# jnoxon_4.R

Jason

2021-10-24

```r
#Assignment 4

library(lpSolveAPI)


WD<-setwd("C:/Users/Jason/Documents/MSBA/Quant")

#Problem 1

######### Plant A ##############
# Plant A Production cost = $600
# Plant A -> Wharehouse 1 = $22
# Plant A -> Wharehouse 2 = $14
# Plant A -> wharehouse 3 = $30
######## Plant B ###############
# Plant B Production cost = $625
# Plant B -> Wharehouse 1 = $16
# Plant B -> Wharehouse 2 = $20
# Plant B -> Wharehouse 3 = $24
####### Supply & Demand ########
# Total Demand = 210
# Total Supply = 220
# Supply > Demand
# Need dummy demand variable

###### Objective Function ######
# C = 622X1 + 614X2 + 630X3 + 631X4 + 645X5 + 649X6

##### Constraints ########
##Demand##
# X1 + X4 = 80
# X2 + X5 = 60
# X3 + X6 = 70
# X7 + X8 = 10
##Supply##
# X1 + X2 + X3 + X7 = 100
# X4 + X5 + X6 + X8 = 120

lpobj1 <- make.lp(0, 8)

set.objfn(lpobj1, c(622, 614, 630, 631, 645, 649, 0, 0))
```

```r
lp.control(lpobj1, sense = "min")
```

```
## $anti.degen
## [1] "fixedvars" "stalling"
##
## $basis.crash
## [1] "none"
##
## $bb.depthlimit
## [1] -50
##
## $bb.floorfirst
## [1] "automatic"
##
## $bb.rule
## [1] "pseudononint" "greedy"       "dynamic"      "rcostfixing"
##
## $break.at.first
## [1] FALSE
##
## $break.at.value
## [1] -1e+30
##
## $epsilon
##       epsb       epsd      epsel     epsint epsperturb   epspivot
##      1e-10      1e-09      1e-12      1e-07      1e-05      2e-07
##
## $improve
## [1] "dualfeas" "thetagap"
##
## $infinite
## [1] 1e+30
##
## $maxpivot
## [1] 250
##
## $mip.gap
## absolute relative
##    1e-11    1e-11
##
## $negrange
## [1] -1e+06
##
## $obj.in.basis
## [1] TRUE
##
## $pivoting
## [1] "devex"    "adaptive"
##
## $presolve
## [1] "none"
##
## $scalelimit
```

```
## [1] 5
##
## $scaling
## [1] "geometric"   "equilibrate" "integers"
##
## $sense
## [1] "minimize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
## [1] 0
##
## $verbose
## [1] "neutral"
```

```
add.constraint(lpobj1, c(1, 0, 0, 1, 0, 0, 0, 0), "=",80)
add.constraint(lpobj1, c(0, 1, 0, 0, 1, 0, 0, 0), "=",60)
add.constraint(lpobj1, c(0, 0, 1, 0, 0, 1, 0, 0), "=",70)
add.constraint(lpobj1, c(0, 0, 0, 0, 0, 0, 1, 1), "=",10)
add.constraint(lpobj1, c(1, 1, 1, 0, 0, 0, 1, 0), "=",100)
add.constraint(lpobj1, c(0, 0, 0, 1, 1, 1, 0, 1), "=",120)

set.bounds(lpobj1, lower = c(0, 0, 0, 0, 0, 0, 0, 0), columns = 1:8)


solve(lpobj1)
```

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

```
get.objective(lpobj1)
```

```
## [1] 131990
```

```
get.variables(lpobj1)
```

```
## [1]  0 60 40 80  0 30  0 10
```

```
# Problem 2

###### Oil Well Production ######
# Well 1 = 93
# Well 2 = 88
# Well 3 = 95
###### Refinery Demand ######
# R1 = 30
# R2 = 57
# R3 = 48
# R4 = 91
# R5 = 48
###### Total Supply ######
93 + 88 + 95 # = 276
```

```
## [1] 276
```

```r
###### Total Demand ######
30 + 57 + 48 + 91 + 48 # = 274
```

```
## [1] 274
```

```r
# Supply > Demand
# Need dummy demand variable
###### Transport Costs Wells -> Pumps ######
## Well 1 ##
# W1 -> P1 = 1.52
# W1 -> P2 = 1.60
# W1 -> P3 = 1.40
## Well 2 ##
# W2 -> P1 = 1.70
# W2 -> P2 = 1.63
# W2 -> P3 = 1.55
## Well 3 ##
# W3 -> P1 = 1.45
# W3 -> P2 = 1.57
# W3 -> P3 = 1.30
###### Transport costs Pumps -> Refineries ######
## Pump 1 ##
# P1 -> R1 = 5.15
# P1 -> R2 = 5.69
# P1 -> R3 = 6.13
# P1 -> R4 = 5.63
# P1 -> R5 = 5.80
## Pump 2 ##
# P2 -> R1 = 5.12
# P2 -> R2 = 5.47
# P2 -> R3 = 6.05
# P2 -> R4 = 6.12
# P2 -> R5 = 5.71
## Pump 3 ##
# P3 -> R1 = 5.32
# P3 -> R2 = 6.16
# P3 -> R3 = 6.25
# P3 -> R4 = 6.17
# P3 -> R5 = 5.87

##### Objective function #####
# C = 1.52X1 + 1.60X2 + 1.40X3 + 1.70X4 + 1.63X5 + 1.55X6 + 1.45X7 +
#   + 1.57X8 + 1.30X9 + 5.15X10 + 5.69X11 + 6.13X12 + 5.63X13 + 5.80X14
#   + 5.12X15 + 5.47X16 + 6.05X17 + 6.12X18 + 5.71X19 + 5.32X20 + 6.16X21
#   + 6.25X22 + 6.17X23 + 5.87X24

##### Constraints #####
## Well Constraints ##
# X1 + X2 + X3 = 93
# X4 + X5 + X6 = 88
# X7 + X8 + X9 = 95
```

```r
## Refinery Constraints ##
# X10 + X15 + X20 = 30
# X11 + X16 + X21 = 57
# X12 + X17 + X22 = 48
# X13 + X18 + X23 = 91
# X14 + X19 + X24 = 48
## Dummy variable constraint ##
# X25 + X26 + X27 = 2
## Transshipment constraints ##
# -X1 + -X4 + -X7 + X10 + X11 + X12 + X13 + X14 = 0
# -X2 + -X5 + -X8 + X15 + X16 + X17 + X18 + X19 = 0
# -X3 + -X6 + -X9 + X20 + X21 + X22 + X23 + X24 = 0

lpobj2 <- make.lp(0, 27)

set.objfn(lpobj2, c(1.52, 1.60, 1.40, 1.70, 1.63, 1.55, 1.45, 1.57, 1.30, 5.15, 5.69, 6.13, 5.63, 5.80,

lp.control(lpobj2, sense = "min")
```

```
## $anti.degen
## [1] "fixedvars" "stalling"
##
## $basis.crash
## [1] "none"
##
## $bb.depthlimit
## [1] -50
##
## $bb.floorfirst
## [1] "automatic"
##
## $bb.rule
## [1] "pseudononint" "greedy"       "dynamic"      "rcostfixing"
##
## $break.at.first
## [1] FALSE
##
## $break.at.value
## [1] -1e+30
##
## $epsilon
##       epsb       epsd      epsel     epsint epsperturb   epspivot
##      1e-10      1e-09      1e-12      1e-07      1e-05      2e-07
##
## $improve
## [1] "dualfeas" "thetagap"
##
## $infinite
## [1] 1e+30
##
## $maxpivot
## [1] 250
##
```

```
## $mip.gap
## absolute relative
##    1e-11    1e-11
##
## $negrange
## [1] -1e+06
##
## $obj.in.basis
## [1] TRUE
##
## $pivoting
## [1] "devex"    "adaptive"
##
## $presolve
## [1] "none"
##
## $scalelimit
## [1] 5
##
## $scaling
## [1] "geometric"   "equilibrate" "integers"
##
## $sense
## [1] "minimize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
## [1] 0
##
## $verbose
## [1] "neutral"
```

```r
add.constraint(lpobj2, c(1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
add.constraint(lpobj2, c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
add.constraint(lpobj2, c(-1, 0, 0, -1, 0, 0, -1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, -1, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0,
add.constraint(lpobj2, c(0, 0, -1, 0, 0, -1, 0, 0, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 0,

set.bounds(lpobj2, lower = c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,

solve(lpobj2)
```

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

```r
get.objective(lpobj2)
```

```
## [1] 1963.82
```

```r
get.variables(lpobj2)
```

```
##  [1] 93  0  0  0 86  0 28  0 67 30  0  0 91  0  0 57 29  0  0  0  0 19  0 48  2
## [26]  0  0
```

```r
#Well 1 & Well 3 are used to capacity
#Well 2 is not
1963.82 * 1000
```

```
## [1] 1963820
```

```r
# $1,963,820 is minimum cost
```