Introduction au logiciel STATA

Antoine Bozio*

28 janvier 2005

Table des matières

1 Les premiers pas **3**
1.1 Les différentes fenêtres . . . 3
1.2 Comment lire des données . . . 4
1.2.1 La commande `use` . . . 4
1.2.2 La commande `insheet` . . . 4
1.2.3 La commande `infile` . . . 5
1.2.4 Le logiciel Stat Transfert . . . 5
1.2.5 Le problème de mémoire insuffisante . . . 5
1.3 Comment sauvegarder données et travail . . . 5
1.3.1 Le do file . . . 5
1.3.2 Sauver les données . . . 5
1.4 Chercher de l'aide . . . 6
1.4.1 Dans les manuels . . . 6
1.4.2 Dans STATA . . . 6
1.4.3 Sur Internet . . . 6
1.5 Décrire les données . . . 6
1.5.1 Regarder les données brutes . . . 6
1.5.2 La commande `list` . . . 7
1.5.3 La commande `describe` . . . 7
1.5.4 La commande `summarize` . . . 7
1.5.5 La commande `tabulate` . . . 7
1.5.6 Les commandes `correlate` et `covariance` . . . 7

2 Travailler sur les données **8**
2.1 Réorganiser les données . . . 8
2.1.1 La commande `rename` . . . 8
2.1.2 Les commandes `recode` et `replace` . . . 8
2.1.3 Les commandes `keep` et `drop` . . . 8
2.1.4 Les commandes `sort` et `by` . . . 8
2.1.5 Comprendre les formats "wide" et "long" . . . 9
2.1.6 Combiner différentes bases de données : `append` et `merge` . . . 10
2.1.7 La dangereuse commande `collapse` . . . 11
2.2 Créer de nouvelles variables . . . 11
2.2.1 Les commandes `generate`, `egen` et `replace` . . . 11
2.2.2 Comment créer des variables avec retard . . . 12
2.2.3 Passer de variables textes à des variables numériques . . . 12
2.2.4 Combiner ou réduire des variables . . . 13

*Antoine Bozio, Ecole Normale Supérieure - EHESS - Campus Paris Jourdan, 48 boulevard Jourdan, 75014 Paris, France. `antoine.bozio@ens.fr`

2.3 Réaliser des graphiques . . . 13
2.4 Les estimations MCO . . . 13
2.4.1 La commande `regress` . . . 13
2.4.2 Les poids . . . 13
2.4.3 Comment créer des variables muettes . . . 14
2.4.4 Prédiction . . . 14
2.4.5 Extraire les résultats . . . 15
2.4.6 Les tests d'hypothèses . . . 15
2.5 Les autres types d'estimation . . . 15
2.5.1 Estimation linéaire contrainte . . . 15
2.5.2 Estimation par variable instrumentale . . . 15
2.5.3 Les autres estimations . . . 15

3 La programmation **16**
3.1 Comment programmer . . . 16
3.2 Macros . . . 16
3.3 Réaliser une boucle . . . 17
3.3.1 La commande `for` ou `foreach` et `forvalues` . . . 17
3.3.2 La commande `while` et comment créer un incrément . . . 17
3.4 Programmer en ramification (branching) . . . 18
3.5 Réaliser des simulations Monte Carlo . . . 18

Ce polycopié vise à aider les étudiants à se lancer à l'assaut de l'économétrie appliquée en se familiarisant avec le logiciel le plus complet et le plus facile d'accès, c'est-à-dire STATA.[1] Il y a bien sûr d'autres logiciels économétriques (SAS est l'un des plus répandus au sein des grosses institutions qui utilisent l'outil économétrique ; E-views est simple mais parfois aussi limité ; les autres sont inconnus de l'auteur de cette note) mais STATA est le plus répandu dans les universités américaines et de façon croissante en Europe. Il ne s'agit nullement de viser à l'exhaustivité de la présentation de ce logiciel (il existe des manuels de référence en cinq tomes pour cela) mais simplement d'aider à la pratique rapide de l'économétrie appliquée.

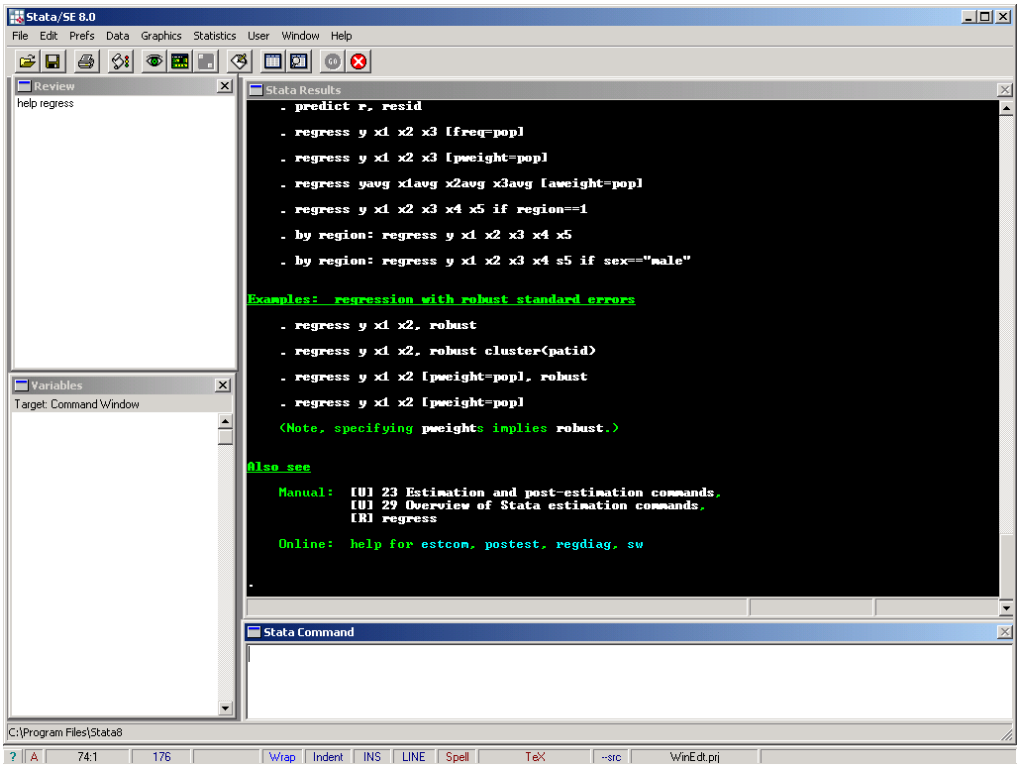
Enfin l'auteur accueille volontiers les commentaires, conseils et corrections que les lecteurs voudront bien lui communiquer.

1 Les premiers pas

1.1 Les différentes fenêtres

A la figure 1, on a reproduit un écran type de Stata. Quatre fenêtres sont repérables :

FIG. 1 – L'écran de Stata



Fenêtre résultat En haut à droite sur fond noir, la fenêtre décrit les résultats des commandes, des régressions.

Fenêtre de commande En bas à droite, la fenêtre commande permet de taper ici des commandes qui peuvent être exécutées par la touche Entrée ↩. On n'utilise cette fenêtre que pour essayer de façon interactive des commandes mais pas pour rédiger un programme.

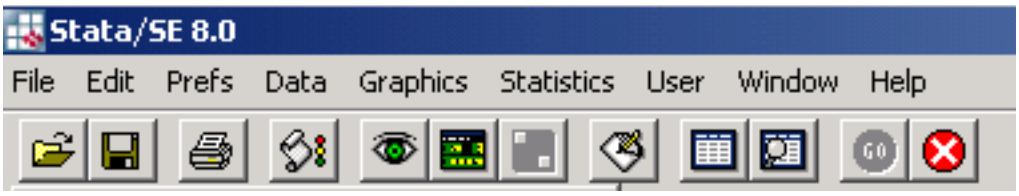
[1] L'auteur reconnaît que son parti pris reste subjectif et tout en précisant qu'il n'a aucune relation financière avec la société distributrice de STATA, souligne trois avantages majeurs : Stata est simple car il est possible d'apprendre des commandes de façon interactive ; avec l'augmentation de la mémoire vive des ordinateurs, l'avantage originel de SAS de pouvoir gérer d'énormes bases de données disparaît puisque Stata peut aussi le faire maintenant ; enfin Stata est rapide puisqu'il utilise les données directement en mémoire.

Fenêtre de variables En bas à gauche la fenêtre de variables liste les variables avec les "labels" de celles-ci quand elles existent. Il suffit de cliquer sur l'une d'elles pour qu'elle soit saisie par la fenêtre commande.

Fenêtre de commandes passées En haut à gauche la fenêtre des commandes passées récapitule les commandes déjà utilisées et il suffit de cliquer sur l'une d'elle pour la rappeler dans la fenêtre commande.

La barre d'outil de Stata permet facilement de gérer les actions de base. De gauche à droite : ouvrir un fichier de donnée Stata, sauvegarder un fichier de donnée (équivalent à la commande `save`), imprimer les résultats tels qu'ils apparaissent dans la fenêtre de résultats, commencer un log qui est une procédure qui stocke dans un fichier ".log" les commandes suivis des résultats, un bouton pour faire apparaître la fenêtre résultat lorsqu'elle n'est pas présente, un bouton avec une enveloppe qui permet de créer un do file (on revient là-dessus plus bas), deux éditeurs des tableaux de données et finalement un bouton pour stopper la commande en cours (si elle est longue et que vous vous êtes aperçu d'une erreur dans votre programme par exemple).

FIG. 2 – La barre d'outil de Stata



1.2 Comment lire des données

1.2.1 La commande `use`

Si vous disposez des données sous format Stata (qui finissent par `.dta`) alors la commande use vous permet simplement de lire les données. Il y a deux façons de procéder, selon que l'on souhaite travailler dans un dossier seul ou juste faire appel au fichier de données :

```
.use "C :\Econometrie\TDmaitrise\ee2002.dta"
```

ou bien

```
. cd "C :\Econometrie\TDmaitrise"
. use ee2002
```

Il faut parfois rajouter l'option `clear`[2] afin d'effacer le fichier de données déjà utilisé par le logiciel.

```
. use ee2002, clear
```

1.2.2 La commande `insheet`

Si vos données sont sous la forme d'un fichier CSV (comma separated values, ou finissant par `.csv`) alors il faut utiliser la commande `insheet`.

```
. insheet using "C :\Econometrie\TDmaitrise\ee2002.csv"
```

Les seules consignes à respecter avec cette commande sont la préparation du fichier de données en indiquant sur la première ligne le nom des variables qui sont disposées en colonne. Il ne faut pas laisser de noms de variables sous format numérique et de préférence transformer les dates (en variables) avec un underscore : 2004 en _2004. Vérifier qu'il n'y a pas de virgule dans les données pour ne pas troubler la reconnaissance des variables/colonnes.

[2] Les options sont situées après les commandes et après une virgule.

1.2.3 La commande `infile`

Si vous avez des données sous format ASCII, format texte `.txt` ou `.asc`, alors il suffit d'utiliser la commande `infile` avec un inconvénient : il faut préciser à la main le nom des variables.

```
. infile age edu salaire using "C :\Econometrie\TDmaitrise\ee2002.dta"
```

1.2.4 Le logiciel Stat Transfert

Enfin si vous avez accès au logiciel Stat Transfert, vous pouvez facilement convertir n'importe quel format de données (SAS, Excel, SPSS, ASCI, Gauss, Matlab...) en format de Stata et inversement. Il faut pourtant toujours vérifier que le transfert n'a pas altéré les données et a bien pris en compte toutes les variables.

1.2.5 Le problème de mémoire insuffisante

Si vous n'avez pas assez de mémoire vive disponible pour Stata (le message d'erreur est : `no room for more observations`) alors il faut utiliser la commande `set memory XXm` pour préciser le nombre de mégabytes[3] que vous souhaitez allouer à Stata.

1.3 Comment sauvegarder données et travail

1.3.1 Le do file

La bonne façon de travailler sous Stata est d'utiliser un fichier .do (un do file) comme fichier de travail. On écrit ainsi un programme que l'on peut sauvegarder et réutiliser la fois suivante. L'utilisation de la fenêtre commande est plus pour tester rapidement des variantes ou pour s'exercer au début avec les commandes. Pour exécuter un do file, il suffit de sélectionner la partie du programme que l'on souhaite appliquer et cliquer sur l'icône représentant une flèche vers le bas à côté d'une feuille, "do current file".

Pour commencer un do file il faut préciser dans quel dossier on travaille. La syntaxe est la même que dans du DOS :

```
. cd "C :\Econometrie\TDmaitrise"
```

Stata execute les commandes par ligne et chaque passage à la ligne correspond à un nouvel ensemble de commandes. Si vous voulez rédiger des commandes de longues lignes (par exemple lister les nombreuses variables que vous voulez conserver dans l'enquête) alors il est utile de commencer votre do file par la commande `#delimit ;` et de finir chaque commande par `;` ainsi une commande sur plusieurs lignes sera exécutée en une fois sans problème.

1.3.2 Sauver les données

La commande la plus simple est `save`. On ne peut l'utiliser simplement que lorsque le fichier de données n'existe pas encore. Sinon il faut ajouter l'option `replace`.

```
. save ee2002, replace
. save ee2002-a
```

Si vous souhaitez opérer des changements qui ne soient pas définitifs (par exemple dans un do file), alors il suffit d'ajouter l'option `clear`. La commande `clear` efface du logiciel la base de données qu'il était en train de lire (elle n'est pas supprimée).

```
. use ee2002, clear
```

[3] Vous pouvez aussi préciser des bytes `b`, des kilobytes `k` ou gigabytes `g`.

1.4 Chercher de l'aide

1.4.1 Dans les manuels

Il y a trois types de manuels STATA. Le plus simple est le *User manual* qui décrit en moins clair les éléments de base qui se trouvent dans le poly que vous avez dans vos mains. Puis vous avez les volumes *References* qui sont un dictionnaire de Stata en 6 volumes, très précis et complet sur chaque commande, mais dont l'inconvénient majeur est qu'il faut connaître la commande avant de les consulter. Enfin, il existe un volume séparé *Graphics Manual* pour faire de jolis graphiques.

1.4.2 Dans STATA

Le logiciel a aussi une version abrégée du manuel en ligne. Il suffit pour y faire référence de cliquer sur Help. Tout en travaillant, on peut rappeler la description d'une commande en utilisant la commande `help` suivi de la commande dont on veut connaître le mode d'utilisation.

```
. help regress
```

Si on ne connaît pas le nom de la commande, il est pratique d'utiliser la commande `search` et de préciser en anglais ce que l'on cherche. Le logiciel d'aide vient en suite suggérer des entrées possibles pour votre demande.

```
. search prediction
```

1.4.3 Sur Internet

Enfin, si vous ne trouvez pas votre bonheur par ces manuels, n'oubliez pas la mine d'or d'Internet. Au sein de Stata, vous pouvez utiliser la commande `net search` *mot clé* pour trouver des programmes stata ou des sites référencés. Je cite ci-dessous quelques sites Internet, trouvés au petit bonheur la chance, mais vous pourrez sûrement en trouver d'autres par Google sur des points spécifiques.

`http ://www.stata.com/support/faqs/` : C'est le site d'aide de la société qui produit STATA. Vous y trouverez beaucoup d'informations et surtout des détails concernant les nouvelles versions et les améliorations récentes.

`http ://www.princeton.edu/~erp/stata/main.html` : c'est le site d'apprentissage de Stata de l'université de Princeton. Bien fait avec des exemples visuels.

`http ://www.ats.ucla.edu/stat/stata/` : c'est le site d'apprentissage de Stata de l'université UCLA. Énormément d'informations qui sont très facilement utilisables. A consulter !

`http ://econ.lse.ac.uk/courses/ec502/stata/` : c'est le cours de Stata de la LSE (London School of Economics). Les notes de cours sont bien faites, et sont dans le même esprit que ce polycopié, avec plus d'exemples et des exercices à faire.

`http ://www.econ.ucdavis.edu/faculty/cameron/stata/stata.html` : C'est le cours de Stata de l'Université de Californie Davis. Il y a un certain nombre d'exemples de fichiers `.do` et de base de données pour travailler des exemples.

`http ://www.hec.unil.ch/schmidheiny/sea2/` : C'est le cours d'économétrie appliquée de l'université suisse de HEC Lausanne. Les fiches sur des points précis sont relativement bien faites.

1.5 Décrire les données

1.5.1 Regarder les données brutes

La première chose à faire une fois qu'on a chargé les données dans le logiciel est d'aller les regarder grâce à l'éditeur. On clique sur la fenêtre avec un fichier (ou on tape la commande `edit`) ou on clique sur la fenêtre avec un fichier et une loupe juste à droite (ou on tape la commande `browse`). Avec la première commande vous pouvez rajouter manuellement des observations, avec la seconde vous ne pouvez que regarder... Il est souvent assez simple de vérifier que les données correspondent bien à ce que vous imaginer, que la variable age ne va pas de -345.568 à 245864 par exemple. Utilisez votre bon sens cartésien !

1.5.2 La commande `list`

La commande `list` toute seule donne l'ensemble des données de façon similaire à `browse` mais sur la fenêtre résultat. Il est évident que cela n'a de sens que si les variables sont peu nombreuses et les données réduites. Mais il est possible d'utiliser `list` suivi de la variable d'intérêt.

```
. list salaire
```

Le résultat est la suite de toutes les observations pour cette variable. Mais il est possible aussi de demander uniquement quelques observations. Ainsi, pour les 10 premières observations : `. list salaire in 1/10`
Ou bien les 10 dernières observations :

```
. list salaire in -10/l
```

1.5.3 La commande `describe`

La commande `describe` décrit des informations de base sur les variables de la base. En ajoutant une variable après `describe` seules les informations sur ces dernières sont affichées. On trouve le type de variable et le format de leur compression, le label...

```
. describe salaire
```

On en profite pour faire un point sur les différents formats de stockage des données. Il y a différents formats afin de minimiser la place de mémoire utilisée par les données. Certaines variables n'ont pas besoin de huit caractères et peuvent être disponible sous un format plus léger. Il y a deux types de base de variables : soit numériques soit texte (string[4]). Dans le cas de variables numériques, il faut distinguer 5 cas :

byte nombre entier entre -127 et 126, comme une variable muette

int nombre entier entre -32'767 et 32'766, comme une année

long nombre entier entre -2'147'483'647 et 2'147'483'646, comme une donnée de population

float nombre réel avec une précision de 8 chiffres, comme des données de production

double nombre réel avec une précision de 16 chiffres

1.5.4 La commande `summarize`

Beaucoup plus intéressant pour vous sera la commande `summarize` (abrégée en `su`). Elle affiche le nombre d'observations, la moyenne, l'écart-type, et les valeurs extrêmes. Si on souhaite plus de précision, il suffit de rajouter une option `detail`.

```
. su salaire, detail
```

1.5.5 La commande `tabulate`

Une commande particulièrement pratique est la commande `tabulate` (abrégée en `ta`). Elle donne le nombre d'observations et la fréquence de chaque valeur de la variable. Elle peut donc utilisée pour regarder la distribution d'une variable. Avec deux variables, elle présente un tableau croisé souvent très utile.

1.5.6 Les commandes `correlate` et `covariance`

Pour obtenir la correlation ou la covariance entre deux (ou plus) variables, il suffit d'utiliser ces deux commandes :

```
. correlate var1 var2
. covariance var1 var2
```

[4]En anglais string signifie chaîne, pour chaîne de caractère

2 Travailler sur les données

2.1 Réorganiser les données

2.1.1 La commande `rename`

Le premier travail de l'économètre est souvent de nettoyer son fichier et pour cela il est parfois nécessaire de renommer les variables sous des noms plus pratiques. La commande `rename` (abrégée en `ren`) permet de changer le nom de la variable qui suit.

```
. rename salredty salaire
```

2.1.2 Les commandes `recode` et `replace`

Le codage des variables n'est pas forcément optimal et parfois faux. L'utilisation de la commande `recode` permet de remédier à ces problèmes :

`. recode marital 1=3` si par exemple le code pour célibataire devrait être 3 au lieu de 1

`. recode salaire 99999=.` si le codage du salaire donne 99999 en cas d'absence de réponse. Le point est considéré comme une observation manquante.

Avec une variable texte (string) il faut utiliser la commande `replace` et la commande logique `if` :

```
. replace pays="Allemagne" if pays=="Germany"
```

Ceci nous amène à effectuer quelques précisions indispensables sur les opérateurs logiques sous Stata :

`==` égal à	`~=` pas égal à
`<=` plus petit ou égal à	`>=` plus grand ou égal à
`>` plus grand que	`<` plus petit que
`&` et	`~` négation
`|` ou	`!` négation

2.1.3 Les commandes `keep` et `drop`

Pour travailler sur une base de donnée pratique en vue des objectifs que vous avez, il sera peut-être nécessaire de supprimer les variables inutiles ou les observations non concernées par vos estimations. La variable `keep` vous permet de garder et `drop` de jeter... facile, non ? On les utilise alternativement selon le nombre de variables à garder ou à jeter.

```
. keep age salaire pays marital
. drop age15 salred salaire150 fdsrt azerty
```

Avec les observations et les commandes logiques, il est possible de préciser ce que l'on veut effacer en le conditionnant à la valeur d'autres variables. Par exemple, on garde les plus de 15 ans :

```
. keep if age>=15
```

ou bien on supprime les individus nés en 1945 et 1968 :

```
. drop if naissance==1915 | naissance==1968
```

Une commande importante sous Stata est `_n` qui donne le rand de l'observation, ainsi que `_N` qui désigne le nombre d'observation dans la base de donnée.[5] Ainsi, si vous voulez supprimer les 15 premières observations, il suffit de :

`. drop if _n<=15` ou pour supprimer la dernière observation : `. drop if _n==_N`

2.1.4 Les commandes `sort` et `by`

La commande `sort` (abrégée en `so`) classe les données par ordre croissant. Il est possible de préciser les variables selon lesquelles le classement peut être effectué :

[5] `_pi` donne la valeur de π à la précision de l'ordinateur.

```
. sort sexe age
```

Cette commande va classer les observations par sexe (d'abord les femmes en numéro 0 et puis les hommes en numéro 1, par exemple) puis au sein de chaque sexe par age (les femmes et les enfants d'abord). On peut utiliser la commande `gsort` pour effectuer des classements dans des ordres croissant ou décroissant. Un + ou un - vient donner le sens du classement au sein de chaque variable.

```
. sort sexe -age
```

Cela classe d'abord par sexe puis par âge décroissant (les femmes et les vieux d'abord).

Le processus `by ... :` qui doit suivre obligatoirement un classement avec `sort` permet d'utiliser la plupart des commandes pour chaque valeur de la variable indiquée par `by`. Les exemples suivants vont vous aider à comprendre le principe :

```
. sort sexe
. by sexe : ta age
```

On obtient ainsi la distribution par âge d'abord pour les femmes, puis pour les hommes. Cela revient en fait à faire successivement les commandes :

```
. ta age if sexe==0
. ta age if sexe==1
```

Lorsque la variable prend plus de 2 valeurs, l'utilisation de `by` est donc très économique.

2.1.5 Comprendre les formats "wide" et "long"

Les formats "wide" et "long" sont des concepts importants dans l'organisation des données. Un format "wide" (en anglais large) correspond à l'utilisation de différentes variables en lieu et place de différentes observations. Le fichier de données est donc plus large car les variables sont rangées en colonne. A l'inverse, le format "long" correspond à l'utilisation minimale des variables avec plus d'observations, ce qui allonge la base de donnée, puisque les observations sont rangées en ligne. Un exemple va clarifier le concept :

Tab. 1 – Format long

pays	année	pop
Allemagne	1996	81896
Allemagne	1997	82052
Allemagne	1998	82029
Allemagne	1999	82087
Allemagne	2000	82168
France	1996	59533
France	1997	59735
France	1998	59942
France	1999	60156
France	2000	60431
Italie	1996	57397
Italie	1997	57512
Italie	1998	57588
Italie	1999	57646
Italie	2000	57728

Source : Penn World Table

La même base de données en format wide donne le tableau suivant :

Tab. 2 – Format wide

pays	pop1996	pop1997	pop1998	pop1999	pop2000
Allemagne	81896	82052	82029	82087	82168
France	59533	59735	59942	60156	60431
Italie	57397	57512	57588	57646	57728

Source : Penn World Table

La plupart du temps les données seront sous la forme "long", ce qui est plus pratique pour travailler avec Stata. Mais parfois vous aurez à faire à des données en "wide", et il vous faudra les convertir facilement en "long". La commande `reshape` sera donc particulièrement utile :

```
. reshape long pop, i(pays) j(année)
```

La commande précédente transforme le tableau 2 en tableau 1. L'indicatrice `i()` désigne la ou les variables que vous souhaitez maintenir dans le format actuel et `j()` désigne la ou les variables que vous souhaitez transformer en "long". Dans l'exemple précédent, le pays est inchangé puisqu'il était en "long", mais année va être créée avec pop à partir des variables pop1996, pop1997 etc... Si la variable `j()` est de format texte (string) alors il faut préciser l'option string. Si vous souhaitez au contraire passer du format "long" à "wide", il suffit de remplacer long par wide. Par exemple :

```
. reshape wide pop, i(pays) j(année)
```

2.1.6 Combiner différentes bases de données : `append` et `merge`

Pour travailler de façon efficace, il faut souvent réunir différentes bases de données. Selon le type de combination, on va utiliser une commande différente.

Ajouter des observations Si vous disposer par exemple de données sur l'emploi dans différents pays et que vous avez une base de donnée par pays avec les mêmes variables (emploi, salaire, temps de travail...), alors vous souhaitez ajouter des observations (rajouter des lignes). Votre premier soin est de créer une variable pays dans chaque base de donnée en indiquant pour toutes les observations de ce pays le même nom ou code. Ensuite vous pouvez utiliser la commande `append` de la façon suivante :

```
. use "C :\enqueteFrance.dta"
. append using "C :\enqueteAllemagne.dta"
. append using "C :\enqueteItalie.dta"
```

Ajouter des variables Si vous souhaitez ajouter des variables, alors il faudra utiliser la commande `merge`. Par exemple, vous avez deux bases de données sur entreprise (les mêmes entreprises) et l'une donne des informations sur la production et l'autre sur les salariés. Si vous voulez calculer la productivité de ces entreprises, il faudra combiner ces deux bases. La procédure est légèrement plus complexe qu'avec `append`. Etant donné que certaines variables sont communes aux deux bases (au moins l'identifiant des entreprises), il faut classer ces variables avec `sort` dans les deux bases pour permettre au logiciel de faire la bonne fusion.

```
. use "C :\production.dta"
. so identif année
. save "C :\production.dta", replace
. use "C :\salariés.dta"
. so identif année
. merge using "C :\production.dta"
. tab _merge
```

La commande merge crée la variable `_merge` qui permet de vérifier que la fusion a été réalisée comme voulu. Elle peut prendre trois valeurs :

1. Les observations de la base principale n'ont pas été retrouvées dans la base ajoutée (celle après `using`)
2. Les observations de la base ajoutée n'ont pas été retrouvées dans la base principale
3. Les observations dans les deux bases ont été retrouvées et connectées.

Il faut toujours vérifier que l'opération s'est bien déroulée en regardant si `_merge` prend des valeurs différentes de 3. Si ce n'est pas le cas alors regardez pour quelles observations l'opération n'a pas fonctionné.

2.1.7 La dangereuse commande `collapse`

La commande `collapse` transforme la base de données en statistiques essentielles sur celle-ci, comme en moyenne, somme, médiane... C'est particulièrement utile si vous souhaitez obtenir une base de données annuelle à partir de données trimestrielles ou mensuelles. Une autre utilisation possible est d'obtenir facilement des statistiques dans l'éditeur que vous pouvez copier dans un tableur (Excel ou autre). Le danger de cette commande est qu'elle efface les données sources. Il faut donc bien vérifier que vous avez sauvegardé le fichier de données sous un autre nom auparavant.

Prenons l'exemple d'une transformation d'une variable population mensuelle en population annuelle. La première parenthèse donne le type de transformation (`sum`=somme ; `mean`=moyenne ; `sd`=écart-type ; `median`=médianne) puis il faut préciser la variable qui subit la transformation. Après une virgule et la commande by, entre parenthèses on inclut les variables qui servent de référence à la transformation ; ici on fait donc la somme de la variable `popmois` par pays et par année. Il faut renommer la variable transformée pour qu'elle désigne ensuite ce qu'elle représente vraiment.

```
. collapse (sum) popmois, by(pays année)
. rename popmois popannée
```

Une deuxième raison de se méfier de l'utilisation de `collapse` est que cette commande traite les observations manquantes comme des zéros. Dans l'utilisation des moyennes, il faut donc se méfier et recoder si nécessaire les observations manquantes avant d'utiliser cette commande.

2.2 Créer de nouvelles variables

2.2.1 Les commandes `generate`, `egen` et `replace`

Les deux variables principales pour créer des variables sont `generate` et `egen`. La commande `generate` (abrégée en `gen`) est la plus simple, utilisée pour toutes les manipulations algébriques des données et `egen` (en anglais l'abréviation de *extended generate*) sert plutôt pour les créations de variables avec des combinaisons de moyennes, sommes, médianes etc...

Quelques exemples pour évoquer les manipulations algébriques :

```
. gen age2= age^2 /* on crée la variable age au carré*/
. gen lnsalaire= ln(salaire) /* on crée la variable logarithme base 10 du salaire*/
. gen id= _n /* on crée la variable id nombre d'observation*/
```

On en profite pour lister quelques expressions mathématiques sous Stata :
`abs(x)` renvoie la valeur absolue de x
`exp(x)` renvoie l'exponentielle de x
`int(x)` renvoie le nombre entier de x par troncature vers 0 (c'est-à-dire que `int(3.4)=3` et `int(-3.4)=-3`)
`ln(x)` ou `log(x)` renvoient au log de x
`sign(x)` renvoie -1 si x ¿0, 0 si x=0 et 1 si x¿0

`sqrt(x)` renvoie la racine carrée de x

Quelques exemples d'utilisation de la variable `egen` :

```
. egen popmondiale= sum(pop), by(année) /* on crée la variable population mondiale par année*/
. egen moypop= mean(pop) /* on crée la variable population moyenne par année*/
. egen maxpop= max(pop)  /* on crée la variable valeur de la plus large population*/
. egen difference= diff(pop1 pop2) /* on crée la variable muette égale à 1 si les deux variables sont différentes et 0 sinon*/
. egen moyenne= rmean(pop1 pop2) /* on crée la variable moyenne entre plusieurs variables, pour chaque observation*/
. egen medpop= median(pop) /* on crée la variable population médiane par année*/
. egen sdpop= sd(pop) /* on crée la variable écart-type de la population par année*/
```

Enfin la commande `replace` permets de modifier une variable déjà existante :

```
. replace salaire= salaire/6.55957 /* on crée la variable salaire en Euros*/
. replace age=99 if age>=100
```

Lorsqu'on crée une nouvelle variable, il est souvent préférable de préciser la définition de cette variable sous la forme d'un label :

```
. label variable sdpop "écart-type de la population"
```

2.2.2 Comment créer des variables avec retard

Imaginons que vous souhaitiez que vous avez une base de données de différents pays avec les valeurs par année du PIB. Vous cherchez à créer une variable retardée du PIB :

```
. so pays année
. by pays : gen pib_ret = pib[_n-1] if année==année[_n-1]+1
```

Une autre façon de procéder si vous disposez de données temporelles est d'utiliser la commande `tsset` qui permet de déclarer une variable comme une série temporelle.

```
. gen t=_n
. tsset t
```

Les variables retardées pourront être créées automatiquement sous la forme de `L.nomdevariable` pour un retard ou `L2.nomdevariable` pour deux retards

2.2.3 Passer de variables textes à des variables numériques

Deux cas de figure peuvent se présenter. Soit il s'agit de variables numériques *per se* (comme age, salaire) mais qui sont stockées en string, et alors il suffit d'utiliser la commande `destring`.

```
. destring age, replace
. destring age, gen(age_numerique)
```

L'autre cas de figure est une variable string par nature (comme pays) que vous souhaitez coder en numérique. Dans ce cas, les commandes `encode` pour passer en numérique ou `decode` pour revenir en string, seront nécessaires.

```
. encode pays, gen(pays1)
. decode pays1, gen(pays)
```

Avec `encode`, la variable pays1 est conservée sous format numérique et non plus caractère.

2.2.4 Combiner ou réduire des variables

Dans de nombreuses enquêtes, comme par exemple l'Enquête Emploi de l'INSEE, les secteurs d'activité ou les CSP sont classés selon une nomenclature à plusieurs chiffres en forme d'arbre. Par exemple pour les secteurs d'activité, la variable secteur7 définit 7 grand secteurs codés de 1 à 7, puis la variable secteur55 qui offre une nomenclature à 2 chiffres et secteur500 une nomenclature à 3 chiffres. Un autre exemple serait les identifiants dans les enquêtes de l'INSEE composés d'une variable ind pour un code individu, loc pour un code pour la localisation et im pour l'identifiant immeuble. Pour créer une variable identifiant globale, indlocim par exemple on peut utiliser la procédure suivante : il faut d'abord convertir toutes ces variables en string puis créer la variable combinée.

```
. gen str11 indlocim=ind+loc+im
```

A l'inverse pour décomposer une longue variable en deux, on utilise la procédure `substr(nom de la variable, rang de début, rang de fin)`.

```
. gen str7 locim=substr(indlocim,5,11)
```

2.3 Réaliser des graphiques

Pour réaliser des graphiques, il y a une multitude de commandes et d'option pour présenter de jolis graphiques. Pour une première approche, il suffit de connaître les quelques commandes suivantes :

Pour faire un histogramme de distribution :

```
. graph twoway histogram salaire
```

Pour représenter des observations sous forme de points en deux dimensions

```
. graph twoway scatter population surface
```

Au plus simple, on peut faire :

```
. graph population surface
```

2.4 Les estimations MCO

2.4.1 La commande `regress`

La commande `regress` (en abrégé `reg`) correspond à l'estimation MCO. La variable dépendante est donnée après la commande et les variables explicatives sont notées à la suite. Une constante est insérée par défaut et il faut utiliser l'option `noconstant` pour faire une régression sans constante.

```
. reg salaire age edu experience csp if sexe==1 & pays="france"
```

Avec l'option `robust`, les écarts-types sont estimés par la méthode de White.

2.4.2 Les poids

Dans les enquêtes, on trouve en général une variable pond qui correspond à la pondération de chaque observation (combien de personnes la personne interrogée représente dans la population totale). La pondération peut être utilisée dans les calculs de Stata, et en particulier avec les régressions. On utilise pour se faire l'option `weight`. On trouve quatre types de poids (weights) :

fweight : pour *frequency weights* ; ce sont les observations qui sont dupliquées en autant de fois que la pondération l'indique. Si pond=127, stata va faire ses calculs en imaginant qu'il y a en fait 127 observations avec les mêmes valeurs de variables. Il ne faut pas utiliser ce type de pondération avec des enquêtes. Vous obtiendrez *de facto* une très forte significativité de vos

coefficients alors que vous ne disposez pas de ces données sur toute la population. La seule utilisation justifiable de `fweight` est si vous disposez de données administratives sous forme de tabulation.

pweight pour *probability weights* ; cela indique l'inverse de la probabilité que cette observation soit inclue dans l'échantillon. Par exemple pond=100 signifie que cette observation a la probabilité 1/100 d'être dans l'échantillon.

aweight pour *analytic weights* ; ils sont inversement proportionnel à la variance de chaque observation cad la variance de la jème observation est supposée être $\frac{\sigma^2}{\omega_j}$ où ω_j est la pondération. Typiquement, les observations représentent des moyennes et les pondérations sont le nombre d'éléments qui ont donné lieu à ces moyennes.

iweight pour *importance weights* ;pas de définition précise...

Lorsque vous ne précisez pas de weight, stata fait comme si `[fweight=1]`. La pondération se place à la fin de la commande entre crochets :

```
. reg salaire age edu experience csp if sexe==1 & pays="france" [aweight=pond]
```

2.4.3 Comment créer des variables muettes

Pour créer des variables muettes ou des variables indicatrices (*dummy variable* en anglais) trois méthodes sont possibles. La première est simple et "à la main" :

```
. gen age12-25 =0
. replace age12-25 =1 if age>=12 & age<=25
. gen age26-60 =0
. replace age26-60 =1 if age>=26 & age<=60
. gen age60 =0
. replace age60 =1 if age>60
```

La seconde méthode (automatique) est beaucoup plus rapide. Il s'agit d'utiliser la commande `tabulate` avec l'option `gen`. Par exemple, la variable `edu` prend 4 valeurs (1, 2, 3 et 4) et vous voulez créer 4 variables indicatrices pour chacune de ces valeurs, il vous suffit de taper :

```
. tabulate edu, gen(edu)
```

La troisième méthode (automatique) est utile lorsque vous voulez créer automatiquement des variables muettes en faisant une régression. Dans ce cas, il faut utiliser la procedure `xi :`

```
. xi : reg salaire age edu experience i.csp
```

Dans l'exemple précédent, on a créé une liste de variable muette à partir des valeurs de la CSP (csp_12, csp_24, csp_36...). Le `i.nomdevariable` désigne la variable dont on veut créer la séquence de variables muettes.

2.4.4 Prédiction

Grâce à la commande `predict`, si elle suit une régression, vous pouvez créer la prédiction de la variable dépendante $\hat{y}$:

```
. reg salaire age edu experience csp if sexe==1 & pays="france" [aweight=pond]
. predict salaire_predit
```

Pour obtenir l'écart-type de la prédiction, il faut ajouter l'option `stdp`. Si on ajoute l'option `resid`, on obtient une variable des résidus :

```
. reg salaire age edu experience csp if sexe==1 & pays="france" [aweight=pond]
. predict residu, resid
```

2.4.5 Extraire les résultats

Il est possible de stocker les résultats dans Stata avec les commandes `estimates` :

```
. reg salaire age edu experience csp if sexe==1 & pays="france" [aweight=pond]
. estimates store coeff
. estimates table coeff
```

Si vous souhaitez réutiliser les résultats dans un calcul, il suffit de faire référence aux variables `_b[`*nomdevariable*`]` qui sont les coefficients de la régression précédente et aux variables `_se[`*nomdevariable*`]` qui sont les écarts-type. Pour la constante, il suffit de faire référence à `_b[_cons]`.

Certains d'entre vous souhaitent peut-être présenter leurs résultats de régression sous LaTeXet pour ce faire, il existe un programme réalisé par Antoine Terracol lorsqu'il était en thèse à Paris 1, `outtex` qui permet de présenter automatiquement les résultats d'une régression sous Stata en format LaTeX.

2.4.6 Les tests d'hypothèses

Stata calcule automatique un t-test (statistique de student) qui est présenté avec les résultats. Avec la commande `test`, il est pratique de pouvoir réaliser un F-test ou χ^2 des hypothèses sur les coefficients des variables explicatives, comme par exemple :

```
. reg salaire age edu experience csp if sexe==1 & pays="france" [aweight=pond]
. test age=0.4
. test edu=experience
```

2.5 Les autres types d'estimation

2.5.1 Estimation linéaire contrainte

Il est possible d'effectuer une régression en imposant une contrainte qui peut être originaire de la théorie économique ou d'une obligation logique. Dans l'exemple suivant, les deux contraintes n'ont aucun sens :

```
. constraint define 1 age=0.4
. constraint define 2 edu=experience
. cnsreg salaire age edu experience csp, constraint(1 2)
```

2.5.2 Estimation par variable instrumentale

On peut naturellement utiliser les commandes `regress` et `predict` pour reconstruire une estimation par variable instrumentale, mais Stata offre la commande `ivreg` qui fait directement l'estimation.

```
. ivreg salaire age edu experience (csp=revenu_parents)
```

2.5.3 Les autres estimations

L'estimation logit utilise la commande `logit`, probit la commande `probit`, les régressions par quantile la commande `qreg`, les régression tobit la commande `tobit`... facile à retrouver !

3 La programmation

3.1 Comment programmer

Si un do file peut être considéré comme un long programme, il est souvent nécessaire à l'intérieur d'un do file de réaliser un programme pour effectuer une tâche répétitive en économisant les lignes de programmes. La commande pour définir un programme est `program define nomprogramme` puis la suite des commandes à appliquer. Par exemple, vous souhaitez nettoyer les enquêtes emploi 1990 à 2000 pour obtenir un unique fichier cohérent.

```
. program define preparation
. keep if age >15
. keep salaire edu emploi age sexe indent ocu
. gen age2 = age^2
. gen exp = age - edu - 6
. end
```

Puis il vos suffit d'appliquer ce programme à chaque enquête :

```
. use ee1990
. preparation
. save 90, replace
. clear . use ee1991
. preparation
. save 91, replace
```

On peut aussi créer un programme avec en argument `‘1’` qui désigne la variable, la valeur ou la commande à laquelle le programme va s'appliquer. Par exemple, vous souhaitez faire un programme qui vous donne la valeur moyenne des variables de votre base de données :

```
. program define moyenne
. egen moyenne‘1’=mean(‘1’)
. display "Moyenne de ‘1’= " moyenne‘1’
. end
```

La commande `display`, comme son nom l'indique, donne le résultat, avec entre guillemet le texte voulu. Cela donnera par exemple :

```
. moyenne age
. Moyenne de age= 41.53
```

N'oubliez pas que si votre argument est du texte (si `‘1’` désigne du texte, alors il faut pensez à placer des guillemets : `"‘1’"`). Si vous voulez effacer un programme pour le redéfinir, il suffit d'utiliser la commande `drop`, précédé de la saisie du programme, c'est-à-dire `capture program` :

```
. capture program drop moyenne
```

Si vous rédigez un long programme qui doit laisser apparaître des résultats qui risquent de bloquer la poursuite automatique du do file, inscrivez en début de programme la commande `set more off` qui empêche l'arrêt du programme lorsque l'écran de résultat est rempli.

3.2 Macros

Les macros peuvent être local ou global selon que leur action se veut limitée au programme ou applicable à l'ensemble de la session avec les commandes `local` ou `global`.

3.3 Réaliser une boucle

3.3.1 La commande `for` ou `foreach` et `forvalues`

La commande `for` date des versions de Stata antérieures à Stata8. Pour cette dernière version, on utilise les commandes `foreach` et `forvalues`.

La commande `for` est très pratique pour effectuer des tâches répétitives. Dans l'exemple suivant, on souhaite créer des variables de moyenne pour différentes variables :

```
. for varlist age edu salaire : egen moyenneX=mean(X)
```

Si vous voulez faire le même type d'exercices mais avec plusieurs commandes, comme créer un un résultat texte à la suite des moyennes, il faut utiliser la commande `foreach... in`. Avec Stata8, la commande `foreach... in` est la seule à utiliser.

```
. foreach VAR of varlist age edu salaire {
. egen mean‘VAR’=mean(‘VAR’)
. display "moyenne de ‘VAR’=" mean‘VAR’
}
```

```
. foreach file in cefichier.dta cetautre.dta etceluila.dta {
. append using "’annexe’"
}
```

Si vous souhaitez utiliser `foreach` pour une boucle numérique, il ne faut pas utiliser `in` ou `of` mais `=`, comme dans l'exemple suivant :

```
. foreach x = 1/1000{
}
```

La commande `forvalues` permet aussi de répéter une action pour un éventail de valeurs. La syntaxe est la suivante :

```
. forvalues valeur = #1(#d)#2 {
commande à répéter pour les valeurs de #1 à #2 avec un interval de #d
}
```

Notez que `#1(1)#2` peut aussi s'écrire `#1/#2`. Ainsi, par exemple, pour créer 100 variables uniformes notées de x1 à x100 :

```
. forvalues i = 1/100 {
generate x‘i’= uniform()
}
```

3.3.2 La commande `while` et comment créer un incrément

La commande `while {... }` oblige le logiciel à répéter la commande entre accolade jusqu'à ce que la condition précisée par `while` ne soit plus vraie. Il faut d'abord lancer un compteur par le biais d'une macro locale, puis préciser la condition de la boucle :

```
. local i=1901
. while ‘i’<=2004 {
. display "année" ‘i’
. local i=‘i’+1
}
```

On obtiendra alors :

```
. année 1901
. année 1902
. année 1903...
```

3.4 Programmer en ramification (branching)

La programmation en ramification consiste simplement à conditionner une boule avec les commandes `if` et `else`.

3.5 Réaliser des simulations Monte Carlo

Pour réaliser des simulations Monte Carlo, il faut d'abord créer un programme pour une simulation, puis utiliser la commande `simulate` pour répliquer la simulation un certain nombre de fois.
Le programme doit avoir la forme suivante (pour créer par exemple la simulation d'une loi normale de moyenne mu et d'écart-type sigma) :

```
. program define loinorm, rclass
. syntax [, obs(integer 1) mu(real 0) sigma(real 1)]
. drop _all
. set obs `obs'
. tempvar z
. gen z = exp(`mu'+`sigma'*invnorm(uniform()))
. summarize `z'
. return scalar mean = r(mean)
. return scalar Var = r(Var)
. end
```

Dans ce programme, l'option `rclass` permet de créer des résulats accessibles par la commande `return`, `obs` définit le nombre d'observations de chaque échantillon, `tempvar` est la création d'une variable temporaire, uniquement pour le programme et syntax est une commande qui définit chaque argument du programme. Ensuite, il faut répliquer la simulation du programme le nombre de fois voulu :

```
. simulate "loinorm, obs(100)" mean=r(mean) var=r(Var), reps(10000)
```

Prenons un exemple simple. Nous voulons simuler l'estimateur MCO sur un modèle à deux variables dont les erreurs suivent une loi normale centrée :

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t \text{ avec } \epsilon_t \rightsquigarrow N(0, \sigma^2)$$

On définit d'abord le programme pour une simulation avec 100 observations en supposant que les vrais paramètres du processus de création des données (*data generating process*) sont $\beta_0 = 3$, $\beta_1 = 5$ et $\sigma = 2$. La variable `x` suit une loi uniforme sur [0,6).

```
. program define loinorm, rclass
. drop _all
. set obs 100
. gen e = invnorm(uniform())*2
. gen x = uniform()*6
. gen y = 3 + 5*x + e
. regress y x
. return scalar b0 = _coef[_cons]
. return scalar b1 = _coef[x]
. end
```

Ensuite on réplique cette simulation 1000 fois.

```
. simulate "loinorm" b0=r(b0) b1=r(b1), reps(1000)
```

Avec les commandes `summarize` et `histogram`, on peut vérifier que la moyenne de b0 est proche de 3 et que celle de b1 est proche de 5.