A Quick Start for The Python Interface to LINGO API

June 23, 2018

1 Introduction

The package *pyLingo* is a Python interface to LINGO API functions, which gives you the ability to run a LINGO command script to access all the major features of LINGO.

LINGO is an algebraic modeling language linked to an array of optimization engines, or solvers. It is a tool for utilizing the power of linear, integer and nonlinear optimization to formulate large problems concisely, solve them, and analyze the solution. Optimization helps you find the answer that yields the best result; attains the highest profit, output, or happiness; or the one that achieves the lowest cost, waste, or discomfort. Often these problems involve making the most efficient use of your resources including money, time, machinery, staff, inventory, and more. For more information on LINGO, please refer to www.lindo.com.

2 Installation

The package is located at the folder "\Lingo18\Programming Samples". To install the package, it requires the installation of LINGO 18 as well. See file INSTALL for details of the installation and platform specifications.

3 Usage

In the LINGO interface for Python, the names of functions use the convention: 'py' + name of LINGO API function, e.g. *pyLScreateEnvLng* in the Python interface corresponds to *LScreateEnvLng* in LINGO API.

4 General commands

To load the package, use sentence:
*from pyLindo import **

To generate a LINGO API environment object, use the command:
pEnv = lingo.pyLScreateEnvLng()

5 An application to the Acceptance Sampling Design

We are sampling items from a large lot. If the number of defectives in the lot is 3% or less, the lot is considered ”good”. If the defects exceed 8%, the lot is considered ”bad”. We want a producer risk (probability of rejecting a good lot) below 9% and a consumer risk (probability of accepting a bad lot) below 5%. We need to determine N and C, where N is the minimal sample size, and C is the critical level of defects such that, if defects observed in the sample are less-than-or-equal-to C, we accept the lot. Note that we make use of LINGO's cumulative Poisson distribution function as an approximation of the Binomial distribution.

Model: samsizr

```
MODEL:
! Acceptance Sampling Design;
! From a large lot, take a sample of size N, accept if C or less are defective;
! Poisson approximation to number defective is used;

DATA:
   AQL = @POINTER( 1); ! ”Good” lot fraction defective;
   LTFD = @POINTER( 2); ! ”Bad” lot fraction defective;
   PRDRISK = @POINTER( 3); ! Tolerance for rejecting good lot;
   CONRISK = @POINTER( 4); ! Tolerance for accepting bad lot;
   MINSMP = @POINTER( 5); ! Lower and upper bounds on sample size;
   MAXSMP = @POINTER( 6);
ENDDATA

   [OBJ] MIN = N;
! Tolerance for rejecting a good lot;
   1 - @PPOISCDF( N * AQL, C) <= PRDRISK;
! Tolerance for accepting a bad lot;
   @PPOISCDF( N * LTFD, C) <= CONRISK;
! Give solver some help in getting into range;
   @BND( MINSMP, N, MAXSMP);
   @BND( 1, C, MAXSMP);
```

```
! Make variables general integer;
    @GIN( N); @GIN( C);
DATA:
    @POINTER( 7) = N;
    @POINTER( 8) = C;
    @POINTER( 9) = @STATUS();
ENDDATA

END
```

Then using the Python interface to LINGO API, we can solve the above integer, non-linear optimization model.

```
from pyLingo import *

def samsizr(AQL,LTFD,PRDRISK,CONRISK,MINSMP,MAXSMP):
    #create Lingo enviroment object
    pEnv = lingo.pyLScreateEnvLng()
    if pEnv is None:
        print("cannot create LINGO environment!")
        exit(1)

    #open LINGO's log file
    errorcode = lingo.pyLSopenLogFileLng(pEnv,'samsizr.log')
    if errorcode != const.LSERR_NO_ERROR_LNG:
     print("errorcode = ", errorcode)
     exit(1)

    #pass memory transfer pointers to LINGO
    #define pnPointersNow
    pnPointersNow = N.array([0],dtype=N.int32)

    #@POINTER(1)
    AQL_1 = N.array([AQL],dtype=N.double)
    errorcode = lingo.pyLSsetDouPointerLng(pEnv, AQL_1, pnPointersNow)
    if errorcode != const.LSERR_NO_ERROR_LNG:
        print("errorcode = ", errorcode)
        exit(1)

    #@POINTER(2)
    LTFD_1 = N.array([LTFD],dtype=N.double)
    errorcode = lingo.pyLSsetDouPointerLng(pEnv, LTFD_1, pnPointersNow)
    if errorcode != const.LSERR_NO_ERROR_LNG:
```

```
    print(”errorcode = ”, errorcode)
    exit(1)

#@POINTER(3)
PRDRISK_1 = N.array([PRDRISK],dtype=N.double)
errorcode = lingo.pyLSsetDouPointerLng(pEnv, PRDRISK_1, pnPointersNow)
if errorcode != const.LSERR_NO_ERROR_LNG:
    print(”errorcode = ”, errorcode)
    exit(1)

#@POINTER(4)
CONRISK_1 = N.array([CONRISK],dtype=N.double)
errorcode = lingo.pyLSsetDouPointerLng(pEnv, CONRISK_1, pnPointersNow)
if errorcode != const.LSERR_NO_ERROR_LNG:
    print(”errorcode = ”, errorcode)
    exit(1)

#@POINTER(5)
MINSMP_1 = N.array([MINSMP],dtype=N.double)
errorcode = lingo.pyLSsetDouPointerLng(pEnv, MINSMP_1, pnPointersNow)
if errorcode != const.LSERR_NO_ERROR_LNG:
    print(”errorcode = ”, errorcode)
    exit(1)

#@POINTER(6)
MAXSMP_1 = N.array([MAXSMP],dtype=N.double)
errorcode = lingo.pyLSsetDouPointerLng(pEnv, MAXSMP_1, pnPointersNow)
if errorcode != const.LSERR_NO_ERROR_LNG:
    print(”errorcode = ”, errorcode)
    exit(1)

#@POINTER(7)
NN = N.array([-1.0],dtype=N.double)
errorcode = lingo.pyLSsetDouPointerLng(pEnv, NN, pnPointersNow)
if errorcode != const.LSERR_NO_ERROR_LNG:
    print(”errorcode = ”, errorcode)
    exit(1)

#@POINTER(8)
C = N.array([-1.0],dtype=N.double)
errorcode = lingo.pyLSsetDouPointerLng(pEnv, C, pnPointersNow)
if errorcode != const.LSERR_NO_ERROR_LNG:
    print(”errorcode = ”, errorcode)
    exit(1)
    #@POINTER(9)
    Status = N.array([-1.0],dtype=N.double)
    errorcode = lingo.pyLSsetDouPointerLng(pEnv, Status, pnPointersNow)
    if errorcode != const.LSERR_NO_ERROR_LNG:
        print("errorcode = ", errorcode)
        exit(1)

    #Run the script
    cScript = "SET ECHOIN 1 \n TAKE samsizr.lng \n GO \n QUIT \n"
    errorcode = lingo.pyLSexecuteScriptLng(pEnv, cScript)
    if errorcode != const.LSERR_NO_ERROR_LNG:
        print("errorcode = ", errorcode)
        exit(1)

    #Close the log file
    errorcode = lingo.pyLScloseLogFileLng(pEnv)
    if errorcode != const.LSERR_NO_ERROR_LNG:
        print("errorcode = ", errorcode)
        exit(1)

    if Status[0] == const.LS_STATUS_GLOBAL_LNG:
        print("\nGlobal optimum found!")
    elif Status[0] == const.LS_STATUS_LOCAL_LNG:
        print("\nLocal optimum found!")
    else:
        print("\nSolution is non-optimal\n")

    #check solution
    print("\nThe Optimal sample size is ",NN,".\nAccept the lot if ",
     C," or less defectives in sample.\n\n")

    #delete Lingo enviroment object
    errorcode = lingo.pyLSdeleteEnvLng(pEnv)
    if errorcode != const.LSERR_NO_ERROR_LNG:
        exit(1)

##################################################
if __name__ == '__main__':
    samsizr(0.03,0.08,0.09,0.05,125.0,400.0)
```