EFFICIENTLY SOLVING DEA MODELS WITH GAMS

ERWIN KALVELAGEN

Abstract. Data Envelopment Analysis deals with solving a series of small linear programming models. This document describes a simple way to combine a number of these small models to improve performance. Especially the with the current crop of state-of-the-art linear programming solvers it is beneficial to solve these small models in relative large batches.

1. Data envelopment analysis

Data envelopment analysis or DEA [3, 4, 7] is an LP based technique for evaluating the relative efficiency of Decision Making Units (DMU's). In many cases the performance non-profit and government organizational units is very difficult to compare: their outputs are not readily comparable and no monetary value can be easily assigned to inputs or outputs. With this technique, one can make draw some conclusions, using concept related to an efficient frontier known from quadratic programming applications in finance [13]. It is a non-parametric method: we don't need an explicit specification of the functional relationship between inputs and outputs (i.e. a production function) [5].

We assume that each DMU j has multiple inputs $x_{i,j}$ and multiple outputs $y_{k,j}$. A relative efficiency measure is defined by:

$$\text{Efficiency} = \frac{\sum_k u_k y_{k,j}}{\sum_i v_i x_{i,j}} \tag{1}$$

where u and v are weights. Often the efficiency is scaled so that it ranges from $[0, 1]$.

The weights form a problem: setting a uniform value for them over all DMU's is rather arbitrary. The main idea behind DEA, is that we allow each DMU j_0 to set its own weights. It can use the following optimization problem for that: maximize the efficiency of DMU j_0 subject to the condition that all efficiencies of other DMU's remain less than or equal to 1. I.e.

$$\begin{aligned} \underset{u,v}{\text{maximize}}\ \theta_0 &= \frac{\sum_k u_k y_{k,j_0}}{\sum_i v_i x_{i,j_0}} \\ \text{subject to } \frac{\sum_k u_k y_{k,j}}{\sum_i v_i x_{i,j}} &\le 1\ \forall j \\ u_k, v_i &\ge 0 \end{aligned} \tag{2}$$

This is not an LP however. A simple work around is to fix the denominator to a constant value, e.g. 1.0, which can be interpreted as setting a constraint on the

Date: November 12, 2002, updated November 2, 2004.

weights v_i (often weights are normalized to add up to one; this can be considered as a slightly more complex normalization). This results in:

$$\begin{aligned} \underset{u,v}{\text{maximize}} \; & \sum_k u_k y_{k,j_0} \\ \text{subject to} \; & \sum_i v_i x_{i,j_0} = 1 \\ & \sum_k u_k y_{k,j} \le \sum_i v_i x_{i,j} \; \forall j \\ & u_k, v_i \ge 0 \end{aligned} \tag{3}$$

It is noted that x and y are no decision variables but rather data. The decision variables are the weights u and v.

In some places [7] the dual has been mentioned as being preferable from a computational point of view (typical primal models have many more rows than columns). The dual DEA model can be stated as:

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \; & z_0 = \Theta_{j_0} \\ & \sum_j \lambda_j y_{k,j} \ge y_{k,j_0} \\ & \Theta_{j_0} x_{i,j_0} \ge \sum_j \lambda_j x_{i,j} \\ & \lambda_j \ge 0 \end{aligned} \tag{4}$$

Other forms for the DEA model have been proposed. The model we discussed above is called the CCR model after the authors of [3]. Some variants set a lower bound on u_k and v_i to prevent zero weights: $u_k \ge \varepsilon$, $v_i \ge \varepsilon$. Another basic model is the BCC model [1]. This model is based on the dual, and adds a restriction on the λ's:

$$\begin{aligned} \underset{\lambda}{\text{minimize}} \; & z_0 = \Theta_{j_0} \\ & \sum_j \lambda_j y_{k,j} \ge y_{k,j_0} \\ & \Theta_{j_0} x_{i,j_0} \ge \sum_j \lambda_j x_{i,j} \\ & \sum_j \lambda_j = 1 \\ & \lambda_j \ge 0 \end{aligned} \tag{5}$$

This transforms the model from being "constant returns-to-scale" to "variable returns-to-scale." The scores from this model are sometimes called "pure technical efficiency scores" as they eliminate scale-efficiency from the analysis [2, 17].

2. GAMS implementation

We have to repeat the solution of the DEA LP model for every DMU. In GAMS this is coded quite easily using a loop:

Model dea.gms. [1]

```
$ontext

   Data Envelopment Analysis (DEA) example

   Erwin Kalvelagen, may 2002

   Data from:
      Emrouznejad, A (1995-2001),
      " Ali Emrouznejad's DEA HomePage",
      Warwick Business School, Coventry CV4 7AL, UK



$offtext



sets i       "DMU's"  /Depot1*Depot20/
     j       'inputs and outputs' /stock, wages, issues, receipts, reqs/
     inp(j)  'inputs'  /stock, wages/
     outp(j) 'outputs' /issues, receipts, reqs/
;


Table data(i,j)
         stock  wages  issues  receipts  reqs
Depot1     3      5      40       55      30
Depot2     2.5    4.5    45       50      40
Depot3     4      6      55       45      30
Depot4     6      7      48       20      60
Depot5     2.3    3.5    28       50      25
Depot6     4      6.5    48       20      65
Depot7     7     10      80       65      57
Depot8     4.4    6.4    25       48      30
Depot9     3      5      45       64      42
Depot10    5      7      70       65      48
Depot11    5      7      45       65      40
Depot12    2      4      45       40      44
Depot13    5      7      65       25      35
Depot14    4      4      38       18      64
Depot15    2      3      20       50      15
Depot16    3      6      38       20      60
Depot17    7     11      68       64      54
Depot18    4      6      25       38      20
Depot19    3      4      45       67      32
Depot20    3      6      57       60      40
;


parameter
   x0(inp)   'inputs of DMU j0'
   y0(outp)  'outputs of DMU j0'
   x(inp,i)  'inputs of DMU i'
   y(outp,i) 'outputs of DMU i'
;

positive variables
   v(inp)  'input weights'
   u(outp) 'output weights'
;

variable
   eff 'efficiency'
;

equations
   objective  'objective function: maximize efficiency'
   normalize  'normalize input weights'
```

[1] http://amsterdamoptimization.com/models/dea/dea.gms

```
   limit(i)   "limit other DMU's efficiency";

objective..  eff =e= sum(outp, u(outp)*y0(outp));

normalize..  sum(inp, v(inp)*x0(inp)) =e= 1;

limit(i)..   sum(outp, u(outp)*y(outp,i)) =l= sum(inp, v(inp)*x(inp,i));


model dea /objective, normalize, limit/;


alias (i,iter);

x(inp,i) = data(i,inp);
y(outp,i) = data(i,outp);

parameter efficiency(i) 'efficiency of each DMU';

loop(iter,

   x0(inp) = x(inp, iter);
   y0(outp) = y(outp, iter);

   solve dea using lp maximizing eff;
   abort$(dea.modelstat<>1) "LP was not optimal";

   efficiency(iter) = eff.l;
);


display efficiency;

*
* create sorted output
*
set r /rnk1*rnk1000/;
parameter rank(i);
alias (i,ii);
rank(i) = sum(ii$(efficiency(ii)>=efficiency(i)), 1);
parameter efficiency2(r,i);
efficiency2(r,i)=efficiency(i)$(rank(i)=ord(r));
option efficiency2:4:0:1;
display efficiency2;
```

The result is:

```
----    105 PARAMETER efficiency2

rnk4 .Depot12 1.0000 rnk4 .Depot14 1.0000 rnk4 .Depot15 1.0000
rnk4 .Depot19 1.0000 rnk5 .Depot9  0.9634 rnk6 .Depot20 0.9517
rnk7 .Depot5  0.9466 rnk8 .Depot2  0.9417 rnk9 .Depot16 0.9091
rnk10.Depot10 0.8889 rnk11.Depot13 0.8254 rnk12.Depot6  0.8228
rnk13.Depot1  0.8204 rnk14.Depot3  0.8148 rnk15.Depot7  0.7111
rnk16.Depot4  0.6528 rnk17.Depot11 0.6313 rnk18.Depot17 0.5495
rnk19.Depot8  0.5169 rnk20.Depot18 0.4201
```

The sorting step is interesting: it is rather non-intuitive, but it works. Notice the behavior when multiple entries have the same values.

An alternative formulation can be formed by not copying data into $x0$ and $y0$ but to make the model "indexed" on a set. We then loop over this set.

Model dea2.gms. [2]

```
$ontext

   Data Envelopment Analysis (DEA) example
   Indexed equations formulation.
```

[2] http://amsterdamoptimization.com/models/dea/dea2.gms

```
   Erwin Kalvelagen, may 2002

   Data from:
      Emrouznejad, A (1995-2001),
      " Ali Emrouznejad's DEA HomePage",
      Warwick Business School, Coventry CV4 7AL, UK



$offtext



sets i       "DMU's"  /Depot1*Depot20/
     j       'inputs and outputs' /stock, wages, issues, receipts, reqs/
     inp(j)  'inputs'  /stock, wages/
     outp(j) 'outputs' /issues, receipts, reqs/
;


set j0(i) 'current DMU';

Table data(i,j)
         stock  wages  issues  receipts  reqs
Depot1     3      5      40       55      30
Depot2     2.5    4.5    45       50      40
Depot3     4      6      55       45      30
Depot4     6      7      48       20      60
Depot5     2.3    3.5    28       50      25
Depot6     4      6.5    48       20      65
Depot7     7     10      80       65      57
Depot8     4.4    6.4    25       48      30
Depot9     3      5      45       64      42
Depot10    5      7      70       65      48
Depot11    5      7      45       65      40
Depot12    2      4      45       40      44
Depot13    5      7      65       25      35
Depot14    4      4      38       18      64
Depot15    2      3      20       50      15
Depot16    3      6      38       20      60
Depot17    7     11      68       64      54
Depot18    4      6      25       38      20
Depot19    3      4      45       67      32
Depot20    3      6      57       60      40
;


parameter
   x(inp,i)  'inputs of DMU i'
   y(outp,i) 'outputs of DMU i'
;

positive variables
   v(inp)  'input weights'
   u(outp) 'output weights'
;

variable
   eff 'efficiency'
;

equations
   objective(i)   'objective function: maximize efficiency'
   normalize(i)   'normalize input weights'
   limit(i)       "limit other DMU's efficiency";

objective(j0)..  eff =e= sum(outp, u(outp)*y(outp,j0));

normalize(j0)..  sum(inp, v(inp)*x(inp,j0)) =e= 1;

limit(i)..   sum(outp, u(outp)*y(outp,i)) =l= sum(inp, v(inp)*x(inp,i));
```

```
model dea /objective, normalize, limit/;



alias(i,iter);

x(inp,i) = data(i,inp);
y(outp,i) = data(i,outp);

parameter efficiency(i) 'efficiency of each DMU';

loop(iter,

*
* set j0 is the current DMU
*
   j0(i) = no;
   j0(iter) = yes;

   solve dea using lp maximizing eff;
   abort$(dea.modelstat<>1) "LP was not optimal";

   efficiency(iter) = eff.l;
);


display efficiency;

*
* create sorted output
*
set r /rnk1*rnk1000/;
parameter rank(i);
alias (i,ii);
rank(i) = sum(ii$(efficiency(ii)>=efficiency(i)), 1);
parameter efficiency2(r,i);
efficiency2(r,i)=efficiency(i)$(rank(i)=ord(r));
option efficiency2:4:0:1;
display efficiency2;
```

Note that the set j_0 is a dynamic set. The equations are therefore declared over the set i, which is a static set. We then define the equations over the set j_0 which will be calculated inside the loop.

GAMS protects the modeler by forbidding the loop set to be used in equations. However that is exactly what we need here. To work around this, we use a different loop set *iter* and calculate the set j_0 inside the loop.

3. Performance issues

The LP's in the model are all very small: 22 equations and 6 variables. Nevertheless GAMS will get slow if the number of DMU's gets large. Part of it we can easily fix: the large amount of data written to the listing file. This can be reduced to a minimum by the following statements:

- `option limrow=0;` to remove the equation listing
- `option limcol=0;` to remove the column listing
- `option solprint=off;` to remove the solution listing
- `model.solprint=2;` to suppress even more solver output
- `model.solvelink=2;` to keep GAMS in memory while the solver executes

This will speed up GAMS but as the loop unfolds, GAMS may still become unbearably slow. Basically, GAMS has too much overhead in solving very small models in a loop. We can alleviate this by folding several small LP's into one. For the model above, we can solve the whole thing in one swoop. Say a single model for DMU i has the standard LP format:

$$\begin{aligned} &\underset{x_i}{\text{maximize}}\ c_i^T x_i \\ &A_i x_i = b_i \\ &\ell_i \le x_i \le u_i \end{aligned} \tag{6}$$

then a combined model can look like:

$$\begin{aligned} &\underset{x}{\text{maximize}}\ \sum_i c_i^T x_i \\ &Ax = b \\ &\ell \le x \le u \end{aligned} \tag{7}$$

where $x^T = \left(x_1^T x_2^T \dots x_n^T\right)$ and

$$A = \begin{pmatrix} A_1 & & & \\ & A_2 & & \\ & & \ddots & \\ & & & A_n \end{pmatrix} \tag{8}$$

I.e. the matrix becomes a block-diagonal matrix with disconnected blocks. In a GAMS model we can implement this by introducing an extra index on all variables and equations.

Model dea3.gms. [3]

```
$ontext

   Data Envelopment Analysis (DEA) example
   One LP formulation.

   Erwin Kalvelagen, may 2002

   Data from:
      Emrouznejad, A (1995-2001),
      " Ali Emrouznejad's DEA HomePage",
      Warwick Business School, Coventry CV4 7AL, UK



$offtext



sets i       "DMU's"  /Depot1*Depot20/
     j       'inputs and outputs' /stock, wages, issues, receipts, reqs/
     inp(j)  'inputs'  /stock, wages/
     outp(j) 'outputs' /issues, receipts, reqs/
;


alias (i,j0);

Table data(i,j)
         stock  wages  issues  receipts  reqs
Depot1     3      5      40       55      30
Depot2     2.5    4.5    45       50      40
Depot3     4      6      55       45      30
Depot4     6      7      48       20      60
Depot5     2.3    3.5    28       50      25
```

[3] http://amsterdamoptimization.com/models/dea/dea3.gms

```
Depot6     4      6.5    48       20      65
Depot7     7     10      80       65      57
Depot8     4.4    6.4    25       48      30
Depot9     3      5      45       64      42
Depot10    5      7      70       65      48
Depot11    5      7      45       65      40
Depot12    2      4      45       40      44
Depot13    5      7      65       25      35
Depot14    4      4      38       18      64
Depot15    2      3      20       50      15
Depot16    3      6      38       20      60
Depot17    7     11      68       64      54
Depot18    4      6      25       38      20
Depot19    3      4      45       67      32
Depot20    3      6      57       60      40
;


parameter
   x(inp,i)  'inputs of DMU i'
   y(outp,i) 'outputs of DMU i'
;

positive variables
   v(inp,j0)   'input weights'
   u(outp,j0)  'output weights'
;

variable
   eff(j0)     'efficiency of each DMU'
   totaleff    'summation of all efficiencies'
;

equations
   objective(j0)   'objective function: maximize efficiency'
   normalize(j0)   'normalize input weights'
   limit(i,j0)     "limit other DMU's efficiency"
   totalobj        'combines all individual objective functions'
;

totalobj..       totaleff =e= sum(j0, eff(j0));

objective(j0)..  eff(j0) =e=  sum(outp, u(outp,j0)*y(outp,j0));

normalize(j0)..  sum(inp, v(inp,j0)*x(inp,j0)) =e= 1;

limit(i,j0)..    sum(outp, u(outp,j0)*y(outp,i)) =l= sum(inp, v(inp,j0)*x(inp,i));


model dea /objective, normalize, limit, totalobj/;

x(inp,i) = data(i,inp);
y(outp,i) = data(i,outp);

solve dea using lp maximizing totaleff;


*
* create sorted output
*
set r /rnk1*rnk1000/;
parameter rank(i);
alias (i,ii);
rank(i) = sum(ii$(eff.l(ii)>=eff.l(i)), 1);
parameter efficiency2(r,i);
efficiency2(r,i)=eff.l(i)$(rank(i)=ord(r));
option efficiency2:4:0:1;
display efficiency2;
```

This model has just 441 equations and 121 variables, so it is still very small for current standards. In this case, a one LP formulation solves much quicker than formulating twenty little models, one for each DMU. (We note that the actual matrix being generated is not block-diagonal, but rather permuted block-diagonal: after some simple row and column swaps the matrix can be made block-diagonal).

If you have many DMU's it is possible to find a balance between looping and solving big LP's. E.g. suppose one has 100 DMU's, then it may make sense to solve 5 batches of 20 combined problems.

In the example below we set up a set *dist* which determines the distribution of DMU's over runs. In this case we have two runs. This first takes care of DMU's 1 through 10, while the second run does DMU's 11 through 20.

Model dea4.gms. [4]

```
$ontext

   Data Envelopment Analysis (DEA) example
   Flexible batch formulation

   Erwin Kalvelagen, may 2002

   Data from:
      Emrouznejad, A (1995-2001),
      " Ali Emrouznejad's DEA HomePage",
      Warwick Business School, Coventry CV4 7AL, UK



$offtext



sets i       "DMU's"  /Depot1*Depot20/
     j       'inputs and outputs' /stock, wages, issues, receipts, reqs/
     inp(j)  'inputs'  /stock, wages/
     outp(j) 'outputs' /issues, receipts, reqs/
;


set j0(i) 'current DMU';

Table data(i,j)
         stock  wages  issues  receipts  reqs
Depot1     3      5     40       55      30
Depot2     2.5    4.5   45       50      40
Depot3     4      6     55       45      30
Depot4     6      7     48       20      60
Depot5     2.3    3.5   28       50      25
Depot6     4      6.5   48       20      65
Depot7     7     10     80       65      57
Depot8     4.4    6.4   25       48      30
Depot9     3      5     45       64      42
Depot10    5      7     70       65      48
Depot11    5      7     45       65      40
Depot12    2      4     45       40      44
Depot13    5      7     65       25      35
Depot14    4      4     38       18      64
Depot15    2      3     20       50      15
Depot16    3      6     38       20      60
Depot17    7     11     68       64      54
Depot18    4      6     25       38      20
Depot19    3      4     45       67      32
Depot20    3      6     57       60      40
;


set run 'batch run' /run1*run2/;
set dist(run,i) 'distribution' /
```

[4] http://amsterdamoptimization.com/models/dea/dea4.gms

```
    run1.(depot1*depot10),
    run2.(depot11*depot20)
/;


parameter
   x(inp,i)  'inputs of DMU i'
   y(outp,i) 'outputs of DMU i'
;

positive variables
   v(inp,i)   'input weights'
   u(outp,i)  'output weights'
;

variable
   eff(i)      'efficiency of each DMU'
   totaleff    'summation of all efficiencies'
;

equations
   objective(i)   'objective function: maximize efficiency'
   normalize(i)   'normalize input weights'
   limit(i,i)     "limit other DMU's efficiency"
   totalobj       'combines all individual objective functions'
;

totalobj..       totaleff =e= sum(j0, eff(j0));

objective(j0)..  eff(j0) =e=  sum(outp, u(outp,j0)*y(outp,j0));

normalize(j0)..  sum(inp, v(inp,j0)*x(inp,j0)) =e= 1;

limit(i,j0)..    sum(outp, u(outp,j0)*y(outp,i)) =l= sum(inp, v(inp,j0)*x(inp,i));


model dea /objective, normalize, limit, totalobj/;

x(inp,i) = data(i,inp);
y(outp,i) = data(i,outp);

parameter efficiency(i) 'efficiency of each DMU';



loop(run,

  j0(i) = no;
  j0(i)$dist(run,i) = yes;

  solve dea using lp maximizing totaleff;

  efficiency(j0) = eff.l(j0);
);


*
* create sorted output
*
set r /rnk1*rnk1000/;
parameter rank(i);
alias (i,ii);
rank(i) = sum(ii$(efficiency(ii)>=efficiency(i)), 1);
parameter efficiency2(r,i);
efficiency2(r,i)=efficiency(i)$(rank(i)=ord(r));
option efficiency2:4:0:1;
display efficiency2;
```

4. Numerical experiments

The best balance between size of a batch and the number of batches need to be determined by experimenting. Some of the state-of-the-art LP solvers are really good now in solving LP models quickly. This means that it is often advantageous to make the batches rather large.

runs	**user time**	**system time**	**total**	
1	0.083	0.052	0.135	all combined
2	0.117	0.062	0.179	$10 + 10$
3	0.136	0.097	0.233	$7 + 7 + 6$
4	0.171	0.091	0.262	$5 + 5 + 5 + 5$
5	0.181	0.125	0.306	$4 + 4 + 4 + 4 + 4$
7	0.210	0.167	0.377	$3 + 3 + 3 + 3 + 3 + 3 + 2$
10	0.320	0.224	0.544	2 each
20	0.527	0.412	0.939	all individual

Table 1. Performance results for `dea4.gms`

The above model is very small, so when we tried actual runs, the fastest strategy was to combine all models in a single run. The timings are on a 1Ghz dual pentium machine running Linux and were obtained using the `time` utility of the c-shell. For this small example we see that combining the 20 models into one run gives us a speed-up of almost a factor 10.

	time				time		
runs	**user**	**system**	**total**	**runs**	**user**	**system**	**total**
1	7.607	0.361	7.968	16	5.107	0.851	5.958
2	6.037	0.408	6.445	17	5.142	0.814	5.956
3	5.777	0.369	6.146	18	5.113	0.875	5.988
4	5.523	0.396	5.919	19	5.146	0.894	6.04
5	5.421	0.423	5.844	20	5.177	0.898	6.075
6	5.392	0.482	5.874	21	5.218	0.955	6.173
7	5.388	0.552	5.94	22	5.222	1.013	6.235
8	5.443	0.523	5.966	23	5.275	0.966	6.241
9	5.451	0.574	6.025	24	5.302	1.019	6.321
10	5.464	0.636	6.1	25	5.271	1.066	6.337
11	5.492	0.589	6.081	30	5.455	1.242	6.697
12	5.341	0.66	6.001	40	6.005	1.503	7.508
13	5.248	0.728	5.976	50	6.48	1.822	8.302
14	5.175	0.769	5.944	100	9.136	3.501	12.637
15	5.181	0.765	5.946	200	15.125	6.589	21.714

Table 2. Performance results for 200 DMU model

For a larger proprietary model we used the following GAMS code:

```
$set n 10
set run 'batch run' /run1*run%n%/;
set dist(run,i);
set current(run);
```

```
current('run1') = yes;
loop(i,
    dist(current,i) = yes;
    current(run++1) = current(run);
);
display dist;
```

Given a value for the environment variable n (the number of batch runs), this fragment will distribute the subproblems i over the runs. We can set n to any number. To perform the timing we used a model with 200 DMU's, and varied n between 1 and 200. Running the model in one run resulted in an LP with 40401 equations and 1601 variables. Each individual model is: 201 equations and 7 variables. The performance results are shown in table 4. Here we see that there is a wide range of relative efficient combinations. Combining all models into one is not the best approach here.

5. Examples

5.1. **Dual formulation.** In this example we show how the dual formulations of the Constant Returns to Scale CCR model (equation 4) and the Variable Returns to Scale BCC model (equation 5) can be solved as one big LP model instead of a series of small models.

We use the data set from [11].

Model bundesliga.gms. [5]

```
$ontext

   DEA models:
         input and output oriented
         constant returns to scale (CCR) and variable returns to scale (BCC)

   Instead of a loop batch equations together to forma single large LP.

   Erwin Kalvelagen jan 2005

   Reference:
       Dieter Haas, Martin G. Kocher and Matthias Sutter,
       "Measuring Efficiency of German Football Teams by Data Envelopment Analysis",
       University of Innsbruck, 12 may 2003

$offtext


set i 'teams' /
     'Bayern München'
     'Bayer Leverkusen'
     'Hamburger SV'
     '1860 München'
     '1. FC Kaiserslautern'
     'Hertha BSC'
     'Vfl Wolfsburg'
     'Vfb Stuttgart'
     'Werder Bremen'
     'SpVgg Unterhaching'
     'Borussia Dortmund'
     'SC Freiburg'
     'FC Schalke'
     'Eintracht Frankfurt'
     'Hansa Rostock'
     'SSV Ulm'
     'Arminia Bielefeld'
```

[5] http://amsterdamoptimization.com/models/dea/bundesliga.gms

```
     'MSV Duisburg'
/;

set j 'data keys' /
    rank    'ranking at end of season 1999/2000'
    wagep   'avg wage for players (annual, million dm)'
    wagec   'wage for coach (monthly, 1000 dm)'
    points  'points determining ranking'
    spect   'spectators (1000)'
    fill    'stadium utilization (%)'
    rev     'total revenue (million DM)'
    CL      'participation in Champions League'
    UC      'participation in UEFA Cup'

/;

table data(i,j)
                         rank    wagep   wagec points  spect fill  rev   CL   UC
   'Bayern München'        1     63.0     300    73     894  83.5  220   1
   'Bayer Leverkusen'      2     30.5     180    73     382  89.7   85   1    1
   'Hamburger SV'          3     31.0     125    59     703  76.6   61
   '1860 München'          4     30.0     160    53     555  51.8   42
   '1. FC Kaiserslautern'  5     31.0     200    50     684  96.9   75        1
   'Hertha BSC'            6     32.5     100    50     809  62.8   42   1
   'Vfl Wolfsburg'         7     19.0      80    49     292  83.5   40        1
   'Vfb Stuttgart'         8     20.5     100    48     500  65.3   52
   'Werder Bremen'         9     20.0      30    47     507  84.5   63        1
   'SpVgg Unterhaching'   10     12.0      30    44     163  76.6   14
   'Borussia Dortmund'    11     60.0     100    40    1099  93.7  150   1    1
   'SC Freiburg'          12      9.5      50    40     420  98.8   31
   'FC Schalke'           13     40.0      70    39     689  65.4   64
   'Eintracht Frankfurt'  14     20.0      80    39     605  58.3   40
   'Hansa Rostock'        15     14.0      35    38     275  66.0   32
   'SSV Ulm'              16      8.0      22    35     371  97.0   26
   'Arminia Bielefeld'    17     16.0      50    30     335  74.4   32
   'MSV Duisburg'         18     11.5      42    22     257  50.1   28
;

display data;

set inp(j) 'inputs'   /wagep,wagec/;
set outp(j) 'outputs' /points,fill,rev/;

parameter x(inp,i);  x(inp,i) = data(i,inp);
parameter y(outp,i); y(outp,i) = data(i,outp);

alias(i,i0);

positive variables lambda(i0,i);

variables
   theta(i0)    'efficiency for i0-th DMU'
   z            'sum of efficiencies'
;

*-----------------------------------------------------------------------------
* input oriented versions of constant returns to scale (CCR) and
* variable returns to scale (BCC) models
*-----------------------------------------------------------------------------


equations
   objective
   input1(i0,outp)
   input2(i0,inp)
   convex(i0)
;

objective..       z =e= sum(i0, theta(i0));

input1(i0,outp).. sum(i,lambda(i0,i)*y(outp,i)) =g= y(outp,i0);
```

```
input2(i0,inp)..  theta(i0)*x(inp,i0) =g= sum(i, lambda(i0,i)*x(inp,i));

convex(i0)..      sum(i, lambda(i0,i)) =e= 1;


model input_ccr /objective,input1,input2/;
model input_bcc /objective,input1,input2,convex/;

parameter results(i0,*,*);

solve input_ccr using lp minimizing z;
results(i0,'input','CRS/CCR') = theta.l(i0);
solve input_bcc using lp minimizing z;
results(i0,'input','VRS/BCC') = theta.l(i0);

*-----------------------------------------------------------------------------
* output oriented versions of constant returns to scale (CCR) and
* variable returns to scale (BCC) models
*-----------------------------------------------------------------------------

equations
   output1(i0,inp)
   output2(i0,outp)
;

output1(i0,inp)..   sum(i,lambda(i0,i)*x(inp,i)) =l= x(inp,i0);

output2(i0,outp)..  theta(i0)*y(outp,i0) =l= sum(i, lambda(i0,i)*y(outp,i));

model output_ccr /objective,output1,output2/;
model output_bcc /objective,output1,output2,convex/;


solve output_ccr using lp maximizing z;
results(i0,'output','CRS/CCR') = theta.l(i0);
solve output_bcc using lp maximizing z;
results(i0,'output','VRS/BCC') = theta.l(i0);


option results:4:1:2;
display results;
```

In the example both input oriented and output oriented efficiency scores are calculated and presented in a results parameter:

```
----    149 PARAMETER results

                          input       input      output      output
                        CRS/CCR     VRS/BCC     CRS/CCR     VRS/BCC

Bayern München           1.0000      1.0000      1.0000      1.0000
Bayer Leverkusen         0.8288      1.0000      1.2065      1.0000
Hamburger SV             0.5897      0.7968      1.6956      1.0757
1860 München             0.4282      0.5918      2.3354      1.3119
1. FC Kaiserslautern     0.7098      1.0000      1.4089      1.0000
Hertha BSC               0.3934      0.5545      2.5419      1.1827
Vfl Wolfsburg            0.6423      0.8394      1.5568      1.0665
Vfb Stuttgart            0.7578      0.8107      1.3196      1.1527
Werder Bremen            1.0000      1.0000      1.0000      1.0000
SpVgg Unterhaching       0.9219      1.0000      1.0847      1.0000
Borussia Dortmund        0.7893      1.0000      1.2670      1.0000
SC Freiburg              1.0000      1.0000      1.0000      1.0000
FC Schalke               0.5037      0.5039      1.9851      1.3067
Eintracht Frankfurt      0.5997      0.6003      1.6674      1.3698
Hansa Rostock            0.7073      0.7443      1.4139      1.1465
SSV Ulm                  1.0000      1.0000      1.0000      1.0000
Arminia Bielefeld        0.6054      0.6069      1.6518      1.3136
MSV Duisburg             0.7242      0.7450      1.3809      1.3695
```

5.2. **Bootstrapping.** Bootstrapping[6, 16] is used to provide additional information for statistical inference. The following model from [19] implements a resampling strategy from [15]. Two thousand bootstrap samples are formed, each resulting in a DEA model of 100 small LP's. In this example we batch the DEA models together in a single large LP, so that we only have to solve 2,000 LP models instead of 200,000.

Model bootstrap.gms. [6]

```
$ontext

   DEA bootstrapping example

   Erwin Kalvelagen, october 2004

   References:

     Mei Xue, Patrick T. Harker
     "Overcoming the Inherent Dependency of DEA Efficiency Scores:
     A Bootstrap Approach", Tech. Report, Department of Operations and
     Information Management, The Wharton School, University of Pennsylvania,
     April 1999

     http://opim.wharton.upenn.edu/~harker/DEAboot.pdf

$offtext


sets
   i 'hospital (DMU)' /h1*h100/
   j 'inputs and outputs' /
     FTE     'The number of full time employees in the hospital in FY 1994-95'
     Costs   'The expenses of the hospital ($million) in FY 1994-95'
     PTDAYS  'The number of the patient days produced by the hospital in FY 1994-95'
     DISCH   'The number of patient discharges produced by the hospital in FY 1994-95'
     BEDS    'The number of patient beds in the hospital in FY 1994-95'
     FORPROF 'Dummy variable, one if it is for-profit hospital, zero otherwise'
     TEACH   'Dummy variable, one if it is teaching hospital, zero otherwise'
     RES     'The number of the residents in the hospital in FY 1994-95'
     CONST   'Constant term in regression model'
   /
   inp(j) 'inputs' /FTE,Costs/
   outp(j) 'outputs' /PTDAYS,DISCH/
;

table data(i,j)

          FTE      Costs        PTDAYS   DISCH   BEDS FORPROF TEACH   RES

  h1     1571.86    174          71986   12665   365
  h2      816.54     69.9        53081    5861   224
  h3      533.74     61.7        25030    4951   286     1
  h4      805.2      75.4        34163   11877   256
  h5     3908.1     396         187462   42735   829            1    136.8
  h6      727.72     63.9        31330    8402   194
  h7     2571.75    220         130077   26877   620            1     42.81
  h8      521        89.1        43390    8598   290     1
  h9      718        50          27896    6113   150            1     23.21
  h10    1504.85    121          75941   16427   393
  h11    1234.49     84.6        57080   14180   317
  h12     873        68.8        48932   12060   281
  h13    1067.17     85.8        50436   11317   278
  h14     668        47.5        67909    6235   244
  h15     452.35     36.4        25200    6860   155     1      1     13.31
  h16    1523        97.4        59809   13180   394
  h17    3152       198         108631   22071   578            1    195.67
  h18     871.96     30.7        17925    4605   160
  h19    2901.86    290         130004   24133   549            1    126.89
```

[6] http://amsterdamoptimization.com/models/dea/bootstrap.gms

h20	902.4	78.2	35743	8664	236		1	12.08
h21	194.69	10.9	15555	1530	132			
h22	713.51	62.6	32558	8966	138			
h23	557.36	23.8	12728	2291	276	1		
h24	2259.2	120	74061	12942	348		1	14.52
h25	462.22	32.4	28886	6101	134			
h26	1212.1	97.3	74194	12681	342			
h27	2391.94	192	89843	18396	336		1	229.19
h28	1637	162	80468	21345	415			
h29	501	37.9	26813	4594	166	1		
h30	412.1	40.2	23217	6044	160	1		
h31	738.56	27	11514	3052	144	1		
h32	414.1	35.7	55611	4354	200			
h33	1097	105	59443	13101	242		1	26.32
h34	742	62.8	42542	8739	172			
h35	1010	97.1	47246	12073	269		1	1.1
h36	440.6	34.2	30773	4305	201			
h37	1203.3	85.4	50710	11470	247		1	13.82
h38	2558.01	195	128450	20441	571		1	5.42
h39	215.45	8.409936	65743	578	238			
h40	599.3	30.4	23299	5338	173			
h41	480.55	29.5	34279	6560	169	1		
h42	634.51	29.9	27157	5198	141			
h43	1211.9	91.4	90008	17666	320		1	6.25
h44	285.5	23.9	16473	2873	135			
h45	1030.36	67.1	43486	9467	235		1	6.44
h46	1374.81	95.5	74279	11862	284			
h47	953.56	49.8	47934	10553	207			
h48	561.11	41.7	24800	5498	132			
h49	644	57.1	39663	8604	260			
h50	376.55	19.6	22003	4759	143			
h51	404.79	32.8	27566	7871	190	1		
h52	397.9	29.4	26072	4248	170			
h53	374.2	3.944649	4179	819	156			
h54	1702	100	114603	17235	438		1	11.81
h55	148.09	5.013379	51660	771	172			
h56	253.48	16.9	17599	4044	178			
h57	1445.68	99.3	81041	12912	475		1	17.53
h58	414.1	26.5	20432	4068	129			
h59	642.58	48.5	42733	5983	181	1		
h60	203.75	13	16923	3467	146	1		
h61	421.8	18.3	16179	2840	160			
h62	320.62	17.3	18882	3370	160			
h63	679.79	25.6	27561	4447	308		1	11.33
h64	2382	226	166559	26019	787		1	7.08
h65	559.29	58.1	40534	8806	342	1		
h66	568.15	35	37120	7242	158			
h67	2408.04	155	70392	9538	266		1	111.33
h68	632.34	54.6	37228	6359	175			
h69	917.22	55.2	42135	7294	215			
h70	554.34	56.9	32352	3320	205		1	1
h71	780	75.9	39213	7154	172			
h72	663.82	56.9	34180	5284	200			
h73	1424	146	107457	18198	432		1	2.75
h74	313	20.7	20110	5967	165	1		
h75	778	78.4	51496	12302	390			
h76	863.37	62	50957	10557	228			
h77	3509.12	290	109673	19213	469		1	290.53
h78	1593.82	152	82400	17707	474		1	11.64
h79	466	40.1	30647	7265	164	1		
h80	666.38	48.2	28048	5182	153			
h81	998.8	121	45513	6855	238		1	88.86
h82	1018	98.2	61176	11386	350			
h83	3238.28	326	122118	19068	514		1	146.33
h84	1431.1	107	48900	10623	208			
h85	1735.99	273	84118	16458	278		1	158.4
h86	1769	190	105741	19256	478		1	0.93
h87	484.56	36.2	24070	6464	125			
h88	204.7	13.9	28137	1615	135	1		
h89	1706.58	287	75153	13465	367		1	91.56
h90	1029.11	71.9	49993	6690	252		1	4
h91	1167.2	111	75004	21334	350			

```
  h92   1657.58     116         77753   17528   413
  h93   1017.16      88.5       64147   11135   316
  h94   1532.7      153         99998   17391   395            1      4.8
  h95   1462        113        119107   16053   484            1      0.5
  h96   1133.8      109         55540   15566   355            1      8.51
  h97    609         48.2       71817    5639   376            1      1
  h98    301.31      20.2       43214    2153   141
  h99   1930.08     201         87197   19315   418
  h100  1573.3      177         88124   19661   458            1     69.71
;

data(i,'CONST') = 1;


*--------------------------------------------------------------------------------
* PHASE 1: Estimation of b(j)
*
* Run standard Constant Returns to Scale (CCR) Input-oriented DEA model
* followed by linear regression OLS estimation
*--------------------------------------------------------------------------------



*
* this is the standard DEA model
* instead of 100 small models we solve one big model, see
* http://www.gams.com/~erwin/dea/dea.pdf
*
parameter
  x(inp,i)  'inputs of DMU i'
  y(outp,i) 'outputs of DMU i'
;

alias(i,j0);
positive variables
  v(inp,j0)    'input weights'
  u(outp,j0)   'output weights'
;
variable
  eff(j0) 'efficiency'
  z 'objective variable'
;

equations
  objective(j0)  'objective function: maximize efficiency'
  normalize(j0)  'normalize input weights'
  limit(i,j0)    "limit other DMU's efficiency"
  totalobj
;

totalobj..      z =e= sum(j0, eff(j0));
objective(j0).. eff(j0) =e= sum(outp, u(outp,j0)*y(outp,j0));
normalize(j0).. sum(inp, v(inp,j0)*x(inp,j0)) =e= 1;
limit(i,j0)..   sum(outp, u(outp,j0)*y(outp,i)) =l= sum(inp, v(inp,j0)*x(inp,i));

model dea /totalobj,objective, normalize, limit/;

alias (i,iter);

x(inp,i) = data(i,inp);
y(outp,i) = data(i,outp);

option limrow=0;
option limcol=0;
dea.solprint=2;
dea.solvelink=2;

solve dea using lp maximizing z;
abort$(dea.modelstat<>1) "LP was not optimal";

display
  "----------------------------------- DEA MODEL ------------------------",
  eff.l;
```

```
*
* now solve the regression problem
*   efficiency = b0 + b1*BEDS + b2*FORPROF + b3*TEACH + b4*RES
* Use b = inv(X^TX) X^Ty
* Standard errors are sigma^2 inv(X^TX)
* See http://www.gams.com/~erwin/stats/ols.pdf
*

set e(j) 'explanatory variables' /BEDS,FORPROF,TEACH,RES,CONST/;

*
* calculate inv(X^TX)
*
alias(e,ee,eee);
parameter XX(e,ee) 'matrix (X^TX)';
XX(e,ee) = sum(i,data(i,e)*data(i,ee));
parameter Xy(e) 'X^Ty';
Xy(e) = sum(i, data(i,e)*eff.l(i));
parameter ident(e,ee) 'Identity matrix';
ident(e,e)=1;

variable
   invXX(e,ee) 'matrix inv(X^TX)'
   dummy
;

equation
   invert(e,ee)
   edummy
;

invert(e,ee).. sum(eee, XX(e,eee)*invXX(eee,ee)) =e= ident(e,ee);
edummy.. dummy=e=0;
model matinv /invert,edummy/;
matinv.solprint=2;
matinv.solvelink=2;
solve matinv using lp minimizing dummy;

*
* calculate estimates and standard errors
*

parameter b(e);
b(e) = sum(ee, invXX.l(e,ee)*Xy(ee));

parameter resid(i) 'residuals';
resid(i) =  eff.l(i) - sum(e,b(e)*data(i,e));
scalar rss 'residual sum of squares';
rss = sum(i, sqr(resid(i)));


*
* calculate standard errors
*

scalar df 'degrees of freedom';
df = card(i)-card(e);
scalar sigma_squared 'variance of estimate';
sigma_squared = rss/df;
parameter variance(e,ee);
variance(e,ee) = sigma_squared*invXX.l(e,ee);
parameter se(e) 'standard error';
se(e) = sqrt(variance(e,e));

parameter tval(e) "t statistic";
tval(e) = b(e)/se(e);

parameter pval(e) "p-values";

*
*  pvalue = 2 * pt( abs(tvalue), df)
```

```
*         = 2 * 0.5 * pbeta( df / (df + sqr(abs(tvalue))), df/2, 0.5)
*         = betareg( df / (df+sqr(tvalue)), df/2, 0.5)
*
pval(e) = betareg( df / (df+sqr(tval(e))), df/2, 0.5);

parameter ols(e,*);
ols(e,'estimates') = b(e);
ols(e,'std.error') = se(e);
ols(e,'t value') = tval(e);
ols(e,'p value') = pval(e);



display
  "------------------------------------ OLS MODEL ------------------------",
  ols;


*------------------------------------------------------------------------------
* PHASE 2: BOOTSTRAP algorithm
*------------------------------------------------------------------------------

set s 'sample' /sample1*sample2000/;


parameter bs(s,i) 'bootstrap sample';
bs(s,i) = trunc( uniform(1,card(i)+0.999999999) );
*display bs;
* sanity check:
loop((s,i),
   abort$(bs(s,i)<1) "Check bs for entries < 1";
   abort$(bs(s,i)>card(i)) "Check bs for entries > card(i)";
);

alias(i,ii);
set mapbs(s,i,ii);
mapbs(s,i,ii)$(bs(s,i) = ord(ii)) = yes;
* this mapping says the i'th sample data record is the ii'th record
* in the original data (for sample s)

loop((s,i),
  abort$(sum(mapbs(s,i,ii),1)<>1) "mapbs is not unique";
);

parameter data_sample(i,j);

parameter sb(s,e) 'b(e) for each sample s';

loop(s,

*
* solve dea model
*

    data_sample(i,j) = sum(mapbs(s,i,ii),data(ii,j));
    x(inp,i) = data_sample(i,inp);
    y(outp,i) = data_sample(i,outp);

    solve dea using lp maximizing z;
    abort$(dea.modelstat<>1) "LP was not optimal";

*
* solve OLS model
*
    XX(e,ee) = sum(i,data_sample(i,e)*data_sample(i,ee));
    Xy(e) = sum(i, data_sample(i,e)*eff.l(i));
    solve matinv using lp minimizing dummy;
    sb(s,e) = sum(ee, invXX.l(e,ee)*Xy(ee));


);
```

```
*
* get statistics
*
parameter bbar(e) "Averaged estimates";
bbar(e) = sum(s, sb(s,e)) / card(s);

parameter sehat(e) "Standard errors of bootstrap algorithm";
sehat(e) = sqrt(sum(s, sqr(sb(s,e)-bbar(e)))/(card(s)-1));

parameter tbootstrap(e) "t statistic for bootstrap";
tbootstrap(e) = b(e)/sehat(e);

scalar dfbootstrap 'degrees of freedom';
dfbootstrap = card(i) - (card(e) - 1) - 1;
parameter pbootstrap(e) "p-values for bootstrap";

*
*  pvalue = 2 * pt( abs(tvalue), df)
*         = 2 * 0.5 * pbeta( df / (df + sqr(abs(tvalue))), df/2, 0.5)
*         = betareg( df / (df+sqr(tvalue)), df/2, 0.5)
*
pbootstrap(e) = betareg( dfbootstrap / (dfbootstrap+sqr(tbootstrap(e))), dfbootstrap/2, 0.5);

parameter bootstrap(e,*);
bootstrap(e,'estimates') = b(e);
bootstrap(e,'std.error') = sehat(e);
bootstrap(e,'t value') = tbootstrap(e);
bootstrap(e,'p value') = pbootstrap(e);


display
  "------------------------------------ BOOTSTRAP MODEL -----------------------",
  bootstrap;
```

The idea of this model is to build a regression equation:

$$\theta_i = \beta_0 + \beta_1 \text{BEDS}_i + \beta_2 \text{FORPROF}_i + \beta_3 \text{TEACH}_i + \beta_4 \text{RES}_i + \varepsilon_i \tag{9}$$

where θ_i are the DEA efficiency scores. From the results

```
----    290 ------------------------------------ OLS MODEL -----------------------

----    290 PARAMETER ols

          estimates   std.error     t value     p value

BEDS    1.040019E-4 1.244050E-4       0.836       0.405
FORPROF       0.099       0.042       2.390       0.019
TEACH        -0.057       0.039      -1.451       0.150
RES          -0.001 3.303407E-4      -3.133       0.002
CONST         0.607       0.035      17.330 3.59753E-31
```

we see that FORPROF is significant at $\alpha = 0.05$ (the corresponding p value is smaller than 0.05). However when we apply the resampling technique from the bootstrap algorithm, the results indicate a different interpretation:

```
----    380 ------------------------------------ BOOTSTRAP MODEL -----------------------

----    380 PARAMETER bootstrap

          estimates   std.error     t value     p value

BEDS    1.040019E-4 1.107967E-4       0.939       0.350
FORPROF       0.099       0.060       1.651       0.102
TEACH        -0.057       0.036      -1.584       0.116
RES          -0.001 2.442416E-4      -4.237 5.234667E-5
CONST         0.607       0.042      14.417 1.18732E-25
```

default		**solvelink=2**	
`real`	`27m12.745s`	`real`	`14m29.518s`
`user`	`20m58.595s`	`user`	`12m58.734s`
`sys`	`5m30.054s`	`sys`	`1m3.559s`

Table 3. Solvelink results

Here the p-value for FORPROF is indicating this parameter is *not* significant at the 0.05 level. The p-values are calculated using the incomplete beta function which is available as `BetaReg()` in GAMS[12].

It is noted that the option `m.solvelink=2;` is quite effective for this model. Timings that illustrate this are reported in table 3.

A further small performance improvement can be achieved to augment the model equations for the DEA model by the equations that calculate $(X^T X)^{-1}$. This will combine the DEA and OLS model into one model. After this has been done there is only one solve for each bootstrap sample.

6. Other DEA sources

We want to mention the work of [8] and [9] for large DEA models in conjunction with GAMS. The software is described on the web page `http://www.gams.com/contrib/gamsdea/dea.htm` [10].

Some earlier DEA modeling work with GAMS is documented in [14, 18].

References

1. R. D. Banker, A. Charnes, and W. W. Cooper, *Some models for estimating technical and scale efficiencies in data envelopment analysis*, Management Science **30** (1984), no. 9, 1078–1092.
2. William F. Bowlin, *Measuring performance: An introduction to data envelopment analysis (dea)*, Tech. report, Department of Accounting, University of Northern Iowa, Cedar Falls, IA, 1998.
3. A. Charnes, W. W. Cooper, and E. Rhodes, *Measuring the efficiency of decision making units*, European Journal of Operational Research **2** (1978), 429–444.
4. A. Charnes, W. W. Cooper, and E. Rhodes, *Evaluating program and managerial efficiency: An application of data envelopment analysis to program follow through*, Management Science **27** (1981), 668–697.
5. Laurens Cherchye, Timo Kuosmanen, and Thierry Post, *New tools for dealing with errors-in-variables in DEA*, Tech. report, Catholic University of Leuven, 2000.
6. Bradley Efron and Robert J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall, 1993.
7. Ali Emrouznejad, *Dea homepage*, `http://www.deazone.com/`, 2001.
8. Michael C. Ferris and Meta M. Voelker, *Slice models in general purpose modeling systems*, Tech. report, Computer Sciences Department, University of Wisconsin, 2000.
9. ———, *Cross-validation, support vector machines and slice models*, Tech. report, Computer Sciences Department, University of Wisconsin, 2001.
10. GAMS Development Corporation, *GAMS/DEA*, `http://www.gams.com/contrib/gamsdea/dea.htm`, 2001.
11. Dieter Haas, Martin G. Kocher, and Matthias Sutter, *Measuring Efficiency of German Football Teams by Data Envelopment Analysis*, Tech. report, University of Innsbruck, May 2003.
12. Erwin Kalvelagen, *New special functions in GAMS*, `http://amsterdamoptimization.com/pdf/specfun.pdf`.
13. ———, *Model building with gams*, to appear.
14. O.B. Olesen and N.C. Petersen, *A presentation of GAMS for DEA*, Computers & Operations Research **23** (1996), no. 4, 323–339.

15. Leopold Simar and Paul W. Wilson, *Sensitivity Analysis of Efficiency Scores: How to Bootstrap in Nonparametric Frontier Models*, Journal of Applied Statistics **44** (1998), no. 1, 49–61.
16. ______, *A general methodology for bootstrapping in nonparametric frontier models*, Journal of Applied Statistics **27** (2000), 779–802.
17. Boris Vujčić and Igor Jemrić, *Efficiency of banks in transition: A DEA approach*, Tech. report, Croatian National Bank, 2001.
18. John B. Walden and James E. Kirkley, *Measuring Technical Efficiency and Capacity in Fisheries by Data Envelopment Analysis Using the General Algebraic Modeling System (GAMS): A Workbook*, NOAA Technical Memorandum NMFS-NE-160, National Oceanic and Atmospheric Administration, National Marine Fisheries Service, Woods Hole Lab., 166 Water St., Woods Hole, MA 02543, 2001.
19. Mei Xue and Patrick T. Harker, *Overcoming the Inherent Dependency of DEA Efficiency Scores: A Bootstrap Approach*, Tech. report, Department of Operations and Information Management, The Wharton School, University of Pennsylvania, April 1999.

Amsterdam Optimization Modeling Group, Washington D.C./The Hague
E-mail address: erwin@amsterdamoptimization.com