Jnoxon_2.R

Jason

2021-10-03

```
library(lpSolveAPI)
library(ggplot2)


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

# Problem 1

#X = Fulltime Exmployees
#Y Part time employees
#3 different full-time employee shifts
#4 different part-time employee shifts

# Part A
#objective function:
# 14X1 + 14X2 + 14X3 + 12Y1 + 12Y2 + 12Y3 + 12Y4
#Constraints:
  #Make sure minimum number met
# X1 + Y1 >= 4
# X1 + X2 + Y2 >= 8
# X2 + X3 + Y3 >= 10
# X3 + Y4 >= 6
  #Make sure there is at least a 1:1 ratio of fulltime to part time
# X1 >= Y1
# X1 + X2 >= Y2
# X2 + X3 >= Y3
# X3 >= Y4

# Part B
# everything stays the same except the ratio of fulltime to part time becomes:
# X1 > Y1
# X1 + X2 > Y2
# X2 + X3 > Y3
# X3 > Y4
# due to the 1 hour lunch break, there will always have to be more fulltime
#than part time since there has to be at least as many fulltime employees on
#duty as part time employees


# Problem 2
#back savers problem
```

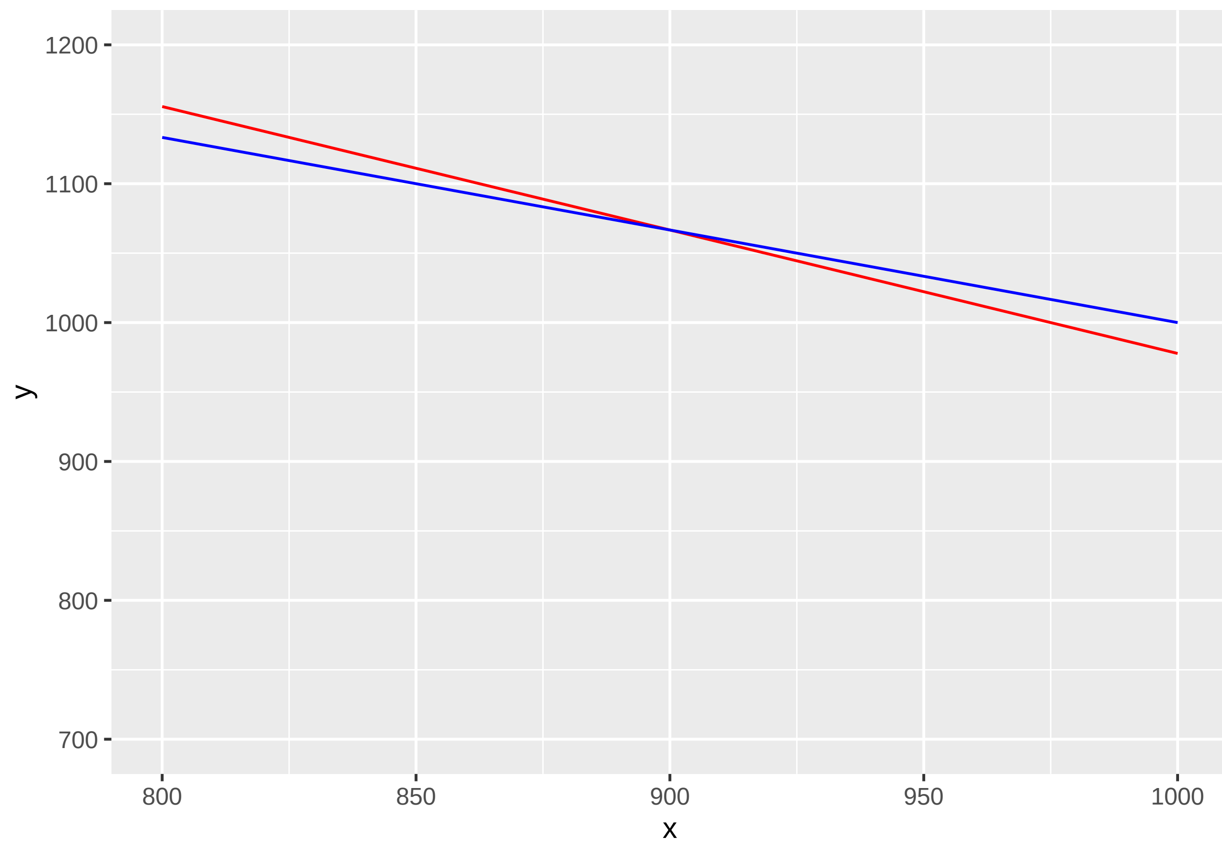
```r
#back savers has 2 products: Collegiate & Mini
#back savers receives 5000 square feet of nylon per week
# EAch Collegiate requires 3 square feet of nylon & 45 minutes of labor
# Each Mini requires 2 square feet of nylon & 40 minutes of labor
# back savers has 35 laborers at 40 hours per week
# 1000 Collegiate max & 1200 mini max

#converting 40 hours/week to minutes per week
#multiplying by 35 for each employee
AvailableLabor <- 40*60*35
AvailableLabor
```

```
## [1] 84000
```

```r
#labor is the only unit that needs to be normalized
#Nylon available & nylon required per product are both in square feet

# X = Collegiate
# Z = Mini
#Labor Constraint = 45*X + 40*Z <= 84,000
#Nylon Constraint = 3*X + 2*Z <= 5,000
#Product Constraint X<=1000
#Product Constraint Z<=1,200
#Objective Function = 32*X + 24*Z


cons.1 <- function(x) (84000 - 40*x)/45
cons.2 <- function(x) (5000 - 2*x)/3


# Build plot
graphsolver <- ggplot(data = data.frame(x = 0), aes(x = x)) +


  # Add constraints lines
  stat_function(colour = "Red", fun = cons.1) +
  stat_function(colour = "Blue", fun = cons.2) +
  geom_vline(xintercept = 32, colour = "Green") +
  geom_hline(yintercept = 24, colour = "Yellow") +

  # Specify axes breaks and limits
  scale_x_continuous(lim = c(800, 1000)) +
  scale_y_continuous(lim = c(700, 1200))

print(graphsolver)
```

```
## Warning: Removed 1 rows containing missing values (geom_vline).
```

```
## Warning: Removed 1 rows containing missing values (geom_hline).
```




```
# Using nylon and labor constraints
#optimal collegiate is roughly 900
#optimal mini is roughly 1075

#Problem 3

#Maximize Profit:
# 420X1 + 360X2 + 300X3 + 420X4 + 360X5+ 300X6 + 420X7 + 360X8+ 300X9
# Plant 1 <=750
# Plant 2 <=900
# Plant 3 <=450
# Space 1: 13000 >= 20X1 + 15X2 + 12X3
# Space 2: 12000 >= 20X1 + 15X2 + 12X3
# Space 3: 5000 >= 20X1 + 15X2 + 12X3
# X1 <= 900 max demand
# X2 <= 1200 max demand
# X3 <= 750 max demand
# X1/750 = X2/900 = X3/450

lpobj1 <- make.lp(0, 9)

set.objfn(lpobj1, c(420, 360, 300, 420, 360, 300, 420, 360, 300))

lp.control(lpobj1, sense = "max")
```

```
## $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] "maximize"
##
## $simplextype
## [1] "dual"   "primal"
##
## $timeout
## [1] 0
##
## $verbose
## [1] "neutral"
```

```
add.constraint(lpobj1, c(20, 15, 12, 0, 0, 0, 0, 0, 0), "<=", 13000)
add.constraint(lpobj1, c(0, 0, 0, 20, 15, 12, 0, 0 , 0), "<=", 12000)
add.constraint(lpobj1, c(0, 0, 0, 0, 0, 0, 20, 15, 12), "<=", 5000)
add.constraint(lpobj1, c(1, 0, 0, 1, 0, 0, 1, 0, 0), "<=", 900) #prod 1 demand
add.constraint(lpobj1, c(0, 1, 0, 0, 1, 0, 0, 1, 0), "<=", 1200) #prod 2 demand
add.constraint(lpobj1, c(0, 0, 1, 0, 0, 1, 0, 0, 1), "<=", 750) #prod 3 demand
add.constraint(lpobj1, c(1, 1, 1, 0, 0, 0, 0, 0, 0), "<=", 750) #plant 1 capacity
add.constraint(lpobj1, c(0, 0, 0, 1, 1, 1, 0, 0, 0), "<=", 900) #plant 2 capacity
add.constraint(lpobj1, c(0, 0, 0, 0, 0, 0, 1, 1, 1), "<=", 450) #plant 3 capacity


#the below code is meant to address this part of the problem:
#"To avoid layoffs if possible, management has decided that the plants should use the same percentage o
#The below code made the problem unsolvable
#Idk how else to address this constraint
#add.constraint(lpobj1, c(1/750, 1/750, 1/750, 0, 0, 0, 0, 0, 0), "=", c(0, 0, 0, 1/900, 1/900, 1/900, 
#add.constraint(lpobj1, c(1/750, 1/750, 1/750, 0, 0, 0, 0, 0, 0), "=", c(0, 0, 0, 0, 0, 0, 1/450, 1/450
#add.constraint(lpobj1, c(0, 0, 0, 1/900, 1/900, 1/900, 0, 0, 0), "=", c(1/750, 1/750, 1/750, 0, 0, 0, 
#add.constraint(lpobj1, c(0, 0, 0, 1/900, 1/900, 1/900, 0, 0, 0), "=", c(0, 0, 0, 0, 0, 0, 1/450, 1/450
#add.constraint(lpobj1, c(0, 0, 0, 0, 0, 0, 1/450, 1/450, 1/450), "=", c(0, 0, 0, 1/900, 1/900, 1/900, 
#add.constraint(lpobj1, c(0, 0, 0, 0, 0, 0, 1/450, 1/450, 1/450), "=", c(1/750, 1/750, 1/750, 0, 0, 0, 



lpobj1
```

```
## Model name:
##   a linear program with 9 decision variables and 9 constraints
```

```
#non zero
set.bounds(lpobj1, lower = c(0,0,0,0,0,0,0,0, 0), columns = c(1, 2, 3, 4, 5, 6, 7, 8, 9))

solve(lpobj1)
```

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

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

```
## [1] 708000
```

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

```
## [1] 350.0000 400.0000   0.0000   0.0000 400.0000 500.0000   0.0000 133.3333
## [9] 250.0000
```