Package 'qDEA'

June 5, 2023

Type Package

Title Quantile DEA

Version 1.0.0

Author Joe Atwood

Maintainer The package maintainer <jatwood@montana.edu>

Depends R (>= 4.2.1), dplyr, doBy

Description Quantile DEA functions written by Joe Atwood
Author recommends clpAPI be installed but can also be run with glpkAPI, highs, or Rglpk
Atwood, J., and S. Shaik. (2020) ``Theory and Statistical Properties of Quantile Data Envelopment Analysis." European Journal of Operational Research. 286:649-661.
Atwood, J., and S. Shaik. (2018) ``Quantile DEA: Estimating qDEA-alpha Efficiency Estimates with Conventional Linear Programming." Productivity and Inequality. Springer Press.

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

NeedsCompilation no

R topics documented:

`A2SM` *A2SM: Convert a matrix A to sparse matrix form*

Description

A2SM: Convert a matrix A to sparse matrix form

Usage

```
A2SM(A, SMM = "CRI", ZINDEX = F, eps = .Machine$double.eps, NA_flag = (-1e+06))
```

Arguments

`A`	The matrix A.
`SMM`	Sparse matrix method with SMM='A',CRI','RCI','CMO', or 'RMO'
`ZINDEX`	T = Use zero indexing or F = Use one indexing.
`eps`	Value to use in non-zero test.
`NAflag`	Number to uses as NA flag

Value

A list object containing the matrix components:

nnz = the number of non-zero elements in ra.

nr = the number of rows in matrix A.

nc = the number of columns in matrix A.

ia = the row index.

ja = the column index.

ra = the non-zero coefficients in A

rnames = matrix row names – may be ""

cnames = matrix column names – may be ""

SMM = the sparse matrix type.

ZINDEX with T = Use zero indexing or F = Use one indexing.

Examples

```
## Not run:
(A = matrix(c(1,0,0,2,0,3,0,0,0,5,0,6),3,4,byrow=T))
(SM1 = A2SM(A,SMM='CMO',ZINDEX=T))
(SM2=SM2SM(SM1,SMM='CRI',ZINDEX=F))
SM2A(SM1)
SM2A(SM2)

(A = matrix(c(1,0,0,2,0,3,0,0,0,5,0,6),3,4,byrow=T)); A[2,3]=NA; A
(ASM = A2SM(A,SMM='CMO',ZINDEX=T)); SM2A(ASM)
(ASM2 = SM2SM(ASM,SMM2='CRI',ZINDEX2=T)); A ; (A2 = SM2A(SM2))

#clpAPI  documentation example
nr=5
nc=8
ra=c(3.0,5.6,1.0,2.0,1.1,1.0,-2.0,2.8,-1.0,1.0,1.0,-1.2,-1.0,1.9)
ia=c(0,4,0,1,1,2,0,3,0,4,2,3,0,4)
ja=c(0,2,4,6,8,10,11,12,14)
SMM='CMO'
ZINDEX=T
ASM=list(nr=nr,nc=nc,ra=ra,ia=ia,ja=ja,SMM=SMM,ZINDEX=ZINDEX)
(A=SM2A(ASM))

## End(Not run)
```

---

cbindSM *cbindSM: "column bind" two sparse matrices*

Description

cbindSM: "column bind" two sparse matrices

Usage

```
cbindSM(SM1, SM2, SMM = "CRI", ZINDEX = F)
```

Arguments

SM1	First sparse matrix object
SM2	Second sparse matrix object
SMM	Sparse matrix method with 'CRI','RCI','CMO',or'RMO'
ZINDEX	T = Use zero indexing or F = Use one indexing.

Value

A 'column bound" sparse matrix object with components"

nnz = the number of non-zero elements in ra.

nr = the number of rows in matrix A.

nc = the number of columns in matrix A.

ia = the row index.

ja = the column index.

ra = the non-zero coefficients in A

rnames = matrix row names – may be ”

cnames = matrix column names – may be ”

SMM = the sparse matrix type.

ZINDEX with T = Use zero indexing or F = Use one indexing.

CST11 *Cooper,Seiford,Tone 2006 One Input One Output Example Data*

Description

Cooper,Seiford,Tone 2006 One Input One Output Example Data

Format

A data frame with 8 rows and 5 variables:

`STORE` A-H

`EMPLOYEES` employees per store

`SALES` sales per store (CST 2006 example)

`SALES_EJOR` sales per store (modified CST data used in Atwood-Shaik(2020)

`SALES_EJOR_APDX` sales per store (modified CST data used in Appendix Atwood-Shaik(2020)

CST12 *Cooper,Seiford,Tone 2006 One Input Two Output Example Data Table 1.4*

Description

Cooper,Seiford,Tone 2006 One Input Two Output Example Data Table 1.4

Format

A data frame with 7 rows and 4 variables:

`STORE` A-G

`EMPLOYEES` employees per store

`CUSTOMERS` sales per store (CST 2006 example)

`SALES` sales per store

CST21 *Cooper,Seiford,Tone 2006 Two Input One Output Example Table 1.3*

Description

Cooper,Seiford,Tone 2006 Two Input One Output Example Table 1.3

Format

A data frame with 9 rows and 4 variables:

`STORE` A-I

`EMPLOYEES` employees per store

`FLOOR_AREA` floor area per store

`SALES` sales per store

CST22 *Cooper,Seiford,Tone 2006 Two Input Two Output Example Data Table 1.5*

Description

Cooper,Seiford,Tone 2006 Two Input Two Output Example Data Table 1.5

Format

A data frame with 12 rows and 5 variables:

`HOSPITAL` A-L

`DOCTORS`

`NURSES`

`OUT_PATIENTS`

`IN_PATIENTS`

DEAbuild *DEAbuild: Builds DDEA LP object for use in qDEA_solve function*

Description

DEAbuild: Builds DDEA LP object for use in qDEA_solve function

Usage

```
DEAbuild(
  X,
  Y,
  X0,
  Y0,
  DX0,
  DY0,
  dmu0 = 1,
  RTS = "CRS",
  unbounded = -1000,
  solver = "clp"
)
```

Arguments

X	Reference dmu's = ndmu x number of inputs input matrix.
Y	Reference dmu's = ndmu x number of outputs output matrix.
X0	Inputs for set of ndmu0 dmu's to be processed.
Y0	Outputs for set of ndmu0 dmu's to be processed.
DX0	Input directions for ndmu0 dmu's in X0 and Y0.
DY0	Output directions for ndmu0 dmu's in X0 and Y0.
dmu0	Row in (X0,Y0,DX0,DY0) to use as given dmu
RTS	Returns to scale: 'CRS,'VRS,'DRS','IRS.
unbounded	DEA obj restricted >= to unbounded. Default = -1E3
solver	LP solver Default='clp' Options:'clp','highs','glpk' or 'rglpk'

Value

Returns an LP list object containing the following elements:

LPsense = 'max' or 'min'

nnz = number of nonzero elements in 'A' matrix

nr = number or rows in 'A' matrix

nc = number of columns in 'A' matrix

obj = nc length vector of objective coefficients

ra = nonzero coefficients in 'A' matrix

ia = row indexes for non zero elements in 'A' matrix

ja = column indexes for non zero elements in 'A' matrix

SMM = Sparse matrix method: 'CMO','RMO','CRI',or 'RCI'

ZINDEX = 'zero indexing' (T) or 'one indexing' (F)

dirs = nr length vector of constaint signs ('<=','>=', or '=')

rhs = nr length vector of RHS coefficients

xlower = nc vector of lower bounds on 'x' choice variables

xupper = nc vector of upper bounds on 'x' choice variables

rlower = nr vector of Ax lower bounds i.e. rlower <= Ax

rupper = nr vector of Ax upper bounds i.e. Ax <= rupper

vartypes = nc vector of variable types: ('C','B','I') – qDEA:rep('C',nc)

yxchngC = index of given dmu's output-input values locations in ra vector (used to edit LP problem as we loop through DMU's in (X0,Y0,DX0,DY0)

dyxchngC = index of given dmu's direction values locations in ra vector (used to edit LP problem as we loop through DMU's in (X0,Y0,DX0,DY0)

yxchngR = index of given dmu's obj restriction output-input values locations in ra vector (used to edit LP problem as we loop through DMU's in (X0,Y0,DX0,DY0)

iyxchngC = row indexes associated with yxchngC cells

idyxchngC = row indexes associated with dyxchngC cells

iyxchngR = row indexes associated with yxchngR cells

DOBJ = matrix of obj values to change by dmu with DOBJ=cbind(-Y0,X0)

DYX = matrix of (dy,dx) values to change by dmu with DYX=cbind(DY0,DX0)

RTS = Returns to scale: 'CRS,'VRS,'DRS','IRS.

dmu0 = Row in (X0,Y0,DX0,DY0) to use as given dmu

`iter_delete` *iter_delete: Function used for 'peeling' that supplements or supplants qDEA 'slicing' procedure. Intended to be called from qDEA function*

Description

iter_delete: Function used for 'peeling' that supplements or supplants qDEA 'slicing' procedure. Intended to be called from qDEA function

Usage

```
iter_delete(
  LP0,
  ndmus,
  nout0,
  iterlim = max(250, nout0),
  BIGM = 1e+06,
  unbounded = 0.001,
  eps = 1e-06,
  solver = "clp"
)
```

Arguments

LP0	qDEA Stage-2 LP object.
ndmus	Number of dmus in reference data (X,Y).
nout0	Number of external points in current LP object's solution.
iterlim	Maximal number of 'peeling' iterations.
BIGM	Default Big M in RHS of qDEA stage 2 process.
unbounded	LP reported as unbounded if obj<unbounded.
eps	iter_delete convergence test parameter
solver	LP solver Default='clp' Options:'clp','highs','glpk' or 'rglpk'

Value

obj2 = best objective level found

nout = number of external points

outlist = list of external dmus

iter = number of iterations completed

LP2 = LP object for best solution found

LP2sol = optimal LP solution

---

lagMat *lagMat: Create a matrix of lags*

Description

lagMat: Create a matrix of lags

Usage

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

Arguments

x	Vector to be 'lagged'
lags	Number of lagged columns in resulting matrix
Lzero	(F= do not include x[i] in row i) (T = include x[i] in row i)

Value

= lagged matrix

lagum *lagum: Lag a vector of values*

Description

lagum: Lag a vector of values

Usage

```
lagum(x, nlag = 1)
```

Arguments

x	A vector of values to be lagged
nlag	Length of lag: negative value indicates an 'uplag'

Value

= a lagged vector

LPSOLVER *LPSOLVER Function to call specified solver to sparse LP problem*

Description

LPSOLVER Function to call specified solver to sparse LP problem

Usage

```
LPSOLVER(LP, solver = "clp")
```

Arguments

LP	An LP object
solver	LP solver Default='clp' Options:'clp','highs','glpk' or 'rglpk'

Value

A list containing following components:
status = status of solver
objval = objective value
solution = lp primal solution
dual = lp dual solution
rcost = lp reduced cost
lhs = lp lhs associated with primal solution
TIME = total seconds to execute LP_* function
SMtime = seconds to transform form of sparse LP (if needed)
proctime = seconds required by specified solver to solve LP
LP = LP object

LP_clp *LP_clp Function to solve sparse LP problem using clp_API*

Description

LP_clp Function to solve sparse LP problem using clp_API

Usage

```
LP_clp(LP)
```

Arguments

LP	An LP object

Value

A list containing following components:

status = status of clpAPI solver

objval = objective value

solution = lp primal solution

dual = lp dual solution

rcost = lp reduced cost

lhs = lp lhs associated with primal solution

TIME = total seconds to execute LP_clp function

SMtime = seconds to transform form of sparse LP (if needed)

proctime = seconds required by clpAPI to solve LP

LP = LP object (transformed to SMM='CMO', ZINDEX=T if needed)

LP_glpk *LP_glpk Function to solve sparse LP problem using glpkAPI package*

Description

LP_glpk Function to solve sparse LP problem using glpkAPI package

Usage

```
LP_glpk(LP)
```

Arguments

LP	An LP object

Value

A list containing following components:
status = status of glpkAPI solver
objval = objective value
solution = lp primal solution
dual = lp dual solution
rcost = lp reduced cost
lhs = lp lhs associated with primal solution
TIME = total seconds to execute LP_glpk function
SMtime = seconds to transform form of sparse LP (if needed)
proctime = seconds required by glpkAPI to solve LP
LP = LP object (transformed to SMM='CRI', ZINDEX=F if needed)

`LP_highs` *LP_highs Function to solve sparse LP problem using highs package*

Description

LP_highs Function to solve sparse LP problem using highs package

Usage

```
LP_highs(LP)
```

Arguments

`LP`	An LP object

Value

A list containing following components:
status = status of highs solver
objval = objective value
solution = lp primal solution
dual = lp dual solution
rcost = lp reduced cost
lhs = lp lhs associated with primal solution
TIME = total seconds to execute LP_highs function
SMtime = seconds to transform form of sparse LP (if needed)
proctime = seconds required by highs to solve LP
LP = LP object (transformed to SMM='CRI', ZINDEX=F if needed)

LP_Rglpk *LP_Rglpk Function to solve sparse LP problem using Rglpk package*

Description

LP_Rglpk Function to solve sparse LP problem using Rglpk package

Usage

```
LP_Rglpk(LP)
```

Arguments

LP	An LP object

Value

A list containing following components:

status = status of Rglpk solver

objval = objective value

solution = lp primal solution

dual = lp dual solution

rcost = lp reduced cost

lhs = lp lhs associated with primal solution

TIME = total seconds to execute LP_Rglpk function

SMtime = seconds to transform form of sparse LP (if needed)

proctime = seconds required by Rglpk to solve LP

LP = LP object (transformed to SMM='CRI', ZINDEX=F if needed)

---

merge_lists *merge_lists: Merges two list objects. This function appends any objects in list2 and not in list 1 to list1 with priority given to list 1 components.*

Description

merge_lists: Merges two list objects. This function appends any objects in list2 and not in list 1 to list1 with priority given to list 1 components.

Usage

```
merge_lists(list1, list2)
```

Arguments

list1	A list object
list2	A list object

Value

list3 A merged object

Examples

```
## Not run:
L1=list(a=1:3,b=4:6,c=7:9,e=10:12)
L2=list(b=13:15,d=16:18,e=19:21,f=22:24)
(L3=merge_lists(L1,L2))
(L4=merge_lists(L2,L1))

## End(Not run)
```

`my_seconds` *my_seconds: Function to pull seconds from proc.time function*

Description

my_seconds: Function to pull seconds from proc.time function

Usage

```
my_seconds()
```

Value

= seconds from proc.time function

`nCm_mpick` *nCM_mpick: Select subsample size m from mlist !!!!Intended to be called from qDEA function!!!!*

Description

Pulls associated set of nboot bootstrapped values and computes bias corected "s(m)" values. ** Uses Simar and Wilson (2011) suggested procedure for selecting m **

Usage

```
nCm_mpick(stat, boot, n, mlist, beta = 0.5, alpha = 0.05, CILag = 1)
```

Arguments

`stat`	DDEA or qDEA distance point estimate
`boot`	nboot by mcells matrix of bootstrapped DDEA and qDEA distance
`n`	Number of dmus in full reference set (X,Y)
`mlist`	Vector of subsample sizes
`beta`	Convergence rate. 0.5 => "root n" convergence
`alpha`	Alpha level for Simar-Wilson (2011) subsample size selection procedure
`CILag`	CILag level for Simar-Wilson (2011) subsample size selection procedure

Value

mpick = selected subsample size,

S = nboot by mcell matrix of s = stat(n) - (m/n)^beta(boot(m)-stat(n) bias corrected values for each sample size m

s = vector (of length nboot) of 'bias.corected' s-values for sample size 'mpick'

stat.bc = bias corrected point estimate of stat = mean(s)

---

qDEA *qDEA: Calling function for set of DEA and qDEA processes*

Description

Note: Optional arguments in function call have default values of 'NULL'

Usage

```
qDEA(
  X,
  Y,
  qout = 1/nrow(X),
  qoutS = qout,
  X0 = NULL,
  Y0 = NULL,
  DX0 = NULL,
  DY0 = NULL,
  orient = "out",
  RTS = "CRS",
  dmulist = NULL,
  nqiter = 1,
  nboot = 0,
  transform = T,
  mcells = 5,
  mlist = NULL,
  seedval = 1001,
  qtol = 1e-06,
  BIGM = 1e+09,
  eps = 1e-06,
  skipzprob = T,
  replaceA1 = F,
  baseqDEA = F,
  unbounded = (-1000),
  obj2test = 1e-04,
  replaceM = F,
  alpha = 0.05,
  betaq = 0.5,
  siglist = c(0.1, 0.05, 0.01),
  CILag = 1,
  printlog = T,
  prntmod = 100,
```

```
  printtxt = "",
  getproject = F,
  getbootpeers = F,
  solver = "clp"
)
```

Arguments

X	Reference dmu's = ndmu x number of inputs input matrix.
Y	Reference dmu's = ndmu x number of outputs output matrix.
qout	Maximal proportion of dmu's allowed external to DEA hull. Default = 1/ndmu
qoutS	Proportion of external points to identify using qDEA slicing (Atwood and Shaik 2020 EJOR)with qoutS <= qout. Default = qout
X0	Inputs for set of ndmu0 dmu's to be processed. Default = X
Y0	Outputs for set of ndmu0 dmu's to be processed. Default = Y
DX0	Input directions for ndmu0 dmu's in X0 and Y0. (Must be provideed if orient = 'ddea'). Default = NULL
DY0	Output directions for ndmu0 dmu's in X0 and Y0. (Must be provideed if orient = 'ddea') Default = NULL
orient	Model orientation ('in','out','inout','oneone','ddea'). (!!If orient='ddea',DX0,and DY0 must be provided) Default='out'
RTS	Returns to scale.('CRS,'VRS,'DRS','IRS) Default 'CRS'.
dmulist	Index vector of dmus in (X0,Y0) to process. NULL = process all.
nqiter	Maximal number of qDEA iterations. Default = 1.
nboot	Number of bootstrap replications. Default = 0.
transform	Transform DDEA distance to traditional efficiency metrics (input or output orientations only) Default = T
mcells	Number of subsample sizes for bootstapping. Default = 5
mlist	Optional list of user chosen subsample sizes. Default = NULL
seedval	Seed value for random number generator - used in bootstapping. Default = 1001.
qtol	q search tolerance with iterative qDEA. Default = 1E-6
BIGM	Default Big M in RHS of qDEA stage 2 process. Default = 1E9
eps	Search tolerance in qDEA improvement tests. Default = 1E-6
skipzprob	Skip qDEA if qout=0. Default = T.
replaceA1	Put dmu0's data in first row of reference sets. Default = F
baseqDEA	Use basic qDEA model from EJOR article. Default = F
unbounded	qDEA reported as unbounded if obj<unbounded. Default = (-1E3)
obj2test	Convergance tol for objective in iterative qDEA.Default = 1E-4
replaceM	Subsample with replacement in subsample bootstrap. Default = F
alpha	Alpha level for Simar-Wilson(2011)subsample size selection procedure. Default = 0.05
betaq	qDEA convergence rate. 0.5 => "root n" convergence.
siglist	Vector of user's desired confidence interval widths Default = c(0.10, 0.05, 0.01)

CILag	CILag level for Simar-Wilson (2011) subsample size selection procedure. Default = 1
printlog	Progress of DDEA dmu's solved when (X0,Y0) >1 dmus.
prntmod	Print progress every prntmod dmus. Default=100.
printtxt	Additional text to print with progress printlog.
getproject	Compute projected values for dmu's in dmulist. Default=F
getbootpeers	Pull and store peers for each bootstrapped solution. Default = F
solver	LP solver Default='clp' Options:'clp','highs','glpk' or 'rglpk'

Value

'##'

A list of individual items and lists of addiitional/optional output

effvals = vector of DDEA distances (efficiencies if transform=T and input or output orientation).

effvalsq = vector of qDDEA distances (efficiencies if transform=T and input or output orientation).

distvals = vector of DDEA distances

distvalsq = vector of qDEA distances

distMAT = ndmu0 by 3 matrix with DDEA,qDDEA-S1,qDDEA-S2 distances. '##################################

INPUT_DATA = A list containing:

X = reference dmu's = ndmu x number of inputs input matrix.

Y = reference dmu's = ndmu x number of outputs output matrix.

X0 = inputs for set of ndmu0 dmu's processed.

Y0 = outputs for set of ndmu0 dmu's processed.

DX0 = input directions for ndmu0 dmu's processed.

DY0 = output directions for ndmu0 dmu's processed.

qout = maximal proportion of dmu's allowed external to DEA hull.

qoutS = proportion of external points to identify using qDEA slicing.

RTS = returns to scale.

orient = model orientation. ('ddea','in','out','inout','oneone')

baseqDEA = use basic qDEA model from EJOR article

dmulist0 = DMU0 index in originally inputs X0,Y0,DX0,and DY0

'##'

BOOT_DATA = A list containing:

effvals.bc = vector of DDEA bias corrected distance or efficiency metrics (efficiencies if transform=T and input or output orientation)

effvalsq.bc = vector of qDEA bias corrected distaince or efficiency metrics (efficiencies if transform=T and input or output orientation)

distvals.bc = vector of bias corrected DDEA distances

distvalsq.bc = vector of bias corrected qDEA distances

mcells = number of subsample sizes for bootstapping

mlist = list of subsample sizes used in bootstraps

BOOTdmus = matrix containing indexes of reference dmus chosen for each bootstrap

BOOT = nboot by mcells by ndmu0 array of bootstrapped DDEA distances (efficiencies if transform=T and input or output orientation)

BOOTS = nboot by mcells by ndmu0 array of DDEA bootstrapped s-statistics (statn = (m(n)/n)^beta (statm - statn)

BOOTs = nboot by ndmu0 matrix of DDEA bootstrapped s-statistics (at sample size chosen by Simar and Wilson (2011) suggested process)

BOOTq = nboot by mcells by ndmu0 array of bootstrapped qDEA distances (efficiencies if transform=T and input or output orientation)

BOOTSq = nboot by mcells by ndmu0 array of qDEA bootstrapped s-statistics (statn = (m(n)/n)^beta (statm - statn)

BOOTsq = nboot by ndmu0 matrix of qDEA bootstrapped s-statistics (at sample size chosen by Simar and Wilson (2011) suggested process

mpick = length(ndmu0) index vector of DDEA bootstrap sample size chosen from mlist using Simar and Wilson (2011) suggested sample size selection process

mpickq = length(ndmu0) index vector of qDDEA bootstrap sample size chosen from mlist using Simar and Wilson (2011) suggested sample size selection process

beta = DDEA convergence rate indicated Simar-Wilson (2011)

betaq = qDEA convergence rate indicated Atwood and Shaik(2020) = 0.5

siglist = vector of user's desired confidence interval widths

CI = ndmu0 by 2 by length(siglist) array of DDEA confidence intervals (estimated using quantiles on bias corrcted "s" statistics)

CIq = ndmu0 by 2 by length(siglist) array of qDEA confidence intervals (estimated using quantiles on bias corrcted "s" statistics)

CIq_norm = ndmu0 by 2 by length(siglist) array of qDEA confidence intervals (estimated using sample mean and standard error of bootstrapped "s" statistics and assuming normality) (Atwood and Shaik 2020)

'###'

PEER_DATA = A list containing:

PEERS = dataframe of DDEA peers and projection weights for each dmu0

PEERSq = dataframe of qDDEA-S2 peers and projection weights each dmu0

DOUTq = data frame indicating external dmus for each dmu's qDEA solution

BPEERS = ndmu0 by mcells by (nIN+nOUT) by 2(dmu-z,zweight) array of DDEA subsampled peers (dmu-z) and projection weights (z) if(nboot>0&getbootpeers==T)

BPEERSq = ndmu0 by mcells by (nIN+nOUT) by 2(dmu-z,zweight) array of qDEA subsampled peers (dmu-z) and projection weights (z) if(nboot>0&getbootpeers==T)

BPEERS1 = data.frame of DDEA subsampled peers and projection weights (z) if(nboot>0&getbootpeers==T) for chosen subsample size data.frame: nb = bootstrap rep, dmu0 = given dmu, dmuz = reference dmu, z = reference dmu projection weight

BPEERS1q = data.frame of qDEA subsampled peers and projection weights (z) if(nboot>0&getbootpeers==T) for chosen subsample size data.frame: nb = bootstrap rep, dmu0 = given dmu, dmuz = reference dmu, z = reference dmu projection weight

'###'

PROJ_DATA = Projected Values. A list containing:

X0HAT = DDEA projected input levels (if getproject=T)

Y0HAT = DDEA projected output levels (if getproject=T)

X0HAT.bc = bias corrected DDEA projected input levels (if nboot>0 and getproject=T) (if nboot>0 and getproject=T))

X0HATq = qDEA projected input levels (if getproject=T)

Y0HATq = qDEA projected output levels (if getproject=T)

X0HATq.bc = bias corrected qDEA projected input levels (if nboot>0 and getproject=T)

Y0HATq.bc = bias corrected qDEA projected output levels (if nboot>0 and getproject=T))

'###'

LP_DATA = LP Models, data, and results. A list containing:

status = ndmu0 by 3 matrix with LP status of DDEA,qDDEA-S1,qDDEA-S2

qhat = proportion of dmu's external to hull in qDEA solution (NA indicates qDEA for given DMU was unbounded)

qiter = number of qDEA iterations completed.

PSOL = ndmu0 by ? matrix with DDEA LP solutions for each dmu0

PSOLq1 = ndmu0 by ? matrix with qDDEA-S1 LP solutions for each dmu0

PSOLq2 = ndmu0 by ? matrix with qDDEA-S2 Lp solutions for each dmu0

RCOST = ndmu0 by ? matrix of LP reduced costs for DDEA solutions

RCOSTq1 = ndmu0 by ? matrix of LP reduced costs for qDDEA-S1 solutions

RCOSTq2 = ndmu0 by ? matrix of LP reduced costs for qDDEA-S2 solutions

LPModels = list of LP0, LP1, and LP2 LP objects from qDEA_solve function

Examples

```
## Not run:
#############################################################################
# Note: The package author recommends using the solver R package clpAPI
# but the package allows the use of solver R packages glpkAPI, highs, or
# Rglpk. The package glpkAPI is used in the following CST(2006) examples.
#############################################################################
# Examples from CST(2006): Cooper, W., Seiford, L., and Tone, K., 2006.
# Introduction to Data Envelopment Analysis and Its Uses.  Springer. NewYork.
#############################################################################
library(qDEA);library(glpkAPI)
#############################################################################
#CST One Input- One Output Example - Table 1.1
data(CST11)
CST11
attach(CST11)
(X = as.matrix(EMPLOYEES))
(Y = as.matrix(SALES_EJOR))
detach(CST11)
(qout=1/nrow(X))
sol=qDEA(X,Y,RTS='crs',orient='in',qout=qout,solver='glpk')
# DEA efficiency scores
sol$effvals
# qDEA input efficiency scores allowing one external point
sol$effvalsq
#############################################################################
#CST Two Input- One Output Example - Table 1.3
```

```
data(CST21)
CST21
attach(CST21)
(X = as.matrix(cbind(EMPLOYEES,FLOOR_AREA)))
(Y = as.matrix(SALES))
detach(CST21)
(qout=1/nrow(X))
sol=qDEA(X,Y,RTS='crs',orient='in',qout=qout,solver='glpk')
# DEA efficiency scores
sol$effvals
# qDEA input efficiency scores allowing one external point
sol$effvalsq
##############################################################################
#CST One Input- Two Output Example - Table 1.4
data(CST12)
CST12
attach(CST12)
(X = as.matrix(EMPLOYEES))
(Y = as.matrix(cbind(CUSTOMERS,SALES)))
detach(CST12)
(qout=1/nrow(X))
sol=qDEA(X,Y,RTS='crs',orient='out',qout=qout,solver='glpk')
# DEA efficiency scores
sol$effvals
# qDEA output efficiency scores allowing one external point
sol$effvalsq
##############################################################################
#CST Two Input- Two Output Example - Table 1.5
data(CST22)
CST22
attach(CST22)
(X = as.matrix(cbind(DOCTORS,NURSES)))
(Y = as.matrix(cbind(OUT_PATIENTS,IN_PATIENTS)))
detach(CST22)
(qout=1/nrow(X))
sol=qDEA(X,Y,RTS='crs',orient='in',qout=qout,solver='glpk')
# DEA efficiency scores - see table 1.6 CCR estimates
round(sol$effvals,2)
# qDEA efficiency scores allowing one external point
round(sol$effvalsq,2)
##############################################################################



##############################################################################
#Atwood-Shaik EJOR qDEA Examples using clpAPI.
##############################################################################
library(qDEA); library(clpAPI)
data(CST11)
CST11
(X = as.matrix(CST11$EMPLOYEES))
(Y = as.matrix(CST11$SALES_EJOR))
##############################################################################
# EJOR Efficiency Results Table 1 Input Orientation
tmpC1=qDEA(X,Y,RTS='crs',orient='in',qout=1/8,getproject = T)
tmpC2=qDEA(X,Y,RTS='crs',orient='in',qout=2/8,getproject = T)
```

```
# Table 1 Input Orientation
round(cbind(tmpC1$distvals,tmpC1$PROJ_DATA$X0HAT,tmpC1$PROJ_DATA$Y0HAT,
      tmpC1$distvalsq,tmpC1$PROJ_DATA$X0HATq,tmpC1$PROJ_DATA$Y0HATq,
      tmpC2$distvalsq,tmpC2$PROJ_DATA$X0HATq,tmpC2$PROJ_DATA$Y0HATq),3)
##############################################################################
# EJOR Efficiency Results Table 1 Output Orientation
tmpC3=qDEA(X,Y,RTS='crs',orient='out',qout=1/8,getproject = T)
tmpC4=qDEA(X,Y,RTS='crs',orient='out',qout=2/8,getproject = T)

# Table 1 Output Orientation
round(cbind(tmpC3$distvals,tmpC3$PROJ_DATA$X0HAT,tmpC3$PROJ_DATA$Y0HAT,
      tmpC3$distvalsq,tmpC3$PROJ_DATA$X0HATq,tmpC3$PROJ_DATA$Y0HATq,
      tmpC4$distvalsq,tmpC4$PROJ_DATA$X0HATq,tmpC4$PROJ_DATA$Y0HATq),3)
##############################################################################
# EJOR Efficiency Results Table 1 one-one Orientation
tmpC5=qDEA(X,Y,RTS='crs',orient='oneone',qout=1/8,getproject = T)
tmpC6=qDEA(X,Y,RTS='crs',orient='oneone',qout=2/8,getproject = T)

# Table 1 one-one Orientation
round(cbind(tmpC5$distvals,tmpC5$PROJ_DATA$X0HAT,tmpC5$PROJ_DATA$Y0HAT,
     tmpC5$distvalsq,tmpC5$PROJ_DATA$X0HATq,tmpC5$PROJ_DATA$Y0HATq,
     tmpC6$distvalsq,tmpC6$PROJ_DATA$X0HATq,tmpC6$PROJ_DATA$Y0HATq),3)
##############################################################################

## End(Not run)
```

---

qDEAbuild *qDEAbuild: Builds qDEA LP object for use in qDEA_solve function*

Description

qDEAbuild: Builds qDEA LP object for use in qDEA_solve function

Usage

```
qDEAbuild(
  X,
  Y,
  X0,
  Y0,
  DX0,
  DY0,
  qout = 0.1,
  dmu0 = 1,
  RTS = "CRS",
  unbounded = -1000,
  solver = "clp"
)
```

Arguments

X	Reference dmu's = ndmu x number of inputs input matrix.
Y	Reference dmu's = ndmu x number of outputs output matrix.

X0	Inputs for set of ndmu0 dmu's to be processed.
Y0	Outputs for set of ndmu0 dmu's to be processed.
DX0	Input directions for ndmu0 dmu's in X0 and Y0.
DY0	Output directions for ndmu0 dmu's in X0 and Y0.
qout	Maximal proportion of dmu's allowed external to DEA hull.
dmu0	Row in (X0,Y0,DX0,DY0) to use as given dmu
RTS	Returns to scale: 'CRS,'VRS,'DRS','IRS.
unbounded	DEA obj restricted >= to unbounded. Default = -1E3
solver	LP solver Default='clp' Options:'clp','highs','glpk' or 'rglpk'

Value

Returns an LP list object containing the following elements:

LPsense = 'max' or 'min'

nnz = number of nonzero elements in 'A' matrix

nr = number or rows in 'A' matrix

nc = number of columns in 'A' matrix

obj = nc length vector of objective coefficients

ra = nonzero coefficients in 'A' matrix

ia = row indexes for non zero elements in 'A' matrix

ja = column indexes for non zero elements in 'A' matrix

SMM = Sparse matrix method: 'CMO','RMO','CRI',or 'RCI'

ZINDEX = 'zero indexing' (T) or 'one indexing' (F)

dirs = nr length vector of constaint signs ('<=','>=', or '=')

rhs = nr length vector of RHS coefficients

xlower = nc vector of lower bounds on 'x' choice variables

xupper = nc vector of upper bounds on 'x' choice variables

rlower = nr vector of Ax lower bounds i.e. rlower <= Ax

rupper = nr vector of Ax upper bounds i.e. Ax <= rupper

vartypes = nc vector of variable types: ('C','B','I') – qDEA:rep('C',nc)

yxchngC = index of given dmu's output-input values locations in ra vector (used to edit LP problem as we loop through DMU's in (X0,Y0,DX0,DY0)

dyxchngC = index of given dmu's direction values locations in ra vector (used to edit LP problem as we loop through DMU's in (X0,Y0,DX0,DY0)

yxchngR = index of given dmu's output-input values locations in ra vector (used to edit LP problem as we loop through DMU's in (X0,Y0,DX0,DY0)

iyxchngC = row indexes associated with yxchngC cells

idyxchngC = row indexes associated with dyxchngC cells

iyxchngR = row indexes associated with yxchngR cells

DOBJ = matrix of obj values to change by dmu with DOBJ=cbind(-Y0,X0)

DYX = matrix of (dy,dx) values to change by dmu with DYX=cbind(DY0,DX0)

qchng = index of 1/q value location in ra vector (used to edit LP problem as we iterate qDEA

RTS = Returns to scale: 'CRS,'VRS,'DRS','IRS.

dmu0 = Row in (X0,Y0,DX0,DY0) to use as given dmu.

devstart = LP index for column of first qDDEA-S1 LPM "deviation" value

devend = LP index for column of last qDDEA-S1 LPM "deviation" value

tau0 = parameter used in iterative qDEA process

`qDEA_mlist` *qDEA_mlist: Obtain subsample qDEA results for a set (vector) of subsample sizes !!!Intended to be called from qDEA function!!!!*

Description

mcells = number of subsample sizes

Usage

```
qDEA_mlist(
  XM,
  YM,
  mpick,
  qout = 0.1,
  qoutS = qout,
  Bdrop = NULL,
  Bdropq = NULL,
  X0 = XM,
  Y0 = YM,
  DX0 = XM,
  DY0 = YM,
  mlist,
  RTS = "CRS",
  nqiter = 1,
  qtol = 1e-06,
  BIGM = 1e+09,
  eps = 1e-06,
  skipzprob = T,
  unbounded = (-1000),
  obj2test = 1e-04,
  replaceA1 = replaceA1,
  baseqDEA = baseqDEA,
  printlog = F,
  prntmod = 100,
  getbootpeers = F,
  dmulist0 = 1:nrow(X0),
  solver = "clp"
)
```

Arguments

XM	Input levels for subsample of reference set X.
YM	Output levels for subsample of reference set Y.
mpick	Index of subsample dmus in original (X,Y) reference set.
qout	Maximal proportion of dmu's allowed external to DEA hull.
qoutS	Proportion of external points to identify using qDEA slicing.
Bdrop	Index list of DMU's whose initial DDEA solutions were unbounded.
Bdropq	Index list of DMU's whose initial qDEA solutions were unbounded.
X0	Inputs for set of ndmu0 dmu's to be processed.
Y0	Outputs for set of ndmu0 dmu's to be processed.
DX0	Input directions for ndmu0 dmu's in X0 and Y0.
DY0	Output directions for ndmu0 dmu's in X0 and Y0.
mlist	Vector of subsample sizes
RTS	Returns to scale default='CRS' options are 'CRS,'VRS,'DRS','IRS.
nqiter	Maximal number of qDEA iterations.
qtol	qout search tolerance with iterative qDEA.
BIGM	Default Big M in RHS of qDEA stage 2 process.
eps	Search tolerance in qDEA improvement tests.
skipzprob	Skip qDEA if qout=0.
unbounded	qDEA reported as unbounded if obj<unbounded.
obj2test	Converge tol for objective in iterative qDEA.
replaceA1	Put dmu0's data in first row of reference sets. .
baseqDEA	Use basic qDEA model from EJOR article
printlog	Progress of dmu's solved when (X0,Y0) >1 dmus.
prntmod	Print progress every prntmod dmus.
getbootpeers	Return dmu projection weights for each dmu and each bootstrap.
dmulist0	DMU0 index in originally inputs X0,Y0,DX0,and DY0
solver	LP solver Default='clp' Options:'clp','highs','glpk' or 'rglpk'

Value

A list containing the following components:

EFFmlist = ndmu0 by mcell matrix of subsample DDEA distances.

EFFmlistq = ndmu0 by mcell matrix of subsample qDEA distances.

XM = Input levels for subsample of reference set X.

YM = Output levels for subsample of reference set X.

qout = Maximal proportion of dmu's allowed external to DEA hull.

qoutS = Proportion of external points to identify using qDEA slicing.

X0 = Inputs for set of ndmu0 dmu's to be processed.

Y0 = Outputs for set of ndmu0 dmu's to be processed.

DX0 = Input directions for ndmu0 dmu's in X0 and Y0.

DY0 = Output directions for ndmu0 dmu's in X0 and Y0.

mlist = Vector of subsample sizes

mcells = Number of subsample sizes

peers = A data frame containing DDEA peer dmus and projection weights

peersq = A data frame containing qDEA peer dmus and projection weights

---

`qDEA_solve` *qDEA_solve: Sets up LP objects and solves DDEA dual DEA and qDEA using clp_API !!!! Intended to be called from qDEA function.!!!! Note: ndmu=number of dmus in reference set, ndmu0 = number dmus to process.*

Description

qDEA_solve: Sets up LP objects and solves DDEA dual DEA and qDEA using clp_API !!!! Intended to be called from qDEA function.!!!! Note: ndmu=number of dmus in reference set, ndmu0 = number dmus to process.

Usage

```
qDEA_solve(
  X,
  Y,
  qout = 0.1,
  qoutS = qout,
  X0,
  Y0,
  DX0,
  DY0,
  RTS = "CRS",
  nqiter = 1,
  qtol = 1e-06,
  BIGM = 1e+09,
  eps = 1e-06,
  skipzprob = T,
  unbounded = (-1000),
  obj2test = 1e-04,
  replaceA1 = F,
  baseqDEA = F,
  printlog = T,
  prntmod = 100,
  printtxt = "",
  dmulist0 = 1:nrow(X0),
  solver = "clp"
)
```

Arguments

X	Reference dmu's = ndmu x number of inputs input matrix.
Y	Reference dmu's = ndmu x number of outputs output matrix.
qout	Maximal proportion of dmu's allowed external to DEA hull.
qoutS	Proportion of external points to identify using EJOR qDEA slicing.
X0	Inputs for set of ndmu0 dmu's to be processed.
Y0	Outputs for set of ndmu0 dmu's to be processed.
DX0	Input directions for ndmu0 dmu's in X0 and Y0.
DY0	Output directions for ndmu0 dmu's in X0 and Y0.
RTS	Returns to scale default='CRS' options are 'CRS,'VRS,'DRS','IRS.
nqiter	Maximal number of qDEA iterations.
qtol	q search tolerance with iterative qDEA.
BIGM	Default Big M in RHS of qDEA stage 2 process.
eps	Zero-test criterion. abs(x) > eps implies x!=0
skipzprob	Skip qDEA if qout=0.
unbounded	qDEA reported as unbounded if obj<unbounded.
obj2test	Convergence tol for objective in iterative qDEA.
replaceA1	Put dmu0's data in first row of reference sets.
baseqDEA	Use basic qDEA model from EJOR article
printlog	Progress of dmu's solved when (X0,Y0) >1 dmus.
prntmod	Print progress every prntmod dmus. default=100
printtxt	Additional text to print with progress printlog.
dmulist0	DMU0 index in originally inputs X0,Y0,DX0,and DY0
solver	LP solver Default='clp' Options:'clp','highs','glpk' or 'rglpk'

Value

A list containing the following components:

effvals = ndmu0 by 3 matrix with DDEA,qDDEA-S1,qDDEA-S2 distances (NA indicates qDEA for given DMU was unbounded)

qhat = proportion of dmu's external to hull in qDEA solution (NA indicates qDEA for given DMU was unbounded)

qiter = number of qDEA iterations completed.

D2OUT = data frame indicating qDEA external reference dmus

status = ndmu0 by 3 matrix with LP status of DDEA,qDDEA-S1,qDDEA-S2

PSOL = ndmu0 by ? matrix with DDEA (dual-side) solutions for each dmu0

PSOLq1 = ndmu0 by ? matrix with qDDEA-S1 solutions for each dmu0

PSOLq2 = ndmu0 by ? matrix with qDDEA-S2 solutions for each dmu0

PEERS = dataframe of non-zero DDEA dmuz's and projection weights for each dmu0

PEERSq = dataframe of non-zero qDDEA-S2 dmuz's projection weights for each dmu0

RCOST = ndmu0 by ? matrix of LP reduced costs for DDEA solutions

RCOSTq1 = ndmu0 by ? matrix of LP reduced costs for qDDEA-S1 solutions

RCOSTq2 = ndmu0 by ? matrix of LP reduced costs for qDDEA-S2 solutions

devstart = LP index for column of first qDDEA-S1 LPM "deviation" value

devend = LP index for column of last qDDEA-S1 LPM "deviation" value

LP0 = DDEA LP object for last dmu0 processed

LP1 = qDDEA-S1 LP object for last dmu0 processed

LP2 = qDDEA-S2 LP object for last dmu0 processed

`rbindSM` *rbindSM: "row bind" two sparse matrices*

Description

rbindSM: "row bind" two sparse matrices

Usage

```
rbindSM(SM1, SM2, SMM = "CRI", ZINDEX = F)
```

Arguments

`SM1`	First sparse matrix object
`SM2`	Second sparse matrix object
`SMM`	Sparse matrix method with 'CRI','RCI','CMO',or'RMO'
`ZINDEX`	T = Use zero indexing or F = Use one indexing.

Value

A 'row bound" sparse matrix object with components"

nnz = the number of non-zero elements in ra.

nr = the number of rows in matrix A.

nc = the number of columns in matrix A.

ia = the row index.

ja = the column index.

ra = the non-zero coefficients in A

rnames = matrix row names – may be ”

cnames = matrix column names – may be ”

SMM = the sparse matrix type.

ZINDEX with T = Use zero indexing or F = Use one indexing.

SM2A *SM2A: Convert a sparse matrix object ASM into a matrix A*

Description

ASM is a sparse matrix object containing: nr = number of rows, nc = number of columns, ra = nonzero coeff, ia = row indices, ja = col indices, rnames = row names,cnames = column names # Note these may be missing or "" SMM = space matrix method, and ZINDEX = use 0-indexing

Usage

```
SM2A(ASM, NA_flag = (-1e+06))
```

Arguments

ASM	A sparse matrix object
NAflag	Number to uses as NA flag

Value

The matrix A.

Examples

```
## Not run:
(A = matrix(c(1,0,0,2,0,3,0,0,0,5,0,6),3,4,byrow=T))
(SM1 = A2SM(A,SMM='CMO',ZINDEX=T))
(SM2=SM2SM(SM1,SMM='CRI',ZINDEX=F))
SM2A(SM1)
SM2A(SM2)

(A = matrix(c(1,0,0,2,0,3,0,0,0,5,0,6),3,4,byrow=T)); A[2,3]=NA; A
(ASM = A2SM(A,SMM='CMO',ZINDEX=T)); SM2A(ASM)
(ASM2 = SM2SM(ASM,SMM2='CRI',ZINDEX2=T)); A ; (A2 = SM2A(SM2))

#clpAPI  documentation example
nr=5
nc=8
ra=c(3.0,5.6,1.0,2.0,1.1,1.0,-2.0,2.8,-1.0,1.0,1.0,-1.2,-1.0,1.9)
ia=c(0,4,0,1,1,2,0,3,0,4,2,3,0,4)
ja=c(0,2,4,6,8,10,11,12,14)
SMM='CMO'
ZINDEX=T
ASM=list(nr=nr,nc=nc,ra=ra,ia=ia,ja=ja,SMM=SMM,ZINDEX=ZINDEX)
(A=SM2A(ASM))

## End(Not run)
```

SM2SM *SM2SM: Convert a sparse matrix form into a different sparse matrix form.*

Description

ASM is a sparse matrix object containing: nr = number of rows, nc = number of columns, ra = nonzero coeff, ia = row indices, ja = col indices, SMM = space matrix method, ZINDEX=use 0-indexing

Usage

```
SM2SM(ASM, SMM2 = "CRI", ZINDEX2 = F, NA_flag = (-1e+06))
```

Arguments

ASM	A sparse matrix object
SMM2	sparse matrix method with SMM2 ='A','CRI','RCI','CMO',or'RMO'
ZINDEX2	T = Use zero indexing or F = Use one indexing.
NAflag	Number to uses as NA flag

Value

A list object containing the following sparse matrix components:

nnz = the number of non-zero elements in ra.

nr = the number of rows in matrix.

nc = the number of columns in matrix.

ia = the revised row index.

ja = the revised column index.

rnames = row names – may be missing or ""

cnames = column names – may be missing or ""

ra = the revised non-zero coefficients in A

SMM = the revised sparse matrix type.

ZINDEX = T or F for the revised sparse form.

Examples

```
## Not run:
(A = matrix(c(1,0,0,2,0,3,0,0,0,5,0,6),3,4,byrow=T))
(SM1 = A2SM(A,SMM='CMO',ZINDEX=T))
(SM2=SM2SM(SM1,SMM='CRI',ZINDEX=F))
SM2A(SM1)
SM2A(SM2)

(A = matrix(c(1,0,0,2,0,3,0,0,0,5,0,6),3,4,byrow=T)); A[2,3]=NA; A
(ASM = A2SM(A,SMM='CMO',ZINDEX=T)); SM2A(ASM)
(ASM2 = SM2SM(ASM,SMM2='CRI',ZINDEX2=T)); A ; (A2 = SM2A(SM2))
```

```
#clpAPI  documentation example
nr=5
nc=8
ra=c(3.0,5.6,1.0,2.0,1.1,1.0,-2.0,2.8,-1.0,1.0,1.0,-1.2,-1.0,1.9)
ia=c(0,4,0,1,1,2,0,3,0,4,2,3,0,4)
ja=c(0,2,4,6,8,10,11,12,14)
SMM='CMO'
ZINDEX=T
ASM=list(nr=nr,nc=nc,ra=ra,ia=ia,ja=ja,SMM=SMM,ZINDEX=ZINDEX)
(A=SM2A(ASM))

## End(Not run)
```

`sort_list` *sort_list: Sorts objects in list by object name*

Description

sort_list: Sorts objects in list by object name

Usage

```
sort_list(list1)
```

Arguments

`list1`	A list object

Value

list2 A resorted list object

`Write.LPC` *Write.LPC: Writes LP to .csv file format*

Description

Write.LPC: Writes LP to .csv file format

Usage

```
Write.LPC(LP, fname = "LPC.csv")
```

Arguments

`LP`	A sparse LP object
`fname`	.csv file name (include path if desired)

Value

= .csv file (can be opened/solved with spreadsheet solver)

Index