Operations Research

Laboratory Session 5: Solving linear models with `lpSolveAPI`

The aim of this laboratory session is to learn how to use an R implementation used to solve optimization models: the `lpSolveAPI` package. For more information, have a look at the reference manual (https://cran.r-project.org/web/packages/lpSolveAPI/index.html) and lp_solve 5.5.2.5, a Mixed Integer Linear Programming (MILP) solver (http://lpsolve.sourceforge.net/5.5/index.htm).

You can download and install it directly from RStudio.

```
install.packages("lpSolveAPI")
library(lpSolveAPI)
```

Once the package is installed and loaded, you can go to RStudio "Packages" and click on the name of the package to see the help pages of all the functions defined in it. We'll use the package to solve linear problems, interpret the shadow prices and analyze some discrete changes in the model (sensitivity analysis). Let us consider a linear problem as an example (Linear modeling unit, exercise 12).

An example: a production problem

A company manufactures 3 brands of fertilizers, $F_{5-10-5}, F_{5-10-10}$ and $F_{10-10-10}$. The Nitrate, Sulphur and Potash contents (in percentages) of these brands are 5-10-5, 5-10-10 and 10-10-10 respectively. The rest of the content is an inert matter, which is available in abundance.

The company has 1000 tons of Nitrates, 1800 tons of Sulphur, and 1200 tons of Potash available. They were bought at different prices. Costs per ton: Nitrate (1600), Sulphur (400), Potash (1000), Inert matter (50).

The production cost is the same for the three brands of fertilizer: 150 units per ton. However, the three fertilizers are sold at a different price per ton: F_{5-10-5} (400), $F_{5-10-10}$ (500), $F_{10-10-10}$ (600).

The company has to satisfy a demand of 6000 tons of fertilizer F_{5-10-5}. The objective is to maximize the total profit. The following linear model helps to decide how much of each brand should the company manufacture.

$$
\begin{aligned}
&\max\ z = 40x_1 + 92.5x_2 + 115x_3 \\
&\text{subject to} \\
&0.05x_1 + 0.05x_2 + 0.1x_3 \leq 1000 \quad \text{(Nitrate)} \\
&0.1x_1 + 0.1x_2 + 0.1x_3 \leq 1800 \quad \text{(Sulphur)} \\
&0.05x_1 + 0.1x_2 + 0.1x_3 \leq 1200 \quad \text{(Potash)} \\
&x_1 \geq 6000 \\
&x_1, x_2, x_3 \geq 0
\end{aligned}
$$

How to solve linear models with `lpSolveAPI` package

Let us observe some useful functions defined in the package.

Defining a linear model

First, the linear model has to be defined. Some very useful functions are the following: `make.lp` creates a new `lpSolve` linear program model object, `set.column` sets a column in it, `set.objfn` sets the objective function, `set.constr.value` sets the constraint values of vector **b** and `set.constr.type` sets the constraint types. `dimnames` can be used to name variables and constraints. Note that since `set.column` affects the

coefficients in the objective function, it is recommended to set columns in matrix **A** before setting the objective function coefficients (vector **c**). The nonnegativity of variables is the default. Take into account that the implementation is designed to minimize the objective function.

```
rm(list=ls())
model.example <- make.lp(nrow=4, ncol=3)
model.example
set.objfn(lprec=model.example, obj=c(-40, -92.5, -115), indices=1:3)
model.example
set.column(lprec=model.example, column=1, x=c(0.05, 0.1, 0.05, 1))
set.column(lprec=model.example, column=2, x=c(0.05, 0.1, 0.1, 0))
set.column(lprec=model.example, column=3, x=c(0.1, 0.1, 0.1, 0))
model.example
set.objfn(lprec=model.example, obj=c(-40, -92.5, -115), indices=1:3)
model.example
set.constr.value(lprec=model.example, rhs=c(1000, 1800, 1200, 6000), constraints=1:4)
set.constr.type(model.example, types=c("<=", "<=", "<=", ">="), constraints=1:4)
model.example
variables   <- c("x1","x2","x3")
constraints <- c("Constraint 1", "Constraint 2", "Constraint 3", "Constraint 4")
dimnames(model.example) <- list(constraints, variables)
model.example
```

It is also possible to read an `lpSolve` linear model object from a file using the `read.lp` function. If we use the "lp" type format, the model looks like this:

```
max: 40x1 + 92.5x2 + 115x3;
Nitrate: 0.05x1 + 0.05x2 + 0.1x3 <= 1000;
Sulphur: 0.1x1 + 0.1x2 + 0.1x3 <= 1800;
Potash: 0.05x1 + 0.1x2 + 0.1x3 <= 1200;
Demand: x1>=6000;
```

Saved in a file, we can load it to RStudio:

```
rm(list=ls())
model.example <- read.lp(file="model_exercise_12.lp", type="lp", verbose="full")
model.example
```

Solving a linear model

Function `solve` can be used to solve a linear model, once the object has been created. For more information, type `?solve.lpExtPtr` on RStudio "help" window. An integer value that contains the status code is returned; if 0 is returned, this means that the optimal solution was found. Moreover, some more information is shown on the screen. To see the optimal solution and optimal objective value, functions `get.variables` and `get.objective` can be used.

```
?solve.lpExtPtr
solve(model.example)
get.variables(model.example)
get.objective(model.example)
message("Optimal solution: ", get.variables(model.example),
        "; Optimal objective value=", get.objective(model.example))
```

The optimal solution for the production problem we are considering is $x_1^* = 6000$, $x_2^* = 4000$, $x_3^* = 5000$ which means that the enterprise should produce 6000 tons of fertilizer F_{5-10-5}, 4000 tons of $F_{5-10-10}$ and 5000 tons of $F_{10-10-10}$. This optimal production gives a maximum profit of 1185000.

Shadow prices interpretation

Thanks to the shadow prices (economical interpretation of the dual solution) we can observe the use of the resources, and are suggested about the price that should be paid at most for each unit of resource, if the enterprise decides to buy some more units of any limiting resource.

To do the shadow prices interpretation we need to extract the optimal solution to the dual. To check the interpretation, we'll change vector **b** in the linear model and solve it again. We'll also check the constraints to see the use of the resources.

Some very useful functions are: `get.sensitivity.rhs` to obtain the values of the dual variables and the range of values for the right-hand-side vector (upper and lower limits), `set.rhs` to set elements on the right-hand-side and `get.constraints` to check the constraints.

```
rm(list=ls())
model.example <- read.lp(file="model_exercise_12.lp", type="lp", verbose="full")
model.example
solve(model.example)
get.variables(model.example)
z_opt <- get.objective(model.example)
z_opt
dual <- get.sensitivity.rhs(model.example)
dual
dual$duals

# Shadow price interpretation y1^* related to resource b1 (Nitrate)
get.rhs(model.example)
set.rhs(model.example, c(1001, 1800, 1200, 6000))
get.rhs(model.example)
solve(model.example)
get.variables(model.example)
z_opt_new <- get.objective(model.example)
z_opt_new
z_opt
dual$duals[1]
z_opt_new == z_opt + dual$duals[1]

# Shadow price interpretation y2^* related to b2 (Sulphur).
set.rhs(model.example, c(1000, 1799, 1200, 6000))
get.rhs(model.example)
solve(model.example)
get.variables(model.example)
z_opt_new <- get.objective(model.example)
z_opt_new
z_opt
dual$duals[2]
z_opt_new == z_opt - dual$duals[2]

get.constraints(model.example)
```

Questions

- Write the dual optimal solution, $y_1^* =$, $y_2^* =$, $y_3^* =$, $y_4^* =$, and explain its meaning. Note that some variables are negative. Explain their interpretation.

- Interpret the shadow prices y_1^* and y_2^*.

- Check the constraints and reason about the use of the resources (consider the optimal solution obtained).

Sensitivity analysis

Shadow prices interpretation refers to unitary changes in the resources. To analyze general changes in vector **b** and how they affect to the optimal solution and benefit, we can use the sensitivity analysis.

Changes in vector b

First, it is interesting to observe the ranges of values (upper and lower limits) in which the current basis remains unchanged (primal feasibility is not lost) for each element in the right-hand-side vector **b**. In fact, shadow prices interpretation can only be done for changes of the components of vector **b** that are inside these ranges of values. As soon as the primal feasibility is lost, it makes no sense to try to interpret the dual solution (shadow prices), since it is not optimal any more.

Some very useful functions are: `get.sensitivity.rhs` computes the ranges of values. Functions `set.rhs` `solve`, `get.variables` and `get.objective` can be used to change the model, solve it and obtain the optimal solution of any change in vector **b**.

```
rm(list=ls())
model.example <- read.lp(file="model_exercise_12.lp", type="lp", verbose="full")
solve(model.example)
get.variables(model.example)
get.objective(model.example)
dual <- get.sensitivity.rhs(model.example)
dual
dual$duals
dual$dualsfrom

dual$dualstill


set.rhs(model.example, 1100, 1)
solve(model.example)
get.variables(model.example)
z_opt_new <- get.objective(model.example)
z_opt_new
```

Questions

- For the first resource (Nitrate), what is the range of values in which the current basis remains unchanged?

- What is the optimal solution if the company has 1100 tons of Nitrates (first resource) instead of 1000?

$x_1^*(F_{5-10-5}) =$, $x_2^*(F_{5-10-10}) =$, $x_3^*(F_{10-10-10}) =$.

What is the optimal benefit? $z^* =$. Reason about the results obtained in terms of shadow prices and taking into account the range of values for b_1.

Changes in vector c

Changes in vector **c** and their effect on the optimal solution and benefit can also be studied using the package functions. Have a look at function `get.sensitivity.obj`, observe the results obtained for the example and answer the following questions.

```
rm(list=ls())
model.example <- read.lp(file="model_exercise_12.lp", type="lp", verbose="full")
solve(model.example)
get.variables(model.example)
get.objective(model.example)
get.sensitivity.obj(model.example)

set.objfn(model.example, c(40, 92.5, 180), 1:3)
solve(model.example)
get.variables(model.example)
get.objective(model.example)
```

Questions

- For the third coefficient in the objective function (c_3, unitary benefit obtained from the production of $F_{10-10-10}$), what is the range of values in which the current basis remains unchanged?

- What is the optimal solution if the unitary benefit obtained from the production of the third type of fertilizer ($F_{10-10-10}$) is $c_3 = 180$ instead of $c_3 = 115$? What is the optimal benefit? Reason about the results obtained.

$x_1^*(F_{5-10-5}) =$, $x_2^*(F_{5-10-10}) =$, $x_3^*(F_{10-10-10}) =$, $z^* =$.

Exercises

Now it is your turn to use the appropriate functions to solve the problem and answer the question. Let us continue with the same model (linear modeling unit, model 12) presented in the introduction of this lab session.

Exercise 1.

If the company needs to satisfy a larger demand of fertilizer F_{5-10-5} (6001 tons instead of 6000 tons) will this change affect (increase/decrease) the total profit? Does it affect to the optimal solution? Justify your answer in terms of shadow prices interpretation.

Reasoning:

Exercise 2.

Assume that there is place enough in the store for 300 more tons of resource. Which one of the three (Nitrate, Sulphur, Potash) would you select to increase its availability? Why?

Reasoning:

Exercise 3.

Suddendly, everybody wants to buy the first type of fertilizer (F_{5-10-5}). Consequently, it is sold at a higher price now for the clients and therefore the benefit obtained by the company is of 100 instead of 40. What is the optimal production now? And the optimal benefit? Reason about the results obtained taking into account the range of values and the use of the resources.

Reasoning:

Integer Programming

The `lpSolveAPI` package can also be used to solve integer linear problems. Let us consider an integer problem as an example (Integer Programming unit, exercise 4).

An integer problem

Six components, $C_1, C_2, C_3, C_4, C_5, C_6$, must be carried in a box that can hold up to 15 kg. The values (euros) associated to each of the components are 4, 2, 1, 7, 3, 6 respectively and the weights (kg) are 5, 8, 8, 6, 1, 5. At least 3 components must be carried in the box. Since the objective is to maximize the total value of the components introduced in the box and the weight capacity does not allow to carry them all, we need to choose some of them. Six binary variables are defined to decide if component C_i is selected to be carried in the box. We need to determine which of the 6 components will be selected to be carried in the box so as to maximize the total value of the selected components. The 0-1 IP model that represents the problem is the following:

$$\begin{aligned} \max\ & z = 4x_1 + 2x_2 + x_3 + 7x_4 + 3x_5 + 6x_6 \\ \text{subject to} & \\ & 5x_1 + 8x_2 + 8x_3 + 6x_4 + x_5 + 5x_6 \leq 15 \\ & x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3 \\ & x_1, x_2, x_3, x_4, x_5, x_6 = 0 \text{ or } 1 \end{aligned}$$

We have already used the functions `make.lp` to create a new model, and `set.column`, `set.objfn`, `set.constr.value` and `set.constr.type` to introduce the coefficients and define the sense of the constraints, and the functions `solve`, `get.objective` and `get.variables` to solve the model and obtain the optimal solution. Function `set.type` can be used to declare that a variable is integer or binary.

Have a look at the following commands, observe the results obtained and answer the following questions.

```
model.binary <- make.lp(nrow=2, ncol=6)
set.column(lprec=model.binary, column=1, x=c(5, 1))
set.column(lprec=model.binary, column=2, x=c(8, 1))
set.column(lprec=model.binary, column=3, x=c(8, 1))
set.column(lprec=model.binary, column=4, x=c(6, 1))
set.column(lprec=model.binary, column=5, x=c(1, 1))
set.column(lprec=model.binary, column=6, x=c(5, 1))
set.constr.value(lprec=model.binary, rhs=c(15, 3), constraints=1:2)
set.constr.type(lprec=model.binary, types=c("<=", ">="), constraints=1:2)
set.type(model.binary, 1, type ="binary")
set.type(model.binary, 2, type ="binary")
set.type(model.binary, 3, type ="binary")
set.type(model.binary, 4, type ="binary")
set.type(model.binary, 5, type ="binary")
set.type(model.binary, 6, type ="binary")
set.objfn(lprec=model.binary, obj=c(-4, -2,-1,-7,-3,-6), indices=1:6)
model.binary
solve(model.binary)
get.objective(model.binary)
get.variables(model.binary)
get.total.nodes(model.binary)
get.solutioncount(model.binary)
```

Questions

- Which of the 6 components will be selected to be carried in the box? What is the maximum value of the box?

- Have a look at the help pages in RStudio or in the reference manual provided in CRAN to obtain some information about functions `get.total.nodes` and `get.solutioncount`. What is the algorithm used to solve the model? Observe the results obtained and interpret them.

Exercise

Now let us suppose that there are some units for each type of component; 4, 3, 5, 2, 5 and 1, respectively. What's the optimal solution now? And, what is the maximum value of the box? Have a look at the function `set.bounds`; it may be useful to use in this case.