

# Package: qDEA (via r-universe)

May 14, 2026

**Type** Package

**Title** Quantile Data Envelopment Analysis

**Version** 1.0.0

**Depends** R (>= 4.2)

**Imports** dplyr, doBy, highs, Matrix, utils

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**Description** R implementation of Quantile Data Envelopment Analysis.

The package 'qDEA' allows a user specified proportion of observations to lie external to a given Decision Making Units's (DMU's) reference hyperplane. 'qDEA' can be used to detect and address influential outliers or to implement quantile benchmarking, as discussed in Atwood and Shaik (2020). Quantile benchmarking is accomplished by using heuristic procedures to find a DMU's closest input-output projection point in a specified direction while allowing a specified proportion of observations to lie external to the projected point's hyperplane. The 'qDEA' package accommodates standard (DEA) and quantile DEA estimation, returns to scale CRS(constant), VRS(variable), DRS(decreasing) or IRS(increasing), the use of directional vectors, bias correction through subsample bootstrapping and subsample size selection procedures. The user can also recover each DMU's reference DMUs and external DMUs if desired. The implemented procedures are based on discussions in: Atwood and Shaik (2020) [doi:10.1016/j.ejor.2020.03.054](https://doi.org/10.1016/j.ejor.2020.03.054) Atwood and Shaik (2018) [doi:10.1007/978-3-319-68678-3\\_4](https://doi.org/10.1007/978-3-319-68678-3_4) Walden and Atwood (2023) [doi:10.1086/724932](https://doi.org/10.1086/724932) Walden and Atwood (2025) [doi:10.1086/736554](https://doi.org/10.1086/736554).

**License** GPL-2 | GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

**NeedsCompilation** no

**Author** Joe Atwood [aut, cre], John Walden [aut]

**Maintainer** Joe Atwood <jatwood@montana.edu>

**Config/pak/sysreqs** cmake libicu-dev pkg-config

**Repository** <https://uaeja.r-universe.dev>

**Date/Publication** 2026-04-13 15:51:02 UTC

**RemoteUrl** <https://github.com/cran/qDEA>

**RemoteRef** HEAD

**RemoteSha** 5622d94de241d622e28c6caa9e77d5f6d55a034f

## Contents

A2SM . . . . .	3
cbindSM . . . . .	4
CST11 . . . . .	5
CST12 . . . . .	5
CST21 . . . . .	6
CST22 . . . . .	6
DEAbuild . . . . .	7
iter_delete . . . . .	8
lagMat . . . . .	9
LP_highs . . . . .	10
LPSOLVER . . . . .	11
merge_lists . . . . .	11
my_lag . . . . .	12
my_seconds . . . . .	13
nCm_mpick . . . . .	13
qDEA . . . . .	14
qDEA_mlist . . . . .	20
qDEA_solve . . . . .	22
qDEAbuild . . . . .	24
rbindSM . . . . .	26
SM2A . . . . .	27
SM2SM . . . . .	28
sort_list . . . . .	29
Write.LP . . . . .	29

**Index**

**31**

**Description**

A2SM: Convert a matrix A to sparse matrix object ASM

**Usage**

```
A2SM(
  A,
  SMM = "CRI",
  ZINDEX = FALSE,
  eps = .Machine$double.eps,
  NA_flag = (-1e+12)
)
```

**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.
NA_flag	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**

```
(A = matrix(c(1,0,0,2,0,3,1,0,0,5,0,6),3,4,byrow=TRUE))
(ASM1 = A2SM(A,SMM='CMO',ZINDEX=TRUE))
(ASM2=SM2SM(ASM1,SMM2='CRI',ZINDEX2=FALSE))
SM2A(ASM1)
SM2A(ASM2)

(A = matrix(c(1,0,0,2,0,3,1,0,0,5,0,6),3,4,byrow=TRUE)); A[2,3]=NA; A
(ASM1 = A2SM(A,SMM='CMO',ZINDEX=TRUE)); A; SM2A(ASM1)
(ASM2 = SM2SM(ASM1,SMM2='CRI',ZINDEX2=TRUE)); A ; SM2A(ASM2)

#CMO 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=TRUE
ASM=list(nr=nr,nc=nc,ra=ra,ia=ia,ja=ja,SMM=SMM,ZINDEX=ZINDEX)
(A=SM2A(ASM))
```

cbindSM

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

cbindSM: "column bind" two sparse matrices

**Usage**

cbindSM(SM1, SM2, SMM = "CRI", ZINDEX = FALSE)

**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	<i>Cooper,Seiford,Tone 2006 One Input One Output Example Data</i>
-------	---

---

### **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	<i>Cooper,Seiford,Tone 2006 One Input Two Output Example Data Table 1.4</i>
-------	---

---

### **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 = "highs"
)
```

**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='highs'

**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 constraint 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)  
 yxchgC = 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))  
 dyxchgC = 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))  
 yxchgR = 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))  
 iyxchgC = row indexes associated with yxchgC cells  
 idyxchgC = row indexes associated with dyxchgC cells  
 iyxchgR = row indexes associated with yxchgR 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 = "highs"
)

```

**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='highs'

**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 = FALSE)
```

**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

---

 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=FALSE if needed)

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 = "highs")

**Arguments**

LP	An LP object
solver	LP solver Default='highs' Note: Versions for 'clp','highs','glpk' or 'rglpk' are available from author

**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

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**

```
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))
```

---

my_lag	<i>my_lag: Lag a vector of values</i>
--------	---------------------------------------

---

**Description**

```
my_lag: Lag a vector of values
```

**Usage**

```
my_lag(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
```

---

my_seconds	<i>my_seconds: Function to pull seconds from proc.time function</i>
------------	---

---

**Description**

my\_seconds: Function to pull seconds from proc.time function

**Usage**

my\_seconds()

**Value**

= seconds from proc.time function

---

nCm_mpick	<i>nCM_mpick: Select subsample size m from mlist !!!!Intended to be called from qDEA function!!!!</i>
-----------	---

---

**Description**

Pulls associated set of nboot bootstrapped values and computes bias corrected "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  $s = \text{stat}(n) - (m/n)^{\text{beta}}(\text{boot}(m) - \text{stat}(n))$  bias corrected values for each sample size m

s = vector (of length nboot) of 'bias.corrected' 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 = TRUE,  
  mcells = 5,  
  mlist = NULL,  
  seedval = 1001,  
  qtol = 1e-06,  
  BIGM = 1e+09,  
  eps = 1e-06,  
  skipzprob = TRUE,  
  replaceA1 = FALSE,  
  baseqDEA = FALSE,  
  unbounded = (-1000),  
  obj2test = 1e-04,  
  replaceM = FALSE,  
  alpha = 0.05,  
  betaq = 0.5,  
  siglist = c(0.1, 0.05, 0.01),  
  CILag = 1,  
  printlog = TRUE,  
  prntmod = 100,  
  printtxt = "",  
  getproject = FALSE,  
  getbootpeers = FALSE,  
  solver = "highs"  
)
```

**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 provided if orient = 'ddea'). Default = NULL
DY0	Output directions for ndmu0 dmu's in X0 and Y0. (Must be provided 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=TRUE
mcells	Number of subsample sizes for bootstrapping. Default = 5
mlist	Optional list of user chosen subsample sizes. Default = NULL
seedval	Seed value for random number generator - used in bootstrapping. 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=TRUE.
replaceA1	Put dmu0's data in first row of reference sets. Default=FALSE
baseqDEA	Use basic qDEA model from EJOR article. Default=FALSE
unbounded	qDEA reported as unbounded if obj<=unbounded. Default = (-1E3)
obj2test	Convergence tol for objective in iterative qDEA.Default = 1E-4
replaceM	Subsample with replacement in subsample bootstrap. Default=FALSE
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 dmus in dmulist. Default=FALSE  
 getbootpeers     Pull and store peers for each bootstrapped solution. Default=FALSE  
 solver            LP solver Default='highs'

### Value

'#####'

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

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

effvalsq = vector of qDDEA distances (efficiencies if transform=TRUE 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 dmus = ndmu x number of inputs input matrix.

Y = reference dmus = ndmu x number of outputs output matrix.

X0 = inputs for set of ndmu0 dmus processed.

Y0 = outputs for set of ndmu0 dmus processed.

DX0 = input directions for ndmu0 dmus processed.

DY0 = output directions for ndmu0 dmus processed.

qout = maximal proportion of dmus 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=TRUE and input or output orientation)

effvalsq.bc = vector of qDEA bias corrected distance or efficiency metrics (efficiencies if transform=TRUE 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 bootstrapping

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=TRUE 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=TRUE 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 corrected "s" statistics)  
 CIq = ndmu0 by 2 by length(siglist) array of qDEA confidence intervals (estimated using quantiles on bias corrected "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==TRUE)  
 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==TRUE)  
 BPEERS1 = data.frame of DDEA subsampled peers and projection weights (z) if(nboot>0&getbootpeers==TRUE)  
 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==TRUE)  
 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=TRUE)

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

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

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

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

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

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

```
'#####'
```

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

```
# Examples from CST(2006): Cooper, W., Seiford, L., and Tone, K., 2006.  

# Introduction to Data Envelopment Analysis and Its Uses. Springer, New York.
```

```
# CST One Input - One Output Example - Table 1.1
```

```
data(CST11)
```

```
X = as.matrix(CST11$EMPLOYEES)
```

```
Y = as.matrix(CST11$SALES_EJOR)
```

```
qout = 1/nrow(X)
```

```
sol = qDEA(X, Y, RTS = 'crs', orient = 'in', qout = qout)
```

```
# 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)
X = as.matrix(cbind(CST21$EMPLOYEES, CST21$FLOOR_AREA))
Y = as.matrix(CST21$SALES)
qout = 1/nrow(X)
sol = qDEA(X, Y, RTS = 'crs', orient = 'in', qout = qout)
sol$effvals
sol$effvalsq

# CST One Input - Two Output Example - Table 1.4
data(CST12)
X = as.matrix(CST12$EMPLOYEES)
Y = as.matrix(cbind(CST12$CUSTOMERS, CST12$SALES))
qout = 1/nrow(X)
sol = qDEA(X, Y, RTS = 'crs', orient = 'out', qout = qout)
sol$effvals
sol$effvalsq

# CST Two Input - Two Output Example - Table 1.5
data(CST22)
X = as.matrix(cbind(CST22$DOCTORS, CST22$NURSES))
Y = as.matrix(cbind(CST22$OUT_PATIENTS, CST22$IN_PATIENTS))
qout = 1/nrow(X)
sol = qDEA(X, Y, RTS = 'crs', orient = 'in', qout = qout)
# 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 (longer running examples)
data(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=TRUE)
tmpC2=qDEA(X,Y,RTS='crs',orient='in',qout=2/8,getproject=TRUE)

# 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=TRUE)
tmpC4=qDEA(X,Y,RTS='crs',orient='out',qout=2/8,getproject=TRUE)

# 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=TRUE)
tmpC6=qDEA(X,Y,RTS='crs',orient='oneone',qout=2/8,getproject=TRUE)

# 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)
#####

```

---

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 = TRUE,
  unbounded = (-1000),
  obj2test = 1e-04,
  replaceA1 = replaceA1,
  baseqDEA = baseqDEA,
  printlog = FALSE,
  prntmod = 100,
  getbootpeers = FALSE,
  dmulist0 = 1:nrow(X0),

```

```

    solver = "highs"
)

```

### 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 dmus 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 dmus to be processed.
Y0	Outputs for set of ndmu0 dmus to be processed.
DX0	Input directions for ndmu0 dmus in X0 and Y0.
DY0	Output directions for ndmu0 dmus 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 dmus data in first row of reference sets. .
baseqDEA	Use basic qDEA model from EJOR article
printlog	Progress of dmus solved when (X0,Y0) >1 dmus.
prntmod	Print progress every prntmod dmus.
getbootpeers	Return dmus projection weights for each dmus and each bootstrap.
dmulist0	DMU0 index in originally inputs X0,Y0,DX0,and DY0
solver	LP solver Default='highs'

### 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 dmus allowed external to DEA hull.  
 qoutS = Proportion of external points to identify using qDEA slicing.  
 X0 = Inputs for set of ndmu0 dmus to be processed.  
 Y0 = Outputs for set of ndmu0 dmus to be processed.  
 DX0 = Input directions for ndmu0 dmus in X0 and Y0.  
 DY0 = Output directions for ndmu0 dmus 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 highs !!!! 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 highs !!!! 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 = TRUE,
  unbounded = (-1000),
  obj2test = 1e-04,
  replaceA1 = FALSE,
  baseqDEA = FALSE,

```

```

    printlog = TRUE,
    prntmod = 100,
    printtxt = "",
    dmulist0 = 1:nrow(X0),
    solver = "highs"
)

```

### 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. $\text{abs}(x) > \text{eps}$ implies $x \neq 0$
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='highs'

### 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

---

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 = "highs"
)

```

**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='highs'

**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)

iychngC = row indexes associated with yxchgC cells  
 idyxchgC = row indexes associated with dyxchgC cells  
 iychngR = row indexes associated with yxchgR 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

---

 rbindSM

*rbindSM: "row bind" two sparse matrices*


---

### Description

rbindSM: "row bind" two sparse matrices

### Usage

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

### 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+12))
```

**Arguments**

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

**Value**

The matrix A.

**Examples**

```
(A = matrix(c(1,0,0,2,0,3,1,0,0,5,0,6),3,4,byrow=TRUE))
(ASM1 = A2SM(A,SMM='CMO',ZINDEX=TRUE))
(ASM2=SM2SM(ASM1,SMM2='CRI',ZINDEX2=FALSE))
SM2A(ASM1)
SM2A(ASM2)

(A = matrix(c(1,0,0,2,0,3,1,0,0,5,0,6),3,4,byrow=TRUE)); A[2,3]=NA; A
(ASM1 = A2SM(A,SMM='CMO',ZINDEX=TRUE)); A; SM2A(ASM1)
(ASM2 = SM2SM(ASM1,SMM2='CRI',ZINDEX2=TRUE)); A ; SM2A(ASM2)

#CMO 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=TRUE
ASM=list(nr=nr,nc=nc,ra=ra,ia=ia,ja=ja,SMM=SMM,ZINDEX=ZINDEX)
(A=SM2A(ASM))
```

---

SM2SM	<i>SM2SM: Convert a sparse matrix object ASM into a different sparse matrix form.</i>
-------	---

---

### 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 = sparse matrix method, ZINDEX=use 0-indexing

### Usage

```
SM2SM(ASM, SMM2 = "CRI", ZINDEX2 = FALSE, NA_flag = (-1e+12))
```

### 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.
NA_flag	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 = TRUE or F for the revised sparse form.

### Examples

```
(A = matrix(c(1,0,0,2,0,3,1,0,0,5,0,6),3,4,byrow=TRUE))
```

```
(ASM1 = A2SM(A, SMM='CMO', ZINDEX=TRUE))
```

```
(ASM2 = SM2SM(ASM1, SMM2='CRI', ZINDEX2=FALSE))
```

```
SM2A(ASM1)
```

```
SM2A(ASM2)
```

```
(A = matrix(c(1,0,0,2,0,3,1,0,0,5,0,6),3,4,byrow=TRUE)); A[2,3]=NA; A
```

```
(ASM1 = A2SM(A,SMM='CMO',ZINDEX=TRUE)); SM2A(ASM1)
(ASM2 = SM2SM(ASM1,SMM2='CRI',ZINDEX2=TRUE)); A ; SM2A(ASM2)

#CMO 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=TRUE
ASM=list(nr=nr,nc=nc,ra=ra,ia=ia,ja=ja,SMM=SMM,ZINDEX=ZINDEX)
SM2A(ASM)
```

---

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.LP

*Write.LP*

---

### Description

Write a linear programming model to a csv file.

### Usage

```
Write.LP(LP, path = NULL, filename = "LP.csv")
```

**Arguments**

LP	A linear programming object.
path	Directory to write LP .csv file to. REQUIRED to use function
filename	Character string specifying the csv file name.

**Details**

This function converts a linear programming model to a dense representation and writes it to a csv file that can be opened or solved using spreadsheet based solvers.

**Value**

A csv file written to disk.

# Index

A2SM, [3](#)

cbindSM, [4](#)

CST11, [5](#)

CST12, [5](#)

CST21, [6](#)

CST22, [6](#)

DEAbuild, [7](#)

iter\_delete, [8](#)

lagMat, [9](#)

LP\_highs, [10](#)

LPSOLVER, [11](#)

merge\_lists, [11](#)

my\_lag, [12](#)

my\_seconds, [13](#)

nCm\_mpick, [13](#)

qDEA, [14](#)

qDEA\_mlist, [20](#)

qDEA\_solve, [22](#)

qDEAbuild, [24](#)

rbindSM, [26](#)

SM2A, [27](#)

SM2SM, [28](#)

sort\_list, [29](#)

Write.LP, [29](#)