R: Color Package

Eurípides Rivera Negrón
Computer Engineering
Jeffrey L. Duffany, Ph.D.
Computer Engineering Department
Polytechnic University of Puerto Rico

Abstract *— R is a software language mostly used of statistical and mathematical purporses. R capabilities are extended by user's submitted packages. The gColor Package provides R Language a method to solve Systems of Inequation (represented as a matrix) in files created by the Discrete Mathematics and Theoretical Computer Science (SIMACS). The advantange of the gColor Package is the import fo DIMACS files which supports compress (binary) and uncompress (ASCII) format, currently not available in R Language,and the conversion of the data imported from vertices and edges into a an adjacency matrix; allowing users to import graphs from other systems and use it as a matrix object. This document is divided in four sections. The Introduction provides and overviedw of the problem and the justification to the development of this project. The R Language brings an introduction to R. The system of Inequation explains briefly what is it and how can be representd in R. The gColor Package explains the structure of the package. How the package was build and integrated with R is explained in section Building the Package.*

Key Terms *— ASCII, CRAN, DIMACS, Dynamic Link Library (DLL).*

INTRODUCTION

R is a language used for mathematics purposes, mostly in the statistics environment. The purpose of the Color Package is to provide the necessary mechanisms to solve Systems of Inequations [1]. Although the package was created to solve binary symmetric square matrix, it did not provide a method to import data from other systems (like DIMACS).

The first version of the Color Package was composed by the methods ineq() and test(). The ineq() function contains the code to solve System of Inequation using an argument of *n* x *n* binary matrix. The test() method was developed to create an *n* x *n* binary symmetric square matrix to test the ineq() functions. This first version was good for test purposes, but did not have all the requirements to be used and solve graphs created by other systems.

The new update to the Color Package provides two new methods that brings the users to import DIMACS standard format files (binary and ASCII). This two new functions, that allow the user to import DIMACS standard format ASCII files and DIMACS standard format binary files [7] respectively without any effort are the methods: importDIMACSAscii() and importDIMCASCBin(). The value returned will be a *n* x *n* binary matrix which can be solved using the ineq() function.

R LANGUAGE

R is a computer programming language is mostly used for statistical computing and graphics. It is an implementation of the S programming language with lexical scoping semantics inspired by Sheme. Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand invented it. Now R Development Core Team are the developers [2] and maintainers of the suite. R source code is available for everyone under the terms of GNU General Public License.

The suite of software integrated in R brings data manipulation, calculation and graphical display in one environment. Also, R is platform independent. The same code can be executed in a Windows or a Linux environment without problem. Among other things R language has: [3]

- An effective data handling and storage facility.
- A suite of operators for calculations on arrays, in particular matrices.
- A large, coherent, integrated collection of intermediate tools for data analysis.
- Graphical facilities for data analysis and display either directly at the computer or on hardcopy.
- A well developed, simple and effective programming language (called 'S') which includes conditionals, loops, user defined recursive functions and input and output facilities.

R is mostly related with statistics and other practitioners requiring an environment for statistical computation and software development, but it is preferred to think of it as an environment within which many classical and modern statistical techniques have been implemented. Most of these techniques have been developed by R users and have been supplied as packages.

Packages

The capabilities of R is extended through user-submitted packages, which allow specialized statistical techniques, graphical devices, as integration with other systems or language like C++, Java, MySQL, and Oracle [4]. A core set of packages are included in R and many others can be downloaded or installed directly from the Comprehensive R Archive Network (CRAN). Other unpublished beta packages can be obtained from R-Forge, which offers a central platform for the development of R packages and other related projects [2].

Examples

R language is easy to learn because is like you are speaking to the machine. Also it provides some powerful functions to solve math problems. Figure 1 [2] shows the basic syntax of the language and sage of the command-line interface. Figure 2 [2] show an example of graphics created in R.

```
> x <- c(1,2,3,4,5,6)   # Create ordered collection
> y <- x^2              # Square the elements of x
> mean(y)               # Calculate arithmetic mean of y
[1] 15.16667
> var(y)                # Calculate sample variance
[1] 178.9667
> summary(lm(y ~ x))    # Fit a linear regression model

Call:
lm(formula = y ~ x)

Residuals:
1       2       3       4       5       6
3.3333 -0.6667 -2.6667 -2.6667 -0.6667  3.3333

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept)  -9.3333     2.8441  -3.282 0.030453 *
x             7.0000     0.7303   9.585 0.000662 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.055 on 4 degrees of freedom
Multiple R-squared: 0.9583,     Adjusted R-squared: 0.9478
F-statistic: 91.88 on 1 and 4 DF,  p-value: 0.000662

> par(mfrow=c(2, 2))    # Request 2x2 plot layout
> plot(lm(y ~ x))       # Diagnostic plot of regression model
```

Figure 1
Basic syntax and usage of R

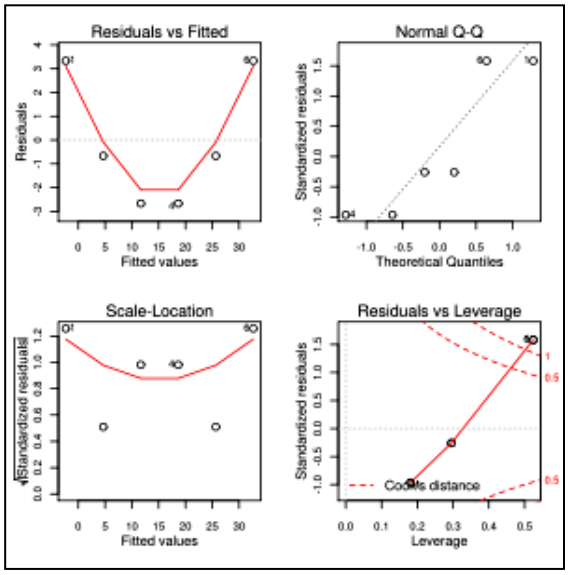

Figure 2
Graphs produced by plot.lm() function in R

SYSTEM OF INEQUATIONS

An inequation is a statement that two objects or expressions are not the same, or do not represent the same value [5]. A representation of an inequation is presented in equation (1).

$$x_i \neq x_j \qquad (1)$$

A system of inequation can be represented by a binary symmetric square matrix, with a zero representing a compatibility and a one an incompatibility [1]. The matrix A in equation (2) [1] is a representation of the system of inequations

$x_i \neq x_j$. The necessary code to create the equation (2) in R is presented in Figure 3.

$$A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \quad (2)$$

```
>
> A<-matrix(0,2,2)      #Create the symmetric matrix
> A[1,2]<-1             #Assign 1 to column 1, row 2
> A[2,1]<-1             #Assign 1 to column 2, row 1
> A                     #The result
     [,1] [,2]
[1,]    0    1
[2,]    1    0
>
```

Figure 3
$X_i \neq X_j$ inequation in R

For every system of n inequations there is at least one optimal solution of minimum cardinality k* and exactly one trivial solution s(n)={1,2,3…n} [1]. The optimal solution and trivial solution of equation (2) are the same, s*=s(n)={1,2}.

Algorithm to solve System of Inequation

Figure 4 [1] presents an algorithm to solve systems of inequations using a decision function f(A) = max (A2). The solution of the algorithm will be represented as a vector s. The algorithm inititialize the solution vector to the trivial solution s(n)={1,2,3,…n}. Then it steps through a series fo feasible solutions. Each time the dimension of the matrix is reduced by one the number of equivalence classes in the solution vector s is also reduced by one. The solution vector is updated by talking all variables that currently have solution value j and assigning a new solution value i. For further information of how the algorithm works, please refer to the document "Systems of Inequations" by Duffany, J. L. The representation of the algorithm in R Language is provided in the gColor Package inside ineq() function (Figure 5).

ineq(A)
s={1,2,3,...n}
ij ⇔ max(A^2)
if ij ={ϕ} return s
xi=xi ∪ xj
A=A-xj
s[s=j]=i
s[s>j]=s[s>j]-1
ineq(A)

Figure 4
Algorithm with f(A) = max(A^2) and solution vector s

```
ineq<-function(a)
{
a1<-a
ra<-nrow(a) #size of matrix a (n)
s<-1:ra # initial solution vector (1:n)
while(sum(a==0)>ra) #while some off-diagonal zeros remain
{                   #begin main loop
a2<-a%*%a #square the a matrix: a2=a**2
x<-row(a) #find row i values for each element in matrix a
y<-col(a) #find column j values for each element in matrix a
c<-x<y    #select upper triangular portion of a
w<-a[c]   #select all values in upper triangular portion of a
x<-x[c]   #find corresponding i values
y<-y[c]   #find corresponding j values
z<-a2[c]  #find corresponding a2 values
d<-w==0   #only consider cases where a[i,j]=0
x<-x[d]   #find corresponding i values
y<-y[d]   #find corresponding j values
z<-z[d]   #find corresponding a2 values
e<-rev(order(z)) #need to reverse increasing order to get the maximum
x<-x[e]   #i values in decreasing order of a2
y<-y[e]   #j values in decreasing order of a2
i<-x[1] #extract i value corresponding to max(a2)
j<-y[1] #extract j value corresponding to max(a2)
if(i>j) #ensure that i<j for correct update of s
{
tmp<-i  #if i>j reverse i and j
i<-j
j<-tmp
}
u<-a[i,] #select row i of a
v<-a[j,] #select row j of a
uv<-u|v  #bitwise logical or of row i and row j
a[i,]<-uv #replace row i of a
a[,i]<-uv #replace column i of a
a<-a[-j,-j] #remove row j and column j from a
s[s==j]<-i  #begin update solution vector s
s<-s-(1)*(s>j) #complete update of solution vector s
ra<-ra-1 #dimension of matrix a reduced by 1
}                   #end of main loop
a<-a1 #restore a to original input matrix
z<-order(s) #find ordering of solution vector
bdfa<-a[z,z] #put matrix a into block diagonal form
z<-s[z] #z is solution vector of block diagonal form of a
#print(bdfa) #print matrix a in block diagonal form (disabled)
return(s) #return solution vector s for input matrix a
}
```

Figure 5
Color Package ineq() function

COLOR PACKAGE

The gColor Package is a suite of functions that brings to R users the benefit to import binary symmetric matrix from DIMACS standard format files [7], process the data imported, and get the system of inequations solution. The suite is composed by four public functions, four private functions, and a library coded in C. The four public functions are the methods which the users can call, used and integrate with their code. The four private functions are used internally by the package in the methods importDIMACSAscii() and importDIMACSBin(). The library is also used internally by the import methods and it purpose is to convert the DIMACS binary files into ASCII file before the data is imported. In the following section will be described each method.

A diagram of the gColor Package is presented in Figure 6. The private functions they are accessed only from the import functions; as these private functions are important for the package, they have been created hidden, making them secure and difficult to reach and modify. In the Color Library there are also five important methods that are set private; the reason is that they are in charge to make the conversion.

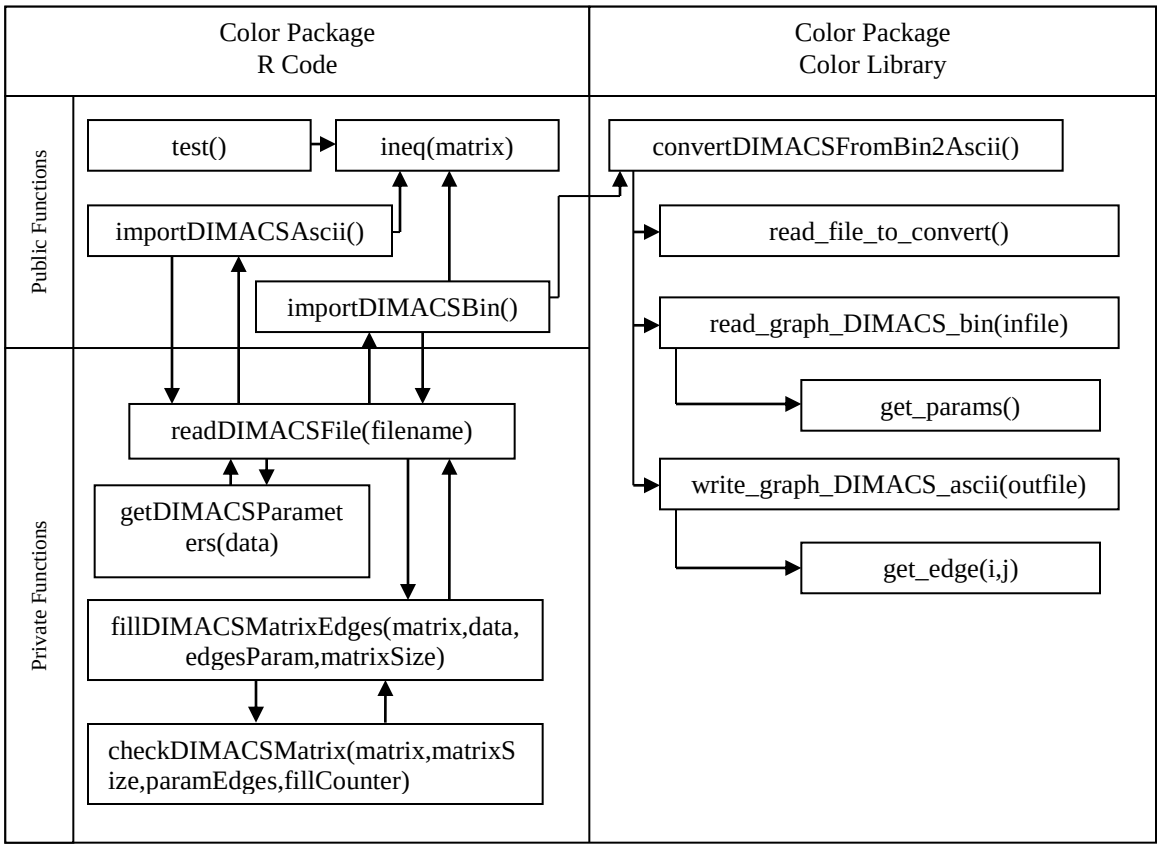

Figure 6
Color Package Diagram

Public Functions

The public functions inside the package are: ineq(), test(), importDIMACSAscii(), and importDIMACSBin(). As mention, these are the functions that the users will be able to use in order to solve Systems of Inequations. The purpose and usage of each method is described below:

- **test():** The purpose of this function is to quickly create a system of inequation to test the ineq() function. The method expects two arguments, the size of the matrix to be created and the optimal solution cardinality of the coloring of the graph represented by the matrix. The return of the function is a matrix. Figure 7 five an example of how to create a binary matrix 10 x 10 with the optimal solution cardinality of 3 using this function.

```
> test(10,3)
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    1    0    1    0    0    1     0
 [2,]    0    0    0    0    0    0    0    0    0     0
 [3,]    0    0    0    0    0    0    0    0    0     0
 [4,]    1    0    0    0    0    0    0    0    1     1
 [5,]    0    0    0    0    0    0    0    0    0     0
 [6,]    1    0    0    0    0    0    1    0    0     0
 [7,]    0    0    0    0    0    1    0    0    0     0
 [8,]    0    0    0    0    0    0    0    0    0     0
 [9,]    1    0    0    1    0    0    0    0    0     0
[10,]    0    0    0    1    0    0    0    0    0     0
```

Figure 7
10 x 10 binary matrix with an optimal solution of 3

- **ineq():** This function returns a vector with the trivial and optimal solution of a system of inequation. The code of the function (Figure 5) is based in the algorithm to solve symmetric binary matrix. An example of how to use the ineq() function is presented in Figure 8.

```
> ineq(test(10,3))
 [1] 1 2 1 3 3 3 1 3 2 2
>
```

Figure 8
The solution of a symmetric binary matrix with cardinality 3

- **importDIMACSAscii():** This function imports a DIMACS Graph Coloring Instance Ascii file which should have a .col as extension [6]. The method will be expecting a filename as an argument, but in case it is not provided, a popup window will be shown to allow the user to browse the file. After the file has been selected or provided, the method will import the data to the memory, and using the internal functions of the package, it creates an empty (value 0) symmetric matrix. Once the matrix has been created, it will assign a 1 in the location matrix(x,y) if that location was found in the DIMACS Graph Coloring Instance file (the locations in the file are the edges). The value returned will be a symmetric binary matrix that represents that Graphic Coloring Instance. Figure 9 gives an example of how can be used the importDIMACSAscii() function (the david.col [6] file was used for the example).

```
> i<-importDIMACSAscii()
> i[1:10,1:10]   #show only the data of 10 x 10
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    1    0    0    0    0    0     0
 [2,]    0    0    0    1    0    0    1    0    0     0
 [3,]    0    0    0    0    0    0    0    0    0     0
 [4,]    1    1    0    0    0    0    0    0    0     0
 [5,]    0    0    0    0    0    0    0    0    0     0
 [6,]    0    0    0    0    0    0    0    0    0     0
 [7,]    0    1    0    0    0    0    0    0    0     0
 [8,]    0    0    0    0    0    0    0    0    0     0
 [9,]    0    0    0    0    0    0    0    0    0     0
[10,]    0    0    0    0    0    0    0    0    0     0
```

Figure 9
Usage of importDIMACSAscii()

- **importDIMACSBin():** This function imports a DIMACS Graph Coloring Instance binary file which commonly have a .col.b extension. The process starts writing a file that contain the path and the filename that will be converted. Once the file is written, the function calls the method convertDIMACSFromBin2Ascii() inside the library to convert the binary information to ASCII. After the data have been converted, it uses the same internal methods as importDIMACSAscii() to create the matrix and import the data. Figure 10 gives an example of how to use this method (the file used in the example was DSJC125.5.col.b [6]).

```
> i<-importDIMACSBin()
> i[1:10,1:10]
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    0    0    0    0    1    0    0    0    0     0
 [2,]    0    0    0    0    0    1    0    0    0     0
 [3,]    0    0    0    0    0    0    0    0    0     0
 [4,]    0    0    0    0    0    0    0    1    1     0
 [5,]    1    0    0    0    0    0    0    0    0     0
 [6,]    0    1    0    0    0    0    0    0    1     0
 [7,]    0    0    0    0    0    0    0    0    0     0
 [8,]    0    0    0    1    0    0    0    0    0     0
 [9,]    0    0    0    1    0    1    0    0    0     0
[10,]    0    0    0    0    0    0    0    0    0     0
```

Figure 10
Usage of importDIMACSBin()

Private Functions

As many systems packages, the Color Package of R Language has internal functions used only by some method of the package. The purpose of these internal methods is to reuse code without duplicating it. The internal functions of the package are listed and describe below.

- **readDIMACSFile():** This method reads the DIMACS Graph Coloring Instance ASCII file. It is used by importDIMACSAscii() and importDIMACSBin() functions. This is the main method of the import process, because is the one which open the file, reads the data, creates the matrix, call the function to fill the edges, call the function to validate the data, close the file and return the matrix with the data imported. Also, it's the only function that accesses the other two internal methods.
- **getDIMACSParameters():** This function look over the data imported for the graph parameters. The parameters are specified with a "p" at the beginning of the line. These parameters are the number of vertices and the number of edges that the graph contains. The value returned is a vector of two, which contains the size in the first position and the quantity of edges in the second position. If the parameters are not found, it returns a vector of size 0.
- **fillDIMACSMatrixEdges():** As the name said, this function run over the data imported and fill the edges in the matrix assigning a 1. The method should receive the matrix, the data, the quantity of Edges from the parameter and the size of the matrix from the parameter. Once it finished to import the edges into the matrix it

call the function checkDIMACSMatrix() to validate the data imported. After the data is verified, it returns the matrix.

- **checkDIMACSMatrix():** This function verifies if the matrix accomplish with the DIMACS standard. If it found something wrong, like if the matrix is not symmetric, or the quantity of edges in the parameter doesn't match the edges imported, it will show a warning. The function expects the following arguments: the matrix, the matrix size, the edges of the parameter, the quantity of edges imported. After it verifies the data, it returns the matrix.

The Color Library

To convert a DIMACS Graph Color Instance binary file into DIMACS Graph Color Instance ASCII file [6], so it can be imported to R Language, the Color Package uses a library that was coded in C Language. This library is compiled when the package is installed in R, bringing the advantage to be compiled depending in which environment has been installed.

A DIMACS Graph Color Instance binary file [6] actually is composed of ASCII and binary. The file consists in three sections, which the first 2 sections are ASCII and the last section is the binary code. These sections are [7]:

- **The first line:** this section tells let know the program or system that will used the file how many characters contains the Preamble.
- **Preamble:** is a section where the creator of the file can include information about the graph. The lines of information are tagged at the beginning with a "c" to let the user know that is a comment. Also, this section contains a line which let the user know the number of vertices and the edges in the graph (the format is "p type num_vertices num_edges [7]).
- **Binary block:** this section contains the lower triangular part of the vertex-vertex adjacency matrix of the graph. Each row is stored as sequence of bits, where the j'th bit is 1 if the edge (i, j) is in the graph, otherwise the bit is 0 [7].

The reason of the Color Library is that DIMACS Graph Color Instance binary file contains mixed data type (ASCII and binary), and the import of mixed data files in R is really complex. As C Language can manage better files, making an external library was a better solution to import the files. The only problem found in use an external library was that a string data type cannot be pass as a parameter to the method that convert the file because of an incompatibility from R Language to C Language. The solution to this problem was to export the filename to be converted to a text file, which will be read from the library, get the file name, and convert it. In the future, this approach will allow the users to convert batch of files instead of one file at a time.

The Color library consists of six methods. Five of the methods are private, used internal by the package, and one is the public method used by R, to convert the binary file to ASCII. What the library does is:

1. Get the name of the file to convert
2. Open the file to convert
3. Read the first line of the DIMACS binary file
4. Read the preamble of the DIMACS binary file
5. Get the number of vertices and edges
6. Put in memory the conversion of the binary block
7. Close the file to convert
8. Create a new DIMACS Graph Color Instance file in ASCII format.

Once the library was created, I use the method .C to call the method inside the library [10]. This method permits to use a method inside a library that was created with C Language. Previously, a loading of the package need to be made. For test purposes I used the dyn.load method, which load the library to R and make it available. After the package was build, the dyn.load was deleted, because the package will load the library automatically.

With the Color Library, the import of DIMACS Graph Color Instance binary files makes the Color Package unique and on top of other packages (like igraph [8]) that import DIMACS files, because this package manages DIMACS Graph Color Instance binary files.

BUILDING THE PACKAGE

R provides several shell scripts in order to build the package, but these shell scripts only works in Linux/Unix environment. In order to build a package in a Windows environment, it is necessary to install the R Tools [8].

After you have the necessary scripts installed, you need to create the package structure. The basic structure consists of one directory which should have the name of the package, a file 'DESCRIPTION' and the sub-directories 'R', 'data', 'demo', 'exec', 'inst', 'man', 'po', 'src', and 'tests' (some of which can be missing) [9]. The Color Package structure is shown in Figure 11.

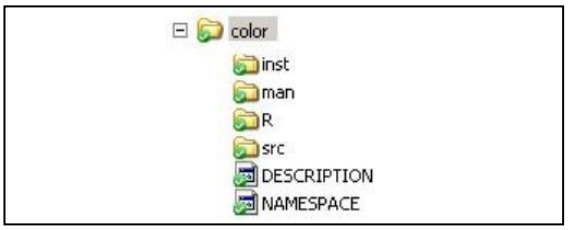

Figure 11
Color Package Structure

The DESCRIPTION file is required and it contains information related to the package, like package name, the version, the creation date, the author, the license, and others. The NAMESPACE file tells R to create the color library (a DLL or a SO file, depends of the environment) and to load it using the command useDynLib(). Also it permits to declare which functions will be available to the users using the command export().

Each of the sub-directories has a specific purpose. The following list describes each sub-directory and what it contains.

- **inst:** inside this directory are located all the pdf files that will be included in the package as references. When the package is build, these files will be move to the root of the package.
- **man:** this directory contains the manuals of each public method. The building process validates that each method have a manual. Each manual file should contains the name, the title, a description, the usage, the description of the arguments (if the method have arguments/parameters), the value returned and an example. The file supports Latex and the format to create it can be found in the *Writing R Extensions* manual.
- **R:** this directory has all the R code that constitutes the package.
- **src:** inside this directory is located the C Code which will be converted automatically to a DLL or SO file. The file of this name and the name used in the useDynLib() command should be the same.

After all the components were completed and verified, in order to create the package we use the command "R CMD build color". The previous command creates the package as a .tar.gz file, which can be used by a Unix/Linux environment (the color library will be a .so file). In order to create the package for a Windows environment the command to be used is "R CMD build --binary --use-zip color"; this will create a .zip file with a color library as a DLL file.

One best practice is to check the package before it is submitted to CRAN. This process will tell you as a warning any incompatible or error found. To make a check, the command to be used is: "R CMD check color". After the check ran, and no warning was found, I was able to submit the sources of the package to CRAN.

Once the package is created, it can be installed and loaded by any other user. In order to install in a Windows environment, use the "Install Packages from local zip file" located in the top menu inside Packages. After you install it, to make it usable, use the "Load package…" command inside the Packages menu, select color from the list and click OK button.

CONCLUSION

The gColor Package will bring R a method to import DIMACS Graph Color Instance binary files, which does not exist. This will provide other users to import this type of files to their methods. Another useful method is the validation of the files; this will allow verifying the DIMACS standards in the files imported. Also, the gColor Package brings a solution to solve System of Inequations, another attribute that make the package special and useful.

ACKNOWLEDGEMENT

I want to thank Prof. Jeffrey L. Duffany for bring me this opportunity and help me during the project.

REFERENCES

[1] Duffany, J.L., *Systems of Inequations*, 4th LACCEI Conference, Mayaguez, PR, June 21-23, 2006.

[2] "Wikipedia – R", http://en.wikipedia.org/wiki/R_Project

[3] Venables, W. N. and Smith, D. M., "Introduction to R", *An Introduction to R*, Version 2.9.1, Jun 26, 2009 http://cran.cnr.berkeley.edu/doc/manuals/R-intro.pdf

[4] "R Packages", http://cran.cnr.berkeley.edu/web/packages/

[5] "Wikipedia – Inequation", http://en.wikipedia.org/wiki/Inequation

[6] "DIMACS Graph Coloring Instances", http://mat.gsia.cmu.edu/COLOR/instances.html

[7] "DIMACS standard format", http://mat.gsia.cmu.edu/COLOR/format/README.binformat

[8] Ripley, B and Murdoch, D, "R Tools home page", http://www.murdoch-sutherland.com/Rtools/

[9] R Development Core Team, *Writing R Extensions*, http://cran.r-project.org/doc/manuals/R-exts.pdf

[10] R Development Core Team, *R Internals*, http://cran.r-project.org/doc/manuals/R-ints.pdf