

**Session de formation sur Stata:
Banque Mondiale**

Nouakchott : ...Date

Animateur : Araar Abdelkrim

1 Introduction

1.1 Qu'est-ce que c'est que Stata?

Crée par [Statacorp](#), Stata est un logiciel statistique utilisé par plusieurs firmes et institutions académiques à travers le monde. La plupart des utilisateurs sont des chercheurs spécialisés dans les domaines de l'économie, de la sociologie, des sciences politiques et de l'épidémiologie.

Le logiciel Stata permet entre autres de faire:

- Le traitement des données;
- L'analyse statistique ;
- Les graphiques;
- Les simulations;
- La conception des programmes spécialisés ou complémentaires.

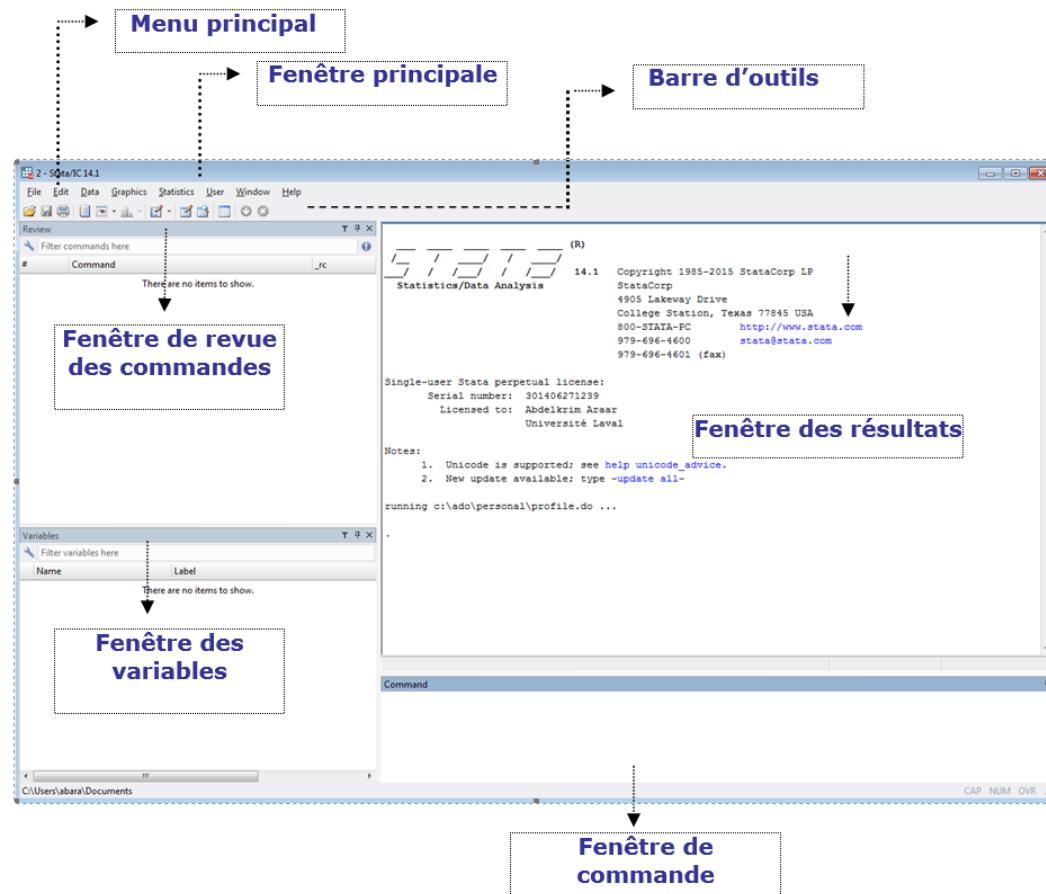
Stata est devenu de plus en plus populaire au cours des dernières années. Sa conception optimisée et son interface graphique d'utilisation simplifiée ont fait que les gens l'ont trouvé plus convivial relativement à d'autres logiciels statistiques.

1.2 Quel système d'exploitation est compatible avec Stata?

Le **système d'exploitation** (Operating System ou OS en anglais) est un ensemble de programmes responsables de la liaison entre les ressources matérielles d'un ordinateur et les applications de l'utilisateur. Le logiciel Stata peut fonctionner avec les systèmes d'exploitation Windows, Mac OS X ou Unix.

2 L'interface graphique d'utilisateur de Stata

Figure 1: L'interface graphique d'utilisateur de Stata 14.X



2.1 Se familiariser avec l'interface graphique de Stata

Les principales fenêtres de Stata sont:

A- La fenêtre de commande (*Command window*)

Les commandes sont soumises à Stata en ligne de commande (une seule ligne de commande à la fois).

- Haut de page (Page-Up) : éditer la commande précédente;
- Bas de page (Page-Down) : éditer la commande suivante;
- Tab : compléter le nom d'une variable.

B- La fenêtre de revue des commandes (*Review window*)

Cette fenêtre montre l'historique des commandes soumises

- Cliquer une seule fois sur la commande pour la copier dans la fenêtre de commande;
- Cliquer deux fois sur la commande pour l'exécuter;
- En cliquant sur le bouton gauche de la souris, un menu apparaît pour que l'utilisateur puisse copier ou sauvegarder les commandes dans un fichier *.do.

C- La fenêtre des variables (*Variables window*)

Cette fenêtre montre la liste des variables du fichier de données en mémoire et leurs étiquettes.

- Cliquer une seule fois sur la variable pour la copier dans la fenêtre de commande;
- Cliquer sur le bouton gauche de la souris pour ajouter une note ou changer la police (*font*) d'affichage dans cette fenêtre.

D- La fenêtre des résultats (*Results window*)

Cette fenêtre affiche les résultats après l'exécution des commandes soumises.

- Sélectionner une partie ou l'ensemble des résultats et cliquer par la suite sur le bouton gauche de la souris pour copier la partie sélectionnée (format texte ou format tableau).

Le menu principal comprend aussi des items qui permettent, entre autres, d'accéder aux différentes boîtes de dialogue proposées initialement par Stata.

2.2 Réorganiser l'interface graphique de l'utilisateur

L'utilisateur peut choisir entre deux formats de positionnement des différentes fenêtres de Stata :

- 1- Les différentes fenêtres sont situées dans la fenêtre principale. Cela fait aussi que leur positionnement va dépendre de la position de la fenêtre principale.

Menu Principal: Prefs → Manage preferences → Load preferences → Compact window settings

- 2- Le positionnement des différentes fenêtres est indépendant

Menu Principal: Prefs → Manage preferences → Load preferences → Maximized window settings

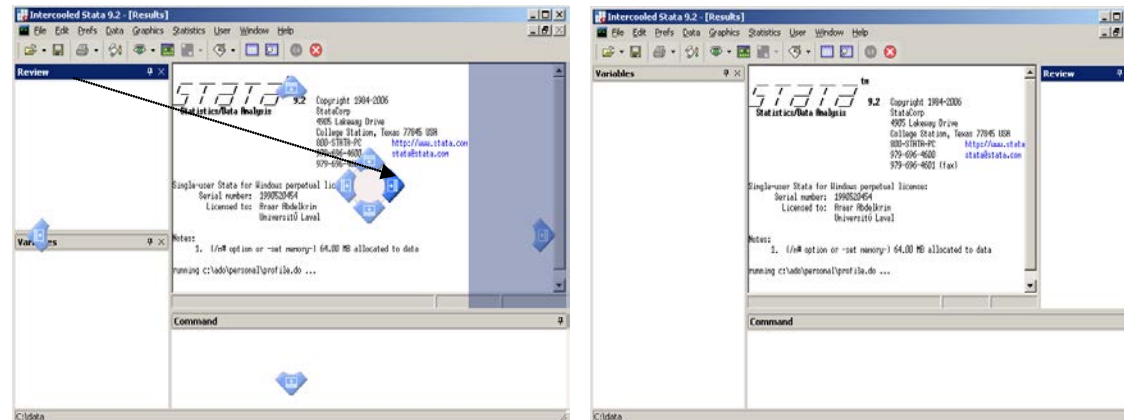
Pour chacun de ces deux formats de positionnement, l'utilisateur peut continuer de réorganiser le format d'affichage des fenêtres et sauvegarder par la suite les formats désirés. Pour le format **Compact window settings**, Stata offre un environnement d'organisation des fenêtres assez simple.

Exemple 1

Positionner la fenêtre **Review** relativement à gauche de la fenêtre **Results**.

Sélectionner la fenêtre **Review** et glisser cette fenêtre en maintenant le bouton droit de la souris enfoncé, comme l'indique le graphique qui suit.

Figure 2: Positionnement de la fenêtre Review relativement à gauche de la fenêtre Results

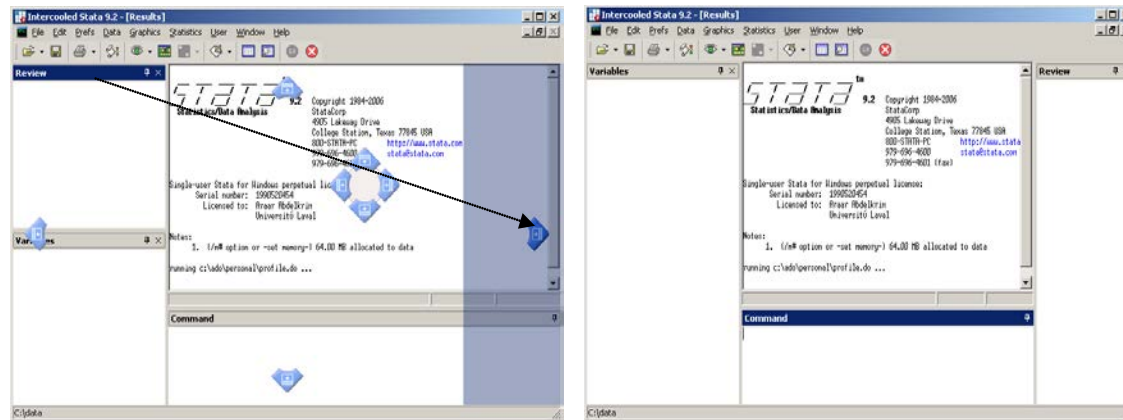


Exemple 2

Positionner la fenêtre **Review** relativement à gauche de la fenêtre principale de Stata

Sélectionner la fenêtre **Review** et glisser cette fenêtre en maintenant le bouton droit de la souris enfoncé, comme l'indique le graphique qui suit.

Figure 3: Positionnement de la fenêtre Review relativement à gauche de la fenêtre principale



2.3 Le menu principal et les boîtes de dialogue

Comme la plupart des logiciels, Stata contient dans son menu principal des items usuels tels que **File** qui permettent par exemple d'accéder aux sous-menus pour ouvrir ou sauvegarder des fichiers. Stata a misé dans ces dernières années sur l'interface graphique et plus précisément sur la conception des boîtes de dialogue. Ces derniers facilitent l'utilisation des différentes commandes –applications- que Stata offre. Principalement, Stata regroupe les différentes commandes officielles sous les trois items du menu principal, soit: **Data**, **Graphics** et **Statistics**.

Pour l'exécution des commandes, Stata offre trois possibilités:

1. Écrire la commande dans la fenêtre de commande et taper par la suite le bouton du clavier **Entrer (Enter)** ;
2. Exécuter un fichier .do (un fichier ASCII qui peut contenir plus qu'une commande) ;
3. Exécuter la commande à partir de sa boîte de dialogue.

Pour afficher la boîte de dialogue pour une commande ou application donnée, l'utilisateur peut choisir entre l'une des deux possibilités.

1. Sélectionner l'item de la commande à partir du menu principal.

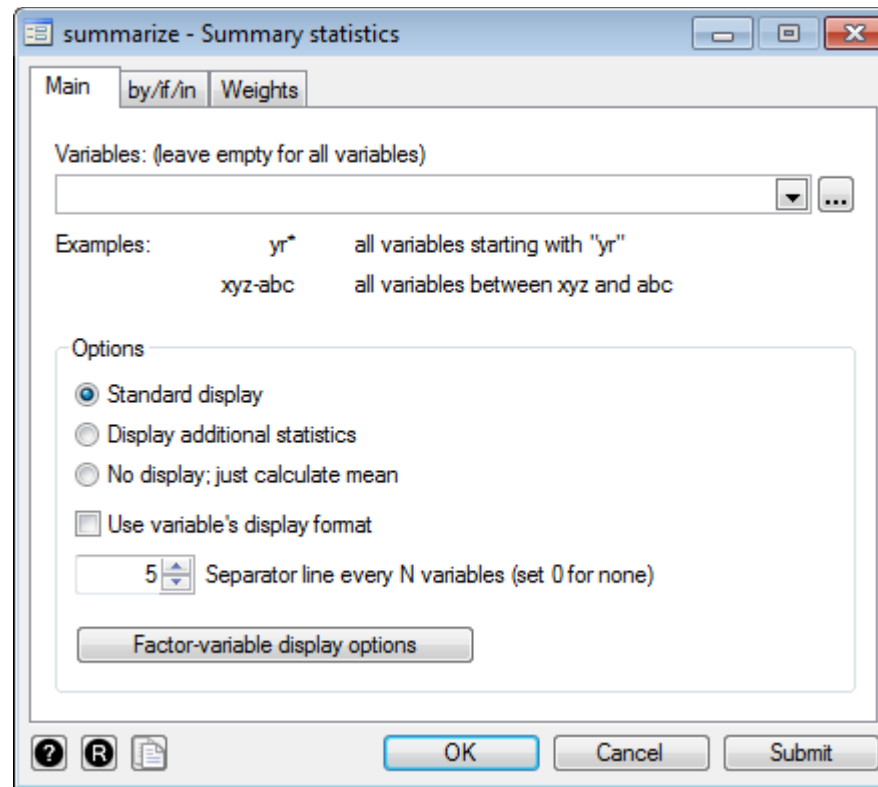
Exemple

Menu Principal: Statistics → Summaries... → Summary statistics → Summary statistics

2. Écrire **db** suivi du nom de la commande qu'on veut exécuter avec la boîte de dialogue dans la fenêtre de commande et taper le bouton du clavier **Entrer**.

Exemple
db summarize

Figure 4: Boîte de dialogue de la commande *Summarize*



On trouve les mêmes six boutons, qui apparaissent en bas de la boîte, dans toutes les boîtes de dialogue. La fonction de chacun d'eux est présentée dans le tableau suivant :



Afficher l'aide (**help**) de la commande avec le surfeur de Stata.



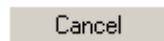
Effacer (**Reset**) : Initialiser les champs à leurs valeurs par défaut.



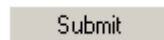
Copier la syntaxe qui sera générée en cliquant sur le bouton **OK**.



Lancer l'exécution de la commande et fermer la boîte de dialogue.



Fermer la boîte de dialogue sans lancer l'exécution de la commande.



Exécuter la commande sans fermer la boîte de dialogue. Ce bouton est utile lorsqu'on prévoit faire plus d'explorations, telle qu'exécuter la même commande avec d'autres variables ou options.

Remarque

En cliquant sur le bouton **Submit** ou **OK** la syntaxe de la commande est générée automatiquement et elle apparaît dans la fenêtre de commande.

Chacune des trois possibilités d'exécution des commandes a une utilité particulière :













1. L'utilisation des boîtes de dialogue permet de générer une syntaxe (ligne de commande) qui ne comporte pas d'erreurs lorsque le choix des options est adéquat. Cette procédure permet aussi d'apprendre rapidement la syntaxe des différentes commandes.
2. Comme nous allons le voir plus loin, le fichier .do peut contenir plus qu'une ligne de commande et il est réutilisable (il peut être exécuté chaque fois qu'on l'invoque). L'ensemble des lignes de commandes que ce fichier peut contenir représente ainsi un programme que l'utilisateur peut corriger ou ajuster rapidement au besoin.
3. Ceux qui sont bien familiarisés avec Stata peuvent écrire directement la ligne de commande dans la fenêtre de commande et l'exécuter en tapant sur le bouton **Entrer** du clavier. Cette procédure permet entre autres de tester rapidement la commande ou de faire plus d'explorations.

2.4 La barre d'outils

La barre d'outils de Stata contient des boutons qui permettent un accès rapide aux différents utilitaires couramment utilisés, tels que la visionneuse -ou surfeur- de Stata (**Viewer**), l'éditeur des fichiers .do et l'éditeur des données.

Figure 5: La barre d'outils de Stata

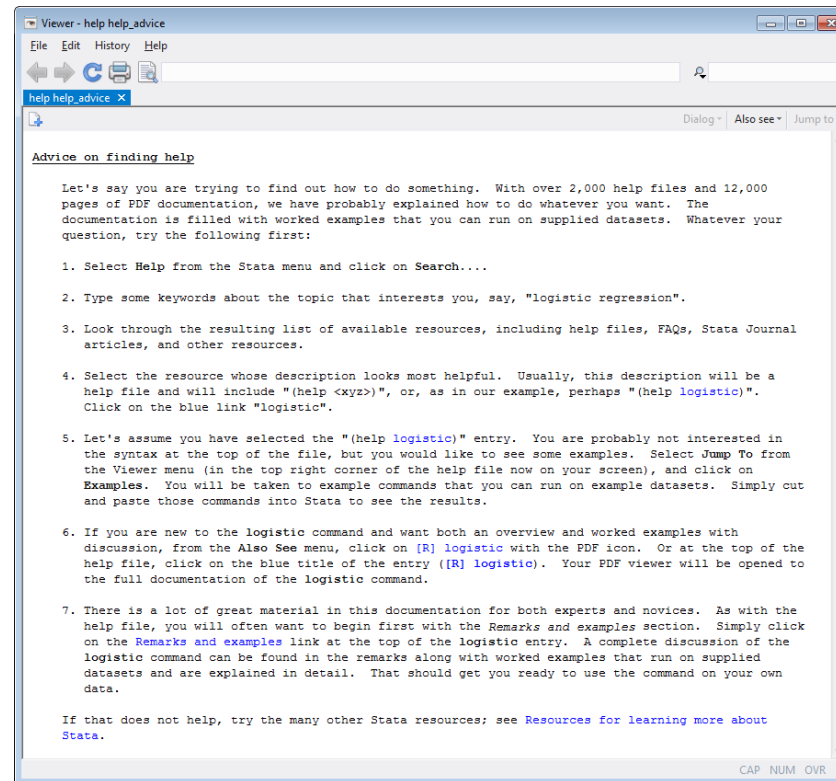


	Ouvrir un fichier de données de type Stata (extension *.dta)
	Sauvegarder le fichier de données actives - en mémoire-.
	Imprimer les résultats, le contenu de la visionneuse ou le graphique.
	Commencer, ajouter, suspendre ou fermer un fichier .log.
	Ouvrir le surfeur de Stata.
	Faire apparaître la fenêtre des graphiques
	Ouvrir un fichier .do avec son éditeur Stata ou faire apparaître l'éditeur d'un fichier .do donné.
	Éditer les données.
	Visualiser les données.
	Ouvrir la fenêtre de gestion des variables.
	Clear plus de conditions
	Arrêter l'exécution du programme en cours.

2.5 Utilisation du surfeur (Viewer)

La fenêtre du surfeur (**Viewer**) est conçue principalement pour l'affichage de l'aide (**help**) sur l'utilisation de Stata. Cette fenêtre permet aussi d'afficher les fichiers de format SMCL (Stata Markup and Control Language) ainsi que les fichiers texte (ASCII).

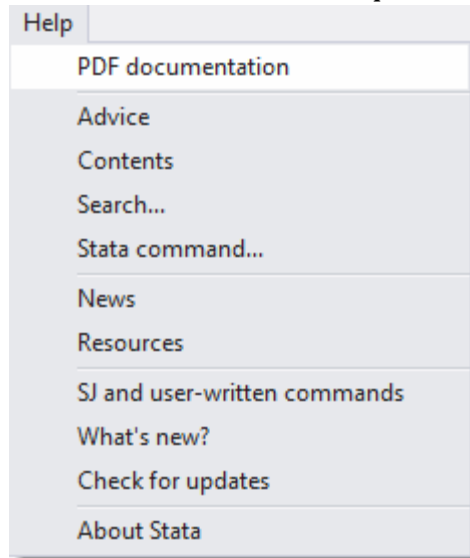
Figure 6: Surfeur de Stata (Stata Viewer)



Avec cette utilitaire, on peut :

- Voir les fichiers d'aide;
- Chercher la documentation de Stata;
- Chercher sur le net en utilisant des mots clés;
- Obtenir de l'aide sur l'utilisation de la commande search;
- Trouver et installer les modules publiés dans Stata Journal;
- Vérifier les mises à jour de Stata;
- Éditer les fichiers .log, .SMCL et les fichiers ASCII;
- Etc.

Dans les versions récentes telles que Stata 14., on peut accéder à plusieurs sous items à partir de l'item Help du menu principal.



PDF	Accéder au help sous sa forme la plus détaillée en format PDF.
Search	Lancer une recherche en utilisant des mots clés pour les fichiers d'aide, les FAQ (Questions fréquemment posées), les articles du Journal Stata, etc.
Contents	Afficher sous forme de tableau le contenu des fichiers d'aides de Stata classés par sujet.
News	Afficher les nouvelles qui concernent les utilisateurs de Stata.

2.6 Éditer ou visualiser des fichiers de données Stata

L'éditeur des données (Data Editor)

L'éditeur des données est un utilitaire permettant à la fois d'afficher, de saisir ou de modifier les données.

Pour ouvrir l'éditeur des données :

- cliquer sur le bouton de la barre d'outils **Data Editor**;
- ou taper la commande **edit**.

L'éditeur des données a la forme d'un tableur:

- Les colonnes sont des variables;
- Les lignes sont des observations.

On peut utiliser les commandes copier/coller pour copier à partir du tableur d'un autre logiciel, tel qu'Excel par exemple. On peut également modifier la valeur d'une cellule en la sélectionnant et en changeant son contenu.

Pour insérer les données d'une nouvelle variable:

- Se positionner dans la première cellule de la colonne;
- Entrer la valeur de la cellule et taper le bouton du clavier **Enter**;
- Entrer la valeur de la deuxième observation et répéter l'opération jusqu'à ce que toutes les valeurs de la colonne (variable) soient insérées.

D'une manière générale, nous avons deux types de données.

- Numérique (**numeric**) (Ex. 1 / 1.1)
- Alphanumérique, c.-à-d., chaîne de caractères (**string**) (Ex. Rurale). Ce type de données apparaît en couleur rouge dans l'éditeur des données.

Au lieu d'éditer l'ensemble des données du fichier en mémoire, l'utilisateur peut aussi éditer simplement quelques variables, ou encore quelques observations, comme il est indiqué dans ce qui suit.

- edit** varname Pour afficher une seule variable (ex. **edit** var1)
- edit** varlist Pour afficher une liste de variables (ex. **edit** var1 var3)
- edit** in range Pour afficher les observations comprises dans un intervalle
- 1 : afficher uniquement la première observation
 - 1/10 : afficher les dix premières observations
 - 3/-3 : afficher toutes les observations excepté les deux premières et les deux dernières.
- edit** if exp afficher les observations qui respectent la condition exprimée par l'expression (**exp**).

On peut combiner les différentes options :

Exemple

```
use data\data1.dta, replace
edit revenu age_cm in 1/20 if age_cm>30
```

L'éditeur des données contient les onglets suivants :

Figure 7: L'éditeur des données

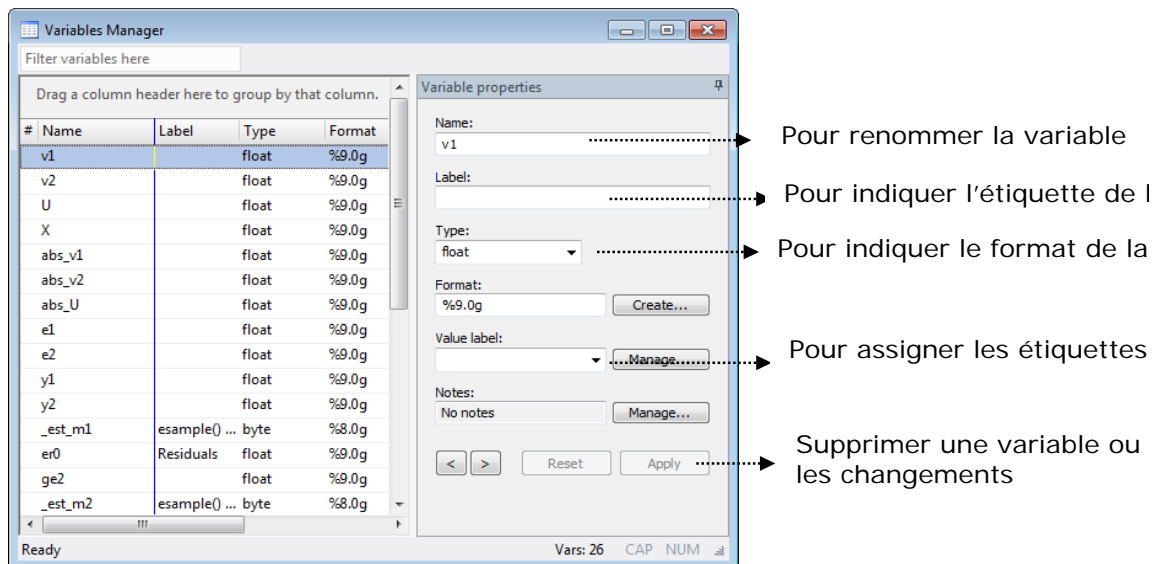
The screenshot displays the SPSS Data Editor window for an untitled dataset. The main data grid shows 24 rows of data with 10 columns: v1, v2, U, X, abs_v1, abs_v2, abs_U, e1, e2, and y1. The 'abs_U' column is highlighted in yellow, and its value for the first row is .21621647. The right-hand side of the window contains two panels: 'Variables' and 'Properties'. The 'Variables' panel lists all 26 variables, with checkboxes for each. The 'Properties' panel shows the details for the selected variable 'abs_U', including its name, label, type (float), format (%9.0g), and value label. The bottom status bar indicates 26 variables, 10,000 observations, and the current mode is 'Edit'.

	v1	v2	U	X	abs_v1	abs_v2	abs_U	e1	e2	y1
1	-13.97497	-1.972451	.4833452	-.6806995	1.394346	.9877538	.2162165	-13.24995	-1.4	
2	-3.027108	3.359333	1.57643	-.1745043	.306728	1.683361	.7861079	-.6624627	4.9	
3	-14.36568	1.247126	1.290127	.6345091	1.433161	.6224109	.636954	-12.43049	2.5	
4	-3.018715	1.677101	2.376935	1.866459	.3058941	.8419705	1.2127	.5466877	4.0	
5	.3764808	1.710257	-4.241498	-.4872429	.0314024	.8599013	2.113538	-5.985765	-2.	
6	19.08398	-.0286255	-.3248757	-2.105773	1.889903	.0024055	.1563442	18.59667	-.35	
7	2.972354	-1.798668	.751009	-1.111867	.28929	.8941102	.3690112	4.098868	-1.0	
8	-10.07241	-.943862	5.128314	1.08438	1.006645	.4717414	2.561926	-2.379937	4.1	
9	-23.30559	-.8974158	-.1116366	.1512708	2.321298	.4537604	.0824327	-23.47304	-1.0	
10	-15.82528	-.3484161	-1.854283	.9029323	1.578165	.1762025	.9293357	-18.6067	-2.2	
11	-8.293807	-1.455059	2.521423	-.8752381	.8299493	.7267188	1.238028	-4.511672	1.0	
12	6.356232	.2493314	.8760456	.6179175	.6254621	.1315695	.4608985	7.6703	1.1	
13	-.864582	-2.664968	-.8530619	.2191682	.0918913	1.328925	.4181746	-2.144175	-3.5	
14	8.616189	2.180202	1.268834	-.21151	.849978	1.09822	.6492514	10.51944	3.4	
15	7.230558	1.451188	-.932866	.418883	.7123222	.7330428	.4417724	5.831259	.51	
16	.2781706	.3895741	-3.728462	-.6945893	.0216357	.1993028	1.861263	-5.314522	-3.3	
17	-21.12651	.7595802	.0892136	-1.165766	2.104817	.3758754	.0033189	-20.99269	.84	
18	-20.22146	-3.152071	-1.332248	-.2474934	2.014905	1.580237	.6924714	-22.21984	-4.4	
19	-5.843442	-1.869721	.7516264	.3921653	.5865173	.9331459	.376884	-4.716002	-1.1	
20	10.18948	.4099524	.3335484	.7876012	1.006277	.2134273	.1986053	10.6898	.74	
21	.1531591	4.227971	-1.686692	1.813151	.0092164	2.119085	.8079237	-2.376879	2.5	
22	-24.94966	2.537563	-.8920488	1.315005	2.484629	1.263644	.4566955	-26.28773	1.6	
23	14.35214	.4622511	-1.690055	1.719955	1.419818	.2412366	.7909563	11.81706	-1.2	
24	-2.808454	-3.1727	.6524133	-1.115316	.2850057	1.583646	.310748	-1.829834	-2.5	

Gestionnaire des variables

On peut modifier le nom, le format et l'étiquette de la variable en utilisant le gestionnaire des variables.

Figure 8: Fenêtre pour la mise à jour des propriétés d'une variable



La visionneuse des données (Data Browser)

L'utilitaire visionneuse des données (**Data Browser**) ressemble dans son format à l'éditeur des données. Cependant, cet utilitaire ne sert qu'à visualiser les données sans qu'on puisse faire aucune modification sur celles-ci.

Pour afficher la visionneuse des données :

- cliquer sur le bouton de la barre d'outils **Data Browser**;
- ou taper la commande **browse**.

Comme pour l'éditeur des données, on peut aussi avec la visionneuse des données restreindre l'information qui sera affichée.

Exemple

<u>u</u>se	data\data1.dta, replace
<u>b</u>rowse	revenu age_cm if age_cm>40

Bonnes pratiques

- 👉 Si vous voulez simplement explorer les données sans faire de modifications, privilégiez plutôt l'utilisation de la visionneuse des données (**Data Browser**) pour éviter une modification involontaire des données.

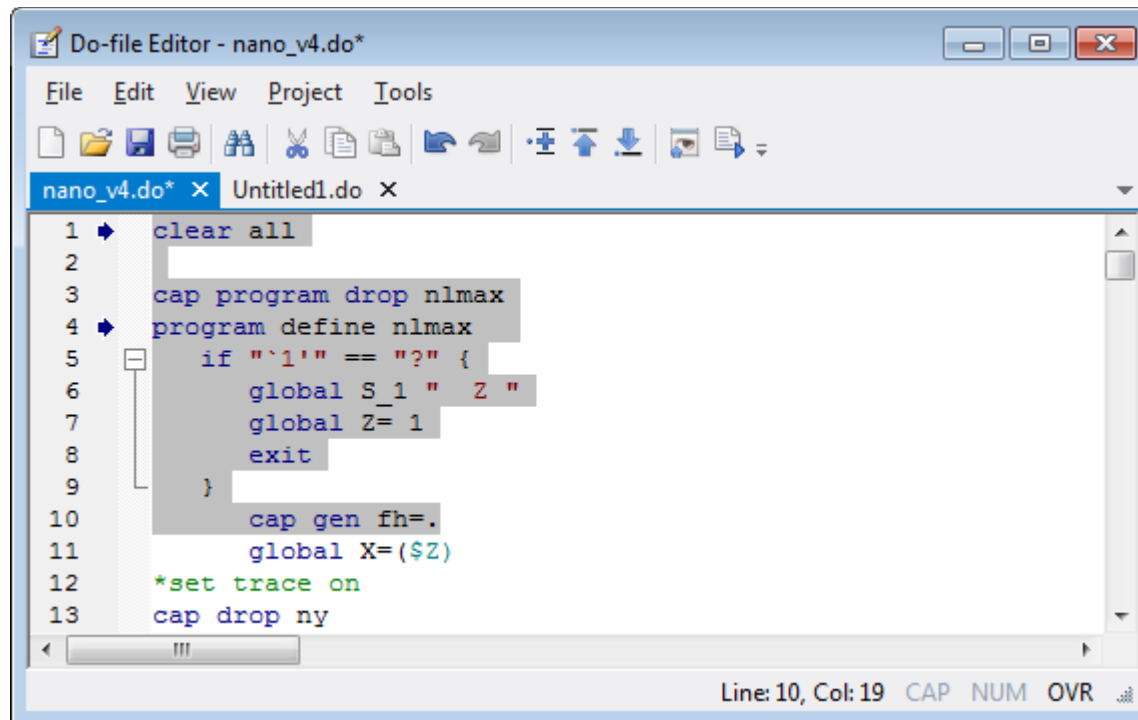
Astuces

- 💡 Par défaut, pour une variable dont les valeurs sont étiquetées (Ex. 1 : zone rurale / 2 : zone urbaine) c'est bien les étiquettes des variables qui seront affichées dans l'éditeur ou la visionneuse des données. Si vous ne souhaitez pas faire apparaître les étiquettes des variables, vous devez ajouter l'option nolabel à la commande **edit** (Ex. **edit** revenu zone, nolabel).
- 💡 Alors que le contenu alphanumérique (string) s'affiche en rouge, les étiquettes (labels) s'affichent en bleu.

2.7 L'éditeur des fichiers .do

L'éditeur des fichiers .do sert à écrire, éditer ou même exécuter une partie ou la totalité du contenu d'un fichier .do.

Figure 9: L'utilitaire éditeur des fichiers do de Stata


















The screenshot shows the 'Do-file Editor - nano_v4.do*' window in Stata. The window has a menu bar (File, Edit, View, Project, Tools) and a toolbar with icons for file operations. Two tabs are open: 'nano_v4.do*' and 'Untitled1.do'. The main text area contains the following Stata code:

```
1 clear all
2
3 cap program drop nlmax
4 program define nlmax
5     if "`1'" == "?" {
6         global S_1 " 2 "
7         global Z= 1
8         exit
9     }
10    cap gen fh=.
11    global X=($Z)
12    *set trace on
13    cap drop ny
```

The status bar at the bottom right indicates 'Line: 10, Col: 19' and shows the 'CAP NUM OVR' indicators.

Les boutons de l'éditeur des fichiers .do ont les fonctions suivantes :

	Lancer un nouvel éditeur .do.
	Ouvrir un fichier .do.
	Sauvegarder le fichier .do courant.
	Imprimer le fichier .do.
	Chercher un mot dans le fichier .do.
	Couper la partie sélectionnée et la couper du fichier courant.
	Copier la partie sélectionnée dans le fichier courant.
	Coller le texte.
	Annuler le dernier changement.
	Accepter le changement succédant
	Marquer la ligne
	Aller à la ligne marquée précédente
	Aller à la ligne marquée succédente
	Éditer le fichier .do courant dans la visionneuse (le fichier .do doit être préalablement sauvegardé).
	Exécuter les lignes de commandes dans le fichier .do (ou les lignes de commandes sélectionnées).

Les fichiers do sont simplement des fichiers de type texte qui ont l'extension .do et qui contiennent un ensemble de ligne de commande Stata.

Exécuter un fichier do à partir de la fenêtre de commande

Syntaxe

{do|run} filename [arguments] [, nostop]


La commande **do** ou **run** exécute les lignes de commande sauvegardées dans le fichier **filename**. La commande **run** exécute le fichier sans afficher de résultats. L'option **nostop** force l'exécution de toutes les commandes même si quelques lignes de commandes contiennent des erreurs.

Insérer des commentaires dans le fichier do

/// Commentaire Le commentaire est écrit dans une seule ligne.

/* Commentaire */ Le commentaire débute après /* et finie à */. On peut écrire le commentaire dans plusieurs lignes.

Astuces

 Pour bloquer l'exécution d'une partie du programme do, on peut mettre la partie qu'on ne souhaite pas exécuter comme commentaire.

Commandes couramment évoquées à l'entête du fichier do sont :

#delimit Pour indiquer le délimiteur des lignes de commande.

clear	Pour libérer la mémoire
capture log close	Pour fermer le fichier log s'il est déjà ouvert
log using	Pour ouvrir un fichier log
set more off	Pour suspendre les pauses Stata pour l'affichage des résultats
set mem	Pour augmenter la mémoire allouée à Stata.
log off/on	Suspendre ou reprendre l'écriture dans le fichier log
save	sauvegarder le fichier Stata
log close	Fermer le fichier log

Les principales commandes pouvant être exécutées dans la fenêtre des commandes ou à travers un fichier .do sont exposées dans la section 3. Toutefois, nous présentons dans ce qui suit l'utilité des fichiers .log.

2.8 Utiliser un fichier log pour sauvegarder ou mettre à jour les résultats

La commande **log** permet de copier le contenu d'une session Stata (lignes de commande et les résultats) dans un fichier de type texte ou SMCL. On peut aussi ouvrir plus qu'un fichier log au même temps. La commande **cmdlog** est similaire à celle de **log** mais avec cette commande on ne copie que les lignes de commande dans le fichier log. La syntaxe pour ouvrir ou créer un fichier log est :

```
log using filename [, append replace [text|smcl] name(logname)]
```

Options

filename	Pour spécifier le nom du fichier log (avec arborescence).
append	Pour ajouter au contenu du fichier log
replace	Remplacer le fichier log
[text smcl]	Indiquer le format du fichier log
name()	Indiquer le nom du fichier log

Exemples

```
log using c:\results\res1, replace
```

Permet de sauvegarder les commandes Stata à exécuter dans le fichier res1.smcl sous le répertoire c:\results.

```
log using c:\results\log1, name(log1) text replace  
log using c:\results\log2, name(log2) smcl replace
```

Fermer, suspendre ou relancer l'écriture dans le fichier log

log {close|off|on} [logname]

close	Fermer le fichier log.
off	Suspendre temporairement l'écriture dans le fichier log.
on	Relancer l'écriture dans le fichier log.

Exemples

log close
log off log1

Afficher le statut du fichier log

log
log query [logname]

Exemples

log
log query log1

3 La syntaxe des commandes Stata

3.1 La forme générale de la syntaxe des commandes Stata

D'une manière générale, la syntaxe d'une ligne de commande Stata prend la forme suivante:

[prefix :] **command** [varlist] [=exp] [if] [in] [weight] [using filename] [, options]

Les éléments	La description
--------------	----------------

<i>Prefix :</i>	Commande préfixe qui précède la commande principale (<i>prefix command</i>).
-----------------	--

Exemple

by zone : sum v1

<i>command</i>	Commande de Stata (Stata command).
----------------	------------------------------------

Exemple

list

Remarque

La partie soulignée d'une commande Stata correspond à son abréviation reconnue par Stata.

Exemple

list ou **l**

describe ou **d**

generate ou **gen**

<i>varname</i>	Nom d'une seule variable.
----------------	---------------------------

<i>varlist</i>	<p>Une liste de noms de variables.</p> <p><i>Exemple</i> ...var1 var2 var3 var4...</p> <p>Lorsque var1 à var4 sont ordonnés - var1 var2 var3 var4 -, on peut écrire simplement :</p> <p>...var1-var4...</p>
<i>=exp</i>	<p>Expression algébrique.</p> <p><i>Exemple</i> gen xvar=var1+var2</p>
<i>if</i>	<p>Sert à indiquer une condition exprimée par une expression algébrique.</p> <p><i>Exemple</i> ...if rural==1...</p>
<i>in</i>	<p>sert à indiquer un intervalle d'observations.</p> <p><i>Exemple</i> ...in 1/10...</p>
<i>weight</i>	<p>Pour indiquer le poids.</p> <p><i>Exemple</i> ...[pweight=wvar]...</p>
<i>using filename</i>	<p>Pour indiquer le nom d'un fichier.</p>

Exemple

...using(data1)...

options

Pour indiquer des options de la commande.

Exemple

..., nolabel...

Les lignes de commande Stata ont une forme unifiée. Le diagramme présenté plus haut décrit la structure de base d'une ligne de commande, les options de la commande et l'abréviation minimale de chaque item. Les crochets [] sont utilisés dans ce diagramme pour distinguer les options des items requis.

Exemple

```
use                data\burkina94I.dta, clear  
by zone, sort :    summarize depal [aweight = tpond]
```

3.2 Commandes Stata de base

Les commandes système de base

dir	Lister les fichiers situés dans le répertoire de travail.
copy	Copier un fichier à partir d'un disque ou d'une URL (localisateur uniforme de ressource).
erase	Supprimer un fichier.
cd	Afficher l'arborescence du répertoire de travail courant (équivalent à la commande pwd).
mkdir	Créer un répertoire.

■ Présentation sommaire de certaines commandes Stata de base

version	Afficher la version installée de Stata.
update	<ul style="list-style-type: none">• Afficher les versions du fichier exécutable et des fichiers *.ado.• Permet aussi de faire la mise à jour des fichiers Stata. <p><i>Exemple</i> update from c:/temp update all</p>
which	Afficher la version d'un fichier Stata (*.ado, *.hlp ou autre). <p><i>Exemple</i> which svydes.ado</p>
query	Afficher les paramètres du système Stata.
memory	Afficher des informations sur l'utilisation courante de la mémoire allouée à Stata.
set memory	Permet de changer la mémoire allouée à Stata. <p><i>Exemples</i> set memory 20m set memory 60m, permanently</p>

clear

libérer la partie de la mémoire qui contient la base des données courante.

Remarque

clear est équivalente dans la version 9.2 de Stata à:

drop _all
label drop _all
scalar drop _all
matrix drop _all
cluster drop _all
eq drop _all
constraint drop _all
postutil clear
_return drop _all
discard
mata: mata clear

more

faire une pause dans l'exécution jusqu'à avoir tapé une touche sur le clavier.

Remarques

1. La ligne de commande « set more off » fait que le programme s'exécute sans aucune pose pour l'affichage des résultats. Elle désactive aussi l'effet de la commande **more**.
2. la ligne de commande « set more on » active à nouveau l'effet de la commande **more**.

#delimit

Sert à indiquer le caractère marquant la fin de la ligne de commande.

Remarque

Stata offre deux possibilités pour marquer la fin d'une ligne de commande et qui sont :

1. **#delimit** cr : La fin de commande est marquée par le caractère retour de chariot.
2. **#delimit** ; : La fin de commande est marquée par le caractère point-virgule.

count

Afficher le nombre d'observations.

Remarque

la ligne de commande « **count** if exp » affiche le nombre d'observations qui respectent la condition exp.

Exemple

count if age>=10

set obs #

Initialiser ou augmenter le nombre d'observations.

Exemple

clear

set obs 100

quietly

Exécuter une commande sans afficher les résultats.

Exemple 1

quietly sum age

Exemple 2

```
qui {  
    lignes de commandes Stata  
}
```

notes

Permet d'ajouter des notes au fichier de données dans la mémoire. Ces notes deviennent une partie de l'ensemble de données et se sauvegardent avec la sauvegarde du fichier des données.

Exemples

```
/* afficher les notes */
```

notes

```
/* Ajouter une note au fichier des données */
```

notes : Enquête sur les dépenses des ménages (2002).

```
/* Ajouter une note à la variable equi */
```

notes equi : Nombre d'adultes + 0.5 * nombre d'enfants.

```
/* Supprimer les notes sur le fichier des données */
```

note drop _dta equi

```
/* Supprimer les notes sur la variable equi */
```

note drop equi

list

Lister les variables dans la fenêtre des résultats

Exemples

```
/* lister toutes les variables */
```

list



```
/* lister les variables var1, var2 et var3 */
```

```
l var1 var2 var3
```

```
/* lister les observations 1 à 10 */
```

```
list in 1/10
```

Astuces

-  Pour utiliser la virgule dans l'affichage comme séparateur de décimales, taper la ligne de commande, « **set dp comma** ».
-  Pour utiliser le point comme séparateur de décimales, taper la ligne de commande, « **set dp period** ».

3.3 Opérateurs arithmétiques, logiques et relationnels

Le tableau suivant résume les principaux opérateurs reconnus par Stata.

Arithmétique	Logique	Relationnel
+ addition	~ Non	> plus grand
- subtraction	! Non	< plus petit
* multiplication	ou	>= plus grand ou égale
/ division	& et	<= plus petit ou égale
^ à la puissance		= égale
		~= différent
+ concaténation de texte (string)		!= différent

Remarques

- Pour écrire une expression qui contient une condition d'égalité, il faut utiliser le « == » au lieu du caractère « = ».
- Les valeurs manquantes (symbolisées par un « . » dans Stata) sont considérées les plus grandes valeurs. Ainsi, l'expression `taille > 6` est vraie si la valeur de la variable `taille` est strictement supérieure à 6 ou manquante. Pour ne conserver que les valeurs supérieures à 6 et non manquantes, il faut énoncer : `taille > 6 & taille != .`
- Les opérateurs arithmétiques suivent l'ordre de priorité habituel. L'opérateur « ^ » est prioritaire sur l'opérateur « + ».

3.4 Les fonctions mathématiques

Stata possède plusieurs fonctions prédéfinies permettant d'exécuter plusieurs opérations mathématiques à partir des variables en cours d'utilisation. Le tableau suivant présente une sélection des fonctions mathématiques les plus utilisées.

Fonction	Description
abs(x)	génère la valeur absolue de la variable x
ceil(x)	produit la valeur entière par excès de la variable x, soit n tel que $n-1 < x \leq n$. Voir aussi int(x), floor(x), et round(x).
exp(x)	retourne l'exponentielle de x. Voir aussi ln(x), log(x), et log10(x).
floor(x)	produit la valeur entière par défaut de la variable x, soit n tel que $n \leq x < n+1$. Voir aussi int(x), ceil(x), et round(x).
int(x)	permet d'avoir la valeur entière de x en tronquant vers 0. Exp. int(5.2) = 5 et int(-5.2) = -5. La fonction trunc(x) produit le même résultat.
ln(x)	génère le logarithme de pour les valeurs de x strictement positives. La fonction log(x) produit le même résultat.
max(x1, x2, , xn)	retourne le maximum de x1, x2, , xn. Voir aussi min(x1, x2, , xn).
min(x1, x2, , xn)	retourne le minimum de x1, x2, , xn. Voir aussi max(x1, x2, , xn).
round(x)	retourne la valeur entière la plus proche de la valeur de x.
sign(x)	retourne -1 si x est négatif, 0 si x est nul et 1 si x est positif.
sqrt	retourne la racine carrée de x pour lorsque x est positive ou nulle.
sum(x)	retourne la somme cumulative de x.

3.5 Les qualificateurs : **by**, **if** et **in**

Ces qualificateurs peuvent être utilisés avec la plupart des commandes Stata. Elles peuvent également être combinées les unes aux autres.

Le qualificateur **by** permet d'appliquer la commande spécifiée à chaque valeur d'une variable. La syntaxe de cette expression est :

by varlist : **commande** varlist

Il est nécessaire que la base de données soit classée par cette variable. Cela peut se faire de deux façons. La première consiste à classer les données par cette variable en utilisant la commande **sort** (voir plus bas) puis exécuter le qualificateur **by** comme le montre l'exemple suivant :

```
sort education  
by education : summarize revenu
```

La deuxième méthode consiste à classer les données et exécuter la commande **by** en même temps. Ceci est possible avec la commande **bysort** :

```
bysort education : summarize revenu
```

Avec le qualificateur **if**, la commande spécifiée ne s'applique qu'aux observations respectant la condition écrite après **if**. La syntaxe de cette expression est :

commande varlist **if** condition

À titre d'exemple, la ligne de commande suivante :

```
summarize revenu if education == 13
```

permet d'obtenir des statistiques descriptives de la variable **revenu** pour les observations dont la variable **education** est égale à 13 (remarquer le double signe égal, il s'agit bien d'une condition).

Le qualificateur **in** permet d'appliquer la commande aux observations qui se situent dans un intervalle indiqué par l'expression qui suit **in**. La syntaxe de cette expression est :

commande varlist **in** condition

Exemple

summarize revenu **in** 100/200

Cette ligne de commandes permet de produire les statistiques descriptives de la variable revenu lorsque les observations retenues sont situées entre la 100^{ème} à la 200^{ème} ligne de la base de données.

3.6 Pondération des observations : weight

La plupart des commandes de Stata peuvent être exécutées en utilisant les poids (*weight*). Stata permet quatre genres de poids:

- fweights (poids de fréquence) indique la fréquence de l'observation.
- pweights (poids d'échantillonnage) est égale à l'inverse de la probabilité que l'observation soit sélectionnée.
- aweights (poids analytique), sont des poids qui sont inversement proportionnels à variance d'une observation ; c.-à-d., on suppose que la variance de la j -ième observation est σ^2/w_j , où le w_j sont les poids.
- iweights (poids d'importance) les iweights n'ont aucune définition statistique formelle; n'importe quelle commande qui soutient des iweights définira exactement comment ils sont traités. Dans la plupart des cas, ils sont prévus à l'usage des programmeurs qu'ils veulent produire un certain calcul.

Remarques

1. D'une manière générale, lorsque la commande offre la possibilité de choisir entre les trois formes de poids, les statistiques calculées sont les mêmes, mais leurs écarts types peuvent différer.
2. Pour calculer correctement l'écart type de quelques statistiques descriptives telles que la moyenne et lorsqu'on utilise des enquêtes auprès des ménages, il faut initialiser préalablement la structure d'échantillonnage (commande **svyset**) et utiliser des commandes appropriées (voir l'aide de la commande **svy**).

4 Stata et les bases de données

Stata ne peut gérer qu'une seule base de données à la fois. Avant d'en ouvrir une nouvelle, il faut préalablement fermer celle qui est actuellement ouverte en tapant la commande **clear**.

4.1 Ouvrir des fichiers de données

Pour importer des données, Stata offre plusieurs possibilités en fonction de la nature des données disponibles.

Ouvrir des fichiers de données Stata : La commande use

Si l'utilisateur dispose déjà d'une base de données en format Stata (c.-à-d. un fichier avec une extension .dta), la commande **use** permet d'ouvrir le fichier Stata. La syntaxe de cette commande est :

```
use data\nom_du_fichier [, clear nolabel]
```

Noter que :

- Si le nom du fichier contient des espaces, ajouter des doubles guillemets au nom : (Ex. "data 1").
- Le nom peut contenir l'emplacement du fichier (Ex. "c:/for1/data/data 1").

Options

clear	ouvrir le fichier même si le fichier en mémoire n'est pas encore sauvegardé après des modifications.
nolabel	ouvrir le fichier sans charger les étiquettes des valeurs ou celles des variables.

Bonnes pratiques

- 👉 Ajouter l'option clear pour s'assurer que le fichier Stata s'ouvre.
- 👉 Augmenter préalablement la mémoire allouée à Stata si vous prévoyez ouvrir des fichiers de grandes tailles.

Insertion manuelle des données : La commande **input**

La commande **input** permet d'insérer directement des données à partir de la fenêtre de commande. Cette commande est utile lorsqu'il n'y a que quelques données à rentrer. La syntaxe de cette commande est :

input [varlist] [, automatic label]

Exemple

```
input taille poids revenu  
      5 120 2000  
      7 180 1500  
      3 100 3000  
      2 200 4000  
      4 140 2500  
  
end
```

Remarque

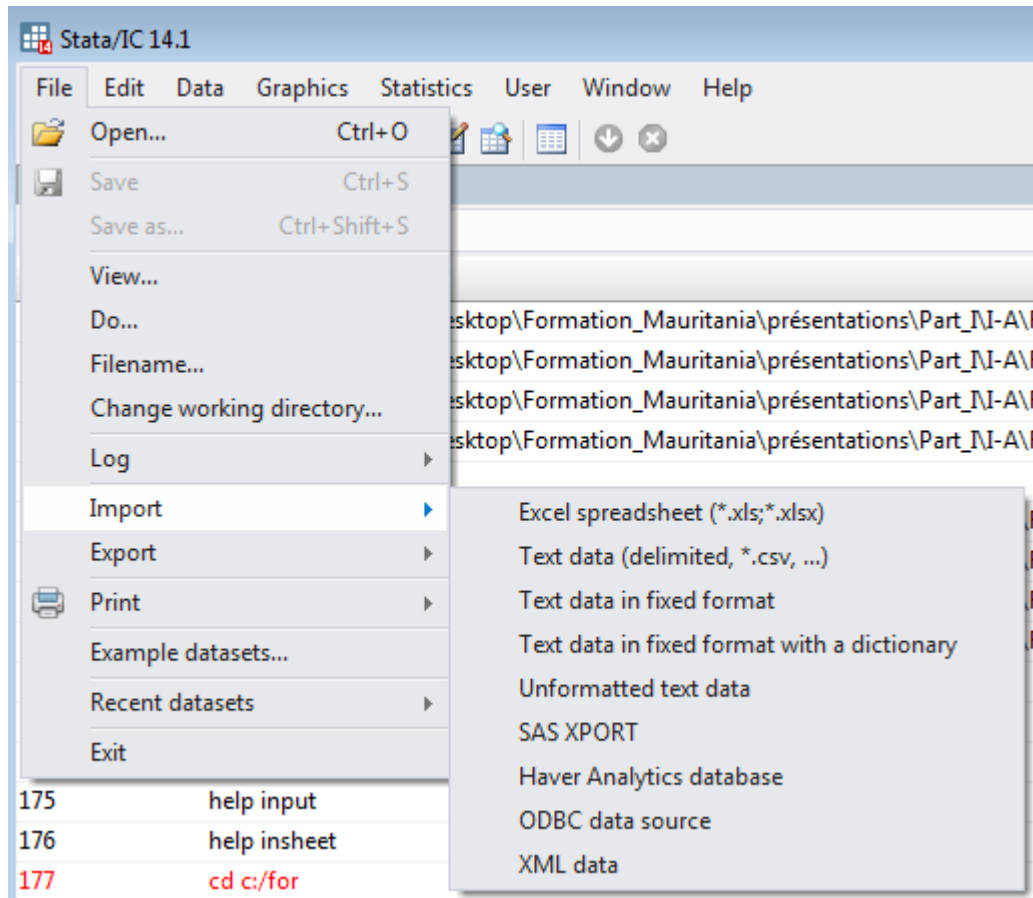
Il est également possible de saisir directement les données dans le **Data Editor (edit)**, comme on le fait avec le tableur Excel, voir section 2.6.1.

Automatic Stata crée des étiquettes de valeur à partir des données non numériques qu'il rencontre.

label permet de taper les étiquettes (chaînes) au lieu des valeurs numériques des variables associées.

Importer des données d'un autre format

Il existe plusieurs commandes avec plusieurs options permettant de charger des fichiers de différents formats, comme par exemple du texte (ASCII) formaté ou non. Parmi ces commandes, on trouve **import**, **insheet**, **infile** et **infix**.



4.1.3.1 La commande import

Cette commande permet de lire un fichier de données qui n'a pas le format Stata. Elle est prévue pour des données créées à partir d'autres logiciels conçus pour l'analyse des données. Le fichier de données doit être organisé avec une ligne par observation, les variables en colonne

- **import delimited**

Options

<code>delimiters("chars"[, collapseasString])</code>	Utiliser chars comme délimiteurs
<code>rowrange([start][:end])</code>	Rangée de données à charger
<code>colrange([start][:end])</code>	Colonne de données à charger
<code>varnames(#nonames)</code>	Traiter la rangée de données sous forme de noms de variables ou les données n'ont pas de variable
<code>case(preservelowerupper)</code>	Conserver le cas ou lire les noms des variables en minuscules (par défaut) ou
<code>asdouble</code>	Importer toutes les données à virgule flottante en double
<code>asfloat</code>	Importer toutes les données à virgule flottante en tant que flottants
<code>clear</code>	Remplacer les données en mémoire
<code>bindquotes(loosestrictnbind)</code>	Spécifier comment gérer les guillemets doubles dans les données
<code>stripquotes(yesnodefault)</code>	Supprimer ou conserver les guillemets dans les données
<code>numericcols(numlist_all)</code>	Force les colonnes spécifiées à être numériques
<code>stringcols(numlist_all)</code>	Force les colonnes spécifiées à être une chaîne
<code>encoding("encoding")</code>	Spécifiez l'encodage du fichier texte en cours d'importation

Exemple

Le contenu du fichier texte « fichier_1.raw » est:

===== Contenu du fichier_1.raw =====

Revenu ,age ,zone

1000 ,34 ,1

3200 ,39 ,2

1700 ,40 ,1

2700 ,54 ,2

=====

La ligne de commande appropriée pour lire ce fichier sera :

import delimited C:\for1\data\fichier_1.raw, delimiter(comma)

Si l'une des conditions énoncées plus haut n'est pas respectée, le recours à la commande **infile** s'impose.

4.1.3.2 La commande infile

Elle est similaire à la commande **insheet** mais elle est moins restrictive sur le formatage de l'information du fichier **ASCII** (par défaut, .raw est l'extension du fichier ASCII). La syntaxe de cette commande est :

infile varlist using filename [if] [in] [, options]

Options

automatic	donne des valeurs de label pour les données non numériques.
byvariable(#)	insère le fichier par variable; # est le nombre de valeurs à lire en successif pour chaque variable.
clear	Pour remplacer les données en mémoire.

Deux cas se présentent. Le premier concerne l'ouverture de données à format non fixe, mais avec un séparateur entre les variables qui est connu (espace, de tabulation ou de virgule, etc.).

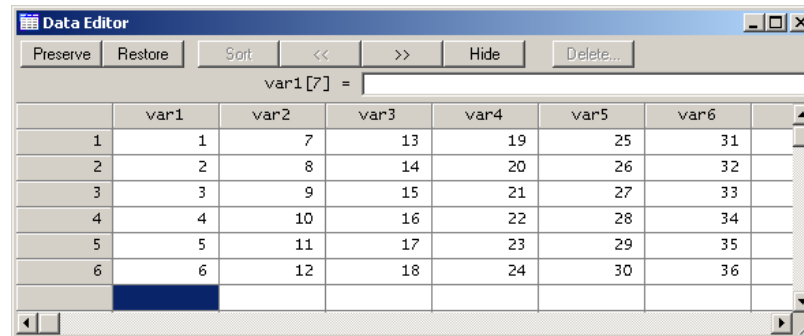
Exemple 1

infile var1 var2 var3 var4 var5 var6 using data\fichier_2, clear

fichier_2.raw

```
1 7 13 19 25 31
2 8 14 20 26 32
3 9 15 21 27 33
4 10 16 22 28 34
5 11 17 23 29 35
6 12 18 24 30 36
```

L'éditeur des données



	var1	var2	var3	var4	var5	var6
1	1	7	13	19	25	31
2	2	8	14	20	26	32
3	3	9	15	21	27	33
4	4	10	16	22	28	34
5	5	11	17	23	29	35
6	6	12	18	24	30	36

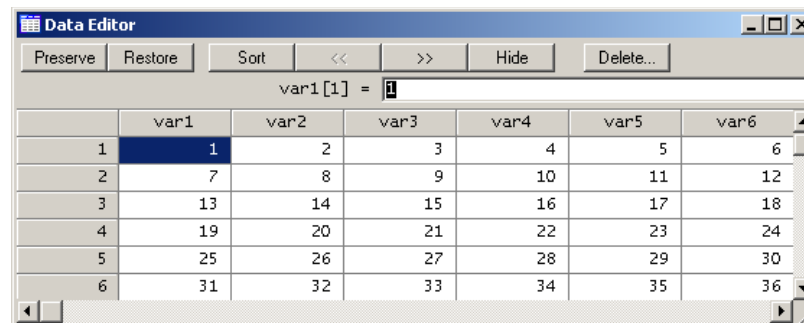
Exemple 2

infile var1-var6 using data\fichier_2, clear byvariable(6)

fichier_2.raw

```
1 7 13 19 25 31
2 8 14 20 26 32
3 9 15 21 27 33
4 10 16 22 28 34
5 11 17 23 29 35
6 12 18 24 30 36
```

L'éditeur des données



	var1	var2	var3	var4	var5	var6
1	1	2	3	4	5	6
2	7	8	9	10	11	12
3	13	14	15	16	17	18
4	19	20	21	22	23	24
5	25	26	27	28	29	30
6	31	32	33	34	35	36

Le deuxième cas concerne l'ouverture de données à format fixe. Le recours à un dictionnaire s'impose donc pour définir le format de positionnement des valeurs ou texte de chaque variable, comme l'illustre l'exemple 3 suivant.

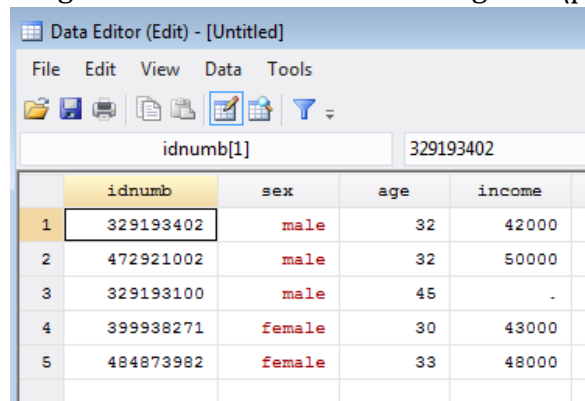
Exemple 3

Les fichiers *persons.raw* et *persons.dct* devront être dans le même emplacement (répertoire).

```
infile using data\persons
persons.raw
329193402male 32 42000
472921002male 32 50000
329193100male 45
399938271female30 43000
484873982female33 48000
persons.dct
dictionary using data\persons.raw {
    _column(5)
        long idnumb %9f "Identification number"
        str6 sex %6s "Sex"
        int age %2f "Age"

    _column(27)
        float income %6f "Income"
}
```

Après l'exécution de la ligne de commande « *infile using data\persons* »



The screenshot shows the SPSS Data Editor window titled "Data Editor (Edit) - [Untitled]". The menu bar includes File, Edit, View, Data, and Tools. Below the menu is a toolbar with icons for opening files, saving, printing, and filtering. A filter icon is currently active. Below the toolbar, a text box shows "idnumb[1]" with the value "329193402" entered. The main data grid has five columns: "idnumb", "sex", "age", and "income". The first row is highlighted in yellow. The data is as follows:

	idnumb	sex	age	income
1	329193402	male	32	42000
2	472921002	male	32	50000
3	329193100	male	45	.
4	399938271	female	30	43000
5	484873982	female	33	48000

4.1.3.3 La commande **infix**

Elle permet de lire les fichiers ASCII avec un format fixe.

Syntaxe

```
infix using dfilename [if] [in] [, using(filename2) clear]
```

où dfilename, s'il existe, est un fichier dictionnaire qui doit contenir l'information suivante:

-----fichier dictionnaire -----

```
infix dictionary [using filename] {  
* commentaires  
  Spécifications  
}  
(Les données peuvent être ici)
```

----- fin du fichier dictionnaire -----

si dfilename est indiqué sans extension, .dct est alors l'extension par défaut.
si filename2 est indiqué sans extension, .raw est alors l'extension par défaut.

Dans la première syntaxe, si l'option **using filename2** n'est pas indiquée dans la ligne de commande, les données sont supposées commencer comme suite à la ligne qui contient la parenthèse} du dictionnaire.

Options

using(filename2) Pour indiquer le fichier qui contient les données.

clear Pour remplacer les données en mémoire.

Exemple

infix poids 1-5 age 7-8 using data\fichier_4, clear

fichier_4.raw

60.30 30
40.20 56
45.45 80
36.10 67

L'éditeur des données

	poids	age
1	60.3	30
2	40.2	56
3	45.45	80
4	36.1	67

4.2 Exportation et sauvegarde des données

Pour sauvegarder les données en cours d'utilisation, Stata offre plusieurs possibilités selon le format que l'utilisateur souhaite obtenir.

Sauvegarder une base de données format Stata: La commande `save`

Cette commande sert à sauvegarder le fichier en mémoire dans un fichier de format Stata (avec extension *.dta).

Syntaxe

save [nom_du_fichier] [, save_options]

Remarques

- Si le nom du fichier contient des espaces, ajouter des doubles guillemets au nom : (Ex. "data 1").
- Le nom peut contenir l'emplacement du fichier (Ex. "c:/data/data 1").

Options

nolabel	Sauvegarder le fichier sans les étiquettes des valeurs.
replace	Remplacer le fichier des données s'il existe déjà.
orphans	Sauvegarder toutes les étiquettes des valeurs.
emptyok	Sauvegarder même si le nombre d'observations est zéro et le nombre de variables est zéro.

Exemple

save data\lsms, replace nolabel

On sauvegarde les données en mémoire dans le répertoire indiqué sous le nom de lsms.dta sans les étiquettes des valeurs.¹

Bonnes pratiques

- 👍 Ajoutez l'option replace pour s'assurer que le fichier Stata s'enregistre même s'il existe déjà.
- 👍 Choisissez un nom court, mais qui indique bien le contenu du fichier.
- 👍 Assurez-vous que toutes les variables et valeurs aient leurs étiquettes avant de faire la sauvegarde.
- 👍 Au besoin, ajoutez une note décrivant le fichier des données avant de faire la sauvegarde.

¹ Nous consacrons plus loin une section qui explique comment étiqueter les variables ou les valeurs des variables modales.

■ Sauvegarder une base de données de format ASCII

Il est également possible de sauvegarder les données en mémoire en format ASCII pour les utiliser ensuite avec un autre logiciel. Pour cela, Stata offre deux possibilités selon que le séparateur de données souhaité est un espace, une tabulation ou une virgule. La commande **outfile** permet de sauvegarder les données en cours avec séparation entre les données sous forme d'espace. La syntaxe est :

outfile varlist using nom_fichier [if] [in] [, options]

Si l'extension du nom_fichier n'est pas spécifiée, l'extension .raw est attribuée par défaut au fichier sauvegardé. Dans la mesure où un logiciel ne peut lire des données séparées par un espace, la commande **outsheet** permet de sauvegarder les données en cours avec séparation entre les données sous forme de tabulation par défaut. Si l'utilisateur préfère une séparation par virgule, l'option comma doit être précisée. La syntaxe est :

outsheet varlist using nom_fichier [if] [in] [, options]

Si l'extension du nom_fichier n'est pas spécifiée, l'extension .out est attribuée par défaut au fichier sauvegardé. Les bases de données exportées avec **outsheet** peuvent être réimportées dans Stata avec la commande **insheet**, et **outfile** avec **infile**.

4.3 Étiqueter les variables et les valeurs des variables (label)

La commande **label** permet d'assigner des étiquettes au fichier des données, aux variables ainsi qu'aux modalités des variables modales. En général, les noms des variables ne permettent pas d'avoir une définition claire pour chaque variable. La syntaxe de cette commande dépend de l'objectif poursuivi :

Étiqueter le fichier des données

label data ["label"]

Étiqueter une variable

label variable varname ["label"]

Étiqueter les modalités d'une variable modale. Ceci requiert deux étapes :

1- Définir les étiquettes des valeurs modales ;

Label define lblname m1 "label_m1" m2 "label_m2"

2- Assigner les étiquettes des modalités à une variable modale.

label values varname lblname

Lister les étiquettes

label list

Supprimer toutes les étiquettes

label drop _all

Exemple

use data\data1

lab drop _all

lab def lzone 1 "rurale" 2 "urbaine"

lab val zone lzone

lab var hhid "Identificateur du ménage"

lab var zone "La zone où vit le ménage"

lab var revenu "Revenu total du ménage"

lab var age_cm "Âge du chef de ménage"

5 Exploration et analyse descriptive des données

Stata permet d'inspecter les variables ou de calculer leurs statistiques descriptives simples.

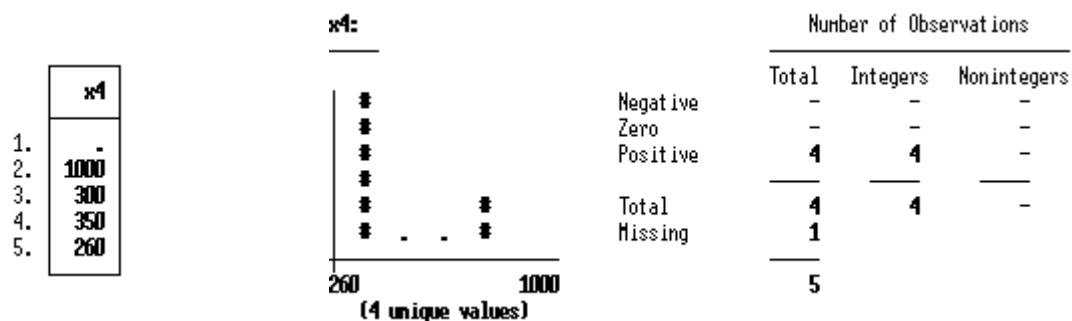
5.1 Inspecter et comparer des variables

La commande **inspect** fournit un sommaire rapide d'une variable numérique. Elle rapporte le nombre de négatifs, de zéros, et de valeurs positives ; le nombre de nombres entiers et de réels ; le nombre de valeurs uniques ; et le nombre de valeurs manquantes ; et il produit un petit histogramme. Son but n'est pas analytique, mais il permet de se familiariser rapidement avec des données inconnues. La syntaxe de cette commande est :

inspect [varlist] [if] [in]

Exemple

```
use      data\fichier1
insp x4
```



La commande **compare** rapporte les différences et les similarités entre deux variables :

compare varname1 varname2 [if] [in]

Exemple

use data\ichier1
compare x4 y

	x4	y		count	difference minimum average maximum
1.	.	.	x4<y	4	-3000 -1680 -520
2.	1000	4000			
3.	300	2100	jointly defined	4	-3000 -1680 -520
4.	350	1750	jointly missing	1	
5.	260	780			
			total	5	

5.2 Les statistiques descriptives simples : Les commandes **summarize** et **tabstat**

La commande **summarize** calcule des statistiques descriptives pour les variables numériques de la base de données². La syntaxe de cette commande est :

[by varlist :] summarize varlist [if] [in] [weight] [, options]

Dans la mesure où aucune option n'est spécifiée, Stata produit pour chaque variable de la varlist le nombre d'observations (Obs), la moyenne (Mean), l'écart-type (Std. Dev.), la valeur minimale (Min.) et la valeur maximale (Max.). L'option **detail** génère des statistiques plus détaillées sur la distribution des variables indiquées par varlist, telles que le coefficient de dissymétrie (*skewness*) (une mesure de l'asymétrie d'une distribution) et la statistique **kurtosis** (une mesure du degré d'aplatissement de la distribution).

Exemple :

La ligne de commande

bysort education : summarize revenu

produit des statistiques descriptives de la variable revenu en fonction des modalités de la variable education.

La commande **tabstat** permet de faire presque la même chose que **summarize**, mais permet plus de flexibilité dans le choix des statistiques descriptives.

² Voir aussi la commande **means** qui calcule les moyennes arithmétique, géométrique et harmonique des variables spécifiées dans varlist.

Exemple :

La ligne de commande

tabstat revenu taille, stats(mean, median, variance, sd, skewness)

produit la moyenne, la médiane, la variance, l'écart-type, et le skewness de la variable revenu ainsi que de la variable taille.

5.3 Tri à plat d'une variable et tri croisé de deux variables : La commande **tabulate**

La commande **tabulate** permet, d'une part, de faire des tris à plat d'une variable, c.-à-d. elle calcule les fréquences des valeurs prises par la variable spécifiée dans varname. La syntaxe de cette commande pour les tris à plat est :

[by varlist :] **tabulate** varname [if] [in] [weight] [, options]

Exemples :

La ligne de commande

tabulate sexe if strate == 5, nolabel

donne le tri à plat de la variable sexe lorsque la variable strate est égale à 5 sans afficher les étiquettes des modalités de la variable sexe.

tabulate sexe, generate(x)

donne le tri à plat de la variable sexe et génère des nouvelles variables discrètes qui reflètent les valeurs observées (les modalités) de la variable sexe.

D'autre part, la commande **tabulate** permet de créer un tableau croisé pour deux variables. La syntaxe de cette commande pour les tris croisés est :

[by varlist :] **tabulate** var1 var2 [if] [in] [weight] [, options]

L'option chi2 permet de procéder au test de Chi-deux (χ^2) de Pearson (Hypothèse nulle : indépendance des lignes et colonnes du tableau croisé).

Remarques

1. La commande **tabulate** ne doit être utilisée qu'avec des variables numériques. En pratique, elle est souvent utilisée avec des variables discrètes ou des variables catégorielles (comme sexe, éducation, région, etc.).
2. Si l'objectif est de faire des tris à plat de plusieurs variables, la commande **tab1** produit un tableau de fréquences pour chaque variable spécifiée dans varlist :

tab1 varlist [if] [in] [weight] [, options]

Si l'objectif est de faire des tris croisés de plusieurs variables, la commande **tab2** produit des tableaux de fréquences croisées pour toutes les combinaisons de deux variables possible des variables spécifiées dans varlist :

tab2 varlist [if] [in] [weight] [, options]

5.4 Obtenir des statistiques descriptives plus élaborées sur une variable : La commande **table**

La commande **table** représente une combinaison des commandes **summarize** et commande **tabulate**. Elle fournit un tableau de statistiques descriptives.

Exemples :

table région

produit une table de fréquence de la variable région.

table région, contents(mean revenu median revenu)

fournit la moyenne et la médiane de la variable revenu par région.

table région education, c(mean revenu median taille)

fournit la moyenne de la variable revenu et la médiane de la variable taille pour chaque modalité de la variable région et par niveau d'éducation.

5.5 Analyse de la corrélation entre les variables : La commande **correlate**

La commande **correlate** permet de calculer les matrices de corrélation ou de covariance entre les variables spécifiées dans varlist. La syntaxe de cette commande est :

[**by** varlist :] **correlate** varlist [if] [in] [weight] [, options]

Les options les plus courantes sont :

Options

covariance Produit la matrice de variance-covariance plutôt que la matrice des corrélations

means Présente des statistiques descriptives.

Exemples

correlate revenu education taille in 1/100, means

calcule et présente la matrice de corrélation (qui aura 3 lignes et 3 colonnes) et les statistiques descriptives (moyenne, écart-type, min, et max) des 100 premières observations variables revenu, education et taille.

correlate revenu education taille, c

calcule et présente la matrice de variance-covariance (qui aura 3 lignes et 3 colonnes) des variables revenu, education et taille.

Remarque

La commande **pwcorr** permet de calculer la matrice des corrélations des variables en permettant d'autres options telles que l'affichage du nombre d'observations (l'option **obs**) et du résultat du test de nullité des différents coefficients de corrélation (l'option **sig**).

5.6 Tests sur la moyenne, la variance et la distribution des variables : Les commandes **ttest** et **prtest**

La commande **ttest** permet de tester la moyenne d'une variable ou de comparer les moyennes de deux variables. La syntaxe de cette commande lorsqu'il s'agit de tester la moyenne d'une variable est :³

```
ttest varname == # [if] [in] [, level(#)]
```

Par contre, lorsqu'il s'agit de comparer les moyennes de deux variables, la syntaxe de **ttest** est :

```
ttest varname1 == varname2 [if] [in] [, options]
```

La commande **ttest** permet aussi de tester la différence de moyenne d'une variable entre deux groupes de la population. Dans ce cas, ce test repose sur l'hypothèse d'égalité de variance des deux variables. L'option unequal permet de relâcher celle-ci. La syntaxe de cette commande est dans ce cas :

```
ttest varname [if] [in], by(groupvar) [ options]
```

où la variable spécifiée dans groupvar ne doit pas contenir plus que 2 modalités.

Exemples

```
ttest taille == 5 if région==3
```

permet de savoir si la moyenne de la taille des ménages de la troisième région est égale à 5 personnes.

```
ttest revenu1990 == revenu2000
```

permet de savoir si la moyenne des revenus en 1990 est différente de la moyenne des revenus en 2000.

```
ttest revenu, by(sexe) unequal // deux échantillons indépendants
```

permet de tester la significativité de la différence de revenu entre les ménages dont le chef est un homme et ceux dont le chef est une femme sous l'hypothèse que la variance de la distribution du revenu des hommes est différente de celle des femmes.

³ Les pondérations `weight` ne sont pas possibles avec **ttest** et **prtest**.

La commande **prtest** ressemble dans sa syntaxe à la commande **ttest**, mais elle concerne les tests de proportion. Lorsqu'il s'agit de tester la proportion d'une variable est :

```
prtest varname == p [if] [in] [, level(#)]
```

Par ailleurs, lorsqu'il s'agit de tester si deux variables ont la même proportion, la syntaxe est :

```
prtest varname1 == varname2 [if] [in] [, options]
```

Enfin, lorsqu'il s'agit de tester la différence de proportion d'une variable entre deux groupes de la population, la syntaxe est:

```
prtest varname [if] [in], by(groupvar) [ options]
```

La commande **sdtest** ressemble à **ttest** mais elle permet de tester la variance d'une variable ou de comparer les variances de deux variables. La syntaxe de cette commande lorsqu'il s'agit de tester la variance d'une variable est :

```
sdtest varname == # [if] [in] [, level(#)]
```

Par contre, si l'objectif est de comparer les variances de deux variables, la syntaxe de **sdtest** est :

```
sdtest varname1 == varname2 [if] [in] [, level(#)]
```

La commande **sdtest** permet également de tester la différence de variance d'une variable entre deux groupes de la population. Dans ce cas, la syntaxe de cette commande est :

```
sdtest varname [if] [in], by(groupvar) [, level(#)]
```

où la variable spécifiée dans groupvar ne doit pas contenir plus que 2 modalités.

6 Manipulation des variables et des observations

6.1 Types de variables

Les variables sous Stata peuvent être numériques ou alphanumériques. Les variables numériques peuvent être de différents types (voir le tableau suivant), selon la précision et la place en mémoire nécessaire. Les variables alphanumériques sont des chaînes de caractères quelconques (*string*).

Les différents types de variables d'une base de données en format Stata sont résumés dans le tableau suivant :

Les types de variables

Type	Description	Minimum	Maximum	bites
Byte	entier (integer)	-127	127	1
Int	entier (integer)	-32,767	32,740	2
Long	entier (integer)	-2,147,483,647	2,147,483,647	4
Float	variable avec decimal	- 1.70141173319*10 ³⁸	1.70141173319*10 ³⁸	4
Double	variable avec decimal	- 8.9884656743*10 ³⁰⁷	8.9884656743*10 ³⁰⁷	8
str#	texte de # caractères (ASCII)	--	80 (Intercolled)	#

La syntaxe suivante permet de transformer une variable alphanumérique en une variable numérique :

destring varlist [, options]

Les principales options sont **gen(var)** et **replace**. La première option crée une nouvelle variable, nommée **var**, contenant la transformation demandée. La seconde option détruit la variable alphanumérique originale pour la remplacer par sa transformation numérique. Par défaut, Stata considère qu'une variable est alphanumérique lorsqu'au moins une observation contient un caractère non numérique.

La commande **recast** permet de changer le type de variable. La syntaxe de cette commande est :

recast type varlist [, force]

L'option **force** oblige l'exécution de la commande spécifiée même lorsque celle-ci entraîne une importante perte d'information.

Exemple :

Supposons que la variable **revenu** est de type **float** et que nous désirons la transformer en une variable **int**. Ce changement peut se faire par l'instruction :

recast int revenu, force

Revenu (type float)	Revenu après l'instruction recast , force
25800,8	25800
30000	30000
32740	32740
35880.4	.

Après cette instruction, la quatrième observation devient manquante, car une variable type **int** ne peut être supérieure à 32740.⁴

⁴ Voir les fonctions `round(varname)`, `floor(varname)`, `ceil(varname)`, `int(varname)` pour arrondir les valeurs des variables.

6.2 Renommer et changer le format d'affichage des variables

La commande **rename** permet de changer le nom d'une variable. Sa syntaxe est la suivante :

rename ancien_nom nouveau_nom

La commande **format** permet de changer le format d'affichage d'une variable. Sa syntaxe est la suivante :

Syntaxe

format varlist %fmt

quelques exemples des formats (fmt)

fmt

%*&.#g format général des valeurs numériques.

%*&.#f format fixe des valeurs numériques.

%*&s le format des variables de type texte (string).

& : espace réservé à l'affichage.

: précision de l'affichage (voir pour les types de données avec Stata).

* : (* = + ou vide) → affichage à droite // (* = -) → affichage à gauche.

Exemple

clear

set obs 1

gen float f_x = 1.1234567890123456

gen double d_x = 12345.1234567890123456

list

format f_x %10.4g

list

format f_x %20.16g

list

format d_x %20.16g

list

6.3 Création de nouvelles variables

Pour créer de nouvelles variables, deux commandes existent. La commande **generate** permet de créer des variables qui n'exigent que des calculs simples et **egen** (*extended generate*) s'impose lorsque les valeurs générées font référence à des statistiques spécifiques telles que la moyenne, la somme, etc.

La commande **generate**

La commande **generate** permet de générer de nouvelles variables. Les valeurs de la variable générée sont indiquées par **= exp** comme le montre la syntaxe suivante :

generate [type] newvar[:lblname] =exp [if] [in]

Si aucun type de variable n'est indiqué, le nouveau type est déterminé automatiquement par le type de résultat retourné par l'expression **= exp**. Une variable de type float (ou un double, selon le type d'ensemble) est créée si le résultat est numérique, et une variable de type string est créée si le résultat est un texte.

Exemples

```
use      data\data1, clear
generate age_cm2 = age_cm*age_cm
generate gap    = revenu<800 & revenu !=.

gen  annee = 2007 /* crée une constante annee, égale à 2007 */
gen  x1 = "pauvre" in 1/10 /* crée une variable string égale à pauvre dans les 10 premières observations */
gen  x2 = (x1 == "pauvre") /* x2 = 1 si x = "pauvre" et 0 sinon */
gen  x3 = (revenu <= 500) /* x3 = 1 si revenu <= 800 et 0 sinon */
gen  x4 = _n /* crée une variable nommée x4 égale au numéro de la ligne */
gen  x5 = revenu[_n-1] /* x5 égale à la valeur de l'observation précédente du revenu */
gen  x6 = ln(revenu) /* crée une variable x6 égale au logarithme du revenu. */
gen  x7 = sum(revenu) /* x7 contient les valeurs cumulées de revenu */
```

Notons qu'il existe plusieurs autres fonctions avec la commande **generate**. Pour plus de détails, se reporter à l'aide de Stata.

La commande egen

La commande **egen** est une extension de la commande **generate**. Sa syntaxe est :

egen [type] newvar = fcn(arguments) [if] [in] [, options]

Exemples

egen x = sum(revenu) /* crée une constante x égale à la somme des revenus */

egen t = mean(revenu) /* crée une constante t, égale à la moyenne des revenus */

egen y = count(revenu), by(région) /* crée une variable y qui indique le nombre d'observations non manquantes de la variable revenu dans chaque région */

```

clear
input zone revenu
      1      10
      1      20
      1      30
      2      15
      2      25
      2      20
      2      90

end
lab def      lzone 1 rurale 2 urbaine
lab val      zone lzone
egen        rev_m  = mean(revenu)
list       zone revenu rev_m, mean( revenu rev_m) labvar(zone) sep(0)

```

	zone	revenu	rev_m
1.	rurale	10	30
2.	rurale	20	30
3.	rurale	30	30
4.	urbaine	15	30
5.	urbaine	25	30
6.	urbaine	20	30
7.	urbaine	90	30
	Mean	30	30

```

by zone, sort: egen rev_m_z = mean(revenu)
list    zone revenu rev_m rev_m_z, mean( revenu rev_m rev_m_z) sep(0) labvar(zone)

```

	zone	revenu	rev_m	rev_m_z
1.	rurale	10	30	20
2.	rurale	20	30	20
3.	rurale	30	30	20
4.	urbaine	15	30	37.5
5.	urbaine	25	30	37.5
6.	urbaine	20	30	37.5
7.	urbaine	90	30	37.5
	Mean	30	30	30

Question

Indiquez l'utilité de chacune des deux lignes de commande suivante :

sum revenu

et

egen rev_moyen = mean(revenu)

Notons qu'il existe plusieurs autres options avec la commande **egen**. Pour plus de détails, se reporter à l'aide de Stata

Bonnes pratiques

- 👍 Utiliser la commande **egen** pour générer aisément une variable qui contient une statistique pour la population ou par groupe telle que la moyenne, la somme, l'écart type, etc. Pour d'autres statistiques, consulter l'aide de Stata pour la commande **egen**.
- 👍 Chaque fois que vous générez ou modifiez une variable, vérifier qu'elle contient bien la valeur de l'expression. Pour cela, vous pouvez l'éditer ou afficher quelques statistiques sommaires sur la variable avec la commande **summarize**, faire un tri à plat, etc.

6.4 Manipulations des variables

Pour manipuler les variables, plusieurs commandes existent. Les sections suivantes résument les commandes les plus utilisées.

La commande **replace** et **recode**

La commande **replace** permet de modifier le contenu d'une variable qui existe déjà.

Syntaxe

replace oldvar =exp [if] [in] [, nopromote]

Exemples

replace taille = 6 if age > 46 & age != . */* Remplace le contenu de la variable taille par 6 si age est supérieure à 46 et contient une valeur non manquante. */*

gen x = "pauvre" in 1/10 */* crée une variable string égale à pauvre dans les 10 premières observations. Par conséquent, x prendra une valeur manquante à partir de la 11^{ème} observation si la taille du fichier est supérieure à 10. */*

replace x = "non pauvre" if x == . */* Remplace toutes les valeurs manquantes par "non pauvre" */*

replace x = "classe moyenne" in 11/20

La commande **recode** transforme les valeurs numériques de la variable spécifiée selon la règle (*rule*) indiquée. Les observations qui ne respectent pas la règle indiquée par la commande **recode** restent inchangées. La syntaxe de base de cette commande est :

recode varlist (règle) [(règle)...] [if] [in] [, options]

où les règles les plus communes sont:

règle	Exemple	Signification
# = #	3 = 1	3 recodé 1
# # = #	2 . = 9	2 et . recodé 9
#/# = #	1/5 = 4	1 à 5 recodé 4
non manquante = #	nonmiss = 8	toutes les valeurs non manquantes
#	miss = 9	deviennent 8
manquante = #		toutes les manquantes deviennent 9

Éliminer des variables ou des observations (drop et keep)

La commande **drop** permet d'éliminer des variables ou des observations. Par contre, la commande **keep** permet de spécifier les variables ou les observations que l'on souhaite conserver.

Exemple 1

Supposons qu'on veut éliminer les observations dont la variable x2 prend une valeur supérieure ou égale à 6 de la base suivante nommée fichier1.dta :

	hhid	x1	x2	x3	x4	y
1.	110101	11	6	6	.	.
2.	110103	11	4	17	1000	4000
3.	210202	21	7	9	300	2100
4.	310101	31	5	0	350	1750
5.	310102	31	3	3	260	780

Cela peut se faire de deux façons :

drop
use data\fichier1
drop if x2 >= 6

keep
use data\fichier1
keep if x2 < 6

Résultat

	hhid	x1	x2	x3	x4
1	110103	11	4	17	1000
2	310101	31	5	0	350
3	310102	31	3	3	260

Exemple 2

Conserver uniquement de la base fichier1.dta les observations dont la variable x1 prend la valeur 31. Proposez une commande permettant de vérifier que vous avez bien réussi votre tâche.

Cela peut se faire de trois façons :

Keep

```
use data\fichier1  
keep if x1 == 31  
summarize x1
```

drop

```
use data\fichier1  
drop if x1 != 31  
tabulate x1
```

drop

```
use data\fichier1  
drop if x1 < 31 | x1 > 31  
tabstat x1, stats (me, sd, mi, ma)
```

Résultat

	hhid	x1	x2	x3	x4
1	310101	31	5	0	350
2	310102	31	3	3	260

Exemple 3

À partir de la base de données suivante, nommée fichier2.dta:

	hhid	x1	x2	x3	x4	x5	x6
1.	110101	11	5	6	220	1100	45
2.	110102	11	3	13	430	1290	39
3.	110103	11	2	17	850	1700	32
4.	210201	21	4	0	180	720	69
5.	210202	21	5	9	340	1700	58

créer deux fichiers. Le premier fichier doit être sauvegardé sous le nom de fichier3.dta et il doit inclure les variables hhid, x1, x2 et x3. Le deuxième doit être sauvegardé sous le nom de fichier4.dta et doit inclure les variables x4 et x5.

Création du fichier fichier3.dta

```
use data\fichier2
keep hhid x1-x3
save data\fichier3, replace
```

	hhid	x1	x2	x3
1	110101	11	5	6
2	110102	11	3	13
3	110103	11	2	17
4	210201	21	4	0
5	210202	21	5	9

Création du fichier fichier4.dta

```
use data\fichier2
keep hhid x4
save data\fichier4, replace
```

	x4	x5
1	220	1100
2	430	1290
3	850	1700
4	180	720
5	340	1700

Ordonner des variables ou des observations (order et sort)

La commande **order** permet d'ordonner les variables. Par exemple :

```
use data\fichier3  
order x1 x2 x3 hhid
```

place la variable x1 en première colonne de la base de données fichier3.dta, x2 en deuxième colonne, x3 en troisième colonne, et hhid en quatrième colonne.

	x1	x2	x3	hhid
1	11	5	6	110101
2	11	3	13	110102
3	11	2	17	110103
4	21	4	0	210201
5	21	5	9	210202

La commande **sort** permet d'ordonner les observations selon l'ordre croissant d'une variable données. Il faut préciser que **sort** réarrange les lignes, donc les autres variables associées à chaque ligne suivent. Dans l'exemple suivant,

```
use data\fichier3
sort x2
```

Stata ordonne les observations du fichier3.dta par ordre croissant de la variable x2. La variable x2 se présente alors dans l'ordre croissant :

	hhid	x1	x2	x3
1	110103	11	2	17
2	110102	11	3	13
3	210201	21	4	0
4	210202	21	5	9
5	110101	11	5	6

La commande **sort** peut être aussi utilisée avec plusieurs variables numériques ou de caractère. Par exemple :

```
use data\fichier3
sort x2 hhid
```

réorganise les données du *fichier3.dta* par ordre croissant de la variable x2 puis par ordre croissant de la variable hhid.

	hhid	x1	x2	x3
1	110103	11	2	17
2	110102	11	3	13
3	210201	21	4	0
4	110101	11	5	6
5	210202	21	5	9

Contrairement à la commande **sort** (qui ne permet de faire qu'un classement par ordre croissant), la commande **gsort** permet de faire un classement par ordre croissant (par défaut ou en ajoutant un signe + devant la variable)⁵ ou un classement par ordre décroissant (en ajoutant un signe - devant la variable). Par exemple :

gsort nom

réarrange les observations par ordre alphabétique croissant de la variable nom. Par contre :

sort age -depense

réarrange les observations par ordre par ordre croissant de la variable age puis par ordre décroissant de la variable dépense.

L'utilisation de l'option stable avec la commande **sort** indique que les observations avec les mêmes valeurs des variables d'ordinations maintiennent le même ordre initial. Par exemple, considérer les données suivantes :

x	b
3	1
1	2
1	1
1	3
2	4

Si vous tapez :

sort x

sans utiliser l'option stable, alors vous pouvez obtenir l'un des 6 résultats possibles suivants :

x b		x b		x b		x b		x b		x b
1 2		1 2		1 1		1 1		1 3		1 3
1 1		1 3		1 3		1 2		1 1		1 2
1 3		1 1		1 2		1 3		1 2		1 1
2 4		2 4		2 4		2 4		2 4		2 4
3 1		3 1		3 1		3 1		3 1		3 1

Sans option stable, l'ordre des observations avec des valeurs égales de la variable x est randomisé. Par contre, si vous tapez :

⁵ Ainsi **sort** gse, **gsort** gse ou **gsort** +gse constituent trois façons différentes pour produire exactement le même résultat.

sort x, stable

vous obtiendrez toujours le premier résultat et jamais l'un des cinq derniers.

L'utilisation des boucles (foreach/forvalues/while)

La commande **foreach** sert à exécuter la même ligne de commande pour plus qu'une variable.

Syntaxe

```
foreach lname {in|of listtype} list {  
    commande faisant référence à `lname'  
}
```

Exemple

Supposons qu'on veut diviser les variables revenu, dep_scol, dep_hab et dep_alim par 12, pour calculer des valeurs mensuelles. On veut aussi générer la variable dep_tot qui contient les dépenses totales par mois.

```
use data/ex_foreach, replace  
gen dep_tot = 0  
foreach var of varlist revenue dep_scol dep_hab dep_alim {  
    qui replace `var' = `var'/12  
    qui replace dep_tot = dep_tot+`var'  
}
```

La commande **forvalues** sert à répéter l'exécution des commandes pour un intervalle donné.

Syntaxe

```
forvalues lname = range {  
    commandes faisant référence à `lname'  
}
```

Exemple

Supposons qu'on veut faire la somme des six variables var1 à var6.

```
generate svar=0
```

```
forvalues i = 1/3 {  
    replace svar = svar + var`i'  
}
```


Syntaxe

```
while exp {  
    commandes Stata  
}
```

Tant que l'expression est vérifiée, les commandes dans la boucle continueront à s'exécuter.

Exemple

```
    local i = 1  
while `i' < 11 {  
    display "`i'"  
    local i = `i' + 1  
}
```

La commande **while** sert à répéter l'exécution des commandes tant que l'expression qui succède cette commande est valide.

Syntaxe

```
while exp {  
    commandes Stata  
}
```

Tant que l'expression exp est vérifiée, les commandes dans la boucle continueront à s'exécuter.

Exemple

```
    local i = 1  
while `i' < 11 {  
    display "`i'"  
    local i = `i' + 1  
}
```

7 Combinaison des bases de données

Stata ne peut ouvrir qu'une seule base de données à la fois. Pour libérer la mémoire de Stata, il faut utiliser la commande `clear`. C'est une opération indispensable avant de charger une autre base de données.

Pour utiliser plusieurs bases de données, la méthode la plus simple consiste à ouvrir la première base de données, l'utiliser, puis la fermer et ouvrir par la suite la deuxième base de données, etc. Toutefois, lorsqu'on a besoin en même temps de variables ou d'observations présentes dans différentes bases de données, il faut combiner ces bases et en créer une nouvelle. Pour cela, trois principales méthodes peuvent être utilisées. Chacune d'elles répond à un besoin spécifique.

7.1 Concaténation (verticale) de plusieurs fichiers (**`append`**)

Cette commande sert à ajouter de nouvelles observations à la base de données en cours d'utilisation. Une première base de données doit tout d'abord être ouverte :

`use` *nom_du_premier_fichier*, `clear`

Ensuite, la commande **`append`** :

`append` `using` *nom_du_deuxième_fichier* [, *options*]

permet de compléter les observations contenues dans *le premier fichier* avec celles contenues dans *le deuxième fichier*.

Exemple

Ajouter les observations du fichier2.dta dans fichier1.dta, éliminer ensuite la variable x6, puis classer les observations par ordre croissant des variables hhid et x2, enfin sauvegarder la nouvelle base de données sous le nom fichier1_2.dta.

Cela peut se faire de deux façons :

Première façon

```
use data\fichier1
append using data\fichier2
drop x6
sort hhid x2, stable
save data\fichier1_2, replace
```


Deuxième façon

```
use data\fichier2
append using data\fichier1
drop x6
sort hhid x2, stable
save data\fichier1_2, replace
```

Résultat

	hhid	x1	x2	x3	x4	y	x5
1	110101	11	5	6	220	.	1100
2	110101	11	6	6	.	.	.
3	110102	11	3	13	430	.	1290
4	110103	11	2	17	850	.	1700
5	110103	11	4	17	1000	4000	.
6	210201	21	4	0	180	.	720
7	210202	21	5	9	340	.	1700
8	210202	21	7	9	300	2100	.
9	310101	31	5	0	350	1750	.
10	310102	31	3	3	260	780	.

Astuce

-  Il est possible d'utiliser la commande **cf** pour s'assurer que les deux fichiers à concaténer verticalement possèdent les mêmes variables avec les mêmes noms (ex. `cf _all using data\fichier2`).

Remarques

1. Si la variable *y* dans le fichier *fichier1.dta* désigne la même chose que la variable *x5* dans le fichier *fichier2.dta* (par exemple le revenu total du ménage, soit le revenu par tête du ménage, nommé *x4*, multiplié par la taille du ménage, nommée *x2*), la nouvelle base *fichier1_2.dta* comprendra deux variables, *y* et *x5* (avec des observations manquantes dans chacune de ces deux variables). Il est donc important que les variables qui désignent la même chose aient le même nom.⁶
2. Si ces deux bases de données sont relatives à deux années différentes (par exemple 2001 pour le *fichier1.dta* et 2002 pour le *fichier2.dta*), il n'est pas toujours possible de distinguer dans le fichier *fichier1_2.dta* les observations de 2001 des observations de 2002.⁷ Pour résoudre ce problème, il est recommandé de créer une variable qui indique l'année de l'enquête avant de concaténer les fichiers.

Le programme suivant indique une parmi plusieurs façons pour tenir compte des deux remarques précédentes :

Programme
<pre>use data\fichier1, clear rename y x5 generate annee = 2001 append using data\fichier2 drop x6 replace annee = 2002 if annee == . sort hhid x2, stable save data\fichier1_2, replace</pre>
Résultat

⁶ La commande **rename** permet de changer le nom d'une variable.

⁷ Dans ce cas particulier, cette distinction est possible tant que les variables *x5* et *y* sont considérées deux variables différentes. Toutefois, si nous substituons le nom de l'une de ces deux variables à l'autre, cette distinction ne sera plus possible.

	hhid	x1	x2	x3	x4	x5	annee
1	110101	11	5	6	220	1100	2002
2	110101	11	6	6	.	.	2001
3	110102	11	3	13	430	1290	2002
4	110103	11	2	17	850	1700	2002
5	110103	11	4	17	1000	4000	2001
6	210201	21	4	0	180	720	2002
7	210202	21	5	9	340	1700	2002
8	210202	21	7	9	300	2100	2001
9	310101	31	5	0	350	1750	2001
10	310102	31	3	3	260	780	2001

7.2 Fusion (horizontale) de plusieurs fichiers (merge)

Pour les besoins d'une analyse, on peut avoir besoin de variables relatives à la même unité d'analyse (mêmes ménages, mêmes individus, mêmes entreprises par exemple), mais qui ne sont pas réunies dans une seule base de données. Ce cas est assez fréquent avec les enquêtes auprès des ménages pour lesquelles les données sont emmagasinées dans plusieurs fichiers et où chacun fait référence à une section principale du questionnaire de l'enquête, par exemple, les caractéristiques socio-démographiques du ménage ou les différentes dépenses sur les biens et services.

La commande **merge** vous permet d'ajouter de nouvelles variables au fichier en cours d'utilisation (plutôt que de nouvelles observations aux variables en cours d'utilisation).

Contrairement à la commande **append**, la commande **merge** nécessite l'observation de certaines règles, notamment :

- Il y a une base de données maître (*master dataset*) et une base de données secondaire (*using dataset*).
- Par défaut, si une variable est présente dans les deux bases, alors les valeurs de la base maître resteront inchangées après la fusion.
- Si certaines variables de la base secondaire ont les mêmes noms que celles de la base maître, mais que le contenu de ces variables est différent, alors on doit changer le nom de ces variables dans l'une des deux bases avant de faire la fusion à l'aide de la commande **rename**.

L'utilisation de la commande **merge** entraîne automatiquement la création par Stata d'une nouvelle variable nommée `_merge` qui résume le résultat de la fusion. Les valeurs possibles de `_merge` sont :

- `_merge = 1` lorsque les données de cette observation proviennent exclusivement de la base maître (*master dataset*);
- `_merge = 2` lorsque les données de cette observation proviennent exclusivement de la base secondaire (*using dataset*) ;
- `_merge = 3` lorsque les données de cette observation proviennent des deux bases.

Il est préférable de produire des statistiques descriptives de la variable `_merge` afin de vérifier si vous avez bien obtenu le résultat désiré.

Fusion observation par observation (one to one by observation)

Lorsque les différents fichiers à fusionner présentent les mêmes observations, comme c'est le cas pour les fichiers fichier3.dta et fichier4.dta (issus de l'exemple 3 de la section 6.4.2) qui concernent les 5 mêmes ménages, alors ils peuvent être fusionnés de la façon suivante :

Programme

```
use      data\fichier3, clear
merge   1:1 _n using data\fichier4
tabulate _merge
```

Résultat

	hhid	x1	x2	x3	x4	x5	_merge
1	110101	11	5	6	220	1100	3
2	110102	11	3	13	430	1290	3
3	110103	11	2	17	850	1700	3
4	210201	21	4	0	180	720	3
5	210202	21	5	9	340	1700	3

Remarques

1. Il ne faut surtout pas réordonner les observations des fichiers à fusionner avant l'opération de *fusion observation par observation*.
2. Dans les fusions observation par observation, _merge = 3 signifie surtout que les deux bases de données ont le même nombre d'observations.
3. Il est fortement recommandé de faire la fusion en utilisant une variable-clé, qui représente aussi un identifiant unique des observations, comme nous allons le voir plus loin.

Dans le cas d'une fusion observation par observation (*one to one merge by observation*), la variable `_merge` ainsi créée doit prendre uniquement la valeur de 3 puisque les données de chaque observation doivent provenir des deux bases de données. Une valeur de `_merge` différente de 3 signifie que la procédure de fusion observation par observation (*one to one merge by observation*) n'est pas adéquate (car les fichiers fusionnés ne présentent pas les mêmes observations). Il faut donc utiliser une procédure de fusion à l'aide d'une variable-clé.

Fusion à l'aide de variables-clés (one to one by key-variables)

Cette procédure est utile lorsque les observations sont pour certaines communes aux deux bases, pour d'autres différentes (comme par exemple, pour les données contenues dans *fichier1.dta* et *fichier2.dta* sous l'hypothèse que ces données sont relatives à même période). En plus des règles générales qui doivent être observées avec la commande **merge**, les règles suivantes sont tout aussi nécessaires :

1. Les deux bases de données doivent contenir au moins une variable commune. Il s'agit d'une variable-clé (*key variable* ou *matching variable*) par laquelle les observations vont être fusionnées.
2. Il est possible d'utiliser plusieurs variables-clés (exemple : par région/département, etc.). Ces variables doivent être, toutefois, du même type (numérique ou de caractère alphanumérique) dans les deux fichiers.
3. Les deux bases de données seront, si nécessaire, classées par ordre croissant de la variable-clé (ou des variables-clés).

D'une manière générale, la fusion de plusieurs fichiers à l'aide de variables-clés peut être exécutée comme suit :

```
use          data\base_1
merge       x1 x2 using data\base_2 data\base_3
```

où `x1` et `x2` sont les variables-clés et `unique` et `sort` sont deux parmi plusieurs autres options possibles qui peuvent être utilisées avec la commande **merge**.

L'option `unique` indique que les variables clés `x1` et `x2` (appelées *matching variables*) représentent des identificateurs iniques des observations dans les fichiers maître et secondaire. Si tel n'est pas le cas, Stata affiche un message d'erreur et la fusion ne sera pas accomplie.⁸

Exemple

*En choisissant le fichier1.dta comme base maître, changer le nom de la variable y par x5 ensuite fusionner le avec le fichier2.dta. Quelle peut être la variable-clé de ces deux fichiers ? Interpréter les résultats de la variable **_merge**. Indiquer les ménages qui se trouvent a priori dans les deux bases, ceux qui se trouvent a priori dans base maître exclusivement et ceux qui se trouvent a priori dans base secondaire exclusivement.*

⁸ La commande `isid` permet aussi de vérifier si les variables-clés représentent un identifiant unique des observations.

Programme

```
use data\fichier1, clear
isid hhid
rename y x5
merge 1:1 hhid using data\fichier2
sort hhid
tabulate _merge
edit, nol
* drop _merge
```

Résultat

	hhid	x1	x2	x3	x4	x5	x6	_merge
1	110101	11	6	6	.	.	45	3
2	110102	11	3	13	430	1290	39	2
3	110103	11	4	17	1000	4000	32	3
4	210201	21	4	0	180	720	69	2
5	210202	21	7	9	300	2100	58	3
6	310101	31	5	0	350	1750	.	1
7	310102	31	3	3	260	780	.	1

Force est de constater que pour les observations communes aux deux bases (_merge = 3), ce sont les valeurs de la base maître qui sont maintenues pour les variables communes aux deux bases (c.-à-d. les variables x1 à x5). Ceci est vrai même lorsque certaines valeurs sont égales à des valeurs manquantes dans la base maître. Pour la mise à jour d'une base maître, nous avons besoin, selon les cas, soit de l'option update uniquement, soit de l'option update suivie de l'option replace.

Mise à jour d'une base de données (merge, update et merge, update replace)

Il s'agit de mettre à jour ou de compléter des données (remplacer des valeurs anciennes ou manquantes par de nouvelles données). Dans ce cas, la base de données secondaire (*using dataset*) est celle contenant les nouvelles données.

Lorsque l'option `update` n'est pas suivie de l'option `replace`, c.-à-d.

```
use      data\base_1
merge    1:1 x using data\base_2, update
```

Les valeurs de la base maître ne seraient modifiées que si elles étaient des valeurs manquantes. Par contre, si l'option `replace` est utilisée avec l'option `update`, c.-à-d.

```
use      data\base_1
merge    1:1 x using data\base_2, update replace
```

même les valeurs non manquantes de la base maître seront remplacées par les valeurs de la base secondaire.

Avec la commande **merge**, `update`, les valeurs possibles de la variable `_merge`, générée par Stata afin de synthétiser les résultats obtenus, sont :

- | | |
|-------------------------|---|
| <code>_merge = 1</code> | lorsque les données de cette observation proviennent exclusivement de la base maître (<i>master dataset</i>); |
| <code>_merge = 2</code> | lorsque les données de cette observation proviennent exclusivement de la base secondaire (<i>using dataset</i>) ; |
| <code>_merge = 3</code> | lorsque les données de cette observation proviennent des deux bases et qu'il n'y a pas de différence entre les deux bases. |
| <code>_merge = 4</code> | lorsque les données de cette observation proviennent des deux bases et que les valeurs manquantes de la base maître sont remplacées par celles qui se trouvent dans la base secondaire. |

`_merge= 5` lorsque les données de cette observation proviennent des deux bases et que les valeurs de la base maître sont différentes de celles qui se trouvent dans la base secondaire.

Exemple 1

Supposons que les bases de données fichier1.dta et fichier2.dta font référence aux mêmes observations, mais produites par deux organismes différents. Supposons que vous faites plus confiance aux données du fichier1.dta, mais que les données du fichier2.dta restent utiles, car elles peuvent servir à compléter les valeurs manquantes du fichier1.dta et à augmenter le nombre total d'observations. Quelle est la meilleure stratégie de fusion de ces deux fichiers ?

Programme

```
use data\fichier1, clear
rename y x5
merge 1:1 hhid using data\fichier2, update
sort hhid
tabulate _merge
```

Résultat

	hhid	x1	x2	x3	x4	x5	x6	_merge
1	110101	11	6	6	220	1100	45	5
2	110102	11	3	13	430	1290	39	2
3	110103	11	4	17	1000	4000	32	5
4	210201	21	4	0	180	720	69	2
5	210202	21	7	9	300	2100	58	5
6	310101	31	5	0	350	1750	.	1
7	310102	31	3	3	260	780	.	1

Dans ce cas, seules les valeurs manquantes de la base maître (*fichier1.dta*) sont remplacées par les valeurs correspondantes dans la base secondaire (*fichier2.dta*). Ainsi, lorsque la commande **merge**, update est utilisée sans l'option replace et que la variable _merge prend la valeur de 5, Stata conserve les valeurs du fichier maître suite à la fusion (sauf lorsque ces valeurs sont des valeurs manquantes). Dans la mesure où les valeurs du fichier secondaire sont considérées plus fiables, il faut alors utiliser la commande **merge** avec les options update replace, c.-à-d.

merge varlist using nom_de_fichier , update replace unique sort

Exemple 2

Vous désirez compléter le fichier1.dta par les variables du fichier2.dta. En outre, vous faites davantage confiance aux données contenues dans le fichier2.dta pour les valeurs des variables communes aux deux fichiers. Fusionnez ces deux fichiers en fixant le fichier1.dta comme fichier maître.

Programme

```
use data\fichier1, clear
rename y x5
merge 1:1 hhid using data\fichier2, update replace
sort hhid
tabulate _merge
```

Résultat

	hhid	x1	x2	x3	x4	x5	x6	_merge
1	110101	11	5	6	220	1100	45	5
2	110102	11	3	13	430	1290	39	2
3	110103	11	2	17	850	1700	32	5
4	210201	21	4	0	180	720	69	2
5	210202	21	5	9	340	1700	58	5
6	310101	31	5	0	350	1750	.	1
7	310102	31	3	3	260	780	.	1

Dans ce cas, les valeurs manquantes et non manquantes de la base maître (*fichier1.dta*) sont remplacées par les valeurs de la base secondaire (*fichier2.dta*).

Structure et fusion

Dans la section précédente, nous avons couvert le cas de fusion d'un par un, soit que chaque observation du fichier maître sera fusionnée avec l'observation correspondante dans le fichier secondaire. Sous forme réduite on note cette structure par 1 :1. Cependant, on peut avoir d'autres structures est ou une observation peut être fusionné avec plusieurs observations. A priori, les trois autres structures sont 1 n // n : 1 // n n.

Exemple 1 : n : 1

Base1			<pre>clear input identifiant Revenu zone 1 10 1 2 3 1 3 4 1 4 2 2 5 2 2 6 3 2 end save data/dt1, replace</pre>		
Base 2			<pre>clear input zone prix_pain 1 10 2 30 end save data/dt2, replace</pre>		
			<pre>use data/dt1, replace merge n:1 zone using data/dt2</pre>		

Exemple 2 : 1 : n

Base1 Identifiant Revenu zone 1 10 1 2 3 1 3 4 1 4 2 2 5 2 2 6 3 2			clear input identifiant Revenu zone 1 10 1 2 3 1 3 4 1 4 2 2 5 2 2 6 3 2 end save data/dt1, replace		
Base 2 zone Prix du pain 1 10 2 30			clear input zone prix_pain 1 10 2 30 end save data/dt2, replace use data/dt2, replace merge 1:n zone using data/dt1		

Exemple 2 : 1 : n

Base1			Base 2		
Identifiant	Revenu	zone	zone	Prix du pain	
1	10	1	1	10	
2	3	1	2	30	
3	4	1			
4	2	2			
5	2	2			
6	3	2			

```
clear
input identifiant Revenu zone
1 10 1
2 3 1
3 4 1
4 2 2
5 2 2
6 3 2
end
save data/dt1, replace
```

```
clear
input zone prix_pain
1 10
2 30
end
save data/dt2, replace

use data/dt2, replace
merge 1:n zone using data/dt1
```

Note : n :n il est difficile d'imaginer un exemple de quand il serait utile. Pour une fusion N: N, la clé n'identifie pas uniquement les observations dans l'un ou l'autre des données. La correspondance est réalisée en combinant des observations avec des valeurs égales de la clé; Dans les valeurs correspondantes, la première observation dans l'ensemble de données maître est appariée à la première observation correspondante dans l'ensemble de données secondaire; La seconde, avec la seconde; etc. S'il ya un nombre inégal d'observations au sein d'un groupe, la dernière observation du groupe plus court est utilisée de façon répétée pour correspondre aux observations ultérieures du groupe plus long. L'utilisation de la fusion m: m n'est pas encouragée.

8 Manipulation des bases de données

Stata contient plusieurs commandes très utiles pour la manipulation des bases de données. La structure des données peut être différente d'un fichier de données à un autre pour des raisons pratiques. Pour l'analyse distributive, on utilise, dans la plupart des cas, les enquêtes auprès des ménages qui portent sur les revenus et dépenses des ménages. Ces fichiers contiennent aussi d'autres informations utiles, telles que les caractéristiques sociodémographiques des membres composant le ménage. En général, il existe deux formes courantes de structure de l'information pour ce type d'enquête. La première forme est de type ménages. Chaque ligne contient de l'information sur un seul ménage sondé. Dans ces fichiers, on doit trouver l'identificateur unique du ménage (c.-à-d. la variable clé). On peut aussi trouver d'autres variables telles que le revenu total du ménage, la taille du ménage, etc. La deuxième forme est de type individus. Chaque ligne contient de l'information sur un seul membre d'un ménage sondé. On trouve aussi dans ce type de fichier l'identificateur unique du ménage. Les variables de ce fichier peuvent concerner l'individu ou le ménage auquel l'individu appartient. Par exemple, on peut trouver des variables telles que l'âge de l'individu, son niveau d'éducation ainsi que le revenu par tête du ménage. À noter ici que pour les fichiers individu, le nombre d'observations qui font référence à un même ménage est égal à la taille de ce ménage.

8.1 La commande **collapse**

La commande **collapse** permet de faire l'agrégation des données. Par exemple, à partir du fichier individus, on peut créer le fichier ménage. La syntaxe de la commande **collapse** est :

collapse clist [if] [in] [weight] [, options]

Options

by(varlist) liste des variables indiquant les groupes à partir desquels les statistiques seront calculées. Si on n'utilise pas cette option, les statistiques seront calculées sur la base de toute la population.

Exemple

Dans cet exemple, on suppose que nous avons le fichier individus suivant :

<i>Fichier individus</i>				
hhid	revenu	taille	f_exp	age
11	300	3	10	34
11	0	3	10	29
11	0	3	10	4
13	260	2	20	34
13	0	2	20	35
16	780	4	12	69
16	140	4	12	45
16	0	4	12	13
16	0	4	12	16

On suppose aussi que le chef du ménage apparaît toujours dans la première ligne des enregistrements qui le concernent. Le but est de créer une base de données de type ménages qui prend la forme suivante :

<i>Fichier ménages</i>				
hhid	revpc	taille	f_exp	age_cm
11	100	3	10	34
13	130	2	20	45
16	230	4	12	69

Les variables du fichier ménages sont définies comme suivant :

hhid	identificateur unique du ménage
revpc	Revenu par tête du ménage
taille	Taille du ménage
f_exp	Facteur d'expansion (ou d'extrapolation)
age_cm	Âge du chef de ménage

La variable `hhid` est celle qu'on peut utiliser comme variable de regroupement dans notre exemple. Les moyennes des variables `revenu` et `taille` et `f_exp` par ménage sont équivalentes aux variables `revpc`, `taille` et `f_exp` dans le fichier ménage. Pour l'âge du chef de ménage, nous commençons par générer une variable égale à l'âge du chef de ménage.

```
use      data\ex_collapse_ind, clear
collapse (mean) revenu taille f_exp (first) age, by(hhid)
rename age age_cm
rename   revenu revpc
lab var  hhid  "Identificateur unique du ménage"
lab var  revpc "Revenu par tête du ménage"
lab var  taille "Taille du ménage"
lab var  f_exp "Facteur d'expansion"
lab var  age_cm "Âge du chef de ménage"
save     c:\data\ex_collapse_men.dta, replace
```

8.2 La commande **expand**

La commande **expand** permet d'augmenter ou de remplacer chaque observation par n copies, où n est un entier. Si l'expression indiquant le n est inférieure à 1 ou si elle est une valeur manquante, elle est interprétée comme étant égale à 1. La syntaxe est :

expand [=]exp [if] [in]

Exemple

À partir du fichier ex_collapse_men.dta, qui se trouve sous le répertoire data, créer un fichier qui contient les variables hhid et f_exp, ce qui correspond dans ce cas à un fichier de données individuelles.

```
use          data\ex_collapse_men, clear
expand       taille
sort  hhid
keep  hhid f_exp
```

8.3 La commande reshape

La commande **reshape long** permet de convertir une base de données d'un format « large » (wide format) à un format « long » (long format) et la commande **reshape wide** permet de faire l'opération inverse.

reshape long stubnames, i(varlist) j(varname) [options]

reshape wide stubnames, i(varlist) j(varname) [options]

Exemple :

Soit les deux bases de données suivantes. La première présente un format large tandis que la seconde présente un format long :

Base 1 : format large (wide format)

	hhid	revenu1980	revenu1990	revenu2000	sexe
1	1	5000	5500	6000	0
2	2	2000	2200	3300	1
3	3	3000	200	1000	0

Base 2 : format long (long format)

	hhid	annee	revenu	sexe
1	1	1980	5000	0
2	1	1990	5500	0
3	1	2000	6000	0
4	2	1980	2000	1
5	2	1990	2200	1
6	2	2000	3300	1
7	3	1980	3000	0
8	3	1990	200	0
9	3	2000	1000	0

Si on possède la première base et qu'on désire la transformer, pour faire des estimations économétriques, de sorte qu'elle ait le format de la deuxième base, on doit exécuter la commande **reshape long** comme suit :

```
use data/ex_reshape_l, replace  
reshape long revenu, i(hhid) j(annee)
```

Par ailleurs, si on dispose de la deuxième base et désire la transformer, pour obtenir le format de la première, on doit plutôt recourir à la commande **reshape wide** :

```
use data/ex_reshape_w, replace  
reshape wide revenu, i(hhid) j(annee)
```

8.4 La commande **contract**

La commande **contract** remplace la base de données en mémoire par une nouvelle base qui contient toutes les combinaisons des variables spécifiées dans varlist qui existent dans la base et une nouvelle variable qui indique la fréquence de chaque combinaison. La syntaxe est :

contract varlist [if] [in] [weight] [, options]

L'option freq(varname) spécifie le nom de la variable qui indique la fréquence de chaque combinaison de varlist. Si cette option n'est pas utilisée, la variable de fréquence sera appelée par défaut _freq.

Exemple :

Supposons que nous disposons du fichier ménage suivant:

<i>Fichier ménages</i>			
hhid	revenu	taille	age
11	100	3	34
13	130	2	45
16	230	4	69
20	130	2	45
24	100	3	34
33	130	2	45

L'exécution de la commande suivante :

contract revenue age, freq(poids)

produit le résultat qui suit:

revenu	age	poids
100	34	2
130	45	3
230	69	1
130	45	2

9 Les bases de programmation en stata

En plus de la possibilité d'écrire un ensemble de lignes de commande Stata et de les enregistrer dans des fichiers texte avec extension .do, Stata permet également aux programmeurs de fournir des routines .ado spécialisées (un fichier automatique) pour ajouter à la puissance du logiciel.

Les fichiers Stata ado servent généralement à effectuer des tâches précises en utilisant des entrées prédéfinies. Par exemple, la commande ***mean*** permet d'estimer la moyenne d'une variable et d'afficher le résultat. L'information d'entrée minimale requise pour cette commande est le nom de la variable dont la moyenne sera estimée.

9.1 Terme de base: local and global macros et scalars

En Stata, une macro peut contenir plusieurs éléments qui sont une combinaison de caractères alphanumériques (plus de 8000 caractères). Une macro locale est généralement définie dans un fichier do ou ado. Une macro globale peut être initialisée à n'importe quel niveau d'exécution Stata et continue d'exister jusqu'à ce qu'elle soit explicitement supprimée par l'utilisateur ou à la fin d'une session Stata.

Exemple 1: Local macros

```
local lcountry CAM UGA BOT SAF
dis "'lcountry'"
local count 0
foreach c of local lcountry {
local count = `count'+1
display "Country `count': `c'"
}
```

Les résultats affichés:

```
Country 1: CAM
Country 2: UGA
Country 3: BOT
Country 4: SAF
```

Dans cet exemple, le local macro `lcountry` contient une liste des noms de quatre pays africains. Remarquez que nous n'utilisons pas le caractère "=" pour assigner la valeur à la macro locale. Cette pratique n'est pas recommandée. En effet, l'utilisation du caractère "=" forcera Stata à évaluer le contenu assigné. Dans Stata, la macro locale peut être considérée comme un alias qui contient le nom et implicitement la valeur.

Exemple 2: Local macros

```
local a = 2
local b `a'
dis "the name of macro b is : `b' "
dis "the value of macro b is : ``b'' "
```

$$\underbrace{\underbrace{\text{'b'}}_a}_2$$

Les résultats affichés:

```
. dis "the name of macro b is : `b' "
the name of macro b is : a

. dis "the value of macro b is : ``b'' "
the value of macro b is : 2
```

- ☐ Remarquez que nous devons mettre le nom de la macro local entre les deux caractères spécifiques: (`) - caractère à cocher gauche et (') - l'apostrophe- pour invoquer son contenu.

Example 3: Global macros

```
global nsqpi square_of_pi
global square_of_pi sqrt(_pi)
dis "$nsqpi"
dis $$nsqpi
```

The displayed results are:

```
. dis "$nsqpi"
square_of_pi

. dis $$nsqpi
1.7724539
```

- Remarquez que nous devons précéder le nom d'une macro globale par le caractère \$ pour inculquer son contenu.
- Les scalaires sont généralement utilisés pour stocker des valeurs numériques ou des résultats numériques. Contrairement aux macros locales ou globales, nous n'avons pas besoin de précéder le scalaire d'un caractère spécial pour faire référence à sa valeur.

Example 4: scalars

```
scalar pi = _pi
dis pi
```

The displayed results are:

```
. scalar pi = _pi

. dis pi
3.1415927
```

9.2 Les *.ado programme de Stata

Cette section traite d'une tâche plus ambitieuse, à savoir comment écrire notre propre programme Stata. Notez tout d'abord qu'un fichier ado qui est enregistré dans les chemins Stata ado peut être exécuté jusqu'à la redéfinition de la commande.

Définir de nouveaux programmes Stata

La première étape dans la conception d'un nouveau fichier Stata ado sera d'écrire un fichier texte contenant le contenu du programme et de l'enregistrer dans un dossier Stata ado (par exemple c: / ado / personal) portant le même nom que celui de la nouvelle commande et avec l'extension *.ado.

```
=====Contents of the file c:/ado/personal/myprog.ado=====
*! myprog v1.0.1 WB Jan2017
capture program drop myprog
program define myprog
version 10.0
args nvar
quietly sum `nvar'
dis "The mean of `nvar' equals:" %16.3f `r(mean)'
end
```

- 1- La première line: `*! myprog v1.0.1 WB Jan2017` est utilisé pour afficher des informations sur la commande ou le programme (version, auteurs, dates, etc.).
- 2- La ligne de commande `capture program drop myprog` équivaut à demander à Stata d'essayer de supprimer le programme avec le nom myprog. Cela évite l'erreur de définition d'un programme déjà défini.
- 3- La ligne de commande `program define myprog` est utilisé pour définir le nouveau programme avec le nom `myprog` et pour marquer son début.
- 4- La ligne de commande `version 10.0` est utilisée pour indiquer la version minimale requise de Stata pour exécuter la nouvelle commande.
- 5- La ligne de commande `args nvar` est utilisée pour indiquer les arguments des entrées au programme. Ce programme estime et affiche la moyenne d'une variable donnée. L'information minimale requise est le nom de cette variable.
- 6- La ligne de commande `end` marque la fin du programme.

La syntaxe d'un programme

La définition de la syntaxe du nouveau programme permet à Stata d'analyser le contenu de la ligne de commande et de saisir les informations saisies (nom des variables, options, etc.). La forme générale de la syntaxe est la suivante:

command [varlist] [=exp] [if] [in] , options

Exemple 4: the syntax of the program

```
*! myprog v1.0.2 WB Jan2017
capture program drop myprog
program define myprog
version 10.0
syntax varlist(min=1) [if] [in]
foreach var of varlist `varlist' {
quietly sum `var' `if' `in'
dis "The mean of `var' equals:" %16.3f `r(mean)'
}
end
```

La ligne de commande - **syntax varlist(min=1) [if] [in]** – montre la forme souhaitée de la syntaxe de la nouvelle commande myprog. Après avoir tapé la commande, l'utilisateur peut indiquer une liste de variables pour estimer leurs moyens. De plus, le programme permet de limiter les observations à utiliser par le qualifiant if et in

Les résultats d'un programme

Les résultats du programme peuvent prendre différentes formes, tel que l'

- affichage des résultats dans la fenêtre des résultats;
- générer une nouvelle variable;
- afficher un graphique;
- stocker les résultats sous forme de scalaires et de matrices;

L'option `rclass` permet de retourner les résultats dans des formats scalaires ou macro.

Exemple 5: the return list

```
*! myprog v1.0.3 UNDP 17April2010
capture program drop myprog
program define myprog, rclass
version 10.0
syntax varlist(min=1) [if] [in]
foreach var of varlist `varlist' {
  quietly sum `var' `if' `in'
  dis "The mean of `var' equals:" %16.3f `r(mean)'
  return scalar m_`var' = `r(mean)'
}
return local var `varlist'
end
```


Faire un programme byable

L'option `byable` permet d'exécuter la commande à travers les groupes de la population.

.

Exemple 6: the return list

```
*! myprog v1.0.4 UNDP 17April2010
capture program drop myprog
program define myprog, rclass byable(recall) sortpreserve
version 10.0
syntax varlist(min=1) [if] [in]
foreach var of varlist `varlist' {
  marksample touse
  quietly sum `var' if `touse'
  dis "The mean of `var' equals to:" %16.3f `r(mean)'
  return scalar m_`var' = `r(mean)'
}
return local var `varlist'
end
```