Assignment Module_4

Nemin Dholakia

9/23/2022

R Markdown

```
##Load the "lpSolveAPI" package into the R environment.
library(lpSolveAPI)

## Warning: package 'lpSolveAPI' was built under R version 4.1.3
```

We must first make a linear programming object. This program will include 9 decision variables, 12 constraints, and minimal boundary conditions for each variable based on our model from steps A and B.

```
##Begin the lp problem with 12 constraints and 9 decision variables.
lprec <- make.lp(12,9)
```

The objective function must then be added to our program. In this scenario, we will need to locate the maximum profits, thus we will also adjust the objective function to look for the maximum value. By default, the program sets the objective function to find the minimum.

```
##Set the objective function for the problem.
set.objfn(lprec, c(420,420,420,360,360,360,300,300,300))
##Change the direction to set maximization
lp.control(lprec, 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 program will also need to include all of the constraint values. The limitations on capacity, storage, sales, and capacity consumption that were previously outlined in parts A

and B are these. These will be entered by specifying the index to place the values at and using the "set.row" command. This will spare us from manually entering numerous "0" values.

```
##Set the constraint values row by row
##Capacity constraints:
set.row(lprec, 1, c(1,1,1), indices = c(1,4,7))
set.row(lprec, 2, c(1,1,1), indices = c(2,5,8))
set.row(lprec, 3, c(1,1,1), indices = c(3,6,9))
##Storage constraints:
set.row(lprec, 4, c(20,15,12), indices = c(1,4,7))
set.row(lprec, 5, c(20,15,12), indices = c(2,5,8))
set.row(lprec, 6, c(20,15,12), indices = c(3,6,9))
##Sales constraints:
set.row(lprec, 7, c(1,1,1), indices = c(1,2,3))
set.row(lprec, 8, c(1,1,1), indices = c(4,5,6))
set.row(lprec, 9, c(1,1,1), indices = c(7,8,9))
##Capacity usage constaints:
set.row(lprec, 10, c(900,900,900,-750,-750,-750), indices = c(1,4,7,2,5,8))
set.row(lprec, 11, c(450,450,450,-900,-900,-900), indices = c(2,5,8,3,6,9))
set.row(lprec, 12, c(450,450,450,-750,-750,-750), indices = c(1,4,7,3,6,9))
```

The constraint values from the problem statement must now be set. These statistics here represent the limits on capacity and storage at each factory, the limits on sales for each size, and the percentage of capacity utilized at each plant.

```
##Set the right hand side values
rhs <- c(750,900,450,13000,12000,5000,900,1200,750,0,0,0)
set.rhs(lprec, rhs)
```

The inequality values for the problem must then be defined. The majority of them will be less than or equal to because the problem's description included the maximum capacity restrictions.

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

For this problem, all values must be greater than 0.

```
##Set the boundary condiiton for the decision variables
set.bounds(lprec, lower = rep(0, 9))
```

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

```
##Set the names of the rows (constraints) and columns (decision variables)
lp.rownames <- c("Plant 1 Capacity", "Plant 2 Capacity", "Plant 3 Capacity",
"Plant 1 Storage", "Plant 2 Storage", "Plant 3 Storage", "Large Sales",
"Medium Sales", "Small Sales", "Plant 1 and 2 Usage", "Plant 2 and 3 Usage",
"Plant 1 and 3 Usage")
lp.colnames <- c("Plant 1L", "Plant 2L", "Plant 3L", "Plant 1M", "Plant 2M",
```

```
"Plant 3M", "Plant 1S", "Plant 2S", "Plant 3S")
dimnames(lprec) <- list(lp.rownames, lp.colnames)
```

Before running the code, this command should produce the linear program outline so we can verify that all the values are accurate.

```
##Return the linear programming object to ensure the values are correct
lprec

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

##The model can also be saved to a file
write.lp(lprec, filename = "Assignment_2_Problem3C.lp", type = "lp")
```

The following code will now look for an optimal solution. If it returns a "0" value then that means the model has found an optimal solution.

```
##Solve the linear program
solve(lprec)

## [1] 0
```

The model returned a "0", so it has found an optimal solution to the problem.

The function below will return what the maximum value for the objective function will be.

```
##Review the objective function value
get.objective(lprec)

## [1] 696000
```

In this case, the maximum profits that can be achieved with these constraints is $696,000 per day.

In order to determine how many units of each kind of product the plants should produce, we will then return the values of the decision variables.

```
##Get the optimum decision variable values
get.variables(lprec)

## [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 next two lines of code will both return the surplus between the constraint and the actual value resulting from the constraints as well as indicate where our values lie within the constraints.

```
##Get the constraint values for the problem
get.constraints(lprec)

##  [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

##Review the surplus for each constraint
get.constraints(lprec) - 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
```