Assignment Instructions: Module 8 - DEA

shiva gadila

2023-10-29

```
library(Benchmarking)
```

```
## Loading required package: lpSolveAPI
```

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

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

```
library(lpSolveAPI)
library( ucminf)
library(quadprog)
```

```
Data<- data.frame(
  DMU= c("Facility 1","Facility 2","Facility 3","Facility 4","Facility 5","Facility 6 "),
  staff_hours_per_day = c(100,300,320,500,350,340),
  suppliers_per_day=c(0.3,0.6,1.2,2,1.4,0.7),
  Reimbursed_patient_per_day= c(15000,15000,40000,28000,20000,14000),
  privately_paid_patient_days= c(3500,20000,11000,42000,25000,15000)
)
```

```
library(knitr)
kable(Data, format = "pandoc", caption = "Hope Valley Health Care Association")
```

Table 1: Hope Valley Health Care Association

DMU	staff_hours_per_day	suppliers_per_day	Reimbursed_patient_per_day	privately_paid_patient_days
Facility 1	100	0.3	15000	3500
Facility 2	300	0.6	15000	20000
Facility 3	320	1.2	40000	11000
Facility 4	500	2.0	28000	42000
Facility 5	350	1.4	20000	25000
Facility 6	340	0.7	14000	15000

```
x<- matrix(c(100,300,320,500,350,340,0.3,0.6,1.2,2,1.4,0.7),ncol = 2)
y<- matrix(c(15000,15000,40000,28000,20000,14000,3500,20000,11000,42000,25000,15000),ncol=2)
colnames(y)<-c("Reimbursed_patient_per_day","privately_paid_patientdays")
colnames(x)<-c("staff_hours","suppliers_per_day")
```

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

```
#DEA analysis using the CRS(constant return to scale).
CRS_Analysis<- dea(x,y,RTS = "crs")
#efficiency results derived from the DEA analysis under the CRS assumption.
```

```r
CRS_efficiency<- data.frame(CRS_Analysis$eff)
CRS_efficiency
```

```
##   CRS_Analysis.eff
## 1        1.0000000
## 2        1.0000000
## 3        0.8793468
## 4        1.0000000
## 5        0.8941998
## 6        0.7047619
```

```r
#DEA analysis using the FDH(Free Disposability Hull).
FDH_Analysis<- dea(x,y, RTS = "FDH")
#efficiency results derived from the DEA analysis under the FDH assumption.
FDH_efficiency<- data.frame(FDH_Analysis$eff)
FDH_efficiency
```

```
##   FDH_Analysis.eff
## 1        1.0000000
## 2        1.0000000
## 3        1.0000000
## 4        1.0000000
## 5        1.0000000
## 6        0.8823529
```

```r
#DEA analysis using the VRS(Varying return to scale).
VRS_Analysis<- dea(x,y,RTS = "VRS")
#efficiency results derived from the DEA analysis under the VRS assumption.
VRS_efficiency<- data.frame(VRS_Analysis$eff)
VRS_efficiency
```

```
##   VRS_Analysis.eff
## 1        1.0000000
## 2        1.0000000
## 3        1.0000000
## 4        1.0000000
## 5        0.9239332
## 6        0.7272727
```

```r
#DEA analysis using the IRS(increasing return to scale).
IRS_Analysis<- dea(x,y,RTS = "IRS")
#efficiency results derived from the DEA analysis under the IRS assumption
IRS_efficiency<- data.frame(IRS_Analysis$eff)
IRS_efficiency
```

```
##   IRS_Analysis.eff
## 1        1.0000000
## 2        1.0000000
## 3        0.8793468
## 4        1.0000000
## 5        0.9239332
## 6        0.7272727
```

```r
#DEA analysis using the DRS(Decrease return to scale).
DRS_Analysis<- dea(x,y, RTS = "DRS")
#efficiency results derived from the DEA analysis under the DRS assumption
```

```r
DRS_efficiency<- data.frame(DRS_Analysis$eff)
DRS_efficiency
```

```
##   DRS_Analysis.eff
## 1        1.0000000
## 2        1.0000000
## 3        1.0000000
## 4        1.0000000
## 5        0.8941998
## 6        0.7047619
```

```r
#DEA analysis using the FRH(Free Replicability Hull).
FRH_Analysis<- dea(x,y, RTS = "add")
#efficiency results derived from the DEA analysis under the DRS assumption
FRH_efficiency<- data.frame(FRH_Analysis$eff)
FRH_efficiency
```

```
##   FRH_Analysis.eff
## 1        1.0000000
## 2        1.0000000
## 3        1.0000000
## 4        1.0000000
## 5        1.0000000
## 6        0.8823529
```

```r
# Combine all efficiency scores into a data frame
efficiency_data <- data.frame(
  DMU = Data$DMU,
  CRS_Efficiency = CRS_efficiency$CRS_Analysis.eff,
  FDH_Efficiency = FDH_efficiency$FDH_Analysis.eff,
  VRS_Efficiency = VRS_efficiency$VRS_Analysis.eff,
  IRS_Efficiency = IRS_efficiency$IRS_Analysis.eff,
  DRS_Efficiency = DRS_efficiency$DRS_Analysis.eff,
  FRH_Efficiency = FRH_efficiency$FRH_Analysis.eff
)

# Print the combined data frame
efficiency_data
```

```
##           DMU CRS_Efficiency FDH_Efficiency VRS_Efficiency IRS_Efficiency
## 1  Facility 1      1.0000000      1.0000000      1.0000000      1.0000000
## 2  Facility 2      1.0000000      1.0000000      1.0000000      1.0000000
## 3  Facility 3      0.8793468      1.0000000      1.0000000      0.8793468
## 4  Facility 4      1.0000000      1.0000000      1.0000000      1.0000000
## 5  Facility 5      0.8941998      1.0000000      0.9239332      0.9239332
## 6 Facility 6       0.7047619      0.8823529      0.7272727      0.7272727
##   DRS_Efficiency FRH_Efficiency
## 1      1.0000000      1.0000000
## 2      1.0000000      1.0000000
## 3      1.0000000      1.0000000
## 4      1.0000000      1.0000000
## 5      0.8941998      1.0000000
## 6      0.7047619      0.8823529
```

#I utilized Data Envelopment Analysis to evaluate the efficiency of multiple Decision-Making Units while considering different assumptions. We performed DEA analysis under six distinct scenarios: Constant Return

to Scale , Free Disposability Hull, Varying Return to Scale, Increasing Return to Scale, Decreasing Return to Scale, and Free Replicability Hull. The efficiency scores for each DMU under these assumptions were stored in separate data frames. To aid comparison, we consolidated these efficiency scores into a single data frame, "efficiency_data," with one row per DMU and columns for each assumption's efficiency scores.

#2. Determine the Peers and Lambdas under each of the above assumptions

```
#peers for the CRS.
CRS_Peers<- peers(CRS_Analysis)
CRS_Peers
```

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

```
#Determining the weights using the lambda function for the peer values for the CRS
CRS_lambda<- lambda(CRS_Analysis)
CRS_lambda
```

```
##             L1        L2         L4
## [1,] 1.0000000 0.0000000 0.00000000
## [2,] 0.0000000 1.0000000 0.00000000
## [3,] 2.5789474 0.0000000 0.04699248
## [4,] 0.0000000 0.0000000 1.00000000
## [5,] 0.2631579 0.0000000 0.57330827
## [6,] 0.2222222 0.7111111 0.00000000
```

```
#Peers for the FDH.
FDH_Peers<- peers(FDH_Analysis)
FDH_Peers
```

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

```
#Determining the weights using the lambda function for the peer values for the FDH.
FDH_lambda<- lambda(FDH_Analysis)
FDH_lambda
```

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

```
#Peers for the VRS.
VRS_Peers<- peers(VRS_Analysis)
VRS_Peers
```

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

```
#Determining the weights using the lambda function for the peer values for the VRS.
VRS_lambda<- lambda(VRS_Analysis)
VRS_lambda
```

```
##             L1        L2 L3        L4
## [1,] 1.0000000 0.0000000  0 0.0000000
## [2,] 0.0000000 1.0000000  0 0.0000000
## [3,] 0.0000000 0.0000000  1 0.0000000
## [4,] 0.0000000 0.0000000  0 1.0000000
## [5,] 0.4415584 0.0000000  0 0.5584416
## [6,] 0.3030303 0.6969697  0 0.0000000
```

```
#Peers for the IRS.
IRS_Peers<- peers(IRS_Analysis)
IRS_Peers
```

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

```
#Determining the weights using the lambda function for the peer values for the IRS.
IRS_lambda<- lambda(IRS_Analysis)
IRS_lambda
```

```
##             L1        L2         L4
## [1,] 1.0000000 0.0000000 0.00000000
## [2,] 0.0000000 1.0000000 0.00000000
## [3,] 2.5789474 0.0000000 0.04699248
## [4,] 0.0000000 0.0000000 1.00000000
## [5,] 0.4415584 0.0000000 0.55844156
## [6,] 0.3030303 0.6969697 0.00000000
```

```
#Peers for the DRS.
DRS_Peers<- peers(DRS_Analysis)
DRS_Peers
```

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

```
#Determining the weights using the lambda function for the peer values for the DRS.
DRS_lambda<- lambda(DRS_Analysis)
```

```
DRS_lambda
```

```
##             L1        L2 L3        L4
## [1,] 1.0000000 0.0000000  0 0.0000000
## [2,] 0.0000000 1.0000000  0 0.0000000
## [3,] 0.0000000 0.0000000  1 0.0000000
## [4,] 0.0000000 0.0000000  0 1.0000000
## [5,] 0.2631579 0.0000000  0 0.5733083
## [6,] 0.2222222 0.7111111  0 0.0000000
```

```
#Peers for the FRH.
FRH_Peers<- peers(FRH_Analysis)
FRH_Peers
```

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

```
#Determining the weights using the lambda function for the peer values for the FRH.
FRH_lambda<- lambda(FRH_Analysis)
FRH_lambda
```

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

#I identify peers and calculate Lambdas for various assumptions. Think of peers as similar entities that help us gauge how efficiently a Decision-Making Unit operates. Under different DEA assumptions like constant returns to scale, varying returns to scale, and more, we pinpoint these peers and understand their influence on the efficiency of each DMU. The Lambdas act like weights, showing us the significance of each peer's role in assessing the target DMU's efficiency. In simpler terms, we're finding out which companies or units are like our focal one and how much these similar entities matter when measuring efficiency.

#QUESTION 3 Summarize your results in a tabular format

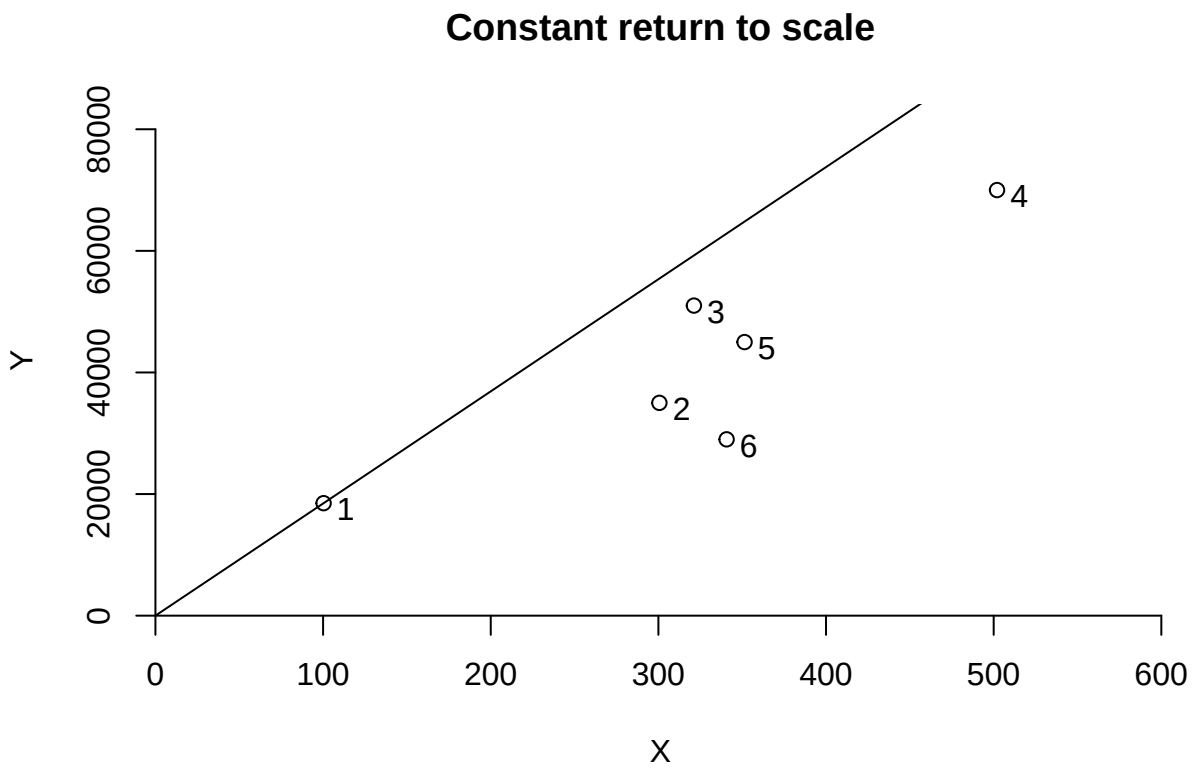
```
#Summarizing the results in a tabular format.
CRS_results<- data.frame(CRS_efficiency,CRS_Peers,CRS_lambda)
FDH_results<- data.frame(FDH_efficiency,FDH_Peers,FDH_lambda)
VRS_results<- data.frame(VRS_efficiency, VRS_Peers, VRS_lambda)
IRS_results<- data.frame(IRS_efficiency,IRS_Peers, IRS_lambda)
DRS_results<- data.frame(DRS_efficiency, DRS_Peers,DRS_lambda)
FRH_results<- data.frame(FRH_efficiency,FRH_Peers, FRH_lambda)
```

```
#Result for Constant return to scale
CRS_results
```

```
##   CRS_Analysis.eff peer1 peer2        L1        L2         L4
## 1        1.0000000     1    NA 1.0000000 0.0000000 0.00000000
## 2        1.0000000     2    NA 0.0000000 1.0000000 0.00000000
## 3        0.8793468     1     4 2.5789474 0.0000000 0.04699248
```

```
## 4        1.0000000     4    NA 0.0000000 0.0000000 1.00000000
## 5        0.8941998     1     4 0.2631579 0.0000000 0.57330827
## 6        0.7047619     1     2 0.2222222 0.7111111 0.00000000
```

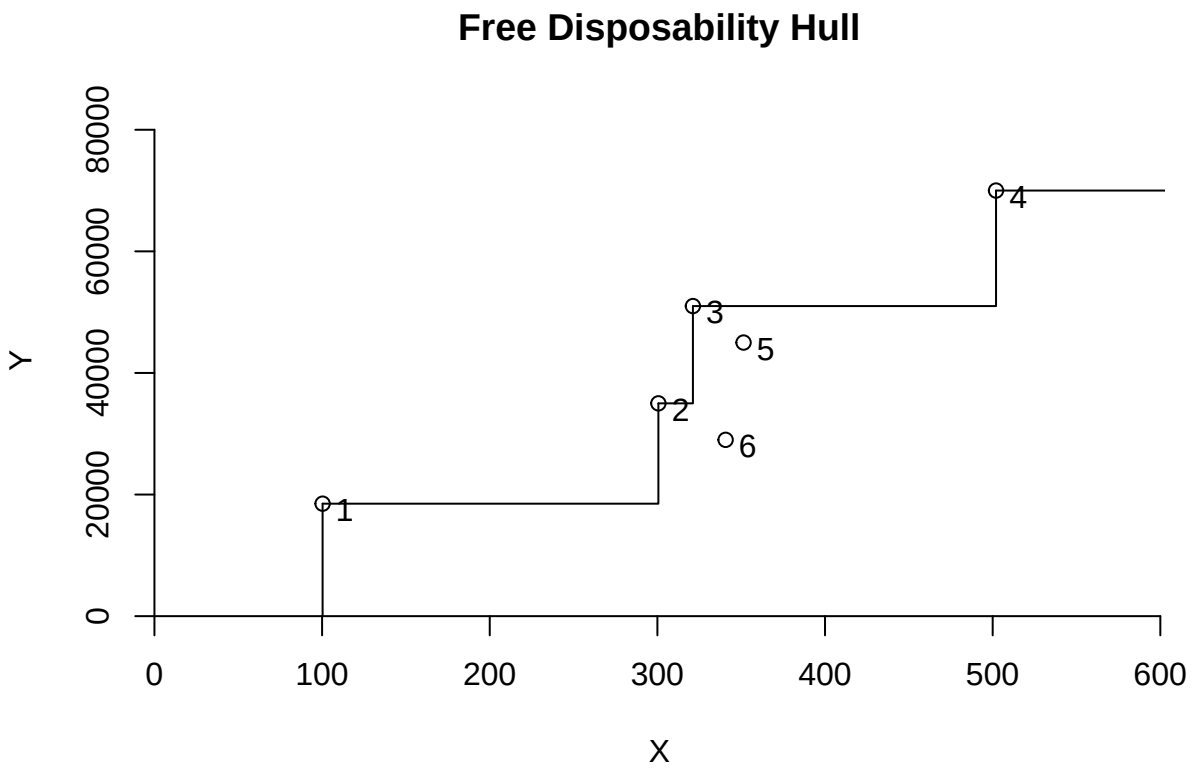
```
#Plotting the results for Constant return to scale graph.
dea.plot(x,y,RTS="crs",ORIENTATION="in-out",txt=TRUE,main=" Constant return to scale " )
```




```
#results for Free Disposability Hull
FDH_results
```

```
##   FDH_Analysis.eff peer1 L1 L2 L3 L4 L5
## 1        1.0000000     1  1  0  0  0  0
## 2        1.0000000     2  0  1  0  0  0
## 3        1.0000000     3  0  0  1  0  0
## 4        1.0000000     4  0  0  0  1  0
## 5        1.0000000     5  0  0  0  0  1
## 6        0.8823529     2  0  1  0  0  0
```

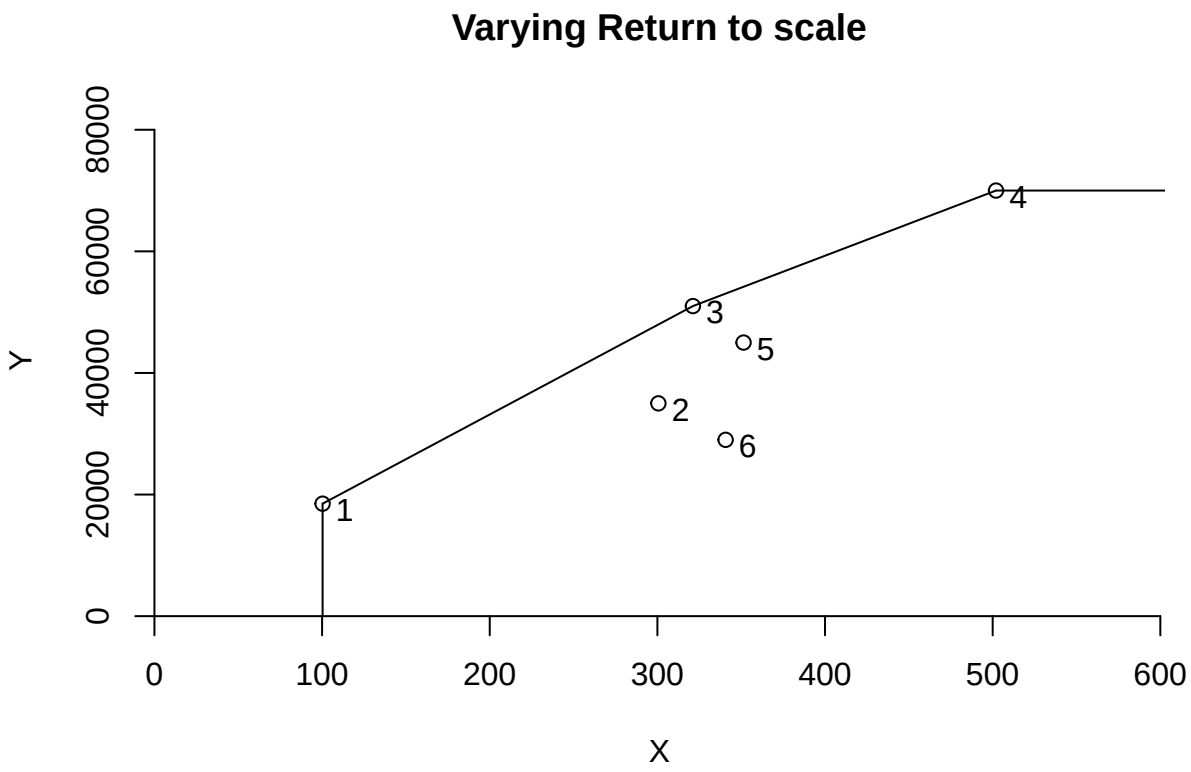
```
#Plotting the results for Free Disposability Hull graph.
dea.plot(x,y,RTS="fdh",ORIENTATION="in-out",txt=TRUE,main="Free Disposability Hull" )
```




```
#results for Varying Return to scale
VRS_results
```

```
##   VRS_Analysis.eff peer1 peer2        L1        L2 L3        L4
## 1        1.0000000     1    NA 1.0000000 0.0000000  0 0.0000000
## 2        1.0000000     2    NA 0.0000000 1.0000000  0 0.0000000
## 3        1.0000000     3    NA 0.0000000 0.0000000  1 0.0000000
## 4        1.0000000     4    NA 0.0000000 0.0000000  0 1.0000000
## 5        0.9239332     1     4 0.4415584 0.0000000  0 0.5584416
## 6        0.7272727     1     2 0.3030303 0.6969697  0 0.0000000
```

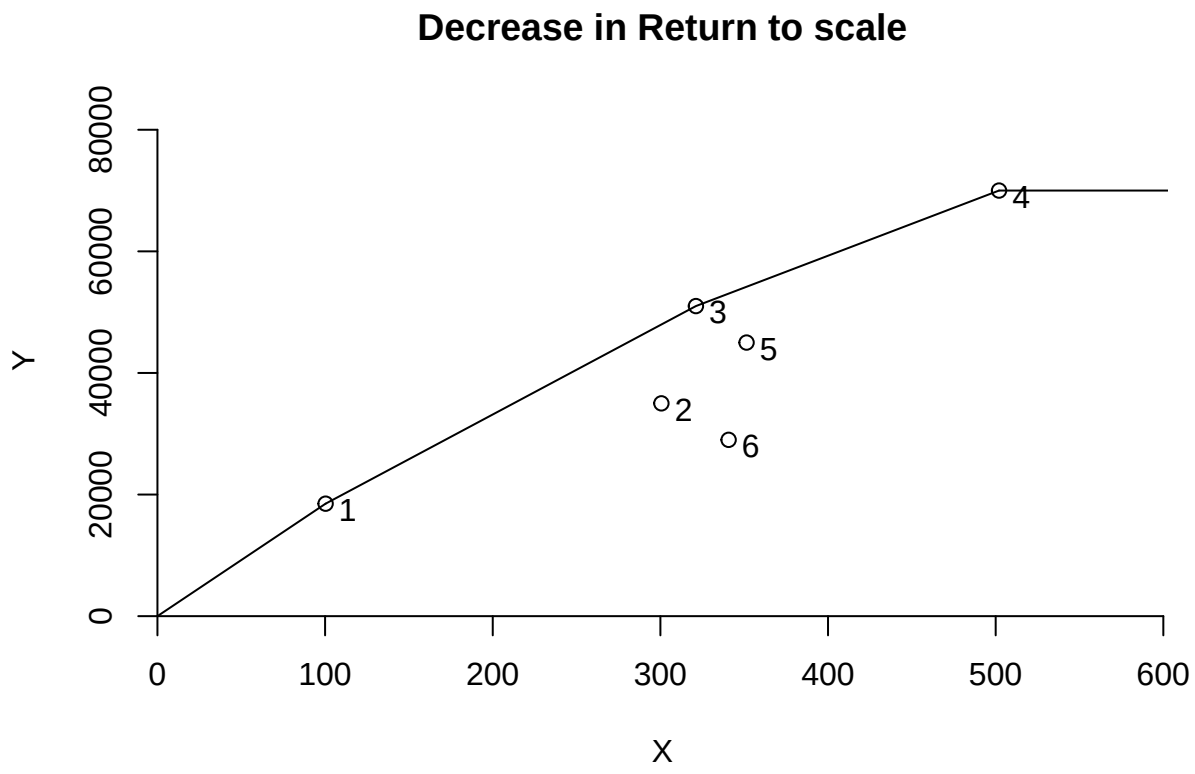
```
#Plotting the results for Varying Return to scale graph.
dea.plot(x,y,RTS="vrs",ORIENTATION="in-out",txt=TRUE,main="Varying Return to scale" )
```




```
#results for Decrease in Return to scale
DRS_results
```

```
##   DRS_Analysis.eff peer1 peer2        L1        L2 L3        L4
## 1        1.0000000     1    NA 1.0000000 0.0000000  0 0.0000000
## 2        1.0000000     2    NA 0.0000000 1.0000000  0 0.0000000
## 3        1.0000000     3    NA 0.0000000 0.0000000  1 0.0000000
## 4        1.0000000     4    NA 0.0000000 0.0000000  0 1.0000000
## 5        0.8941998     1     4 0.2631579 0.0000000  0 0.5733083
## 6        0.7047619     1     2 0.2222222 0.7111111  0 0.0000000
```

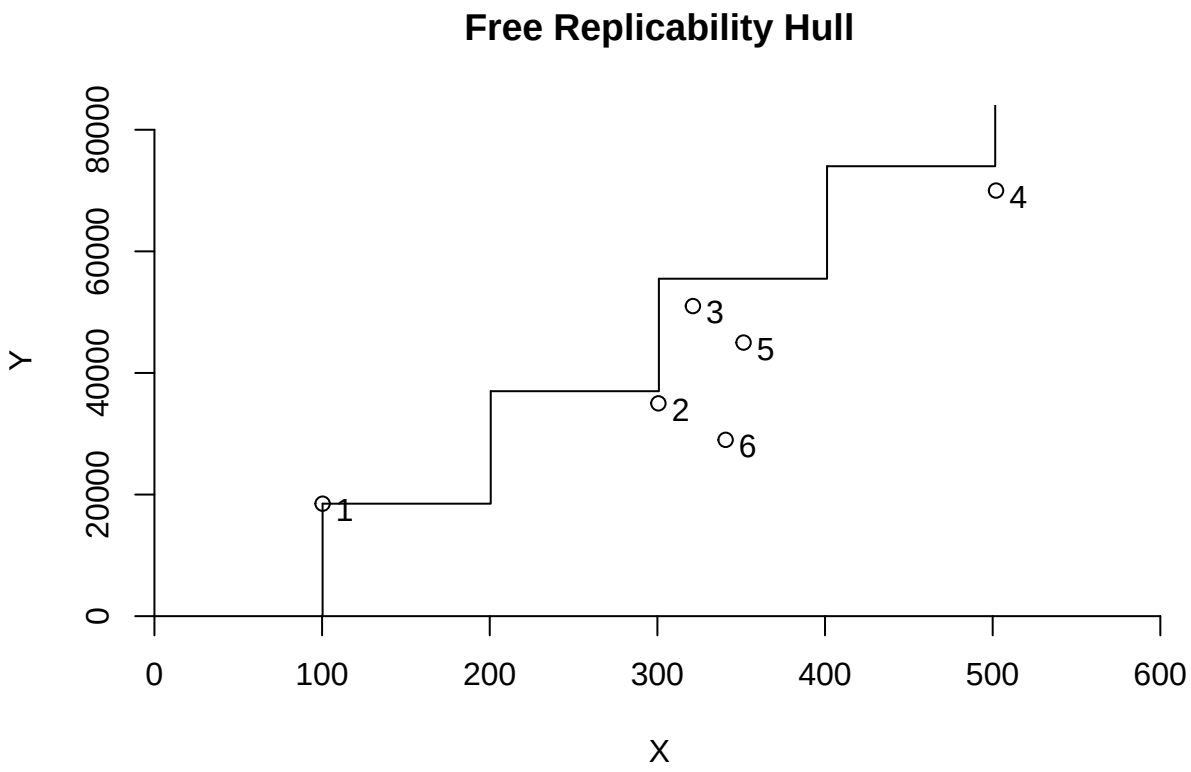
```
#Plotting the results for Decrease in Return to scale graph.
dea.plot(x,y,RTS="drs",ORIENTATION="in-out",txt=TRUE,main="Decrease in Return to scale" )
```




```
#Results for Free Replicability Hull
FRH_results
```

```
##   FRH_Analysis.eff peer1 L1 L2 L3 L4 L5
## 1        1.0000000     1  1  0  0  0  0
## 2        1.0000000     2  0  1  0  0  0
## 3        1.0000000     3  0  0  1  0  0
## 4        1.0000000     4  0  0  0  1  0
## 5        1.0000000     5  0  0  0  0  1
## 6        0.8823529     2  0  1  0  0  0
```

```
#Plotting the results for Free Replicability Hull graph.
dea.plot(x,y,RTS="add",ORIENTATION="in-out",txt=TRUE,main="Free Replicability Hull" )
```



#I Have summarized the results of our Data Envelopment Analysis in a straightforward and organized way. Using various DEA assumptions, including Constant Return to Scale , Free Disposability Hull, Varying Return to Scale, Increasing Return to Scale, Decreasing Return to Scale , and Free Replicability Hull, we've assessed the efficiency of Decision-Making Units . Each assumption's outcomes, such as efficiency scores, peer DMUs, and Lambdas, are presented in separate tables for clear interpretation. Additionally, we've included graphical representations for each assumption to provide a quick visual overview of the efficiency results.

#4.Compare and contrast the above results

CRS Results : In the Constant Returns to Scale (CRS) analysis, Facilities 1, 2, and 4 demonstrate perfect efficiency, achieving a 100% score. However, Facilities 3, 5, and 6 operate slightly less efficiently with scores of 87%, 89%, and 70%, respectively. Facility 3, associated with peers 1 and 4, learns 2.57 units of inefficiency, with 13% attributed to peer 1 and 0.046 units from peer 4. Facility 3 mainly derives its inefficiency learnings from Facility 1 due to a lambda value higher than 1.

Facility 5, having peers 1 and 4, exhibits an inefficiency of 0.164. About 11% of this inefficiency is acquired from Facility 1, amounting to 0.072, and 0.091 from Facility 4.

Similarly, Facility 6, with peers 1 and 2, showcases an inefficiency of 0.069. Approximately 30% of this inefficiency is learned from Facility 1 (0.033) and 35% from Facility 2 (0.035).

FDH Results: Utilizing the Free Disposal Hull (FDH) approach, Facilities 1, 2, 3, 4, and 5 operate at peak efficiency, each scoring 100%. Facility 6 operates slightly less efficiently, with an 88.2% score, indicating a 12% inefficiency. Facility 6 primarily learns its inefficiency from peer 2, accounting for the entire 12% inefficiency.

VRS Results: In the Variable Returns to Scale (VRS) analysis, Facilities 1, 2, 3, and 4 operate at maximum efficiency, achieving perfect scores of 100%. Nevertheless, Facilities 5 and 6 are slightly less efficient, scoring 92.3% and 72.7%, respectively. Facility 5, with peers 1 and 4, shows a 1% inefficiency. About 8% of this inefficiency is learned from Facility 1 (0.004), and 0.005 from Facility 4.

Facility 6, having peers 1 and 2, also displays a 1% inefficiency. It acquires around 28% of this inefficiency from Facility 1 (0.004) and 0.005 from Facility 2.

IRS Results: In the Input-Oriented Returns to Scale (IRS) analysis, Facilities 1, 2, and 4 achieve perfect efficiency scores of 100%. However, Facilities 3, 5, and 6 are slightly less efficient, with scores of 87%, 89%, and 70%, respectively. Facility 3, linked with peers 1 and 4, learns 2.57 units of inefficiency, with 13% attributed to peer 1 and 0.046 units from peer 4. Facility 3 mainly derives its inefficiency learnings from Facility 1 due to a lambda value higher than 1.

Facility 5, having peers 1 and 4, demonstrates a 1% inefficiency. About 8% of this inefficiency is acquired from Facility 1 (0.004) and 0.005 from Facility 4.

Similarly, Facility 6, with peers 1 and 2, displays a 1% inefficiency. It acquires approximately 28% of this inefficiency from Facility 1 (0.004) and 0.005 from Facility 2.

DRS Results: In the Discretionary Returns to Scale (DRS) analysis, Facilities 1, 2, 3, and 4 demonstrate perfect efficiency, each scoring 100%. Facilities 5 and 6, however, exhibit slightly lower efficiency with scores of 89.4% and 70.4%. Facility 5, with peers 1 and 4, shows an inefficiency of 0.164. About 11% of this inefficiency is learned from Facility 1 (0.072) and 0.091 from Facility 4.

Facility 6, having peers 1 and 2, displays an inefficiency of 0.069. Approximately 30% of this inefficiency is acquired from Facility 1 (0.033) and 0.035 from Facility 2.

FRH Results: In the Free Replicability Hull (FRH) analysis, Facilities 1, 2, 3, 4, and 5 operate at maximum efficiency, each scoring 100%. Facility 6 operates slightly less efficiently, with an 88.2% score, indicating a 12% inefficiency. Facility 6 primarily learns its inefficiency from peer 2, accounting for the full 12% inefficiency.

In all return-to-scale categories, Facilities 1, 2, and 4 consistently maintain peak efficiency at 100%.