Final_Project_DEA

Mukhtar A. Yusuf

This contains the code for the examples provided in the DEA module

---

The Hope Valley Health Care Association owns and operates six nursing homes in adjoining states. An evaluation of their efficiency has been undertaken using two inputs and two outputs. The inputs are staffing labor (measured in average hours per day) and the cost of supplies (in thousands of dollars per day). The outputs are the number of patient-days reimbursed by third-party sources and the number of patient-days reimbursed privately.

Let us now calculate the weights to acheive the efficiency values for each DMU (Facility)

Data Preparation

```
getwd()
```

```
## [1] "C:/Users/Mukht/OneDrive/Desktop/Kent State University/College of Business Admin-Bus. Analytics
```

```
setwd("C:\\Users\\Mukht\\OneDrive\\Desktop\\Kent State University\\College of Business Admin-Bus. Analy
```

DMU(1)

```
library(lpSolveAPI)
dmu1 <-read.lp("Final_Proj_DEA.lp")
dmu1
```

```
## Model name:
##                U1      U2      V1
## Maximize     0.75    0.51       0
## R1          10.95    6.09  -17.05  <=  0
## R2           7.01     2.7   -9.71  <=  0
## R3           3.29    1.48   -4.77  <=  0
## R4           0.46    0.79   -1.25  <=  0
## R5           0.34    0.78   -1.11  <=  0
## R6           0.75    0.51   -1.25  <=  0
## Kind          Std     Std     Std
## Type         Real    Real    Real
## Upper         Inf     Inf     0.8
## Lower           0       0     0.8
```

```
solve(dmu1)
```

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

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

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

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

```
## [1] 0.8139346 0.7638217 0.8000000
```

The solution indicates that the objective value is 1, which indicates that we are able to achieve maximum efficiency for DMU(1). This happens when we use the weights 0 and 0.03226 for the outputs, and 0.01 for the input. In other words, if we provide the greatest weight to deposits, then DMU(1) is the most efficient.

Can you modify and rerun the model for all other DMUs?

Using Benchmarking Libraries for DEA

We will now run DEA analysis using the benchmarking library. First, install the library, if you don't have it already. Uncomment the code, i.e., remove the #, to run it.

```
#install.packages("Benchmarking")
library(Benchmarking)
library(ucminf)
library(quadprog)
library(rts)
library(terra)
library(lpSolveAPI)
library(gridExtra)
```

Now, we read our input data. We will read the data as input and output as vectors. Remember our problem had 6 DMUs with investment facilitation and investment promotion as input and FDI inflow as outputs.

```
x <- matrix(c(10.95,7.01,3.29,0.46,0.34,0.75,6.09,2.70,1.48,0.79,0.78,0.51),ncol = 2)
y <- matrix(c(17.05,9.71,4.77,1.25,1.11,1.25),ncol = 1)
colnames(y) <- c("FDI Inflow-Year")
colnames(x) <- c("Invest. facil. Service per year", "Invest. Prom. Service per day")
x
```

```
##      Invest. facil. Service per year Invest. Prom. Service per day
## [1,]                           10.95                          6.09
## [2,]                            7.01                          2.70
## [3,]                            3.29                          1.48
## [4,]                            0.46                          0.79
## [5,]                            0.34                          0.78
## [6,]                            0.75                          0.51
```

```
y
```

```
##      FDI Inflow-Year
## [1,]           17.05
## [2,]            9.71
## [3,]            4.77
## [4,]            1.25
## [5,]            1.11
## [6,]            1.25
```

We now run the DEA analysis. We use the option of CRS, Constant Return to Scale. More on this later.

```
e<-dea(x,y,RTS = "crs")            # provide the input and output
e
```

```
## [1] 1.0000 1.0000 0.9998 1.0000 1.0000 0.9916
```

```
peers(e)                            # identify the peers
```

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

```
lambda(e)                           # identify the relative weights given to the peers
```

```
##              L1        L2        L4 L5
## [1,] 1.00000000 0.0000000 0.0000000  0
## [2,] 0.00000000 1.0000000 0.0000000  0
## [3,] 0.11362609 0.2917276 0.0000000  0
## [4,] 0.00000000 0.0000000 1.0000000  0
## [5,] 0.00000000 0.0000000 0.0000000  1
## [6,] 0.06067381 0.0000000 0.1724092  0
```

```
#dea.plot.isoquant(x,y,RTS="crs")     # plot the results
```

```
***
The results indicate that DMUs 1, 2, 3, and 4 are efficient. DMU(5) is only 98% efficient, and DMU(6) i
***
```

```
e<-dea(x,y,RTS = "drs")             # provide the input and output
e
```

```
## [1] 1.0000 1.0000 0.9998 1.0000 1.0000 0.9916
```

```
peers(e)                             # identify the peers
```

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

```
lambda(e)                            # identify the relative weights given to the peers
```

```
##              L1        L2        L4 L5
## [1,] 1.00000000 0.0000000 0.0000000  0
## [2,] 0.00000000 1.0000000 0.0000000  0
## [3,] 0.11362609 0.2917276 0.0000000  0
## [4,] 0.00000000 0.0000000 1.0000000  0
## [5,] 0.00000000 0.0000000 0.0000000  1
## [6,] 0.06067381 0.0000000 0.1724092  0
```

```
#dea.plot.isoquant(x,y,RTS="drs")     # plot the results
```

```
***
The results indicate that DMUs 1, 2, 3, and 4 are efficient. DMU(5) is only 98% efficient, and DMU(6) i
***
```

```
e<-dea(x,y,RTS = "irs")             # provide the input and output
e
```

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

```
peers(e)                             # identify the peers
```

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

```
lambda(e)                            # identify the relative weights given to the peers
```

```
##      L1 L2 L3 L4 L5 L6
## [1,]  1  0  0  0  0  0
## [2,]  0  1  0  0  0  0
## [3,]  0  0  1  0  0  0
## [4,]  0  0  0  1  0  0
## [5,]  0  0  0  0  1  0
## [6,]  0  0  0  0  0  1
```

```
#dea.plot.isoquant(x,y,RTS="irs")     # plot the results
```

```
***
The results indicate that DMUs 1, 2, 3, 4, and 5 are efficient. DMU(6) is only 90% efficient.Further, tl
***
```

```
e<-dea(x,y,RTS = "vrs")             # provide the input and output
e
```

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

```
peers(e)                              # identify the peers
```

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

```
lambda(e)                             # identify the relative weights given to the peers
```

```
##      L1 L2 L3 L4 L5 L6
## [1,]  1  0  0  0  0  0
## [2,]  0  1  0  0  0  0
## [3,]  0  0  1  0  0  0
## [4,]  0  0  0  1  0  0
## [5,]  0  0  0  0  1  0
## [6,]  0  0  0  0  0  1
```

```
#dea.plot.isoquant(x,y,RTS="vrs")     # plot the results
```

```
***
The results indicate that DMUs 1, 2, 3, 4,and 5 are efficient. DMU(6) is only 90% efficient. Further, tl
***
```

```
e<-dea(x,y,RTS = "fdh")             # provide the input and output
e
```

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

```
peers(e)                                # identify the peers
```

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

```
lambda(e)                               # identify the relative weights given to the peers
```

```
##      L1 L2 L3 L4 L5 L6
## [1,]  1  0  0  0  0  0
## [2,]  0  1  0  0  0  0
## [3,]  0  0  1  0  0  0
## [4,]  0  0  0  1  0  0
## [5,]  0  0  0  0  1  0
## [6,]  0  0  0  0  0  1
```

```
#dea.plot.isoquant(x,y,RTS="fdh")      # plot the results
```

```
***
The results indicate that DMUs 1, 2, 3, 4, 5, and 6 are efficient.
***
```

```
e<-dea(x,y,RTS = "add")               # provide the input and output
e
```

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

```
peers(e)                                # identify the peers
```

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

```
lambda(e)                               # identify the relative weights given to the peers
```

```
##      L1 L2 L3 L4 L5 L6
## [1,]  1  0  0  0  0  0
## [2,]  0  1  0  0  0  0
## [3,]  0  0  1  0  0  0
## [4,]  0  0  0  1  0  0
## [5,]  0  0  0  0  1  0
## [6,]  0  0  0  0  0  1
```

```
#dea.plot.isoquant(x,y,RTS="add")     # plot the results
```

“‘

The results indicate that DMUs 1, 2, 3, 4,and 5 are efficient. DMU(6) is only 90% efficient. Further, the peer units for DMU(6) are 2 and 5, with relative weights 0.34 and 0.25. ***