Package ‘MMLPDEA’

March 7, 2017

Type Package
Title FORTRAN LP interface and DEA-DDEA modeling and bootstrapping
Version 1.07
Date 2017-3-06
Author Joe Atwood
Maintainer Joe Atwood <jatwood@montana.edu> Github Project:Erik Rupert <erik.rupert1@gmail.com>
Depends Benchmarking,lpSolveAPI
Description Provides access to FORTRAN LP code and for DEA-DDEA modeling and bootstrapping.
License LGPL-2.1
LazyLoad yes
NeedsCompilation yes
Archs i386, x64

R topics documented:

MMLPDEA-package .......................................................... 2
DDEAnCm ................................................................. 3
DDEAnCm_write ........................................................... 6
DEAboot ................................................................. 8
DEAboot_write ........................................................... 10
lagMat ................................................................. 11
lagum ................................................................. 13
LP_DF ................................................................. 14
MMLP ................................................................. 15
nCm ................................................................. 21
newseed ............................................................... 22
normgen ............................................................... 22
RDDEAnCm ............................................................ 24
sampleum ............................................................... 27
seconds ............................................................... 28
ugen ................................................................. 28

Index 30
MMLPDEA-package

MMLPDEA

Description

Uses FORTRAN to bootstrap input/output/DEA/DDEA models.

Also provides an interface to Morris and Miller’s open source linear programming Fortran code.

Details

Package: MMLPDEA
Type: Package
Version: 1.07
Date: 2017-3-06
License: LGPL-2.1
LazyLoad: yes

Author(s)

R-Fortran bootstrapping and random number generation interface/fortran code: Joe Atwood
Correction to Miller’s code: Joe Atwood August 2015

The author makes no guarantee nor assumes any liability with respect to the accuracy of the results obtained from the code included in this package.

This material is based upon work partially supported by the National Institute of Food and Agriculture, U.S. Department of Agriculture, Hatch Project under 1002636 and Hatch/Multi-State Project under 1005034.

Alan Miller fortran code obtained from: http://jblevins.org/mirror/amiller/

Site Statement: “This is an archived copy of the Fortran source code repository of Alan Miller previously located at http://users.bigpond.net.au/amiller/. It is hosted by Jason Blevins with permission. The site has been slightly reformatted, but the source code and descriptions below have not been modified. All code written by Alan Miller is released into the public domain.”

Fortran linear programming code listing:
"smplx.f90 Linear programming using the simplex algorithm. This is a translation of the Fortran 66 program from the NSWC (Naval Surface Warfare Center) library written by Alfred Morris. There is also a simple test program t_smplx.f90. Needs the module constant.f90 which defines the precision and returns certain machine constants."

Fortran random number generation code listing:
"mt19937.f90 The ‘Mersenne Twister’ random number generator from Japan with a cycle of length (2^19937 - 1). mt19937a.f90 is a version for compilers which stop when there are integer overflows,
as some do when compiler check options are enabled for debugging purposes. ...

GPL license statement contained in mt19937.f90 code:

```
! A Fortran-program for MT19937: Real number version
! Code converted using TO_F90 by Alan Miller ! Date: 1999-11-26 Time: 17:09:23 ! Latest version dated 26 January 2002 ! This version assumes that integer overflows do NOT cause crashes. ! This version is compatible with Lahey's ELF90 compiler, and should be compatible with most full Fortran 90 or 95 compilers. ! Notice the strange way in which umask is specified for ELF90.

! This library is free software; you can redistribute it and/or modify it under the terms of the GNU Library General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Library General Public License for more details.

! Copyright (C) 1997 Makoto Matsumoto and Takuji Nishimura. ! When you use this, send an email to: matumoto@math.keio.ac.jp ! with an appropriate reference to your work.

```

### DDEAnCm

**Fortran nCm bootstrapping of DDEA models**

#### Description

Fortran nCm bootstrapping of DDEA models. Uses our modification of Geyer's subsampling bootstrap suggestion to increase computational efficiency.

**Note:** Although this function allows the user to complete a nCm subsampling process for multiple DMU's at the same time, it is recommended that the user complete this process on one DMU at a time and that the user carefully examine the boxplots of the bootstrapped values. Our endogenous process for determining the subsample sizes in mlist may need to be overridden by the user's exogenously generated mlist when a given DMU's efficiency scores are close to a boundary such as one for the input model or zero for the DDEA model.

#### Usage

```
DDEAnCm(X,Y,orient='ddea',RTS='crs',nboot=250,bootlist=NULL,DX=NULL,DY=NULL,mlist=NULL,
mcells=10,seedval=1001,replacement=FALSE,MMLPV=2,alpha=0.05,CILag=1,plotum=FALSE,plottxt='",
plotlist=NULL,itermax=1000,pulDATA=FALSE)
```
Arguments

X  An nDMU x nX matrix of Input observations
Y  An nDMU x nY matrix of Output observations
orient  Input efficiency "in" output efficiency "out"
RTS  Returns to Scale: "vrs","drs","crs", and "irs"
nboot  Number of bootstraps to complete for each sample size m.
bootlist  list of nDMUboot DMU’s to bootstrap. Set to 1:nDMU if no entry.
DX  An nDMUboot x nX matrix of Input directions. Set internally if no entry.
DY  An nDMUboot x nY matrix of Output directions. Set internally if no entry.
mlist  list of subsample sizes m. If NA, an mlist will be generated internally
mcells  number of mlevels to use or construct
seedval  A positive 32-bit integer
replaceum  Sample with replacement
MMLPV  1=Miller’s original code, 2=Miller’s corrected code (Atwood-2015)
alp ha  Confidence Interval prob
CILag  Lag for m interval selection process
plotum  plot CI diagnostics use plotum=TRUE to plot
plottxt  text to be included in plot
plotlist  vector of dmus to plot
itermax  iteration limit for LP for each mememeber of bootlist
pullDATA  compute LPsol,DUAL, and "reduced cost" if TRUE

Value

effvals  Vector of Efficiency Scores
effvals.bc  Vector of Bias-Corrected Efficiency Scores
bias  Vector of estimated bias levels
mlist  list of sample sizes m
boot  nDMUboot by length(mlist) by nboot array of bootstrapped efficiency scores
mchosen  chosen m interval
alpha  alpha level computed
beta  beta level computed
CI  Confidence Intervals
effstatus  Status of Efficiency Scores indstat = 0 the problem was solved; indstat = 1 the problem has no solution; indstat = 2 itermax iterations were performed-more needed; indstat = 3 sufficient accuracy could not be maintained to solve the problem; indstat = 4 the problem has an unbounded solution; indstat = 5 input error detected; indstat = 6 the solution may have been obtained;
bootstatus  status array of bootstrapped efficiency scores (equal in dimension to boot array)
LPsol  matrix of LP solutions for constraint matrix without slack variables
DUALS  matrix of dual values-INPUT DUALS then OUTPUT DUALS
RC  matrix of "reduced costs"
seedval  seedval used
iternum  number of first stage LP iterations for each dmu in bootlist
iternumboot  number of bootstrap stage LP iterations
Author(s)

Joe Atwood

References


Examples

```r
## not run

# # "Note:" These examples have been conducted with a dense primal
# # DEA or DDEA problem. In our experience MMLP is not time competitive
# # with lpSolveAPI or other R lp packages when solving the
# # sparse dual DEA problem with nDMU=1 constraints.
# # simulate coverage level for DDEAnCm confidence intervals
# # require(MMLPDEA)
# # set.seed(1001)
# # model inputs
# nDMU=1000
# nIN=3
# nOUT=1
# RTS='VRS'
# delta=0.8
# nsims=100
# nboot=2000
# alpha=0.05
# CI=matrix(0,nsims,2)
# directional efficiency score for DMU 0
# eff=0.50
# DDEA "efficient" input/output values for DMU 0
# (xe=rep(10,nIN))
# (ye=prod(xe^(1/nIN))*delta)
# [1] 6.369573
# with in-out model dy=0 and dx=x0
# ye=y0+eff*y0 <= ye=(1+eff)*y0 <= ye=ye/(1+eff) with 0<=eff
# xe=x0-eff*dx <= xe=x0-eff*x0 <= xe=(1-eff)*x0 <= xe=xe/(1-eff) with 0<=eff<1
# Generate "inefficient point for DMU 0
```
Create data file for use with DDEAnCm Fortran interface
**Description**

Creates/writes data file that can be read by the DDEAnCm_MASTER interface available (but commented out) in the package’s source code MMLPDEA_subroutines.f90 file. To use this data, copy the DDEAnCm_Master code from the source file, uncomment the code, and save a copy in an directory. You can then use Simply Fortran, Visual Studio, CodeBlocks, or similar software to compile and step into the fortran code.

**Usage**

```fortran
DDEAnCm_write(X,Y,orient='ddea',RTS='crs',nboot=250,bootlist=NULL,DX=NULL,DY=NULL,mlist=NULL,
mcells=10,seedval=1001,replaceum=FALSE,MMLPV=2,alpha=0.05,CILag=1,plotum=FALSE,plottxt=''
,itermax=1000,pullDATA=FALSE,fname='DDEAnCm_data.csv')
```

**Arguments**

- **X**: An nDMU x nX matrix of Input observations
- **Y**: An nDMU x nY matrix of Output observations
- **orient**: Input efficiency "in" output efficiency "out"
- **RTS**: Returns to Scale: "vrs","drs","crs", and "irs"
- **nboot**: Number of bootstraps to complete for each sample size m.
- **bootlist**: list of nDMUboot DMU’s to bootstrap. Set to 1:nDMU if no entry.
- **DX**: An nDMUboot x nX matrix of Input directions. Set internally if no entry.
- **DY**: An nDMUboot x nY matrix of Output directions. Set internally if no entry.
- **mlist**: list of subsample sizes m. If NA, an mlist will be generated internally
- **mcells**: number of mlevels to use or construct
- **seedval**: A positive 32-bit integer
- **replaceum**: Sample with replacement
- **MMLPV**: 1=Miller’s original code, 2=Miller’s corrected code (Atwood-2015)
- **alpha**: Confidence Interval prob
- **CILag**: Lag for m interval selection process
- **plotum**: plot CI diagnostics use plotum=TRUE to plot
- **plottxt**: text to be included in plot
- **itermax**: iteration limit for LP for each memeber of bootlist
- **pullDATA**: compute LPsol,DUAL,and "reduced cost"if TRUE
- **fname**: file name for fortran data

**Author(s)**

Joe Atwood
DEAboot

Fortran bootstrapping of input/output DEA models

Description

Fortran bootstrapping of input/output DEA models

Usage

DEAboot(X,Y,orient='in',RTS='crs',nboot=250,bootlist=NULL,alpha=0.05,
seedval=1001,MMLPV=2,itermax=1000,pullDATA=FALSE)

Arguments

X An nDMU x nX matrix of Input observations
Y An nDMU x nY matrix of Output observations
orient Input efficiency "in" output efficiency "out"
RTS Returns to Scale: "vrs","drs","crs", and "irs"
nboot Number of bootstraps to complete. nboot=0 calculates DEA efficiency scores
for each dmu but does not bootstrap the results
bootlist list of DMU’s to bootstrap. Set to 1:nDMU if no entry.
alpha Desired confidence interval
seedval A positive 32-bit integer
MMLPV 1=Miller’s original code, 2=Miller’s corrected code (Atwood-2015)
itermax iteration limit for the LP
pullDATA compute LPsol,DUAL, and "reduced cost" if TRUE

Value

h bootstrapping h value
effvals Vector of Efficiency Scores
effvals.bc Bias Corrected Vector of Efficiency Scores
bias estimated bias
var parameter variances - see Benchmarking package
boot nDMUboot by nboot matrix of bootstrapped efficiency scores
alpha alpha level computed
CI confidence intervals
effstatus Status of Efficiency Scores indstat = 0 the problem was solved; indstat = 1 the
problem has no solution; indstat = 2 itermax iterations were performed-more
needed; indstat = 3 sufficient accuracy could not be maintained to solve the
problem; indstat = 4 the problem has an unbounded solution; indstat = 5 input
error detected; indstat = 6 the solution may have been obtained;
bootstatus status matrix of bootstrapped efficiency scores
LPsol matrix of LP solutions
DUALS matrix of dual values.INPUT DUALS then OUTPUT DUALS
RC matrix of "reduced costs" for constraint matrix without slack variables
seedval seedvalue used in Fortran bootstrapping
iternum number of first stage LP iterations for each dmu in bootlist
iternumboot number of bootstrap stage LP iterations

Author(s)
Joe Atwood

References
Peter Bogetoft and Lars Otto; *Benchmarking with DEA, SFA, and R*; Springer 2011.

Examples

```r
#not run
# ################################################################################
# # NOTE:These examples have been conducted with a dense primal
# # DEA or DDEA problem. In our experience MMLP is not time competitive
# # with lpSolveAPI or other R lp packages when solving the
# # sparse dual DEA problem with nDMU+1 constraints.
# ################################################################################

# simulate coverage level for DEAboot confidence intervals
# ################################################################################
#set.seed(1001)
#model inputs
# ################################################################################
#nDMU=1000
#nIN=3
#nOUT=1
#orient='in'
#RTS='VRS'
#delta=0.8
#nsims=100
#nboot=2000
#alpha=0.05
#CI=matrix(0,nsims,2)
#input efficiency score for DMU 0
#eff=0.50
#DEA "efficient" input values for DMU 0
#(xe=rep(10,nIN))
#([[1] 10 10 10
#(ye=prod(xe^((1/nIN)))*delta)
#([[1] 6.389573
```

```r
# ################################################################################
```
## Generate "inefficient point for DMU 0"
#
#
#
## Generate "efficient points" for population of DMUs
#
#
## Generate nDMU points on efficient frontier
#
#
## Generate "inefficient points" for population of DMUs
#
#
## Generate DEA efficiency scores for population
#
#
## put DMU 0 data in matrices
#
#
## tmp=DEAboot(X,Y,orient='in',RTS=RTS,nboot=nboot,booldist=1,alpha=alpha)
#
## C11[,]=tmp$C1
## INC1=ifelse(eff0>C11[,1]&eff0<=CI1[,2],1,0)
## (cover=round(mean(INCI),3))
## txt=paste0('eff0',eff0,'rep',ns,'of',nsims,'coverest=',cover)
## M=cbind(as.matrix(C11[,],eff0)
## matplot(M,type='l',lty=1,col=c(1,1,2),lwd=c(1,1,3),main=txt,ylab='CI')
#
## end loop for(ns in 1:nsims)
## time2=seconds()
## (cover=mean(INCI))
## c1[1] 0.89
## 1-alpha
## c1[1] 0.95
## time2=timel
## c1[1] 67.85
**lagMat**

**Description**

Creates/writes data file that can be read by the DEAboot_MASTER interface available (but commented out) in the package's source code MMLPDEA_subroutines.f90 file. To use this data, copy the DEAboot_Master code from the source file, uncomment the code, and save a copy in a directory. You can then use Simply Fortran, Visual Studio, CodeBlocks, or similar software to compile and step into the fortran code.

**Usage**

```
DEAboot_write(X,Y,orient='in',RTS='crs',nboot=250,bootlist=NULL,alpha=0.05,
seedval=1001,MMLPV=2,itermax=1000,pullDATA=FALSE,fname='DEAboot_data.csv')
```

**Arguments**

- **X**
  - An nDMU x nX matrix of Input observations
- **Y**
  - An nDMU x nY matrix of Output observations
- **orient**
  - Input efficiency "in" output efficiency "out"
- **RTS**
  - Returns to Scale: "vrs","drs","crs", and "irs"
- **nboot**
  - Number of bootstraps to complete. nboot=0 calculates DEA efficiency scores for each dmu but does not bootstrap the results
- **bootlist**
  - list of DMU’s to bootstrap. Set to 1:nDMU if no entry.
- **alpha**
  - Desired confidence interval
- **seedval**
  - A positive 32-bit integer
- **MMLPV**
  - 1=Miller’s original code, 2=Miller’s corrected code (Atwood-2015)
- **itermax**
  - iteration limit for the LP
- **pullDATA**
  - compute LPsol,DUAL,and "reduced cost"if TRUE
- **fname**
  - file name for fortran data

**Author(s)**

Joe Atwood

---

**lagMat**

Generate a non-time series lagged matrix

**Description**

lagMat generates a lagged matrix

**Usage**

```
lagMat(x, lags=2, Lzero='F')
```

**Arguments**

- **x**
  - vector of data to be lagged
- **lags**
  - specifies the lag length(s)
- **Lzero**
  - include the zero lag vector in matrix
Value

lagMat returns a lagged matrix – See examples below

Author(s)

Joe Atwood

Examples

```r
#not run
#----------------------------------------------------------------------------------
# (x=1:10)
# [1] 1 2 3 4 5 6 7 8 9 10
#lagum(x,1)
# [1] NA 1 2 3 4 5 6 7 8 9
#lagum(x,3)
# [1] NA NA NA 1 2 3 4 5 6 7
#uplag(x)
# [1] 2 3 4 5 6 7 8 9 10 NA
#lagum(x,-1)
# [1] 2 3 4 5 6 7 8 9 10 NA
#----------------------------------------------------------------------------------
#lagMat(x,2)
# [,1] [,2]
# [1,] NA NA
# [2,] 1 NA
# [3,] 2 1
# [4,] 3 2
# [5,] 4 3
# [6,] 5 4
# [7,] 6 5
# [8,] 7 6
# [9,] 8 7
#[10,] 9 8
#lagMat(x,Lzero='T')
# [,1] [,2] [,3]
# [1,] 1 NA NA
# [2,] 2 1 NA
# [3,] 3 2 1
# [4,] 4 3 2
# [5,] 5 4 3
# [6,] 6 5 4
# [7,] 7 6 5
# [8,] 8 7 6
# [9,] 9 8 7
#[10,] 10 9 8
#lagMat(x,-1:2)
# [1,] 2 1 NA NA
# [2,] 3 2 1 NA
# [3,] 4 3 2 1
# [4,] 5 4 3 2
# [5,] 6 5 4 3
# [6,] 7 6 5 4
# [7,] 8 7 6 5
# [8,] 9 8 7 6
```

**Description**

lagum generates a lagged vector

**Usage**

`lagum(x, nlag = 1)`

**Arguments**

- `x` vector of data to be lagged
- `nlag` specifies the lag length - can be negative for ‘uplag’

**Value**

lagum returns a vector – See examples below

**Author(s)**

Joe Atwood

**Examples**

```
# not run
#--------------------------------------------------------------------------------
#x=1:10
#lagum(x,1)
# [1] NA 1 2 3 4 5 6 7 8 9
#lagum(x,3)
# [1] NA NA NA 1 2 3 4 5 6 7
#uplag(x)
# [1] 2 3 4 5 6 7 8 9 10 NA
#--------------------------------------------------------------------------------
```
Example data to demonstrate breakdown in MMLP code.

Description
Example data to demonstrate breakdown in MMLP code.

Usage
data("LP_DF")

Format
A data frame with 6 observations on the following 8 variables.

- V1: a numeric vector
- V2: a numeric vector
- V3: a numeric vector
- V4: a numeric vector
- V5: a numeric vector
- V6: a numeric vector
- rest: a character vector
- rhs: a numeric vector

Details
Example data to demonstrate breakdown in MMLP code.

Examples
```
# not run
# data(LP_DF)
# require(lpSolve)
# (obj2=as.vector(t(LP_DF[6:1:6])))
# (A2=as.matrix(LP_DF[1:5,1:6]))
# (rest2=as.vector(LP_DF$rest[1:5]))
# (rhs2=as.vector(LP_DF$rhs[1:5]))
# MMLP(objtype="min", obj=obj2, A=A2, rest=rest2, rhs=rhs2, MMLPV=1)$objval
# lp("min",obj2,A2,rest2,rhs2)
# MMLP(objtype="min", obj=obj2, A=A2, rest=rest2, rhs=rhs2, MMLPV=2)$objval
```
Morris-Miller Fortran LP code interface

**Usage**

```r
MMLP(objtype='max', obj, a, rest, rhs, itermax=1000, nsims=1, MMLPV=2)
```

**Arguments**

- `objtype` character string ‘max’ or ‘min’
- `obj` vector of objective coefficients
- `a` matrix of constraint coefficients
- `rest` character vector of constraint signs ‘<=’, ‘>=’, or ‘=’
- `rhs` vector of RHS values
- `itermax` maximal number of LP iterations
- `nsims` number of repititions before returning results
- `MMLPV` 1=Miller’s original code, 2=Miller’s corrected code (Atwood-2015)

**Value**

- `objval` objective value
- `xvals` If `indstat = 0 or 6`, `xvals` returns the solution, the slack, and the surplus variable levels
- `duals` dual values
- `rc` "reduced costs"
- `indstat` `indstat = 0` the problem was solved; `indstat = 1` the problem has no solution; `indstat = 2` `itermax` iterations were performed-more needed; `indstat = 3` sufficient accuracy could not be maintained to solve the problem; `indstat = 4` the problem has an unbounded solution; `indstat = 5` input error detected; `indstat = 6` the solution may have been obtained;
- `iternum` number of iterations

**Author(s)**

R and Fortran interface, duals calculations: Joe Atwood

MMLP code:

WRITTEN BY ALFRED H. MORRIS JR.
NAVAL SURFACE WEAPONS CENTER
DAHLGREN, VIRGINIA

INITIAL VERSION DEC 1977
LAST UPDATE OCT 1990

Converted using F90 intrinsics by
Alan Miller
CSIRO Mathematical & Information Sciences
CLAYTON, VICTORIA, AUSTRALIA 3169
Latest revision - 5 February 1997
obtained from: http://jblevins.org/mirror/amiller/

Site Statement: "This is an archived copy of the Fortran source code repository of Alan Miller previously located at http://users.bigpond.net.au/amiller/. It is hosted by Jason Blevins with permission. The site has been slightly reformatted, but the source code and descriptions below have not been modified. All code written by Alan Miller is released into the public domain."

Fortran linear programming code listing:

"smplx.f90 Linear programming using the simplex algorithm. This is a translation of the Fortran 66 program from the NSWC (Naval Surface Warfare Center) library written by Alfred Morris. There is also a simple test program t_smplx.f90. Needs the module constant.f90 which defines the precision and returns certain machine constants."

NOTE: Atwood modified Miller’s code in August 2015 to correct solution error. See example below:

Examples

```fortran
# not run
#******************************************************************************
# Example 1
#******************************************************************************
# Demonstrate potential error in original MM LP code
#******************************************************************************
#data(LP_DF)
#require(IpSolve)
#(obj2=as.vector(t(LP_DF[6,1:6])))
#(A2=as.matrix(LP_DF[1:5,1:6]))
#(rest2=as.vector(LP_DF$rest[1:5]))
#(rhs2=as.vector(LP_DF$rhs[1:5]))
#MMLP(objtype='min',obj=obj2,A=A2,rest=rest2,rhs=rhs2,MMLPV=1)$objval
#lp("min",obj2,A2,rest2,rhs2)
#MMLP(objtype='min',obj=obj2,A=A2,rest=rest2,rhs=rhs2,MMLPV=2)$objval
#******************************************************************************
# # #
# # # Example 2
#******************************************************************************
# # Determine computation times required to solve the Wyndor example 100,000
# # times using loops with lpSolveAPI and MMLP and constrasting the
# # results to solving the Wyndor problem 100,000 times within the MMLP fortran code
# #******************************************************************************
# rm(list=ls())
# require(MMLPDEA)
# require(IpSolve)
# require(lpSolveAPI)
```
```r
# Construct problem
# nr = 3
# nc = 2
# objtype = 'max'
# objmax = 1
# obj = c(3, 5)
# A = matrix(c(
# 1, 0,
# 0, 2,
# 3, 2
# ), 3, 2, byrow = T)
# b = c(4, 12, 18)
# rest = c('<', '<', '<')

# setup lpSolveAPI object
# LP_API = make.lp(nrow = nr, ncol = nc)
# lp.control(LP_API, sense = objtype)
# set.objfn(LP_API, obj)
# for (i in 1:nr) {
#   set.row(LP_API, i, A[i, ])
# }
# set.constr.type(LP_API, rest)
# set.rhs(LP_API, b)

# solve with lpSolve
# tmp = lp(objtype = obj, A, b, compute.sens = TRUE)
# tmp$objval, tmp$solution, tmp$duals[1:nr]
# [1] 36
# [2] 2 6
# [3] 0 0 1.5 1.0

# solve with lpSolveAPI
# (status = solve(LP_API))
# [1] 0
# get.objective(LP_API); get.variables(LP_API); get.dual.solution(LP_API)[2:(nr + 1)]
# [1] 36
# [2] 2 6
# [3] 0 0 1.5 1.0

# solve with MMLP
# tmp2 = MMLP(objtype = objtype, obj = obj, A = A, rest = rest, rhs = b)
# tmp2$objval; tmp2$xvals[1:nc]; tmp2$duals
# [1] 36
# [2] 2 6
# [3] 0 0 1.5 1.0
```
```r
# # time to obtain nsims solutions
# # timeP
# time0=seconds()
# # lpSolveAPI with loops
obj1=0
# for(j1 in 1:nsims){
#  set.objfn(LP_API,obj)
#  for(i in 1:nr){
#    set.row(LP_API,i,A[i,j])
#  }
#  set.constr.type(LP_API,rest)
#  set.rhs(LP_API,b)
#  (status=solve(LP_API))
#  obj1[j1]=get.objective(LP_API)
# }# end loop
# time1=seconds()
# # remove lpSolveAPI object
# delete.lp(LP_API)
# time2=seconds()
# # MMLP with loops
obj2=0
# for(j2 in 1:nsims){
#  tmp=MMLP(objtype=objtype,obj=obj,A=A,rest=rest,rhs=b)
#  obj2[j2]=tmp$objval
# }
# time3=seconds()
# # MMLP with internal loops
# tmp=MMLP(objtype=objtype,obj=obj,A=A,rest=rest,rhs=b,nsims=nsims)
# time4=seconds()
# time1-time0  # lpSolveAPI loop time
# #[1] 17.89
# time3-time2  # MMLP loop time
# #[1] 14.36
# time4-time3  # MMLP internal loop time
# #[1] 0.89
```

# Example 3: A more realistic example
# A DEA example that computes the efficiency score for each
# of 10000 DMUs. The example uses traditional looping
# with both lpSolve and MMLP and constrasts the "R" loop
# times to the results of using fortran loops within DEAbot
# (without running the bootstraps i.e. by setting nloop = 0)
# to obtain the efficiency score estimates for each DMU.
# require(MMLPDEA)
require(lpSolveAPI)
set.seed(2015)
## Define Cobb-Douglas technology with CRS
b1=0.5
b2=1-b1
## Generate "input levels"
x1=runif(nDMU,5,10)
x2=runif(nDMU,5,10)
## Generate "frontier" output levels
y1=x1*b1*x2*(1-b1)
## Generate "inverse output efficiency scores"
inv_eff=seq(0.25,1.0,length.out=nDMU)
## Contract output levels away from the efficient frontier
y1=inv_eff*y1
eff=1/inv_eff
## Put input and output quantities into matrices X and Y
X=as.matrix(cbind(x1,x2))
Y=as.matrix(y1)
## set up output orientation model for DMU 1 and VRS
objtype='max'
obj=c(rep(0,nDMU),1)
A=matrix(0,4,nDMU+1)
A[1,1:nDMU]=t(Y); A[1,(nDMU+1)]=-Y[1,1]
A[2:nDMU,1:nDMU]=t(X)
A[4,1:nDMU]=1
b=c(0,X[1,],1)
rest=c(">=','<=','#','=')
## set up lpSolveAPI object
LP_API=make.lp(nrow=nrow(A),ncol=ncol(A))
lp.control(LP_API,sense=objtype)
set.objfn(LP_API,obj)
for(i in 1:nrow(A)){
set.row(LP_API,i,A[i,])
}
set.constr.type(LP_API,rest)
set.rhs(LP_API,b)
##solve with lpSolveAPI
# (status=solve(LP_API))
# #[1] 0
# get.objective(LP_API)
# #[1] 3.976337
#solve with MMLP
# tmp2=MMLP(objtype=objtype, obj=obj, A=A, rest=rest, rhs=b)
# tmp2$objval
# #[1] 3.976337
#solve efficiency scores for all DMU's
# time_API=0; time_MMLP=0
# effhat_API=0; effhat_MMLP=0
#
# i=1
# for(i in 1:nDMU){
# A[1, ncol(A)]= -Y[i,1]
# b=c(0, X[i,1,1])
# time0=seconds()
# set.mat(LP_API, 1, ncol(A), -Y[i,1])
# set.rhs(LP_API, b)
# status=solve(LP_API)
# (effhat_API[i]=get.objective(LP_API))
# time1=seconds()
# time_API=time_API+(time1-time0)
#
# time0=seconds()
# tmp2=MMLP(objtype=objtype, obj=obj, A=A, rest=rest, rhs=b)
# (effhat_MMLP[i]=tmp2$objval)
# time1=seconds()
# time_MMLP=time_MMLP+(time1-time0)
#
# if(i%%10==0) plot(i, nDMU, main=paste('DMU, nDMU', i, nDMU))
#
# } C end loop "for(i in 1:nDMU)"
# Compute eff scores using DEAboot with nboot=0
# time0=seconds()
# tmp=DEAboot(X, Y, orient='out', RTS='vrs', nboot=0)
# time1=seconds()
# time_DEAboot=time1-time0
#summary(effhat_API-effhat_MMLP)
# # Min. 1st Qu. Median Mean 3rd Qu. Max.
# # -1.028e-09 -6.000e-13 0.000e+00 1.841e-10 7.000e-13 4.352e-07
#summary(effhat_MMLP-temp$objvals)
# # Min. 1st Qu. Median Mean 3rd Qu. Max.
# # 0 0 0 0 0 0
# time_API; time_MMLP; time_DEAboot
# #[1] 33.3 # R looping time lpSolveAPI (no bootstrapping)
# #[1] 12.22 # R looping time MMLP (no bootstrapping)
# #[1] 6.63 # Fortran looping time DEAboot (no bootstrapping)
# summary(effhat_API-effhat_MMLP)
# # Min. 1st Qu. Median Mean 3rd Qu. Max.
# # -1.028e-09 -6.000e-13 0.000e+00 1.841e-10 7.000e-13 4.352e-07
#summary(effhat_MMLP-temp$objvals)
# # Min. 1st Qu. Median Mean 3rd Qu. Max.
# # 0 0 0 0 0 0
# time_API; time_MMLP; time_DEAboot
# #[1] 33.3 # R looping time lpSolveAPI (no bootstrapping)
# #[1] 12.22 # R looping time MMLP (no bootstrapping)
# #[1] 6.63 # Fortran looping time DEAboot (no bootstrapping)
nCm

pulls nCm samples

Description

pulls nCm samples giving same results as in the Fortran DDEAnCm bootstrap code

Usage

nCm(nvals=1:10,m=5,replaceum=FALSE,seedval=1001)

Arguments

nvals values to sample from
m number of values to sample
replaceum sample with TRUE or without FALSE replacement
seedval seed value

Value

vector of m sampled values

Author(s)

Joe Atwood

Examples

```r
# not run
nCm(1:100,10)
```
newseed

Description

generates new 32 bit seed value from seedval \( u = \text{ugen}(1, \text{seedval}) \) \( \text{newseed} = \text{floor}(u[1] \times 2^{147483645}) \) if(newseed==0,newseed=1)

Usage

newseed(seedval)

Arguments

seedval seed value for fortran random number generator

Value

newseed new 32 bit seed value for fortran random number generator

Author(s)

Joe Atwood <jatwood@montana.edu>

Examples

newseed(1001)

normgen

generates normal random variates using uniform variates generated from Fortran code: mt19937.f90

Description

generates normal random variates using the uniform twister algorithm in Fortran code: mt19937.f90

Usage

normgen(n, seedval)

Arguments

\( n \) number of random numbers to simulate

seedval a positive 32 bit integer seedvalue

Value

Returns a vector of normal variates
Author(s)

R and Fortran interface to mt19937.f90 code: Joe Atwood
Alan Miller fortran code obtained from: http://jblevins.org/mirror/amiller/

Site Statement: "This is an archived copy of the Fortran source code repository of Alan Miller previ-ously located at http://users.bigpond.net.au/amiller/. It is hosted by Jason Blevins with permission. The site has been slightly reformatted, but the source code and descriptions below have not been modified. All code written by Alan Miller is released into the public domain."

Fortran random number generation code listing:

"mt19937.f90 The 'Mersenne Twister' random number generator from Japan with a cycle of length \((2^{19937} - 1)\). mt19937a.f90 is a version for compilers which stop when there are integer overflows, as some do when compiler check options are enabled for debugging purposes. ..... . mt19937.f90 was revised on 5 February 2002;"

GPL license statement contained in mt19937.f90 code:

! A Fortran-program for MT19937: Real number version
!
! Code converted using TO_F90 by Alan Miller ! Date: 1999-11-26 Time: 17:09:23 ! Latest revision - 5 February 2002 ! A new seed initialization routine has been added based upon the new ! C version dated 26 January 2002. ! This version assumes that integer overflows do NOT cause crashes. ! This version is compatible with Lahey's ELF90 compiler, ! and should be compatible with most full Fortran 90 or 95 compilers. ! Notice the strange way in which umask is specified for ELF90.

! genrand() generates one pseudorandom real number (double) which is ! uniformly distributed on [0,1]-interval, for each call. ! sgenrand(seed) set initial values to the working area of 624 words. ! Before genrand(), sgenrand(seed) must be called once. (seed is any 32-bit ! integer except for 0). ! Integer generator is obtained by modifying two lines. ! Coded by Takuji Nishimura, considering the suggestions by ! Topher Cooper and Marc Rieffel in July-Aug. 1997.

! This library is free software; you can redistribute it and/or modify it ! under the terms of the GNU Library General Public License as published by ! the Free Software Foundation; either version 2 ! of the License, or (at your option) any later version. This library is distributed in the hope that ! it will be useful, but WITHOUT ANY WARRANTY; without even the implied ! warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. ! See the GNU Library General Public License for more details. ! You should have received a copy of the GNU Library General Public License ! along with this library; if not, write to the Free Foundation, Inc., ! 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

! Copyright (C) 1997 Makoto Matsumoto and Takuji Nishimura. ! When you use this, send an email to: matumoto@math.keio.ac.jp ! with an appropriate reference to your work.


Examples

#----------------------------------------------------------
# x=normgen(100,seedval=2014)
**RDDEAnCm**

**R nCm bootstrapping of DDEA models**

**Description**

R nCm bootstrapping of DDEA models. Uses our modification of Geyer’s subsampling bootstrap suggestion to increase computational efficiency.

Comparison of R to Fortran nCm results and times Replicates Fortran procedures in R to facilitate user understanding of Fortran DDEAnCm process

**Usage**

```r
RDDEAnCm(X,Y,orient='ddea',RTS='crs',nboot=250,bootlist=NULL,DX=NULL,DY=NULL,mlist=NULL,
plotlist=NULL,mcells=10,seedval=1001,replaceum=FALSE,alpha=0.05,CILag=1,plotum=FALSE,plottxt='')
```

**Arguments**

- `X` An nDMU x nX matrix of Input observations
- `Y` An nDMU x nY matrix of Output observations
- `orient` Input efficiency "in" output efficiency "out"
- `RTS` Returns to Scale: "vrs","drs","crs", and "irs"
- `nboot` Number of bootstraps to complete for each sample size m.
- `bootlist` list of nDMUboot DMU’s to bootstrap. Set to 1:nDMU if no entry.
- `DX` An nDMUboot x nX matrix of Input observations. Set internally if no entry.
- `DY` An nDMUboot x nY matrix of Output observations. Set internally if no entry.
- `mlist` list of subsample sizes m. If NA, an mlist will be generated internally
- `mcells` number of mlevels to use or construct
- `seedval` A positive 32-bit integer
- `replaceum` Sample with replacement
- `alpha` Confidence Interval prob
- `CILag` Lag for m interval selection process
- `plotum` plot CI diagnostics use plotum=TRUE to plot
- `plottxt` text to be included in plot
- `plotlist` vector of dmus to plot

**Value**

- `effvals` Vector of Efficiency Scores
- `effvals.bc` Vector of Bias-Corrected Efficiency Scores
- `bias` Vector of estimated bias levels
- `mlist` list of sample sizes m
- `boot` nDMUboot by length(mlist) by nboot array of bootstrapped efficiency scores
- `mchosen` chosen m interval
- `alpha` alpha level computed
beta level computed

Confidence Intervals

Status of Efficiency Scores

indstat = 0 the problem was solved; indstat = 1 the problem has no solution; indstat = 2 itermax iterations were performed-more needed; indstat = 3 sufficient accuracy could not be maintained to solve the problem; indstat = 4 the problem has an unbounded solution; indstat = 5 input error detected; indstat = 6 the solution may have been obtained;

status array of bootstrapped efficiency scores (equal in dimension to boot array)

seedval used

Joe Atwood


# not run
# contrast and time R versus Fortran nCm bootstrap results
# require(MMLDEA)
# graphics.off()
# set.seed(101)
## model inputs
# nDMU=1000
# nIN=3
# nOUT=1
# delta=1
## directional efficiency score for given DMU1
# eff=0.50
## DDEA "efficient" input/output values for DMU1
# (xe=rep(10,nIN))
# [(y=prod(xe^((1/nIN))*delta))
## [1] 10 10 10
# (ye=prod(xe^((1/nIN))*delta))
## [1] 10
## With in-out model dy=y0 and dx=x0
# ye=y0+eff*dy with eff>0 <= ye=y0+eff*y0 <= ye=(1+eff)*y0 <= ye=y0/(1+eff) with eff>0
# xe=x0-eff*dx <= xe=x0-eff*x0 <= xe=(1-eff)*x0 <= xe=x0/(1-eff) with 0<=eff<1

Examples
### Generate "inefficient" point for DMU 1

(x[0]=xe/(1-eff0))

### [1] 20 20 20

(y[0]=ye/(1+eff0))

### [1] 6.666667

### Generate "efficient points" for population of DMUs

### Generate nDMU points on efficient frontier

# XE=matrix(runif(nDMU*nIN,5,15),nDMU,nIN)
# YE=matrix((apply(XE^((1/nIN),1,prod))*delta,nDMU,1)

### Generate "inefficient points" for population of DMUs

### Generate DEA efficiency scores for population

# eff=rbeta(nDMU,1,5)

# summary(ef)

# Min. 1st Qu. Median Mean 3rd Qu. Max.
# 0.0002952 0.0554500 0.1266000 0.1654000 0.2321000 0.7771000

### Put DMU data in matrices

# (X[1,]=x[0])
### [1] 20 20 20

# (Y[1,]=y[0])
### [1] 6.666667

### [1] eff[1]=0.9

### Estimate eff scores for all DMU's

# time1=seconds()

# tmp1=RDDEAnCm(X,Y,orient='inout',RTS='CRS',nboot=0,bootlist=1:nDMU)

# time2=seconds()

# tmp2=RDDEAnCm(X,Y,orient='inout',RTS='CRS',nboot=0,bootlist=1:nDMU)

# time3=seconds()

# summary(tmp1$effvals-tmp2$effvals)

### Min. 1st Qu. Median Mean 3rd Qu. Max.
### -6.05e-13 -5.762e-14 5.153e-14 4.751e-14 1.561e-13

### Max.

### 5.463e-13

### Plot(ef, tmp2$effvals)

### time2-time1

### [1] 1.7

### time3-time2

### [1] 0.13

### (time2-time1)/(time3-time2)

### [1] 13.07692

### Conduct, time, and contrast R versus Fortran nCm bootstraps

# time4=seconds()

# tmp3=RDDEAnCm(X,Y,orient='inout',RTS='CRS',nboot=2000,bootlist=1)

# time5=seconds()

# tmp4=RDDEAnCm(X,Y,orient='inout',RTS='CRS',nboot=2000,bootlist=1)

# time6=seconds()}
Sample with replacement using Fortran generated random numbers.

Description

Samples with replacement using Fortran generated random numbers.

Usage

```r
sampleum(nobs, x, seedval)
```

Arguments

- `nobs` sample size
- `x` values to sample from
- `seedval` seed value for fortran random number generator

Value

- `xsample` `nobs` values resampled from `x`

Author(s)

Joe Atwood <jatwood@montana.edu>

Examples

```r
sampleum(10, 1:5)
```
seconds  
Pulls seconds from system clock

Description
Pulls seconds from system clock.

Usage
seconds()

Arguments
none

Value
seconds from system clock

Author(s)
Joe Atwood <jatwood@montana.edu>

Examples
seconds()
Author(s)

R and Fortran interface to mt19937.f90 code: Joe Atwood

Alan Miller fortran code obtained from: http://jblevins.org/mirror/amiller/

Site Statement: "This is an archived copy of the Fortran source code repository of Alan Miller previously located at http://users.bigpond.net.au/amiller/. It is hosted by Jason Blevins with permission. The site has been slightly reformatted, but the source code and descriptions below have not been modified. All code written by Alan Miller is released into the public domain."

Fortran random number generation code listing:

"mt19937.f90 The 'Mersenne Twister' random number generator from Japan with a cycle of length (2^19937 - 1). mt19937a.f90 is a version for compilers which stop when there are integer overflows, as some do when compiler check options are enabled for debugging purposes. .... . mt19937.f90 was revised on 5 February 2002;"

GPL license statement contained in mt19937.f90 code:

! A Fortran-program for MT19937: Real number version
! Code converted using TO_F90 by Alan Miller ! Date: 1999-11-26 Time: 17:09:23 ! Latest revision - 5 February 2002 ! A new seed initialization routine has been added based upon the new ! C version dated 26 January 2002. ! This version assumes that integer overflows do NOT cause crashes. ! This version is compatible with Lahey's ELF90 compiler, ! and should be compatible with most full Fortran 90 or 95 compilers. ! Notice the strange way in which umask is specified for ELF90.

! genrand() generates one pseudorandom real number (double) which is ! uniformly distributed on [0,1]-interval, for each call. ! sgenrand(seed) set initial values to the working area of 624 words. ! Before genrand(), sgenrand(seed) must be called once. (seed is any 32-bit ! integer except for 0). ! Integer generator is obtained by modifying two lines. ! Coded by Takuji Nishimura, considering the suggestions by ! Topher Cooper and Marc Rieffel in July-Aug. 1997.

! This library is free software; you can redistribute it and/or modify it ! under the terms of the GNU Library General Public License as published by ! the Free Software Foundation; either version 2 of the License, or (at your ! option) any later version. This library is distributed in the hope that ! it will be useful, but WITHOUT ANY WARRANTY; without even the implied ! warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. ! See the GNU Library General Public License for more details. ! You should have received a copy of the GNU Library General Public License ! along with this library; if not, write to the Free Foundation, Inc., ! 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

! Copyright (C) 1997 Makoto Matsumoto and Takuji Nishimura. ! When you use this, send an email to: matumoto@math.keio.ac.jp ! with an appropriate reference to your work.

Examples

x=ugen(100,seedval=2014)
Index

*Topic MMLP
  LP_DF, 14

DDEAnCm, 3
DDEAnCm_write, 6
DEAboot, 8
DEAboot_write, 10

lagMat, 11
lagum, 13
LP_DF, 14

MMLP, 15
MMLPDEA-package, 2

nCm, 21
newseed, 22
normgen, 22

RDDEAnCm, 24
sampleum, 27
seconds, 28

ugen, 28
uplag (lagum), 13