A Quick Start for The R Interface to LINDO API

July 15, 2018

1 Introduction

The package *rLindo* is an R interface to LINDO API C functions. It supports Linear, Integer, Quadratic, Conic, General Nonlinear, Global, and Stochastic models.

2 Installation

We describe the following steps: a) Install R, b) Install LINDO API, c) Connect R and LINDO API, d) Test the connection on included examples, and e) Outline how to build your own optimization examples.

2.1 Installation Details

2.1.1 Windows

a) Download R, if you have not already, from https://cran.r-project.org. Installation is usually straightforward. The biggest decision is 32 bit vs. 64 bit. If you have a 64 bit computer, choose 64 bit.

b) Download LINDO API, if you have not already, from www.lindo.com. Installation is usually straightforward. The biggest decision is 32 bit vs. 64 bit. If you have a 64 bit computer, choose 64 bit. Also Make sure LINDOAPI_HOME points to the installation path of LINDO API. If not,

> set LINDOAPI_HOME="c:\Lindoapi"

c) Connecting R and LINDO API. You need to run the command script install-rlindo.bat. Under the default installation of LINDO API, this file is in directory: c:\Lindoapi\R. There are a variety of ways of running this script. One way is from a command line as follows:

Press the "Windows" key (4 dark squares) on the lower left of the keyboard.

Type the word “command” in the search bar that is displayed.
Click right button on “command prompt” and choose “Run as administrator”.
In the command prompt window that is displayed, type in:

cd c:\Lindoapi\R
install-rlindo

If all goes well you should see a message ending with:

*** testing if installed package can be loaded*
** DONE (rLindo)*
Finished

d) You can test if the connection works by running several rLindo optimization model scripts that are included in the LINDO distribution. In a default installation these scripts are in c:\Lindoapi\R\samples. One way of testing them is from the command prompt. Continuing from above, type:

cd samples (or also *cd c:\Lindoapi\R\samples*)

Start R by typing:

R

Tell R to connect to rLindo by typing:

library(rLindo)

2.1.2 Linux or other Unix-like systems

NOTE: Make sure to login as the 'root' user to perform these steps.

1. Install LINDO API 12.0 for the host platform (e.g. Linux 32 or 64-bit).
2. Install R (and R-devel if R headers are missing on your system).
3. Make sure $LINDOAPI_HOME points to the installation path of LINDO API 12.0. If not,

   $export LINDOAPI_HOME=“/home/lindoapi”

   To automatically set this variable, please add the above command to the ~/.bashrc file. Note that your LINDO API folder may lie in a folder other than /home/lindoapi, in which case, you should modify the above statement accordingly.
4. Navigate to the “lindoapi/R” directory where rLindo_12.0.tar.gz is located.
5. Run the following command to install the rLindo package.

   $./install-rlindo.sh
6. Test whether the rLindo package has been installed successfully with the following R-command:

   library(rLindo)

3 Usage

To use the package users must have a valid license file named lndapi110.lic under the folder LINDOAPI_HOME/license. The R interface function names use the convention of 'r' + name of LINDO API function, e.g. *rLScreateEnv* in the R interface corresponds to *LScreateEnv* in LINDO API. All LINDO parameters and constants are the same with LINDO API.

4 Complete Examples

4.1 An application to the least absolute deviations (LAD) estimation

4.1.1 Problem definition

Let
n = number of observations,
k = number of explanatory variables,
d_i = value of the dependent variable in observation i, for $i = 1, 2, ..., n$,
e_{ij} = value of the jth independent variable in observation i, for $i = 1, 2, ..., n$ and $j = 1, 2, ..., k$,
x_j = prediction coefficient applied to the jth explanatory variable,
ω_i = error of the forecast formula applied to the ith observation,

A LAD regression can be described as the following:
Minimize

$$|\omega_1| + |\omega_2| + |\omega_3| + ... + |\omega_n|$$

subject to

$$\omega_i = d_i - x_0 - \sum_{j=1}^{k} e_{ij} x_j$$

where ω_j, x_j are unconstrained in sign.
Linear programming can be applied to this problem if we define:

$$u_i - v_i = \omega_i$$

then the LAD regression model can be rewritten as:
Minimize

$$u_1 + v_1 + u_2 + v_2 + ... + u_n + v_n$$

subject to

$$u_i - v_i = d_i - x_0 - \sum_{j=1}^{k} e_{ij} x_j$$

where u_i and v_i are nonnegative, x_j are unconstrained in sign.

4.1.2 An example

We have five observations on a single explanatory variable,

d_i	e_{i1}
2	1
3	2
4	4
5	6
8	7

Then the linear programming model for the LAD regression is:

Minimize

$$U_1 + V_1 + U_2 + V_2 + U_3 + V_3 + U_4 + V_4 + U_5 + V_5$$

subject to

$$U_1 - V_1 = 2 - X_0 - X_1$$
$$U_2 - V_2 = 3 - X_0 - 2X_1$$
$$U_3 - V_3 = 4 - X_0 - 4X_1$$
$$U_4 - V_4 = 5 - X_0 - 6X_1$$
$$U_5 - V_5 = 8 - X_0 - 7X_1$$

All variables are nonnegative.

4.1.3 Solve the linear programming model in R

Using the R interface to LINDO API, we can solve the above linear programming model.

#load the package

```
> library(rLindo)
```

```
rLindo package for LINDO API 11.0.3301.172, built on Mar  2 2017 08:11:39
Barrier Solver Version 8.0.0.046, Nonlinear Solver Version 3.15P
Platform Linux 64x86 (D)
```

#create LINDO enviroment object

```
> rEnv <- rLScreateEnv()
```

#create LINDO model object

```
> rModel <- rLScreateModel(rEnv)
```

#disable printing log

```
> rLSsetPrintLogNull(rModel)

$ErrorCode
[1] 0
```

#number of variables

```
> nVars <- 12
```

#number of constraints

```
> nCons <- 5
```

#maximize or minimize the objective function

```
> nDir <- LS_MIN
```

#objective constant

```
> dObjConst <- 0.
```

#objective coefficients

```
> adC <- c(1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 0., 0.)
```

#right hand side coefficients of the constraints

```
> adB <- c( 2., 3., 4., 5., 8.)
```

#constraint types are all Equality

```
> acConTypes <- "EEEEE"
```

#number of nonzeros in LHS of the constraints

```
> nNZ <- 20
```

#index of the first nonzero in each column

```
> anBegCol <- c( 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 15, 20)
```

#nonzero coefficients of the constraint matrix by column

```
> adA <- c(1.0,-1.0,1.0,-1.0,1.0,-1.0,1.0,-1.0,1.0,-1.0,
+          1.0,1.0,1.0,1.0,1.0,1.0,2.0,4.0,6.0,7.0)
```

#row indices of the nonzeros in the constraint matrix by column

```
> anRowX <- c(0,0,1,1,2,2,3,3,4,4,0,1,2,3,4,0,1,2,3,4)
```

#lower bound of each variable (X0 and X1 are unconstrained)

```
> pdLower <- c(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -LS_INFINITY, -LS_INFINITY)
```

#load the data into the model object

```
> rLSloadLPData(rModel, nCons, nVars, nDir, dObjConst, adC, adB, acConTypes,
+               nNZ, anBegCol, NULL, adA, anRowX, pdLower, NULL)

$ErrorCode
[1] 0
```

#solve the model.

```
> rLSoptimize(rModel,LS_METHOD_FREE)

$ErrorCode
[1] 0

$pnStatus
[1] 2
```

#retrieve value of the objective and display it

```
> rLSgetDInfo(rModel,LS_DINFO_POBJ)

$ErrorCode
[1] 0

$pdResult
[1] 2.666667
```

#get primal solution and display it

```
> rLSgetPrimalSolution(rModel)

$ErrorCode
[1] 0

$padPrimal
 [1] 0.0000000 0.0000000 0.3333333 0.0000000 0.0000000 0.0000000 0.0000000
 [8] 0.3333333 2.0000000 0.0000000 1.3333333 0.6666667
```

#get dual solution and display it

```
> rLSgetDualSolution(rModel)

$ErrorCode
[1] 0

$padDual
[1] -0.3333333  1.0000000 -0.6666667 -1.0000000  1.0000000
```

#delete enviroment and model objects to free memory

```
> rLSdeleteModel(rModel)

$ErrorCode
[1] 0

> rLSdeleteEnv(rEnv)

$ErrorCode
[1] 0
```

Then the optimal value for X_0 and X_1 specify the prediction formula:

$$d_i = 1.3333 + 0.666667 e_{i1}$$