Quantitative Management Modeling- Assignment-4

Elmy Luka

2022-10-30

```
library("Benchmarking")
```

```
## Loading required package: lpSolveAPI

## Loading required package: ucminf

## Loading required package: quadprog

##
## Loading Benchmarking version 0.30h, (Revision 244, 2022/05/05 16:31:31) ...

## Build 2022/05/05 16:31:40
```

```
data_1_values<- matrix(c("Facility 1","Facility 2",
                         "Facility 3","Facility 4","Facility 5", "Facility 6",
                150,400,320,520,350,320,
                0.2,0.7,1.2,2.0,1.2,0.7,
                14000,14000,42000,28000,19000,14000,
                3500,21000,10500,42000,25000,15000), ncol=5, byrow=F)
colnames(data_1_values) <- c("DMU", "Staff.Hours.Per.Day","Supplies.Per.Day",
      "Reimbursed.Patient_Days","Privately.Paid.Patient.Days")
table.data <- as.table(data_1_values)
table.data
```

```
##   DMU        Staff.Hours.Per.Day Supplies.Per.Day Reimbursed.Patient_Days
## A Facility 1 150                 0.2              14000
## B Facility 2 400                 0.7              14000
## C Facility 3 320                 1.2              42000
## D Facility 4 520                 2                28000
## E Facility 5 350                 1.2              19000
## F Facility 6 320                 0.7              14000
##   Privately.Paid.Patient.Days
## A 3500
## B 21000
## C 10500
## D 42000
## E 25000
## F 15000
```

#Calculating Constant Returns to Scale (CRS)

```r
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,
                3500,21000,10500,42000,25000,15000),ncol=2)
colnames(y) <- c("Reimbursed.Patient.Days","Privately.Paid.Patient.Days")
colnames(x) <- c("Staff.Hours.Per.Day","Supplies.Per.Day")
D.E.A_CRS<-dea(x, y, RTS = "crs")
D.E.A_CRS
```

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

```r
peers(D.E.A_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
```

```r
lambda(D.E.A_CRS)
```

```
##             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
```

#CRS Observations

#It is analysed from the observation that Facility 1, Facility 2, Facility 3 and Facility 4 are efficient.Also the Facility 5 and Facility 6 are inefficient facilities to which Facility1, Facility 2 and Facility 4 are the peer members. #Facility 5 is 97.75 % efficient leaving 2.25 % as inefficient and Facility 6 is 86.75 % efficient leaving 13.25 % as inefficient.

#Calculating Decreasing Returns to Scale (DRS)

```r
D.E.A_DRS <- dea(x, y, RTS = "drs")
D.E.A_DRS
```

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

```r
peers(D.E.A_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
```

```r
lambda(D.E.A_DRS)
```

```
##              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
```

#DRS Observations

#We are able to observe the efficiency of Facilities 1, 2, 3, and 4. Additionally, we learn that the inefficient facilities i.e. Facility 5 and Facility 6 have Facility 1, Facility 2, and Facility 4 as peer members. Facilities 5 and 6 are both 96.75% efficient, leaving 2.25% and 13.25% of inefficiencies, respectively.

#Calculating Increasing Returns to Scale (IRS)

```r
D.E.A_IRS <- dea(x, y, RTS = "irs")
D.E.A_IRS
```

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

```r
peers(D.E.A_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
```

```r
lambda(D.E.A_IRS)
```

```
##              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
```

#IRS Observations

#Here we get to observe the efficiency of Facilities 1, 2, 3, 4, and 5. Additionally, we learn that Facility 6, the only inefficient facility has Facility 1, Facility 2, and Facility 5 as peer members. Facility 6 has an efficiency rate of 89.63%, leaving a 10.37% inefficiency.

#Calculating Variable Returns to Scale (VRS)

```r
D.E.A_VRS <- dea(x, y, RTS = "vrs")
D.E.A_VRS
```

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

```r
peers(D.E.A_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
```

```r
lambda(D.E.A_VRS)
```

```
##             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
```

#VRS Observations

#We get to observe the efficiency of Facilities 1, 2, 3, 4, and 5. Additionally, we learn that Facility 6, the only inefficient facility has Facility 1, Facility 2, and Facility 5 as peer members . Facility 6 has an efficiency rate of 89.63%, leaving a 10.37% inefficiency.

#Calculating Free Disposability Hull (FDH)

```r
D.E.A_FDH <- dea(x, y, RTS = "fdh")
D.E.A_FDH
```

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

```r
peers(D.E.A_FDH)
```

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

```r
lambda(D.E.A_FDH)
```

```
##      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
```

#FDH Observations

#Every DMU is efficient. This is mostly caused by the principle that the FDH technique follows, which allows it to identify even a very low level of efficiency.

#Calculating Free Replicability Hull (FRH)

```
#here FRH is calculated by specifying RTS = "add"
D.E.A_FRH <- dea(x, y, RTS = "add")
D.E.A_FRH
```

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

```
peers(D.E.A_FRH)
```

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

```
lambda(D.E.A_FRH)
```

```
##      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
```

#FRH Observations

#The DMUs are all effective. Because it adheres to the no convexity assumption, the output is protected against disposal and reproduction.

#Summary of Inefficient DMUs

```
summarise_inefficient_data_1 <- matrix(c("CRS","DRS","IRS","VRS","FDH","FRH",
2,2,1,1,0,0,
"Facility 5 & 6", "Facility 5 & 6","Facility 6", "Facility 6", "-","-",
"97.75% & 86.7%","97.75% & 86.7%","89.63%","89.63%","-","-",
"Facility 1, 2 & 4","Facility 1, 2 & 4","Facility 1, 2 & 5","Facility 1, 2 & 5","-","-",
"0.2, 0.08, 0.54 and 0.34, 0.4, 0.13", "0.2, 0.08, 0.54 and 0.34, 0.4, 0.13", "0.4, 0.34 and 0.26", "0.
colnames(summarise_inefficient_data_1) <- c("RTS","Count_Inefficient_DMUs","Name_DMUs","%_Inefficiency"
as.table(summarise_inefficient_data_1 )
```

```
##   RTS Count_Inefficient_DMUs Name_DMUs      %_Inefficiency Peers
## A CRS 2                      Facility 5 & 6 97.75% & 86.7% Facility 1, 2 & 4
## B DRS 2                      Facility 5 & 6 97.75% & 86.7% Facility 1, 2 & 4
## C IRS 1                      Facility 6     89.63%         Facility 1, 2 & 5
## D VRS 1                      Facility 6     89.63%         Facility 1, 2 & 5
## E FDH 0                      -              -              -
## F FRH 0                      -              -              -
##   Lambda
## A 0.2, 0.08, 0.54 and 0.34, 0.4, 0.13
## B 0.2, 0.08, 0.54 and 0.34, 0.4, 0.13
## C 0.4, 0.34 and 0.26
## D 0.4, 0.34 and 0.26
## E -
## F -
```

#Summary of Efficient DMUs

```
summarise_efficient_data_1 <- matrix(c("CRS","DRS","IRS","VRS","FDH","FRH",
"Facility 1, 2, 3 & 4","Facility 1, 2, 3 & 4","Facility 1, 2, 3, 4 & 5", "Facility 1, 2, 3, 4 & 5", "All
colnames(summarise_efficient_data_1) <- c("RTS", "Efficient_DMUs")
as.table(summarise_efficient_data_1)
```

```
##   RTS Efficient_DMUs
## A CRS Facility 1, 2, 3 & 4
## B DRS Facility 1, 2, 3 & 4
## C IRS Facility 1, 2, 3, 4 & 5
## D VRS Facility 1, 2, 3, 4 & 5
## E FDH All DMUs
## F FRH All DMUs
```

#DEA Analysis - Interpretation

#CRS - Constant Returns to Scale

#On the basis of our initial analysis, it can be inferrd from the observation above that DMUs 1, 2, 3, and 4 are efficient. Only 97.75% of DMU(5) and 86.7% of DMU(6) are efficient. . In addition, DMU(4) peer's units are 1, 2, and 4, with respective weights of 0.2, 0.08, and 0.54. The peer units for DMU(6) are 1, 2, and 4, with respective weights of 0.34, 0.4, and 0.13.

#DRS - Decreasing Returns to Scale

#The findings show that DMUs 1, 2, 3, and 4 are efficient. Only 97.75% of DMU(5) and 86.7% of DMU(6) are efficient. On the basis of our initial analysis, we discovered this. In addition, DMU(4) peer's units are 1, 2, and 4, with respective weights of 0.2, 0.08, and 0.54. The peer units for DMU(6) are 1, 2, and 4, with respective weights of 0.34, 0.4, and 0.13. #By looking at the inefficient DMUs i.e DMUs 5 and 6, we may determine whether there are any alternative DMUs where we can scale the processes. As the base original scale, the CRS values can also be used to obtain this information.

#IRS - Increasing Returns to Scale

#The findings show that DMUs 1, 2, 3, 4, and 5 are productive. Only 89.63% of the DMU(6) is efficient. Additionally, the peer units for DMU(6) are 1, 2, and 5, with corresponding relative weights of 0.4, 0.34, and 0.26. By examining the efficiency scores, it can tell any company if they can arbitrary raise the scope of operations.

#VRS - Variable Returns to Scale

#The outcomes show that DMUs 1, 2, 3, 4, and 5 are productive. Only 89.63% of the DMU(6) is efficient. Additionally, DMU(6) peer units are 1, 2, and 5, with respective relative weights of 0.4, 0.34, and 0.26. Understanding the scale of operations with adjustments in the input and output factor either by increasing or decreasing or by using both—is made possible by varying or variable returns to scale.

#FDH - Free Disposability Hull

#The findings show that all DMUs are efficient. This is primarily because the scale is able to measure even the lowest degree of efficiency because there is no convexity assumption.

#FRH - Free Replicability Hull

#All DMUs are efficient, according to the FRH data. This is mainly because there isn't a convexity assumption used, and most of the time, this technique enables the scale to capture even the tiniest amount of efficiency that is not subject to replication or disposal.

#Only inefficient DMUs will receive the peer values, also known as neighbors and lambda values, or weights of the peers. Peers and lambda weights are not present in efficient DMUs.

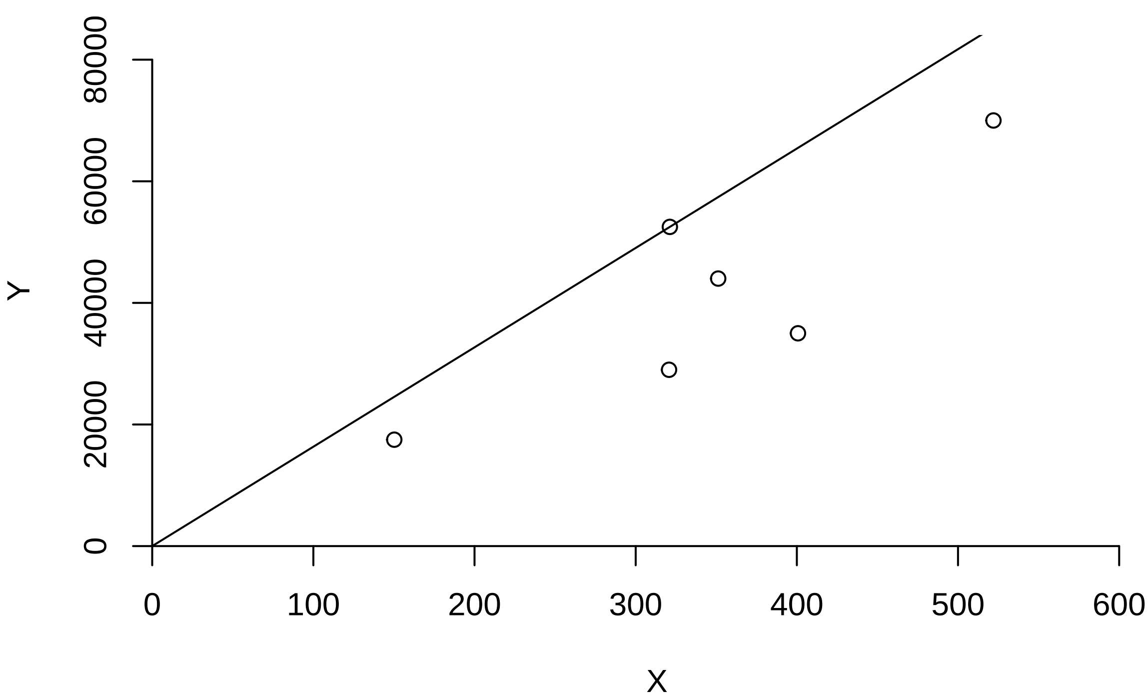
#Conclusion

#It is crucial to remember that DEA is a very helpful tool for any company in determining which DMU is the best, i.e., which of the Decision Making Units has to be maximized so that there would be an increase, decrease, or any other modifications to the output by feeding input into it. A corporation can choose which Returns to Scale (RTS) scale to use based on its needs; each of these scales has a different level of significance.

#Plotting the Graphs

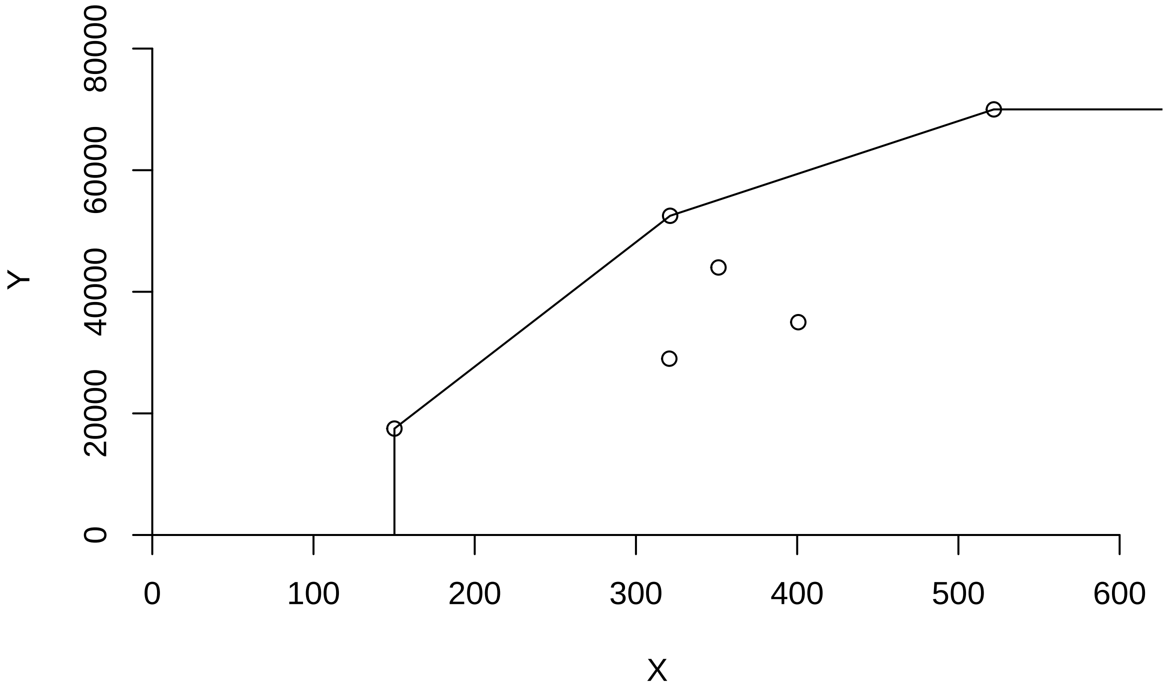
#CRS Plot

```
dea.plot(x, y, RTS="CRS")
```



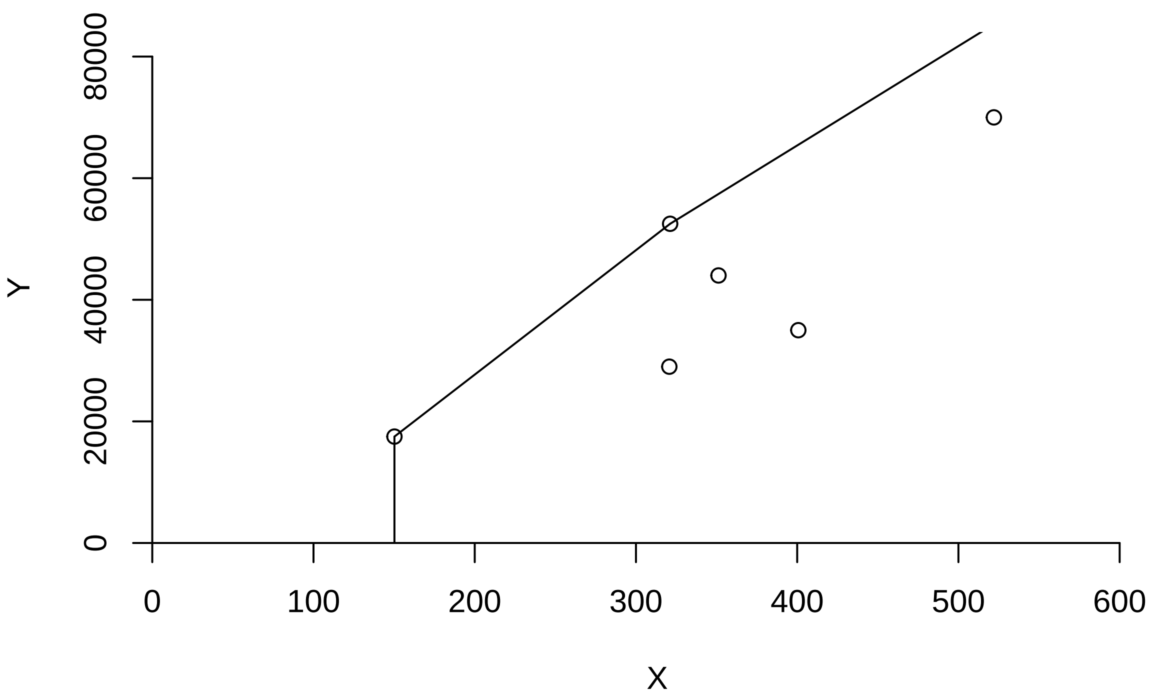
#DRS Plot

```
dea.plot(x,y,RTS="VRS")
```



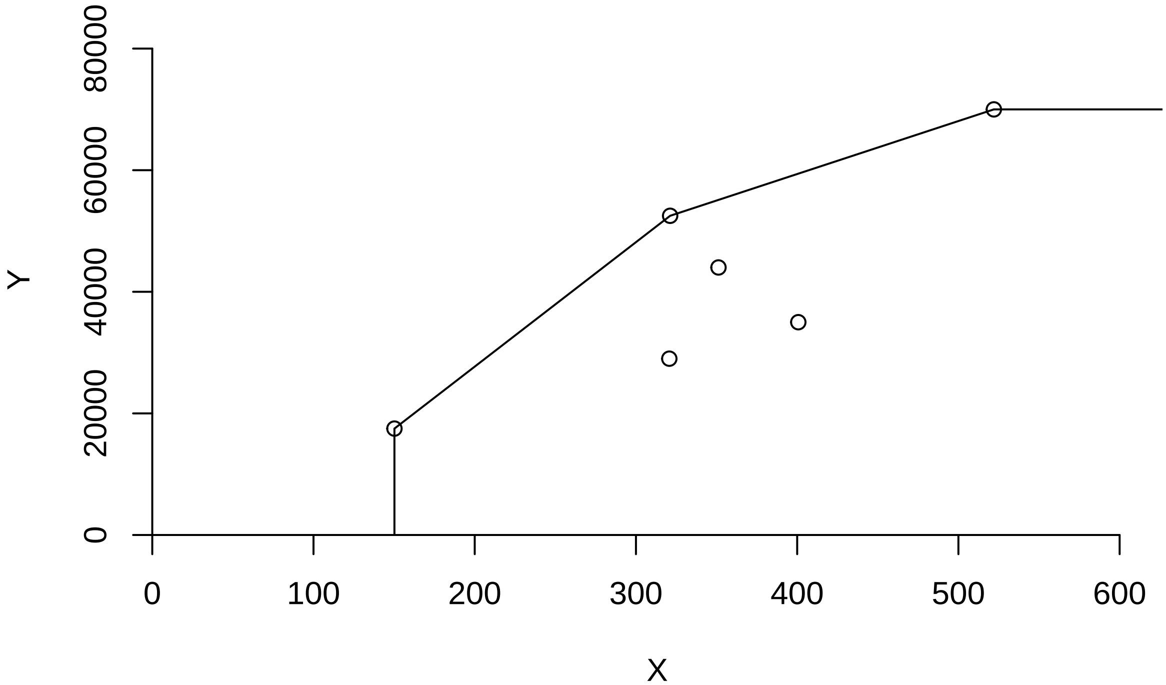
#IRS Plot

```
dea.plot(x,y,RTS="IRS")
```



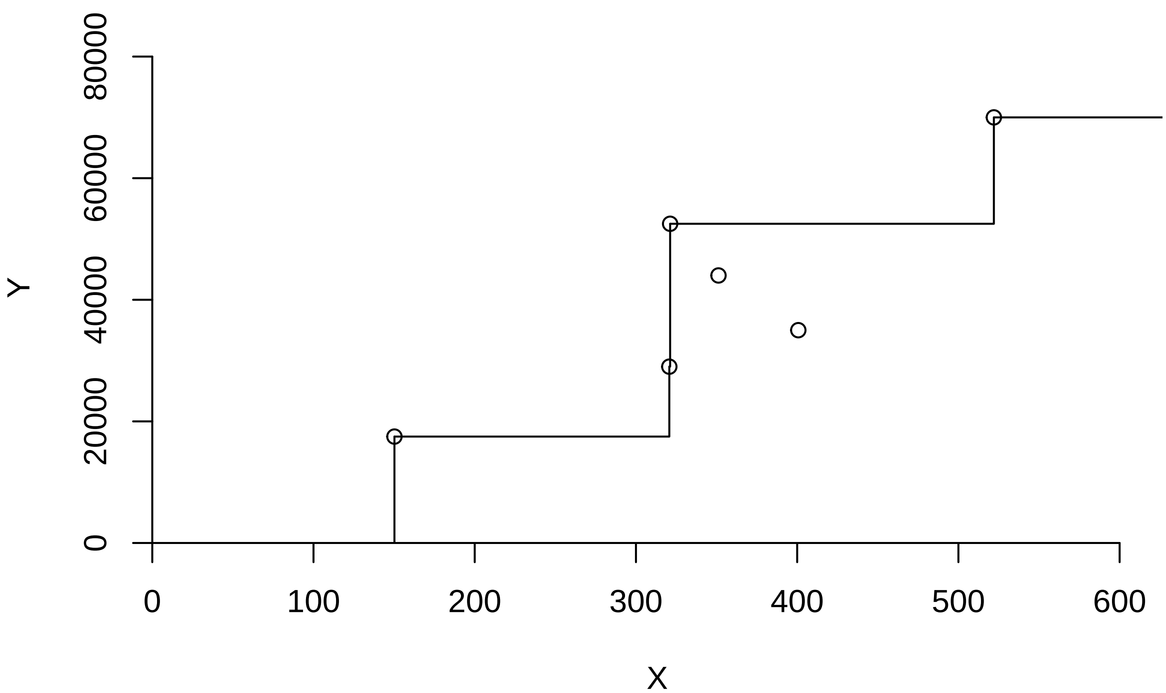
#VRS Plot

```
dea.plot(x,y,RTS="VRS")
```



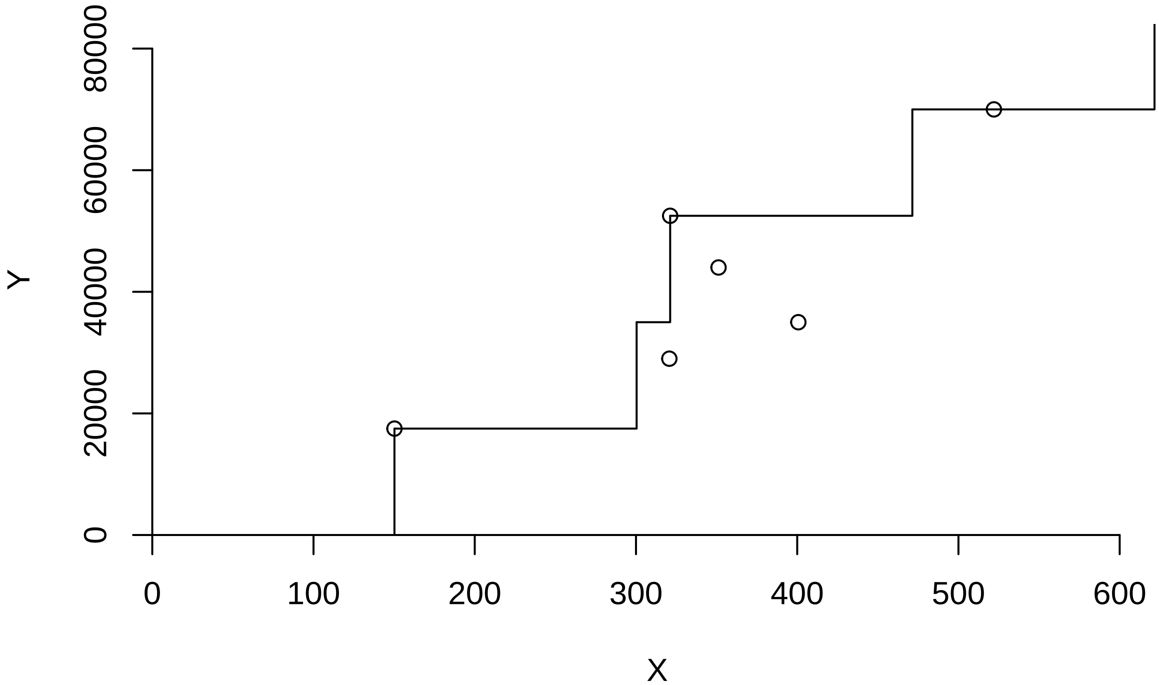
#FDH Plot

```
dea.plot(x,y,RTS="FDH")
```



#FRH Plot

```
dea.plot(x,y,RTS="ADD")
```



0
20000
40000
60000
80000
Y
0
100
200
300
400
500
600
X

“‘