Assignment 5 - DEA and Goal Programming

Fabrizio Fiorini

11/06/2021

Part 1 - Data Envelopment Analysis

Loading the necessary libraries

```
library(lpSolveAPI)
library(Benchmarking)
```

```
## Warning: package 'Benchmarking' was built under R version 4.0.5
```

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

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

Introduction

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. A summary of performance data is shown in the table below. Questions: 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 3) Summarize your results in a tabular format 4) Compare and contrast the above results

Solve the LP Model

We are going to solve the problem performing DEA analysis. DEA is a mathematical approach of estimating the best performance standards and then evaluating the relatively efficiency of the entities. These entities (the six nursing homes) are called Decision Making Units (DMUs). We first evaluate each DMU to determine its efficiency along the dimensions that are most favorable to the single DMU, and then for those DMUs that are not fully efficient, we establish a reference set, or technology set, to provide a way for them to improve efficiency. Entities in the technology set are called Peer Units.

One critical assumption in the analysis is that which DMU can belong to this technology set. Our choice of which model to use has significant implications on the technology set and on determining who is efficient and who is not.

We are going to use a specific package for DEA, “Benchmarking”, that tells us not only the weights for each DMU, but also the peer units in the hypothetical comparison set (HCUs) and the relative weights needed to make the DMU efficient.

The first step is to create the inputs and outputs by typing both the values and the relative column names. We have two columns of input (matrix x) and two columns of output (matrix y).

```
x <- matrix(c(150, 400, 320, 520, 350, 320,
              0.2, 0.7, 1.2, 2, 1.2, 0.7),
            ncol = 2)
y <- matrix(c(14000, 14000, 42000, 28000, 19000, 14000,
              3500, 21000, 10500, 42000, 25000, 15000),
            ncol = 2)
colnames(x) <- c("Staff H/Day", "Supplies/Day")
colnames(y) <- c("Reimbursed P-Ds","Privately Paid P-Ds")

x
```

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

```
y
```

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

Then, we can apply the dea function with x, y, and the technology assumption. In the following section of code we are going to use six different DEA models and at the end we compare and comment the results. These models differ in the assumptions they make. The output of the dea function provides the efficiency of each DMU.

The peers command provides the technology set for the not efficient DMUs, while the Lambda command provides the weights for each DMU in the peer set for that non-efficient DMU.

```
e1 <- dea(x,y,RTS = "fdh")            # provide the input and output
e1
```

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

```
peers(e1)                             # identify the peers
```

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

```
lambda(e1)                                # 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
```

Here we used FDH (free disposability hull). All of the DMUs are efficient.

```
e2 <- dea(x,y,RTS = "crs")
e2
```

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

```
peers(e2)
```

```
##      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
```

```
lambda(e2)
```

```
##             L1         L2 L3        L4
## [1,] 1.0000000 0.00000000  0 0.0000000
## [2,] 0.0000000 1.00000000  0 0.0000000
## [3,] 0.0000000 0.00000000  1 0.0000000
## [4,] 0.0000000 0.00000000  0 1.0000000
## [5,] 0.2000000 0.08048142  0 0.5383307
## [6,] 0.3428571 0.39499264  0 0.1310751
```

Here we used CRS (constant return to scale). As we can see from the output, inefficient DMUs are 5 and 6. DMU 5 is efficient for only 97.75%, while DMU 6 for 86.75%. Peers for both of them are 1, 2, and 4. The relative weights for them are shown in the lambda output: for example to get the hypothetical comparison unit (HCU) for DMU 5 we have to assess 0.2 to DMU 1, 0.08 to DMU 2, and 0.54 to DMU 4.

```
e3 <- dea(x,y,RTS = "vrs")
e3
```

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

```
peers(e3)
```

```
##      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
```

```
lambda(e3)
```

```
##             L1        L2 L3 L4        L5
## [1,] 1.0000000 0.0000000  0  0 0.0000000
## [2,] 0.0000000 1.0000000  0  0 0.0000000
## [3,] 0.0000000 0.0000000  1  0 0.0000000
## [4,] 0.0000000 0.0000000  0  1 0.0000000
## [5,] 0.0000000 0.0000000  0  0 1.0000000
## [6,] 0.4014399 0.3422606  0  0 0.2562995
```

Here we used VRS (varying return to scale). According to this technology, DMU 6 is inefficient with 89.63%. Its peer units are 1, 2, and 5, with their relative weights shown in the table with lambdas.

```
e4 <- dea(x,y,RTS = "irs")
e4
```

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

```
peers(e4)
```

```
##      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
```

```
lambda(e4)
```

```
##              L1        L2 L3 L4        L5
## [1,] 1.0000000 0.0000000  0  0 0.0000000
## [2,] 0.0000000 1.0000000  0  0 0.0000000
## [3,] 0.0000000 0.0000000  1  0 0.0000000
## [4,] 0.0000000 0.0000000  0  1 0.0000000
## [5,] 0.0000000 0.0000000  0  0 1.0000000
## [6,] 0.4014399 0.3422606  0  0 0.2562995
```

Here we used IRS (increasing return to scale). We obtained the same result as for VRS.

```
e5 <- dea(x,y,RTS = "drs")
e5
```

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

```
peers(e5)
```

```
##      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
```

```
lambda(e5)
```

```
##              L1         L2 L3        L4
## [1,] 1.0000000 0.00000000  0 0.0000000
## [2,] 0.0000000 1.00000000  0 0.0000000
## [3,] 0.0000000 0.00000000  1 0.0000000
## [4,] 0.0000000 0.00000000  0 1.0000000
## [5,] 0.2000000 0.08048142  0 0.5383307
## [6,] 0.3428571 0.39499264  0 0.1310751
```

Here we used DRS (decreasing return to scale). We obtained the same result as for CRS.

```
e6 <- dea(x,y,RTS = "add")
e6
```

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

```
peers(e6)
```

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

```
lambda(e6)
```

```
##      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
```

Here we used FRH (free replicability hull). We obtained the same result as for FDH.

In general, the larger the sets, the larger the estimated technology, the more optimistic we are in estimating the improvement potential of the inefficient DMUs. FDH is the smallest technology set. VRS is larger. By allowing some scaling, we have DRS and IRS, both larger than the first two. By allowing full rescaling and convexity, we determine the largest technology, CRS. FRH is between FDH and CRS.

Part 2 - Goal Programming

The Research and Development Division of the Emax Corporation has developed three new products. A decision now needs to be made on which mix of these products should be produced. Management wants primary consideration given to three factors: total profit, stability in the workforce, and achieving an increase in the company's earnings next year from the $75 million achieved this year. In particular, using the units given in the following table, they want to

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.

The amount of any increase in earnings does not enter into Z, because management is concerned primarily with just achieving some increase to keep the stockholders happy. (It has mixed feelings about a large increase that then would be difficult to surpass in subsequent years.) The impact of each of the new products (per unit rate of production) on each of these factors is shown in the following table:

Questions:

1. Define y1+ and y1-, respectively, as the amount over (if any) and the amount under (if any) the employment level goal. Define y2+ and y2- in the same way for the goal regarding earnings next year. Define x1, x2, and x3 as the production rates of Products 1, 2, and 3, respectively. With these definitions, use the goal programming technique to express y1+, y1-, y2+ and y2- algebraically in terms of x1, x2, and x3. Also express P in terms of x1, x2, and x3.
2. Express management's objective function in terms of x1, x2, x3, y1+, y1-, y2+ and y2-.

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

Our problem is a goal programming problem, where the goals are nonpreemptive, meaning that they have a comparable importance. Specifically, we have a goal regarding the employment level that is a two-sided type of goal, and a goal on the earnings level that is a lower goal.

The three variables that will give us the optimal production mix are the quantities of the new products, x1, x2, and x3.

We can create two auxiliry variables that we call y1 and y2.

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

Now, y1 can be substituted with (y1p-y1n). So y1p is the case when the goal of maintaining the employment level equal to 50 is missed and we have increased that level. On contrary, if the goal is not reached, meaning that the employment level is less than the goal, y1 is equal to y1n.

For the second goal it is the same, with the difference that for our objective function Z, we need to track only the case when we have a decrease in earnings from the current year’s level. Therefore, we consider only y2n.

Rewriting the objective function with the variables we created, we are going to maximize:

Z = 20x1 + 15x2 + 25x3 - 6y1p - 6y1n -3 y2n

We have created an .lp file containing the formulation of this problem in a way that lpSolveAPI can solve. Therefore, first we read the file and then we get the results.

```
library(lpSolveAPI)
gp <- read.lp("emaxcorp.lp")
gp
```

```
## Model name:
##               x1    x2    x3   y1p   y1n   y2n
## Maximize      20    15    25    -6    -6    -3
## R1             6     4     5    -1     1     0   =  50
## R2             8     7     5     0     0     1  >=  75
## Kind         Std   Std   Std   Std   Std   Std
## Type        Real  Real  Real  Real  Real  Real
## Upper        Inf   Inf   Inf   Inf   Inf   Inf
## Lower          0     0     0     0     0     0
```

```
solve(gp)
```

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

```
get.objective(gp)
```

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

```
get.variables(gp)
```

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

From the result we know that the maximum value of Z is 225, the product mix for the three new products is respectively 0, 0, and 15 (we will produce only the product number 3), and finally we will miss the target for the employment level by +25.

With these information we can calculate the total profit from the production:

P = 25*15 = 375