Économétrie appliquée avec Stata

Nicolas Couderc[1]

« Dans un temps peut-être pas très lointain, on comprendra que pour former le citoyen efficace, il est aussi nécessaire de calculer, de penser en termes de moyenne de maxima et de minima qu'il est maintenant nécessaire de savoir lire et écrire »

H. G. Wells, *Mankind in the Making*, 1903, Chap. 6

Introduction

Stata est un logiciel complet permettant l'analyse statistique et économétrique, développé par *Stata Corporation*. Ce logiciel en est aujourd'hui à la version 9.1 (une version majeure sort tous les deux ou trois ans), et existe pour à peu près tous les systèmes d'exploitation.

Ce petit guide a pour but de vous faire découvrir Stata et de vous donner les bases pour une utilisation efficace. Il n'aborde bien évidemment qu'une infime fraction des capacités de Stata, et présente simplement les commandes basiques. Pour approfondir, vous pouvez vous reporter à l'aide incluse dans le logiciel ou aux manuels.

Stata n'est pas le seul logiciel d'économétrie existant. Il en existe en effet plusieurs dizaines. Ils possèdent des capacités différentes, sont plus ou moins conviviaux, flexibles, et leur prix peut aller de 0 à plusieurs milliers d'euros... Il est fréquent de devoir passer d'un logiciel à un autre, en fonction de leurs points forts respectifs (les rendements croissants s'appliquent ici, il est en général assez simple de passer de l'un à l'autre). Parmi les logiciels les plus connus :

- E-views. Très convivial, assez performant pour les séries temporelles, mais rapidement limité et peu flexible en ce qui concerne la programmation et certains modèles complexes.
- Excel. On peut faire de l'économétrie avec Excel. Toutefois, pour aller au delà des MCO, cela impose de la programmation VB assez lourde. La taille de la base de données est limitée par le nombre de cases de la feuille Excel.
- SAS. Extrêmement flexible et puissant, mais peu intuitif. Aucune limite à la quantité de données qu'il peut traiter.
- Gauss. Logiciel très peu convivial (il faut presque tout programmer soi-même), mais très rapide et puissant. Presque indispensable pour traiter de très grosses bases de données.
- RATS : Spécialisé séries temporelles.

[1] TEAM-CNRS. Université Paris 1 Panthéon-Sorbonne, 106-112 boulevard de l'Hôpital, 75647 Paris Cédex 13, France. Émail : couderc@univ-paris1.fr . Pour tout commentaire, n'hésitez pas à me contacter.

– R : Logiciel *open source* gratuit. Fonctions encore en nombre limité (mais logiciel en évolution rapide).
– Stata : Intuitif, assez flexible et complet. Possibilité de programmer. Stata a toutefois des problèmes pour gérer de très grosses bases de données. C'est le logiciel que nous présentons ici.

Avant de débuter la présentation de Stata, quelques précisions syntaxiques :

– Dans ce document, la fonte `commande` identifie des éléments qui doivent être entrés dans Stata. Ces éléments peuvent être de 3 ordres : **`commande`** *`variable`*`, options`. Le premier argument (gras) est à entrer tel quel dans Stata, le second (italique) est choisi par l'utilisateur (nom de fichier, de variable, etc) et le troisième est, comme son nom l'indique, optionnel.
– En ce qui concerne les noms de variables, Stata fait la différence entre majuscules et minuscules. Par ailleurs, Stata ne reconnaît pas les caractères accentués, pas plus que les espaces et caractères spéciaux.
– Les renvois d'une section à l'autre sont indiqués par le symbole : ⇒.
– La syntaxe Stata est économe : toute commande peut être autant abrégée que possible, *i.e.* que cela ne crée aucune ambiguïté. Ainsi, **`summarize`** peut s'abréger indifféremment en **`summari`**, **`sum`** ou **`su`**. On ne peut pas l'abréger en **`s`**, car il y aurait alors confusion possible avec une autre commande, **`sort`**.

1 Prise en main de Stata

1.1 Principes de base de Stata

1.1.1 L'installation

L'installation de Stata pour Windows s'effectue comme celle de n'importe quel autre logiciel. Il est par contre important, une fois le logiciel installé, de le mettre à jour. Pour savoir s'il existe des mises à jour, **`update query`**, ou à partir de Stata 9, utiliser la fonction « Vérification automatique de mises à jour ». Ces mises à jour sont de deux types : mise à jour de l'exécutable (**`update exe`**, suivi de **`update swap`**) et/ou des fichiers programmes (**`update ado`**). Pour tout mettre à jour, **`update all`**. Il faut bien entendu régulièrement vérifier la disponibilité de mises à jour.

Pour vérifier que l'installation de Stata est correcte, **`verinst`**. En cas de problème, supprimez Stata et réinstallez le intégralement.

Il est possible, une fois Stata installé et mis à jour, d'ajouter des programmes supplémentaires (appelés « modules externes ») créés par des utilisateurs. Par exemple, il n'est pas possible d'estimer un modèle de panel dynamique à la Arellano-Bover (1995) avec Stata « officiel ». Heureusement, il existe un module externe permettant de le faire. Celui-ci s'appelle **`xtabond2`**. Avant de pouvoir l'utiliser, il convient de l'installer.

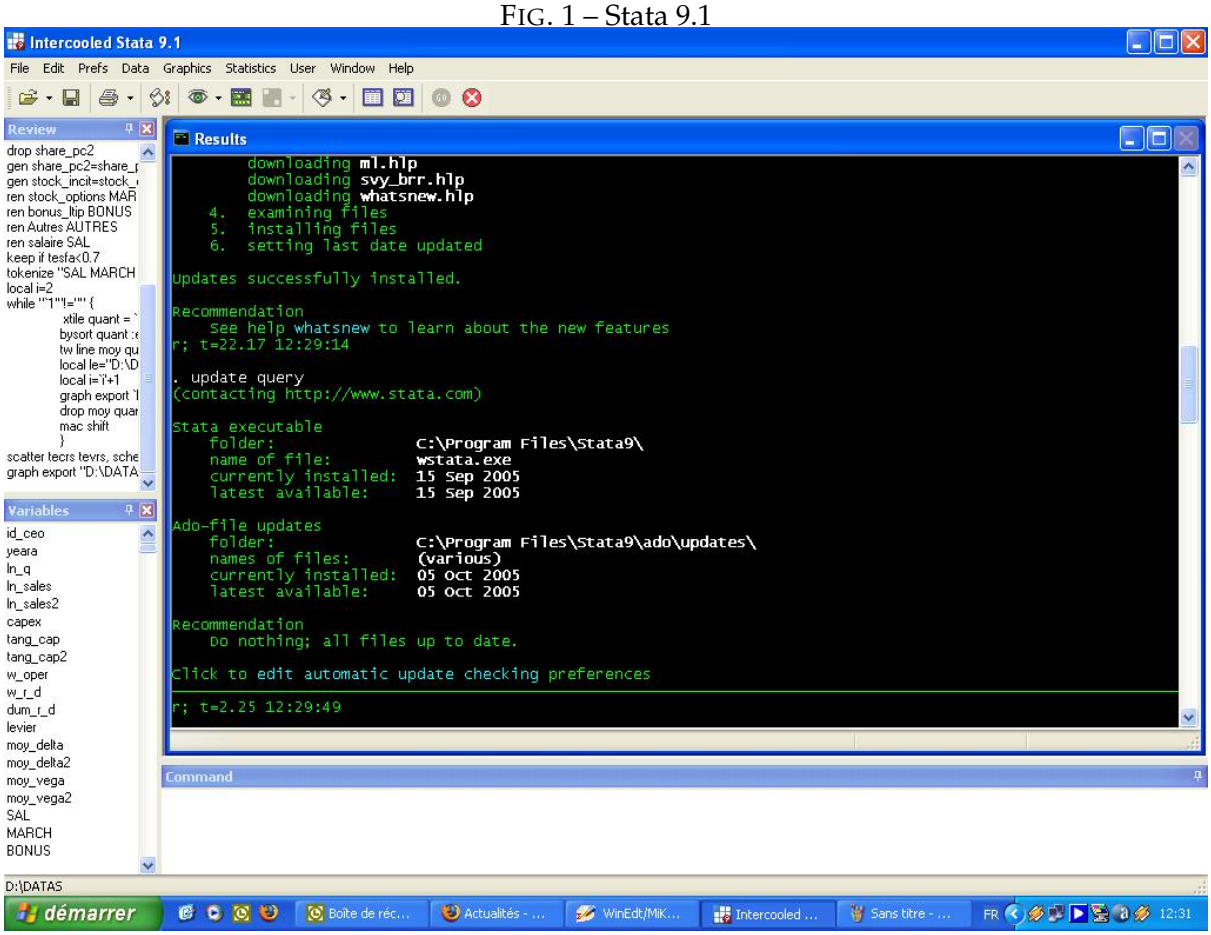
La plupart de ces modules externes sont disponibles sur le serveur du *Boston College*. Si c'est le cas, pour installer le module souhaité, sous Stata, tapez **`ssc install`** *`nom_du_module`*`, all` (ici, le *`nom_du_module`* est *`xtabond2`*). Il est également possible d'installer un module externe manuellement (⇒ 1.1.3). On trouve également de nombreux modules sur le site du Stata Journal (⇒ 1.2).

Il existe plusieurs centaines de modules externes disponibles sur Internet. D'une manière générale, pour savoir s'il existe un module proposant une fonctionnalité donnée, on peut le chercher et l'installer sans quitter Stata :
`net search` *`mot-clé`*, puis **`net install`** *`nom_module`*`, all`

1.1.2 Les fenêtres

Stata se présente sous la forme de 4 fenêtres. De haut en bas, et de droite à gauche (fig. 1), on trouve successivement la fenêtre Review, qui affiche l'historique des commandes tapées par l'utilisateur et permet d'en rappeler une facilement. La fenêtre Results est celle qu'utilise Stata pour afficher tous les résultats des commandes tapées par l'utilisateur. La fenêtre Variables détaille toutes les variables présentes dans la base de données actuellement ouverte dans Stata (Stata ne peut ouvrir qu'une seule base de données en même temps). Enfin, la fenêtre Command permet à l'utilisateur d'entrer les commandes.

FIG. 1 – Stata 9.1



On trouve au dessus de ces fenêtres une barre de menus, permettant d'exécuter les commandes les plus courantes sous Stata sans avoir à se servir de la fenêtre Command. On trouve par exemple deux icônes permettant d'afficher la base de données (Data Editor ou Data Browser), commandes également accessibles par le biais du menu (Data/Data Editor ou Data/Data Browser) et directement à partir de la fenêtre de commandes (**`edit`** ou **`browse`**). De même, l'ouverture du Do-File Editor peut se faire en appuyant sur l'icône correspondante dans la barre de menu, en sélectionnant dans le menu Data/Do-file Editor/New file, ou en entrant **`doedit`** dans la fenêtre Command. L'utilisation de ces barres de menu est donc facultative, et finalement assez rare lorsqu'on utilise Stata.

1.1.3 Fichiers et répertoires

Stata utilise plusieurs types de fichiers, nommés d'après leurs extensions :

– Fichier .do : Fichiers de commandes, au format ASCII, lisibles dans n'importe quel éditeur de texte et bien entendu par le Do-file Editor (**`doedit`**) inclus dans Stata. Ils permettent à l'utilisateur de lancer plusieurs commandes Stata en une seule opération et de garder une trace des commandes exécutées (⇒ 6.2). Pour lancer un do-file, il suffit de

cliquer sur Run ou Do dans le Do-File Editor. La seule différence entre Run et Do est que la seconde commande affiche les résultats dans la fenêtre Results, alors que la première est silencieuse. Il est possible de lancer l'intégralité du do-file ou d'en exécuter seulement une partie. Pour cela, il suffit de sélectionner la partie souhaitée du .do avant de cliquer sur Do ou Run.

- Fichier .ado : Fichier de programmes, définissant une ou plusieurs commandes. Ces fichiers sont au format ASCII (lisible dans n'importe quel éditeur de texte) ; l'utilisateur peut en créer de nouveaux ou en installer à partir d'Internet (⇒ 1.1.1).
- Fichier smcl : Fichiers d'aide (.hlp, ⇒ 1.2) ou de log (extension libre, en général .log. ⇒ 1.3.2). Ces fichiers s'affichent dans le Viewer Stata ou dans n'importe quel éditeur de texte, ils sont au format ASCII. La syntaxe smcl permet à Stata d'afficher le gras, les italiques, etc...
- Fichier .dta : fichiers de données au format Stata. L'utilisation de ce format de données permet d'accélérer l'ouverture de bases de données de grande taille (⇒ 2).

L'installation de Stata crée plusieurs répertoires :

- un répertoire qui contient le programme (Stata.exe) et les fichiers indispensables au fonctionnement du logiciel. Ce répertoire contient deux sous-répertoires. Le premier « ado », contient les versions originales des fichiers .ado officiels installés par Stata et le second « updates » contient les versions mises à jour par Stata des fichiers .ado officiels.
- un répertoire (par défaut c :\ado) qui contient le sous-répertoire « personal », dans lequel doivent se trouver tous les fichiers .ado créés par l'utilisateur ou installés manuellement, et un répertoire « plus », qui contient les .ado créés par d'autres utilisateurs et installés directement par Stata.
- un répertoire de travail (par défaut c :\data), appelé à contenir bases de données, do-files, *etc*...

1.2 Où trouver de l'aide ?

De nombreuses ressources existent pour aider l'utilisateur de Stata faisant face à un problème. Parmi celles-ci, les plus utiles sont probablement :

- L'aide incluse dans le logiciel **`whelp`**, ou **`whelp`** *`command`*.
- Les manuels Stata (Stata Press). Ils reprennent et complètent l'aide en ligne, détaillent des exemples, *etc*... On peut les acheter indépendamment du logiciel.
- Le *Stata Technical Bulletin* et le *Stata Journal*, disponibles dans quelques BU françaises, présentent des méthodes et programmes nouveaux.
- Plusieurs sites Web proposent des ressources, des aides ou des programmes pour Stata. Les deux principaux : http ://www.stata.com et http ://www.stata-press.com. Le site web de U.C.L.A. (http ://www.ats.ucla.edu/stat/stata/) propose un annuaire des ressources disponibles sur Internet pour Stata. http ://www.indiana.edu/ jslsoc/, le site de J. Scott Long et Jeremy Freese, propose ainsi des programmes très utiles pour qui veut faire de l'économétrie des variables qualitatives. Il en existe des dizaines.
- La *Statalist*. Liste de diffusion utilisée pour poser des questions techniques à la communauté Stata. De nombreux « Stata gurus » y sont abonnés et répondent fréquemment à des questions posées par les autres membres. Pour consulter les archives, *cf.* http ://www.stata.com/statalist/archive/. C'est une source extrêmement précieuse de renseignements, plusieurs milliers de messages sont archivés, décrivant les problèmes rencontrés par certains utilisateurs et les solutions proposées par un membre de la liste. Pour s'y inscrire (et pouvoir envoyer des mails à la liste !), il suffit d'envoyer un mail à majordomo@hsphsun2.harvard.edu, avec pour objet « subscribe statalist ». Attention : vérifiez avant de poster un message que votre question est claire, qu'elle appelle une réponse, et surtout que celle-ci ne fait pas l'objet d'une réponse dans l'aide Stata, les manuels et les archives de la liste... Vous risqueriez de recevoir une réponse expéditive.

1.3 Commandes diverses (et utiles !)

1.3.1 Gestion de la mémoire

Lorsque Stata doit ouvrir une base de données, celle-ci est intégralement chargée en mémoire. Pour un état des lieux de la mémoire, **`memory`**. Stata dispose par défaut de 10 mégas de mémoire. Résultat : toute base de plus de 10 mégas ne s'ouvrira pas...sauf si on modifie la quantité de mémoire disponible pour Stata. Pour vérifier la taille de la base à charger, on utilise la commande **`describe using`** *`mabase.dta`*. Si celle-ci nécessite 100 mégas, **`set memory`** *`100m`*. Lorsque la base est plus grosse que la taille de la mémoire vive installée, il faut recourir à la mémoire virtuelle, **`set virtual on`**. Cela ralentit considérablement Stata. Pour économiser de l'espace mémoire, on peut compresser la base **`compress`**.

Il existe trois versions de Stata (*Small*, *Intercooled* et *SE*). La seule différence entre ces trois versions tient à la quantité de mémoire qui peut être allouée à différentes opérations. **`set maxvar`** *`nombre`* permet d'augmenter le nombre de variables (max : 32 700 pour Stata SE). Lorsque la limite de mémoire touche la taille des matrices, on peut l'augmenter grâce à la commande : **`set matsize`** *`nombre`*.

1.3.2 Log

Pour conserver une trace complète de tout ce qui s'affiche dans la fenêtre Results au cours d'une session (commandes/résultats...), il est possible de créer un fichier *log*. Stata permet de créer des *log* au format texte (option `text`), ou au format *smcl* (option `smcl`, ⇒ 1.1.3). Ce dernier format permet un affichage du *log* dans le Viewer Stata très propre, mais complique la lecture du *log* avec d'autres logiciels. Pour commencer un nouveau fichier de *log* : **`log using`** *`monfichier`*. Pour le fermer *log* : **`log close`**. Pour mettre le log à la suite du log issu de la précédente session Stata : `append`, pour débuter à partir d'un fichier vierge : `replace`. Exemple :

Début de session : **`log using`** *`monfichier`*`, text replace`

Fin de session : **`log close`**

1.3.3 Gestion de l'affichage

Pour supprimer l'affichage des résultats d'une commande, **`quietly`** *`commande`*.

Pour détailler au maximum les opérations réalisées par Stata (à utiliser principalement pour debugger un .ado) : `set trace on`.

Pour que Stata ne stoppe pas au cours de l'exécution d'un do-file lorsque l'affichage des résultats atteint le bas de la fenêtre Results, **`set more off`**.

1.4 Fonctions et expressions

1.4.1 Opérateurs et fonctions mathématiques

Le tableau 1 détaille les opérateurs mathématiques et logiques reconnus par Stata. Quelques remarques à leur propos :

- En ce qui concerne le signe =, il existe une particularité. Lorsque le signe = est une conjecture (à tester) ou une condition et non une définition, on doit le remplacer par ==.
- Les valeurs manquantes (symbolisées par un point « . » sous Stata) sont les plus grandes valeurs. Ainsi, l'expression *`salaire>1500`* est vraie si le salaire est supérieur strictement à 1500, ou manquant. Pour ne conserver que les valeurs supérieures à 1500 et non-manquantes, il faut préciser : *`salaire>1500 & salaire<.`*
- Les opérateurs suivent l'ordre de priorité habituel. L'opérateur « Et » est prioritaire sur l'opérateur « Ou ».

TAB. 1 – Opérateurs et fonctions sous Stata

Addition	+	Soustraction	-
Multiplication	*	Division	/
Égalité	=	Inégalité	~= ou !=
Exposant	^	Partie entière	**int()**
Racine	**sqrt()**	Exponentielle	**exp()**
Logarithme	**log()**	Valeur absolue	**abs()**
Sup. (resp. Inf.)	> (resp. <)	Sup. (resp. Inf) ou égal	>= (resp. <=)
Ou	|	Et	&
Minimum	**min()**	Maximum	**max()**

1.4.2 Trois expressions : `by`, `if` et `in`

Ces trois expressions peuvent s'utiliser avec la majorité des commandes Stata. On peut les combiner les unes avec les autres.

by permet d'appliquer une commande à chaque valeur d'une variable. La syntaxe de cette expression est **by** *variable* **: commande** *variable*. Il faut que la base de données soit classée par cette variable. On peut utiliser **sort** avant le **by**, ou pour classer et effectuer le **by** en même temps, **bysort**.

Exemple : **bysort** *sexe* **: summarize** salaire permet d'obtenir le salaire moyen des femmes et des hommes.

if permet de n'appliquer la commande qu'aux observations remplissant une condition particulière. Syntaxe : **commande** *variable* **if** *condition*.

Exemple : **summarize** *salaire* **if** *age==18 | age>20 & age<=50* donne les statistiques descriptives de la variable salaire pour les observations dont la variable age est égale à 18 (remarquer le double signe =, il s'agit d'un test) ou comprise entre 20 (exclu) et 50 (inclus). L'opérateur & est prioritaire, les parenthèses sont inutiles ici.

in permet de n'appliquer la commande qu'aux observations se situant dans un intervalle donné. Syntaxe : **commande** *variable* **in** *condition*.

Exemple : **summarize** *salaire* **in** *10/20* donne les statistiques descriptives de la variable *salaire* pour les observations de la 10ème à la 20ème ligne de la base de données.

2 Stata et les données

2.1 Importation et ouverture de bases de données

Stata ne peut gérer qu'une seule base de données à la fois. Avant d'en ouvrir une, il convient donc de fermer celle qui est actuellement utilisée (**clear**). Pour entrer des données dans Stata, il existe plusieurs moyens, à utiliser en fonction de la nature des données :

- Si il n'y a que quelques données à rentrer, on peut utiliser la commande **input** :

```
input str20 nom age str6 sexe
"Jean Némard" 22 homme
"Paule Pot" 37 femme
"Guy Tare" 48 homme
end
```

  On peut également saisir directement les données dans le Data Editor (**edit**), comme on ferait avec Excel.
- Si l'utilisateur dispose déjà d'une base de données au format Stata (.dta), nommée par exemple base.dta, **use** *base.dta*

– Si l'utilisateur dispose de données « propres » (par exemple en provenance d'un tableur), **insheet** *base.txt*. Des données propres sont des données organisées avec une ligne par observation, les variables en colonne, les chaînes de caractères identifiées par des guillemets et les séparateurs tous identiques (virgule ou tabulation. Pas d'espace !).
– Si les données sont plus « folkloriques » (*i.e.* au moins une des conditions précédente n'est pas remplie), le recours à **infile** s'impose. Deux cas se présentent. Les données peuvent être à format indéfini (séparation entre les données sous forme d'espace, de tabulation ou de virgule, chaînes de caractères entre guillemets, une observation sur plusieurs lignes ou plusieurs observations par ligne) ou fixé (pas de guillemets autour des chaînes de caractères, etc). Si les données sont à format indéfini, **infile** *var_a var_b* **using** *base.txt*. Si elles sont à format fixé, le recours à un dictionnaire s'impose, pour définir le format des données. Dans ce cas, voir l'aide de Stata **whelp infile**.

Pour gagner du temps, si les données proviennent d'un tableur ou sont formatées pour un autre logiciel d'économétrie, l'utilisation du logiciel StatTransfer peut faire gagner un temps non négligeable (et éviter des erreurs)...

2.2 Exportation et sauvegarde des données

Plusieurs formats de sauvegarde des données sont possible. On peut tout simplement sauver des données dans le format de Stata (**save** *mabase.dta*), pour utilisation future. On peut également exporter des données pour les utiliser ensuite avec un tableur ou un autre logiciel, grâce aux commandes **outsheet** *var* **using** *mabase.txt* et **outfile** *var* **using** *mabase.txt*. Les bases exportées avec **outsheet** peuvent être réimportées avec **insheet**, et **outfile** avec **infile**.

2.3 Gestion des variables

2.3.1 Créer des variables

Pour créer de nouvelles variables, deux commandes existent. **generate** permet de créer des variables qui nécessitent des calculs « simples » et **egenerate** (*extended generate*) s'impose lorsque les calculs se complexifient un peu ou que l'utilisation de fonctions statistiques spécifiques est nécessaire.

Exemples d'utilisation de **generate** :

1. **gen** *x = 12*. Crée une constante nommée x, qui vaut 12.
2. **gen** *x = log(a*b)-sqrt(abs(b))*. La nouvelle variable x est égale au logarithme de $(a \times b)$ moins la racine carrée de la valeur absolue de b.
3. **gen** *x = "Bonjour"*. x est une chaîne de caractères, égale à « Bonjour » pour toutes les observations.
4. **gen** *x = (sexe=="Homme")*. x est égale à 1 si la variable $sexe$ contient la chaîne de caractère « Homme », zéro sinon.
5. **gen** *x = salaire[4]*. x est une constante, égale au contenu de la $4^{\text{ème}}$ observation pour la variable $salaire$.
6. **gen** *x = y[_n-1]*. x est égale à la valeur de l'observation précédente de y.

Exemples d'utilisation de **egenerate** :

1. **egen** *x = sd[y]*. x, constante, est égale à l'écart-type de y.
2. **egen** *x = pctile(y), p(50)*. x est égale au percentile 50 de y (la médiane).
3. **egen** *x = group(sexe)*. x est une variable qui contient un numéro différent pour chaque modalité de sexe (on peut supposer que x sera égal à 0 pour femme et 1 pour homme).

La commande **xi** *i.variable* permet de créer des variables indicatrices[2].
Exemples :

1. **xi** *i.individu*. Crée une variable indicatrice pour chaque individu présent dans la base, moins un.
2. **xi** *i.individu*pcs*. Crée les variables indicatrices correspondant aux individus et les termes d'interaction entre la profession et l'individu.

2.3.2 Manipuler des variables

1. **list** *pi**. Affiche toutes les variables dont le nom commence par *pi*.
2. **rename** *ancien_nom nouv_nom*. Renomme la variable *ancien_nom* en *nouv_nom*.
3. **replace** *x = 12*. Remplace le contenu de la variable x par 12 (x devient une constante).
4. **replace** *x = y* **if** *y>10 & y!=.*. Remplace la valeur de x par celle de y, si y est supérieur à 10 et contient une valeur.
5. **drop** *x*. Supprime la variable x.
6. **drop** *_all*. Supprime toutes les variables (conserve en mémoire les matrices, constantes, etc...). Pour nettoyer complètement la mémoire, il faut utiliser **clear**.
7. **keep** *x*. Conserve la variable x, supprime toutes les autres.
8. **label var** *variable "label"*. Attribue à une variable x un label (une étiquette). Exemple : **label var** *lnsal "logarithme du salaire brut"*.
9. **describe** *x y* affiche les types et les labels des variables x et y.

2.3.3 Types de variables

Les variables sous Stata peuvent être numériques ou alphanumériques. Les variables numériques peuvent être de différents types (voir tableau 2), selon la précision (et la place en mémoire) nécessaire.

Les variables alphanumériques sont des chaînes de caractères quelconques (string, str), d'une longueur maximale de 244 caractères. Pour transformer une chaîne en variable numérique, **destring** *variable*, options. Parmi les options, gen(*var*) ou replace. La première option crée une nouvelle variable, nommée *var* contenant la transformation demandée, la seconde écrase au contraire la variable chaîne pour la remplacer par sa transformation. Stata stocke par défaut une variable sous forme de chaîne lorsque pour au moins une observation, la variable contient au moins un caractère non-numérique. Lorsqu'une variable est au format alphanumérique, il est impossible de l'utiliser dans une régression. Si le code pour les valeurs manquantes dans une base de données est "na", toutes les variables dont une valeur au moins manque seront ainsi automatiquement sauvées sous forme de chaîne. Pour contraindre Stata à ignorer les caractères non numériques (et à les remplacer par des valeurs manquantes), l'option force est à ajouter à la commande **destring**.

Pour faire l'opération inverse (d'une variable numérique à une chaine de caractères, la commande est **tostring** (mêmes syntaxe et options).

Pour changer le format d'affichage d'une variable, **format** *variable format*. Pour n'afficher que les 20 premiers caractères d'une chaîne de caractères, **format** *chaine %20s*.

2.4 Gestion des données

2.4.1 Fusion de bases

Stata ne peut ouvrir qu'une seule base de données en même temps. Pour nettoyer la mémoire de Stata, **clear**. C'est une opération indispensable avant de charger une autre base

[2] Cette commande peut aussi s'utiliser directement lors d'une régression, ⇒ 4.1.

TAB. 2 – Datatypes numériques

Nom	Min/Max	Valeur la plus proche de 0	Taille mémoire (bytes)
byte	$-127/100$	$+/-1$	1
int	$-32\,767/32\,740$	$+/-1$	2
long	$-2\,147\,483\,647/2\,147\,483\,620$	$+/-1$	4
float	$-1,70... \times 10^{38}/1,70... \times 10^{36}$	$+/-10^{-36}$	4
double	$-/+8,98... \times 10^{307}$	$+/-10^{-323}$	8

de données. Pour utiliser plusieurs bases, deux solutions. Première solution : on ouvre la première, on l'utilise, on la ferme, puis on ouvre la seconde, *etc*. Simple, mais lorsqu'on a besoin en même temps de variables ou d'observations présentes dans différentes bases il faut fusionner ces bases pour les unifier. Pour cela, trois possibilités :

1. Il s'agit d'ajouter de nouvelles observations. On doit ouvrir la première base sous Stata, puis **`append using`** *`nom_de_la_seconde_base`*. Il faut sauver la nouvelle base, contenant les observations des deux bases initiales. Attention : si dans la première base, le salaire est nommé *sal* et dans la seconde *salaire*, la base finale comprendra deux variables, *sal* et *salaire*...
2. Il s'agit d'ajouter de nouvelles variables à des observations déjà existantes. Deux possibilités : les observations sont les mêmes dans les deux bases (*one to one merge*). Par exemple, les deux bases concernent les 100 mêmes individus. Il faut alors classer les deux bases dans le même ordre, puis les fusionner. Il faut donc ouvrir la première base, la classer, la sauvegarder, la fermer. Ouvrir la seconde, la classer. Puis on fusionne les deux, **`merge using`** *`nom_de_la_première_base`*. Second cas, les observations sont pour certaines communes aux deux bases, pour certaines différentes (*match merge*). Il faut tout de même que les observations soient classées de la même manière dans les deux bases. Par exemple, si les observations dans les deux bases concernent des entreprises identifiées par leur code SIRET, il faut classer les observations des deux bases par SIRET. Il faut donc ouvrir la première base, la classer, la sauvegarder, la fermer. Ouvrir la seconde, la classer. Puis on fusionne les deux, en utilisant la variable *`siret`* pour faire se correspondre les données : **`merge`** *`siret`* **`using`** *`nom_de_la_première_base`*. La fusion implique que le nom de la variable servant à fusionner soit exactement le même dans les deux bases. Dans la nouvelle base, une variable *_merge* a été créée ; elle donne le résultat du merging. Si *_merge*$= 1$, les données de cette observation proviennent exclusivement de la base maître (*i.e.* la seconde). Si *_merge*$= 2$, les données de cette observation proviennent exclusivement de la base using (ie la première). Enfin, si *_merge*$= 3$, les données proviennent des deux bases.
3. Il s'agit de compléter ou de mettre à jour des données (remplacer des valeurs anciennes ou manquantes par de nouvelles données). Dans ce cas, la première base est celle contenant les nouvelles données. Après avoir classé les deux bases, et ouvert la base contenant les données à mettre à jour,
   `merge using` *`nom_de_la_première_base`*`, update`.

Attention : la fusion de bases est une opération qui peut rapidement créer des erreurs dans la base si l'utilisateur ne prend pas toutes les précautions. Ainsi, un **`tab`** *`_merge`* et une vérification « visuelle » poussée des résultats de la procédure de fusion sont plus que conseillés.

2.4.2 Opérations sur des observations

Pour réorganiser la base de données, lorsqu'elle est « à l'envers » (*wide format* dans le langage Stata), l'utilisation de **`reshape`** s'impose. Qu'est ce qu'une base de données « à l'envers » ? C'est une base qui se présente comme celle du tableau 3. Pour l'utiliser à des fins économétriques, il faut la transformer pour qu'elle ressemble à celle du tableau 4, grâce à la

commande **reshape long** *salaire*, i(*id*) j(*annee*). Pour faire l'inverse (passer d'un format *long* à un format *wide*), **reshape wide** *salaire*, i(*id*) j(*annee*).

TAB. 3 – Base de données *wide*

id	sexe	salaire1995	salaire1996	salaire1997
1	0	1500	1500	2000
2	1	1000	1200	1400
3	0	3000	3000	3000
		etc...		

TAB. 4 – Base de données *long*

id	annee	sexe	salaire
1	1995	0	1500
1	1996	0	1500
1	1997	0	2000
2	1995	1	1000
2	1996	1	1200
2	1997	1	1400
3	1995	0	3000
3	1996	0	3000
3	1997	0	3000
	etc...		

Il est possible de sélectionner des observations dans une base de données pour les supprimer **drop in** ou les conserver (supprimer les autres) **keep in**, avec la même logique que les commandes **drop** et **keep** appliquées aux variables : **drop in** *1/10* supprime les 10 premières observations de la base, **keep in** *1/10* conserve les 10 premières et supprime les autres.

Pour classer les observations dans la base de données, les commandes **sort** et **gsort** sont à utiliser. **sort** *variable* classe la base par ordre croissant des valeurs de la variable. Pour des tris en ordre décroissant, **gsort** *-variable*. Par exemple, **gsort** *-nom +annee* classe la base dans l'ordre alphabétique inverse et classe les observations correspondant à un individu donné dans l'ordre chronologique croissant.

Il est possible d'attribuer des labels différents à chaque valeur d'une variable. Si la variable *sexe* est égale à 1 pour les hommes et 0 pour les femmes, il est possible de labéliser les valeurs : **label define** *deflab 1 "Hommes" 0 "Femmes"* puis **label values** *sexe deflab*. La variable sexe est toujours numérique, bien que soient associées à ces valeurs des chaînes de caractères.

3 Statistique descriptive

3.1 Statistiques descriptives

summarize *variable* permet d'obtenir les statistiques descriptives usuelles. L'option detail permet d'en avoir plus. La commande **tabstat** *variable* permet de faire presque la même chose que **summarize**, mais permet plus de flexibilité pour faire un tableau de statistiques.

tabulate *variable* calcule les fréquences des valeurs prises par une variable, et permet de créer des tableaux croisés pour deux variables.

collapse permet de créer une base de données contenant les statistiques descriptives d'une autre. Exemple : **collapse** *(mean) age educ (median)revenu*, by(*pays*) cal-

cule les âges et niveaux d'éducation nationaux moyens et les revenus nationaux médians et ne conserve que ces moyennes dans la base de données.

⇒ Voir également **histogram**, section 5.1 et **outtex**, section 6.1.

3.2 Corrélation

pwcorr *variable1 variable2* donne la matrice de corrélations entre les variables. L'option `sig` permet d'obtenir le résultat du test de nullité du coefficient de corrélation.

corr *variable1 variable2*, `cov` permet d'obtenir la matrice des variance-covariances.

pwcorr *variable1 variable2 variable3* permet d'obtenir les coefficients de corrélation partielle entre les variables prises deux à deux.

3.3 Tests sur la moyenne, la variance et la distribution des variables

ttest permet de comparer les moyennes de deux variables. Attention : ce test repose sur l'hypothèse implicite d'égalité des variances des deux variables. L'option `unequal` permet de relâcher celle-ci.

Exemples :

1. **ttest** *salaire*, **by(***sexe***)**, pour tester la significativité de la différence de salaire entre hommes et femmes.
2. **ttest** *salaire=1000* pour savoir si la moyenne des salaires est égale à 1 000 euros.

sdtest permet de comparer les variances de deux variables. La syntaxe et le fonctionnement de cette commande sont identiques à ceux de **ttest**. Exemple : **sdtest** *salaire*, **by** *sexe*.

tabulate *var1 var2*, **chi2** permet de procéder au test du χ^2 de Pearson (H_0 : indépendance des lignes et colonnes du tableau croisé).

4 L'économétrie avec Stata

4.1 Généralités sur l'économétrie avec Stata

Toutes les commandes d'estimation fonctionne de la même manière sous Stata. La syntaxe est identique : **by** *variable* **: commande** *y x1 x2...* **if/in** *condition*, `options`[3]

Lorsque plusieurs équations sont à estimer, on écrit les *variables* par équation : *(y1 x1 x2) (y2 x3 x4)*.

On peut très facilement ajouter aux variables de l'équation des variables indicatrices, grâce à la commande **xi**. Ainsi, **xi : regress** *y x i.pays* lance une régression qui comporte comme variables explicatives x et $n-1$ variables indicatrices (s'il y a n pays dans la base).

Lorsqu'on tape la commande sans rien ensuite (**commande**), les résultats du dernier modèle estimé avec cette commande sont affichés à nouveau.

4.2 Données transversales

regress permet de réaliser une régression MCO. Sa syntaxe, identique à celle de la majorité des commandes Stata est : **regress** *y x1 x2*, `options`. Il est inutile de se préoccuper de la constante, Stata l'ajoute automatiquement à toutes les régressions (pour forcer Stata à ne pas la rajouter, option `nocons`). Les options possibles sont `level(`*nombre*`)`, pour définir la taille de l'intervalle de confiance (par défaut : intervalle de confiance à 95 %, *nombre=95*). L'option `vce` permet d'afficher la matrice de variance-covariance des estimateurs.

[3] **by**, **if** et **in** sont bien évidemment optionnels.

4.3 Variables qualitatives

On peut utiliser le modèle **logit** ou **probit**. Voir les packages spost.ado pour l'économétrie des variables qualitatives, ils proposent de nombreuses commandes très utiles.

Parmi les modèles disponibles sous Stata, le logit multinomial (**mlogit**), le logit conditionnel (**clogit**), le logit ordonné (**ologit**), le probit bivarié (**biprobit**), *etc*...

Pour obtenir les effets marginaux, on peut, après l'estimation d'un modèle, utiliser la commande **mfx compute,** `options`, ou bien utiliser les commandes **dlogit2** et **dprobit** pour calculer directement les effets marginaux. **lstat** permet d'obtenir le taux de bonnes et mauvaises prédictions.

A partir de Stata 9, **estat gof** permet d'obtenir les test de qualité de la régression de Pearson et de Hosmer-Lemeshow.

4.4 Séries temporelles

4.4.1 Opérateurs de séries temporelles

Les commandes Stata concernant les séries temporelles débutent toutes par **ts**.

Lorsqu'on s'intéresse aux séries temporelles, pour créer des variables retardées ou avancées, il convient d'éviter de créer une variable grâce à [_n-1] ou [_n+1]. Au contraire, il faut définir la nature temporelle des variables : **tsset** *var_temporelle*. La commande **tsfill** permet de compléter une base de données en remplissant tous les « trous » par des valeurs manquantes. **tsreport** permet de vérifier la structure des séries temporelles présentes dans la base (trous, observations multiples pour un même identifiant temporel, etc).

Stata définit les opérateurs temporels courants, lag, lead, etc :

1. L. Opérateur retard (x_{t-1}).
2. L2. Deuxième retard (x_{t-2}).
3. F. Opérateur lead (x_{t+1}).
4. D. Opérateur différence $x_t - x_{t-1}$.
5. D2. Opérateur différence $x_t - x_{t-1} - (x_{t-1} - x_{t-2})$.
6. S. Opérateur différence saisonnière $x_t - x_{t-1}$.
7. S2. Opérateur différence saisonnière $x_t - x_{t-2}$.

Ces opérateurs peuvent être utilisés de manière synthétique : L(1/3).PIB signifie ainsi : « L.PIB L2.PIB L3.PIB ». Ils tiennent compte des valeurs manquantes. On peut les utiliser dans la majorité des commandes Stata, en particulier dans les commandes de régression, sans avoir besoin de créer par avance la variable concernée : **tabulate** *L.pib*, par exemple.

4.4.2 Modèles disponibles

Une fois précisé le caractère temporel des données, on peut simplement utiliser la commande **regress**. Les modèles spécifiques « séries temporelles » suivants sont également disponibles :

- Les modèles AR(p), MA(q) ou ARMA(p,q) : **arima** *y x1 x2*, ar(*nombre*) ma(*nombre*))[4],
- Les modèles ARIMA(p,d,q) : **arima** *y x1 x2*, arima(*p*,*d*,*q*),
- Les modèles ARCH et leurs dérivés (**arch**, **garch**, etc...),
- Les modèles VAR, S-VAR, VECM (**var**, **svar**, **vec**).

Pour obtenir les autocorrélogrammes et autocorrélogrammes partiels jusqu'au k$^{\text{ième}}$ retard **corrgram** *variable*, lags(*k*).

[4] Les nombres sont les termes de retards autorégressifs et de moyenne mobile à inclure dans le modèle : (*nombre*) peut être *(1 4)* (premier et quatrième termes) ou *(1/4)* du premier au quatrième terme.

4.5 Données de panel

4.5.1 Données de panel

Les commandes Stata concernant les données de panel débutent en général par **xt**. Il faut définir la nature de panel des données : **tsset** *var_individu var_temps*.

xtsum et **xttab** permettent d'obtenir des statistiques descriptives intra et inter-individuelles. Leurs syntaxes sont identiques à **sum** et **tab**.

4.5.2 Modèles à effets fixes

Pour estimer un modèle statique en données de panel, **xtreg** *y x1 x2*, `option`. Plusieurs modèles sont possibles : l'option `fe` estime le modèle à effets fixes. Ce modèle repose sur la différenciation des variables par rapport à la moyenne individuelle pour éliminer les effets fixes[5]. Quelques particularités des résultats : trois statistiques de R^2 sont affichées. Le R^2 *within* donne la part de la variabilité intra-individuelle de la variable de gauche expliquée par celles des variables de droite. C'est le plus important des trois. Le R^2 *between* estime l'apport des effets fixes au modèle. Le R^2 *overall* traduit la qualité globale de la régression. D'autre part, deux F-stat sont proposées. La première indique la significativité jointe des variables explicatives, la seconde la significativité jointe des effets fixes.

Le LM-test de Breush-Pagan (**xttest2**) teste la corrélation entre observations dans le modèle à effets fixes.

Le test d'autocorrélation sérielle au premier ordre des résidus peut s'effectuer grâce à la commande **pantest2** *id_temps*. On peut aussi utiliser **xtserial** *y x1 x2* qui n'impose pas de choix entre le modèle à effets fixes ou aléatoires. Enfin, la commande **xtregar** *y x1 x2*, `lbi` permet d'estimer modèles a effets fixes ou aléatoires avec erreurs autocorrélées d'ordre 1. L'option `lbi` permet d'effectuer le test de Baltagi-Wu et celui de Durbin-Watson.

4.5.3 Modèles à effets aléatoire

xtreg *y x1 x2*, `re` estime le modèle à effets aléatoires. On retrouve les trois R^2. Le R^2 *within* donne la contribution des effets aléatoires au modèle. Le R^2 *between* indique la part de la variabilité inter-individuelle expliquée par celles des variables de droite. Il se focaliser sur celui-ci. Le R^2 *overall* est le même que précédemment.

Le test de Breush-Pagan (**xttest0**) teste la significativité des effets aléatoires. La commande **xttest1** permet d'obtenir plusieurs les résultats de plusieurs tests classiques à effectuer après le modèle à effets aléatoires (test de Breush-Pagan, test de Baltagi-Li (1995) de corrélation sérielle d'ordre 1, le test joint de Baltagi-Li (1991) de corrélation sérielle et d'effets aléatoires, etc...).

Le test d'autocorrélation sérielle au premier ordre des résidus peut s'effectuer grâce à la commande **xtserial** *y x1 x2*.

Le test de Hausman est a utiliser pour déterminer s'il faut recourir à un modèle à effets fixes ou aléatoires. Pour l'implémenter, il faut estimer le modèle à effets fixes, stocker les résultats (**est store** *fixe*), estimer le modèle à effets aléatoires puis effectuer le test : **hausman** *fixe* `.`.

4.5.4 Autres modèles

L'estimation de modèles logit (**xtlogit**) et probit (**xtprobit**), avec effets fixes (option `fe`) ou aléatoires (option `re`) est possible avec Stata. Syntaxe et fonctionnement proches des commandes **logit** et **probit**.

[5] L'autre possibilité pour estimer un modèle à effets fixes est d'intégrer des variables indicatrices pour chaque individu, voir **areg**. On peut également avoir recours à l'estimateur de Hausman-Taylor, **xthtaylor** lorsqu'on doit intégrer dans les variables explicatives des variables invariantes au fil du temps. Pour estimer un modèle à effets fixes en utilisant des variables instrumentales, voir **xtivreg, fe**

Pour l'estimation d'un modèle de panel dynamique, utiliser **xtabond** et/ou le module externe **xtabond2**. On peut également estimer des modèles de frontière stochastique en panel (**xtfrontier**), *etc*...

4.6 Commandes "post-estimation"

Lorsque Stata estime un modèle, les résultats sont affichés à l'écran et disponibles pour l'utilisateur sous forme de scalaires, matrices ou fonctions (Stata qualifie ces éléments créés lors d'une régression de « e()-class »).

Il existe dans Stata des commandes qualifiées de « post-estimation », *i.e.* commandes à exécuter après l'estimation d'un modèle. Ces commandes, par défaut, s'appliquent aux résultats du dernier modèle estimé. La majorité de ces commandes utilise, explicitement ou non les éléments « e-class ».

Pour savoir quels éléments « e-class » sont disponibles à la suite d'une régression, il suffit de taper **ereturn list**. Pour récupérer le R^2 d'une régression, par exemple, **scalar** *rdeux=e(r2)*. La matrice des coefficients est **e(b)** et celle des variances-covariances **e(V)** [6]. Une fonction très utile est **e(sample)**. Elle permet de sélectionner dans la base les observations qui ont réellement été utilisée lors de l'estimation.

Tous les coefficients et écart-types estimés sont disponibles :
scalar *coeff*=**_b[***nomvariable***]** pour les coefficients (le nom de la constante est _cons) et **scalar** *stand*=**_se[***nomvariable***]** pour les écart-types.

Il est possible de stocker ces éléments pour utilisation ultérieure : **estimates store** *nom*[7]. En effet, si on lance une seconde régression, les éléments « e-class » de la première seront effacés.

Les commandes que l'utilisateur peut exécuter après une régressions sont **predict** et les multiples commandes de test.

4.6.1 predict

La commande **predict** permet d'obtenir, suite à l'estimation d'un modèle quelconque, les résidus, les valeurs prédites, *etc*... Attention : Stata applique par défaut la commande **predict** à toutes les observations de la base pour lesquelles les variables explicatives sont disponibles, même si l'estimation a été réalisée sur un sous-échantillon (prédiction *out-of-sample*). Pour n'appliquer la commande qu'aux observations effectivement utilisées lors de la régression (prédiction *in-sample*), **predict** *var* **if** *e(sample)*, options).

Les possibilités de la commande dépendent du modèle estimé. En général :
- **predict** *var*, xb permet d'obtenir les valeurs prédites pour la variable expliquée.
- **predict** *stand*, stdp permet d'obtenir l'écart-type de la prédiction linéaire.
- **predict** *residus*, re permet d'obtenir les résidus.
- **predictnl** permet d'obtenir des prédictions non-linéaires, lorsque le modèle s'y prête.

Des options supplémentaires apparaissent suivant le modèle estimé. Ainsi, après l'estimation d'un logit, **predict** *var*, pr permet d'obtenir la probabilité d'une issue positive. Se reporter à l'aide de chaque commande d'estimation pour connaître l'intégralité des options de **predict**.

4.6.2 Tests d'hypothèse sur les coefficients

Ces tests sont fondés sur la matrice de variance-covariance des estimateurs (tests de Wald). La commande est **test**. L'option accum permet d'ajouter le test précédent au test en cours.

[6] Sous Stata 9, il suffit de taper la commande **vce** pour l'obtenir.
[7] C'est indispensable pour procéder au test d'un modèle contre un autre. ⇒ Voir **hausman** section 3, **suest** et **lrtest**, section 3.

Exemples :

reg *salaire age homme diplome france italie allemagne espagne*

test *france=allemagne* — Test d'égalité de coefficients

test *italie* — Test de nullité du coefficient

test *espagne=1* — Teste que le coef soit égal à 1

test *(france = allemagne) (italie = espagne)*

Teste que le coef de fr. est égal à celui d'all. ET que le coef d'it. est égal à celui d'esp.

test *france = allemagne*

test *italie = espagne*, accum

Test joint au précédent, équivalent au test ci-dessus

Pour effectuer des tests d'hypothèses non-linéaires sur les coefficients, utiliser la commande **testnl**. Exemple : **testnl _b[***x1***]/_b[***x2***] = _b[***x3***]**.

En général, **test** est à utiliser avec précaution après l'estimation d'un modèle par le maximum de vraisemblance. Il vaut mieux dans ce cas procéder à des tests du ratio de vraisemblance (**lrtest**). Exemple :

logit *vote_bush salaire white relig state* — Modèle complet

est store *modele_complet* — Sauve les paramètres

logit *vote_bush salaire* — Modèle contraint

lrtest *modele_complet .* — LR-test

4.6.3 Tests classiques

Test de normalité d'une variable (les résidus...) : **sktest** *variable*.

Test d'hétéroscédasticité de Breush-Pagan : **hettest**[8]. La régression qui précède le test ne doit pas comporter l'option `robust`, évidemment. Si de l'hétéroscédasticité est détectée, relancer la régression avec l'option `robust`.

Test d'hétéroscédasticité de Goldfeld-Quandt : Si l'on connaît la variable à l'origine de l'hétéroscédasticité, on peut ordonner les observations selon la valeur de la variable suspecte, éliminer de l'échantillon les variables centrales (le tiers central de l'échantillon). On effectue la régressions sur les deux sous-échantillons composés des premières et dernières observations. On récupère la somme des carrés expliqués des deux sous-échantillons en vérifiant, afin de construire la statistique, que $SCR_1 > SCR_2$. On compare SCR_1/SCR_2 à $F(x-k, x-k)$ avec k le nombre de variables explicatives constante incluse, et x le nombre d'observations de chaque sous-échantillon.

Le test de stabilité des coefficients (test de Chow) n'est pas programmé sous Stata, il faut le faire soi-même. On doit estimer trois modèles (le modèle complet et les deux sous-modèles) et récupérer leurs sommes des carrés des résidus (SCR pour le modèle complet, SCR_1 et SCR_2 pour les deux sous-modèles). La statistique de test

$$\frac{SCR - (SCR_1 + SCR_2)}{SCR_1 + SCR_2} \times \frac{n-2k}{k}$$

[8] Sous Stata 9, **estat hettest**.

suit une loi de Fisher $F(k, n-2k)$, avec k le nombre de variables explicatives et n le nombre d'observations.

```
regress salaire age                                    Modèle complet
scalar scr=e(rss)                                      SCR
scalar n=e(N)                                          n
regress salaire age if sexe=="Femme"                   Sous-modèle 1
scalar scr1=e(rss)                                     SCR1
regress salaire age if sexe=="Homme"                   Sous-modèle 2
scalar scr2=e(rss)                                     SCR2
scalar stat_chow=(scr-(scr1+scr2)/(scr1+scr2))*((n-2*2)/2)
                                                       Statistique du test
display F(2,n-2*2, stat_chow)                          Probabilité du test
```

4.6.4 Tests spécifiques pour données de panel

Le test de stationnarité de Im-Pesaran-Shin (**ipshin** *variable*, lags(*nombre*) trend) et celui de Levin-Lin-Shu (**levinlin** *variable*, lags(*nombre*) trend) sont disponibles. L'option trend ajoute une tendance.

4.6.5 Tests pour les séries temporelles

Plusieurs tests de racine unitaire existent :

- **dfuller** *variable*, lags(*nombre*) trend permet de réaliser le test de Dickey – Fuller augmenté. L'option trend ajoute une tendance.
- **pperron** *variable*, lags(*nombre*) trend pour le test de Phillips-Perron.
- **dfgls** *variable*, lags(*nombre*) notrend pour le test DF-GLS. Il faut ajouter l'option notrend pour enlever la constante.
- **kpss** *variable*, lags(*nombre*) notrend pour le test de Kwiatkowski – Phillips – Schmidt – Shin.

En ce qui concerne les tests d'autocorrélation des résidus, récurrents lorsqu'on utilise des séries temporelles :

- Le test de Durbin-Watson permet de tester la présence d'un processus AR(1) dans les données (**dwstat**[9]). Si il y a autocorrélation, relancer la régression avec une correction de Prais-Winsten (**prais** *y x1 x2*).
- Lorsqu'on soupçonne la présence d'autocorrélation d'ordre supérieur à 1, le test de Breush-Godfrey (**bgodfrey**[10]) ou celui de de Durbin-Watson (**durbina**[11]) doivent être réalisés. Le test de Engle (**estat archlm**) permet de tester la présence d'éléments ARCH dans les résidus.

5 Graphiques

5.1 Faire un graphique

La création de graphiques avec Stata n'est pas toujours simple, en particulier lorsqu'on souhaite que les graphiques soient « présentables » dans un mémoire. Un manuel entier est consacré aux graphiques. Nous présentons ici les commandes pour faire des graphiques simples :

- Pour obtenir un graphique circulaire (un « camembert »), **graph pie** *salaire bonus prime avantages_nature*.
- Pour un diagramme en bâtons (ne pas le confondre avec un histogramme...) : **graph bar** *salaire*, over(*sexe*, *descend gap(-20)*) over(*pays*).
  Pour des bâtons verticaux : **graph hbar**.

[9] **estat dwatson** sous Stata 9.
[10] **estat bgodfrey** sous Stata 9.
[11] **estat durbinalt** sous Stata 9.

– Les chandeliers japonais : **graph box** *taille*, over(*sexe*) over(*annee*).

Pour tous les graphiques (X,Y), la commande débute par **twoway** suivi du type de graphique X,Y souhaité. Parmi les principaux types :

– **tw histogram** *variable*, options génère un histogramme. L'option normal ajoute à l'histogramme une loi normale, de mêmes moyenne et variance que la variable faisant l'objet de l'histogramme. L'option bin(*nombre*) définit le nombre de tranches.
– **tw scatter** *variable_ordonnee variable_abscisse*, options permet d'obtenir un nuage de points.
– **tw function** *y=exp(sin x)*, options permet d'obtenir une représentation graphique de la fonction spécifiée.
– **tw line** *variable_ordonnee variable_abscisse*, options permet d'obtenir un graphique avec des points reliés par une ligne.
– **tw kdensity** *variable*, options permet d'obtenir l'estimation de la densité du noyau de la variable.
– **tw area** *variable_ordonnee variable_abscisse*, options crée un graphique avec une ligne reliant les points (x,y) et une aire colorée entre la ligne et l'axe des abscisses.
– **tw rarea** *var_ordonnee1 var_ordonnee2 var_abscisse*, options permet d'obtenir un graphique avec une aire colorée correspondant à l'espace compris entre les valeurs de var_ordonnee1 et de var_ordonnee2.

Il est possible de superposer facilement des graphiques avec Stata. Ainsi, il est possible d'obtenir sur un même graphique le nuage de point, la droite de régression MCO et l'intervalle de confiance. Supposons que la variable expliquée soit *salaire*, la variable explicative *age* :

```
regress salaire age
predict val_pred, xb
predict ecarttype, stdf
gen interv_bas = val_pred - 1.96*ecarttype
gen interv_haut = val_pred + 1.96*ecarttype
scatter salaire age || line val_pred interv_bas interv_haut
age, sort
```

On peut également créer des ensembles de graphiques (graphiques les uns à côté des autres, voire des matrices de graphiques). Pour créer un ensemble de deux graphiques (un pour les hommes, un pour les femmes) représentant le nuage de points (age/salaire) : **twoway scatter** *salaire age*, by(*sexe*).

5.2 Options indispensables (à l'esthétique)

Les options sont indispensables à un graphique pour qu'il ressemble à quelque chose. Il en existe des centaines. Les principales options disponibles, qui fonctionnent avec la majorité des graphiques générés par **tw** *commande variables*, options :

– title("*titre*") définit le titre général du graphique. On peut y ajouter un sous-titre, subtitle("*sous-titre*").
– note("*bla-bla*") permet d'ajouter des notes au bas du graphique (sources, etc).
– legend(label(1 "*Hommes*") label(2 "*Femmes*")) permet de définir la légende. Ici, le mot « Homme » apparaîtra pour la première courbe, le mot « Femme » pour la seconde.
– En ce qui concerne la gestion des axes, on peut gérer leur échelle : xscale(*log*) (resp. yscale (*log*)) pour une échelle logarithmique sur l'abscisse (resp. l'ordonnée). Pour définir le titre des axes, on ajoute les options xtitle("*titre*") et ytitle("*titre*"). xlabel(*0 (10) 100*) permet d'afficher un axe des abscisses partant de 0 et allant

jusqu'à 100 par pas de 10. Stata est assez capricieux sur les longueurs des axes, et refusera cette option si par exemple la variable x prend la valeur 115 dans la base.

D'une manière générale, absolument tout ce qui apparaît dans un graphique peut être personnalisé, de la couleur de fond à l'épaisseur des axes, en passant par la place du titre. Toutefois, cela impose à l'utilisateur de préciser de nombreuses options. Une solution pour obtenir rapidement des graphiques présentables est de recourir aux styles prédéfinis (`schemes`) par Stata. Les `schemes(s1mono)` et `schemes(s1color)` permettent d'obtenir des graphiques N&B ou couleur rapidement.

Exemple : **`twoway scatter`** *`salaire age`*`, scheme(s1mono) xtitle("Age") title("Age et salaire") ytitle("Salaire").`

5.3 Sauver et exporter des graphiques

Pour sauvegarder un graphique au format Stata, **`graph save`** *`nom_fichier`*. Dans ce cas, le graphique est au format .gph, lisible exclusivement sous Stata.

Les graphiques Stata peuvent très facilement être exportés vers Word ou d'autres logiciels acceptant le copier/coller. En effet, un clic-droit Copier exécuté sur le graphique Stata copie l'image au format .wmf dans le presse-papier. Il ne reste plus qu'à la coller à l'endroit désiré.

Pour sauver le graphique dans un format autre que celui de Stata et pouvoir ensuite l'ouvrir dans un logiciel quelconque, **`graph export`** *`nom_fichier`*`, as(format)`. Les formats disponibles avec Stata pour Windows sont : .png, .wmf, .emf, .eps et .ps. On peut ajouter à la commande l'option `replace`.

Le plus simple pour intégrer un graphique Stata dans un document LaTeXest de sauvegarder le fichier au format .eps puis d'utiliser le package epsfig dans LaTeX. Une commande Stata permet de sauver le graphique en .eps et de générer le code à insérer dans le document LaTeX : **`graph2tex`**.

6 Divers

6.1 Exporter des résultats

Il existe plusieurs techniques, plus ou moins recommandables, pour exporter des résultats ou des tableaux statistiques vers d'autres logiciels (Word, Excel, LaTeX). De la moins recommandable à la « meilleure » :

1. Le faire « à la main » (Copier/Coller). Outre le temps perdu, la probabilité d'erreurs en recopiant est élevée. A éviter absolument.
2. En nettoyant un peu le fichier .log pour qu'Excel puisse l'ouvrir à peu près correctement. On reste dans le domaine de l'a peu près.
3. En utilisant la commande **`outreg`**. Cette commande permet d'obtenir des tableaux contenant les coefficients, les écart-types, les intervalles de confiance, et même les étoiles de significativé. La commande crée un fichier ASCII, lisible directement par Excel ou Word. Cette commande permet de créer un tableau unique à partir des résultats de deux régressions. Exemple :
   ```
   regress salaire sexe age diplome
   outreg using tab1.out, ctitle("Modèle 1 ")
   gen age2=age ^2
   regress salaire sexe age age2 diplome
   outreg using tab1.out, ctitle("Modèle 2") append
   ```
   Principal défaut : cette commande est ancienne et n'est plus mise à jour par son auteur.
4. La commande **`parmest`**`, saving(results.dta)` permet de sauver au format .dta les résultats de l'estimation.

5. Si on désire obtenir des tableaux au format LaTeX, en utilisant les commandes **outtex** (pour les résultats de régression) et **sutex** (pour les tableaux de statistiques descriptives). Ces commandes sont pratiques, rapides, et relativement souples. Principal défaut : **outtex** ne permet pas de construire un tableau avec des résultats provenant de plusieurs régressions.
6. Pour LaTeXet tous les autres programmes (Excel en particulier), utiliser la commande **estout**. La syntaxe de la commande est complexe, mais celle-ci permet d'obtenir toutes les tables possibles, d'agréger plusieurs régressions en un tableau, de sortir un fichier au format Excel ou LaTeX, *etc*...

6.2 Créer un *do file*

Un fichier .do est un fichier texte qui contient une suite de commandes Stata, que l'utilisateur peut faire exécuter par Stata. Pour leur création, le recours au Do-file Editor de Stata (**doedit**) est la solution la plus simple[12]. Nous présentons ici un exemple de .do.

```
capture clear                                  /*Nettoie la mémoire vive si besoin*/
capture log close                              /*Ferme le fichier log si besoin*/
log using monlog, text replace                 /*Ouvre un log*/
set memory 100m                                /*Fixe la mémoire vive à 100m*/
set more off                                   /*Pas d'interruption avant la fin du .do*/
use "mabase.dta"                               /* Ouvre mabase.dta*/
gen x1=ln(age)                                 /*Création de x1*/
gen x2=ln(educ)                                /*Création de x2*/
gen y=ln(salaire)                              /*Création de y*/
sum x1 y                                       /*Affiche les stat desc de x et y*/
tab y x1 x2                                    /*Fréq. des valeurs des variables*/
histogram y                                    /*Histogramme de y*/
graph export histo, as(png) replace
                                               /*Enregistre le graphe*/
reg y x1 x2                                    /*Régression MCO*/
outtex, leg                                    /*Affichage du tableau pour LaTeX*/
test x1=x2                                     /*Test d'égalité des coef de x1 et x2*/
save "mabase.dta",replace                      /*Sauve la base modifiée*/
log close                                      /*Ferme le log*/
clear                                          /* Vide la mémoire vive*/
exit                                           /*Ferme Stata*/
```

Pour « commenter » un passage du .do (*i.e.* que Stata ne le lise pas), il suffit de mettre au début /* et */ en fin.

L'utilisation de .do est indispensable dès que l'on veut faire « sérieusement » de l'économétrie. Les .do permettent d'appeler d'autres .do, de lancer des commandes complexes, de faire des boucles, *etc*...

6.3 Boucles

Il est possible d'intégrer des boucles à un .do. Les boucles possibles sont des boucles **while**, **foreach** ou **forvalues**. Quelques conseils :

– N'oubliez jamais de sauvegarder le .do avant de tester une boucle. L'oubli de l'incrémentation du compteur ferait boucler à l'infini Stata...

[12] Pour une utilisation régulière de Stata, le recours à un éditeur de texte plus performant fait gagner du temps. Ainsi, WinEdt – par exemple – permet d'ouvrir plusieurs .do en même temps, rend possible la coloration syntaxique du .do, offre des outils de validation du code, *etc*.

– L'emploi de boucles avec Stata est souvent inutile, il existe fréquemment des commandes permettant de l'éviter, ce qui permet d'obtenir un code plus compact et donc plus rapide.

Quelques exemples :

1. Cette boucle permet de fusionner 10 bases de données nommées temp1.dta, temp2.dta, . . . , temp10.dta en une seule, base_finale.dta.

```
use "temp1.dta"                                  On ouvre temp1
save "base_finale.dta",replace                   sauvée sous base_finale
local iter=2                                     Compteur de la boucle
while 'iter'<11{                                 Entrée dans la boucle
   clear                                         On nettoie la mémoire vive
   local chemin="temp"+string('iter')+".dta"
                                                 Définition de la base temp à traiter
   use 'chemin'                                  Ouverture de temp2, 3...
   qui compress                                  Optimisation mémoire
   qui append using "base_totale.dta"            Fusion
   qui save "base_totale.dta",replace            Sauvegarde
   local iter='iter'+1                           Incrémentation compteur
}                                                Sortie de la boucle
```

2. Cette boucle illustre la possibilité d'appliquer successivement à plusieurs variables les mêmes commandes. Celle-ci permet de créer un graphique représentant le salaire touché par chaque décile de salariés classés par âge, puis par taux d'absentéisme, puis de sauver le graphique.

```
tokenize "age absent"                            Deux variables
local i=1                                        Compteur annexe
while "'1'"!=""{                                 Entrée dans la boucle
   xtile quant = '1', nq(10)                     Création des quantiles
   bysort quant :egen moy=mean(salaire)          Moyenne par décile
   tw line moy quant, xtitle('1')                Graphique
   local chemin="fig"+string('i')+".png"         Chemin
   local i='i'+1                                 Incrément compteur
   graph export 'chemin', as(png) replace        Sauve graphique
   drop moy quant                                Suppression variables
   mac shift                                     Variable suivante
}                                                Sortie de la boucle
```

3. La troisième boucle recourt à **forvalues** pour générer des statistiques descriptives pour les variables x5, x10, . . . , x300.

```
forvalues k = 5 10 to 300{                       Entrée dans la boucle
   summarize x'k', det                           Stat. desc.
}                                                Sortie de la boucle
```

Table des matières