 50 y2 = 8x1 + 7x2 + 5x3 - 75

For Employment level goal y1 = y1+ - y1- where y1 +, y1 - >=0 y1+ - y1-= 6x1 + 4x2 + 5x3 - 50

For Next year earnings goal y2 = y2+ - y2- where y2+, y2- >=0 y2+ - y2- = 8x1 + 7x2 + 5x3 -75

Final Formulation is expressed as Max P = 20x1 +15x2 + 25x3 6x1 + 4x2 + 5x3 - (y1+ - y1-) = 50 8x1 + 7x2 + 5x3 - (y2+ - y2-) = 75

xj >= 0, where j = 1,2,3 yi + >= 0, where i = 1,2 yi - >= 0, where i = 1,2

#2)Managements objective function Objective Function

Maximize Z = P - 6C - 3D Objective function in terms of x1, x2, x3, y1+, y1- , y2+ and y2- Max Z = 20x1 +15x2 + 25x3 - 6y1+ - 6y1- - 3y2- S.T.: 6x1 + 4x2 + 5x3 - y1+ + y1- = 50 8x1 + 7x2 + 5x3 - y2+ + y2- = 75 xj >=0 where j=1,2,3 yi + >=0 where i= 1,2 yi - >=0 where i= 1,2

3) Formulate and solve the linear programming model. What are your findings?

```
library(lpSolveAPI)
Maximum<- read.lp("max.lp")
solve(Maximum)
```

```
## [1] 0
```

```
#Print the model
Maximum
```

```
## Model name:
##              x1    x2    x3   y1p   y1q   y2q   y2p
## Maximize     20    15    25    -6    -6    -3     0
## R1            6     4     5    -1     1     0     0  =  50
## R2            8     7     5     0     0     1    -1  =  75
## Kind        Std   Std   Std   Std   Std   Std   Std
## Type       Real  Real  Real  Real  Real  Real  Real
## Upper       Inf   Inf   Inf   Inf   Inf   Inf   Inf
## Lower         0     0     0     0     0     0     0
```

```
#To identify the Optimal Solution
get.objective(Maximum)
```

```
## [1] 225
```

The Optimal Value for the model is 225

```
#To Identify the variables
get.variables(Maximum)
```

```
## [1]  0  0 15 25  0  0  0
```

When the labor and earnings goals are prioritized, the optimal value can be obtained simply by producing products 2 and 3 at rates of 8.33 and 3.33, respectively. While both the workforce and earnings targets are met.

Problems Found

According to the Non Preemptive goal programming model, the objective function can be maximized by adding 25 additional people, which is a non-feasible approach for management. In the preemptive goal programming model using the streamlined approach, we have given the Employment and earnings goals a higher priority than the total profit goal, and the solution indicates that the optimal value has been achieved by achieving all of the goals, as opposed to the previous model where the Employment goal was not achieved.