Guide d'économétrie appliquée pour Matlab

Simon Leblond[1]
Université de Montréal
simon.leblond@umontreal.ca

26 décembre 2003

[1]Merci à William McCausland, François Vaillancourt et Benoit Perron pour leurs commentaires utiles dans l'élaboration de ce document. Je demeure seul responsable de toutes les erreurs.

Table des Matières

1 Commandes générales importantes **5**
- 1.1 Importation des données . . . 5
- 1.2 Manipulation des données . . . 7
  - 1.2.1 Opérateurs mathématiques . . . 7
  - 1.2.2 Opérateurs logiques et de comparaison . . . 7
  - 1.2.3 Fonctions et Opérateurs matriciels . . . 7
  - 1.2.4 Autres trucs utiles . . . 9
- 1.3 Autres transformations des variables . . . 9
- 1.4 Boucles et tests logiques . . . 10
- 1.5 Divers . . . 11
- 1.6 Exemples et résultats pour le Chapitre 1 . . . 12

2 Visualisation des données **13**
- 2.1 Impression/Exportation des données . . . 13
  - 2.1.1 Impression à l'écran . . . 13
  - 2.1.2 Exportation . . . 13
  - 2.1.3 Impression . . . 13
- 2.2 Exemples et résultats pour le Chapitre 2 . . . 14

3 Graphiques **15**
- 3.1 Exemples et résultats pour le Chapitre 3 . . . 17

4 Régressions Simples **18**
- 4.1 MCO . . . 18
  - 4.1.1 Tests d'hypothèses et Intervalles de confiance . . . 19
  - 4.1.2 Estimateurs de Variance Robustes . . . 20
- 4.2 Tests d'hétéroscédasticité . . . 20
- 4.3 Test de Changement structurel (Test de Chow) . . . 21

5 Variables instrumentales et Doubles Moindres Carrés **23**
- 5.1 Estimateur Variables Instrumentales . . . 23
- 5.2 DMCO . . . 24

5.3 Tests d'endogénéité 25

6 Estimateur du Maximum de Vraissemblance (EMV) 26

7 Moindres Carrés Généralisés 28

8 Variables dépendantes qualitatives 30
8.1 Probit/Logit 30
8.2 Tobit 32

9 Séries Chronologiques 33
9.1 Opérateurs de séries temporelles 33
9.2 Tests d'autocorrélation 33
9.3 Méthode de Box-Jenkins 34
9.3.1 Stationnarité des données 34
9.3.2 Sélection de Modèle 35

10 Données longitudinales (Panel) 37
10.1 Effets Fixes et Effets Aléatoires 37
10.1.1 Variables binaires 37

11 Interaction avec les tableurs et les traitements de texte 39
11.1 Remarques 39
11.2 Tableur 39
11.3 Traitement de texte 40

12 Où trouver ses données et comment les extraires 41
12.1 Liens utiles 41
12.1.1 À l'Université 41
12.1.2 À l'Extérieur 41
12.2 Accès au Données 42
12.2.1 Statcan 42
12.2.2 OCDE 43
12.2.3 Sherlock 43

A Tableaux Récapitulatifs 45
A.1 Fonctions de Matlab 45
A.2 Opérateurs 48
A.3 Symboles Mathématiques 48
A.4 Alphabet Grec 49

Introduction

Ce guide est d'abord et avant tout adressé aux étudiants du cours ECN 3949. Bien qu'il indique le chemin pour résoudre une grande quantité de problèmes économétriques, il ne prétend pas offrir des solutions optimales. Matlab étant un logiciel de manipulation de matrices très général, pour l'utiliser efficacement comme logiciel économétrique, il est préférable de faire appel à des fonctions déjà construites. Ainsi, les fonctions et les approches présentés ont d'avantage un objectif pédagogique que purement économétrique.

Aussi, ce manuel ne vient aucunement se substituer à vos notes de cours. En ce sens, bien que vous trouverez quelques fois des explications plus importantes sur la nature d'un problème, la majorité du temps on supposera que vous possédez déjà les connaissances économétriques liées à la section consultée.

Chaque section présente rapidement le but de l'opération qui y est traité. Les commandes appropriées sont ensuite présentées, d'abord individuellement, puis dans le cadre d'exemples concrets. Il existe deux versions de ce guide, celle-ci pour Matlab et une autre pour Stata. Comme il s'agit d'un manuel encore en développement, des changements lui seront constamment apportés en cours de session et la version distribuée sera toujours la plus récente. Tout commentaire, suggestion ou correction sera bienvenu et apprécié.

Prenez note que ce texte décrit seulement certaines fonctions ainsi que leurs options les plus souvent utilisées pour le genre de recherches effectuées au bac et à la maîtrise en économie, il n'est donc pas du tout exhaustif. Un conseil: apprenez à utiliser l'aide de Matlab. Il s'agit là d'un outil fort utile pour découvrir de nouvelles fonctions ou pour connaître l'ensemble des options disponibles pour les fonctions décrites dans ce guide.

Les fonctions sont présentées dans le format suivant:

1. Le nom de la fonction;
2. Le format d'entrée;
3. Sa description;
4. Ses options (s'il y en a);
5. Un exemple court.

La majorité des chapitres se terminent par une section donnant un exemple plus long et plus concret d'applications des informations présentés dans le chapitre.

La nomenclature suivante est suivie dans ce guide:

- Le texte en `machine à écrire` désigne les fonctions dans leur forme générique.
- Le texte en *italique* désigne les variables et autres chaînes de caractères qui doivent être remplacées.
- Le texte en sans serif désigne le texte tel qu'il serait entré à l'ordinateur.

Les chapitres 1 à 3 font le tour des commandes de base dans Stata et Matlab, ainsi que leur format de saisie. À la suite de ces chapitres vous devriez être en mesure d'importer, de manipuler, puis d'exporter vos données et de tracer des graphiques.

Les chapitres 4 à 10 abordent quant à eux chacun un sujet spécifique de l'économétrie. Ils prennent donc une approche quelque peu différente puisqu'ils introduisent peu de nouvelles fonctions, se concentrant plutôt sur la démarche à adopter pour effectuer l'opération en question.

Finalement, les deux derniers chapitres (11 et 12) sortent quelque peu du cadre de ce guide en abordant respectivement la manipulation des données par Word et Excel et la recherche de données. Ces chapitres ont pour but de vous aider dans le cadre plus général de la production d'un travail de recherche.

Chapitre 1

Commandes générales importantes

1.1 Importation des données

Note importante sur l'importation de données:

Lorsque vous faites le transfert des données, vous devez vous assurer que le format de celles-ci est compatible avec Matlab. Outre les séparateurs de données qui doivent correspondre à la commande choisie, il faut aussi s'assurer que le séparateur de décimales soit un point (.) et que les milliers ne soient pas séparés par un espace. Consultez la section 11 pour plus de détails à ce sujet.

`load`
`load` $nom\_de\_fichier$[1]
`load` fonctionne seulement avec les données numériques, mais a l'avantage d'être la commande d'importation de données la plus simple de Matlab. `load` importe le fichier spécifié dans une matrice du même nom. Les données doivent être séparées par des espaces ou des tabulations.

ex: importe les données du fichier 'monfichier.dat' dans la matrice 'monfichier'
load monfichier.dat

Note: `load` devrait être suffisant pour les besoin du cours ECN 3949.

`xlsread`
$[A, B]$ = `xlsread`($nom\_de\_fichier$)
`xlsread` lit la première feuille d'un tableur Microsoft Excel ('.xls') et renvoit toutes les

[1] Puisque le dossier de travail est déjà spécifié dans Matlab, il est inutile de donner le chemin d'accès avec le nom des fichiers. Les extensions doivent par ailleurs être présentes.

données numériques dans la matrice A et toutes les données texte dans la matrice B de même taille. Les élément de texte sont remplacés par 'NaN' dans la matrice A et les éléments numériques sont laissés vides dans la matrice B.
Si `xlsread` rencontre des données texte dans la première rangée et la première colonne, la matrice B ne contiendra que ces valeurs.

ex: importe les données du fichier 'tableur.xls' dans les matrices 'num'
[num,txt] = xlsread('tableur.xls')

`textread`
$[A, B, C \ldots]$ = `textread(`'$nom\_de\_fichier$', '$format$'`)`
`textread` permet d'importer des données d'un fichier formatté $nom\_de\_fichier$ vers les variables spécifiées A, B, C, etc. $format$ est une chaîne de caractères et spécifie le format des variables à importer, il doit y avoir un format par variable. N'importe quel séparateur de données peut-être utilisé par le fichier.
Les formats possibles sont:

- `%u`: données numériques, entier positif
- `%f`: données numériques non-entiéres
- `%s`: texte
- `%q`: texte, si entre guillemets (' "xyz" '), renvoie seulement le texte ('xyz')
- 'lettres': ignore le texte écrit; par exemple si on a 'etage6', " '$format$' = 'etage%u' renverait '6' "

Il est suggéré de se limiter à `%f` et `%q` qui sont amplements suffisants dans le cadre du cours.
Options: `headerlines`: spécifie un nombre de lignes à sauter au début du fichier.
$[A, B, C, \ldots]$ = `textread(`'$fichier$','$format$'`,'headerlines', #)`

ex: fmt = '%f%f%f%q'
[y,var1,var2,var3] = textread('ECN3949TP1.txt',fmt,headerlines,2)

`dlmread`
M = (`'`$nom\_de\_fichier$`', `*'séparateur de données'*)
Moins flexible que `textread`, mais plus que `load`, `dlmread` fonctionne seulement avec les données numériques, mais a l'avantage d'être plus simple. `dlmread` importe le fichier spécifié dans la matrice M.
'séparateur de données' spécifie le type de séparateur dans le fichier, il peut prendre les valeurs suivantes:

- `','`: virgule, valeur par défaut

- ';': point-vigule
- '\t': tabulation
- ' ': espace

  ex: M = ('ECN3949TP1.txt','\t')

1.2 Manipulation des données

1.2.1 Opérateurs mathématiques

Addition:	+
Soustraction:	-
Multiplication:	*
Division:	/
Puissance:	^

1.2.2 Opérateurs logiques et de comparaison

ET:	&
OU:	|
Non (¬):	~
Égal:	==
Différent:	~=
Plus grand:	>
Plus petit:	<
Plus grand ou égal:	>=
Plus petit ou égal:	<=

1.2.3 Fonctions et Opérateurs matriciels

Note:

Matlab étant un logiciel de manipulations de matrices, il n'y a pas de distinction faite entre une matrice et une variable comme c'est le cas avec d'autres logiciels économétriques (Stata par exemple). Les variables sont automatiquement traité comme des vecteurs dans Matlab.

Création de matrices

Les matrices et les vecteurs sont crées dans Matlab en inscrivant directement le nom de la matrice et les opérations qu'on désire effectuer séparées par le signe égal ('=').

ex:
A = [1,2,3,4;5,6,7,8]; où les virgules séparent les colonnes et les points-virgules séparent les rangées. (matrice 2×4)
B = A*2
C = A^2

Fonctions matricielles

- Créer une matrice identité 2×2: I = `eye(2)`
- Créer une matrice 4x5 dont chaque élément égale à zéro: Z = `zeros(4, 5)`
- Créer un vecteur 1x6 dont chaque élément égale à un: O = `ones(1, 6)`
- Créer une matrice 2x2 dont chaque élément est un aléa U(0,1): U = `rand(2)`, équivalent à U = `rand(2, 2)`
- Créer une matrice 3x8 dont chaque élément est un aléa N(0,1): N = `randn(3, 8)`
- Obtenir le produit kronecker de deux matrices: K = `kron(`A`,`B`)`
  Le produit kronecker, $\otimes$, est défini par:

$$K_{T^*m \times K^*n} = A_{T \times K} \otimes B_{m \times n} = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1K}B \\ a_{21}B & a_{22}B & \cdots & a_{2K}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{T1}B & a_{T2}B & \cdots & a_{TK}B \end{bmatrix}$$

Opération sur les matrices

- Extraction d'une sous-matrice:
  - Extraire la deuxieme colonne de A: A(:,2)
  - Extraire la deuxieme rangée de A: A(2,:)
  - Extraire l'élément A_{12}: A(1,2)
  - Extraire le vecteur (A_{12}, A_{13}, A_{14}): A(1,2:4)
- Empilage de matrices

  - Empiler horizontalement (mettre les rangées une sur l'autre): `[A,A]` (4×4)
  - Empiler verticalement (mettre les colonnes une à côté de l'autre): `[A;A]` (2×8)

- Opération élément par élément: opérateur mathématique précédé d'un point ('.')
  - Multiplication de A par B élément par élément: `A.*B`

- Opérations conditionnelles
  - Retirer les éléments ne répondant pas a une condition logique (peut donner un vecteur): `X = X(X > 5)`
  - Transformer les éléments répondants a une condition logique: `X(X > 5) = 0`
  - Modifier une rangée / une colonne complète: `X(X(:,1) == 4,:)  = 0`
  - Retirer une rangée / une colonne complète: `X(X(:,1) == 4,:)  = []`

- Inverse: `Inv(`A`)`

- Transposée: A'

- Matrice diagonale $n \times n$, avec pour diagonale les éléments de V, où V est un vecteur $n \times 1$ ou $1 \times n$: `diag(`V`)`

- Extraire la diagonale d'une matrice carrée A sous forme de vecteur: `diag(`A`)`

1.2.4 Autres trucs utiles

- Création d'une variable binaire: g `= (`*condition_logique*`)`;
  ex: `d = (y >= 0)`

- Création d'un vecteur dont les éléments sont une suite: v `=` *début*`:`*incrément*`:`fin;
  ex: `x = 1.0;0.1;2.0` produit le vecteur [1.0,1.1,1.2, ...,2.0];
  notez que l'argument *incrément* est facultatif, ainsi `x = 1:10` produit le vecteur [1,2, ...,10].

1.3 Autres transformations des variables

Soit un vecteur z $1 \times n$:
`exp(`z`)`: exponentielle de z élément par élément
`log(`z`)`: logarithme naturel de z élément par élément
`sqrt(`z`)`: racine carrée de z élément par élément

`mean(`z`)`: moyenne des éléments du vecteur z
`std(`z`)`: écart-type des éléments du vecteur z
`var(`z`)`: variance des éléments du vecteur z
`sum(`z`)`: somme des éléments du vecteur z
`cumsum(`z`)`: somme cumulative des éléments du vecteur z (retourne un vecteur $1 \times n$)
`min(`z`)`: renvoie l'éléments du vecteur z ayant la valeur la moins élevée
`max(`z`)`: renvoie l'éléments du vecteur z ayant la valeur la plus élevée
`length(`z`)`: nombre de colonnes du vecteur z
`size(`z`)`: renvoie un vecteur 1×2 contenant la taille de la matrice (rangées,colonnes)

1.4 Boucles et tests logiques

Les boucles et les tests logiques ('si') se montrent utiles lorsqu'on désire construire des programmes plus complexes. Attention de ne pas en abuser! Bien souvent le même résultat peut être obtenu par des opérateurs matriciels, ce qui est beaucoup plus efficace.

`for`
`for` *variable* `=` *valeurs*
opérations à effectuer
`end`
Effectue les *opérations* pour chacune des valeurs spécifiées de la *variable*.

`while`
`while` *condition logique*
opérations à effectuer
`end`
Effectue les *opérations* tant que la *condition logique* est vraie.

ex: trois façons d'effectuer une multiplication élément par élément de deux vecteur 1xT (la troisième est bien sûr la meilleure).

1. for i = 1:T
   C(1,i) = A(1,i)*B(1,i)
   end

2. while i ¡= T
   C(1,i) = A(1,i)*B(1,i)
   i = i + 1

```
end
```

3. C = A.*B

`if`
`if` *condition logique*
opérations à effectuer si vrai
`else`
opérations à effectuer si faux
`end`
Vérifie la *condition logique* et effectue l'opération correspondante au résultat du test logique. Notez que la section "`else`, *opération si faux*" est facultative.

ex: deux façons de transformer les éléments d'une matrice TxK en log et de contrôler pour les valeurs égales à zéro (la deuxième est bien sûr la meilleure).

1.
```
for i = 1:T
for j = 1:K
if A(i,j) == 0
A(i,j) = log(0.000001)
else
A(i,j) = log(A(i,j))
end
end
end
```

2.
```
A(A == 0) = log(0.000001)
A(A != 0) = log(A)
```

1.5 Divers

Commentaires
Il est possible d'insérer des commentaires dans son programme en prenant soin de débuter la ligne de commentaire par le symbole '%'.

ex: % Ceci est un commentaire.

Suppression de la sortie
Pour que Matlab effectue une opération sans que l'on voit le résultat, il suffit de terminer la ligne de commande par un point-virgule (';').

ex: B = [8,7,6,5;4,3,2,1]; N'affichera pas la matrice B.

Commande sur plusieurs lignes
Il est possible d'écrire une commande sur plusieurs lignes. Les '...' indiquent à Matlab que la commande se poursuit à la ligne suivante.

ex:
[v1,v2,v3,v4,v5,v6,v7,v8,v9,v10,v11,v12,v13,v14,v15,v16] = ...
textread('TP1donnees.txt', fmt);

1.6 Exemples et résultats pour le Chapitre 1

Lecture des données à partir du fichier *Donnees.dat* et manipulation des variables dans le but de faire une régression. Le fichier *Donnees.dat* contient 5 variables, comptant chacune 100 observations.

```
load Donnees.dat;
% Attribution de noms à certaines des variables.
px = Donnees(:,1);
qt = Donnees(:,2);
% Construction d'une variable binaire: le rapport de la 3e sur la 4e variable doit
% être inclu dans l'ensemble [0,25;0,5]et la 5e variable doit être égale
% à 1 pour que la variable binaire égale 1.
bin = (0.25 = Donnees(:,3)/Donnees(:,4) = 0.5 & Donnees(:,5) == 1);
% Création d'une variable indice.
no = [1:100]
```

Chapitre 2

Visualisation des données

2.1 Impression/Exportation des données

2.1.1 Impression à l'écran

Matlab affiche par défaut toutes les opérations effectuées à l'écran. Pour éliminer la sortie, se référer à la section 1.5.

2.1.2 Exportation

`diary`
Permet de sauvegarder une copie de sa session dans le fichier spécifié. Doit être suivi de `diary('off')` à la fin du programme.
`diary('`*nom_de_fichier*`')`
programme
`diary('off')`

`wk1write`
`wk1write('`*nom_de_fichier*`',`*M*`)`
Sauvegarde la matrice M dans un fichier de tableur *nom_de_fichier*.wk1; aucune extension ne doit donc être spécifiée.

2.1.3 Impression

La façon la plus pratique d'imprimer ses résultats est d'utiliser la fonction `diary`, puis de traiter le fichier de sortie avec son traitement de texte préféré.

2.2 Exemples et résultats pour le Chapitre 2

Reprenons l'exemple du chapitre 1, en incluant cette fois les fonctions du chapitre 2.

```
diary('ExChap2.out');
load Donnees.dat;
% Attribution de noms à certaines des variables.
px = Donnees(:,1);
qt = Donnees(:,2);
% Construction d'une variable binaire: le rapport de la 3e sur la 4e variable doit
% être inclu dans l'ensemble [0,25;0,5] et la 5e variable doit être égale
% à 1 pour que la variable binaire égale 1.
bin = (0.25 = Donnees(:,3)/Donnees(:,4) = 0.5 & Donnees(:,5) == 1);
# Création d'une variable indice.
no = [1:100]
diary('off')
```

Chapitre 3

Graphiques

`plot`
Trace des nuages de point en deux ou trois dimensions.
`plot(`$x, y,$`'`*couleur_style_marqueur*`',`$x, y2,$`'`*couleur_style_marqueur_de_*`2',`$x, y3,$ `...)`
Où '*couleur_style_marqueur*' est une chaîne de caractère optionnelle contenant de un à quatre caractères et qui définit la couleur de la série, le style de la ligne et le type de marqueur des points.
Les possibilités sont:

- Couleurs:
  - `'c'`: cyan
  - `'m'`: magenta
  - `'y'`: jaune
  - `'r'`: rouge
  - `'g'`: vert
  - `'b'`: bleu
  - `'w'`: blanc
  - `'k'`: noir
- Styles:
  - `'-'`: solide
  - `'--'`: tiret (???)
  - `':'`: pointillé
  - `'-.'`: tiret-point
  - `'none'`: pas de ligne

- Marqueurs:
  - `'+', 'o', '*', 'x'`: marqueur du même signe
  - `'s'`: carré (plein)
  - `'d'`: losange (plein)
  - `'^','v','<'`: triangle vers le haut, le bas ou la gauche respectivement (plein)
  - `'p'`: pentagone (plein)
  - `'h'`: hexagone (plein)

Il est possible de donner une seule variable, Matlab fait alors un graphique de cette variable par rapport à l'indice des observations.

ex:
`plot(y)`: courbe de y par rapport à l'indice des observations
`plot(x,y)`: courbe de y par rapport à x, options standard (ligne pleine,pas de marqueur)
`plot(x,y,'k+')`: courbe de y par rapport à x, marqueurs en `'+'`, noirs (pas de ligne)
`plot(x,y,'b-d',x,y2,'r:')`: courbe de y par rapport à x et de $y2$ par rapport à x (sur le même graphique), première courbe bleue, pleine, marqueurs en losanges, deuxième courbe rouge, pointillée (pas de marqueurs)

Il est possible une fois le graphique tracé de modifier une foule de ses paramètres, voici quelques-unes de ces fonctions:

`xlabel('titre')`: spécifie le titre de l'axe des x
`ylabel('titre')`: spécifie le titre de l'axe des y
`title('titre)'`: spécifie le titre du graphique.
`axis`: permet de définir les propriétés des axes, voir l'aide de Matlab pour plus de détails.
`grid`: permet de définir les propriétés des axes, voir l'aide de Matlab pour plus de détails.

`hold`
Permet de superposer plusieurs graphiques en spécifiant `hold on`.
Pour ne plus superposer les graphiques suivant, il suffit de spécifier `hold off`.
`plot(`x`,`y`)`
`hold on`
`plot(`x`,`$y2`)`
`plot(`x`,`$y3`)`
`hold off`

Pour sauvegarder un graphique, il suffit de sélectionner `Save` dans le menu `File` de la fenêtre

du graphique.

3.1 Exemples et résultats pour le Chapitre 3

```
diary('ExChap3.out');
% création d'un indice de temps commençant à 4.
t = 4:103;
% création du log de cet indice.
lnt = log(t);
plot(t,lnt,'k')
plot(t,lnt,'k-')
plot(t,lnt,'k-p')
hold on
plot(t,-lnt,'b-+')
xlabel('temps')
ylabel('log du temps')
title('courbe logarithmique')
diary('off')
```

Chapitre 4

Régressions Simples

Dans ce chapitre nous considérerons le modèle suivant:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + uy = X\beta + U$$

4.1 MCO

Définition des variables:
X = [ones(size(x1),x1,x2,. . . ,xk]
[T,K] = size[X]
T et K sont respectivement le nombre d'observations et le nombre de variables indépendantes.
Procédure pour calculer les moindres carrés ordinaires (MCO):
Calcul des coefficient:
b = inv(X'*X)*X'*y
où X est une matrice $T \times K$, Y est un vecteur $t \times 1$ et b un vecteur $k \times 1$
Calcul des résidus:
e = y-X*b
Calcul de la variance des résidus:
sigma2 = (e'*e)/(T-K)
Calcul de la variance des coefficients:
varb = sigma2 * inv(X'*X)
Calcul de l'écart-type des coefficients:
etype = sqrt(diag(varb))
Création d'un vecteur de y-moyen:
ybar = [ones(size(y))*mean(y)]
Vecteur des y prédits:
yhat = X*b
Variation expliquée:
SSE = sum((yhat-ybar).^2)

Variation totale:
SST = sum((y-ybar).^2)
Calcul du R^2:
R2 = SSE/SST

4.1.1 Tests d'hypothèses et Intervalles de confiance

Test de H_0,$\beta_2 = 0$:
b2 = b(3)
etype2 = etype(3)
t2 = b2/etypeb2
On obtient ainsi la valeur de la statistique t.

Intervalle de confiance à 95% de β_2:
bornesupp = b2 + etype2*1.96
borneinf = b2 + etype2*(-1.96)

Fonctions dans Matlab

La notion de fonction est introduite, car vous effectuerez des MCO tellement souvent qu'il vous est suggéré d'en faire une fonction dès maintenant.

Une fonction est un **fichier matlab indépendant** qui contient du code à être exécuté lorsque son nom est invoqué.
La première ligne du fichier doit être sous la forme suivante:
`function` $[varo1, varo2, \dots]$ = $Nom\_de\_fonction(vari1, vari2, \dots)$
où *varo* désigne une variable retournée par la fonction et *vari* désigne une variable utilisée par la fonction.
Les lignes subséquentes contiennent le code à être exécuté.
La fonction peut ensuite être appelée par la commande suivante:
$[vars1, vars2, \dots]$ = $Nom\_de\_fonction(vare1, vare2, \dots)$
où *vars* sont les nom qu'on désire utiliser pour les variables *varo* et *vare* le nom de variables que l'on donne à la fonction.

ex: MCO.m
```
function [b,e,etype,R2] = MCO(X,y)
[T,K] = size(X);
b = inv(X'*X)*X'*y;
```

```
e = y-X*b;
sigma2Ch = (e'*e)/(T-K);
varChb = sigma2Ch * inv(X'*X);
etype = sqrt(diag(varChb));
ybar = [ones(size(y))*mean(y)];
yCh = X*b;
SSE = sum((yCh-ybar).^2);
SST = sum((y-ybar).^2+);
R2 = SSE/SST;
```

Cette fonction serait appelée par la commande suivante:
`[beta,u,etype,R2] = MCO(Z,y)`
où $beta, u, etype$ et $R2$ sont les noms de variables qui seront utilisées dans le code et Z et y sont les variables contenant les données à être traitées. Z prend généralement la forme `Z = [ones(size(x1)),x1,x2,...]`.

4.1.2 Estimateurs de Variance Robustes

Il peut être utile de pouvoir calculer une matrice de variance-covariance qui soit robuste à la présence d'hétéroscédascticité. L'estimateur robuste de cette matrice le plus utilisé est celui de **White** qui est donné par la formule suivante:

$$\hat{Var}[b] = \frac{1}{T}\left(\frac{X'X}{T}\right)^{-1}\left(\frac{1}{T}\sum_{i=1}^{T} e_i^2 x_i x_i'\right)\left(\frac{X'X}{T}\right)^{-1}$$

En matlab, toujours en utilisant les résultats de la fonction MCO.m, on devrait écrire:
```
e2 = e.^2
varWhite = T*inv(X'*X)*sum((e2.*X)*X')/T*inv(X'*X)
```

4.2 Tests d'hétéroscédasticité

La présence d'hétéroscédasticité ne vient pas biaiser vos résultats, elle biaise plutôt les écarts-types obtenus par MCO. Il existe plusieurs méthodes similaires de tester pour la présence d'hétéroscédasticité. La plus simple est le **test de Breusch-Pagen**:

1. récupérer les résidus de la régression qu'on désire tester;
2. générer le carré des résidus;
3. régresser la carré des résidus sur les variables dépendantes de la régression originale;

4. tester si les coefficients sont conjointement significatifs (test F ou test LM).

   [b,u,etype,R2] = MCO(X,y)
   u2 = u.^2
   [bu,e,etypeu,R2u] = MCO(X,u2)

   Test F:
   F = (R2u/length(bu'))/((1-R2u)/(length(u2)-length(bu'))) [1]
   Test LM [2]:
   LM = length(u2)*R2u
   Notez bien que le R^2 utilisé est celui de la régression auxiliaire effectuée en 3.

La faiblesse du test de Breusch-Pagan est qu'il suppose les erreurs normalement distribuées. Afin de laisser tomber cette hypothèse, il suffit d'ajouter le carré des variables dépendantes et leurs produits croisés dans la régression de l'étape 3, il s'agît là du test de White. Afin de limiter le nombre de régresseurs, on peut utiliser un **test de White** légèrement modifié:

$$u^2 = \beta_0 + \beta_1 \hat{y} + \beta_2 \hat{y}^2 + e$$

On procède pour le reste exactement de la même façon que pour le test de Breusch-Pagan.

Que faire lorsque vous trouvez la présence d'hétéroscédasticité? Deux options s'offrent à vous:

- Calculer des variances robustes par la méthode de White[3]
- Estimer le modèle par MCG, i.e. modéliser la forme d'hétéroscédasticité (voir le chapitre 7).

4.3 Test de Changement structurel (Test de Chow)

Considérez le modèle suivant:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + u$$

[1] Rappel: la forme générale du test F pour la signification conjointe de tous les coefficients est: $F = \frac{R^2/k}{(1-R^2)/(n-k-1)}$

[2] Rappel: la statistique LM suit une χ^2_k

[3] Il peut-être bien tentant de procéder systématiquement avec les variances robustes Eicker-White pour éviter de faire le test d'hétéroscédasticité, mais cette façon de faire réduit la précision de vos résultats (i.e. gonfle les écarts-types et réduit la puissance des tests) lorsque les données sont homoscédastiques.

Le **test de Chow** sert à vérifier s'il existe une différence dans l'influence d'une variable dépendante entre deux groupes de données, i.e. si le coefficient est statistiquement différent. Les deux groupes de données pourraient être deux séries d'observations ou deux périodes de temps par exemple.
La façon "classique" d'effectuer le test de Chow est d'effectuer la régression du modèle pour les deux groupes de façon indépendante et pour les deux groupes ensemble:

$$\begin{aligned} \hat{y_1} &= \beta_{10} + \beta_{11}x_{11} + \beta_{12}x_{12} \\ \hat{y_2} &= \beta_{20} + \beta_{21}x_{21} + \beta_{22}x_{22} \\ \hat{y} &= \beta_0 + \beta_1 x_1 + \beta_2 x_2 \end{aligned}$$

puis de tester si les coefficient sont statistiquement différents par un test F:

$$F = \frac{(S\hat{S}R_y - S\hat{S}R_{y_1} - S\hat{S}R_{y_2})/q}{(S\hat{S}R_{y_1} - S\hat{S}R_{y_2})/n_1 + n_2 - 2k)}$$

Rappel: q est le nombre de contraintes et k le nombre de coefficients, ici $q = k = 3$
Une autre façon plus rapide d'effectuer ce test est de construire une variable binaire égale à un pour les observations du deuxième groupe et de faire une seule régression sur les variables originales et sur les termes d'interaction avec la variable binaire[4]:

Soit δ la variable binaire:

$$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 \beta_3 \delta + \beta_4 x_1 \delta + \beta_5 x_2 \delta$$

On désire maintenant tester si $\beta_0 = (\beta_0 + \beta_3)$, si $\beta_1 = (\beta_1 + \beta_4)$ et si $\beta_2 = (\beta_2 + \beta_5)$. Ce qui revient à tester si β_3, β_4 et β_5 sont conjointement différent de 0. Ceci peut être facilement effectué par un test de F.

ex:

```
g2 = (groupe ==2);
g2x1 = g2*X(:,2);
g2x2 = g2*X(:,3);
Xg = [X,g2,g2x1,g2x2];
function [b,u,etype,R2] = MCO(X,y);
function [bg,ug,etypeg,R2g] = MCO(Xg,y);

# Test F:
F = (R2 - R2g/length(b'))/((1-R2g)/(length(u)-length(b')))
```

[5]

[4]Cette section et l'exemple qui la suit sont inspirés de la rubrique de l'aide de Stata: *How can I compute the Chow test statistic?* par Bill Gould.

[5]Rappel: la forme générale R^2 du test F est: $F = \frac{(R_u^2 r - R_r^2)/k}{(1-R_u^2 r)/(n-k-1)}$

Chapitre 5

Variables instrumentales et Doubles Moindres Carrés

Lorsqu'une variable "indépendante" est corrélée avec le terme d'erreur, les hypothèses classiques du modèle linéaire sont violées et on se retrouve face à un problème d'**endogénéité**. Dans ces cas, on peut faire appel à l'**estimateur de variables instrumentales** (VI) ou aux **doubles moindres carrés ordinaires** (DMCO).

5.1 Estimateur Variables Instrumentales

Soit Z, une matrice de VI et X, la matrice originale. L'estimateur VI est donné par:

$$\hat{\beta}_{(VI)} = (Z'X)^{-1}Z'y$$

et l'estimateur VI de la covariance par:

$$\hat{\sigma}^2(Z'X)^{-1}(Z'Z)(X'Z)^{-1}$$

où

$$\hat{\sigma}^2 = \frac{1}{T}(y - X\hat{\beta}_{(IV)})'(y - X\hat{\beta}_{(IV)}).$$

ou, lorsque $J > K$ (J étant le nombre de VI et K le nombre de variables indépendantes), par:

$$\hat{\beta}_{(IV)} = [X'Z(Z'Z)^{-1}Z'X]^{-1}X'Z(Z'Z)^{-1}Z'y.$$

$$\hat{\sigma}^2[X'Z(Z'Z)^{-1}Z'X]^{-1}.$$

Créons la fonction `VI` pour l'estimateur VI:
function [bvi,evi,etypevi,varChbvi,R2vi] = VI(X,Z,y)

[T,K] = size(X);
bvi = inv(X'*Z)*X'*y;
evi = y-Z*bvi;
sigma2Ch = (evi'*evi)/(T-K);
varChbvi = sigma2Ch * inv(X'*Z)*X'*X*inv(Z'*X);
etypevi = sqrt(diag(varChbvi));
ybar = [ones(size(y))*mean(y)];
yCh = Z*bvi;
SSEvi = sum((yCh-ybar).$\hat{2}$);
SSTvi = sum((y-ybar).$\hat{2}$);
R2vi = SSEvi/SSTvi;
Notez que cette fonction est optimale seulement pour $K \leq J$.

5.2 DMCO

Le principe des doubles moindres carrés ordinaires est d'utiliser une estimation de la variable endogène qui ne soit pas corrélée avec le terme d'erreur pour effectuer la régression.
Soit le modèle suivant:

$$y_1 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 y_2 + u$$

et soit z une VI de y_2.

Comme leur nom l'indique, les DMCO se font en deux étapes.

1. Estimation de la variable endogène:
Régression de y_2 sur toutes les variables indépendantes (x_1 et x_2 ici) et la/les VI pour y_2 (z ici).
On récupère $\hat{y_2}$, l'estimation linéaire de y_2.

2. Régression du modèle avec $\hat{y_2}$:
Régression de y_1 sur une constante, x_1, x_2 et $\hat{y_2}$.

Cette dernière régression ne souffrant plus d'endogénéité, les $\hat{\beta}$ ainsi obtenus sont non-biaisés.

Z = [ones(size(y1)),x1,x2,z]
[bz,uz,etypez,R2z] = MCO(Z,y2)
y2hat = Z*bz

X = [ones(size(y1)),x1,x2,y2hat]
[b,u,etype,R2] = MCO(X,y1)

5.3 Tests d'endogénéité

Le **test de Hausman** permet de vérifier s'il existe bel et bien une différence entre l'estimateur VI et l'estimateur MCO, vérifiant ainsi s'il y a bel et bien endogénéité des variables (si les deux estimateurs sont consistants, ils seront asymptotiquement égaux). Sous H_0, la statistique de Hausman est:

$$H = [\hat{\beta}_{(VI)} - b]'[\hat{\sigma}^2[(X'Z(Z'Z)^{-1}Z'X]^{-1} - \hat{\sigma}^2(X'X)^{-1}]^{-1}[\hat{\beta}_{(VI)} - b] \sim \chi^2(J)$$

En utilisant la fonction développée à la section précédente:
[bmco,umco,etypemco,varChbmco,R2mco] = MCO(X,y)
[biv,uiv,etypeiv,varChbiv,R2iv] = IV(X,y)
H = (biv-bmco)'*(varChbiv-varChbmco))'*(biv-bmco)

Chapitre 6

Estimateur du Maximum de Vraissemblance (EMV)

La **fonction de vraisemblance** est la probabilité jointe des observations étant donné les paramètres d'intérêts, i.e.:

$$L(\theta|y) = f(y_1, \ldots, y_n|\theta) = \prod_{i=1}^{n} f(y_i|\theta)$$

L'**estimateur du maximum de vraisemblance** (**EMV**) a pour but de choisir le vecteur de paramètres θ qui maximise la fonction de vraisemblance, i.e. pour lequel les données observées sont les plus probables. Pour simplifier les choses, la fonction de log-vraisemblance,$\mathcal{L}(\theta|y)$, est généralement utilisée[1].
Prenons l'exemple d'un échantillon normalement distribué, de moyenne 0 et de variance σ^2:

$$\begin{aligned} f(y|X, \beta, \sigma^2) &= \prod_{t=1}^{T} (2\pi\sigma^2)^{-1/2} \exp[(y_t - x_t'\beta)^2] \\ &= (2\pi\sigma^2)^{-T/2} \exp\left[-\frac{(y - X\beta)'(y - X\beta)}{2\sigma^2}\right]. \end{aligned}$$

La log-vraisemblance est

$$\mathcal{L}(\beta, \sigma^2) = -\frac{T}{2}\log(2\pi) - \frac{T}{2}\log\sigma^2 - \frac{(y - X\beta)'(y - X\beta)}{2\sigma^2}.$$

Les CPO sont:

$$\frac{\delta lnL}{\delta\beta} = \frac{(y - X\beta)(y - X\beta)}{2\sigma^2}$$

[1]Le logarithme étant une fonction montone, la valeur qui maximise $\mathcal{L}(\theta|y)$ est la même que celle qui maximise $L(\theta|y)$.

$$\frac{\delta lnL}{\delta \sigma^2} = -\frac{T}{2\sigma^2} + \frac{(y - X\beta)'(y - X\beta)}{2\sigma^4}$$

Ce qui nous permet de trouver

$$\hat{\beta} = (X'X)X'y$$

$$\hat{\sigma^2} = \frac{(y - X\beta)'(y - X\beta)}{T} = \frac{\hat{e}'\hat{e}}{T}$$

Il n'existe pas de fonction de maximisation dans Matlab, il faut donc minimiser la négative de la fonction à l'aide de la fonction suivante:

`fminsearch`
$[x, fval]$ `= fminsearch(`$fonction, options, variables$`)`
où x est la variable qui contiendra la/les valeur(s) qui minimise(nt)la fonction, $fval$ est une variable facultative où seront stockées les valeurs de la fonction pour chacune des valeur minimale, $fonction$ est le nom de la fonction à minimiser (réfère à un fichier '.m') et *variables* est une liste de variables utilisées par la fonction, mais qui ne doivent pas être minimisées. Les options possibles sont:

- `Display`: ajuste l'affichage de la progression de la minimisation; se référer à l'aide de Matlab pour plus d'info.
- `MaxFunEvals`: nombre maximal d'évaluations de la fonction permises
- `MaxIter`: nombre maximal d'itérations permises

Si aucune option n'est désirée, il faut utilisée l'ensemble vide `[]` pour *options*.

ex: voir section 8.1

Chapitre 7

Moindres Carrés Généralisés

La méthode des **moindres carrés généralisés** (**MCG**) cherche à modéliser la fonction de la variance. Nous obtenons alors l'estimateur MCG

$$\hat{\beta}^{MCG} = (X'V^{-1}X)^{-1}X'V^{-1}y$$

ou encore

$$\hat{\beta}^{MCG} = (X'W^{-1}X)^{-1}W'V^{-1}y$$

et sa variance est

$$\text{var}[\hat{\beta}] = \sigma^2(X'V^{-1}X)^{-1}.$$

où V et W sont égaux à

$$W = \sigma^2 \begin{bmatrix} x_1 & 0 & \cdots & 0 \\ 0 & x_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & x_n \end{bmatrix} \equiv \sigma^2 V$$

Le principe de base ici est de construire la matrice V avec la forme appropriée de variance pour pouvoir ensuite calculer les $\hat{\beta}^{MCG}$.
Voyons comment on estimerait le modèle $y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \varepsilon$ où l'on suppose que la variance suivre une fonction de type $\sigma^2(x) = \sigma^2 x_{2i}$:

```
X = [ones(size(y)),x1,x2];
V = diag(x2);

[T,K] = size(X);

bGLS = inv(X'*inv(V)*X)*X'*inv(V)*y;
eGLS = y-X*bGLS;
```

sigma2ChGLS = (eGLS'*eGLS)/(T-K);
varChbGLS = sigma2ChGLS * inv(X'*inv(V)*X);
etypeGLS = sqrt(diag(varChbGLS));

ybar = [ones(size(y))*mean(y)];
yChGLS = X*bGLS;
SSEGLS = sum((yChGLS-ybar).^{2});
$SST = sum((y - ybar).^2)$;
$R2GLS = SSEGLS/SST$

Chapitre 8

Variables dépendantes qualitatives

8.1 Probit/Logit

Un probit et un logit s'appuient en fait sur le même principe, ils ne diffèrent que dans la forme de la fonction de répartition qu'ils utilisent pour calculer l'effet sur la probabilité d'une variation de la variable latente. En effet, lorsque la variable dépendante ne prend que des valeurs qualitatives (oui ou non par exemple), l'effet d'une variable indépendante sur la probabilité de dire oui doit être "traduit" par une fonction de répartition. Cette dernière nous donne la probabilité associée à une valeur donnée de la valeur latente exprimée par la combinaison linéaire des variables indépendantes.

Pour le probit, on doit utiliser la fonction de répartition de la loi normale:

$$F(x) = \Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-z^2}{2}} dz$$

`normcdf`
`normcdf`(X)
Donne la valeur de la fonction de répartition de la loi normale pour la valeur de X spécifiée.

Pour le logit, on utilie plutôt la fonction de répartition de la loi logistique:

$$F(x) = \frac{1}{1 + e^{-x}}$$

Comme il n'existe pas de fonction prédéfinie dans Matlab pour la fonction de répartition de la loi logistique, on doit en construire une:

```
function [Fx] = logitcdf(x)
Fx = 1/(1+exp(-x))
```

La procédure étant la même pour un probit et un logit, elle ne sera démontrée qu'une seule fois pour le cas de la loi logistique (logit).

Voyons d'abord comment construire une fonction de log-vraisemblance pour une loi normale:

```
function LL = logvrais(b,X,y)
[T,K] = size(y);
lXb = logitcdf(X*b') L = y.*log(lXb) + (ones(T,1)-y).*log(ones(T,1)-lXb);
LL = -sum(L)
```

Voyons maintenant comment il nous est possible d'utiliser cette fonction pour trouver les $\hat{\beta}$ par maximum de vraisemblance:

```
options = []
bmin = fminsearch(@logvrais,[0,0,0,0,0,0,0,0,0,0,0,0],options,X,y)
```

Il ne nous reste maintenant qu'à calculer l'effet d'une variable sur la probabilité. Ceci peut être fait en fixant toutes les autres variables à une valeur (généralement leur moyenne échantillonalle), puis en estimant le modèle successivement pour deux valeurs de la variable. La différence de la probabilité de ces deux estimations linéaires de la variable dépendante ($\hat{y}$) donne la variation de probabilité.

ex: le modèle estimé est $\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 bin1 + \mu$ où $bin1$ est une variable binaire.

```
options = []
bmin = fminsearch(@logvrais,[0,0,0,0,0,0,0,0,0,0,0,0],options,X,y)
yhat1 = bmin(1) + bmin(2)*mean(x1) + bmin(3)*mean(x2) + bmin(4)
yhat2 = bmin(1) + bmin(2)*mean(x1) + bmin(3)*mean(x2)
diff = normcdf(yhat1)-normcdf(yhat2)
```

Note: on pourrait aussi calculer l'effet marginal directement en calculant la dérivée de l'espérance de y sachant X:

$$\frac{\partial E[y|X]}{\partial X} = \left\{ \frac{dF(X'\beta)}{d(X'\beta)} \right\} \beta = f(X'\beta)\beta$$

Par exemple, dans le cas d'un probit:

$$\begin{aligned} \Pr(y=1) &= \Phi(X'\beta) \\ \frac{\partial \Pr(y=1)}{\partial X} &= \phi(X'\beta)\beta \end{aligned}$$

8.2 Tobit

Un tobit est essentiellement un modèle dont les données sont censurées. Comme le probit, le tobit suit une loi normale.

MATLAB
Soit un modèle censuré à gauche à zéro (y^* étant une variable latente de y):

$$\begin{aligned} y^* &= X'\beta + \mu \\ y &= \max(0, y^*) \end{aligned}$$

Encore ici, on procédera par maximum de vraisemblance, mais on devra cette fois considérer qu'il existe deux fonctions: une pour $y = 0$ et une pour $y > 0$. La log vraisemblance pour le tobit sera donnée par la somme de la log vraisemblance des deux fonctions:

$$\begin{aligned} &\text{si } y = 0 : \\ &\quad l1(\beta, \sigma) = \log[1 - \Phi(x_i\beta/\sigma) \\ &\text{si } y > 0 : \\ &\quad l2(\beta, \sigma) = \log\{(1/\sigma)\phi[(y_i - x_i\beta)/\sigma]\} \end{aligned}$$

En matlab:

```
ybin = (y == 0)
lvrais = ybin.*log(1-normcdf(X'*b/sqrt(sigma2Ch))) + ...
(ones(T,1)-ybin).*log(1/sqrt(sigma2Ch)*normpdf((y-X'*b)/sqrt(sigma2Ch))
P = -sum(lvrais)
```

Afin de calculer l'espérance de y étant donné x, il suffit de calculer:

$$E(y|x) = \Phi(x\beta/\sigma)x\beta + \sigma\phi(x\beta/\sigma)$$

Le transfert de cette fonction en Matlab ne devrait plus, à ce stade, vous causer aucun problème...

Chapitre 9

Séries Chronologiques

9.1 Opérateurs de séries temporelles

Il n'existe pas d'opérateur de séries temporelles en soit dans Matlab, il faut donc utiliser les opérateurs matriciels vus au chapitre 1 pour introduire des retards ou des avances..

ex:

- Opérateur Retard:
  T = length(X)
  x = X(3:T)
  x_lag1 = X(2:T-1)
  x_lag2 = X(1:T-2)
  Note: il faut ajuster manuellement le nombre d'observations considérées.
- Opérateur Avance:
  T = length(X)
  x = X(1:T-2)
  x_fwd1 = X(2:T-1)
  x_fwd2 = X(3:T)

9.2 Tests d'autocorrélation

Inutile de mentionner que l'autocorrélation est un problème qui n'est pertinent que dans le cas des séries temporelles...
Le test ρ est le test le plus simple à effectuer pour tester la présence d'autocorrélation:

1. récupérer les résidus de la régression qu'on désire tester;

2. régresser $\hat{u}_t$ sur $\hat{u}_{t-1}$ à $\hat{u}_{t-n}$ et X
3. Tester la signification conjointe des coefficients de cette régression par un test F.

   Choisissons n égal à 3.

```
T = length(y)
[b,u,etype,R2] = MCO(X,y)
U = [ones(T-3,1),u(3:T-1),u(2:T-2);u(1:T-3)];
[bu,e,etypeu,R2u] = MCO(U,u(4:T))

Test F:
F = (R2u/length(bu'))/((1-R2u)/(length(u2)-length(bu')-1))
Test LM:
LM = (T-3)*R2u
```

Le test de Durbin-Watson est aussi utilisé pour tester la présence d'autocorrélation, mais comme il est moins précis et ne considère qu'une seule période, nous ne le couvrirons pas ici.

9.3 Méthode de Box-Jenkins

Ce qu'il est important de comprendre, à mon avis, dans la méthode de Box-Jenkins, c'est que l'objectif de toutes les opérations que nous effectuons est de se retrouver avec un résidu qui est un **bruit-blanc**. Le but ultime étant de modéliser la série afin de faire des prédictions, nous pouvons seulement être certain d'avoir tout extrait lorsqu'il nous reste seulement un bruit-blanc: un processus qui est par définission impossible à prédire.

9.3.1 Stationnarité des données

La première étape de la méthode de Box-Jenkins consiste à effectuer les transformations nécessaires afin de s'assurer que notre série est stationnaire, si elle ne l'est pas, il nous sera impossible de travailler dessus.
Première question à se poser: doit-on travailler en log ou pas? Si la variable croît à un taux constant, elle sera linéaire en log. De plus, les propriétés du logarithme font en sorte qu'il "écrase" une variance croissante. Outre la transformation logarithmique, il existe trois cas possibles de non-stationnarité qui impliqueront des changements dans la série (ou sa modélisation):

- Changement structurel
- Tendance déterministe
- Racine unitaire

Changement structurel

Les changement structurels peuvent être détectés à l'aide du Test de Chow (voir section 4.3). Malheureusement, rien ne peut généralement être fait pour stationnariser une série dans le cas d'un changement structurel.

Tendance déterministe

Afin de régler le problème de la présence d'une tendance temporelle, il suffit de la modéliser la tendance. Il faut faire attention de bien choisir la tendance la mieux adaptée à nos données: linéaire, quadratique, logarithmique, etc.

ex: tendance quadratique
```
T = length(y);
t = 1:T;
t2 = t^2;
X = [ones(size(y)),t,t2]
[b,u,etype,R2] = MCO(X,y)
```

Racines Unitaires

On fait face à un problème de racine unitaire lorsque $\rho = 1$ dans le modèle suivant:

$$y_t = \alpha + \rho y_{t-1} + e_t$$

Afin de régler le problème de racine unitaire, il faut différencier la série, i.e. travailler sur $\Delta y_t = y_t - y_{t-1}$ plutôt que y_t. Le modèle devient donc:

$$\Delta y_t = \alpha + \theta y_{t-1} + \epsilon_t$$

ex:
```
T = length(y);
x = y(2:T) - y(1:T-1)
...
```

9.3.2 Sélection de Modèle

À chaque étape de la modélisation de notre série chronologique, il est important de choisir le meilleur des choix qui s'offre à nous: tendance quadratique ou logarithmique? AR(2) ou AR(3)?
Plusieurs critères existent pour nous aider dans nos choix, nous en explorerons trois qui se basent tous sur le principe de pénalité pour le nombre de variables.

R-carré ajusté

Le **R-carré ajusté** ($\bar{R}^2$) est donné par la formule suivante:

$$\bar{R}^2 = 1 - \frac{n-1}{n-K}(1-R^2)$$

Akaike information criterion (AIC)

Voici la formule habituelle du **critère d'Akaike**:

$$AIC(K) = log(\frac{e'e}{n}) + \frac{2K}{n}$$

Bayesian information criterion (BIC)

Voici la formule habituelle du **critère de Schwartz ou Bayesien**:

$$BIC(K) = log(\frac{e'e}{n}) + \frac{Klogn}{n}$$

À la fois dans Matlab et Stata, ces critères doivent être construits manuellement.

Chapitre 10

Données longitudinales (Panel)

Il existe bon nombre de méthodes pour traiter les données en Panel et la littérature sur le sujet est très exhaustive, nous ne traiterons donc dans ce chapitre que des méthodes de base.

10.1 Effets Fixes et Effets Aléatoires

Lorsqu'on a des données longitudinales, on voudra souvent isoler l'effet associé à chaque année, à chaque individu ou aux deux. Il existe deux façons de modéliser ces effets, soit comme des **effets fixes**, soit comme des **effets aléatoires**.
Effets fixes (α_i, μ_t):

$$y_{it} = \alpha_i + \mu_t + X_{it}\beta + e_{it}$$

Effets aléatoires (α_i, μ_t):

$$\begin{aligned} y_{it} &= X_{it}\beta + e_{it} \\ e_{it} &= \alpha_i + \mu_t + \varepsilon_{it} \end{aligned}$$

Les effets fixes ont l'avantage de permettre une corrélation avec les variables explicatives, mais imposent une structure aux effets. À l'inverse, les effets aléatoires seront biaisés s'il y a corrélation avec certaines varibles explicatives, mais permettent beaucoup plus de flexibilité.

10.1.1 Variables binaires

Dans le cas d'un effet fixe, la méthode la plus simple de capter cet effet est de supposer qu'il existe pour chacun de nos groupes et, ainsi, d'ajouter une variable binaire par groupe (sans oublier, comme d'habitude, d'en laisser tomber une). Donc si nous avons cinq groupes et quatre périodes de temps, nous aurons un total de sept variables binaires. Il peut être

préférable dans certains cas de ne pas inclure de constante pour comparer tous les groupes entre eux. Dans le dernier exemple, on pourrait ainsi laisser tomber la constante et inclure cinq variables binaires pour les groupes et trois variables binaires pour les annés.

Ajout manuellement de variables binaires pour chaque groupe et chaque année.

ex: Régression sur cinq échantillons tirés de 1980,81,82 et 83.

```
* création des variables binaires
a81 = (annee == 1981)
a82 = (annee == 1982)
a83 = (annee == 1983)
g2 = (groupe == 2)
g3 = (groupe == 3)
g4 = (groupe == 4)
g5 = (groupe == 5)
* régression
X = [X,a81,a82,a83,g2,g3,g4,g5]
function [b,u,etype,R2] = MCO(X,y)
```

Chapitre 11

Interaction avec les tableurs et les traitements de texte

11.1 Remarques

Quelques remarques importantes lorsque vous travaillez avec des données numériques:

- Pour être utilisables, les données numériques doivent être séparées par variable et par observation, chaque valeur étant séparée de la suivante par un 'séparateur'.
- Généralement, il est plus facile de travailler si les variables constituent les colonnes et les observations les rangées.
- Comme les logiciels sont (tous?) américains, le séparateur de décimales doit être un point ('.') et non pas une virgule (','). Si ce n'est pas le cas, ceci peut facilement être changé par la commande *replace* du menu *Edit* de Excel.
- Assurez-vous que le séparateur de valeurs est compatible avec la méthode utilisée pour importer les données dans le logiciel économétrique.
- Assurez-vous également que si vous avez du texte dans vos observations, cela est permis par votre méthode d'importation.
- Évitez les lignes de commentaire ou de texte avant vos données ou le nom de vos variables. Bien qu'il soit possible de contourner cette difficulté, ça évite souvent des problèmes.

11.2 Tableur

Cette section est surtout orientée vers Excel puisque c'est le tableur le plus utilisé sur le marché.

- Exportation des données: Dans le menu *Fichier*, *Sauvegarder sous*, sélectionnez un format text avec séparateur: soit des tabulations (.txt ou .tab), soit des virgules (.txt ou .csv).

- Importation de données: En ouvrant un fichier ASCII (sans formatage), l'assistant importation-données de Excel devrait s'ouvir automatiquement. Vous devrez alors seulement sélectionner le type de séparateur (étape 2) et le type de données (étape 3: optionnel) pour pouvoir accéder à votre fichier.

11.3 Traitement de texte

Cette section est surtout orientée vers Word puisque c'est le traitement de texte le plus utilisé sur le marché.

- Création de Tableau: Le copier-coller est la solution de choix ici. L'idéal est de passer par Excel après avoir importer le fichier de données (sélectionnez les cases désirées, copier, coller...). Sinon, vous pouvez également copier directement les résultats à partir du gestionnaire de données.

- Insertion de Graphique: Menu *insertion*, *objet*, *du fichier....* Trouvez l'image qui vous sert de graphique et appuyez sur *OK*.

- Transcription de résultats: Malheureusement, il n'existe pas de moyen rapide de transcrire vos résultats s'ils ne peuvent pas être mis en tableau. Copier-coller ou la transcription manuelle demeurent les seuls moyens d'effectuer ce travail... faites des tableaux!

Chapitre 12

Où trouver ses données et comment les extraires

12.1 Liens utiles

12.1.1 À l'Université

Votre premier arrêt pour trouver des données se doit d'être sur le site web de la bibliothèque des sciences humaines au:
http://www.bib.umontreal.ca/SB/num/
Cette page vous donne accès aux plus importantes sources officielles de données numériques, notamment, Statistiques Canada (E-STAT et CHASS), l'institut de la statistique du Québec et l'OCDE. La majorité de ces données sont des séries chronologiques ou des données en panel.

Pour obtenir des données d'enquêtse, il faut se tourner vers *Sherlock* (aussi accessible par cette page) ou vers des organismes privés. Certaines données d'enquêtes sont également accessible par l'institut de la statistique du Québec.

12.1.2 À l'Extérieur

Google www.google.ca
: Moteur de recherche très puissant qui devrait vous aider pour toutes vos requêtes.

Gouvernement du Québec www.gouv.qc.ca
: Plusieurs ministères ont des données télchargeables qui sont accessibles par leur site web.

Gouvernement du Canada www.gc.ca
: Plusieurs ministères ont des données télchargeables qui sont accessibles par leur site web.

Eurostat http://europa.eu.int/comm/eurostat/
: Plusieurs indicateurs économiques et sociaux de l'Union Européenne.

US Census Bureau http://www.census.gov/
: Données des recensement américains.

Agences Nationales de Statistiques http://www.census.gov/main/www/stat_int.html
: Liens vers toutes les agences nationales de statistiques.

Fedstat http://www.fedstats.gov/
: Liens vers la majorité des organismes fédéraux américains produisant des données qui son accessibles.

Données Spatiales http://data.geocomm.com/catalog/
: Plusieurs liens vers des données codées géographiquement.

DataLinks http://www.econ-datalinks.org/
: Une foule de liens vers des données économiques et financières.

Cette liste est TRÈS TRÈS loin d'être exhaustive, donc n'hésitez pas à pousser vos recherches sur internet plus loin. Si vous trouver des liens intéressant, envoyez-les moi et je me ferrai un plaisir de les ajouter aux versions futures de ce guide.

Un conseil: Les sites gouvernementaux et ministériels au niveau national et sous-national sont souvent des mines d'or de données!

12.2 Accès au Données

Cette section décrit seulement comment accéder et télécharger les données de quelques sources particulièrement importantes.

12.2.1 Statcan

http://www.bib.umontreal.ca/SB/num/statcan.htm

Vous avez deux choix pour accéder aux séries chronologiques de Statistiques Canada:

E-Stat Accès à la grande majorité des données de Statistiques Canada, environnement très convivial.

CHASS Accès à envrions 600 000 séries de plus qu'avec E-Stat, mais environnement de navigation moins convivial.

Nous décrirons seulement l'usage de E-Stat ici.

1. Trouvez le tableau qui vous intéresse en effectuant une recherche par sujet ou par mot-clé.
2. Une fois dans le tableau de votre choix, vous aurez généralement à sélectionner des séries spécifiques en choisissant parmis une liste déroulante et ce, pour plusieurs catégories (ex: géographie, fréquence, sexe, dates, etc.).
   Choisissez les séries désirés (tenez la touche `ctrl` enfoncée pour sélectionner plusieurs items dans une même liste) et appuyez sur *série chronologiques*.
3. Choisissez votre format de sortie parmis les choix offerts (suggestion: CSV ou PRN, périodes = lignes).
4. Appuyez sur *extraire* et sauvegarder le fichier qui apparaitra sous le nom désiré.

12.2.2 OCDE

http://www.sourceoecd.org/ (doit être accédé par l'UdeM)

Généralement, les données les plus intéressantes se trouvent dans *statistiques de l'OCDE / Perspectives Économiques*.

1. Une fois dans *Perspectives Économiques*, appuyez sur *Bases de données statistiques* puis, à la page suivante, sur *tableaux*, puis sur *données* et, enfin, sur *accès aux données*.
2. Vous devrez ensuite sélectionner dans l'ordre vos *Pays*, vos *Variables* et votre *Période* de couverture.
3. La dernière étape consiste à sauvegarder vos données dans le format désiré.

12.2.3 Sherlock

1. Une fois votre enquête sélectionnée, cliquez dessus.
2. Choisissez le format d'extraction. À moins que vous connaissiez SAS ou SPSS, il vous est suggéré de choisir *Extraction par variables* et un fichier *.tab*.
3. Vous devrez ensuite sélectionner les variables désirées en cochant les cases correspondantes.

4. Finalement, vous devez choisir les valeurs des variables que vous désirées.
5. Pour extraire les données, entrez votre adresse courriel et appuyez sur *Extraire*. Les données vous seront envoyées en différé à l'adresse spécifiée.

Annexe A

Tableaux Récapitulatifs

A.1 Fonctions de Matlab

Fonction	Description	Forme
Importation de Données		
dlmread	Importe des données **numériques** d'un fichier.	M = (‘$nom\_de\_fichier$’, ‘$séparateur\_de\_données$’)
load	Importe des données **numériques** d'un fichier.	load $nom\_de\_fichier$
textread	Importe les données d'un fichier.	$[A, B, C \dots]$ = textread(‘$nom\_de\_fichier$’, ‘$format$’)
xlsread	Importe les données d'un tableur Excel.	$[A, B]$ = xlsread($nom\_de\_fichier$)

Transformation de Variables

cumsum	Somme cumulative des éléments d'un vecteur.	`cumsum(`z`)`
exp	Exponentielle d'une matrice élément par élément.	`exp(`z`)`
length	Nombre de colonnes d'un vecteur.	`length(`z`)`
log	Logarithme naturel d'une matrice élément par élément.	`log(`z`)`
max	Renvoie l'éléments du vecteur ayant la valeur la plus élevée.	`max(`z`)`
mean	Moyenne des éléments d'un vecteur.	`mean(`z`)`
min	Renvoie l'éléments du vecteur ayant la valeur la moins élevée.	`min(`z`)`
size	renvoie un vecteur 1×2 contenant la taille d'une matrice (rangées,colonnes).	`size(`z`)`
std	Écart-type des éléments d'un vecteur.	`std(`z`)`
sqrt	Racine carrée d'une matrice élément par élément.	`sqrt(`z`)`
sum	Somme des éléments du vecteur.	`sum(`z`)`
var	Variance des éléments d'un vecteur.	`var(`z`)`

Fonctions Matricielles

diag	Extraction de la diagonale d'une matrice. / Création d'une matrice diagonale.	`diag(`A`)`
eye	Crée une matrice identité.	I `= eye(`$\#$`)`
inv	Inverse d'une matrice.	`Inv(`A`)`
ones	Créer une matrice dont chaque élément égale à un.	O `= ones(`$\#, \#$`)`
rand	Créer une matrice dont chaque élément est un aléa U(0,1)	U `= rand(`$\#, \#$`)`
randn	Créer une matrice dont chaque élément est un aléa N(0,1)	N `= randn(`$\#, \#$`)`
zeros	Créer une matrice dont chaque élément égale à zéro.	Z `= zeros(`$\#, \#$`)`
kron	Produit kronecker de A et B.	K `= kron(`A`,`B`)`

Autres Fonctions

diary	Permet de sauvegarder une copie de sa session dans le fichier spécifié.	`diary('`$nom\_de\_fichier$`')` *programme* `diary('off')`
fminsearch	Minimise une fonction.	$[x, fval]$ = `fminsearch(`$fonction, options, variables$`)`
hold	Permet de superposer plusieurs graphiques.	`hold on` *graphiques* `hold off`
normcdf	Donne la valeur de la fonction de répartition de la loi normale.	`normcdf(`X`)`
normpsf	Donne la valeur de la fonction de probabilité de la loi normale.	`normpdf(`X`)`
plot	Trace un graphique en nuage de points.	`plot(`$x, y,$`'`$couleur\_style\_marqueur$`')`
wk1write	Sauvegarde une matrice dans un fichier de tableur.	`wk1write('`$nom\_de\_fichier$`',`M`)`

A.2 Opérateurs

Description	Forme
Opérateurs Mathématiques	
Addition	`+`
Soustraction	`-`
Multiplication	`*`
Division	`/`
Puissance	`^`
Opérateurs Logiques	
ET	`&`
OU	`|`
Non ($\neg$)	`~`
Opérateurs de Comparaison	
Égal	`==`
Différent	`~=`
Plus grand	`>`
Plus petit	`<`
Plus grand ou égal	`>=`
Plus petit ou égal	`<=`

A.3 Symboles Mathématiques

Symbole	Description	Symbole	Description	Symbole	Description
$\sum$	somme	$!$	factoriel	$\perp$	perpendiculaire
$\prod$	produit	$\mid x \mid$	valeur absolue	$\parallel$	parallèle
$\int$	intégrale	$\parallel x \parallel$	norme de x	$\ll$	bcp plus petit
∂	dérivée partielle	$\prec$	précède, préféré	$\gg$	bcp plus grand
$\neq$	différent	$\emptyset$	ensemble vide	$\forall$	pour tout
$\equiv$	équivalent	$\subset$	sous-ensemble de (inclus dans)	$\exists$	il existe
$\approx$	approximativement égal à	$\in$	élément de	$\neg$	non (négation)
$\cong$	congruent, isomorphique	$\bigcap$	intersection	$\Rightarrow$	implique
$\propto$	proportionel	$\bigcup$	union	$\Leftrightarrow$	équivalent
$\sim$	similaires (géométrie), asymptotiquement, suit (une loi)				

A.4 Alphabet Grec

minuscule	majuscule	nom	minuscule	majuscule	nom
α		alpha	ν		nu
β		beta	ξ	Ξ	xi
γ	Γ	gamma	o		o
δ	Δ	delta	π, ϖ	Π	pi
ϵ, ε		epsilon	ρ, ϱ		rho
ζ		zeta	σ, ς	Σ	sigma
η		eta	τ		tau
θ, ϑ	Θ	theta	υ	Υ	upsilon
ι		iota	ϕ, φ	Φ	phi
κ		kappa	χ		chi
λ	Λ	lambda	ψ	Ψ	psi
μ		mu	ω	Ω	omega