Assignment\_2

Swetha

9/20/2022

# Loading the "lpSolve","lpSolveAPI" packages.  
library(lpSolve)

## Warning: package 'lpSolve' was built under R version 4.1.3

library(lpSolveAPI)

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

# setting up the working directory  
getwd()

## [1] "C:/Users/mercy/OneDrive/Desktop/QMM/Assignment\_2"

setwd("C:/Users/mercy/OneDrive/Desktop/QMM/Assignment\_2")

# From the LP problem we have to find objective function, constraints and decision variables.

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

# Setting the objective function for the problem.  
set.objfn(lprec, c(420,420,420,360,360,360,300,300,300))  
# and Changing the direction to setup 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"

# Setting up all 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))

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

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

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

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

# Setting the names of the rows as constraints and the columns as 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 return the linear program outline so we can verify that all the values are valid.

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

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

# This model can also be saved to a file  
write.lp(lprec, filename = "Assignment-2.lp", type = "lp")

This following code will now look for an optimal solution. If it returns “0” value then that means the model has found an optimal solution.

# Solving the linear program  
solve(lprec)

## [1] 0

This 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.

# Reviewing 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

The 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.

# 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

# Reviewing 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

The “get.sensitivity.rhs()” function will be used to identify the shadow prices for the right hand side and objective function of the formulated linear programming problem.

get.sensitivity.rhs(lprec)

## $duals  
## [1] 0.00 0.00 0.00 12.00 20.00 60.00 0.00 0.00 0.00  
## [10] -0.08 0.00 0.56 0.00 -40.00 -360.00 0.00 0.00 -120.00  
## [19] -24.00 0.00 0.00  
##   
## $dualsfrom  
## [1] -1.000000e+30 -1.000000e+30 -1.000000e+30 1.122222e+04 1.150000e+04  
## [6] 4.800000e+03 -1.000000e+30 -1.000000e+30 -1.000000e+30 0.000000e+00  
## [11] -1.000000e+30 0.000000e+00 -1.000000e+30 -1.000000e+02 -2.000000e+01  
## [16] -1.000000e+30 -1.000000e+30 -4.444444e+01 -2.222222e+02 -1.000000e+30  
## [21] -1.000000e+30  
##   
## $dualstill  
## [1] 1.000000e+30 1.000000e+30 1.000000e+30 1.388889e+04 1.250000e+04  
## [6] 5.181818e+03 1.000000e+30 1.000000e+30 1.000000e+30 0.000000e+00  
## [11] 1.000000e+30 0.000000e+00 1.000000e+30 1.000000e+02 2.500000e+01  
## [16] 1.000000e+30 1.000000e+30 6.666667e+01 1.111111e+02 1.000000e+30  
## [21] 1.000000e+30

get.sensitivity.obj(lprec)

## $objfrom  
## [1] 3.60e+02 -1.00e+30 -1.00e+30 3.45e+02 3.45e+02 -1.00e+30 -1.00e+30  
## [8] 2.52e+02 2.04e+02  
##   
## $objtill  
## [1] 4.60e+02 4.60e+02 7.80e+02 4.20e+02 4.20e+02 4.80e+02 3.24e+02 3.24e+02  
## [9] 1.00e+30

## Dual Problem:

# Begining the dual lp problem with 9 constraints and 12 decision variables.  
lprecdual <- make.lp(9,12)

# Setting up the objective function for the problem.  
set.objfn(lprecdual, c(750,900,450,13000,12000,5000,900,1200,750,0,0,0))  
# Changing the direction to set minimization  
lp.control(lprecdual, 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"

# Setting up all the constraint values row by row  
set.row(lprecdual, 1, c(1,20,1,900,450), indices = c(1,4,7,10,12))  
set.row(lprecdual, 2, c(1,15,1,900,450), indices = c(1,4,8,10,12))  
set.row(lprecdual, 3, c(1,12,1,900,450), indices = c(1,4,9,10,12))  
set.row(lprecdual, 4, c(1,20,1,-750,450), indices = c(2,5,7,10,11))  
set.row(lprecdual, 5, c(1,15,1,-750,450), indices = c(2,5,8,10,11))  
set.row(lprecdual, 6, c(1,12,1,-750,450), indices = c(2,5,9,10,11))  
set.row(lprecdual, 7, c(1,20,1,-900,-450), indices = c(3,6,7,11,12))  
set.row(lprecdual, 8, c(1,15,1,-900,-450), indices = c(3,6,8,11,12))  
set.row(lprecdual, 9, c(1,12,1,-900,-450), indices = c(3,6,9,11,12))

# Setting all the right hand side values  
rhs2 <- c(420,420,420,360,360,360,300,300,300)  
set.rhs(lprecdual, rhs2)

# Now Setting the constraint type  
set.constr.type(lprecdual, c(">=",">=",">=",">=",">=",">=",">=",">=",">="))

# Setting up the boundary condiiton for the decision variables  
set.bounds(lprecdual, lower = c(0,0,0,0,0,0,0,0,0,-1.00e+30,-1.00e+30,-1.00e+30))

# Solving the linear program  
solve(lprecdual)

## [1] 0

# Reviewing the objective function value  
get.objective(lprecdual)

## [1] 0

# Getting the optimum decision variable values  
get.variables(lprecdual)

## [1] 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000 0.000000  
## [8] 0.000000 0.000000 -2.400000 -3.200000 5.733333

get.sensitivity.rhs(lprecdual)

## $duals  
## [1] 0 0 0 0 0 0 0 0 0 750 900 450  
## [13] 13000 12000 5000 900 1200 750 0 0 0  
##   
## $dualsfrom  
## [1] 4.200000e+02 -1.000000e+30 -1.000000e+30 -1.000000e+30 3.600000e+02  
## [6] -1.000000e+30 -1.000000e+30 -1.000000e+30 3.000000e+02 -1.000000e+30  
## [11] -1.000000e+30 -1.000000e+30 -1.000000e+30 -2.382043e-10 -7.914723e-11  
## [16] -6.331779e-10 -1.512103e-09 -1.511488e-09 -1.000000e+30 -1.000000e+30  
## [21] -1.000000e+30  
##   
## $dualstill  
## [1] 1.000000e+30 1.000000e+30 1.000000e+30 1.000000e+30 1.000000e+30  
## [6] 1.000000e+30 1.000000e+30 1.000000e+30 1.000000e+30 1.032000e+03  
## [11] 7.199263e+02 1.152000e+03 1.889360e-10 4.799508e+01 9.600000e+01  
## [16] 1.511488e-09 1.191021e-09 6.331779e-10 1.000000e+30 1.000000e+30  
## [21] 1.000000e+30

get.sensitivity.obj(lprecdual)

## $objfrom  
## [1] 0 0 0 0 0 0 0 0 0 0 0 0  
##   
## $objtill  
## [1] 1.00e+30 1.00e+30 1.00e+30 1.00e+30 1.00e+30 1.00e+30 1.00e+30 1.00e+30  
## [9] 1.00e+30 1.00e+30 1.00e+30 1.17e+05