**‘Rglpk’ Package**

The ‘Rglpk’ package is a “high level R interface to the CPLEX\\_LP, MATHPROG and MPS reader of the GNU Linear Programming Kit (GLPK)”. GLPK version 4.52.1 is an open source software for solving large-scale linear programs as well as mixed integer linear programs. This package was written by Stefan Theussl and others while being maintained by Theussl. The ‘slam’ package is automatically installed through the installation of ‘Rglpk’ as ‘Rglpk’ depends on it (most linear programming packages in R depend on the ‘slam’ package). [JRW]

To install ‘Rglpk’, you must open up the R software, select the “Packages” tab and then select “Install Package(s)…”. From here you select the area of the world that you live closest to (the installation tends to run faster the closer the selection). Next you scroll down and double click the package titled “Rglpk”. Your R Console will then show in the output that “package ‘Rglpk’ successfully unpacked…”. It will also say the same for the ‘slam’ package. Now that the package has been installed into your R software, you must then activate its usage by simply typing in the command “> library(Rglpk)”. The output from doing this will show the following under a warning message:

1: package ‘Rglpk’ was built under R version 3.0.2

2: package ‘slam’ was built under R version 3.0.2

This is an indication that the package was loaded successfully and that you may proceed to use it.

The usage of this package involves two functions: “Rglpk\_read\_file” and “Rglpk\_solve\_LP”. The first function has the ability to read in .mps files into R and recognize what the linear program is. The input code for this function looks like the following:

>Rglpk\_read\_file(file, type = c("MPS\_fixed", "MPS\_free", "CPLEX\_LP", "MathProg"),ignore\_first\_row = FALSE, verbose = FALSE)

Here the “file” inside of the parenthesis indicates the path of the model file, the “type” indicates what type of file (the options are above), the “ignore\_first\_row=FALSE” indicates whether or not it is needed to ignore the first row of the model and finally “verbose=FALSE” indicates whether or not to turn on additional solver output.

So once this function is used, a description of the linear program model is shown in the output where the number of objective variables and constraints are given. As an example, the following code shows the reading in of a linear program model that is stored in the package as an .mps file with 7 objective variables and 8 constraints:

x <- Rglpk\_read\_file( system.file(file.path("examples", "plan.lp"), package = "Rglpk"), type = "CPLEX\_LP")

> x

A linear program with 7 objective variables.

This problem has 8 constraints with 48 non-zero values in the constraint matrix.

Now that this LP is read in and saved as an object (“x”), we can use the “Rglpk\_solve\_LP” function to solve it. This is shown by the following code:

> Rglpk\_solve\_LP(x$objective, x$constraints[[1]], x$constraints[[2]],

+ x$constraints[[3]], x$bounds, x$types, x$maximum)

$optimum

[1] 296.2166

$solution

[1] 0.0000 665.3430 490.2527 424.1877 0.0000 299.6390 120.5776

$status

[1] 0

The above output code (in blue) shows the solution to the read-in LP model. The benefit of this is that you do not need any other software other than R in order to solve a .mps file.

Now building off of the solve function, we can input all of our arguments to create our LP without reading in a .mps file. Using nothing more than the package’s function Rglpk\_solve\_LP, we can solve any LP of our choosing. To define our LP, we must add in the following inputs for our function : Rglpk\_solve\_LP(obj, mat, dir, rhs, bounds = NULL, types = NULL, max = FALSE)

* ‘obj’ – vector of objective function coefficients
* ‘mat’ – matrix of the constraint coefficients
* ‘dir’ – a vector defining the direction of each constraint
* ‘rhs’ – a vector of the right-hand-side values for each constraint
* ‘bounds’ – defines bounds of decision variables (NULL is non-negativity)
* ‘types’ – defines the measure of the output variables (NULL is continuous, integer and binary are other options)
* ‘max’ – defines the optimization, set = TRUE is a max, FALSE is a min

With a LP model having many objective variables and constraints, inputting all of this information can get very messy. To make this process much easier, I developed code that gives step by step instructions on how to input everything in a notepad. This way copying and pasting the entire code will give you your solution once the input is added. The full code is shown in the appendices with the inputs for the following example.

As an example, the following problem (chosen from Dr. Chrispell’s Homework 3) is solved using my original code and the “Rglpk\_solve\_LP” function.

A cargo plane has three compartments that can be used for storing cargo; front, center, and back. These compartments have the capacity limits on total weight and space available given by the table below:

|  |  |  |
| --- | --- | --- |
| Compartment | Weight (tons) | Space (cu-ft) |
| Front | 12 | 7,000 |
| Center | 18 | 9,000 |
| Back | 10 | 5,000 |

Also the weight of the cargo actually placed in the compartments must be in the same proportion as the compartment’s weight capacity, in order to maintain the balance of the plane. The following four cargos have been offered for shipment on an upcoming flight as space is available. Any portion of these cargos can be accepted. The objective is to maximize the total profit for the flight.

|  |  |  |  |
| --- | --- | --- | --- |
| Cargo | Weight (tons) | Volume (cu-ft) | Profit ($/ton) |
| 1 | 20 | 8,000 | 280 |
| 2 | 16 | 8,000 | 360 |
| 3 | 25 | 14,000 | 320 |
| 4 | 13 | 13,000 | 310 |

The LP model can be set up as the following (the full detailed model is shown in the appendices):

Decision Variables: the amount in tons of the cargo to be placed in the section of the plane where and . Here represents the front of the plane, represents the center of the plane and represents the back of the plane.

s.t.

With

Here we can see that the above LP model has 12 objective variables and a total of 13 constraints. Entering the values directly into R under the defined objects can be quite messy. The use of the notepad for entering in the input makes for an easy finding to the solution of the LP.

Instead of entering in the constraint matrix in one object, my code allows for each constraint coefficient to be entered one constraint at a time and then combined into matrix form. The direction vector is made up of objects that are defined early in the code defining the direction. The following input code is the condensed version of the notepad where only the uncommented code is shown (the commented code has a “#” symbol in front of it in the notepad):

|  |  |
| --- | --- |
| > library(Rglpk)  > LE = "<="  > GE = ">="  > E = "=="  > L = "<"  > G = ">"  > obj = c(280, 360, 320, 310, 280, 360, 320, 310, 280, 360, 320, 310)  > con1 = c(1,1,1,1,0,0,0,0,0,0,0,0 )  > con2 = c(0,0,0,0,1,1,1,1,0,0,0,0 )  > con3 = c(0,0,0,0,0,0,0,0,1,1,1,1 )  > con4 = c(28,28,28,28,-12,-12,-12,-12,-12,-12,-12,-12 )  > con5 = c(-18,-18,-18,-18,22,22,22,22,-18,-18,-18,-18 )  > con6 = c(-10,-10,-10,-10,-10,-10,-10,-10,30,30,30,30 )  > con7 = c(1,0,0,0,1,0,0,0,1,0,0,0 )  > con8 = c(0,1,0,0,0,1,0,0,0,1,0,0 )  > con9 = c(0,0,1,0,0,0,1,0,0,0,1,0 )  > con10 = c(0,0,0,1,0,0,0,1,0,0,0,1 )  > con11 = c(400,500,560,1000,0,0,0,0,0,0,0,0 )  > con12 = c(0,0,0,0,400,500,560,1000,0,0,0,0 )  > con13 = c(0,0,0,0,0,0,0,0,400,500,560,1000 ) | > mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9, con10, con11, con12, con13),nrow=13, byrow=TRUE) # thirteen constraints  > dir = c(LE,LE,LE,E,E,E,LE,LE,LE,LE,LE,LE,LE )  > rhs = c(12,18,10,0,0,0,20,16,25,13,7000,9000,5000 )  > max = max = TRUE  > Rglpk\_solve\_LP(obj, mat, dir, rhs, bounds=NULL, types= NULL, max)  $optimum  [1] 13260  $solution  [1] 0.000000e+00 0.000000e+00 1.200000e+01 0.000000e+00 4.500000e+00  [6] 6.000000e+00 7.500000e+00 0.000000e+00 7.253273e-15 1.000000e+01  [11] 0.000000e+00 0.000000e+00  $status  [1] 0 |

The results of the output are shown to have found the optimum solution to be $13,260.00 when we have 4.5 tons of cargo 1 going to the center of the plane, 6 tons of cargo 2 to the center of the plane, 10 tons of cargo 2 to the back of the plane, 12 tons of cargo 3 to the front of the plane, 7.5 tons of cargo 3 to the center of the plane and 0 tons of cargo 4 are added. This same solution can be found using other software.

As can be seen from the output, the values of the objective variables are shown under the “$solution” as a vector of values. The “$status” shows the status of the findings – the zero represents that we have found an optimum. If any other value is shown, it represents a GLPK status code. The reason for each error in the solution (such as invalid bounds) is represented by a different code, each of which can be found at the following link:

<http://www.gnu.org/software/octave/doc/interpreter/Linear-Programming.html>

So overall, the “Rglpk\_solve\_LP” function can be executed to a .mps file that has been read in by the “Rglpk\_read\_file” function or to LP that has been inputted directly in R.

**References**

[JRW] Theussl, Stefan. "Package ‘Rglpk’." *cran.at.r-project.org*. N.p., 27 Nov 2013. Web. 11

Mar 2014. <http://cran.at.r-project.org/web/packages/Rglpk/Rglpk.pdf>.

**Appendix**

**‘Rglpk’ Input Code**

# Initial code

library(Rglpk)

LE = "<="

GE = ">="

E = "=="

L = "<"

G = ">"

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Type in the coefficients of the objective function variables

# seperated by commas inside the paranthesis ( for all variables)

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

obj = c(280, 360, 320, 310, 280, 360, 320, 310, 280, 360, 320, 310)

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Remove the hashtag from the front of each constraint you

# are using. Type in the coefficients for each variable in the

# LP for each constraint separated by commas in the parenthesis

#\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

con1 = c(1,1,1,1,0,0,0,0,0,0,0,0 )

con2 = c(0,0,0,0,1,1,1,1,0,0,0,0 )

con3 = c(0,0,0,0,0,0,0,0,1,1,1,1 )

con4 = c(28,28,28,28,-12,-12,-12,-12,-12,-12,-12,-12 )

con5 = c(-18,-18,-18,-18,22,22,22,22,-18,-18,-18,-18 )

con6 = c(-10,-10,-10,-10,-10,-10,-10,-10,30,30,30,30 )

con7 = c(1,0,0,0,1,0,0,0,1,0,0,0 )

con8 = c(0,1,0,0,0,1,0,0,0,1,0,0 )

con9 = c(0,0,1,0,0,0,1,0,0,0,1,0 )

con10 = c(0,0,0,1,0,0,0,1,0,0,0,1 )

con11 = c(400,500,560,1000,0,0,0,0,0,0,0,0 )

con12 = c(0,0,0,0,400,500,560,1000,0,0,0,0 )

con13 = c(0,0,0,0,0,0,0,0,400,500,560,1000 )

#con14 = c( )

#con15 = c( )

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Remove the hashtag from the front depending on the

# number of constraints

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#mat = matrix(c(con1),nrow=1, byrow=TRUE) # one constraint

#mat = matrix(c(con1, con2),nrow=2, byrow=TRUE) # two constraints

#mat = matrix(c(con1, con2, con3),nrow=3, byrow=TRUE) # three constraints

#mat = matrix(c(con1, con2, con3, con4),nrow=4, byrow=TRUE) # four constraints

#mat = matrix(c(con1, con2, con3, con4, con5),nrow=5, byrow=TRUE) # five constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6),nrow=6, byrow=TRUE) # six constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7),nrow=7, byrow=TRUE) # seven constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8),nrow=8, byrow=TRUE) # eight constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9),nrow=9, byrow=TRUE) # nine constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9, con10),nrow=10, byrow=TRUE) # ten constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9, con10, con11),nrow=11, byrow=TRUE) # eleven constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9, con10, con11, con12),nrow=12, byrow=TRUE) # twelve constraints

mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9, con10, con11, con12, con13),nrow=13, byrow=TRUE) # thirteen constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9, con10, con11, con12, con13, con14),nrow=14, byrow=TRUE) # fourteen constraints

#mat = matrix(c(con1, con2, con3, con4, con5, con6, con7, con8, con9, con10, con11, con12, con13, con14, con15),nrow=15, byrow=TRUE) # fifteen constraints

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Type in the directions for each constraint by using LE for less

# than or equal to, GE for greater than or equal to, E for equal

# to, L for strictly less than and G for strictly greater than

# Put these in order of their constraint inside the parentheses

# separated by commas

dir = c(LE,LE,LE,E,E,E,LE,LE,LE,LE,LE,LE,LE )

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Type in the Right Hand Side values of the constraints inside of

# the parentheses separated by commas

rhs = c(12,18,10,0,0,0,20,16,25,13,7000,9000,5000 )

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# Remove that hashtag for the type of optimization. FALSE represents

# a min problem and TRUE represents a max problem

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#max = max = FALSE

max = max = TRUE

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# The following will execute the linear program that you created.

# The bounds is set to NULL indicating the variables are bounded

# by zero and infinity. The types is set to NULL indicating that

# our solution can be found using continuous parameters. If we want

# only integer solutions we type in "I" in the place of NULL for

# types.

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Rglpk\_solve\_LP(obj, mat, dir, rhs, bounds=NULL, types= NULL, max)

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Copy and Paste All of the Above\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*