MIS 64018 - Assignment 2

Tejasvini Mavuleti

2022-09-22

Say there are 3 plants - Plant 1, Plant 2, Plant 3 and three sizes S, M, L, we need to maximize the total net profit per day

Objective function

max: +420 P 1L +420 P 2L +420 P 3L +360 P 1M +360 P 2M +360 P 3M +300 P 1S +300 P 2S +300 P 3S

The constraints are

Storage P 1 Storage: +20 P 1L +15 P 1M +12 P 1S <= 13000 P 2 Storage: +20 P 2L +15 P 2M +12 P 2S <= 12000 P 3 Storage: +20 P 3L +15 P 3M +12 P 3S <= 5000

Capacity P 1 Capacity: +P 1L +P 1M +P 1S <= 750 P 2 Capacity: +P 2L +P 2M +P 2S <= 900 P 3 Capacity: +P 3L +P 3M +P 3S <= 450

Sales Large Sales: +P 1L +P 2L +P 3L <= 900 Medium Sales: +P 1M +P 2M +P 3M <= 1200 Small Sales: +P 1S +P 2S +P 3S <= 750

Together P 1 and 2 Usage: +900 P 1L -750 P 2L +900 P 1M -750 P 2M +900 P 1S -750 P 2S = 0 P 2 and 3 Usage: +450 P 2L -900 P 3L +450 P 2M -900 P 3M +450 P 2S -900 P 3S = 0 P 1 and 3 Usage: +450 P 1L -750 P 3L +450 P 1M -750 P 3M +450 P 1S -750 P 3S = 0

```
# Using the lpSolve packages to solve the problem
library(lpSolve)

## Warning: package 'lpSolve' was built under R version 4.2.1

library(lpSolveAPI)

## Warning: package 'lpSolveAPI' was built under R version 4.2.1
```

#Create a linear programming object using 9 decision variables, 12 constraints, and minimum boundary conditions for each variable

```
# Running the linear programming problem with 12 constraints and 9 decision variables
f.obj <- make.lp(12,9)
```

Now we set the objective function to find the maximum profits for all the three plants.

```r
# Set the objective function
set.objfn(f.obj, c(420,420,420,360,360,360,300,300,300))

# Change the direction to set maximization
lp.control(f.obj, 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"
```

The next step is when we add all the constraints to the problem and fit it to model. Here, we use set.row to remove all the 0 values.

```
# Add the constraint values to every row

# Storage
set.row(f.obj, 4, c(20,15,12), indices = c(1,4,7))
set.row(f.obj, 5, c(20,15,12), indices = c(2,5,8))
set.row(f.obj, 6, c(20,15,12), indices = c(3,6,9))

# Capacity
set.row(f.obj, 1, c(1,1,1), indices = c(1,4,7))
set.row(f.obj, 2, c(1,1,1), indices = c(2,5,8))
set.row(f.obj, 3, c(1,1,1), indices = c(3,6,9))

# Capacity usage
set.row(f.obj, 10, c(900,900,900,-750,-750,-750), indices = c(1,4,7,2,5,8))
set.row(f.obj, 11, c(450,450,450,-900,-900,-900), indices = c(2,5,8,3,6,9))
set.row(f.obj, 12, c(450,450,450,-750,-750,-750), indices = c(1,4,7,3,6,9))

# Sales
set.row(f.obj, 7, c(1,1,1), indices = c(1,2,3))
set.row(f.obj, 8, c(1,1,1), indices = c(4,5,6))
set.row(f.obj, 9, c(1,1,1), indices = c(7,8,9))
```

This step is where we split the problem into two parts - lhs and rhs. These values need to match up to all our constraints to all sizes in all the three plants

```
# Calculate the right hand side values for the objective function
f.rhs <- c(750,900,450,13000,12000,5000,900,1200,750,0,0,0)
set.rhs(f.obj, f.rhs)
```

Now start to fir the maximum capability of our constraints to be less than or equal to our problem and then the optimizing function. Eliminate the inequalities

```
# Set the constraint type
set.constr.type(f.obj,
c("<=","<=","<=","<=","<=","<=","<=","<=","<=","=","=","="))
```

We need to make sure that that all our values should be great than 0 to get the correct value.

```
# Adding limits for the decision variables
set.bounds(f.obj, lower = rep(0, 9))
```

This set of code will name the decision variables and the constraints for the model.

```
# Now add the names of all the constraints and decision variables
lp.rownames <- c("P 1 Capacity", "P 2 Capacity", "P 3 Capacity", "P 1
Storage", "P 2 Storage", "P 3 Storage", "Large Sales", "Medium Sales", "Small
Sales", "P 1 and 2 Usage", "P 2 and 3 Usage", "P 1 and 3 Usage")
lp.colnames <- c("P 1L", "P 2L", "P 3L", "P 1M", "P 2M", "P 3M", "P 1S", "P
2S", "P 3S")
dimnames(f.obj) <- list(lp.rownames, lp.colnames)
```

Now review and check if all the values and constraints are correctly added.

```
# Double checking the linear programming object
f.obj

## Model name:
##   a linear program with 9 decision variables and 12 constraints
```

Find the optimal function and if it is 0 then the result is that there is a correct optimal solution.

```
# Solve the linear program again
solve(f.obj)

## [1] 0
```

The result is 0, so the objective function will show the maximum profit

```
# The optimal solution is
get.objective(f.obj)

## [1] 696000
```

The maximum profit for all the plants including the constraints and decision variables is $696,000 per day.

Now we need to see the values of the decision variables to and plan how many units of each size of product every plant should make.

```
# The number of units using the decision variables is
get.variables(f.obj)
```

```
## [1] 516.6667   0.0000   0.0000 177.7778 666.6667   0.0000   0.0000
166.6667
## [9] 416.6667
```

Optimum decision variable values from the model:

Plant 1, Large: 516.67 units/day Plant 2, Large: 0 units/day Plant 3, Large: 0 units/day Plant 1, Medium: 177.78 units/day Plant 2, Medium: 666.67 units/day Plant 3, Medium: 0 units/day Plant 1, Small: 0 units/day Plant 2, Small: 166.67 units/day Plant 3, Small: 416.67 units/day

The following two segments of code will tell us where our values fall within the constraints, as well as return the surplus between the constraint and the actual value from the constraints.

```
# Figure out the constraint values for all the plants
get.constraints(f.obj)
```

```
##  [1]   694.4444   833.3333   416.6667 13000.0000 12000.0000  5000.0000
##  [7]   516.6667   844.4444   583.3333     0.0000     0.0000     0.0000
```

There is going to ba a surplus after production

```
# Calculating the rhs
get.constraints(f.obj) - f.rhs
```

```
##  [1] -5.555556e+01 -6.666667e+01 -3.333333e+01  0.000000e+00  0.000000e+00
##  [6] -9.094947e-13 -3.833333e+02 -3.555556e+02 -1.666667e+02  0.000000e+00
## [11]  0.000000e+00  0.000000e+00
```

Therefore there is a scope of maximizing the profits and the management can have a clear picture of manufacturing bags in all the three plants without laying off employees.