ETM540 - Homework#4 - Data Envelopment Analysis

Mala Daryanani

October 24, 2018

1. Graphical Representation, efficiencies & λ values with additional 5^{th} unit:

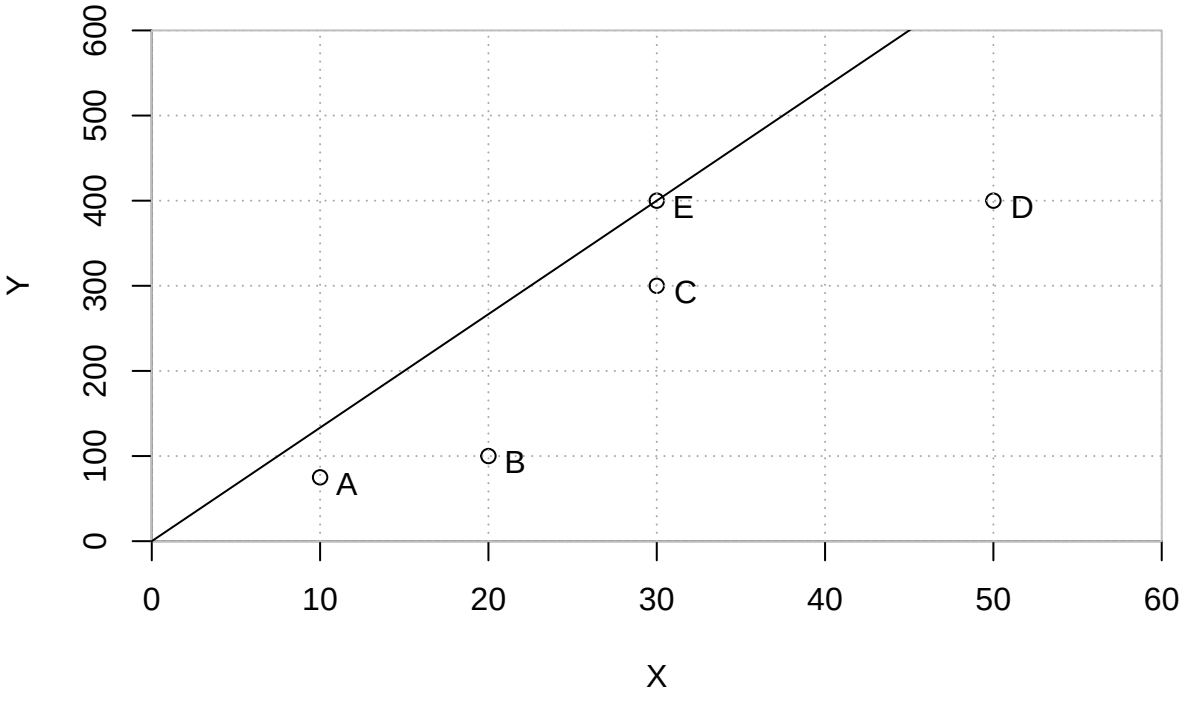
Graphical Analysis (Input-Orientation, Constant-Return-Scale):

```
x <- matrix(c(10,20,30,50,30),ncol=1,dimnames=list(LETTERS[1:5],"x"))
y <- matrix(c(75,100,300,400,400),ncol=1,dimnames=list(LETTERS[1:5],"y"))
pander(cbind(x,y), caption="Ex.1 - Five unit Dataset for DEA")
```

Table 1: Ex.1 - Five unit Dataset for DEA

	x	y
A	10	75
B	20	100
C	30	300
D	50	400
E	30	400

```
dea.plot(x, y, RTS="crs", ORIENTATION="in-out", txt=LETTERS[1:length(x)],
         add=FALSE, wx=NULL, wy=NULL, TRANSPOSE=FALSE, fex=1, GRID=TRUE,
         RANGE=FALSE, param=NULL,ylim = c(0,600))
```



Efficiencies (graphically) & Interpretations:

- $\theta_A = 06/10 = 60\%$ - Implies: 'A' works on 60% of what it's expected. To increase the efficiency of A, there are two options - either increase the output of A (given same input) OR we reduce the input to A and expect the same output.
- $\theta_B = 07/20 = 35\%$ - Implies: 'B' works on 35% of what it's expected. This is way less than the expected output. To improve the efficiency of B, the input should be reduced substantially towards the optimal model line OR increase the output.
- $\theta_C = 22/30 = 73.33\%$ - Implies: 'C' works on 73.33% of what it's expected, which is good compared to the rest. The remaining approx. 26% of it's lag can be achieved by similar ways explained above - adjust the input/output accordingly.
- $\theta_D = 30/50 = 60\%$ - Implies: 'D' works on 60% of what it's expected. To increase the efficiency of D, there are two options - either increase the output of A (given same input) OR we reduce the input to A and expect the same output.
- $\theta_E = 30/30 = 100\%$ - Implies: 'E' works on 100% efficiency. With given input, it provides the maximum output. Hence, we can consider E as the benchmark to compare others and push their efficiencies to match E. However, there are limitations to this theory, as the working model of E may differ as being a big retailer store like Amazon, and we are trying to compare this model with a small unit as Dunkin Donuts - which would not be fair.

λ scores & Interpretations:

- $\lambda_A = 06/30 = 0.2$ - A is composed of approx. 20% of E.
- $\lambda_B = 07/30 = 0.233$ - B is composed of approx. 23.3% of E.
- $\lambda_C = 22/30 = 0.733$ - C is composed of approx. 73.3% of E.
- $\lambda_D = 30/30 = 1$ - D,E have same inputs, so for input-orientation D is not required to reduce any further.
- $\lambda_E = 30/30 = 1$ - E provides the best efficiency, so there is no change required.

2. Interpretation of above solution:

With the addition of E (30,400), the optimal line now passes through E instead of C (in previous example). This is because comparing the both, E generates more output than C with the given same input fed to both. Now E is the best solution and so lambda values are found with reference to this point.

θ_B has the least efficiency amongst all with 35% explaining its poor performance. It needs around 65% of improvment to match the efficieny of E, which can be done by either decreasing the input to B or try increase the output with given resources.

λ scores provide us knowledge of how much scaling up/down of E is required to compose the rest of points. Considering λ_A, it is made of only E with scaling it down to one-fifth of its original amount. The same implies to rest points, as they are all compared to E (being the best solution). Since λ_D & λ_E have same inputs, explains why λ_D is 1.

3. Using R, examine the new unit:

```
  ND <- nrow(x); NX <- ncol(x); NY <- ncol(y);

  xdata<-x[1:ND,]
  dim(xdata)<-c(ND,NX)
  ydata<-y[1:ND,]
  dim(ydata)<-c(ND,NY)

  DMUnames <- list(c(LETTERS[1:ND]))
  Xnames<- lapply(list(rep("X",NX)),paste0,1:NX)
  Ynames<- lapply(list(rep("Y",NY)),paste0,1:NY)
  Lambdanames<- lapply(list(rep("L_",ND)),paste0,LETTERS[1:ND])

k <- 5
  fitModel_E <- MIPModel() %>%
    add_variable(vlambda[j], j = 1:ND, type = "continuous", lb = 0) %>%
    add_variable(vtheta, type = "continuous") %>%
   set_objective(vtheta, "min") %>%
    add_constraint(sum_expr(vlambda[j] * xdata[j,1], j = 1:ND)
                   <= vtheta * xdata[k,1]) %>%
    add_constraint(sum_expr(vlambda[j] * ydata[j,1], j = 1:ND)
                   >= ydata[k,1]) %>%
  solve_model(with_ROI(solver = "glpk"))

omprtheta <-  get_solution(fitModel_E, vtheta)
omprlambda <-  get_solution(fitModel_E, vlambda[j])

fitModel_E.efficiency <- matrix(rep(-1.0, 1), nrow=1, ncol=1)
fitModel_E.lambda     <- matrix(rep(-1.0, ND), nrow=1,ncol=ND)

fitModel_E.efficiency <- t(omprtheta)
colnames(fitModel_E.efficiency) <- c("CCR-IO")

fitModel_E.lambda <- t(omprlambda[3])
colnames(fitModel_E.lambda) <- c("L_A", "L_B", "L_C", "L_D", "L_E")

pander(cbind(fitModel_E.efficiency, fitModel_E.lambda),
       caption="Input-Oriented Envelopment Analysis for DMU E (CCR-IO)")
```

Table 2: Input-Oriented Envelopment Analysis for DMU E (CCR-IO)

	CCR-IO	L_A	L_B	L_C	L_D	L_E
value	1	0	0	0	0	1

5. FOR loop - Including all units:

$$(xi - input; yj - output)$$

$$\text{minimize } \theta$$

$$\text{subject to } \sum_{j=1}^{5} x_{i,j}\lambda_j \leq \theta x_{i,k} \forall\ i = 1, 2.., 5$$

$$\sum_{j=1}^{5} y_{r,j}\lambda_j \geq y_{r,k} \forall\ r = 1, 2.., 5$$

$$\lambda_j \geq 0 \ \forall\ j = A, B, C, D, E$$

```
  results.efficiency <- matrix(rep(-1.0, ND), nrow=ND, ncol=1)
  dimnames(results.efficiency)<-c(DMUnames,"CCR-IO")
  results.lambda     <- matrix(rep(-1.0, ND^2), nrow=ND,ncol=ND)
  dimnames(results.lambda)<-c(DMUnames,Lambdanames)

  for (k in 1:ND) {
 result <- MIPModel() %>%
  add_variable(vlambda[j], j = 1:ND, type = "continuous", lb = 0) %>%
  add_variable(vtheta, type = "continuous") %>%
  set_objective(vtheta, "min") %>%
  add_constraint(sum_expr(vlambda[j] * xdata[j,i], j = 1:ND) - (vtheta * xdata[k,i]) <=0,
                 i = 1:NX) %>%
  add_constraint(sum_expr(vlambda[j] * ydata[j,r], j = 1:ND) - ydata[k,r] >= 0,
                 r = 1:NY) %>%
  solve_model(with_ROI(solver = "glpk"))

  print(c("DMU=",k,solver_status(result)))
  results.efficiency[k] <-  get_solution(result, vtheta)
  results.lambda[k,] <- t(as.matrix(as.numeric(get_solution(result, vlambda[j])[,3] )))
}
```

```
## [1] "DMU="    "1"       "optimal"
## [1] "DMU="    "2"       "optimal"
## [1] "DMU="    "3"       "optimal"
## [1] "DMU="    "4"       "optimal"
## [1] "DMU="    "5"       "optimal"
```

```
Lambdanames <- list("L_A", "L_B", "L_C", "L_D", "L_E")
DMUnames <- list("A", "B", "C", "D","E")
dimnames(results.efficiency)<-list(DMUnames,"CCR-IO")
dimnames(results.lambda)<-list(DMUnames,Lambdanames)
pander (cbind(results.efficiency, results.lambda),caption="Input-Oriented Efficiency Results")
```

Table 3: Input-Oriented Efficiency Results

	CCR-IO	L_A	L_B	L_C	L_D	L_E
A	0.5625	0	0	0	0	0.1875
B	0.375	0	0	0	0	0.25
C	0.75	0	0	0	0	0.75
D	0.6	0	0	0	0	1
E	1	0	0	0	0	1

Results compared with the graphical interpretations:

If we compare the efficiencies θ & λ scores obtained from using R and that from the graph, they are almost similar. While calculating the points from graph, we ball park the values on the x-axis, that's why we see deviation compared to the actual data obtained from the code R.

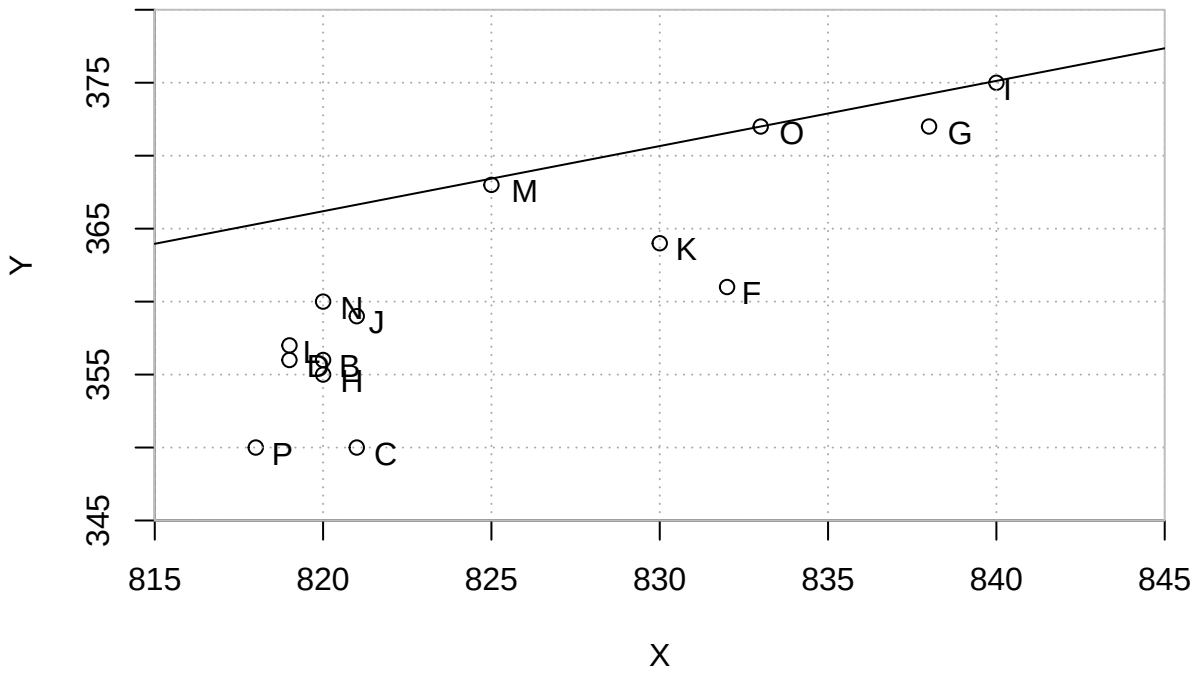
6. Analysis & Interpretation of Fuel Consumption vs. it's Density:

```
y_fuel <- matrix(c(318,356,350,356,320,361,372,355,375,359,364,357,368,360,372,350),
            ncol=1,dimnames=list(LETTERS[1:16],"y Fuel Consumption (g/km)"))
x_fuel <- matrix(c(818,820,821,819,818,832,838,820,840,821,830,819,825,820,833,818),
            ncol=1,dimnames=list(LETTERS[1:16],"x Density (g/L)"))
pander(cbind(x_fuel,y_fuel), caption="DEA - Fuel Consumption vs. Density")
```

Table 4: DEA - Fuel Consumption vs. Density

	x Density (g/L)	y Fuel Consumption (g/km)
A	818	318
B	820	356
C	821	350
D	819	356
E	818	320
F	832	361
G	838	372
H	820	355
I	840	375
J	821	359
K	830	364
L	819	357
M	825	368
N	820	360
O	833	372
P	818	350

```
dea.plot(x_fuel, y_fuel, RTS="crs", ORIENTATION="in-out", txt=LETTERS[1:length(x_fuel)],
         add=FALSE, wx=NULL, wy=NULL, TRANSPOSE=FALSE, fex=1, GRID=TRUE,
         RANGE=FALSE, param=NULL, xlim = c(815,845), ylim = c(345,380))
```




```
ND <- nrow(x_fuel); NX <- ncol(x_fuel); NY <- ncol(y_fuel);

  xdata<-x_fuel[1:ND,]
  dim(xdata)<-c(ND,NX)
  ydata<-y_fuel[1:ND,]
  dim(ydata)<-c(ND,NY)

  results.efficiency <- matrix(rep(-1.0, ND), nrow=ND, ncol=1)
  results.lambda     <- matrix(rep(-1.0, ND^2), nrow=ND,ncol=ND)

for (k in 1:ND) {
 result1 <- MIPModel() %>%
  add_variable(vlambda[j], j = 1:ND, type = "continuous", lb = 0) %>%
  add_variable(vtheta, type = "continuous") %>%
  set_objective(vtheta, "min") %>%
  add_constraint(sum_expr(vlambda[j] * xdata[j,i], j = 1:ND) - (vtheta * xdata[k,i]) <=0,
                 i = 1:NX) %>%
  add_constraint(sum_expr(vlambda[j] * ydata[j,r], j = 1:ND) - ydata[k,r] >= 0,
                 r = 1:NY) %>%
  solve_model(with_ROI(solver = "glpk"))

  print(c("DMU=",k,solver_status(result1)))

  results.efficiency[k] <-  get_solution(result1, vtheta)
  results.lambda[k,] <- t(as.matrix(as.numeric(
                        get_solution(result1, vlambda[j])[,3] )))
}
```

```
## [1] "DMU="    "1"       "optimal"
## [1] "DMU="    "2"       "optimal"
## [1] "DMU="    "3"       "optimal"
## [1] "DMU="    "4"       "optimal"
## [1] "DMU="    "5"       "optimal"
```

```
## [1] "DMU="    "6"       "optimal"
## [1] "DMU="    "7"       "optimal"
## [1] "DMU="    "8"       "optimal"
## [1] "DMU="    "9"       "optimal"
## [1] "DMU="    "10"      "optimal"
## [1] "DMU="    "11"      "optimal"
## [1] "DMU="    "12"      "optimal"
## [1] "DMU="    "13"      "optimal"
## [1] "DMU="    "14"      "optimal"
## [1] "DMU="    "15"      "optimal"
## [1] "DMU="    "16"      "optimal"
```

```
Lambdanames<- lapply(list(rep("L_",ND)),paste0,LETTERS[1:ND])
DMUnames <- list(c(LETTERS[1:ND]))
dimnames(results.efficiency)<-c(DMUnames,"CCR-IO")
dimnames(results.lambda)<-c(DMUnames,Lambdanames)
```

```
pander (cbind(results.efficiency, results.lambda),
        caption="Input-Oriented Efficiency Results")
```

Table 5: Input-Oriented Efficiency Results (continued below)

	CCR-IO	L_A	L_B	L_C	L_D	L_E	L_F	L_G	L_H	L_I	L_J
A	0.8705	0	0	0	0	0	0	0	0	0	0
B	0.9722	0	0	0	0	0	0	0	0	0	0
C	0.9546	0	0	0	0	0	0	0	0	0	0
D	0.9733	0	0	0	0	0	0	0	0	0	0
E	0.876	0	0	0	0	0	0	0	0	0	0
F	0.9716	0	0	0	0	0	0	0	0	0	0
G	0.994	0	0	0	0	0	0	0	0	0	0
H	0.9694	0	0	0	0	0	0	0	0	0	0
I	0.9997	0	0	0	0	0	0	0	0	0	0
J	0.9792	0	0	0	0	0	0	0	0	0	0
K	0.982	0	0	0	0	0	0	0	0	0	0
L	0.9761	0	0	0	0	0	0	0	0	0	0
M	0.9988	0	0	0	0	0	0	0	0	0	0
N	0.9831	0	0	0	0	0	0	0	0	0	0
O	1	0	0	0	0	0	0	0	0	0	0
P	0.9581	0	0	0	0	0	0	0	0	0	0

	L_K	L_L	L_M	L_N	L_O	L_P
A	0	0	0	0	0.8548	0
B	0	0	0	0	0.957	0
C	0	0	0	0	0.9409	0
D	0	0	0	0	0.957	0
E	0	0	0	0	0.8602	0
F	0	0	0	0	0.9704	0
G	0	0	0	0	1	0
H	0	0	0	0	0.9543	0
I	0	0	0	0	1.008	0
J	0	0	0	0	0.9651	0
K	0	0	0	0	0.9785	0
L	0	0	0	0	0.9597	0
M	0	0	0	0	0.9892	0
N	0	0	0	0	0.9677	0
O	0	0	0	0	1	0
P	0	0	0	0	0.9409	0

7. Validate above Results with DEAMultiplier:

```
DEAMultip_Model <- DeaMultiplierModel(x = x_fuel, y = y_fuel, rts = "crs",
                                      orientation = "input")
pander(cbind(DEAMultip_Model$Efficiency, DEAMultip_Model$Lambda),
       caption = "DEAMultiplier used for Analysis")
```

Table 7: DEAMultiplier used for Analysis

	Eff	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
A	0.8705	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8548	0
B	0.9722	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.957	0
C	0.9546	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9409	0
D	0.9733	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.957	0
E	0.876	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.8602	0
F	0.9716	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9704	0
G	0.994	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
H	0.9694	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9543	0
I	0.9997	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1.008	0
J	0.9792	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9651	0
K	0.982	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9785	0
L	0.9761	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9597	0
M	0.9988	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9892	0
N	0.9831	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9677	0
O	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
P	0.9581	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.9409	0