zalshaye_5_Q1

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

Facility 1

```
library(lpSolveAPI)
dmu1 <- read.lp("C:\\Users\\Z\\Desktop\\QMMAssignment5\\DMU1.lp")
dmu1

## Model name:
##              u1     u2     v1     v2
## 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  42000   -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
```

facility 2

```
library(lpSolveAPI)
dmu2 <- read.lp("C:\\Users\\Z\\Desktop\\QMMAssignment5\\DMU2.lp")
dmu2

## Model name:
##              u1     u2     v1     v2
## 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  42000   -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 1.376147e-03 6.422018e-01
```

facility-3

```
library(lpSolveAPI)
dmu3 <- read.lp("C:\\Users\\Z\\Desktop\\QMMAssignment5\\DMU3.lp")
dmu3
```

```
## Model name:
##              u1     u2     v1     v2
## 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  42000   -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
```

facility 4

```
library(lpSolveAPI)
dmu4 <- read.lp("C:\\Users\\Z\\Desktop\\QMMAssignment5\\DMU4.lp")
dmu4
```

```
## Model name:
##              u1     u2     v1     v2
## Maximize  28000  42000      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  42000   -520     -2  <=  0
## R5        19000  25000   -350   -1.2  <=  0
## R6        14000  15000   -320   -0.7  <=  0
## R7            0      0    150      2   =  1
## Kind        Std    Std    Std    Std
## Type       Real   Real   Real   Real
## Upper       Inf    Inf    Inf    Inf
## Lower         0      0      0      0
```

```
solve(dmu4)
```

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

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

```
## [1] 3.466667
```

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

```
## [1] 0.000000e+00 8.253968e-05 6.666667e-03 0.000000e+00
```

facility 5

```
library(lpSolveAPI)
dmu5 <- read.lp("C:\\Users\\Z\\Desktop\\QMMAssignment5\\DMU5.lp")
dmu5
```

```
## Model name:
##              u1     u2     v1     v2
## 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  42000   -520     -2  <=  0
## R5        19000  25000   -350   -1.2  <=  0
## R6        14000  15000   -320   -0.7  <=  0
## R7            0      0    350    1.2   =  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] 0.9774987

get.variables(dmu5)

## [1] 0.0000115123 0.0000303506 0.0010989011 0.5128205128
```

facility 6

```
library(lpSolveAPI)
dmu6 <- read.lp("C:\\Users\\Z\\Desktop\\QMMAssignment5\\DMU6.lp")
dmu6

## Model name:
##              u1     u2     v1     v2
## 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  42000   -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] 0.8674521

get.variables(dmu6)

## [1] 1.620029e-05 4.270987e-05 1.546392e-03 7.216495e-01
```

Benchmarking

```
library(Benchmarking)

## Warning: package 'Benchmarking' was built under R version 4.0.3
```

```
## Loading required package: ucminf

## Warning: package 'ucminf' was built under R version 4.0.3

## Loading required package: quadprog

## Warning: package 'quadprog' was built under R version 4.0.3
```

```
x <- matrix(c(150,400,320,520,350,320,0.2,0.7,1.2,2.0,1.2,0.7),ncol=2)
y <-
matrix(c(14000,14000,42000,28000,19000,14000,3500,21000,10500,42000,25000,150
00),ncol = 2)
colnames(y) <- c("reimbursed","privately paid")
colnames(x) <- c("staff","supplies")
x
```

```
##      staff 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
```

```
y
```

```
##      reimbursed privately paid
## [1,]      14000           3500
## [2,]      14000          21000
## [3,]      42000          10500
## [4,]      28000          42000
## [5,]      19000          25000
## [6,]      14000          15000
```

DEA analysis

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

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

```
DEA1 <- dea(x,y,RTS = "fdh")
DEA1
```

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

```
DEA2 <- dea(x,y,RTS = "vrs")
DEA2
```

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

```
DEA3 <- dea(x,y,RTS = "irs")
DEA3
```

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

DEA4 <- dea(x,y,RTS = "drs")
DEA4

## [1] 1.0000 1.0000 1.0000 1.0000 0.9775 0.8675

DEA5 <- dea(x,y,RTS = "add")
DEA5

## [1] 1 1 1 1 1 1

DEA6 <- dea(x,y,RTS = "vrs+")
DEA6

## [1] 1 1 1 1 1 1

DEA7 <- dea(x,y,RTS = "irs2")
DEA7

## [1] 1 1 1 1 1 1

DEA8 <- dea(x,y,RTS = "fdh+")
DEA8

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

Finding peers

```
peers(DEA)

##      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

peers(DEA1)

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

peers(DEA2)

##      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
```

```
peers(DEA3)
```

```
##      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
```

```
peers(DEA4)
```

```
##      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
```

```
peers(DEA5)
```

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

```
peers(DEA6)
```

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

```
peers(DEA7)
```

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

```
## [5,]     5
## [6,]     6

peers(DEA8)

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

Finding lambda

```
lambda(DEA)

##             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

lambda(DEA1)

##      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

lambda(DEA2)

##             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

lambda(DEA3)

##             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

lambda(DEA4)

##             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

lambda(DEA5)

##      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

lambda(DEA6)

##      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

lambda(DEA7)

##      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

lambda(DEA8)

##      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
```

Compare and contrast the above results

->constant return to scale method results indicates that DMU 1,2,3,4 are efficient. ->DMU 4 is 0.9777 efficient and DMU 5 is 0.8675 efficient. ->Free Disposability Hull)—> All DMUs are efficient. VRS(variable returns to scale) —>DMU 1,2,3,4,5 are efficient and DMU 6 is 0.8963 efficient. IRS(Increasing return to scale) –>DMU 1,2,3,4,5 are efficient and DMU 6 is 0.8963 efficient. DRS(decreasing retun to scale) —>DMU 1,2,3,4 are efficient ,DMU 5 is 0.9775 efficient and DMU 6 is 0.8963 efficient. FRH/Add (Free replicability hull) —> All DMUs are efficient