Calcul et interprétation des scores d'efficience, programmation effective sous Scilab

Thierry Audibert

thierry.audibert@maths2b.fr

2 décembre 2011

Ce document est conçu pour ***accompagner des scripts Scilab*** *qui réalisent le calcul de scores d'efficience (au sens de Farrel) en orientation input ou output, la recherche de paramètres pour des modèles explicatifs dérivés du modèle linéaire, le calcul par simulation (bootstrap) d'intervalles de confiance et autres statistiques pour ces paramètres.*
Il est le fruit d'un travail effectué en collaboration avec une équipe du Cerdi (Unité Mixte de Recherche CNRS 3587 à l'Université d'Auvergne). On peut télécharger librement ces scripts sur le site de ***maths2b*** *à l'adresse*

http ://www.maths2b.fr/

Le mode d'emploi des scripts est décrit en (4), il est relativement bref mais devrait suffire dans la mesure où ceux-ci proposent une interface graphique dispensant de toute programmation. On peut le consulter directement.

La section (1) rappelle brièvement les techniques d'enveloppement des données, les modes de calcul et décrit les scripts Scilab qui les réalisent. Selon le même schéma problème-modes de calculs-programmation, les sections (2) et (3) abordent les aspects statistiques : estimations au maximum de vraisemblance puis statistiques bootstrap.

La section (5) traite des erreurs de calculs machine que l'on doit s'attendre à rencontrer et de la façon dont elles ont été gérées. Elle peut intéresser qui souhaite programmer des calculs analogues.

Ce travail ne prétend à rien d'original sur le fond ; il reprend des idées développées par L. Simar et P. W. Wilson dans [4] par exemple.

Table des matières

1 Scores d'efficience, enveloppement des données

Nous donnons le vocabulaire et les démonstrations nécessaires à la compréhension du problème et à sa programmation : notions de frontière de production, de scores d'efficience, lecture informatique et mise en forme (matricielle) des données en vue de leur traitement par les fonctions de résolution de programmes linéaires préprogrammées.

1.1 Production et efficience, présentation du problème et notations

• **L'ensemble de production admissible** pour une entreprise ou un ensemble d'entreprises d'un même domaine qui, pour des entrées $x \in \mathbb{R}^p_+$, sont susceptibles de produire des biens $y \in \mathbb{R}^q_+$, est l'ensemble $\mathcal{P}$ des couples (x, y) tels que x permet de produire y (c'est le graphe de la relation : $P(x, y)$ ssi x peut produire y).

On suppose que $\mathcal{P}$ vérifie les propriétés suivantes :

1. $\mathcal{P}$ est fermé et convexe ;
2. $\mathcal{P}$ ne contient aucun point $(0, y)$ avec $y > 0$;
3. si $(x, y) \in \mathcal{P}$, alors $y' \leq y \Rightarrow (x, y') \in \mathcal{P}$ et de façon analogue $x' \geq x \Rightarrow (x, y') \in \mathcal{P}$;

• Pour un couple $(x_0, y_0) \in \mathcal{P}$, on envisagera deux façons de mesurer son **efficience technologique.** Soit en cherchant à minimiser les entrées :

$$\Theta(x_0, y_0) = \inf\{\theta > 0;\ (\theta x_0, y_0) \in \mathcal{P}\} \tag{1.1}$$

Soit en cherchant à maximiser les sorties :

$$\delta(x_0, y_0) = \sup\{\theta > 0;\ (x_0, \theta y_0) \in \mathcal{P}\} \tag{1.2}$$

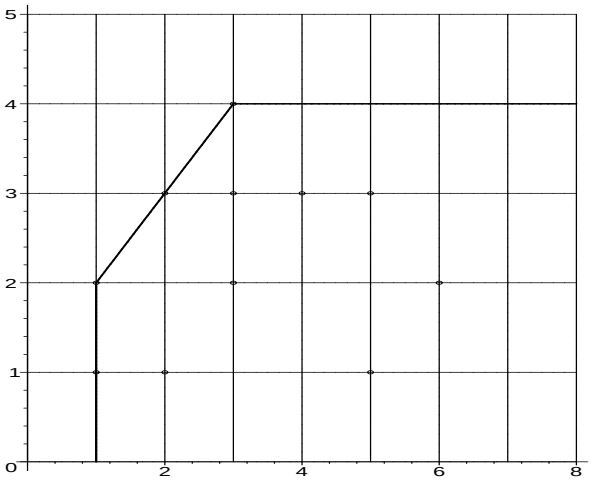
On aura donc dans le premier cas $\Theta(x_0, y_0) \leq 1$ et la valeur 1 est obtenue sur la frontière de $\mathcal{P}$ où les entreprises sont efficientes ; dans le second cas $\delta(x_0, y_0) \geq 1$, la valeur 1 étant obtenue sur la frontière également. Ce sont là les mesures d'efficience de Farrell (orientées entrées et sorties respectivement [1]).

• En pratique on dispose **d'un nombre fini de données** $(x_i, y_i) \in \mathbb{R}^p_+ \times \mathbb{R}^q_+$ à partir desquelles on construit un ensemble de production observé (empirique), $\hat{\mathcal{P}}$, comme étant le plus petit ensemble de $\mathbb{R}^p_+ \times \mathbb{R}^q_+$ tel que

1. $\hat{\mathcal{P}}$ contient les observations (x_i, y_i);
2. $\hat{\mathcal{P}}$ est convexe et fermé ;
3. $\hat{\mathcal{P}}$ ne contient aucun point $(0, y)$ avec $y > 0$;
4. si $(x, y) \in \hat{\mathcal{P}}$, alors $y' \leq y \Rightarrow (x, y') \in \hat{\mathcal{P}}$ et $x' \geq x \Rightarrow (x', y) \in \hat{\mathcal{P}}$;

On observera que, puisque les (x_i, y_i) appartiennent à $\mathcal{P}$, $\hat{\mathcal{P}} \subset \mathcal{P}$.

1. nous dirons inputs et outputs par la suite



• On définit tout naturellement des **scores d'efficience empiriques :**

$$\hat{\Theta}(x_0, y_0) = \inf\{\theta > 0;\ (\theta x_0, y_0) \in \hat{\mathcal{P}}\} \leq 1 \tag{1.3}$$

$$\hat{\delta}(x_0, y_0) = \sup\{\theta > 0;\ (x_0, \theta y_0) \in \hat{\mathcal{P}}\} \geq 1 \tag{1.4}$$

En vue de calculer ces scores, donnons une **caractérisation géométrique de** $\hat{\mathcal{P}}$, qui nous permettra de les exprimer comme solutions d'un problème d'optimisation linéaire.

Notons n le nombre d'observations, X et Y les familles de vecteurs $X = (x_1, ..., x_n)$ et $Y = (y_1, ..., y_n)$ représentants les entrées et sorties observées (on pourra considérer X et Y comme des matrices de $\mathcal{M}_{p,n}(\mathbb{R})$ et $\mathcal{M}_{q,n}(\mathbb{R})$ respectivement).

Théorème 1
Un couple (x_0, y_0) appartient à $\hat{\mathcal{P}}$ ssi il existe un vecteur $w \in [0, 1]^n$ tel que

$$\begin{cases} \sum_i w_i = 1; \\ x_0 \geq X.w; \\ y_0 \leq Y.w; \end{cases} \tag{1.5}$$

Démonstration :
Notons $\mathcal{L}$ l'ensemble des points qui vérifient les relations (1.5) et montrons que $\mathcal{L} \subset \hat{\mathcal{P}}$. Supposons que $(x_0, y_0) \in \mathcal{L}$, puisque $w_i \geq 1$ et $\sum_i w_i = 1$, $(X.w, Y.w)$ est barycentre des points (x_i, y_i) affectés de coefficients w_i positifs et appartient à l'enveloppe convexe des

(x_i, y_i).[2] Celle-ci est contenue dans $\hat{\mathcal{P}}$; ainsi, $(X.w, Y.w) \in \hat{\mathcal{P}}$ de même que (x_0, y_0) par la propriété 4.

$\supseteq$ Pour établir la réciproque, montrons que $\mathcal{L}$ vérifie les propriétés (1) à (4). Comme $\hat{\mathcal{P}}$ est le plus petit ensemble qui vérifie ces propriétés, on aura prouvé $\hat{\mathcal{P}} = \mathcal{L}$.

1. Chaque (x_i, y_i) est dans $\mathcal{L}$ car $(x_i, y_i) = (Xw, Yw)$ avec $w =^t (0, ..., 1, ..., 0)$ et $x_i \geq Xw$, $y_i \leq Yw$;
2. $\mathcal{L}$ est convexe : si $(x, y), (x', y')$ sont dans $\mathcal{L}$, avec $x \geq Xw, y \leq Yw$ et $x' \geq Xw', y' \leq Yw'$ nous aurons pour $w" = tw + (1-t)w'$ $t \in [0, 1]$,

$$tx + (1-t)x' \geq tXw + (1-t)Xw' = X(tw + (1-t)w' = Xw"$$

$$ty + (1-t)y' \leq tYw + (1-t)Yw' = Y(tw + (1-t)w' = Yw".$$

   $\mathcal{L}$ est fermé comme intersection de fermés (défini par $p+q+1$ équations et inéquations larges) ;
3. $\mathcal{L} \subset \hat{\mathcal{P}}$ ne contient pas de point $(0, y)$ avec $y \geq 0$, non nul ;
4. la dernière propriété est obtenue par comparaisons.

1.2 Optimisation linéaire pour le calcul de $\hat{\delta}(x_0, y_0)$

La relation $(x_0, \theta y_0) \in \hat{\mathcal{P}}$ se traduit par :

il existe $w \in \mathbb{R}^n$ tel que $\begin{cases} \sum_{i=1}^n w_i & = 1 \\ X.w & \leq x_0 \\ Y.w & \geq \theta y_0 \end{cases}$ avec $\theta > 0$ et pour tout $i \in [1, n]$, $w_i \geq 0$.

Le réel $\hat{\delta}(x_0, y_0)$ est donc solution du **programme linéaire** suivant que nous écrivons ici sous **forme canonique** :

Rechercher $(\theta, w_1, ..., w_n) \in \mathbb{R}^{n+1}$ tel que $\begin{cases} \sum_{i=1}^n w_i & \leq 1 \\ -\sum_{i=1}^n w_i & \leq -1 \\ X.w & \leq x_0 \\ \theta y_0 - Y.w & \leq 0 \\ \theta & \geq 0 \\ w & \geq 0 \\ \text{maximiser} & \theta \end{cases}$

2. C'est par définition le plus petit convexe contenant ces points ; c'est aussi l'ensemble des barycentres de ces points affectés de coefficients positifs. Dans notre cas, ces barycentres sont de la forme $(X.w, Y.w)$).

• En pratique nous écrirons ce programme d'optimisation linéaire en notation matricielle

$$AZ = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} \\ 0 & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} \\ 0 & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} & x_{3,6} \\ y_{0\,1} & -y_{1,1} & -y_{1,2} & -y_{1,3} & -y_{1,4} & -y_{1,5} & -y_{1,6} \\ y_{0\,2} & -y_{2,1} & -y_{2,2} & -y_{2,3} & -y_{2,4} & -y_{2,5} & -y_{2,6} \\ y_{0\,3} & -y_{3,1} & -y_{3,2} & -y_{3,3} & -y_{3,4} & -y_{3,5} & -y_{3,6} \\ y_{0\,4} & -y_{4,1} & -y_{4,2} & -y_{4,3} & -y_{4,4} & -y_{4,5} & -y_{4,6} \\ y_{0\,5} & -y_{5,1} & -y_{5,2} & -y_{5,3} & -y_{5,4} & -y_{5,5} & -y_{5,6} \end{bmatrix} \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} \leq \begin{bmatrix} 1 \\ -1 \\ x_{0\,1} \\ x_{0\,2} \\ x_{0\,3} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{1.6}$$

Rappel : pour un programme linéaire,

la forme canonique est $\begin{cases} Az \leq b \\ z \geq 0 \\ \text{Maximiser } (c|z) \end{cases}$

la forme standard [3] est $\begin{cases} Az = b \\ z \geq 0 \\ \text{Maximiser } (c|z) \end{cases}$ ou encore $\begin{cases} [A|I_m] \begin{bmatrix} z_N \\ z_B \end{bmatrix} = b \\ z \geq 0 \\ \text{Maximiser } (c|z) = (c_N|z_N) \end{cases}$

1.3 Optimisation linéaire pour le calcul de $\hat{\Theta}(x_0, y_0)$

La relation $(\theta x_0, y_0) \in \hat{\mathcal{P}}$ se traduit par : il existe $w \in \mathbb{R}^n$ tel que $\begin{cases} \sum_{i=1}^n w_i & = 1 \\ X.w & \leq \theta x_0 \\ Y.w & \geq y_0 \end{cases}$ avec $\theta > 0$ et pour tout $i \in [1, n]$, $w_i \geq 0$. $\hat{\Theta}(x_0, y_0)$ est solution du problème linéaire :

Rechercher $(\theta, w_1, ..., w_n) \in \mathbb{R}^{n+1}$ tel que $\begin{cases} \sum_{i=1}^n w_i & \leq 1 \\ -\sum_{i=1}^n w_i & \leq -1 \\ -\theta x_0 + X.w & \leq 0 \\ -Y.w & \leq -y_0 \\ \theta & \geq 0 \\ w & \geq 0 \\ \text{minimiser} & \theta \end{cases}$

3. On se ramène à la forme standard pour la résolution effective, mais cela reste en boîte noire dans les logiciels.

• Soit en notation matricielle :

$$AZ = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ -x_{01} & x_{1,1} & x_{1,2} & x_{1,3} & x_{1,4} & x_{1,5} & x_{1,6} \\ -x_{02} & x_{2,1} & x_{2,2} & x_{2,3} & x_{2,4} & x_{2,5} & x_{2,6} \\ -x_{03} & x_{3,1} & x_{3,2} & x_{3,3} & x_{3,4} & x_{3,5} & x_{3,6} \\ 0 & -y_{1,1} & -y_{1,2} & -y_{1,3} & -y_{1,4} & -y_{1,5} & -y_{1,6} \\ 0 & -y_{2,1} & -y_{2,2} & -y_{2,3} & -y_{2,4} & -y_{2,5} & -y_{2,6} \\ 0 & -y_{3,1} & -y_{3,2} & -y_{3,3} & -y_{3,4} & -y_{3,5} & -y_{3,6} \\ 0 & -y_{4,1} & -y_{4,2} & -y_{4,3} & -y_{4,4} & -y_{4,5} & -y_{4,6} \\ 0 & -y_{5,1} & -y_{5,2} & -y_{5,3} & -y_{5,4} & -y_{5,5} & -y_{5,6} \end{bmatrix} \begin{bmatrix} \theta \\ w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} \leq \begin{bmatrix} 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ -y_{01} \\ -y_{02} \\ -y_{03} \\ -y_{04} \\ -y_{05} \end{bmatrix} \tag{1.7}$$

1.4 Annexe : Les calculs sous Scilab 5.3

1.4.1 La toolbox quapro

Scilab dispose d'une boîte à outils **quapro** permettant la résolution de problèmes d'optimisation. La fonction **linpro** de cette bibliothèque permet de résoudre les programmes linéaires. En voici la syntaxe, avec les seuls éléments que nous utiliserons par la suite (on pourra comparer à la page d'aide complète, ici expurgée) :

Syntaxe de l'appel	**Description** (non traduite)
[x,lagr,f]=linpro(p,C,b)	Minimize p'*x under the constraints $C * x \leq b$
[x,lagr,f]=linpro(p,C,b,ci,cs)	Minimize p'*x under the constraints $C * x <= b, ci \leq x \leq cs$
[x,lagr,f]=linpro(p,C,b,ci,cs,me)	Minimize p'*x under the constraints $C(j,:)x = b(j), j = 1, ..., me$ $C(j,:)x \leq b(j), j = me+1, ..., m$ $ci \leq x \leq cs$
Paramètres	**p** real column vector (dimension n) **C** real matrix (dimension m x n) (If no constraints are given, you can set C = []) **b** RHS column vector (dimension m) (If no constraints are given, you can set b = []) **ci** column vector of lower-bounds (dimension n). If there are no lower bound constraints, put ci = []. If some components of x are bounded from below, set the other (unconstrained) values of ci to a very large negative number (e.g. ci(j) = -number_properties('huge'). **cs** column vector of upper-bounds. Same remarks as above. **me** number of equality constraints (i.e. C(1 :me, :)*x = b(1 :me)) ... **x** optimal solution found. **f** optimal value of the cost function (i.e. f=p'*x). **lagr** vector of Lagrange multipliers...

Nous voyons que pour résoudre le système (1.6) nous pouvons appeler **linpro** sous la forme suivante :

```
[Z, lagr, f] = linpro(-c,A,b,[zeros(n+1,1)],[]);
```

dans laquelle -c est le vecteur donnant la fonction à minimiser (dans notre cas $c =^t [1, 0, ...0]$ puisque nous voulons minimiser $(-c|Z) = -\theta$), A est la matrice des contraintes, b est le second membre de (1.6), les vecteurs `[zeros(n+1,1)],[]` (nommés ci et cs dans l'aide)

servent respectivement à exprimer les contraintes $Z_1 = \theta \geq 0, Z_2 = w_1, ..., Z_{n+1} = w_n \geq 0$, et l'absence de contraintes de majoration.
On récupère donc $\hat{\delta}(x_0, y0) = -f$ (la valeur du minimum de $(-c|Z) = -\theta$) sous contraintes) ou encore $\hat{\delta}(x_0, y0) = Z(1,1) = \theta$ en un maximum de $(c|Z)$.

Qu'avions nous à programmer ? Quelles fonctions avons nous écrites ?

1. **Le calcul des scores d'efficience pour X et Y donnés**
   La mise en forme des matrices A, b, c lorsque les données $(x_i, y_i) \in \mathbb{R}^p \times \mathbb{R}^q$ sont déjà stockées dans des matrices X et Y, pour un $(x_0, y_0) = (x_k, y_k)$ parmi les données est l'objet de la fonction `[A,b,c] = DeaDeltaEquationsCan(X,Y,k)` . Pour le calcul effectif de tous les scores d'efficience, `delta = DeaDeltaScoresCan(X,Y)` retourne le vecteur des n scores.

   Toutes ces fonctions sont regroupées dans le script `DEA.sci`

2. **Des fonctions de lecture des fichiers de données**

   On a choisi de lire les données dans un fichier csv (fichier texte dont les lignes représentent les lignes d'un tableau, les champs de chaque ligne sont séparés par un symbole -séparateur- qui peut être ',' ou ';' ou ' :' ou une tabulation etc... ; le choix du séparateur est proposé en lecture et écriture sous OpenOffice ou excel par exemple quand on sauve ou lit sous ce format). Il faut éviter ',' comme séparateur si le symbole décimal lui-même est une virgule.

   La fonction `[B,val,isN]= lectureNbresCsv(nom, sep)` prend en argument le **nom** d'un fichier csv (nom en est le chemin complet), le **sép**arateur et retourne trois matrices de la taille du tableau (m lignes, n colonnes) qui sont :
   - B matrices des champs sous forme de chaînes de caractères (string) ;
   - val contient les nombres du tableau avec des 0 là où les champs B(i,j) ne prennent pas de valeurs numériques ;
   - isN est une matrice de booléens qui contient %t =true ou %f =false selon que B(i,j) est numérique ou pas.

   C'est lourd en mémoire, mais simple à gérer.
   la fonction `C = convertir(M)` prend en argument une matrice M de nombres et retourne une matrice C de chaînes (en vue de stocker au format csv (donc texte) les données de M).

   Ces fonctions sont regroupées dans le script `GestionCsv.sci`

3. La mise en place des matrices X et Y avec une grille de lecture des variables input et output est proposée avec une interface graphique dans les scripts `DEAOutputGui.sce` et `DEAInputGuiW4.sce` décrits en (4.1). Le code n'est pas détaillé ici, son contenu est d'ordre purement pratique.

1.4.2 DEA orienté output, orienté input

Composantes matricielles et résolution du problème linéaire (1.6) pour le calcul des $\hat{\delta}(x_i, y_i)$
Appel avec `DeaDeltaScoresCan(X,Y)` qui utilise `linpro` .

```
 function [A,b,c]= DeaDeltaEquationsCan(X,Y,k)
 [p,n]  = size(X);
 [q,n1] = size(Y);
 if n ~= n1 then
    error("les deux premiers arguments n ont
                pas le même nombre de colonnes");
end
 if k > n then
   error("le troisième argument k, est supérieur
              au nombre de colonnes de X et Y ");
 end;
 // ---  prélevons les colonnes k de X et de Y
 x0 =  X(1:p,k:k);
 y0 =  Y(1:q,k:k);
 // --- la matrice A en quatre étapes et huit blocs
 L1 = [0,ones(1:n)];
 L2 = [0,-ones(1:n)];
 LX = [zeros(p,1), X];
 LY = [y0, -Y];
 A  = [L1; L2; LX; LY];
 //--- no comment
 b  = [1; -1; x0; zeros(q,1)];
 c  = [1;zeros(n,1)];
 endfunction;
// ---------------------------------------------------------------------
function  delta = DeaDeltaScoresCan(X,Y)
 [p,n]  = size(X);
 [q,n1] = size(Y);
 delta1 = zeros(n,1);
 for k=1:n
    [A,b,c]        = DeaDeltaEquationsCan(X,Y,k);
    [Z, lagr1, f1] = linpro(-c,A,b,[zeros(n+1,1)],[]);
    delta1(k,1)    = -f1;
 end;
 delta =delta1;
 endfunction;
```

Un exemple (voir la figure plus loin) :

```
-->  X = [1,2,5,1,3,6,2,3,4,5,3 ];
-->  Y = [1,1,1,2,2,2,3,3,3,3,4 ];
-->  [p ,n] = size(X);
-->       q = size(Y,1);
--> [A,b,c] = DeaDeltaEquationsCan(X,Y,5);
-->  A,b,c
 A  =
    0.    1.    1.    1.    1.    1.    1.    1.    1.    1.    1.
    0.  - 1.  - 1.  - 1.  - 1.  - 1.  - 1.  - 1.  - 1.  - 1.  - 1.
    0.    1.    2.    5.    1.    3.    6.    2.    3.    4.    5.    3.
    2.  - 1.  - 1.  - 1.  - 2.  - 2.  - 2.  - 3.  - 3.  - 3.  - 3.  - 4.
 b  =
    1.
  - 1.
    3.
    0.
 c  =
    1.
    0.
    0.
    0.
    0.
    0.
    0.
    0.
    0.
    0.
    0.
    0.
--> delta1  = DeaDeltaScoresCan(X,Y)
--> delta1  =
    2.
    3.
    4.
    1.
    2.
    2.
    1.
    1.3333333
    1.3333333
    1.3333333
    1.
```

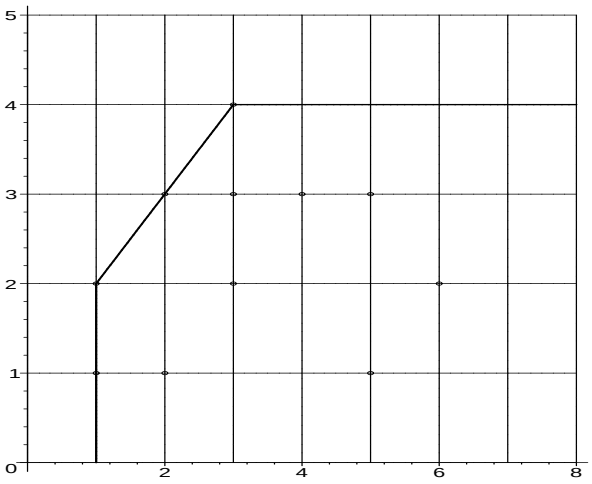
Composantes matricielles et résolution du problème linéaire (1.7) pour le calcul des $\hat{\Theta}(x_i, y_i)$
Appel avec `DeaThetaScoresCan(X,Y)` qui utilise `linpro` .

```
function [A,b,c]= DeaThetaEquationsCan(X,Y,k)
  [p,n]  = size(X);
  [q,n1] = size(Y);

  if n ~= n1 then
    error("les deux premiers arguments n ont pas le même nombre de colonnes");
  end
  if k > n then
    error("le troisième argument k, est supérieur au nombre de colonnes de X et Y ");
  end

  // ---  prélevons les colonnes k de X et de Y
  x0 =  X(1:p,k:k);
  y0 =  Y(1:q,k:k);

 // --- la matrice A en quatre étapes et huit blocs
 L1 = [0,ones(1:n)];
 L2 = [0,-ones(1:n)];
 LX = [-x0, X];
 LY = [zeros(q,1), -Y];
 A  = [L1; L2; LX; LY];
 //--- no comment
 b  = [1; -1; zeros(p,1);-y0];
 c  = [1;zeros(n,1)];
endfunction;

//---------------------------------------------------------------------------

function  delta = DeaThetaScoresCan(X,Y)
  [p,n]  = size(X);
  [q,n1] = size(Y);

  delta1 = zeros(n,1);
  for k=1:n
      [A,b,c]        = DeaThetaEquationsCan(X,Y,k);
      [Z, lagr1, f1] = linpro(c,A,b,[zeros(n+1,1)],[]);
      delta1(k,1)    = f1;
  end;
  delta =delta1;
 endfunction;
```

Toujours le même exemple :

```
-->theta2  = DeaThetaScoresCan(X,Y)
 theta2  =

    1.
    0.5
    0.2
    1.
    0.3333333
    0.1666667
    1.
    0.6666667
    0.5
    0.4
    1.
```

On vérifie sans peine sur la figure les deux calculs de scores (delta1 et theta2) ; on numérotera de gauche à droite de bas en haut :



Pour lire les tableaux des fichiers csv

```
function [B,val,isN]= lectureNbresCsv(nom, sep)
if sep ==',' then
         error ('le séparateur doit être différent d une virgule');
end
[fileId ,err]=  mopen(nom ,'r');
if err <> 0 then
   disp('Le fichier ' + nom + 'n a pas été lu; erreur '+ string(err));
 else
   B =read_csv(nom , sep);
   [m,n] =size(B);
   val =zeros(m,n);
   for i=1:m
      for j=1:n
          s = B(i,j);
          s = strsubst(s,ascii(34),"")
          s = strsubst(s,',','.');
          if isnum(s)  then
               val(i,j)= strtod(s); isN(i,j) =%t;
             else
               isN(i,j) =%f;
           end;
       end;
   end;
   mclose(fileId);
 end;
endfunction
//---------------------------------------------------------------------
function  C = convertirCsv(M)
[m,n] = size(M);
 for i=1:m
    for j=1:n
         C(i,j) =sci2exp(M(i,j));
    end;
 end
endfunction
```

2 Variables explicatives, estimation des paramètres

Ayant observé n scores d'efficience calculés en **orientation outputs** (les scores $(\hat{\delta}_i)_{1\leq i\leq n}$) ou en **orientation inputs** (les scores $(\hat{\theta}_i)_{1\leq i\leq n}$) et n variables d'environnement $(z_i)_{1\leq i\leq n}$ (avec $z_i = (z_1^{(i)}, z_2^{(i)}, ..., z_r^{(i)}) \in \mathbb{R}^r$ pour tout i), on souhaite associer les performances relatives δ ou θ aux variables d'influence $(z_1, z_2, ...z_n)$, par une relation du type $\delta(x, y) = \psi(z, \epsilon)$ ou $\theta(x, y) = \psi(z, \epsilon)$, où ϵ est une variable aléatoire...

C'est là que différents choix sont possibles, dans l'idéal et autant que faire se peut, inspirés par l'observation ou une modélisation des phénomènes étudiés.
Nous proposons des outils de calcul pour des modèles classiques avec :
- En (2.1), un modèle $\delta = \beta z + \epsilon = \max(1, \beta z + u) \in [1, +\infty[$ pour les scores orientés outputs.
- En (2.2), un modèle $\Theta = \beta z + \epsilon = \max(\min(\beta\, z + u, 1), 0) \in]0, 1]$ pour les scores orientés inputs.
- Enfin, toujours pour les scores orientés inputs, un modèle $\ln(1/\Theta) \in [0, +\infty[$.

En pratique le choix des variables explicatives est proposé en même temps que celui des variables X et Y représentant les entrées/inputs et les sorties/outputs avec une interface graphique dans les scripts `DEAOutputGui.sce` et `DEAOutputGuiW4.sce`.

2.1 Scores orientés outputs le modèle $\delta = \beta z + \epsilon = \max(1, \beta z + u)$

2.1.1 Fonction de répartition, probabilités conditionnelles

• **Le modèle** $\delta = \beta z + \epsilon = \max(1, \beta z + u)$
On souhaite exprimer δ en fonction des variables d'influence $(z_1, z_2, ...z_n)$, par une relation du type $\delta(x, y) = \sum_i \beta_k z_k + \epsilon$ (ce qui inclut $\delta(x, y) = \beta_0 + \sum_i \beta_k z_k + \epsilon$ une composante, z_r par exemple, constante égale à 1).
On fait l'hypothèse supplémentaire que $\boxed{\delta = \beta z + \epsilon = \max(1, \beta z + u)}$ où u est une variable aléatoire de loi normale de moyenne 0 et variance σ^2 (rappelons que $\delta, \hat{\delta} \geq 1$).[4]
Dans le but d'estimer les $r + 1$ paramètres β_j et σ, déterminons la loi de δ. Pour cela notons $f(t) = \dfrac{1}{\sqrt{2\pi}}\, e^{\dfrac{-t^2}{2}}$ la densité de la loi normale (centrée réduite) et F sa fonction de répartition et exprimons la fonction de répartition de δ : $\Phi(t) = P_{z,\beta,\sigma}(\delta \leq t)$.

$$\begin{cases} P_{z,\beta,\sigma}(\delta \leq t) = & 0 \text{ si t } < 1; \\ P_{z,\beta,\sigma}(\delta \leq 1) = & P_{z,\beta,\sigma}(u \leq 1 - \beta z); \\ P_{z,\beta,\sigma}(\delta \leq t) = & P_{z,\beta,\sigma}(u \leq 1 - \beta z) + P_{z,\beta,\sigma}(1 - \beta z < u \leq t - \beta z) \text{ si } t > 1. \end{cases}$$

Comme $\dfrac{u}{\sigma}$ suit la loi normale centrée réduite, on réécrit :

$$\begin{cases} P_{z,\beta,\sigma}(\delta \leq t) = & 0 \text{ si t } < 1; \\ P_{z,\beta,\sigma}(\delta \leq t) = & F\left(\dfrac{t - \beta z}{\sigma}\right) \text{ si } t \geq 1; \end{cases}$$

Remarques :

4. C'est ici que l'on parle de modèle censuré à gauche (du point 1)

1. Cette fonction de répartition est discontinue en $t = 1$. Nous en tiendrons compte pour déterminer les fonctions de vraisemblance.
2. La mesure de probabilité $P_{\beta,\sigma}$ est somme d'une mesure atomique/discrète de support $\{1\}$, de masse $F\left(\frac{1-\beta z}{\sigma}\right)$, et de la mesure absolument continue (par rapport à la mesure de Lebesgue) de densité

$$\frac{d}{dt}F\left(\frac{t-\beta z}{\sigma}\right) \times \chi_{(1,+\infty[}(t) = \frac{1}{\sigma}f\left(\frac{t-\beta z}{\sigma}\right) \times \chi_{(1,+\infty[}(t). \tag{2.1}$$

• Probabilités conditionnelles.

Pour $t > 1$, la probabilité conditionnelle d'observer $(\delta \leq t)$ sachant que $(\delta > 1)$ est

$$P\left(\delta \leq t | \delta > 1\right) = \frac{P\left(\delta \leq t \cap \delta > 1\right)}{P\left(\delta > 1\right)} = \frac{P\left(1 < \beta z + u \leq t\right)}{P\left(\beta z + u > 1\right)}$$

Ce qui s'écrit encore :

$$P\left(\delta \leq t | \delta > 1\right) = \frac{P\left(\frac{1-\beta z}{\sigma} < \frac{u}{\sigma} \leq \frac{t-\beta z}{\sigma}\right)}{P\left(\frac{u}{\sigma} > \frac{1-\beta z}{\sigma}\right)} = \frac{F\left(\frac{t-\beta z}{\sigma}\right) - F\left(\frac{1-\beta z}{\sigma}\right)}{1 - F\left(\frac{1-\beta z}{\sigma}\right)} \tag{2.2}$$

Cette loi conditionnelle a pour densité la fonction continue par morceaux, nulle sur $]-\infty, 1]$, et, pour $t > 1$, égale à

$$\phi(t) = \frac{1}{1 - F\left(\frac{1-\beta z}{\sigma}\right)} \times \frac{1}{\sigma}F'\left(\frac{t-\beta z}{\sigma}\right) \tag{2.3}$$

2.1.2 Comment choisir une fonction de vraisemblance ?

Nous disposons d'un échantillon de n observations

$$\hat{\delta}_i = \beta z^{(i)} + \epsilon_i = \sum_{k=1}^{r} \beta_k z_k^{(i)} + \epsilon_i = \max\left(1, \sum_{k=1}^{r} \beta_k z_k^{(i)} + u_i\right)$$

La loi conditionnelle de δ sachant $\delta > 1$, est (à densité) continue sur $]1, +\infty[$; nous pouvons choisir comme fonction de vraisemblance le produit des $\phi(\hat{\delta}_i)$ sur l'ensemble des observations correspondantes $(\hat{\delta}_i > 1)$. Les simulations conduisent à choisir les estimations données par cette fonction définie en (2.4) plutôt que celles obtenues avec un modèle du type Tobit (2.9).

2.1.3 Fonction de vraisemblance associée à la loi conditionnelle

Nous pouvons choisir comme estimations $\hat{\beta}$ et $\hat{\sigma}$ des paramètres β et σ, les valeurs rendant maximale la fonction de vraisemblance (qui est associée à la loi conditionnelle ayant une densité) :

$$V_1(\hat{\delta}, z, \beta, \sigma) = \prod_{\substack{i=1 \\ \hat{\delta}_i > 1}}^{n} \frac{1}{1 - F\left(\frac{1-\beta z^{(i)}}{\sigma}\right)} \times \frac{1}{\sigma}F'\left(\frac{\hat{\delta}_i - \beta z^{(i)}}{\sigma}\right) \tag{2.4}$$

dont le maximum est aussi celui de

$$\ln V_1(\hat{\delta}, z, \beta, \sigma) = -\sum_{\substack{i=1 \\ \hat{\delta}_i>1}}^{n} \left[\ln\left(1 - F\left(\frac{1-\beta z^{(i)}}{\sigma}\right)\right) + \frac{1}{2\sigma^2}\left(\hat{\delta}_i - \beta z^{(i)}\right)^2 \right] - m\ln\sigma + \text{cste} \tag{2.5}$$

(avec $m = \text{Card}(\{i;\ \hat{\delta}_i > 1\})$.

• Le gradient de la fonction $(\beta, \sigma) \in \mathbb{R}^r \times \mathbb{R}^{*+} \to \ln V_1(\hat{\delta}, z, \beta, \sigma)$ nous sera utile pour l'optimisation. Notons $u_i = \dfrac{1-\beta z^{(i)}}{\sigma}$ et $t_i = \dfrac{\hat{\delta}_i - \beta z^{(i)}}{\sigma}$, ce qui donne

$$\ln V_1(\hat{\delta}, z, \beta, \sigma) = -\sum_{\substack{i=1 \\ \hat{\delta}_i>1}}^{n} \left(\ln\left(1 - F\left(u_i\right)\right) + \frac{1}{2}t_i^2 \right) - m\ln\sigma + \text{cste} \tag{2.6}$$

et facilite l'écriture des dérivées partielles par rapport aux paramètres :

$$\frac{\partial u_i}{\partial \beta_k} = \frac{\partial t_i}{\partial \beta_k} = \frac{-z_k^{(i)}}{\sigma}, \quad \frac{\partial u_i}{\partial \sigma} = \frac{-u_i}{\sigma}, \quad \frac{\partial t_i}{\partial \sigma} = \frac{-t_i}{\sigma}$$

$$\frac{\partial \ln V_1}{\partial \beta_k} = \frac{1}{\sigma}\sum_{i=1}^{n} z_k^{(i)} \left(\frac{-F'(u_i)}{1-F(u_i)} + t_i \right) \tag{2.7}$$

$$\frac{\partial \ln V_1}{\partial \sigma} = \frac{1}{\sigma}\sum_{i=1}^{n} \left(\frac{-u_i\, F'(u_i)}{1-F(u_i)} + t_i^2 \right) - \frac{m}{\sigma} \tag{2.8}$$

2.1.4 Fonction de vraisemblance non conditionnelle (modèle Tobit usuel)

Un autre choix est possible pour la fonction de vraisemblance, tenant compte des masses ponctuelles de la loi de probabilité non conditionnelle (c'est d'ailleurs le choix courant pour le modèle Tobit). Après comparaison des résultats obtenus avec la fonction de vraisemblance précédente sur des données simulées, les estimations semblent meilleures avec $\ln V_1$ que nous avons donc retenue pour les calculs. Décrivons donc $\ln V_2$ pour mémoire.
On choisit ici comme estimations $\hat{\beta}$ et $\hat{\sigma}$ des paramètres β et σ, les valeurs rendant maximale la fonction de vraisemblance définie par [5] :

$$V_2(\hat{\delta}, z, \beta, \sigma) = \prod_{\substack{i=1 \\ \hat{\delta}_i=1}}^{n} F\left(\frac{1-\beta z^{(i)}}{\sigma}\right) \times \prod_{\substack{i=1 \\ \hat{\delta}_i>1}}^{n} \frac{1}{\sigma} f\left(\frac{\hat{\delta}_i - \beta z^{(i)}}{\sigma}\right) \tag{2.9}$$

En notant $m = \text{Card}(\{i;\ \hat{\delta}_i > 1\})$, ce maximum est aussi celui de

---

$$\ln V_2(\hat{\delta}, z, \beta, \sigma) = \sum_{\substack{i=1 \\ \hat{\delta}_i=1}}^{n} \ln F\left(\frac{1-\beta z^{(i)}}{\sigma}\right) - \frac{1}{2\sigma^2}\sum_{\substack{i=1 \\ \hat{\delta}_i>1}}^{n} \left(\hat{\delta}_i - \beta z^{(i)}\right)^2 - m\ln\sigma + \text{cste} \tag{2.10}$$

5. C'est le produit des masses ponctuelles de la mesure discrète et des valeurs de la densité continue en les δ_i observés

Notation Si nous définissons/calculons $\omega_p(t)$ et sa dérivée par rapport à t en posant

$$\begin{cases} \omega_p(t) = & \ln F\,(t) \text{ si } p=1 \\ \\ \omega_p(t) = & \dfrac{-t^2}{2} \text{ si } p>1 \end{cases} \quad \begin{cases} \omega'_p(t) = & \dfrac{f(t)}{F\,(t)} = \dfrac{e^{-t^2/2}}{\int_{-\infty}^{t} e^{-s^2/2}\,ds} \text{ si } p=1 \\ \\ \omega'_p(t) = & -t \text{ si } p>1 \end{cases}$$

nous pourrons réécrire par commodité :

$$\ln V_2(\hat{\delta}, z, \beta, \sigma) = \sum_{i=1}^{n} \omega_{\hat{\delta}_i}\left(\frac{\hat{\delta}_i - \beta z^{(i)}}{\sigma}\right) - m\ln\sigma + \text{cste} \tag{2.11}$$

• Gradient de la fonction $(\beta, \sigma) \in \mathbb{R}^r \times \mathbb{R}^{*+} \to \ln V(\hat{\delta}, z, \beta, \sigma)$

Donnons donc les expressions des dérivées partielles par rapport aux paramètres :

$$\frac{\partial \ln V_2}{\partial \beta_k} = \frac{-1}{\sigma}\sum_{i=1}^{n} z_k^{(i)} \omega'_{\hat{\delta}_i}\left(\frac{\hat{\delta}_i - \beta z^{(i)}}{\sigma}\right) \tag{2.12}$$

$$\frac{\partial \ln V_2}{\partial \sigma} = \frac{-1}{\sigma}\sum_{i=1}^{n} \left(\frac{\hat{\delta}_i - \beta z^{(i)}}{\sigma}\right) \omega'_{\hat{\delta}_i}\left(\frac{\hat{\delta}_i - \beta z^{(i)}}{\sigma}\right) - \frac{m}{\sigma} \tag{2.13}$$

2.1.5 Paramètres réalisant le maximum de la fonction de vraisemblance

Nous disposons de n observations $\hat{\delta}_i \in [1, +\infty[$, $z^{(i)} = [z_1^{(i)}, ..., z_r^{(i)}]$ et nous recherchons les valeurs de $\beta =^t [\beta_1, ..., \beta_r]$ et $\sigma > 0$ réalisant le maximum de la fonction $\ln V_1(\hat{\delta}, z, \beta, \sigma)$ définie en (2.5) (qu'une comparaison avec les résultats obtenus pour la fonction $\ln V_2(\hat{\delta}, z, \beta, \sigma)$ définie en (2.10) nous ont conduits à retenir).

En vue de la programmation, nous noterons de la façon suivante les données statistiques observées et les inconnues (paramètres de la loi) :

– les scores $\hat{\delta} = \begin{bmatrix} \hat{\delta}_1 \\ \vdots \\ \hat{\delta}_n \end{bmatrix} \in \mathbb{R}^n$, les explicatives $z = \begin{bmatrix} z_{1,1} & \dots & z_{1,r} \\ \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,r} \end{bmatrix}$ (matrices notées `Delta` et `Zexpl` dans nos programmes)

– pour un jeu de n données observées nous souhaitons optimiser la fonction

$$F : x \to \ln V(\hat{\delta}, z, [x_1, ...x_r], x_{r+1})$$

dont la variable est $x = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_r \\ \sigma \end{bmatrix} \in \mathbb{R}^{r+1}$, notée `X`.

2.1.6 Programmation sous Scilab

De quelles méthodes, de quels outils disposons nous ?

• Les **algorithmes de type gradient ou de type quasi-Newton :**
Ces outils sont en place sous Scilab avec la fonction `optim` dont la syntaxe comprend des paramètres optionnels et qui retourne le minimum (ou **un** minimum local) de la fonction `costf`. La syntaxe la plus simple est du type `[f, opt] = optim(costf, x0, 'gc');` elle est décrite ci-dessous.

Appel	Description
`[f, opt] = optim(costf, x0, 'gc')`	`costf` est la fonction à optimiser (pour nous $\ln V_1$ par exemple) ; `x0` est un vecteur initial (`optim` construit une suite $x_{n+1} = G(x_n)$ qui converge vers `opt`) ; `'gc'` (pour gradient conjugué) que l'on peut remplacer par `'qn'` (pour quasi-newton) permet de choisir l'algorithme ; f est la valeur de la fonction, `opt` le point en lequel le minimum est atteint
`costf` : la syntaxe de cette fonction est imposée, elle doit être de la forme `[f,g,ind] = costf(x, ind)`	où `f` = F(x) est la valeur de la fonction F à minimiser ; `g` =grad(f)(x) est le gradient de F et `ind` un entier dont ne ne nous préoccupons pas ici (nous serons sans doute amenés à l'utiliser selon le comportement de l'algorithme (vitesse de convergence)

Pour notre problème, nous recherchons donc le **minimum** de

$$F : x = (\beta, \sigma) \in \mathbb{R}^r \times \mathbb{R}^{*+} \to -\ln V_1(\hat{\delta}, z, \beta, \sigma).$$

Pour ce faire nous proposons un code Scilab en (2.2.4). Ce script comprend :

Appel	Description
`val = Vrais1(Delta, Zexpl, Beta, sigma)`	qui retourne $F(\beta, \sigma)$
`Gval = GradVrais1(Delta, Zexpl, Beta, sigma)`	qui retourne grad$F(\beta, \sigma)$
`[Beta, sigma, f] = MdresCarres(Delta, Zexpl, B0)`	qui retourne le vecteur $\beta \in \mathbb{R}^r$ qui minimise la somme $\sum_{\delta_i>1}(\delta_i - \beta z^{(i)})^2$
`[Beta, sigma] = MaxVrais1(Delta, Zexpl, X0)`	C'est la fonction que l'utilisateur appelle avec Delta de taille $n \times 1$, Zexpl de taille $n \times r$, X0 de taille $(r+1) \times 1$ pour initialiser l'algorithme. Ses r premières coordonnées `B0`$\in \mathbb{R}^r$ pour initialiser une approximation des moindres carrés des $\hat{\beta}_i$ qui elle-même fournira un vecteur initial `x0` pour la fonction **optim**...
`[f,g,ind] = CostVrais(X,ind)`	cette fonction est définie comme **variable locale** dans `MaxVrais1(Delta, Zexpl, X0)` qui l'envoie en paramètre à la fonction `otpim` de Scilab.

On constate que les algorithmes ne convergent que si on leur fournit un vecteur x0 suffisamment proche du minimum. C'est pourquoi, l'algorithme est initialisé avec une approximation aux moindres carrés des $\hat{\beta}_i$ calculée sur le sous-échantillon des $\hat{\delta}_i > 1$, et une approximation de σ^2 égale à l'écart moyen des $(\delta_i - Bz_i)^2$.

Remarque : La méthode n'est plus robuste si la fonction à optimiser n'est pas convexe/concave ; Olsen [3] et T. Amemiya [**?**] ont prouvé la concavité dans le cadre de la log-vraisemblance du Tobit. **Propriété à vérifier pour notre fonction**.

La fonction de vraisemblance $\ln V_1$ décrite en (2.6) . En boîte noire dans les scripts.

```
function val = Vrais1(Delta, Zexpl, Beta, sigma)

[n,  p]  =  size(Delta);
[n1, r]  =  size(Zexpl);
[r1, q]  =  size(Beta);

if  n ~= n1  then
  error("Les arguments Delta et Zexpl n ont pas le même nombre de colonnes");
end;
if r1 ~= r then
  error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
end;
if sigma <= 0 then
  sigma = -sigma + 0.1;//error("sigma ne peut être négative dans  VraisLin1")
end;

S    = 0;
m    = 0;
for i=1:n
   if Delta(i,1) > 1 then
      betaZ     =   Zexpl(i,:)*Beta;
      t1        =   (Delta(i,1)-betaZ)/sigma;
      u1        =   (1-betaZ)/sigma;
      t2        =   t1^2/2;
      S         =   S  + t2  +log(1-cdfnor("PQ",u1,0,1));
      m         =   m+1;
    end;
end;
val  = -S - m*log(sigma);
endfunction;
```

Pour la ligne
`if sigma <= 0 then sigma = -sigma + 0.1;end;`
se reporter à la section (**??**) où l'on traite des erreurs de calcul.

le gradient de la fonction vraisemblance $\ln V_1$ donné par les formules (2.8) et(2.7).

```
function Gval = GradVrais1(Delta, Zexpl, Beta, sigma)

[n,  p]  =  size(Delta);
[n1, r]  =  size(Zexpl);
[r1, q]  =  size(Beta);

if  n ~= n1  then
  error("Les arguments Delta et Zexpl n ont pas le même nombre de colonnes");
end;
if r1 ~= r then
  error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
end;
if sigma <= 0 then
    sigma = - sigma + 0.01;
    //error("sigma ne peut être négative dans  VraisLin1")
end;

c     = 1/((2*%pi)^(1/2));
Gval  = zeros(r+1,1);
m     = 0;

for i=1:n
  if Delta(i,1) >1 then
    BetaZ = Zexpl(i,:)*Beta;
    t1    = (Delta(i,1)-BetaZ)/sigma;
    u1    = (1-BetaZ)/sigma;
    dw    =  -exp(-u1^2/2)/(1-cdfnor("PQ",u1,0,1))*c  ;
    m     =  m+1;

    for k=1:r
        Gval(k,1) =  Gval(k,1)+Zexpl(i,k)*(dw+t1);
    end;

    Gval(r+1,1) =  Gval(r+1,1) + t1^2-1 +u1*dw
  end;
end;

Gval         =   Gval/(sigma);
endfunction;
```

Les moindres carrés : le but est de déterminer une première approximation des β_i et de σ pour l'initialisation de la recherche du maximum de vraisemblance.
La fonction D2 définie localement calcule $D2(B) = \sum_{\delta_i>1}(\delta_i - Z * B)^2$.
On utilise la fonction `optim` de Scilab pour obtenir une évaluation aux moindres carrés des coefficients β_i, on détermine à partir de cela une approximation de l'écart type σ des $\delta_i - \beta z^{(i)}$ pour les observations $\delta_i > 1$.

```
 function [Beta, sigma, f] = MdresCarres(Delta, Zexpl, B0)
//---------------    verifiée   -------------
   function [d, gd, ind] = D2(B, ind)
     dbZ     =  Delta-Zexpl*B;
     for i=1:size(Delta,1)
       if Delta(i,1) <= 1 + 10^(-5) then dbZ(i,1)=0; end;
     end
     d       =  dbZ'*dbZ;
     gd      = -2*Zexpl'*dbZ;
   endfunction;

   //-------- estimation des Beta_i ----------

     [f, Beta]  =  optim(D2, B0,'gc');

   //-------- estimation de sigma ------------
     m         =  0;
     for i=1:size(Delta,1)
       if Delta(i,1) > 1 + 10^(-5) then m = m +1; end;
     end;
     sigma     =  (D2(Beta)/m)^(1/2);

 endfunction;
```

La fonction qui retourne les paramètres réalisant le maximum de vraisemblance

```
function [Beta, sigma] = MaxVrais1(Delta, Zexpl, X0,trunc)
   //-----------------------------------------------
   function [f,g,ind] = CostVrais(X,ind)
     [r1, p] =   size(X);
     Beta    =   X(1:(r1-1), 1);
     sigma   =   X(r1, 1);
     f       = - Vrais1(Delta, Zexpl, Beta, sigma, trunc)
     g       = - GradVrais1(Delta, Zexpl, Beta, sigma, trunc)
   endfunction;
   //-----------------------------------------------
  [n, u]    = size(Delta);
  [n1, r]   = size(Zexpl);
  [r1, u1]  = size(X0);

  if  n ~= n1  then
    error("Delta et Zexpl n ont pas le même nombre de colonnes");
  end;
  if  r ~= r1  then
    error("Zexpl et X0 ne sont pas compatibles (r col et r lignes)");
  end;

  //--- initialisons par une estimation aux moindres carrés (sur delta_i >1)

  [B0, s0]  =   MdresCarres(Delta, Zexpl, X0, trunc);

  M         =  10^10*ones(r+1,1);
  m         =  10^10*[-ones(r,1);0]
  [f, Xopt] =  optim(CostVrais,  'b', m, M , [B0;s0], 'gc');
  [p,q]     =  size(Xopt);
   Beta     =  Xopt(1:p-1,q);
   sigma    =  Xopt(p, q);

endfunction;
```

2.1.7 Simulations et vérifications

Dans l'intention d'évaluer les performances de ces méthodes nous construisons des fonctions

- `[L] = DataTestDeltaV1(Z, Beta, sigma))` qui construit un jeu de données aléatoires à partir des paramètres $Z \in \mathcal{M}_{n,r}(\mathbb{R})$, Beta $\in \mathbb{R}^r$, sigma > 0.
- `[sigma, sigmaTest, Beta, BetaTest] = LeTestV1(L)` qui permet d'effectuer des comparaisons entre les valeurs réelles et estimées des paramètres β_k et σ.

Construction d'un tableau de **données aléatoires** en vue de tester les méthodes :
`DataTestV1(Z, Beta, sigma)` prend en arguments, `Z` une matrice de variables explicatives de taille $n \times r$, `Beta` un vecteur de r paramètres et un réel `sigma` ; elle retourne une matrice `L` de $n+3$ lignes et $r+2$ colonnes comportant :

- en première ligne les noms des paramètres beta1,beta2,...,betar, sigma, un espace (il s'agit de faciliter la lecture) ;
- en deuxième ligne le rappel des valeurs de ces coefficients ;
- en troisième ligne les noms z1,z2,...,zr, u_i et delta ;
- les lignes suivantes contiennent le bloc matriciel `Z` inchangé suivi des valeurs aléatoires $(u_i)_{1\leq i\leq n}$ et des $(\delta_i)_{1\leq i\leq n}$ avec $\delta_i = \max(1, \beta z^{(i)} + u_i)$.

```
function [L] = DataTestV1(Z, Beta, sigma)
//-Beta: ligne de r coefficients, Z:  matrice de n lignes et r colonnes
[u,r]  = size(Beta);
[n,r1] = size(Z);
if r <> r1 then
   error('Dimensions incompatibles pour Z et Beta');
end
L1     = ["beta1"];
L3     = ["z1"];

for k=2:r
     L1 = [L1, "beta" + sci2exp(k)];
     L3 = [L3, "z"    + sci2exp(k)];
end;
L1     = [L1, "sigma", ""];
L3     = [L3, "u_i"  , "delta"];

L2     = [Beta , sigma, 1];
L       = [L1;  convertirCsv(L2); L3];
U       =  zeros(n,1);
delta   =  zeros(n,1);

for i=1:n
   U(i,1)      =  grand(1,1, 'nor', 0, sigma );
   BetaZ       =  Beta*Z(i,:)';
   delta(i,1)  =  max(1, BetaZ + U(i,1));
end
L    =  [L; convertirCsv( [Z, U, delta] )];
endfunction
```

La fonction qui effectue le test (récupération des données en texte, traitement et comparaison)
sigma et sigmaTest sont respectivement l'écart type empirique $\hat{\sigma}$ calculé sur l'échantillon et l'écart type choisi pour la simulation, $ds = |\hat{\sigma} - \sigma|$;
Beta et BetaTest sont de la même façon $\hat{\beta}$ et β, $db = ||\hat{\beta} - \beta||_2 = \sqrt{\sum_k (\hat{\beta}_k - \beta_k)^2}$;
N est la taille de l'échantillon, m le nombre de $\hat{\delta}_i > 1$;

```
function [sigma, sigmaTest, ds, Beta, BetaTest, db, m, N] = LeTestV1(L)

   [N1, r1]      =  size(L);
   r             =  r1-2;
   N             =  N1-3;

   BetaTest      =  strtod(L(2, 1:r))';
   sigmaTest     =  strtod(L(2,r+1));
   Zexpl         =  strtod(L(4:N+3, 1:r) );
   deltaTest     =  strtod(L(4:N+3, r+2));

   X0            =  [ones(r, 1)];

   [Beta, sigma] =   MaxVrais1(deltaTest, Zexpl, X0);

   db            =   ( (BetaTest-Beta)'*(BetaTest-Beta) )^(1/2);
   ds            =   abs(sigma-sigmaTest);

   m             =  0;
   for k=4:N
      if deltaTest(k,1) > 1 then
          m   = m+1;
      end
   end;

endfunction
```

Un exemple :
On crée aléatoirement avec les appels `Z0=rand(1203,12), b0=rand(1,12)` et `DataTestLin(Z0,b0,0.5)` un jeu de r = 12 paramètres $(\beta_1, ..., \beta_{12})$, un jeu de N = 1203 données fictives $(z_1^{(i)}, ..., z_{12}^{(i)})_{1 \leq i \leq N}$, un échantillon de taille N de la loi normale $\mathcal{N}(0, \sigma = 0.5)$, les résultats sont dans la matrice L (dont les termes sont de type string pour des raisons pratiques : Scilab n'accepte pas encore des matrices contenant à la fois des nombres et des chaînes) ; la feuille suivante présente les résultats du test.

```
Z0=rand(1203,12);
b0=rand(1,12);

[L0] = DataTestV1(Z0, b0, 0.5);

L0(1:4,:)
ans  =

!beta1      beta2      beta3      beta4      beta5      ...    sigma                  !
!
!0.7734171  0.3481372  0.8877226  0.1978170  0.9118797  ...    0.5         1          !
!
!z1         z2         z3         z4         z5         ...    u_i         delta      !
!
!0.8246909  0.1221573  0.7778672  0.1311745  0.5003578  ...   -0.1442251   3.9885892  !
```

La fonction `LeTestV1(L0)` retourne les évaluations au maximum de vraisemblance et les valeurs 'réelles' des β_i et de σ à partir des données de la matrice L. La qualité du résultat dépend du nombre de **variables observables.**

```
[sigma, sigmaTest, ds, Beta, BetaTest, db, m, N] = LeTestV1(L0);

[BetaTest,Beta]
ans  =

   0.7734171    0.8322092
   0.3481372    0.3462427
   0.8877226    0.8885109
   0.197817     0.0895952
   0.9118797    1.0186369
   0.9409176    0.9218390
   0.9945803    0.9241017
   0.1338397    0.1120221
   0.8047588    0.8114586
   0.8898184    0.8491407
   0.721926     0.7074363
   0.5509015    0.5863766

   [sigmaTest,sigma]
```

```
ans  =

    0.5    0.4904049
```

2.2 Scores orientés inputs le modèle $\Theta = \beta z + \epsilon = \max(\min(\beta\, z + u, 1), 0)$

Nous avons là un modèle avec **une double censure** puisque $\Theta \in]0, 1]$. Testons la recherche du maximum de vraisemblance

2.2.1 Fonction de répartition de la loi conditionnelle

• **Le modèle** $\Theta = \beta z + \epsilon = \max(\min(\beta\, z + u, 1), 0)$

On souhaite exprimer comme on l'a fait pour les scores orientés outputs, les scores orientés inputs (que nous avons notés Θ) en fonction des variables d'influence $(z_1, z_2, ...z_n)$ par une relation du type $\Theta(x, y) = \sum_i \beta_k z_k + \epsilon$ (toujours avec une des composantes constante). Les scores sont, cette fois-ci limités à gauche et à droite, puisque, par construction

$$0 < \Theta(x, y) \leq 1,$$

la meilleure efficience étant obtenue avec $\Theta = 1$. On formule l'hypothèse qu'il existe des coefficients β_k tels que

$$\Theta(x, y) = \psi(z, u) = \max(\min(\beta\, z + u, 1), 0) = \begin{cases} 0 & \text{si } \beta\, z + u \leq 0, \\ 1 & \text{si } \beta\, z + u \geq 1, \\ \beta\, z + u & \text{sinon.} \end{cases} \tag{2.14}$$

où u est une variable aléatoire de loi normale de moyenne 0 et variance σ^2. Comme pour les scores orientés outputs, nous déterminons avant tout une loi conditionnelle pour Θ. Nous notons toujours f et F la densité et la fonction de répartition de la loi normale centrée réduite.

• Nous allons nous intéresser à la loi conditionnelle de Θ sachant $\Theta \in]0, 1[$. Sa fonction de répartition s'exprime, pour $0 < t < 1$:

$$P_{\beta,\sigma,z}\left(\Theta \leq t \,|\, 0 < \Theta < 1\right) = \frac{P_{\beta,\sigma,z}\left(0 < \beta\, z + u \leq t\right)}{P_{\beta,\sigma,z}\left(0 < \beta\, z + u < 1\right)} \tag{2.15}$$

$$P_{\beta,\sigma,z}\left(\Theta \leq t \,|\, 0 < \Theta < 1\right) = \frac{F\left(\frac{t-\beta\, z}{\sigma}\right) - F\left(\frac{-\beta\, z}{\sigma}\right)}{F\left(\frac{1-\beta\, z}{\sigma}\right) - F\left(\frac{-\beta\, z}{\sigma}\right)} \tag{2.16}$$

Cette loi (conditionnelle) a pour densité de probabilité

$$\Phi(t) = \frac{1}{\sigma} \frac{f\left(\dfrac{t - \beta z}{\sigma}\right)}{F\left(\frac{1-\beta\, z}{\sigma}\right) - F\left(\frac{-\beta\, z}{\sigma}\right)} \tag{2.17}$$

2.2.2 Fonction de vraisemblance

Nous définissons comme en (2.4) le produit des $\Phi(\Theta_i)$ pour $0 < \Theta_i < 1$:

$$W_1(\hat{\Theta}, z, \beta, \sigma) = \frac{1}{\sigma^m} \prod_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} \frac{f\left(\frac{\Theta_i - \beta z^{(i)}}{\sigma}\right)}{F\left(\frac{1-\beta\, z^{(i)}}{\sigma}\right) - F\left(\frac{-\beta\, z^{(i)}}{\sigma}\right)} \tag{2.18}$$

dont le maximum est aussi celui de

$$\ln W_1(\hat{\Theta}, z, \beta, \sigma) = - \sum_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} \left[\ln \left(F\left(\frac{\hat{\Theta}_i - \beta z^{(i)}}{\sigma} \right) - F\left(\frac{-\beta z^{(i)}}{\sigma} \right) \right) + \frac{1}{2\sigma^2} \left(\hat{\Theta}_i - \beta z^{(i)} \right)^2 \right] - m \ln \sigma \tag{2.19}$$

($m = \text{Card}(\{i;\, \hat{\Theta}_i < 1\})$.

• Nous nous intéressons donc au maximum de la fonction

$$(\beta, \sigma) \in \mathbb{R}^r \times \mathbb{R}^{*+} \to \ln W_1(\hat{\delta}, z, \beta, \sigma).$$

Notons $u_i = \dfrac{-\beta z^{(i)}}{\sigma}$ et $t_i = \dfrac{\hat{\Theta}_i - \beta z^{(i)}}{\sigma}$, ce qui donne

$$\ln W_1(\hat{\Theta}, z, \beta, \sigma) = - \sum_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} \left[\ln \left(F(t_i) - F(u_i) \right) + \frac{1}{2}\, t_i^2 \right] - m \ln \sigma \tag{2.20}$$

$$\frac{\partial u_i}{\partial \beta_k} = \frac{\partial t_i}{\partial \beta_k} = \frac{-z_k^{(i)}}{\sigma}, \quad \frac{\partial u_i}{\partial \sigma} = \frac{-u_i}{\sigma}, \quad \frac{\partial t_i}{\partial \sigma} = \frac{-t_i}{\sigma}$$

$$\frac{\partial \ln W_1}{\partial \beta_k} = \frac{1}{\sigma} \sum_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} z_k^{(i)} \left(\frac{F'(t_i) - F'(u_i)}{F(t_i) - F(u_i)} + t_i \right) \tag{2.21}$$

$$\frac{\partial \ln W_1}{\partial \sigma} = \frac{1}{\sigma} \sum_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} \left(\frac{t_i\, F'(t_i) - u_i\, F'(u_i)}{F(t_i) - F(u_i)} + t_i^2 \right) - \frac{m}{\sigma} \tag{2.22}$$

2.2.3 Paramètres réalisant le maximum de la fonction de vraisemblance

Comme précédemment, disposant de n observations $\hat{\Theta}_i \in]0, 1]$, $z^{(i)} = [z_1^{(i)}, ..., z_r^{(i)}]$, nous recherchons les valeurs de $\beta =^t [\beta_1, ..., \beta_r]$ et $\sigma > 0$ réalisant le maximum de la fonction $\ln W_1(\hat{\Theta}, z, \beta, \sigma)$ définie en (2.19). Nous continuons à noter :

– les scores (orientés input) $\hat{\Theta} = \begin{bmatrix} \hat{\theta}_1 \\ \vdots \\ \hat{\theta}_n \end{bmatrix} \in \mathbb{R}^n$, les explicatives $z = \begin{bmatrix} z_{1,1} & \cdots & z_{1,r} \\ \vdots & & \vdots \\ z_{n,1} & \cdots & z_{n,r} \end{bmatrix}$
(matrices notées `Theta` et `Zexpl` dans nos programmes)

– pour un jeu de n données observées nous souhaitons optimiser la fonction

$$F : x \to \ln W_1(\hat{\theta}, z, [x_1, ...x_r], x_{r+1})$$

dont la variable est $x = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_r \\ \sigma \end{bmatrix} \in \mathbb{R}^{r+1}$, notée `X`.

2.2.4 Programmation sous Scilab

La fonction de vraisemblance $\ln W_1$ définie en (2.26) .

```
function val = VraisW1(Theta, Zexpl, Beta, sigma)

  [n,  p]  =  size(Theta);
  [n1, r]  =  size(Zexpl);
  [r1, q]  =  size(Beta);

  if  n ~= n1  then
    error("Les arguments Theta et Zexpl n ont pas le même nombre de colonnes");
  end;
  if r1 ~= r then
    error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
  end;
  if sigma <= 0 then
        sigma = -sigma+0.01;//--- disp(sigma)
        //error("sigma ne peut être négative dans  VraisLin1")
  end;

  S    = 0;
  m    = 0;
  for i=1:n
     if Theta(i,1) < 1 - 10^(-5) & Theta(i,1) > 10 ^(-3) then
        betaZ     =   Zexpl(i,:)*Beta;
        t1        =   (Theta(i,1)-betaZ)/sigma;
        u1        =   -betaZ/sigma;
        t2        =   t1^2/2;
        S         =   S  + t2  + log( cdfnor("PQ",t1,0,1)-cdfnor("PQ",u1,0,1));
        m         =   m+1;
      end;
  end;
  val  = -S - m*log(sigma);
endfunction;
```

Le gradient de $\ln W_1$ calculé en (2.27),(2.28)

```
function Gval = GradVraisW1(Theta, Zexpl, Beta, sigma)
   //---vérifié avec Maple (formel)
  [n,  p]  =  size(Theta);
  [n1, r]  =  size(Zexpl);
  [r1, q]  =  size(Beta);
  if  n ~= n1  then
    error("Les arguments Theta et Zexpl n ont pas le même nombre de colonnes");
  end;
  if r1 ~= r then
    error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
  end;
  if sigma <= 0 then
      sigma = - sigma + 0.001;
      //---  pour éviter une approximation trop proche de 0
  end;

  c     = 1/((2*%pi)^(1/2));
  Gval  = zeros(r+1,1);
  m     = 0;
  for i=1:n
    if Theta(i,1) < 1 - 10^(-5) & Theta(i,1) > 10 ^(-5)   then
      BetaZi = Zexpl(i,:)*Beta;
      ti    = (Theta(i,1)-BetaZi)/sigma;
      ui    = -BetaZi/sigma;
      Den   =   cdfnor("PQ",ti,0,1)-cdfnor("PQ",ui,0,1);
      dw    =   c*(exp(-ti^2/2) - exp(-ui^2/2) )/Den;
      m     =  m+1;
      for k=1:r
          Gval(k,1) =  Gval(k,1)+Zexpl(i,k)*(dw + ti);
      end;
      dw    =  c*(ti*exp(-ti^2/2) - ui*exp(-ui^2/2) )/Den;
      Gval(r+1,1) =  Gval(r+1,1) + ti^2 - 1 + dw;
    end;
  end;

  Gval         =   Gval/(sigma);
endfunction;
```

Quasi identique à celle du paragraphe (2.1.6), la fonction MdresCarresW1 pour une première approximation des β_i et de σ permettant l'initialisation de la recherche du maximum de vraisemblance.
La fonction D2 définie localement calcule $D2(B) = \sum_{0<\Theta_i<1}(\Theta_i - Z * B)^2$.

```
 function [Beta, sigma, f] = MdresCarresW1(Theta, Zexpl, B0)
//---------------    .....   -------------
   function [d, gd, ind] = D2(B, ind)
     dbZ     =  Theta-Zexpl*B;
     for i=1:size(Theta,1)
       if Theta(i,1) >= 1 - 10^(-5) | Theta(i,1) < 10^(-3) then dbZ(i,1)=0; end;
     end
     d       =  dbZ'*dbZ;
     gd      = -2*Zexpl'*dbZ;
   endfunction;

   //-------- estimation des Beta_i ----------

     [f, Beta]  =  optim(D2, B0,'gc');

   //-------- estimation de sigma ------------
     m         =  0;
     for i=1:size(Theta,1)
       if Theta(i,1) < 1 - 10^(-5) & Theta(i,1) > 10^(-5) then m = m +1;  end;
     end;
     sigma     =  (D2(Beta)/m)^(1/2);

 endfunction;
```

La fonction qui retourne les paramètres réalisant le maximum de vraisemblance

```
function [Beta, sigma] = MaxVraisW1(Theta, Zexpl, X0)
  //-------------------------------------------------
  function [f,g,ind] = CostVrais(X,ind)
    [r1, p] =   size(X);
    Beta    =   X(1:(r1-1), 1);
    sigma   =   X(r1, 1);
    f       = - VraisW1(Theta, Zexpl, Beta, sigma)
    g       = - GradVraisW1(Theta, Zexpl, Beta, sigma)
  endfunction;
  //-------------------------------------------------
 [n, u]    = size(Theta);
 [n1, r]   = size(Zexpl);
 [r1, u1]  = size(X0);

 if  n ~= n1  then
   error("Theta et Zexpl n ont pas le même nombre de colonnes");
 end;
 if  r ~= r1  then
   error("Zexpl et X0 ne sont pas compatibles (r col et r lignes)");
 end;
 //-------------------------------------------------
 //--- initialisons par une estimation aux moindres carrés (sur delta_i >1)

 [B0, s0]  =   MdresCarresW1(Theta, Zexpl, X0);

 M         =  10^10*ones(r+1,1);
 m         =  10^10*[-ones(r,1);0]
 [f, Xopt] =  optim(CostVrais,  'b', m, M , [B0;s0], 'qn');
 [p,q]     =  size(Xopt);
 Beta      =  Xopt(1:p-1,q);
 sigma     =  Xopt(p, q);

endfunction;
```

2.3 Scores orientés inputs, le modèle $\ln(1/\Theta) = \beta z + \epsilon$

Comme $\Theta \in]0,1]$, $\ln(1/\Theta) \in [0, +\infty[$ et nous pouvons aussi envisager un modèle

$$\ln(1/\Theta) = \max(\beta z + u, 0)$$

dans lequel u suit une loi normale centrée réduite.

– Nous calculons en (2.3.1) la loi conditionnelle de $\ln(1/\Theta)$ sachant $0 < \ln(/\Theta) < 1$ pour en déduire une première fonction de log-vraisemblance notée $\ln W_3$; nous traitons donc les seuls scores $\hat{\Theta} \in]0,1[$, comme pour une variable tronquée.

– En (2.3.2), nous calculons la loi (non absolument continue) de cette même variable aléatoire, en tenant compte de toutes les valeurs $0 < \ln(/\Theta) \leq 0$. La fonction de log-vraisemblance est notée $\ln W_4$; nous traitons ici notre variable comme un variable censurée, nous sommes dans le modèle Tobit.

Nous comparerons les résultats obtenus avec ces deux fonctions sur une batterie de tests.

2.3.1 Loi conditionnelle de $\ln(1/\Theta)$ sachant $0 < \ln(1/\Theta)$

• **La fonction de répartition de la loi conditionnelle** de $\ln(1/\Theta)$ sachant $0 < \ln(/\Theta)$ est alors donnée par la relation suivante :

$$P_{\beta,\sigma,z}(\ln(1/\Theta) \leq t | \ln(1/\Theta) > 0) = P_{\beta,\sigma,z}(\beta z + u \leq t | \beta z + u > 0)$$

$$= \frac{P_{\beta,\sigma,z}(0 < \beta z + u \leq t)}{P_{\beta,\sigma,z}(0 < \beta z + u)} = \frac{P_{\beta,\sigma,z}\left(\frac{-\beta z}{\sigma} < \frac{u}{\sigma} \leq \frac{t - \beta z}{\sigma}\right)}{P_{\beta,\sigma,z}\left(\frac{-\beta z}{\sigma} < \frac{u}{\sigma}\right)}$$

Ce qui donne une loi conditionnelle absolument continue de fonction de répartition :

$$P_{\beta,\sigma,z}(\ln(1/\Theta) \leq t | \ln(1/\Theta) > 0) = \frac{F\left(\frac{t - \beta z}{\sigma}\right) - F\left(\frac{-\beta z}{\sigma}\right)}{1 - F\left(\frac{-\beta z}{\sigma}\right)} \tag{2.23}$$

et de densité :

$$\Phi(t) = \frac{f\left(\frac{t - \beta z}{\sigma}\right)}{\sigma\left(1 - F\left(\frac{-\beta z}{\sigma}\right)\right)} \tag{2.24}$$

• **Fonction de vraisemblance de la loi conditionnelle :**

$$W_3(\hat{\Theta}, z, \beta, \sigma) = \prod_{0 < \ln(1/\hat{\Theta}_i)} \frac{f\left(\frac{\ln(1/\hat{\Theta}_i) - \beta z}{\sigma}\right)}{\sigma\left(1 - F\left(\frac{-\beta z}{\sigma}\right)\right)} \tag{2.25}$$

Nous nous intéressons donc au maximum de la fonction

$$(\beta, \sigma) \in \mathbb{R}^r \times \mathbb{R}^{*+} \to \ln W_3(\hat{\Theta}, z, \beta, \sigma).$$

Notons $u_i = \dfrac{-\beta z^{(i)}}{\sigma}$ et $t_i = \dfrac{\ln(1/\hat{\Theta}_i) - \beta z^{(i)}}{\sigma}$, ce qui donne à une constante additive près

$$\ln W_3(\hat{\Theta}, z, \beta, \sigma) = - \sum_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} \left[\ln\left(1 - F(u_i)\right) + \frac{1}{2}\, t_i^2 \right] - m \ln \sigma + \text{Cste} \tag{2.26}$$

$$\frac{\partial u_i}{\partial \beta_k} = \frac{\partial t_i}{\partial \beta_k} = \frac{-z_k^{(i)}}{\sigma}, \quad \frac{\partial u_i}{\partial \sigma} = \frac{-u_i}{\sigma}, \quad \frac{\partial t_i}{\partial \sigma} = \frac{-t_i}{\sigma}$$

$$\frac{\partial \ln W_3}{\partial \beta_k} = \frac{1}{\sigma} \sum_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} z_k^{(i)} \left(\frac{f(u_i)}{1 - F(u_i)} + t_i \right) \tag{2.27}$$

$$\frac{\partial \ln W_3}{\partial \sigma} = \frac{1}{\sigma} \sum_{\substack{i=1 \\ 0<\hat{\Theta}_i<1}}^{n} \left(\frac{-u_i\, f(u_i)}{1 - F(u_i)} + t_i^2 \right) - \frac{m}{\sigma} \tag{2.28}$$

2.3.2 La loi de $\ln(1/\Theta)$ (modèle Tobit)

Nous cherchons toujours à étudier le modèle $\ln(1/\Theta) = \max(\beta z + u, 0)$ dans lequel u suit une loi normale centrée réduite. Nous donnons ici la fonction de vraisemblance attachée à la loi et non plus à la loi conditionnelle sachant $\hat{\Theta} > 0$. C'est dans ce cas celle qui donne les meilleures estimations d'après les tests.

• **Exprimons cette fois directement la loi de** $\ln(1/\Theta)$:

$$P_{\beta,\sigma,z}\left(\ln(1/\Theta) = 0\right) = P_{\beta,\sigma,z}\left(\beta z + u \leq 0\right) = P_{\beta,\sigma,z}\left(\frac{u}{\sigma} \leq \frac{-\beta z}{\sigma}\right) = F\left(\frac{-\beta z}{\sigma}\right)$$

Par ailleurs, pour $t > 0$ nous avons :

$$P_{\beta,\sigma,z}\left(\ln(1/\Theta) \leq t\right) = P_{\beta,\sigma,z}\left(\beta z + u \leq t\right) = P_{\beta,\sigma,z}\left(\frac{u}{\sigma} \leq \frac{t - \beta z}{\sigma}\right) = F\left(\frac{t - \beta z}{\sigma}\right)$$

Comme nous le remarquions dans la situation analogue rencontrée pour l'étude des scores orientés outputs en (2.1.1), cette loi admet une fonction de répartition discontinue en $t = 0$: la mesure de probabilité $P_{z,\beta,\sigma}$ de la variable $\ln(1/\Theta)$ est somme d'une mesure atomique/discrète de support $\{0\}$, de masse $F\left(\frac{-\beta z}{\sigma}\right)$ et de la mesure absolument continue (par rapport à la mesure de Lebesgue) de densité :

$$\frac{d}{dt} F\left(\frac{t - \beta z}{\sigma}\right) \times \chi_{]0,+\infty[}(t) = \frac{1}{\sigma} f\left(\frac{t - \beta z}{\sigma}\right) \times \chi_{]0,+\infty[}(t). \tag{2.29}$$

• **Fonction de vraisemblance**
Nous posons

$$W_4(\hat{\Theta}, z, \beta, \sigma) = \prod_{\ln(1/\hat{\Theta}_i)=0} F\left(\frac{-\beta z^{(i)}}{\sigma}\right) \prod_{0<\ln(1/\hat{\Theta}_i)} \frac{1}{\sigma} f\left(\frac{\ln(1/\hat{\Theta}_i) - \beta z}{\sigma}\right) \tag{2.30}$$

Notons encore $u_i = \dfrac{-\beta z^{(i)}}{\sigma}$ et $t_i = \dfrac{\ln(1/\hat{\Theta}_i) - \beta z^{(i)}}{\sigma}$, ce qui donne comme toujours :

$$\ln W_4(\hat{\Theta}, z, \beta, \sigma) = \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)=0}}^{n} \ln\left(F(u_i)\right) - \frac{1}{2} \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)>0}}^{n} t_i^2 - m \ln \sigma + \text{Cste} \tag{2.31}$$

$$\frac{\partial u_i}{\partial \beta_k} = \frac{\partial t_i}{\partial \beta_k} = \frac{-z_k^{(i)}}{\sigma}, \quad \frac{\partial u_i}{\partial \sigma} = \frac{-u_i}{\sigma}, \quad \frac{\partial t_i}{\partial \sigma} = \frac{-t_i}{\sigma}$$

$$\frac{\partial \ln W_4}{\partial \beta_k} = \frac{1}{\sigma} \left(- \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)=0}}^{n} \frac{z_k^{(i)} f(u_i)}{F(u_i)} + \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)>0}}^{n} z_k^{(i)}\, t_i \right) \tag{2.32}$$

$$\frac{\partial \ln W_4}{\partial \sigma} = \frac{1}{\sigma} \left(- \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)=0}}^{n} \frac{u_i\, f(u_i)}{F(u_i)} + \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)>0}}^{n} t_i^2 \right) - \frac{m}{\sigma} \tag{2.33}$$

2.3.3 Programmation

La fonction de vraisemblance $\ln W_4$ définie en (2.31) .

```
quasiZeroW4 = 10^(-5);
c  = 1/((2*%pi)^(1/2));
function val = VraisW4(Theta, Zexpl, Beta, sigma)
  //-- vérifié (maple)
  [n,  p]  =  size(Theta);
  [n1, r]  =  size(Zexpl);
  [r1, q]  =  size(Beta);
  if  n ~= n1  then
    error("Les arguments Theta et Zexpl n ont pas le même nombre de colonnes");
  end;
  if r1 ~= r then
    error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
  end;
  if sigma <= 0 then
        sigma = -sigma + 0.01;//?
        //--error("sigma ne peut être négative dans VraisW4")
  end;
  S1   = 0;  S2   = 0; m    = 0;
  for i=1:n
     betaZ     =   Zexpl(i,:)*Beta;
     if  Theta(i,1) < 1 - quasiZeroW4     then
        ti        =   (log(1/Theta(i,1))-betaZ)/sigma;
        t2        =   ti^2;
        S1        =   S1  + t2;
        m         =   m+1;
    else
        ui        =   -betaZ/sigma;
        try
            S2        =   S2 + log(cdfnor("PQ",ui,0,1) );
        catch
            disp('erreur évitée avec ui ='+sci2exp(ui)+' dans Vraisw4');
            S2        =   S2 - log(-ui)-ui^2/2+log(c);
            //--- il s'agit d'un DA de l'expression précédente
        end;
      end;
  end;
  val  = S2-S1/2 - m*log(sigma);
endfunction;
```

et son gradient $\ln W_4$ calculé en (2.32) et (2.33) .

```
 function Gval = GradVraisW4(Theta, Zexpl, Beta, sigma)
  //---vérifié avec Maple (formel)
 [n,  p]  =  size(Theta);
 [n1, r]  =  size(Zexpl);
 [r1, q]  =  size(Beta);
 if  n ~= n1  then
   error("Les arguments Theta et Zexpl n ont pas le même nombre de colonnes");
 end;
 if r1 ~= r then
   error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
 end;
 if sigma <= 0 then  n
     sigma = - sigma + 0.001;
     //--- error("sigma ne peut être négative dans  GradVraisW4")
 end;
 c      = 1/((2*%pi)^(1/2));
 Gval   = zeros(r+1,1);
 m      = 0;
 for i=1:n
    betaZi    =   Zexpl(i,:)*Beta;
    if Theta(i,1) < 1 - quasiZeroW4   then
        m     =   m+1;
        ti    = (log(1/Theta(i,1)) -betaZi)/sigma;
        for k=1:r
          Gval(k,1)   =  Gval(k,1) + Zexpl(i,k)*ti;
        end;
        Gval(r+1,1)   =  Gval(r+1,1) + ti^2 -1;
     else
        ui    = - betaZi/sigma;
        try
            Qi   =  c*exp(-ui^2/2)/cdfnor("PQ", ui,0,1);
        catch
            Qi   = -ui-1/ui;
            //--- il s'agit d'un DA de l'expression précédente
        end:
        for k=1:r
          Gval(k,1)   =  Gval(k,1) - Zexpl(i,k)*Qi;
         end;
      end;
   end;
 Gval         =   Gval/(sigma);
endfunction;
```

2.3.4 Simulations et tests de comparaison

Comme nous l'avons fait en (2.1.7) pour l'étude des scores orientés outputs, nous avons simulé des variables fictives et testé la pertinence (numérique) de la fonction de vraisemblance choisie pour estimer les paramètres de la loi d'une variable aléatoire de la forme $\max(\beta z + u, 0)$. Nous présentons un exemple de test de la fonction $\ln W_4$; on ne s'attendait pas à des miracles, d'ailleurs, comme toujours en statistique de trop bons résultats auraient mérités que l'on s'interroge.

```
--> Z0=rand(1203,12);
--> b0=rand(1,12);
--> [L0] = DataTestThetaW4(Z0, b0, 0.5);
--> L0(1:6,[1:3,9:12])
 ans  =
!beta1      beta2      beta3      beta9      beta10     beta11     beta12     !
!                                                                             !
!0.8310247  0.9227482  0.8207030  0.3158927  0.0847504  0.3860950  0.4354146  !
!                                                                             !
!z1         z2         z3         z9         z10        z11        z12        !
!                                                                             !
!0.6914338  0.5665873  0.4789428  0.194643   0.3711332  0.6488165  0.0438165  !
!                                                                             !
!0.5738613  0.3334710  0.6173977  0.0707813  0.4654203  0.7781754  0.3955218  !
!                                                                             !
!0.8251716  0.3469922  0.9175509  0.8770939  0.7443425  0.9899402  0.1343431  !

-->[sigma, sigmaTest, ds, Beta, BetaTest, db, m, N] = LeTestW4(L0);

-->[Beta, BetaTest]
 ans  =
    0.8009372    0.8310247
    0.8614096    0.9227482
    0.8352497    0.820703
    0.7432059    0.7418647
    0.5693813    0.5960354
    0.3341213    0.2887943
    0.5924809    0.6110385
    0.3140852    0.3102259
    0.3152084    0.3158927
    0.1002217    0.0847504
    0.3828991    0.386095
    0.5321053    0.4354146

-->[sigma, sigmaTest]
 ans  =
    0.5010634    0.5
```

3 Intervalles de confiance par simulation (bootstrapping)

3.1 Scores orientés outputs : modèle linéaire tronqué

3.1.1 Estimations des paramètres et bootstrap

– Nous avons fait l'hypothèse que le score d'efficience est de la forme

$$\delta = \max(1, \beta z + u) = \begin{bmatrix} \max(1, \beta z^{(1)} + u_1) \\ \vdots \\ \max(1, \beta z^{(n)} + u_n) \end{bmatrix} \in \mathbb{R}^n;$$

c'est donc une variable aléatoire dont la loi dépend des relevés Z et des paramètres inconnus $\beta = (\beta_1, ..., \beta_r)$ et σ (chaque u_i suit la loi normale $\mathcal{N}(0, \sigma)$);

– nous disposons de relevés de n observations

$$Z = \begin{bmatrix} z_{1,1} & \dots & z_{1,r} \\ \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,r} \end{bmatrix} \text{ et } \hat{\delta} = \begin{bmatrix} \hat{\delta}_1 \\ \vdots \\ \hat{\delta}_n \end{bmatrix} \in \mathbb{R}^n,$$

dans lesquelles $(\hat{\delta}_i)_{1 \leq i \leq n}$ forme une réalisation d'un n-échantillon de la variable δ;

– nous avons calculé $(\hat{\beta}_1, \hat{\beta}_2..., \hat{\beta}_r)$ et $\hat{\sigma}$ estimations au maximum de vraisemblance des paramètres inconnus à partir des données observées $(\hat{\delta}_i)_{1 \leq i \leq n}$ et $(z^{(i)})_{1 \leq i \leq n}$ (variables explicatives).

Nous souhaiterions alors, au vu de ces résultats, **valider ou rejeter des hypothèses de travail** du type $[\beta_k > 0]$ dont la signification est :
[*la $k^{ième}$ variable explicative z_k, intervient comme facteur d'accroissement de δ*].
Bien évidemment, nous ne connaissons pas la précision de notre estimation et le signe de $\hat{\beta}_k$ n'est pas nécessairement celui de β_k.
Nous allons alors procéder par simulation pour déterminer des intervalles de confiance pour la variable $\hat{\beta}_k$ et tester ces hypothèses. C'est ce procédé qui est développé et validé par Simar et Wilson, en particulier dans [4].

Le principe du bootstrap [6] **:**

Nous considérons $(\hat{\beta}_1, \hat{\beta}_2..., \hat{\beta}_r)$ et $\hat{\sigma}$ comme des observations de variables aléatoires dépendant de la loi de $\delta = \max(\beta z + u)$ obtenues par le procédé suivant :

$$\left\{ Z = \begin{bmatrix} z_{1,1} & \dots & z_{1,r} \\ \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,r} \end{bmatrix}, \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_r \end{bmatrix}, \sigma \right\} \rightarrow \left\{ \begin{bmatrix} \max(1, \beta z + u_1) \\ \vdots \\ \max(1, \beta z + u_n) \end{bmatrix} \right\} \rightarrow \left\{ \begin{bmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_r \end{bmatrix}, \hat{\sigma} \right\} \quad (3.1)$$

Connaissant les coefficients β et σ, nous n'aurions pas de problème pour calculer les probabilités

$$P_{Z,\beta,\sigma}(\hat{\beta}_k \in I)$$

6. On consultera [2] pour une présentation simplifiée,[1] pour plus de détails...

pour des intervalles quelconques [7] et, partant, pour définir des tests d'hypothèses et des intervalles de confiance pour des statistiques portant sur les $\hat{\beta}_k$.

Le principe du bootstrap (ici explicité pour notre cas) conduit à remplacer le processus (3.4) par un processus simulé, de paramètres évidemment tous connus :

$$\left\{ Z = \begin{bmatrix} z_{1,1} & \dots & z_{1,r} \\ \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,r} \end{bmatrix}, \begin{bmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_r \end{bmatrix}, \hat{\sigma} \right\} \rightarrow \left\{ \begin{bmatrix} \max(1, \hat{\beta} z + u_1^*) \\ \vdots \\ \max(1, \hat{\beta} + u_n^*) \end{bmatrix} \right\} \rightarrow \left\{ \begin{bmatrix} \hat{\beta}_1^* \\ \vdots \\ \hat{\beta}_r^* \end{bmatrix}, \hat{\sigma}^* \right\} \quad (3.2)$$

On détermine alors approximativement la loi des variables $(\hat{\beta}_1^*, \hat{\beta}_2^* ..., \hat{\beta}_r^*)$ et $\hat{\sigma}_s^*$ à l'aide d'un nombre Nb suffisant de tirages aléatoires (nous choisirons le plus souvent de l'ordre de un à quelques milliers de tirages).

Nous sommes alors en mesure d'observer pour chaque indice $k\,(1 \leq k \leq r)$, des intervalles de confiance au niveau α, $I_{k,\alpha}^*$ pour $\hat{\beta_k}^*$:

$$P_{Z,\hat{\beta},\hat{\sigma}} \left(\hat{\beta_k}^* \in I_{k,\alpha}^* \right) \geq 1 - \alpha.$$

Ce sont ces intervalles que nous prendrons comme estimations des intervalles de confiance $I_{k,\alpha}$ définis de la même façon pour la loi $P_{Z,\hat{\beta},\hat{\sigma}}$.
Les procédures correspondantes (simulation et analyse des résultats simulés) sont présentées en (3.1).

Construction pratique d'intervalles de confiance :

Supposons que nous ayons obtenu une valeur $\hat{\beta}_k > 0$; nous souhaitons naturellement savoir dans quelle mesure son signe est celui de β_k.
Nous considérons alors $I_{k,\alpha}^* =]t_\alpha^*, +\infty[$ tel que $P_{Z,\hat{\beta},\hat{\sigma}} \left(\hat{\beta_k}^* \in I_{k,\alpha}^* \right) \geq 1 - \alpha$.
Si $t_\alpha^* > 0$ nous saurons que si les paramètres de P sont effectivement $(\hat{\beta}, \hat{\sigma})$, alors, les observations $\hat{\beta}_k^{*,s}$ sont positives en proportion supérieure à $1 - \alpha$.
Nous déterminons (grâce à la loi simulée) une approximation de ce réel t_α^*; s'il n'est pas du même signe que $\hat{\beta}_k$ nous devrons renoncer à notre hypothèse. Dans les programmes qui suivent t_α^* est noté icm5 lorsque $\alpha = 0.01$, et icm10 lorsque $\alpha = 0.1$.

On trouvera un bref résumé sur les tests statistiques en section 6 (glossaire).

7. Qu'il s'agisse d'un calcul formel ou d'une approximation numérique par simulation de la loi

3.1.2 La programmation sous Scilab

Simulation

Notons les statistiques de la $s^{ième}$ simulation :

– $\hat{\delta}^{*,s} = \begin{bmatrix} \hat{\delta}_1^{*,s} \\ \vdots \\ \hat{\delta}_n^{*,s} \end{bmatrix} \in \mathbb{R}^n$ le vecteur des n scores $\hat{\delta}_i^{*,s} = \max(1, \hat{\beta} z^{(i)} + u_i)$ où u_i, variable aléatoire simulée, suit la loi $\mathcal{N}(0, \hat{\sigma})$;

– $(\hat{\beta}_{s,1}^*, \hat{\beta}_{s,2}^* ..., \hat{\beta}_{s,r}^*)$ et $\hat{\sigma}_s^*$ les paramètres estimés au maximum de vraisemblance à partir de ce $s^{ième}$ tirage $\hat{\delta}^{*,s}$.

Une fois $\hat{\delta} = \begin{bmatrix} \hat{\delta}_1 \\ \vdots \\ \hat{\delta}_n \end{bmatrix}$, $(\hat{\beta}_1, \hat{\beta}_2 ..., \hat{\beta}_r)$ et $\hat{\sigma}$ calculés comme précédemment, la fonction Scilab ci-dessous retourne un échantillon simulé de taille `Nb` de

$$(\hat{\beta}_{s,1}^*, \hat{\beta}_{s,2}^* ..., \hat{\beta}_{s,r}^*)_{1 \leq s \leq Nb} \text{ et } (\hat{\sigma}_s^*)_{1 \leq s \leq Nb}.$$

La fonction `boostrap` prend en arguments un vecteur `Beta` de r lignes, un réel positif `sigma`, une matrice `Zexpl` de taille $n \times r$, (copie du fichier des variables explicatives), un entier `Nb`. Elle retourne

– `bootBeta` une matrice de r lignes et `Nb` colonnes qui sont des simulations des $[\hat{\beta}_{s,1}^*, \hat{\beta}_{s,2}^* ..., \hat{\beta}_{s,r}^*]'$,

– `mB` le vecteur des moyennes empiriques de chaque ligne, sdB celui des écart-types ;

– `bootSigma, mS, sdS` le vecteur ligne des $\hat{\sigma}_s^*$, sa moyenne et son écart-type.

L'appel est :

```
[bootBeta, mB, sdB, bootSigma, mS, sdS] = bootstrap(Beta,sigma,Zexpl,Nb)
```

Beta et sigma sont les paramètres estimés au maximum de vraisemblance, Zexpl les données explicatives dont on dispose, Nb la taille de l'échantillon bootstrap que l'on veut obtenir ; bootBeta de taille $r \times Nb$ et bootSigma de taille $1 \times Nb$ sont les simulations.

Simulation bootstrap pour le modèle $\max(1, \beta z + u)$ et avec la fonction $\ln V_1$.

```
function   [bootBeta, mB, sdB, bootSigma, mS, sdS]
                                    = bootstrap(Beta,sigma,Zexpl,Nb)

  [r,u]    = size(Beta);
  [n,r1]   = size(Zexpl);

  if r1 ~= r then
     error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
  end;

  bootBeta  = zeros(r,Nb);
  bootSigma = zeros(1,Nb);

  for b = 1:Nb
     deltaStar = zeros(n,1);
     for i =1:n
       U               =  grand(1,1, 'nor', 0, sigma );
       BetaZ           =  Zexpl(i,:)*Beta;
       deltaStar(i,1)  =  max(1, BetaZ + U);
     end;
     [BetaStar, sigmaStar] =   MaxVrais1(deltaStar, Zexpl, [ones(r, 1)]);

      bootBeta(:,b)    =  BetaStar;
      bootSigma(1,b)   =  sigmaStar;
  end;

  mB    =  mean(bootBeta,'c');
  sdB   =  st_deviation(bootBeta,'c');

  mS    =  mean(bootSigma);
  sdS   =  st_deviation(bootSigma);

endfunction;
```

3.2 Scores orientés input : modèle Tobit

3.2.1 Estimations des paramètres et bootstrap

Dans ce cas, le score d'efficience présenté en (2.3.2) est de la forme

$$\ln(1/\Theta) = \max(\beta z + u), 0) = \begin{bmatrix} \max(0, \beta z^{(1)} + u_1) \\ \vdots \\ \max(0, \beta z^{(n)} + u_n) \end{bmatrix} \in \mathbb{R}^n.$$

Le principe du bootstrap est toujours le même, il consiste à remplacer le processus de paramètres inconnus

$$\left\{ Z = \begin{bmatrix} z_{1,1} & \dots & z_{1,r} \\ \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,r} \end{bmatrix}, \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_r \end{bmatrix}, \sigma \right\} \to \left\{ \begin{bmatrix} \max(0, \beta z + u_1) \\ \vdots \\ \max(0, \beta z + u_n) \end{bmatrix} \right\} \to \left\{ \begin{bmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_r \end{bmatrix}, \hat{\sigma} \right\} \tag{3.3}$$

par le processus simulé, de paramètres évidemment tous connus :

$$\left\{ Z = \begin{bmatrix} z_{1,1} & \dots & z_{1,r} \\ \vdots & & \vdots \\ z_{n,1} & \dots & z_{n,r} \end{bmatrix}, \begin{bmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_r \end{bmatrix}, \hat{\sigma} \right\} \to \left\{ \begin{bmatrix} \max(0, \beta z^{(1)} + u_1^*) \\ \vdots \\ \max(0, \beta z^{(n)} + u_n^*) \end{bmatrix} \right\} \to \left\{ \begin{bmatrix} \hat{\beta}_1^* \\ \vdots \\ \hat{\beta}_r^* \end{bmatrix}, \hat{\sigma}^* \right\} \tag{3.4}$$

Les changements dans le code par rapport à (3.1) sont mineurs. L'analyse de la loi simulée pour les paramètres β^* et σ^* est inchangée.

3.2.2 Le code

Simulation bootstrap pour le modèle $\ln(1/\Theta) = \max(0, \beta z + u)$ et avec la fonction de vraisemblance $\ln W_4$.

```
  function [bootBeta, mB, sdB, bootSigma, mS, sdS]
              = bootstrapThetaW4(Beta,sigma,Zexpl,Nb)

  [r,u]    = size(Beta);
  [n,r1]   = size(Zexpl);

  if r1 ~= r then
     error("Les dimensions de Zexpl et de Beta ne sont pas compatibles");
  end;

  bootBeta  = zeros(r,Nb);
  bootSigma = zeros(1,Nb);

  for b = 1:Nb

     ThetaStar = zeros(n,1);
     for i =1:n
       U               =  grand(1,1, 'nor', 0, sigma );
       BetaZ           =  Zexpl(i,:)*Beta;
       ThetaStar(i,1)  =  exp(- max(0, BetaZ + U) );
       //--- penser que ln(1/Theta) = max(0, BetaZ + U);
     end;
      [BetaStar, sigmaStar] =   MaxVraisW4(ThetaStar, Zexpl, [ones(r, 1)]);

      bootBeta(:,b)    =  BetaStar;
      bootSigma(1,b)   =  sigmaStar;
  end;

  mB    =  mean(bootBeta,'c');
  sdB   =  st_deviation(bootBeta,'c');

  mS    =  mean(bootSigma);
  sdS   =  st_deviation(bootSigma);

endfunction;
```

3.3 Analyse statistique des simulations

La simulation réalisée, on détermine sans mystère les statistiques empiriques que sont la moyenne, l'écart-type, la proportion $P_{\hat{\beta},\sigma,Z}(\beta_k^* < 0)$ ainsi que des intervalles de confiance aux niveaux 5% et 10%. Il serait illusoire de chercher plus précis pour des applications en économétrie.
Scilab dispose d'une fonction `perctl` (percentile) telle que `perctl(Liste, n)` pour n entier entre 1 et 100, retourne la valeur (interpolée) du quantile $Q_{n/100}$ pour la loi associée à `Liste` :

$$P(X = k) = \frac{\#\{i; Liste(1, i) = k\}}{\#Liste}.$$

Comme notre but est de tester le signe des coefficients β_k, nous recherchons des intervalles de confiance aux niveaux $\alpha = 0.05$ et $\alpha = 0.1$ de la forme :

$P\left(\hat{\beta}_k^* \geq ic5m\right) = 0.95$ $I_{k,\alpha}^* = [ic5m, +\infty[$	$P\left(\hat{\beta}_k^* \leq ic5p\right) = 0.95$ $I_{k,\alpha}^* =]-\infty, ic5p]$
$P\left(\hat{\beta}_k^* \geq ic10m\right) = 0.90$ $I_{k,\alpha}^* = [ic10m, +\infty[$	$P\left(\hat{\beta}_k^* \leq ic10p\right) = 0.90$ $I_{k,\alpha}^* =]-\infty, ic10p]$

avec pour les bornes des intervalles, les mêmes notations que dans le programme qui suit...

La fonction `analyse` prend en arguments `bootBetak`, une matrice ligne de `Nb` colonnes qui sont les simulations $[\hat{\beta}^*_{1,k}, ..., \hat{\beta}^*_{s,k}..., \hat{\beta}^*_{Nb,k}]'$.
Elle retourne
- `sortBk` la ligne `bootBetak` rangée dans l'ordre croissant ;
- `mBk` la moyenne empirique de `bootBetak` ;
- `sdBk` son écart-type ;
- `f` $= P_{\hat{\beta},\sigma,Z}(\beta^*_k < 0)$, calculée avec `[f, p] = pValNeg(sortBk)`
- `ic5m`, `ic5p`, `ic10m` et `ic10p` sont les bornes respectives des intervalles de confiance pour $\hat{\beta}^*_k$ définis dans le tableau qui précède.

L'appel est : `[sortBk, mBk, sdBk, ic5m, ic5p, ic10m, ic10] = analyse1(bootBetak)`

```
function  [sortBk, mBk, sdBk, f, ic5m, ic5p, ic10m, ic10p] = analyse1(bootBetak)

 sortBk    =   gsort(bootBetak, 'c','i');
 [f, p]    =   pValNeg(sortBk);
 mBk       =   mean(sortBk);
 sdBk      =   st_deviation(sortBk);
 p         =   perctl(sortBk, 5);
 ic5m      =   p(1,1);
 p         =   perctl(sortBk, 95);
 ic5p      =   p(1,1);
 p         =   perctl(sortBk, 10);
 ic10m     =   p(1,1);
 p         =   perctl(sortBk, 90);
 ic10p     =   p(1,1);
endfunction
```

La fonction qui suit retourne la fréquence f des termes nuls et la position p du premier terme positif dans une liste croissante de taille `size(LigneOrd)=(1,n)` ; $p = n + 1$ si tous les termes sont négatifs. La fonction `analyse1` y fait appel.

```
function [f, p] =  pValNeg(LigneOrd)
   //--- attention LigneOrd: ligne ordonnée croissante
   //--- retourne la fréquence des termes négatifs de LigneOrd
   //--- et la position éventuelle du premier terme >= 0;
   [u, n]  = size(LigneOrd);

   if LigneOrd(1,1) > 0  then
       p  = 1;
       f  = 0;
   elseif LigneOrd(1,n) < 0 then
       p  = n+1;
       f  = 1;
   else
       p  = 1;
       z  = LigneOrd(1,1); //--- z < 0
       while z < 0 & p <n
           p  = p+1;
           z  = LigneOrd(1,p);
       end;
       f  = (p-1)/n;
   end;
endfunction
```

La fonction ci-dessous assure l'écriture des fichiers de résultats après analyse des données simulées (voir 4.3) pour une idée plus précise...

```
function Sortie = presenterAnalyse(repertoireBoot, fichierBoot, partieDuNom)
 //--- lecture fichier des valeurs simulées
[Boot,valBoot,isNBoot]  =  lectureNbresCsv(fichierBoot,';');
[lf,cf]                 =  size(valBoot);
//--- les valeurs hat_beta_star sont dans les col 4 à ...Nb+3
//---la première ligne contient les noms des variables de chaque colonne

bootBeta  =  valBoot(2:lf, 4:cf);
[r, Nb]   = size(bootBeta);
Sortie    = ['nom_expl', 'beta', 'beta*E(Z)/s' 'E(beta*)', 'sigma(beta*)',
                            'P(Betak<0)','ic5m', 'ic5p','ic10m', 'ic10p'];

for k = 1:r
  disp('traiter variable k ='+ sci2exp(k));
  bootBetak     = bootBeta(k,:);
 [sortBk, mBk, sdBk, f, ic5m, ic5p, ic10m, ic10p]  = analyse1(bootBetak);

  disp('nom de la variable'+ sci2exp(k)+': '+ Boot(k+1,1));

  Sortiek  = [Boot(k+1,1:3), convertirCsv([mBk, sdBk, f, ic5m, ic5p, ic10m, ic10p])];
  Sortie   = [Sortie; Sortiek];

  scf(k);//--- set the current graphic figure (window)
  h    =  (sortBk(1,Nb)-sortBk(1,1))/100;
  x    =  [sortBk(1,1): h : sortBk(1,Nb)];
  plot(x, densiteNorm( mBk, sdBk, x) );
  histplot(50, sortBk, normalization =%t);

  // attention au décalage d'indices entre Boot (ligne 1 titres) et bootBeta
  titreh = 'k = '+ sci2exp(k)+' ---  '+Boot(k+1,1)+': --- Max vrais: '+Boot(k+1,2);
  titreb =  'moy* = '+ sci2exp(mBk) + '; ecart-type* = '+sci2exp(sdBk);
  xtitle(titreh,titreb); //--- la figure --

  xs2pdf( k , repertoireBoot + '\Beta' + sci2exp(k)+ partieDuNom +'.pdf' );
 end;
 nomfichierRes  = 'DeaAnalyse'+ partieDuNom +'.csv';
 try
      write_csv(Sortie , repertoireBoot + '\'+ nomfichierRes, ";" , ".");
      mclose();
 catch
      disp('Vérifiez que le fichier n est pas déjà ouvert: '+ lasterror());
 end;
endfunction;
```

4 Mise en œuvre, mode d'emploi

Les scripts permettent, à partir d'un fichier de données (au format csv)
• de calculer des scores d'efficience en orientation output ou orientation input ;
• de régresser ces scores sur un certain nombre de variables explicatives ;
• de simuler les lois des scores d'efficience à partir des coefficients obtenus par régression ;
• de calculer des statistiques bootstrap de ces simulations.

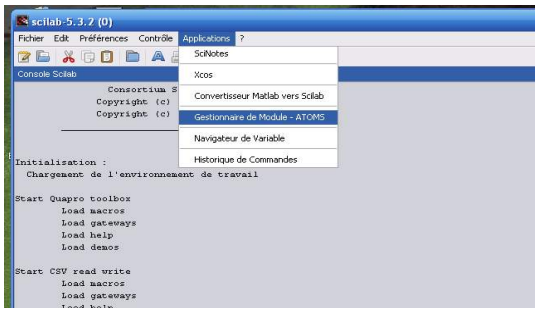
Ils offrent une interface graphique sommaire qui dispense d'écrire quelque ligne de code que ce soit :
• vous choisissez le répertoire et le fichier, les données à prendre en considération, leur nature (inputs, outputs, explicatives) et vous êtes guidés dans la suite des opérations à mener ;
• les résultats sont sauvés au format csv directement lisibles par Excel, OpenOffice ou LibreOffice...

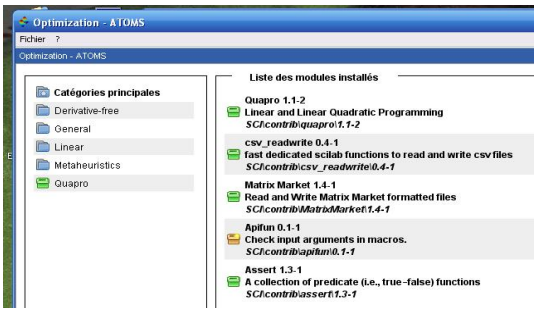
Ce qui suit décrit l'installation et fournit un mode d'emploi. On suppose que Scilab est déjà installé sur votre machine. Vous pouvez le télécharger à l'adresse

http ://www.scilab.org

4.1 Chargement préalable de la la bibliothèque `Quapro`

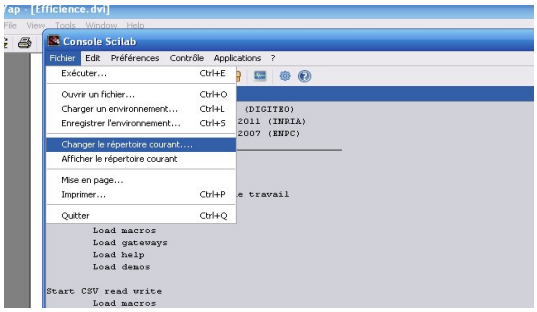
Autant pour l'optimisation linéaire que pour la recherche des extremums des fonctions de vraisemblance, les scripts font appel à la bibliothèque (ou `toolbox`) `Quapro` de Scilab . Il faudra donc l'installer en priorité. Pour cela utiliser le gestionnaire de modules Atoms de Scilab (dans la barre de menu : Applications, Gestionnaire de Modules-Atoms, puis ouvrir le dossier Optimization et choisir `Quapro`) comme illustré par les copies d'écran ci-dessous réalisées avec Scilab 5.3).





4.2 Agencement des scripts

Vous installerez ensuite les 7 scripts dans le répertoire de votre choix. Vous indiquerez de préférence à Scilab ce répertoire comme **répertoire de travail** (Menu : Fichier, Changer le répertoire courant, comme sur la copie d'écran ci-dessous), ce qui vous évitera de saisir le chemin complet lorsque vous exécuterez les scripts.



Comme ces scripts ne tournent qu'avec les données que vous leur proposez, vous penserez à préparer vos **fichiers de données au format csv** [8] dans le répertoire de votre choix, une clé usb etc..., avant de les exécuter. Ces préalables réalisés, vous n'aurez à lancer que les trois premiers scripts (en `***.sce`), les autres, bien que nécessaires aux calculs, restent en boîte noire pour l'utilisateur. Rappelons que c'est l'instruction Scilab `exec` qui appelle un script :

`exec('DEAOutputGui.sce');` [9]

DEAOutputGui.sce
: Vous appellerez ce script pour le calcul de scores δ orientés **outputs**, l'estimation des paramètres explicatifs, le calcul de simulations bootstraps. La fonction de vraisemblance est la fonction $\ln V_1$ de (2.4).

  Ce script vous offre une interface graphique décrite dans le paragraphe suivant.

DEAOutputGuiW4.sce
: L'analogue du précédent pour les scores Θ orientés **inputs**. La fonction de vraisemblance est la fonction $\ln W_4$ associée à la loi de $\ln(1/\Theta) = \max(0, \beta z + u)$ définie en (2.1.3) ;

AnalyseBootstrap.sce
: Vous appellerez ce script pour l'analyse des données simulées à partir d'une première estimation (données bootstraps). Vous aurez au préalable réalisé ces simulations avec un des deux scripts précédents. Offre une interface graphique.

Les scripts suivant restent en boîte noire pour l'utilisateur.

DEA.sci contient les fonctions pour les calculs des scores d'efficience ; boîte noire pour l'utilisateur ;

GestionCsv.sci contient des fonctions pour la lecture des fichiers csv ; boîte noire pour l'utilisateur ;

Vraisemblance_lnV1.sci contient les fonctions pour l'estimation des paramètres, la simulation et les outils de vérification pour δ et $\ln V_1$; boîte noire pour l'utilisateur ; associé à DEAOutput.sce ;

Vraisemblance_lnW4.sci contient les fonctions pour l'estimation des paramètres, la simulation et les outils de vérification pour $\ln(1/\Theta)$ et $\ln W_4$; boîte noire pour l'utilisateur ; associé à DEAInputW4.sce ;

8. La conversion est proposée dans les tableurs

9. Attention, le répertoire n'est pas précisé dans l'appel ; il a été au préalable choisi comme répertoire courant.

4.3 Organisation et traitement des fichiers de données

Nous supposons que les données initiales figurent dans un fichier csv dont la première ligne donne **les noms** des différentes variables (identifiants, inputs, outputs, explicatives). Les n lignes suivantes donnent les valeurs correspondantes. Chaque ligne est une observation dans laquelle on retrouvera **en positions quelconques** les variables

$$id^{(i)}, x_1^{(i)}, .., x_p^{(i)}, y_1^{(i)}, .., y_q^{(i)}, z_1^{(i)}, .., z_r^{(i)}.$$

L'utilisateur désignera lui-même les variables (x_i) (inputs), (y_j) (outputs) et (z_k) (explicatives) parmi celles qui figurent dans le fichier

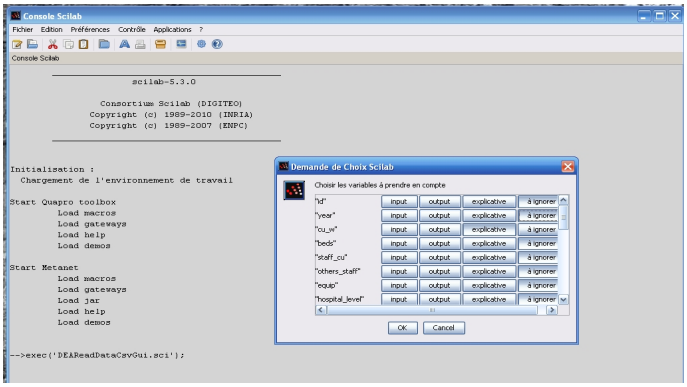
Le type présentation attendue, l'ordre des colonnes importe peu :

Feuille1

idtownship	year	prev_w	prev_staff	others_staff	t_surface	participation	t_ncms	dist	hospital_levet	rural_inc_cons
11	2000	5503.75	4.000	3	52	0	0	23	0	0,35
12	2000	16788.75	5.000	7	82	0	0	15	0	0,33
13	2000	52504.85	20.000	26	161	0	0	26	1	0,41
14	2000	7496.2	2.000	1	54	0	0	33	0	0,36
15	2000	6515.7	10.000	7	113	0	0	50	0	0,27
16	2000	16248.55	4.000	2	85	0	0	12	1	0,39
17	2000	30599	3.000	0	150	0	0	50	0	0,25
21	2000	553.15	7.000	7	48	0	0	10	1	0,41
22	2000	7692.3	2.000	2	37,42	0	0	12	0	0,38
31	2000	14043.35	6.000	15	68,53	0	0	23	1	0,32
32	2000	8935.5	2.000	2	98,3	0	0	45	0	0,3
33	2000	1818.55	2.000	2	78,83	0	0	28	0	0,31
34	2000	2834.2	2.000	0	103,5	0	0	20	1	0,3
41	2000	14282	11.000	0	99	0	0	15	1	0,32
42	2000	25298.75	8.000	0	119,6	0	0	12	0	0,32
43	2000	14589.1	8.000	0	62	0	0	35	0	0,3
44	2000	16063.55	8.000	10	75,12	0	0	20	0	0,33
45	2000	11682.75	7.000	2	78	0	0	22	1	0,33
46	2000	10752.2	4.000	0	229	0	0	26	0	0,2
51	2000	25742.75	7.000	5	78	0	0	15	1	0,44
52	2000	19117.9	4.000	3	87	0	0	30	1	0,35
53	2000	34252.75	8.000	4	62	0	0	16	0	0,39
61	2000	18829.3	3.000	14	125,2	0	0	20	0	0,34
62	2000	8868.9	5.000	10	89,41	0	0	38,5	0	0,31

Les scripts `DEAOutputGui.sce` et `DEAInputGuiW4.sce` permettent de réaliser les opérations suivantes à l'aide de boîtes de dialogue :

1. choix du répertoire et du fichier des données ;
2. lecture du fichier et choix des colonnes contenant les inputs, outputs et déterminants ;



3. calculs des scores $\hat{\delta}_i$ (pour $1 \leq i \leq n$) ou $\hat{\Theta}_i$ (pour $1 \leq i \leq n$) selon le choix du script ; il est possible de choisir un seul calcul d'efficience sur l'ensemble des données ou des calculs par périodes (le nombre d'items par période est pour l'instant supposé constant) ; les résultats de ces opérations sont enregistrés dans des fichiers `DeaOutputScores***.csv` ou `DeaInputScores***.csv` (les `***` désignent une chaîne de caractères choisie par l'utilisateur) ;
4. à la demande, calcul des estimations au maximum de vraisemblance des paramètres $(\hat{\beta}_1, \hat{\beta}_2..., \hat{\beta}_r)$ et $\hat{\sigma}$; les résultats de ces opérations sont enregistrés dans des fichiers `DeaOutputParam***.csv`, `DeaInputParam***.csv`
5. à la demande calcul de simulations bootstrap

$$(\hat{\beta}^*_{s,1}, \hat{\beta}^*_{s,2}..., \hat{\beta}^*_{s,r}), \text{ et } \hat{\sigma}^*_s$$

comme cela peut être long, l'Analyse est proposée dans un autre script (on pourra tester avec N = 50 ou 100 pour évaluer le temps de calcul nécessaire avant de lancer les simulations en grandeur nature avec N = 1000 ou N =2000...).
Les résultats sont sauvegardés dans un fichier `DEAOutputBootstrap***.csv` ou `DEAInputBootstrapW4***.csv` en vue de leur traitement par le script **Analyse-Bootstrap.sce**

4.4 Un exemple

On se propose d'étudier l'efficience d'unités observées sur plusieurs années. Les données issues de cette étude sont regroupées en un fichier comprenant des variables observées que nous considérerons comme Inputs, Outputs, Explicatives. Nous conduisons des études en orientation output et en orientation input. Dans les deux cas, pour une même étude de scores, nous cherchons à expliquer l'efficience à partir de différents jeux d'explicatives.

• **Mise en place, particularités :**

– Ces données concernent 24 unités de production (hôpitaux en l'occurrence) et sont collectées sur plusieurs années ; on a choisi de calculer les scores DEA par année (donc par paquets de 24 items) ; l'interface permet ce choix.

– Certaines explicatives de l'année n sont des relevés de l'année $n-1$, l'étude ne peut commencer que la deuxième année ou deuxième période [10] ; ce cas est prévu dans l'interface, on effectue donc la régression sur les données à partir de la deuxième période.

– On souhaite conduire l'étude avec, pour commencer, les configurations suivantes :

	Configuration C1	Configuration C2	Configuration P1	Configuration P2
Orientation	**Output**	**Output**	**Input**	**Input**
Var output	cu_w	cu_w	prev_w	prev_w
Var input	lits	lits	prev_staff	prev_staff
Var input	staff	staff	other_staff	other_staff
Var input	staff_others	staff_others	-	-
Var input	equipment	equipment	-	-
Var expl	**participation**	**t_nrcms**	**participation**	**t_nrcms**
Var expl	dist	dist	dist	dist
Var expl	hospital_level	hospital_level	hospital_level	hospital_level
Var expl	t_rural_inc_cons	t_rural_inc_cons	t_rural_inc_cons	t_rural_inc_cons
Var expl	hl_pers	hl_pers	hl_pers	hl_pers
Var expl	density	density	density	density
Var expl	**r_curative_q_staff**	**r_curative_q_staff**	**r_prev_q_staff**	**r_prev_q_staff**
Var expl	r_sub	r_sub	r_sub	r_sub
Var expl	r_others_inc_1	r_others_inc_1	r_others_inc_1	r_others_inc_1
Var expl	r_solde_1	r_solde_1	r_solde_1	r_solde_1
Var expl	**scores prev_io**	**scores prev_io**	**score_cu_oo**	**score_cu_oo**

Remarque : scores prev_io : alias $\ln(1/\hat{\Theta})$; score_cu_oo :alias $\hat{\delta}$;

– Les tableaux qui suivent sont produits par le script `AnalyseBootstrap.sce` qui opère successivement sur les fichiers des données simulées (contenant Nb = 2000 tirages) produits par `DEAOutputGui.sce` pour C1 et C2 et par `DEAInputGuiW4.sce` en ce qui concerne P1 et P2.

Ils font apparaître les rapports $\dfrac{\hat{\beta}_k \times E(z_k)}{\hat{\sigma}}$ qui donnent une idée du poids relatif de chacune des variables explicatives (le produit $\hat{\beta}_k \times E(z_k)$ est plus significatif que la seule valeur des coefficients β_k). Ce quotient permet de comparer l'influence de la variable à

10. l'oublier conduit à des résultats aberrants

l'effet du terme aléatoire u_i de loi $\mathcal{N}(0, \hat{\sigma})$.

– $P(\beta_k^* < 0)$ est une estimation de la probabilité que le coefficient $\hat{\beta}$ soit négatif. Rappelons que l'on attend un $\beta_k > 0$ lorsqu'on prévoit que la variable z_k joue un rôle négatif dans l'efficience, que le score étudié soit $\hat{\delta}$ ou $\ln(1/\hat{\Theta})$.

• Configuration C1

Analyse de 2000 simulations :

noms	$\hat{\beta}_k$	$\frac{\hat{\beta}_k * E(Z)}{\hat{\sigma}}$	$E(\hat{\beta}_k^*)$	$\sigma(\hat{\beta}_k^*)$	$P(\beta_k^* < 0)$	ic5m	ic5p	ic10m	ic10p
cste	3.9816019	3.2565067	4.0033268	1.472747	0.0025	1.7341092	6.5353797	2.1727143	5.9414348
part	-0.5143709	**-0.204506**	-0.2701721	0.5226528	**0.705**	-1.143059	0.5738933	-0.9273040	0.3849133
dist	-0.0361330	**-0.7345088**	-0.0290003	0.0167223	**0.9625**	-0.0577071	-0.0026706	-0.0508666	-0.0083929
hosp_lev	-0.5874330	**-0.1801705**	-0.4378318	0.3882537	**0.882**	-1.0934124	0.1513026	-0.9267318	0.0334835
t_r_inc	-1.3398836	-0.4667875	-0.6642071	2.213148	0.61	-4.3670954	2.9118336	-3.4698225	2.1520578
hs_pers	-0.5770343	**-1.1296107**	-0.5138517	0.2845179	**0.9675**	-0.9940543	-0.0579005	-0.8817287	-0.1702007
density	-0.0067260	**-0.7078804**	-0.0059223	0.0054927	**0.8665**	-0.0154107	0.0025110	-0.0130593	0.0007916
r_cur_q_s	0.7596017	0.0959645	0.3668013	1.2365831	0.391	-1.6031515	2.4023881	-1.1601361	1.9268895
r_sub	2.3755357	**0.1597106**	1.2581537	1.3069529	**0.165**	-0.8699422	3.4831533	-0.4088786	2.9692042
r_oth_inc_1	0.3237883	0.0620716	0.2628833	0.8962775	0.379	-1.2485492	1.7606082	-0.8741164	1.3944105
r_solde_1	0.1746396	0.0432902	0.2153210	0.9379151	0.43	-1.2450592	1.762922	-0.8986065	1.4124238
theta	-0.1864630	-0.1028674	-0.1127330	0.5587616	0.57	-1.0551922	0.7930178	-0.8228386	0.6068875
d2001	0.1912454	0.0173797	-0.0524048	0.6756451	0.513	-1.2299234	1.0166463	-0.9198665	0.7631017
d2002	0.1918627	0.0174358	-0.0551841	0.6691002	0.523	-1.197815	1.0362598	-0.9049612	0.7681403
d2003	0.6912099	0.0628147	0.3240987	0.6094202	0.2735	-0.7516891	1.2632982	-0.4882826	1.056687
d2004	1.1155856	0.1013805	0.5658088	0.6288529	0.177	-0.5400507	1.5462105	-0.2536686	1.3579515
d2005	1.2537433	0.1139358	0.6606744	0.6575373	0.1525	-0.4395375	1.6943893	-0.1879569	1.4851436
d2006	1.5907745	0.1445640	0.8215945	0.7537675	0.135	-0.4957667	2.0207473	-0.1917792	1.7794529
d2007	1.8173169	0.1651514	1.0269431	0.7619903	0.0785	-0.2163901	2.2555881	0.0942813	1.9436328
d2008	1.5263604	0.1387103	0.8062300	0.8082725	0.156	-0.4968116	2.1641639	-0.2078957	1.8523764
sigma	1.2226604	1	0.9970131	0.1323594	0	0.8032825	1.2261693	0.8386020	1.1684739

Ces résultats ne semblent pas se prêter à une interprétation claire : trop peu de relations entre les valeurs $\hat{\beta}_k$ et les moyennes des simulations $E(\hat{\beta}_k^*)$ ce qui laisse penser trop de variables sont fortement corrélées (dans la mesure où les simulations purement aléatoires donnent des résultats cohérents).
On peut toutefois observer une évolution temporelle des efficiences relatives (entre les années 2001-2002 et 2004 et suivantes) qui était attendue dans cette étude.

• Configuration C1a

On reprend l'étude avec un choix de variables explicatives restreint (et tenant compte du poids relatif exprimé par $\frac{\hat{\beta}_k * E(Z)}{\hat{\sigma}}$) :
hosp_lev, r_cur_q_s, r_sub, density, hs_pers, dist, participation.

noms	$\hat{\beta}_k$	$\frac{\hat{\beta}_k * E(Z)}{\hat{\sigma}}$	$E(\hat{\beta}_k^*)$	$\sigma(\hat{\beta}_k^*)$	$P(\beta_k^* < 0)$	ic5m	ic5p	ic10m	ic10p
cste	0.2288259	0.1464218	0.6166259	2.4941744	0.354	-4.1034565	4.2311231	-2.7826827	3.5545371
part	0.1872431	0.0582427	0.1126613	0.5749227	0.4195	-0.8181916	1.0626609	-0.5831315	0.8133641
dist	0.0213004	0.3387567	0.0193963	0.0327133	0.269	-0.0324662	0.0774697	-0.0186354	0.0632765
hosp_lev	-0.5772600	-0.1385170	-0.4448347	0.7629248	0.7305	-1.7372318	0.8086334	-1.4337783	0.4887467
hs_pers	-0.6288892	-0.9631804	-0.4965077	0.5068183	0.85	-1.329885	0.3180193	-1.1433575	0.1395257
density	0.0063354	0.5216550	0.0051159	0.0086444	0.2695	-0.0083223	0.0194495	-0.0052423	0.0157730
r_cur_q_s	0.1935834	0.0191337	0.0197745	2.4491588	0.515	-3.9547439	4.2938208	-2.8525463	3.0763906
r_sub	1.3595299	0.0715101	0.7699566	2.3848003	0.374	-3.0314966	4.7692237	-1.9044375	3.6836145
sigma	1.5627861	1	1.3300489	0.3177865	—	0.8979678	1.9393126	0.9713667	1.7509949

Les résultats de C1a sont plus cohérents. Les différences criantes entre les estimations $\hat{\beta}_k$ et les moyennes des simulations $E(\hat{\beta}_k^*)$ de la configuration C1 qui invalidaient toute tentative d'interprétation de certains paramètres, ont pour l'essentiel disparu.

• **Conclusion provisoire**

– les variables dist, density ont les plus fortes probabilités d'être positives et donc de dégrader le score d'efficience (qui se dégrade avec $\delta \nearrow$).
– hs_pers et hosp_lev ont les plus fortes probabilités d'être négatives et ainsi d'améliorer l'efficience puisque $\delta \searrow 1$ quand les performances augmentent.
– Comme distance et densité sont sans doute liées à une même situation géographique et corrélées, on peut envisager de n'en garder qu'une pour une étude plus restreinte encore...

• **Configuration P1**

La première configuration donne après analyse de 2000 simulations [11]

noms	$\hat{\beta}_k$	$\frac{\hat{\beta}_k * E(Z)}{\hat{\sigma}}$	$E(\hat{\beta}_k^*)$	$\sigma(\hat{\beta}_k^*)$	$P(\beta_k^* < 0)$	ic5m	ic5p	ic10m	ic10p
cste	1.1448657	1.7949976	1.0195755	0.2625512	0.002	0.6151765	1.4294034	0.7051722	1.3432813
part	0.0211549	0.0161233	-0.0094464	0.0485206	0.5835	-0.0905983	0.0704492	-0.0703306	0.0538037
dist	-0.0089473	**-0.3486586**	-0.0061263	0.0047068	**0.9355**	-0.0130983	0.0005107	-0.0113317	-0.0010581
hosp_lev	-0.0426833	-0.0250957	0.0071859	0.0380352	0.411	-0.0550277	0.0680869	-0.0425651	0.0564001
t_r_inc	-0.4207095	-0.2809634	0.0985676	0.3000857	0.364	-0.4060066	0.5910116	-0.2885402	0.4774958
hs_pers	0.0635287	**0.2384034**	0.0200941	0.0150344	**0.0945**	-0.0050661	0.0454204	0.0006101	0.0394265
density	-0.0018784	**-0.3789726**	-0.0026557	0.0013719	**0.973**	-0.0048639	-0.0003533	-0.0043745	-0.0008790
r_p_q_s	-1.928531	-0.0606775	0.5292229	0.7482422	0.24	-0.6752311	1.7156712	-0.3973545	1.4740287
r_sub	0.0206089	0.0026561	-0.0071524	0.2396964	0.5185	-0.4055302	0.3940278	-0.3125040	0.2935082
r_oth_inc_1	-0.0665950	-0.0244731	0.0095471	0.1269088	0.48	-0.2006015	0.2243288	-0.1518307	0.1747275
r_solde_1	-0.0151032	-0.0071768	0.0063136	0.1264164	0.48	-0.2025174	0.2187883	-0.1565930	0.1736086
delta	0.0149530	0.0375943	-0.0029078	0.0130544	0.586	-0.0248545	0.0176790	-0.0193346	0.0131562
sigma	0.6378091	1	0.5615770	0.0440877	—	0.5078273	0.6230819	0.5197846	0.6078874

Les β_k qui ont une probabilité importante d'être positifs ou négatifs sont associés aux variables :
- dist, density, pour lesquelles $P(\beta_k^* < 0)$ sont les plus grandes ;
- r_rp_q_s, hs_pers pour lesquelles $P(\beta_k^* > 0)$ est la plus grande ; mais en ce qui concerne r_rp_q_s, la contribution parait très faible.

• **Configuration P1a**

Nous reprenons l'étude avec les 3 variables qui sont en apparence les plus significatives dans la configuration précédente, ce qui donne :

noms	$\hat{\beta}_k$	$\frac{\hat{\beta}_k * E(Z)}{\hat{\sigma}}$	$E(\hat{\beta}_k^*)$	$\sigma(\hat{\beta}_k^*)$	$P(\beta_k^* < 0)$	ic5m	ic5p	ic10m	ic10p
constante	0.7623687	1.1746079	0.6963489	0.1257166	0	0.4939717	0.9041904	0.5361770	0.8629189
dist	-0.0102481	-0.3924391	-0.0081635	0.0040980	0.973	-0.0148516	-0.0011654	-0.0133249	-0.0027893
hosp_lev	-0.0576940	-0.0333342	0.0062607	0.0309492	0.411	-0.0453573	0.0570263	-0.0339882	0.0453655
hs_pers	0.0673838	0.2484941	0.0367885	0.0181640	0.0185	0.0070049	0.0669793	0.0132204	0.0594708
r_pr_q_s	-2.3778873	-0.0735209	0.6172362	0.6626834	0.1685	-0.4679937	1.7388385	-0.2132760	1.4778564
sigma	0.6490410	1	0.5963325	0.0391409	0	0.5330478	0.6622731	0.5459764	0.6465815

• **Configuration P1b**

Nous avions aussi essayé la configuration suivante avec hosp_lev et participation mais :

11. et environ 3200 secondes de temps de calcul avec un Celeron et 2Go de mémoire vive, pour n = 24 x 9 items observés.

noms	$\hat{\beta}_k$	$\frac{\hat{\beta}_k * E(Z)}{\hat{\sigma}}$	$E(\hat{\beta}_k^*)$	$\sigma(\hat{\beta}_k^*)$	$P(\beta_k^* < 0)$	ic5m	ic5p	ic10m	ic10p
constante	0.7791955	1.1914872	0.7016491	0.1271047	0	0.4959528	0.9116915	0.5401927	0.8695102
part	-0.0248851	-0.0184977	0.0016784	0.0257153	0.476	-0.0396071	0.0428900	-0.0292631	0.0338047
dist	-0.0103176	-0.3921213	-0.0082156	0.0041508	0.973	-0.0149021	-0.0011429	-0.0135001	-0.0027295
hosp_lev	-0.0613163	-0.0351601	0.0067840	0.0314661	0.4165	-0.0447639	0.0569649	-0.0331938	0.0471971
hs_pers	0.0666852	0.2440651	0.0361695	0.0184440	0.0195	0.0060699	0.0667426	0.0126258	0.0591039
density	0.0003677	0.0741703	-0.0000076	0.0011464	0.4865	-0.0019367	0.0017944	-0.0014946	0.0013773
r_pr_q_s	-2.232349	-0.0685009	0.5801361	0.6741793	0.188	-0.5261077	1.6984218	-0.2774664	1.4461912
sigma	0.6539688	1	0.6022009	0.0394744	—	0.5389325	0.6693944	0.5527047	0.6531473

• **Conclusion provisoire** La configuration P1a semble offrir des interprétations plus cohérentes que P1b qui ne sera pas retenue pour étude :

– La variable dist avec une forte probabilité d'être négative aurait un effet positif sur l'efficience puisque $\ln(1/\Theta) \searrow 0$ ou $\Theta \nearrow 1$ lorsque celle-ci s'améliore. C'est l'effet inverse que celui observé dans les activités curatives.

– hs_pers et r_pr_q_s ont une forte probabilité d'être positives, mais r_pr_q_s à un poids très faible et les résultats ne semblent pas du tout cohérents.

5 Les erreurs de calcul

Expliquons brièvement, pour qui souhaiterait entrer dans le code, comment nous avons géré les divisions par zéro.

La fonction `VraisW4` (voir (2.3.3)) réalise le calcul de lnW_4 définie en (2.31) par

$$\ln W_4(\hat{\Theta}, z, \beta, \sigma) = \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)=0}}^{n} \ln\left(F(u_i)\right) - \frac{1}{2} \sum_{\substack{i=1 \\ \ln(1/\hat{\Theta}_i)>0}}^{n} t_i^2 - m \ln \sigma + \text{Cste}$$

avec en particulier $u_i = \dfrac{-\beta z^{(i)}}{\sigma}$, et $F(x) = \dfrac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2}\, dt$.

Le codage 'naturel' de cette fonction provoque des divisions par zéro (ou breakdown), c'est pourquoi est réécrit de la façon suivante

```
ui        =   -betaZ/sigma;
try
    S2        =   S2 + log(cdfnor("PQ",ui,0,1) );
catch
    disp('erreur évitée avec ui ='+sci2exp(ui)+' dans Vraisw4');
    S2        =   S2 + -ui^2/2 -log (-ui) +log(c);
    //--- il s'agit d'un DA de l'expression précédente
end;
```

En effet `log(cdfnor("PQ",ui,0,1) )` $= \ln\left(\int_{-\infty}^{u_i} e^{-t^2/2}\, dt\right) + \ln c$, et lorsque u_i est assez petit ($u_i < -14$ sous Scilab), l'intégrale est évaluée à zéro et le calcul du log retourne une erreur. Un étude numérique suggère néanmoins que $\ln(F(x))$ pour $x \to -\infty$, est proche de $-x^2/2$ malgré les erreurs de calcul qui apparaissent assez vite (ici 10 chiffres sous MAPLE) :

x	x^2	$F(x)/c = \int_{-\infty}^{x} e^{-t^2/2}\, dt$	$\ln(F(x)/c)$
-30	900	$1.229930786 \times 10^{-197}$	- 453.4023054
-31	961	$6.755990058 \times 10^{-211}$	- 483.9350251
-32	1024	$1.366633939 \times 10^{-224}$	- 515.4667101
-33	1089	$1.017988270 \times 10^{-238}$	- 547.9974237
-34	1156	$2.792130189 \times 10^{-253}$	- 581.5272237
-35	1225	$2.819732982 \times 10^{-268}$	- 616.0561627
-36	1296	$1.048428375 \times 10^{-283}$	- 651.5842891
-37	1369	$1.435187871 \times 10^{-299}$	- 688.1116470
-38	1444	0.	Float$(-\infty)$
-39	1521	0.	Float$(-\infty)$
-40	1600	0.	Float$(-\infty)$

• Nous allons donc calculer des développements asymptotiques de $F(x)$ et de $\ln(F(x))$ en $-\infty$. On procède par intégrations par parties successives :

$$\frac{F(x)}{c} = \int_{-\infty}^{x} \frac{1}{t} \times (t\, e^{-t^2/2})\; dt = \left[-\frac{1}{t} \times e^{-t^2/2} \right]_{-\infty}^{x} - \int_{-\infty}^{x} \frac{1}{t^2} \times e^{-t^2/2}\; dt \tag{5.1}$$

• On intègre à nouveau par parties avant de remplacer :

$$\int_{-\infty}^{x} \frac{e^{-t^2/2}}{t^2}\; dt = \int_{-\infty}^{x} \frac{1}{t^3}(te^{-t^2/2})\; dt = \left[\frac{-e^{-t^2/2}}{t^3} \right]_{-\infty}^{x} + 3\int_{-\infty}^{x} \frac{1}{t^4}\; e^{-t^2/2}\; dt \tag{5.2}$$

• Comme pour deux fonctions positives u et v, on a :

$$u(x) \underset{x\to-\infty}{=} o\left(v(x)\right) \Rightarrow \int_{-\infty}^{x} u \underset{x\to\infty}{=} o\left(\int_{-\infty}^{x} v\right),$$

nous pouvons déduire de (5.2) que

$$\int_{-\infty}^{x} \frac{e^{-t^2/2}}{t^2}\; dt = \frac{-f(x)}{x^3} + 3\int_{-\infty}^{x} \frac{1}{t^4}\; e^{-t^2/2}\; dt \underset{x\to\infty}{=} \frac{-f(x)}{x^3} + o\left(\int_{x}^{\infty} \frac{e^{-t^2/2}}{t^2}\; dt\right) \underset{x\to\infty}{\sim} \frac{-f(x)}{x^3} \tag{5.3}$$

et la définition des équivalents assure que l'on a aussi :

$$3\int_{-\infty}^{x} \frac{1}{t^4}\; e^{-t^2/2}\; dt \underset{x\to\infty}{=} o\left(\frac{f(x)}{x^3}\right) \tag{5.4}$$

• Ainsi, après avoir remplacé dans (5.1), nous obtenons un développement asymptotique (chaque terme du membre de droite est négligeable devant le précédent) :

$$\frac{F(x)}{c} = \frac{-f(x)}{x} + \frac{f(x)}{x^3} + 3\int_{-\infty}^{x} \frac{1}{t^4}\; e^{-t^2/2}\; dt \tag{5.5}$$

$$\frac{F(x)}{c} \underset{x\to\infty}{=} \frac{f(x)}{|x|}\left(1 - \frac{1}{x^2} + o\left(\frac{1}{x^2}\right)\right) \tag{5.6}$$

d'où enfin,

$$\ln(F(x)) \underset{x\to\infty}{=} \ln f(x) - \ln|x| + \ln c + \ln\left(1 - \frac{1}{x^2} + o\left(\frac{1}{x^2}\right)\right) \tag{5.7}$$

$$\ln(F(x)) \underset{x\to\infty}{=} \frac{-x^2}{2} - \ln|x| + \ln c - \frac{1}{x^2} + o\left(\frac{1}{x^2}\right) \tag{5.8}$$

Le code effectif contient donc une fonction **try catch end**, qui lorsqu'une erreur est rencontrée dans la ligne `S2 = S2 + log(cdfnor("PQ",ui,0,1) )` lui substitue le calcul `S2 = S2 + -ui*ui/2 -log (-ui) +log(c);` plus robuste.

```
try
  S2   =   S2 + log(cdfnor("PQ",ui,0,1) );
catch
  S2   =   S2 + -ui^2/2 -log (-ui) +log(c);
end;
```

• De la même façon, le calcul du gradient de $\ln W_4$ (voir 2.3.3) contient le code

```
try
  Qi   =  c*exp(-ui^2/2)/cdfnor("PQ", ui,0,1);
catch
  Qi   = -ui-1/ui;
end:
```

6 Glossaire et références

Vocabulaire à préciser...

DEA : data envelopment analysis - analyse par enveloppement des données ;

DMU : decision making units - unité de production au sens large (branches d'industrie, firmes, hôpitaux ou établissement d'enseignement selon le cas...)

percentile (et quantile) : le quantile $(1-\alpha)$ de la variable aléatoire X est la valeur Q_α telle que

$$P(x < Q_\alpha) \leq 1-\alpha \leq P(X \leq Q_\alpha).$$

Scilab dispose d'une fonction `perctl` (percentile) telle que `perctl(Liste, n)` pour n entier entre 1 et 100 retourne la valeur (interpolée) de $Q_{n/100}$ pour la loi associée à `Liste`, à savoir :

$$P(X = k) = \frac{\#\{i; Liste(1,i) = k\}}{\#Liste}$$

test statistique

Rappelons le principe d'un test statistique destiné à évaluer une hypothèse H_0 portant sur la loi d'une variable aléatoire X.

(i) On détermine une propriété que la loi d'une variable aléatoire X satisfaisant à l'hypothèse H_0 vérifie avec une probabilité $1-\alpha$. On formule généralement cette condition sous la forme :
Si H_0 est vraie, alors pour tout $n-$échantillon $(X_1, X_2, ..., X_n)$, de la loi de X,

$$\phi(X_1, X_2, ..., X_n) \notin \mathcal{R},$$

avec une probabilité $1-\alpha$.
(ii) On réalise l'expérience aléatoire, et si $\phi(X_1, X_2, ..., X_n)$ est dans la région de rejet $\mathcal{R}$, on rejette l'hypothèse.

Le tableau suivant présente les quatre situations possibles :

	H_0 est vraie	H_0 est fausse
on rejette H_0	erreur (de probabilité α)	bonne décision
on ne rejette pas H_0	bonne décision	erreur

On raisonne en fait comme toujours en sciences expérimentales : tant que l'hypothèse n'est pas manifestement fausse on ne la rejette pas.
La probabilité α que l'hypothèse soit rejetée à l'issue du test, alors qu'elle est vraie, est le niveau de confiance du test. On appelle valeur critique au niveau α un réel t_α tel que

$$P(\phi(X_1, X_2, ..., X_n) \leq t_\alpha) = 1-\alpha.$$

Lorsque la fonction ϕ est telle que la loi de $\phi(X_1, X_2, ..., X_n)$ ne dépend pas des lois des (X_i) on parle de statistique libre. C'est la situation du test de Kolmogorov Smirnov.
Dans le cas des tests du χ^2, ce sont les comportements asymptotiques (les lois limites de $\phi(X_1, X_2, ..., X_n)$ lorsque n tend vers l'infini) qui sont indépendants de la loi de X.

variable dépendante limitée : variable dépendante à densité continue mais observable sur un sous-intervalle de $\mathbb{R}$.

variable tronquée : les observations des variables explicatives et de la variable dépendante sont exclues de l'échantillon lorsque cette dernière se trouve en dehors d'un certain intervalle ;

variable censurée : on dispose de l'ensemble des observations des variables explicatives alors que la variable dépendante est inobservable en dehors d'un certain intervalle et rapportée à une même valeur (dans nos exemples d'études de scores, les valeurs $\beta z + u$ ne sont pas directement observées et remplacées par $\max(\ell, \beta z + u)$).
C'est le cas du modèle Tobit (dont la fonction de vraisemblance est (2.9)).
Références : définitions du cours de H. Harari-Kermadec (ens-cachan - URCA 2008-2009 à préciser)
et Russel (Vieille Charité université de Marseille)

Références

[1] A.C. Davison & D.V. Hinkley . *Bootstrap Methods and their Application*
Cambridge University Press, 1997-2009.

[2] C. Huber . *Une méthode de rééchantillonnage : le bootstrap*
http ://www.biomedicale.univ-paris5.fr/survie/enseign/cours_Bootstrap_C_Huber_web.pdf

[3] R. J. Olsen. *Note on the uniqueness of the maximum likehood estimator for the Tobit model*
Econometrica, Vol. 46, N° 5 (September, 1978)

[4] L. Simar & P. W. Wilson . *Estimation and inference in two-stage, semi-parametric models of production processes*
J. of Econometrics 136 (2007) 31-64, 2007.
disponible en ligne : www.elsevier.com/locate/jeconom

[5] Jan A. Snyman . *Practical Mathematical Optimisation*
Springer, 2005.

Index