Assignment 1 - Linear Programming

Andrea Piancone - 812250

Contents

The problem

A trading company is looking for a way to maximize profit per transportation of their goods. The company has a train available with 3 wagons.

When stocking the wagons they can choose among 4 types of cargo, each with its own specifications. How much of each cargo type should be loaded on which wagon in order to maximize profit?

More data

TRAIN WAGON j	WEIGHT CAPACITY (TONNE) w_j	VOLUME CAPACITY (m^2) s_j
(wag) 1	10	5000
(wag) 2	8	4000
(wag) 3	12	8000

CARGO TYPE i	AVAILABLE (TONNE) a_i	VOLUME (m^2) v_i	PROFIT (PER TONNE) p_i
(cg) 1	18	400	2000
(cg) 2	10	300	2500
(cg) 3	5	200	5000
(cg) 4	20	500	3500

The decision variables

The aim of the problem is find the optimal way to distribute each cargo type among the three wagons in order to maximize the profit. It is possible to distribute the tonnes of each type of product, among the three wagons in twelve combinations. Each combination cargo type-wagon represents a decision variable $x_{i,j}$, where i represents the index of each cargo type, while j represents the number of wagon. Therefore the decision

variables are:

- $x_{1,1}$ = tonnes of cargo type 1 loaded on the first wagon;
- $x_{1,2}$ = tonnes of cargo type 1 loaded on the second wagon;
- $x_{1,3}$ = tonnes of cargo type 1 loaded on the third wagon;
- $x_{2,1}$ = tonnes of cargo type 2 loaded on the first wagon;
- $x_{2,2}$ = tonnes of cargo type 2 loaded on the second wagon;
- $x_{2,3}$ = tonnes of cargo type 2 loaded on the third wagon;
- $x_{3,1}$ = tonnes of cargo type 3 loaded on the first wagon;
- $x_{3,2}$ = tonnes of cargo type 3 loaded on the second wagon;
- $x_{3,3}$ = tonnes of cargo type 3 loaded on the third wagon;
- $x_{4,1}$ = tonnes of cargo type 4 loaded on the first wagon;
- $x_{4,2}$ = tonnes of cargo type 4 loaded on the second wagon;
- $x_{4,3}$ = tonnes of cargo type 4 loaded on the third wagon.

The objective function

As mentioned above, the problem to be solved is a maximization problem, where the objective function is the profit given by the distribution of each cargo type. Regardless of the wagon on which the goods are loaded, their profit per tonne does not change. So, the objective function is:

$max\ 2000x_{1,1} + 2000x_{1,2} + 2000x_{1,3} + 2500x_{2,1} + 2500x_{2,2} + 2500x_{2,3} + 5000x_{3,1} + 5000x_{3,2} + 5000x_{3,3} + 3500x_{4,1} + 3500x_{4,2} + 3500x_{4,3}$

that can be compacted in the following form:

$max\ 2000 \sum_{j=1}^{3} x_{1,j} + 2500 \sum_{j=1}^{3} x_{2,j} + 5000 \sum_{j=1}^{3} x_{3,j} + 3500 \sum_{j=1}^{3} x_{4,j}$

The constraints

Weight constraints

From the problem's data, emerges that each wagon has a maximum capacity of tons which is capable of carrying. In particular, the first wagon is able to transporting a maximum of 10 tonnes, the second wagon is able to transporting 8 tonnes and the third wagon is able to support a maximum of 12 tones. Therefore, it is necessary to introduce weight constraints so that the allocation of the four categories of cargo, among the three wagons, is in line with the maximum tonnes they are capable of carrying.

$$\begin{cases} x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq 10 \\ x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 8 \\ x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 12 \end{cases}$$

Volume constraints

Each wagon, has a maximum volume capacity. In particular, the first wagon has a maximum capacity of $5000m^3$, the second wagon has a maximum capacity of $4000m^3$ and the third wagon has a maximum capacity of $8000m^3$. So, it is essential to introduce some constraints that model this criticality.

$$\begin{cases} 400x_{1,1} + 300x_{2,1} + 200x_{3,1} + 500x_{4,1} \leq 5000 \\ 400x_{1,2} + 300x_{2,2} + 200x_{3,2} + 500x_{4,2} \leq 4000 \\ 400x_{1,3} + 300x_{2,3} + 200x_{3,3} + 500x_{4,3} \leq 8000 \end{cases}$$

Availability constraints

Finally, there is a limited quantity of tonnes of each cargo type. In particular, there are 18 tonnes of cargo type 1, 10 tonnes of cargo type 2, 5 tonnes of cargo type 3 and 20 tonnes of cargo type 4. It is therefore necessary to introduce availability constraints requiring that the quantity of each type of goods, allocated to at least one of the wagons available, does not exceed the maximum stock availability.

$$\begin{cases} x_{1,1} + x_{1,2} + x_{1,3} \leq 18 \\ x_{2,1} + x_{2,2} + x_{2,3} \leq 10 \\ x_{3,1} + x_{3,2} + x_{3,3} \leq 5 \\ x_{4,1} + x_{4,2} + x_{4,3} \leq 20 \end{cases}$$

Non-negativity constraints

It is required that the decision variables must be at least equal to zero.

$$x_{i,j} \geq 0 \; i = 1, 2, 3, 4 \; j = 1, 2, 3$$

The model

Thanks to the package *lpSolveAPI* the following linear programming problem is solved:

$$\begin{aligned} \max \quad & 2000 \sum_{j=1}^{3} x_{1,j} + 2500 \sum_{j=1}^{3} x_{2,j} + 5000 \sum_{j=1}^{3} x_{3,j} + 3500 \sum_{j=1}^{3} x_{4,j} \\ \text{subject to} \quad & \begin{array}{c} x_{1,1} + x_{2,1} + x_{3,1} + x_{4,1} \leq 10 \\ x_{1,2} + x_{2,2} + x_{3,2} + x_{4,2} \leq 8 \\ x_{1,3} + x_{2,3} + x_{3,3} + x_{4,3} \leq 12 \\ 400x_{1,1} + 300x_{2,1} + 200x_{3,1} + 500x_{4,1} \leq 5000 \\ 400x_{1,2} + 300x_{2,2} + 200x_{3,2} + 500x_{4,2} \leq 4000 \\ 400x_{1,3} + 300x_{2,3} + 200x_{3,3} + 500x_{4,3} \leq 8000 \\ x_{1,1} + x_{1,2} + x_{1,3} \leq 18 \\ x_{2,1} + x_{2,2} + x_{2,3} \leq 10 \\ x_{3,1} + x_{3,2} + x_{3,3} \leq 5 \\ x_{4,1} + x_{4,2} + x_{4,3} \leq 20 \\ x_{i,j} \geq 0 \; i = 1, 2, 3, 4 \; j = 1, 2, 3 \end{array} \end{aligned} \tag{1}$$

The model is defined, with twelve decision variables and initializing it with zero constraints, since they will be added later on.

```
library(lpSolveAPI)
library(dplyr)
library(pander)
library(tidyr)
model <- make.lp(0, 12)
```

It is established that the problem is a maximum problem, and the objective function subject to maximization is defined.

```
lp.control(model, sense = "max")
set.objfn(model, obj = c(2000, 2000, 2000, 2500, 2500, 2500,
                         5000, 5000, 5000, 3500, 3500, 3500))
```

The next step is to add the constraints to the model using the command *add.constraint*.

The weight constraints are defined:

```
add.constraint(model,
               xt = c(1,1,1,1),
               type = "<=",
               rhs = 10,
               indices = c(1, 4, 7, 10))

add.constraint(model,
               xt = c(1, 1, 1, 1),
               type = "<=",
               rhs = 8,
               indices = c(2, 5, 8, 11))

add.constraint(model,
               xt = c(1,1,1,1),
               type = "<=",
               rhs = 12,
               indices = c(3, 6, 9, 12))
```

The volume constraints are added:

```
add.constraint(model,
               xt = c(400, 300, 200, 500),
               type = "<=",
               rhs = 5000,
               indices = c(1,4,7,10))

add.constraint(model,
               xt = c(400, 300, 200, 500),
               type = "<=",
               rhs = 4000,
               indices = c(2, 5, 8, 11))


add.constraint(model,
               xt = c(400, 300, 200, 500),
               type = "<=",
               rhs = 8000,
               indices = c(3, 6, 9, 12))
```

The availability constraints are added:

```
add.constraint(model,
               xt = c(1,1,1),
               type = "<=",
               rhs = 18,
               indices = c(1,2,3))

add.constraint(model,
               xt = c(1, 1, 1),
               type = "<=",
               rhs = 10,
               indices = c(4, 5, 6))

add.constraint(model,
               xt = c(1,1,1),
```

```
                type = "<=",
                rhs = 5,
                indices = c(7, 8, 9))

add.constraint(model,
                xt = c(1,1,1),
                type = "<=",
                rhs = 20,
                indices = c(10,11,12))
```

Finally, the lower limit of the decision variables is defined:

```
set.bounds(model, lower = c(rep(0, 12)))
```

Solution exploration

The linear programming problem is solved by the command *solve(model)*.

```
solve(model)
```

First of all, it was decided to show the value of the objetcive function at the optimal solution.

```
get.objective(model)
```

```
## [1] 107500
```

The maximum profit that can be achieved, respecting the constraints, is 107500$.

The optimal value of the decision variables is extracted.

```
get.variables(model)
```

```
##  [1]  0  0  0  5  0  0  5  0  0  0  8 12
```

The value of the decision variables, in correspondence to the optimal solution is shown in the following table.

variable	Value
$x_{2,1}$	5
$x_{3,1}$	5
$x_{4,2}$	8
$x_{4,3}$	12

All the other 8 decision variables that have been omitted from the table above, in correspondence to the optimal solution, are equal to zero. In order to maximize the profit, it is necessary to observe the following rule identified by the resolution of the problem:

- load 5 tonnes of cargo type 2 on the first wagon;
- load 5 tonnes of cargo type 3 on the first wagon;
- load 8 tonnes of cargo type 4 on the second wagon;
- load 12 tonnes of cargo type 4 on the third wagon.

However, it is not yet known which are all the basic variables and their value, so it is necessary to deepen the analysis of the solution by identifying the binding constraints, the value of the slack variables and finally identify which are the basic and not basic variables. Starting with the search of the binding constraints, using the command *get.constraints(model)*, the value of the LHS term is shown for each constraint.

```
get.constraints(model)
```

```
##  [1]   10    8   12 2500 4000 6000    0    5    5   20
```

Given the vector of RHS terms $b^t = [10, 8, 12, 5000, 4000, 8000, 18, 10, 5, 20]$ it is concluded that the first three constraints are binding, as weel the fifth, the ninth and the tenth. On the contrary, the fourth, sixth, seventh and eighth are not binding. From this information we can derive the values of the ten slack variables. In fact, the value of the slack variables is given by the difference between LHS terms and the RHS terms of each constraint. Consequently, the slack variables s_i, $i = 1, 2, 3, 5, 9, 10$ are equal to zero, whereas the slack variables s_4, s_6, s_7, s_8 are equal to 2500, 2000, 18 and 5 respectively. The value of the slack variables represents the amount of available resources that have not been fully exploited. So, if in terms of the maximum weight that each wagon can support there is no waste, the same cannot be said for the volume: for the first wagon, only half of the available volume is used, while for the third wagon, 75% of its total volume is utilised. With regard to the goods, it is noted that the cargo type 1 is completely unused, a sign of the existence of problems related to its profitability. For cargo type 2, only half of the available stocks are used. Finally, the stocks of type 3 and cargo type 4 are completely used.

The next step is to identify the basic variables, using the command *get.basis(model)*.

```
get.basis(model, nonbasic = F)
```

```
##  [1] -14 -21 -22  -4  -5  -6  -7 -15 -17  -8
```

The basic variables are: $x_{2,1}$, $x_{4,2}$, $x_{4,3}$, s_4, s_5, s_6, s_7, $x_{2,2}$, $x_{3,1}$, s_8 . This optimal solution is degenerate, because not all the basic variables are different from zero. In particular it is the decision variable $x_{2,2}$ and the sklack variable s_5 to be equal to zero, even though they are basic variables.

Sensitivity analysis

First of all, are obtained the optimality range, which represent the range of values within which the coefficients of the objective function can fluctuate, so that the optimal solution remains unchanged, assuming that all other coefficients remain unchanged.

```
printSensitivityObj <- function(model){
options(scipen=999)
arg.obj = get.sensitivity.obj(model)
numRows <- length(arg.obj$objfrom)
symb <- c()
j = 1
for (i in c(1:numRows)) {
  if (i == 1 | i == 2 | i == 3) {
    symb[i] <- paste("X", 1, sep = "")
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j +1
    if (i == 3) {j = 1}
  }
  if (i == 4 | i == 5 | i == 6) {
    symb[i] <- paste("X", 2, sep = "")
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j +1
    if(i == 6) {j = 1}
  }
  if (i == 7 | i == 8| i == 9) {
    symb[i] <- paste("X", 3, sep = "")
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j + 1
    if (i == 9) {j = 1}
  }
  if (i == 10 | i == 11 | i == 12) {
    symb[i] <- paste("X", 4, sep = "")
```

```
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j+1
  }

}
obj <- data.frame(Objs = symb, arg.obj)
obj<-
obj %>%
mutate(objfrom=replace(objfrom, objfrom < -1.0e4, "-inf")) %>%
mutate(objtill=replace(objtill, objtill > 1.0e4, "inf")) %>%
unite(col = "Sensitivity",
objfrom, Objs, objtill ,
sep = " <= ", remove = FALSE) %>%
select(c("Objs","Sensitivity"))
return(obj)
}
```

```
pander(printSensitivityObj(model))
```

Objs	Sensitivity
X1,1	-inf <= X1,1 <= 2500
X1,2	-inf <= X1,2 <= 2500
X1,3	-inf <= X1,3 <= 2500
X2,1	2500 <= X2,1 <= 2500
X2,2	-inf <= X2,2 <= 2500
X2,3	-inf <= X2,3 <= 2500
X3,1	5000 <= X3,1 <= inf
X3,2	-inf <= X3,2 <= 5000
X3,3	-inf <= X3,3 <= 5000
X4,1	-inf <= X4,1 <= 3500
X4,2	3500 <= X4,2 <= 3500
X4,3	3500 <= X4,3 <= inf

The table above shows for each coefficient of the objective function, the range within which they can fluctuate, without changing the optimal solution. In particular, it can be seen that as long as the profit of the first type of product does not increase to at least 2500$, this type of cargo continues to be unused. This is a confirmation of the existence of problems related to the profitability of this cargo type, that need to be solved.

Subsequently, the shadow prices of the constraints are derived. They indicate the change in the value of the objective function, against a unitary change of the RHS term of the constraint.

```
printSensitivityRHS <- function(model){
options(scipen=999)
arg.rhs =get.sensitivity.rhs(model)
numRows <- length(arg.rhs$duals)
symb <- c()
for (i in c(1:numRows)) symb[i] <- paste("b", i, sep = "" )
rhs <- data.frame(rhs = symb,arg.rhs)
rhs<-rhs %>%
mutate(dualsfrom=replace(dualsfrom, dualsfrom < -1.0e4, "-inf")) %>%
mutate(dualstill=replace(dualstill, dualstill > 1.0e4, "inf")) %>%
unite(col = "Sensitivity",
dualsfrom,
```

```
rhs,
dualstill ,
sep = " <= ", remove = FALSE) %>%
select(c("rhs","Sensitivity"))
colnames(rhs)[1]<-c('Rhs')
return(rhs[c(1:10),c(1,2)])
}


printShadowPrice <- function(model) {
options(scipen=999)
shadow.price <- get.sensitivity.rhs(model)$duals
arg.rhs <- get.dual.solution(model)
symb <- c()
for (i in c(1:10)) {
  symb[i] <- paste("b", i, sep = "")
}
ShadowPrice <- data.frame(Rhs = symb, Rhs_Value = get.constr.value(model, side = "rhs"),
                          shadow_price = shadow.price[0:10])
return(ShadowPrice)
}
```

```
d <-merge(printShadowPrice(model),
              printSensitivityRHS(model),
              by = "Rhs")
d$Rhs <- factor(d$Rhs, levels(d$Rhs)[c(1,3,4,5,6,7,8,9,10,2)])
d <- d[order(d[,1]),]
row.names(d) <- NULL
pander(d)
```

Rhs	Rhs_Value	shadow_price	Sensitivity
b1	10	2500	5 <= b1 <= 15
b2	8	2500	8 <= b2 <= 8
b3	12	2500	12 <= b3 <= 16
b4	5000	0	-inf <= b4 <= inf
b5	4000	0	-inf <= b5 <= inf
b6	8000	0	-inf <= b6 <= inf
b7	18	0	-inf <= b7 <= inf
b8	10	0	-inf <= b8 <= inf
b9	5	2500	0 <= b9 <= 10
b10	20	1000	15 <= b10 <= 20

The column *shadow_price* represents the shadow prices of the RHS terms, while the column *sensitivity* represents the range of values within which the RHS terms of the constraints can change, so that the corresponding shadow price is valid. The following conclusions can be deduced from reading the table above.

- the shadow prices for the fourth constraint (first wagon volume constraint), the sixth constraint (third wagon volume constraint), the seventh constraint (cargo type 1 availability constraint) and the eighth constraint (cargo type 2 availability constraint) are equal to zero. This result was to be expected as the shadow price of a non binding constraint is always equal to zero. The shadow price of the fifth constraint, relating to the volume of the second wagon, is also equal to zero. It is possible to conclude that increasing the volume of the available wagons would not lead to any increase in the objective function, just as increasing the stocks of cargo type 1 and cargo type 2 would not lead to an increase in

profit. This would occur in the event of any increase or decrease in the availability of these resources;
- an interesting consideration concerns the maximum weight that can be supported by the second wagon. The current availability of this resource is 8 tonnes. Therefore, although the shadow price of such resource is 2500$ is unusable to find the new value of the objective function after variations of such resource. This is happens, because the range shows how even a very small variation would cause the value of this resource to exceed the range where the shadow price is valid;
- with regard to RHS term of the first constraint, which refers to the maximum weight that can be carried by the first wagon, it is possible to see that an increase or decrease of a maximum of 5 tonnes leads to a variation (positive or negative) of the profit of 2500$, for each tonne in more or in less;
- with regard to the third constraint, it is possible to note that any reduction in the maximum tonnes that can be carried by the third wagon would not make the shadow price usable to find the new value of the objective function. On the contrary, an increase of maximum transportable weight up to 4 tonnes would lead to a profit increase of 2500$ for each additional tonne;
- with regard to the RHS term of the ninth constraint, it is possible to see that an increase or decrease of up to 5 tonnes in the availability of cargo type 3 would result in an increase (or decrease) in profit of 2500$, for each tonne in more or in less;
- regarding the last constraint, it's possible to see that an increase in the tonnes available for cargo type 4 would not allow the shadow price to be used to find the new profit value. On the contrary, a decrease up to 5 tons would result in a decrease in profit of 1000$ for each ton in less.

However, it is necessary to specify that this procedure makes it possible to find the new value of the objective function, but does not guarantee that this value will still be the optimal value after the changes recorded. Just like it does not allow conclusions in the event that a resource undergoes a change in its availability, greater than the maximum allowed.

Questions about LP

1. Can an LP model have more than one optimal solution. Is it possible for an LP model to have exactly two optimal solutions? Why or why not?

A linear programming problem may have more than one optimal solution, but not exactly two. In fact, when a linear programming problem has more than one solution it has an infinite number of optimal solutions. This is guaranteed by the fundamental theorem of linear programming. This theorem states that only the following cases can occur: the problem is boundless, so the feasible region is unbounded, the problem is unfeasible, so the feasible region is empty or the problem has an optimal solution, and this solution is unique, or there are infinite optimal solution.

2. Are the following objective functions for an LP model equivalent? That is, if they are both used, one at a time, to solve a problem with exactly the same constraints, will the optimal values for x_1 and x_2 be the same in both cases? Why or why not?

$$\max 2x_1 + 3x_2 \min -2x_1 - 3x_2$$

In general, in mathematical programming problems, given an objective function $f(x)$, it is possible to turn a maximum problem into a minimum problem by changing the sign of the objective function. In fact, the maximum points of the problem (if they exist)

$$\underset{x \epsilon S}{max}(f(x))$$

coincide with the minimum points of the problem

$$\underset{x \epsilon S}{min}(f(-x))$$

where S represents the feasible region, and x the vector of the decision variables.

Given this premise, the two linear programming problems $max 2x_1 + 3x_2$ and $min -2x_1 - 3x_2$, characterized by the same constraints and therefore the same feasible region, have as optimal solution the same values of x_1 e x_2. However, the two objective functions evaluated in the same points have different values.

3. Which of the following constraints are not linear or cannot be included as a constraint in a linear programming problem?

a. $2x_1 + x_2 - 3x_3 \geq 50$

Yes, the constraint a is linear and can be included in a linear programming problem.

b. $2x_1 + \sqrt{x_2} \geq 60$

The constraint b is not linear and therefore cannot be included in a linear programming problem.

c. $4x_1 - \frac{1}{2}x_2 = 75$

Yes, the constraint c is linear and can be included in linear in a programming problem.

d. $\frac{3x_1+2x_2x-3x_3}{x_1+x_2+x_3} \leq 0.9$

The constraint d is not linear and therefore cannot be included in a linear programming problem.

e. $3x_1^2 + 7x_2 \leq 45$

The constraint e is not linear and therefore cannot be included in a linear programming problem.

Appendix

```
library(lpSolveAPI)
library(dplyr)
library(pander)
library(tidyr)
model <- make.lp(0, 12)

#Weight constraints
add.constraint(model,
               xt = c(1,1,1,1),
               type = "<=",
               rhs = 10,
               indices = c(1, 4, 7, 10))

add.constraint(model,
               xt = c(1, 1, 1, 1),
               type = "<=",
               rhs = 8,
               indices = c(2, 5, 8, 11))

add.constraint(model,
               xt = c(1,1,1,1),
               type = "<=",
               rhs = 12,
               indices = c(3, 6, 9, 12))

#Volume constraints
add.constraint(model,
               xt = c(400, 300, 200, 500),
               type = "<=",
```

```
                rhs = 5000,
                indices = c(1,4,7,10))

add.constraint(model,
                xt = c(400, 300, 200, 500),
                type = "<=",
                rhs = 4000,
                indices = c(2, 5, 8, 11))

#Availability Constraints

add.constraint(model,
                xt = c(400, 300, 200, 500),
                type = "<=",
                rhs = 8000,
                indices = c(3, 6, 9, 12))

add.constraint(model,
                xt = c(1,1,1),
                type = "<=",
                rhs = 18,
                indices = c(1,2,3))

add.constraint(model,
                xt = c(1, 1, 1),
                type = "<=",
                rhs = 10,
                indices = c(4, 5, 6))

add.constraint(model,
                xt = c(1,1,1),
                type = "<=",
                rhs = 5,
                indices = c(7, 8, 9))

add.constraint(model,
                xt = c(1,1,1),
                type = "<=",
                rhs = 20,
                indices = c(10,11,12))

#Lower Bounds

set.bounds(model, lower = c(rep(0, 12)))

solve(model)
get.objective(model)
get.variables(model)
get.constraints(model)
get.basis(model, nonbasic = F)

printSensitivityObj <- function(model){
options(scipen=999)
```

```
arg.obj = get.sensitivity.obj(model)
numRows <- length(arg.obj$objfrom)
symb <- c()
j = 1
for (i in c(1:numRows)) {
  if (i == 1 | i == 2 | i == 3) {
    symb[i] <- paste("X", 1, sep = "")
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j +1
    if (i == 3) {j = 1}
  }
  if (i == 4 | i == 5 | i == 6) {
    symb[i] <- paste("X", 2, sep = "")
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j +1
    if(i == 6) {j = 1}
  }
  if (i == 7 | i == 8| i == 9) {
    symb[i] <- paste("X", 3, sep = "")
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j + 1
    if (i == 9) {j = 1}
  }
  if (i == 10 | i == 11 | i == 12) {
    symb[i] <- paste("X", 4, sep = "")
    symb[i] <- paste(symb[i], j, sep = ",")
    j = j+1
  }

}
obj <- data.frame(Objs = symb, arg.obj)
obj<-
obj %>%
mutate(objfrom=replace(objfrom, objfrom < -1.0e4, "-inf")) %>%
mutate(objtill=replace(objtill, objtill > 1.0e4, "inf")) %>%
unite(col = "Sensitivity",
objfrom, Objs, objtill ,
sep = " <= ", remove = FALSE) %>%
select(c("Objs","Sensitivity"))
return(obj)
}

pander(printSensitivityObj(model))

printSensitivityRHS <- function(model){
options(scipen=999)
arg.rhs =get.sensitivity.rhs(model)
numRows <- length(arg.rhs$duals)
symb <- c()
for (i in c(1:numRows)) symb[i] <- paste("b", i, sep = "" )
rhs <- data.frame(rhs = symb,arg.rhs)
rhs<-rhs %>%
mutate(dualsfrom=replace(dualsfrom, dualsfrom < -1.0e4, "-inf")) %>%
```

```
mutate(dualstill=replace(dualstill, dualstill > 1.0e4, "inf")) %>%
unite(col = "Sensitivity",
dualsfrom,
rhs,
dualstill ,
sep = " <= ", remove = FALSE) %>%
select(c("rhs","Sensitivity"))
colnames(rhs)[1]<-c('Rhs')
return(rhs[c(1:10),c(1,2)])
}


printShadowPrice <- function(model) {
options(scipen=999)
shadow.price <- get.sensitivity.rhs(model)$duals
arg.rhs <- get.dual.solution(model)
symb <- c()
for (i in c(1:10)) {
  symb[i] <- paste("b", i, sep = "")
}
ShadowPrice <- data.frame(Rhs = symb, Rhs_Value = get.constr.value(model, side = "rhs"),
                          shadow_price = shadow.price[0:10])
return(ShadowPrice)
}

d <-merge(printShadowPrice(model),
              printSensitivityRHS(model),
              by = "Rhs")
d$Rhs <- factor(d$Rhs, levels(d$Rhs)[c(1,3,4,5,6,7,8,9,10,2)])
d <- d[order(d[,1]),]
row.names(d) <- NULL
pander(d)
```