Cours I. Prise en main du logiciel

1. Le logiciel Stata

Traditionnellement, Stata est un logiciel qui fonctionne avec des lignes de commande tapées par l'utilisateur (différent de SPSS, pareil que SAS). Quelques menus ont été créés dans les dernières versions, notamment pour les graphiques. Le fait de devoir taper des lignes de code plutôt que de cliquer dans des menus est souvent rédibitoire, pourtant cela a des avantages :

- la reproductibilité. Il est souvent nécessaire de pouvoir justifier de la façon dont a été calculé un indicateur. A la lecture d'un article ou d'un rapport, le lecteur doit pouvoir, à partir des mêmes données, obtenir les mêmes résultats. Or avec un logiciel à interface, il n'y a pas de trace de quels clics ont été faits.

- l'extensibilité : le logiciel s'enrichit en permanence. Il est possible de créer une commande en utilisant plusieurs autres. Or les interfaces sont figées et ne changent qu'à la parution de nouvelles versions.

Un des premiers avantages de Stata est son prix. De plus, il est très pratique pour la manipulation de données, notamment de données longitudinales. Enfin il couvre la quasi-totalité des domaines des statistiques et de l'économétrie abordés dans différentes sciences comme l'économie, la santé, l'épidémiologie :
- techniques d'évaluation d'impact (régressions linéaires, modèles de sélection, variables instrumentales)
- modèles dichotomiques (modélisation des déterminants du chômage, de la mortalité)
- données de panel (suivi de gens dans le temps, permet de différencier l'effet individuel de l'effet temps)
- modèles de durée (démographie, durée de chômage)
- séries temporelles (macro-économie, prévisions)
- données d'enquêtes (estimateurs sans biais, précision des indicateurs)

2. Installation de Stata

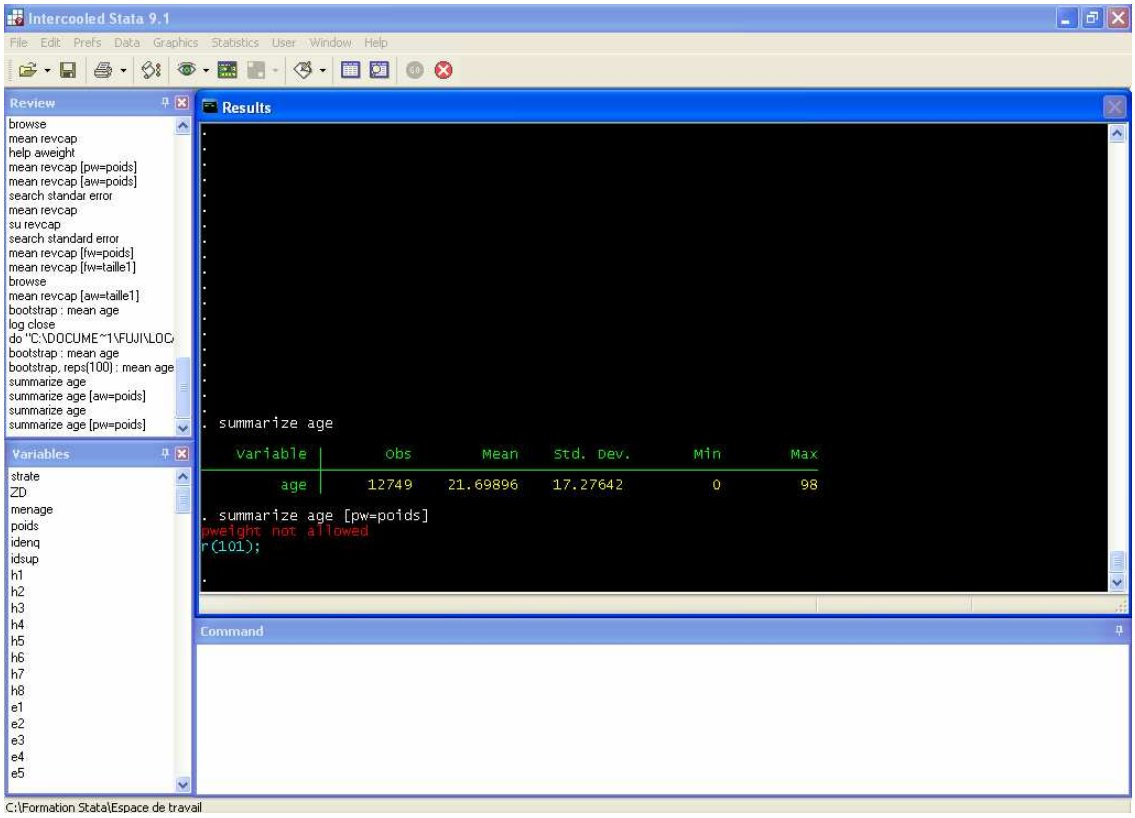
Stata est installé dans C:\Program Files. Dans le dossier Stata 9 se trouve un dossier ado à l'intérieur duquel se trouvent les répertoires base et update (mise à jour). Dans le fichier base sont rangés des fichiers .ado. Chacun de ces fichiers (pouvant s'ouvrir dans l'éditeur de texte, mais à ne pas modifier sans précaution) définit une commande. Ils sont rangés selon la première lettre de la commande. Par exemple, la commande mean est définie dans le fichier mean.ado situé dans C:\Program Files\Stata 9\Stata9\ado\base\m.

Lorsqu'on voudra calculé l'âge moyen des Bamakois, on tapera la commande

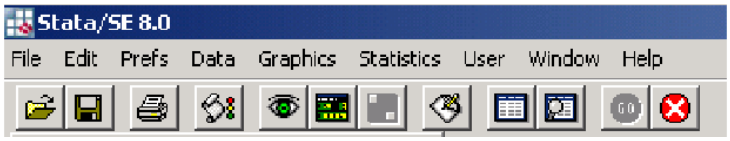
```
mean age
```

qui utilisera le fichier mean.ado pour retourner le résultat attendu. Dans le dossier C:\Program Files\Stata 9\Stata9\ado\update\m se trouve les commandes mises à jour. Si vous achetez le logiciel Stata 10 en 2008, et qu'avant la sortie de la version 11 de nouvelles commandes sont créées, vous pouvez les téléchargez via le site internet de Stata, gratuitement.

3. Les fenêtres



Les 4 fenêtres de base sont les fenêtres Command, Review, Variables et Results. Les commandes sont saisies dans la fenêtre Command puis exécutées en tapant Entrée. Elle sert à tester des commandes mais non à rédiger un programme complet. La fenêtre Review liste les dernières commandes effectuées. Il suffit de cliquer sur l'une d'elle pour la rappeler dans la fenêtre Command. La fenêtre Variables liste les variables figurant dans le base de données. Il suffit de cliquer sur l'une d'elles pour qu'elle soit saisie dans la fenêtre Command. La fenêtre Results, sur fond noir, décrit les résultats de toutes les commandes. Les commandes effectuées y figurent en blanc, les résultats en jaune, vert et blanc, les messages de mise en garde en vert, et les erreurs, stoppant l'exécution, en rouge. Les erreurs sont référencées. En cliquant sur le code de l'erreur, ici r(101), on accède à une description de l'erreur. On peut copier les résultats de Results pour les coller dans Word. Selon qu'on veuille copier du texte ou un tableau (sélectionner uniquement le tableau avec des lignes entières), on utilisera Copy text ou Copy table dans le menu Edition.

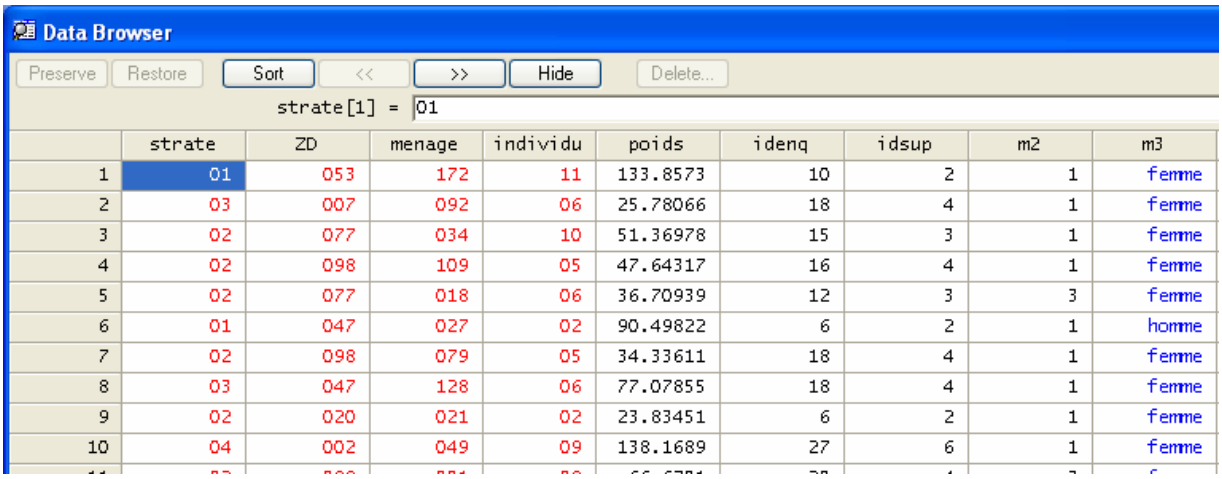


De gauche à droite :
- Ouvrir un fichier de données Stata
- Sauvegarder un fichier de données (équivalent à la commande save)
- Imprimer les résultats tels qu'ils apparaissent dans la fenêtre Results
- Commencer un fichier log, un journal qui conservera tout ce qui a été fait
- Faire apparaître la fenêtre Results lorsqu'elle n'est pas présente
- Ouvrir un do.file
- Afficher la base de données (en pouvant ou pas le modifier manuellement)
- Stopper la commande en cours (si c'est trop long ou qu'on s'est rendu compte d'une erreur)

4. Types de fichiers

dta : Base de données

Les variables sont en colonne, les observations (individus, ménages,…) en ligne. Les quatre premières variables sont des caractères, ce pourquoi elles sont écrites en rouge. La variable poids, idenq, idsup et m2 sont numériques donc écrites en noir. La variable m3 est numérique également, mais elle est "labellisée". C'est-à-dire qu'on a affecté un label qui fait correspondre "homme" à la valeur 1 et "femme" à la valeur 2.



	strate	ZD	menage	individu	poids	idenq	idsup	m2	m3
1	01	053	172	11	133.8573	10	2	1	femme
2	03	007	092	06	25.78066	18	4	1	femme
3	02	077	034	10	51.36978	15	3	1	femme
4	02	098	109	05	47.64317	16	4	1	femme
5	02	077	018	06	36.70939	12	3	3	femme
6	01	047	027	02	90.49822	6	2	1	homme
7	02	098	079	05	34.33611	18	4	1	femme
8	03	047	128	06	77.07855	18	4	1	femme
9	02	020	021	02	23.83451	6	2	1	femme
10	04	002	049	09	138.1689	27	6	1	femme

Pour affecter un label à une variable, on utilise la commande :

```
. label variable m3 genre
```

Ainsi, on saura que m3 contient le genre de l'individu.

On peut également affecter un label à chacune des modalité d'une variable, toujours afin de rendre plus compréhensible la lecture de la base de données ou les résultats.

```
. label define labgenre 1 "homme" 2 "femme"
. label values m3 labgenre
```

ado (déjà vu)

hlp : Fichier d'aide (avec les fichiers ado).

Ils sont appelés en tapant la commande help :

```
help mean
```

qui fait ouvrir un fichier expliquant comment calculer la moyenne d'une variable.

do : programme créé par l'utilisateur.

Il permet de sauver les commandes effectuées et de relancer son programme plus tard ou sur un autre ordinateur. Si je veux calculer le taux de pauvreté du Mali, cela nécessite beaucoup de commandes. Ce n'est pas pratique de les taper une par une dans la fenêtre Command. Donc on les écrit dans un fichier do. En sauvegardant ce fichier do, il sera possible de le relancer et d'obtenir de nouveau les mêmes résultats.

.log : Journal

Fichier pouvant s'ouvrir en dehors de Stata avec le bloc note, il permet de sauvegarder tout ce qui a été fait, les commandes et les résultats.

5. Commencer à travailler

La mémoire :

La mémoire allouée par défaut à la base de données peut parfois être insuffisante lorsque la base contient beaucoup d'observations. Supposons que l'utilisateur veuille dupliquer toutes les observations de la base de données qu'il utilise (command expand), mais que la base devienne alors trop volumineuse. Un message d'erreur apparaît alors. Par exemple, dans l'exemple ci-dessous, la première ligne demandeà Stata d'ouvrir la base de donnée table.dta. Puis la deuxième ligne (expand 2) demande à Stata de dupliquer toutes les observations. Si la mémoire allouée est insuffisante, un message d'erreur d'affiche, en rouge :

```
. use table.dta
. expand 2
no room to add more observations
```

Il est alors nécessaire d'augmenter la mémoire disponible par la commande set memory :

```
set memory 100m
```

Ici, une mémoire de 100 Mo est allouée.

Création d'un répertoire de travail

L'utilisation de Stata nécessite une base de données et un fichier do. Nous verrons par la suite qu'il est possible de créer des fichiers de résultats (log, résultats de régressions, graphiques). Afin de ne pas disperser ses fichiers et de ne pas devoir indiquer le chemin des fichiers à chaque fois, il est utile de créer un répertoire de travail en début de son fichier do.

```
cd "C:\Formation Stata\Espace de travail"
```

Ainsi, par la suite, si l'on souhaite sauvegarder sa base de données, on tapera seulement

```
save table.dta
```

au lieu de

```
save "C:\Formation Stata\Espace de travail\table.dta"
```

Création d'un fichier log, journal des commandes effectuées et des résultats

Afin de garder une trace de toutes les commandes effectuées et des résultats il est conseillé d'ouvrir un fichier log en début de travail

```
log using trace.log
```

ou, si on n'a pas créé de répertoire de travail :

```
log using "C:\Formation Stata\Espace de travail\trace.log"
```

Il se ferme en tapant la commande

```
log close
```

Ouvrir une base de données

```
use table.dta [, clear]
```

Il faut parfois rajouter l'option clear afin d'effacer le fichier de données déjà utilisé par le logiciel. Par précaution, on peut l'écrire tout le temps.

Sauvegarder sa base de données :

```
save table.dta [,replace]
```

Il est également possible de créer "manuellement" une base de données à l'aide de la commande edit qui ouvre une base de données vide de laquelle on remplit les cases. C'est rare d'avoir à le faire. Pour observer la base de données, il suffit de taper la commande browse ou de cliquer sur l'onglet correspondant dans la barre des tâches du logiciel.

Inscrire des commentaires dans son fichier do

Il est très utile de commenter son programme, afin qu'il soit compréhensible à un autre utilisateur et à soi-même si on s'en sert quelques temps après. Pour écrire un commentaire sans que Stata pense qu'il s'agit d'une commande, on utilise l'astérisque ou /*commentaire*/

```
*j'ouvre la base de données
use table.dta
```

ou

```
/*j'ouvre la base de données*/
use table.dta
```

ou encore

```
 use table.dta /*j'ouvre la base de données*/
```

/*commentaire*/ peut également aider à rendre son programme plus lisible. En effet, dans les fichiers do, on peut écrire "sans fin" sur une ligne, pas comme dans Word. Alors, à l'écran on ne voit pas l'ensemble de sa commande. Si on va à la ligne sans précaution, Stata interprètera les deux lignes comme deux commandes distinctes :

```
use "C:\Formation Stata\Documents de travail\base de donnees de la formation\bases
individuelles\donnees sociodemo\base1.dta"
```

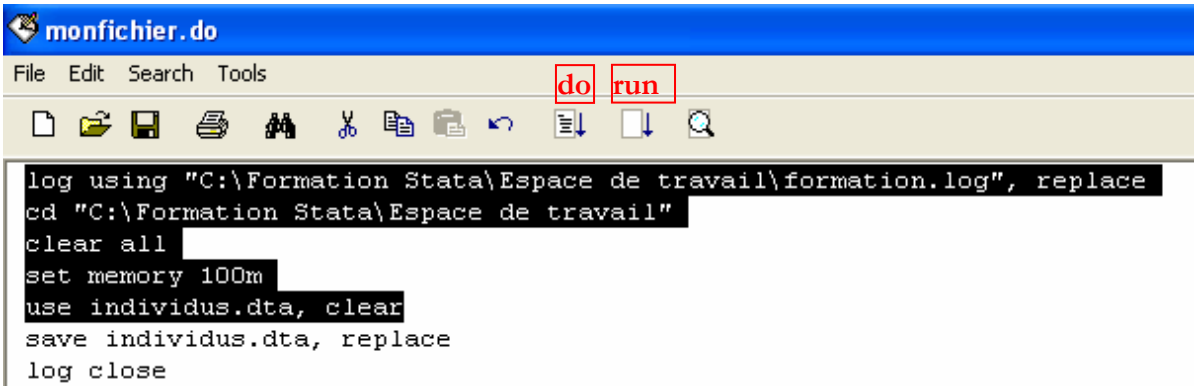
Stata va essayer de lire la ligne

```
individuelles\donnees sociodemo\base1.dta"
```

comme une commande, alors qu'elle n'est que la suite de la commande use. Pour éviter cela, on écrit :

```
use "C:\Formation Stata\Documents de travail\base de donnees /*
*/de la formation\bases individuelles\donnees sociodemo\base1.dta"
```

Exécuter le fichier do : f



Pour exécuter seulement la sélection, on clique sur do. Pour tout exécuter, on ne sélectionne aucune ligne en particulier et on clique sur do. Cliquer sur run exécute également le fichier, mais "silencieusement". C'est-à-dire qu'il n'affiche rien dans la fenêtre Results, et de ce fait rien dans le fichier log. Pourtant il exécute les commandes. De même que pour do, on peut ne sélectionner que certaines lignes.

Cours II. Principes de syntaxe dans Stata

1. Trouver une commande et de l'aide

Pour trouver une commande lorsqu'on ne connaît pas son nom, il suffit de taper la commande search suivie d'un ou plusieurs mots clés. Supposons que l'on souhaite trouver une commande permettant de calculer des statistiques descriptives de base d'une variable :

```
search univariate summary statistics
```

Stata retourne la liste de fichiers d'aide des commandes correspondantes ainsi que des liens hypertextes pointant sur le site web de Stata. Dans l'exemple ci-dessus il retourne deux commandes dont **`summarize`** qui correspond à ce qu'on souhaite.

Pour trouver ensuite des détails sur la façon d'utiliser la commande summarize, on tape

```
help summarize
```

Chaque commande possède un fichier help, en anglais, assez détaillé. Les commandes search et help sont très utilisées car il est impossible de connaître toutes les commandes de Stata.

N'hésitez pas à utiliser la commande help

2. La syntaxe type des commandes dans Stata

La syntaxe est commune à toutes les commandes. La majorité des commandes s'écrive en respectant la syntaxe de base suivante :

`command [`*`varlist`***`] [if `***`exp`***`] [,`***`options`***`]`**

`[`*`varlist`***`]`** :
Il s'agit de la liste de variables à utiliser. Par exemple, si on veut calculer l'age moyen, puis l'âge moyen et le nombre moyen d'années d'études des individus de notre base de données (commande summarize) :

```
. summarize age

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |     12749    21.69896    17.27642          0         98

. summarize age etudes

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |     12749    21.69896    17.27642          0         98
      etudes |     12749    3.350145    4.450341          0         22
```

Pour certaines commandes, si on ne spécifie pas varlist, par défaut Stata applique la commande à toutes les variables _all. C'est le cas entre autres pour summarize.

[if *exp*] :

Supposons qu'on souhaite calculer ces mêmes statistiques descriptives sur les individus vivant dans la commune III de Bamako. On utilise alors if pour indiquer à Stata que la commande ne doit être effectuée que sur les observations vérifiant cette condition d'appartenance à la commune III :

```
. summarize age etudes if commune==3

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |      1924    24.12058     17.8297          0         95
      etudes |      1924    4.954782    4.798545          0         20
```

Si on veut exécuter la commande seulement sur les femmes de la commune III :

```
. summarize age etudes if commune==3 & femme==1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |       923    23.66197    17.39158          0         94
      etudes |       923    4.211268    4.353397          0         20
```

Certains opérateurs sont utilse pour écrire les conditions :

Tableau récapitulatif des opérateurs dans Stata :

Op. arithmétiques	Op. logique	Op. de condition
+ addition	~ not	> supérieur à
- soustraction	! not	< inférieur à
* multiplication	| or	>= supérieur à
/ division	& and	<= inférieur à
^ puissance		== égal à
		~= différent de
+ concaténation		!= différent de

Attention : Pour une condition, l'égalité s'écrit == et non =

[,*options*] :

Quasiment toutes les commandes proposent des options. Celles-ci sont indiquées après une virgule. Par exemple, la commande summarize possède l'option detail qui permet d'obtenir plus de statistiques descriptives (par exemple certains quantiles). Pour connaître toutes les options possibles d'une commande, il faut regarder le fichier d'aide (commande help)

```
. summarize age, detail

                            age
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%            1              0
10%            3              0       Obs               12749
25%            8              0       Sum of Wgt.       12749

50%           17                      Mean           21.69896
                      Largest       Std. Dev.      17.27642
75%           32             96
90%           46             97       Variance       298.4748
95%           55             97       Skewness       .9544977
99%           71             98       Kurtosis        3.52789
```

3. Autres éléments de syntaxe : type de variables et abréviations

Nous avons vu plus haut qu'il existe des variables numériques, contenant des entiers ou des réels, et des variables alphanumériques (string) contenant des chaînes de caractères. Si on manipule une variable alphanumérique, on utilise des guillemets :

```
. generate varnum=10
. generate varalpha="dix"
. generate varalpha2="10"
```

Il est préférable de créer des variables numériques car certaines commandes n'arrivent pas à traiter des variables alphanumériques.

Il n'est pas nécessaire de taper le nom d'une commande en entier pour que Stata la reconnaisse. Par exemple, si j'utilise la commande generate, je peux écrire

`. generate CDM=1 if m5==1` ou `. gener CDM=1 if m5==1` ou
`. gen CDM=1 if m5==1` ou même `. g CDM=1 if m5==1`

On ne peut pas réduire toutes les commandes à une lettre ni même à deux. En effet certaines abréviations peuvent prêter à confusion. Par exemple, je peux abréger :

`. summarize age` en `. su age` mais pas `. s age`
Car Stata ne peut pas s'avoir s'il s'agit de summarize ou de sort age par exemple. Dans le reste du cours, nous écrivons les commandes en entier, mais en TD, vous pouvez les abréger.

Il en est de même pour les noms des variables
`. summarize etudes` en `. su etud` mais pas `. su e`
car il y a une confusion possible entre etudes et une variable nommée e1 par exemple.

Supposons qu'on veuille lancer une commande sur les variables revenu_travail1 revenu_travail2 et revenu_hors_travail.

```
. summarize revenu_travail1 revenu_travail2 revenu_hors_travail
```

On peut taper de façon plus concise :

```
. summarize revenu*
```

Stata exécutera la commande sur toutes les variables dont le nom commence par revenu.

Cours III. Décrire les données

browse :

La commande browse ouvre la base de données, mais ne permet pas de la modifier "à la main"

describe :

La commande describe permet de décrire les données de façon très générale : pour chaque variable, elle retourne le format dans lequel la variable est stockée (double, float, int) le format dans lequel la variable s'affiche (%9.0g, %9s, …) le nom du label des modalités de la variable et le label de la variable. Si on n'indique pas les variables à décrire, Stata décrit toute la base de données, en fournissant également le nombre d'observations, de variables, la taille de la base de données et la date. L'option short permet de n'obtenir que la description de la base.

```
. describe age m3
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
age             float  %9.0g                  age
m3              float  %9.0g       labgenre   genre

. describe
Contains data from table.dta
  obs:        12,749
 vars:           189                          3 Apr 2008 13:42
 size:     9,523,503 (90.9% of memory free)
------------------------------------------------------------------------------
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
strate          str2   %9s
ZD              str3   %9s
menage          str3   %3s                    ménage
```

list :

La commande list permet d'afficher la base de données ou un extrait de cette base dans la fenêtre de résultats. Attention, si on oublie de préciser quelles variables (en tapant seulement list) on veut voir et quelles observations, toutes la base de données s'affiche dans la fenêtre Results !

```
. list age etudes m3 in 1/5
     +----------------------+
     | age   etudes      m3 |
     |----------------------|
  1. |  39       16   homme |
  2. |   4        0   homme |
  3. |  11        3   homme |
  4. |   6        0   femme |
  5. |  25       10   femme |
     +----------------------+
```

N'oubliez pas de préciser les variables et les lignes que vous voulez voir apparaître !

codebook :

La commande codebook permet de créer un dictionnaire des variables indiquant le nom de la variable, son label, son format de stockage, l'intervalle de ses valeurs, ses valeurs uniques, sa moyenne et son écart-type (variable continue), la fréquence des modalités (variable discrète), le nombre de valeurs manquantes, des quantiles, le label de ses modalités. L'option mv fournit des informations sur les valeurs manquantes.

```
. codebook strate age sitac
-------------------------------------------------------------------------------
strate                                                                 strate
-------------------------------------------------------------------------------
                  type:  string (str2)

         unique values:  6                        missing "":  0/12749

            tabulation:  Freq.  Value
                          2271  "01"
                          2115  "02"
                          1924  "03"
                          2291  "04"
                          1992  "05"
                          2156  "06"
-------------------------------------------------------------------------------
age                                                                       age
-------------------------------------------------------------------------------
                  type:  numeric (float)

                 range:  [0,98]                       units:  1
         unique values:  95                       missing .:  0/12749

                  mean:    21.699
             std. dev:    17.2764

           percentiles:        10%       25%       50%       75%       90%
                                 3         8        17        32        46
-------------------------------------------------------------------------------
sitac                                                  situation dans activité
-------------------------------------------------------------------------------
                  type:  numeric (float)
                 label:  lsitac

                 range:  [1,5]                        units:  1
         unique values:  5                        missing .:  0/12749

            tabulation:  Freq.   Numeric  Label
                          4354         1  actif occupe
                           304         2  chomeur BIT
                           277         3  chomeur decourage
                          3937         4  inactif
                          3877         5  moins de 10 ans
. codebook ap13a1 if sitac==1,mv
-------------------------------------------------------------------------------
ap13a1                                                       salaire mensuel
-------------------------------------------------------------------------------
                  type:  numeric (float)

                 range:  [0,5000]                     units:  1
         unique values:  155                      missing .:  1774/4354

                  mean:   49.2143
             std. dev:   191.828

           percentiles:        10%       25%       50%       75%       90%
                                 2         7        20        50        90

        missing values:            h3==mv --> ap13a1==mv
                                  ap2==mv --> ap13a1==mv
```

more :

Si le résultat d'une commande est trop long, par exemple describe sans préciser les variables à décrire, l'édition des résultats se bloque. Il faut alors cliquer sur more (ou taper Entrée) pour relancer l'édition des résultats. Pour que Stata ne bloque pas l'édition des résultats, on peut désactiver cette option avec la commande :

```
set more off
```

Pour réactiver cette option :

```
set more on
```

N'oubliez pas set more off si vous lancez un long programme et vous absentez

Cours IV. Créer des variables

generate :

La commande principale pour créer une variable est la commande generate. Sa syntaxe possède une partie supplémentaire :

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

C'est dans **[=*exp*]** qu'on indique comment construire la variable, par exemple :

```
. generate var3=var1+var2                          /*addition*/
. generate var4=5*var1                             /*multiplication*/
. generate logvar=log(var)                         /*logarithme*/

. generate CDM=1 if m5==1                          /*== : test d'égalité*/
. generate nonCDM=0 if CDM~=1                      /*~= : test de différence*/

. generate CDMfemme=1 if femme==1 & CDM==1         /* & : et*/
. generate CDM_conjoint=1 if CDM==1 | m5==2        /* | : ou*/
```

replace :

La commande replace sert à remplacer la valeur d'une variable déjà existante

```
generate moins10ans=1 if age<10
replace  moins10ans=0 if age>=10
```

La fonction **cond(x,a,b)**

Une fonction est très utile pour créer des variables en tapant moins de ligne de commande, la fonction cond. La création de la variable moins10ans ci-dessus s'écrit en une ligne :

```
generate moins10ans=1 if age<10
replace  moins10ans=0 if age>=10
```

équivaut à

```
generate moins10ans=cond(age<10,1,0)
```

cond(x,a,b) retourne a si la condition x est vérifier, b si elle ne l'est pas. Ici, elle retourne 1 si l'age de l'individu est strictement inférieur à 10, et 0 si cette condition d'age n'est pas vérifiée.

Beaucoup d'autres fonctions existent dans Stata (exponentielle, tirage de nombres aléatoire, extraction de caractères,…). Pour en avoir la liste exhaustive, tapez help functions. Pour avoir la liste des plus utilisées, voir en annexe.

rename :

La commande rename sert à renommer une variable.

```
rename m3 genre
```

recode :

La commande recode sert à recode les modalités d'une variable :

```
recode var1 (règle1) (règle2) …, gen ( )
```

Ces règles peuvent prendre plusieurs formes :

règle	action
3 = 1	3 recodé en 1
2 . = 9	2 et . recodé en 9
1/5 = 4	toute valeur de 1 à 5 recodée en 4
nonmiss = 8	toute valeur non manquante recodée en 8
miss = 9	toute valeur manquante recodée en 9

L'option gen permet de créer une nouvelle variable contenant le nouveau découpage des modalités. Sans spécifier cette option, var1 est remplacée par la variable nouvellement codée.

```
. list
     +------+
     | var1 |
     |------|
  1. |    1 |
  2. |    2 |
  3. |    3 |
  4. |    4 |
  5. |    5 |
     |------|
  6. |    6 |
  7. |    7 |
  8. |    . |
     +------+

. recode var1 (1 2 = 1) (3/7 = 2) (miss = 9), gen (newvar1)
(7 differences between var1 and newvar1)

. list
     +----------------+
     | var1   newvar1 |
     |----------------|
  1. |    1         1 |
  2. |    2         1 |
  3. |    3         2 |
  4. |    4         2 |
  5. |    5         2 |
     |----------------|
  6. |    6         2 |
  7. |    7         2 |
  8. |    .         9 |
     +----------------+
```

encode

La commande encode permet de convertir une variable alphanumérique en une variable numérique discrète, dont les modalités sont "labellisées" avec les chaînes de caractères de la variable initiale. Ici, var1 est initialement une variable alphanumérique prenant les valeurs "absent", "présent" et "visiteur" :

```
. list
     +----------+
     |     var1 |
     |----------|
  1. |   absent |
  2. |  present |
  3. | visiteur |
     +----------+
. encode var1, gen(newvar1)
. list
     +---------------------+
     |     var1    newvar1 |
     |---------------------|
  1. |   absent     absent |
  2. |  present    present |
  3. | visiteur   visiteur |
     +---------------------+

. describe var1 newvar1
              storage  display     value
variable name   type   format      label      variable label
------------------------------------------------------------------------------
var1            str8   %9s
newvar1         long   %8.0g       newvar1
```

La commande decode fait l'opération inverse.

drop :

La commande drop sert à supprimer :

- des variables :

```
drop age
```

- des observations :

```
drop if age<10
```

keep :

La commande keep permet de préciser les variables ou observations que l'on veut conserver, au lieu de celles que l'on veut supprimer :

```
keep etudes genre age
keep if age>=10
```

order :

La commande order sert à ordonner les variables de la base :

```
order strate ZD menage age genre etudes revenu
```

destring :

La commande destring transforme une variable alphanumérique en variable numérique

```
destring strate, gen (str)
```

tostring : Transforme une variable numérique en variable alphanumérique

```
tostring str, gen (strate)
```

Les valeurs manquantes dans Stata :

La façon de Stata de traiter les valeurs manquantes (missing values) peut être différente de celle des autres logiciels de statistique. Il faut y prêter attention lorsqu'on écrit des conditions et lorsqu'on calcule des statistiques.

Par défaut, les valeurs manquantes s'écrivent à l'aide d'un point pour les variables numériques (sysmiss pour system mossing value). Pour les variables alphabétiques, les valeurs manquantes s'écrivent "".

Pour les variables numériques toute valeur non manquante (nmv pour non missing value) est considérée inférieure à .

Vérification d'une condition

Par exemple, si je construis la variable

```
generate moins10ans=1 if age<10
replace moins10ans=0 if age>=10
```

Si l'age d'un individu n'est pas renseigné (age=.), son âge sera considéré comme supérieur à 10 et la variable moins10ans prendra la valeur0, bien qu'on ignore complètement son âge.

Pour éviter cela :

```
generate moins10ans=1 if age<10 & age~=.
replace moins10ans=0 if age>=10 & age~=.
```

ou

```
generate moins10ans=1 if age<10 & age< .
replace moins10ans=0 if age>=10 & age< .
```

ou encore

```
generate moins10ans=1 if age<10
replace moins10ans=0 if age>=10
replace moins10ans=. if age==.
```

Valeurs manquantes en argument de fonction :

```
. + nmv = .
. - nmv = .
. * nmv = .
max(1,5,.)=5
max(.,.,.)=.
```

Certaines fonctions de Stata interprètent . d'une manière spécifique. Par exemple, la fonction inrange.
inrange(x,a,b) retourne la valeur 1 si x appartient à l'intervalle [a,b].
Si a est manquante, elle teste l'appartenance de x à l'intervalle [-∞,b]
Si b est manquante, elle teste l'appartenance de x à l'intervalle [a,+∞]

Les autres fonctions de Stata retourne la valeur . si un ou plusieurs de leurs arguments sont manquants ou invalide.

Regardez toujours si la variable que vous étudiez a des non-réponses.

Les boucles

Supposons qu'on veuille effectuer la même manipulation des variables v1, v2, …, v10, par exemple remplacer la valeur 9 par une valeur manquante

```
replace v1=. if v1==9
replace v2=. if v2==9
…
replace v10=. if v10==9
```

Cela peut allonger le programme et être source d'erreurs. On peut alors créer une boucle :

```
forvalues num=1/10 {
replace v`num'=. if v`num'==9
}
```

Si les variables qu'on veut modifier sont v5 v10 v15 v20, …, v100.

```
forvalues num=5 10 to 100 {
replace v`num'=. if v`num'==9
}
```

Si les variables n'ont pas un nom contenant un chiffre sur lequel effectuer la boucle :

```
foreach statut in actifocc chomeur inactif moins10ans {
summarize `statut'
}

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
    actifocc |     12749     .341517     .474237          0          1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     chomeur |     12749    .0455722    .2085636          0          1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
     inactif |     12749    .3088085    .4620201          0          1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
  moins10ans |     12749    .3041023    .4600442          0          1
```

N'oubliez pas de fermer les accolades. Attention aux quotes (touche 7 puis touche 4)

Cours V. Structure des bases de données

Nous avons vu précédemment que les variables sont en colonne et les observations en ligne. Supposons que nos observations correspondent aux individus des ménages échantillonnés. C'est le cas dans nos exercices et c'est le cas dans la plupart des enquêtes, pour lesquelles on tire un échantillon de ménages puis on interroge chaque membre des ménages tirés.

Créer des identifiants uniques et ayant un sens.

Il faut toujours, avant de travailler sur une base de données, identifier les observations, ici les individus, par un identifiant unique.

Méthode naïve : on crée un identifiant (ident1) dont la valeur est égale au numéro de la ligne.

strate	ZD	menage	individu	age	**ident1**
05	007	146	05	9	**1**
02	039	204	05	5	**2**
02	045	214	01	32	**3**
02	070	094	04	0	**4**
02	077	138	01	38	**5**
02	051	039	08	6	**6**
06	003	058	01	26	**7**
01	106	057	03	2	**8**
01	209	012	01	50	**9**
04	007	162	02	22	**10**

Cet identifiant est bien unique pour chaque individu (on a pas deux individus tels que ident1=3). En revanche il n'a aucun sens.

Supposons que je dispose d'une autre base de données avec les mêmes individus et d'autres variables. Je construis un identifiant selon la même méthode naïve.

strate	ZD	menage	individu	etudes	**ident2**
02	039	204	05	0	**1**
06	003	058	01	15	**2**
05	007	146	05	2	**3**
02	077	138	01	5	**4**
02	070	094	04	0	**5**
04	007	162	02	0	**6**
02	045	214	01	0	**7**
02	051	039	08	2	**8**
01	209	012	01	0	**9**
01	106	057	03	0	**10**

Alors il m'est impossible, à l'aide de ces identifiant seulement, de dire quel âge correspond à quel niveau d'études, puisque l'individu 1 de la 1ère base de donnée n'est pas le même que l'indivdu 1 de la 2ème base de donnée.

En revanche, on remarque tout de suite, qu'il est possible de dire quel âge correspond à quel niveau d'études en regardant les variables strate ZD menage et individu. C'est donc à partir de ces variables qu'on va construire un identifiant efficace. D'une part il a un sens, d'autre part, il est unique, enfin ces variables se trouvent dans toutes les enquêtes auprès des ménages. strate est le numéro de la strate (dans notre base de données, la commune de Bamako). ZD est la zone de dénombrement du recensement. menage est le numéro du ménage échantillonné au sein de la ZD, et individu le numéro de l'individu au sein du ménage.

Il est possible qu'on ait échantillonné le ménage 058 de la ZD 003 et le ménage 058 de la ZD 023. Aussi, le numéro du ménage ne suffit pas à identifier de façon unique un ménage, donc encore moins un individu. La seule façon d'être sûr de l'unicité de l'identifiant est donc d'utiliser ces 4 variables. Ici, on a concaténé les variables strate, ZD, menage et individu en tapant la commande :

```
generate ident3=strate+ZD+menage+individu
```

strate	ZD	menage	individu	etudes	**ident3**
02	039	204	05	0	**0203920405**
06	003	058	01	15	**0600305801**
05	007	146	05	2	**0500714605**
02	077	138	01	5	**0207713801**
02	070	094	04	0	**0207009404**
04	007	162	02	0	**0400716202**
02	045	214	01	0	**0204521401**
02	051	039	08	2	**0205103908**
01	209	012	01	0	**0120901201**
01	106	057	03	0	**0110605703**

strate	ZD	menage	individu	age	**ident3**
05	007	146	05	9	**0500714605**
02	039	204	05	5	**0203920405**
02	045	214	01	32	**0204521401**
02	070	094	04	0	**0207009404**
02	077	138	01	38	**0207713801**
02	051	039	08	6	**0205103908**
06	003	058	01	26	**0600305801**
01	106	057	03	2	**0110605703**
01	209	012	01	50	**0120901201**
04	007	162	02	22	**0400716202**

Il est alors possible, à l'aide de cet identifiant ident3 uniquement, de dire quel âge correspond à quel niveau d'études. En effet l'individu 0500714605 a 9 ans . Ce même individu a déjà fait 2 ans d'années d'études avec succès.

Créez toujours un identifiant des observations, unique et qui ait un sens réel.

Trier la base de données

Supposons qu'on ait construit notre identifiant de l'individu

strate	ZD	menage	individu	idind	genre	age	statut
03	037	097	04	0303709704	femme	24	Autres parents
03	005	070	03	0300507003	homme	26	enfant du CDM
03	073	256	04	0307325604	femme	10	enfant du CDM
03	039	105	09	0303910509	femme	12	enfant du CDM
05	007	118	02	0500711802	femme	18	Conjoint CDM
02	074	009	01	0207400901	homme	65	CDM
05	157	238	03	0515723803	homme	4	enfant du CDM
03	062	151	01	0306215101	homme	33	CDM
06	080	495	09	0608049509	femme	0	enfant du CDM
01	039	033	01	0103903301	homme	35	CDM

Dans la base, on a plusieurs individus pour chaque ménage. Mais si la base n'est pas triée, les individus d'un même ménage ne sont pas à la suite les uns des autres. Il est donc utile de la trier par ménage.

sort :

La commande sort sert à trier la base de données.

```
sort idind
```

Alors les individus d'un même ménage se trouve les uns à la suite des autres :

strate	ZD	menage	individu	segment	idmen	idind	m3	m4	m5
01	009	007	01	01009	01009007	0100900701	homme	39	CDM
01	009	016	01	01009	01009016	0100901601	homme	49	CDM
01	009	016	02	01009	01009016	0100901602	femme	45	Conjoint CDM
01	009	016	03	01009	01009016	0100901603	homme	20	enfant du CDM
01	009	016	04	01009	01009016	0100901604	femme	30	Conjoint CDM
01	009	016	05	01009	01009016	0100901605	femme	14	enfant du CDM
01	009	016	06	01009	01009016	0100901606	femme	6	enfant du CDM
01	009	016	07	01009	01009016	0100901607	homme	11	enfant du CDM
01	009	016	08	01009	01009016	0100901608	femme	25	Conjoint CDM
01	009	016	09	01009	01009016	0100901609	femme	5	enfant du CDM
01	009	016	10	01009	01009016	0100901610	homme	4	enfant du CDM
01	009	025	01	01009	01009025	0100902501	homme	28	CDM
01	009	025	02	01009	01009025	0100902502	femme	23	Conjoint CDM
01	009	025	03	01009	01009025	0100902503	homme	2	enfant du CDM

De plus, comme on a trié par l'identifiant de l'individu et non celui du ménage, les membres d'un même ménage sont rangés par ordre. Souvent, le numéro 1 au chef de ménage, puis les numéro suivants à son (sa ou ses) conjoint(e)(s). C'est le cas ici.

Compte tenu de la façon dont nous avons construit notre identifiant,

sort segment ⇔ sort strate ZD

sort idmen ⇔sort strate ZD menage⇔sort segment menage

sort idind ⇔sort strate ZD menage individu ⇔sort segment menage individu ⇔sort idmen individu

Trier sa base de données n'a pas seulement une utilité visuelle. Une de ses premières fonction est de permettre de calculer des statistiques sur des sous-échantillons définis par les modalités d'une variable. Par exemple, supposons qu'on souhaite calculer ces statistiques descriptives selon le genre. On utilise alors le préfixe by, qui peut s'appliquer à la grande majorité des commandes de statistiques :

[by *varlist***]** : **command [***varlist***] [=***exp***] [if** *exp***] [,***options***]**

Pour pouvoir utiliser by, il faut au préalable trier la base de données selon la variable de genre.

```
. sort femme
. by femme : summarize age etudes
-------------------------------------------------------------------------------
-> femme = 0

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |      6380    22.48652    18.11568          0         97
      etudes |      6380    3.966458    4.855989          0         22
-------------------------------------------------------------------------------
-> femme = 1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
         age |      6369    20.91003    16.35614          0         98
      etudes |      6369    2.732768    3.907078          0         20
```

Le tri permet également de créer des variables constantes pour certains groupes d'observations. Si on dispose du revenu individuel de chaque individu et qu'on souhaite calculer le revenu total du ménage. Cette nouvelle variable doit avoir la même valeur pour tous les membres d'un même ménage.

egen :

La commande egen, extension de generate, permet de créer ces variables :

```
. sort idmen
. egen revmen=total(revind), by(idmen)
. list  idmen idind revind revmen in 1/18
     +---------------------------------------+
     |    idmen        idind   revind   revmen |
     |---------------------------------------|
  1. | 01009025   0100902501      120      252 |
  2. | 01009025   0100902502      132      252 |
  3. | 01009025   0100902503        0      252 |
  4. | 01009034   0100903401     1584     2844 |
  5. | 01009034   0100903402     1200     2844 |
     |---------------------------------------|
  6. | 01009034   0100903403        0     2844 |
  7. | 01009034   0100903404        0     2844 |
  8. | 01009034   0100903405       60     2844 |
  9. | 01009052   0100905201      132      132 |
 10. | 01009052   0100905202        0      132 |
     |---------------------------------------|
 11. | 01009061   0100906101      528      660 |
 12. | 01009061   0100906102      132      660 |
 13. | 01009061   0100906103        0      660 |
 14. | 01009061   0100906104        0      660 |
 15. | 01009061   0100906105        0      660 |
```

Il existe beaucoup d'autres fonctions que total( ), par exemple mean( ), max( ), min( ). Pour en avoir la liste complète, consultez le fichier d'aide de la commande egen (help egen).

Fusionner des bases de données.

append :

La commande append permet de fusionner "horizontalement" deux bases de données en ajoutant les observations de l'une à l'autre. Supposons que j'ai une base de données avec les ménages de la commune I de Bamako et une autre base de données contenant les ménages de la commune II. Je veux les assembler pour ne travailler que sur une base de données. Alors j'utilise la commande append :

```
use table_CommI.dta
append using table_CommII.dta
```

table_CommI.dta :

commune	strate	ZD	idmen	idind	m3
1	01	009	01009025	0100902501	homme
1	01	009	01009025	0100902502	femme
1	01	009	01009025	0100902503	homme
1	01	009	01009034	0100903401	homme
1	01	009	01009034	0100903402	femme

table_CommII.dta :

commune	strate	ZD	idmen	idind	m3
2	02	003	02003011	0200301101	homme
2	02	003	02003011	0200301102	femme
2	02	003	02003011	0200301103	homme
2	02	003	02003020	0200302001	homme
2	02	003	02003020	0200302002	femme

Résultat de la commande append :

commune	strate	ZD	idmen	idind	m3	
1	01	009	01009025	0100902501	homme	
1	01	009	01009025	0100902502	femme	table_CommI
1	01	009	01009025	0100902503	homme	
1	01	009	01009034	0100903401	homme	
1	01	009	01009034	0100903402	femme	
2	02	003	02003011	0200301101	homme	
2	02	003	02003011	0200301102	femme	table_CommII
2	02	003	02003011	0200301103	homme	
2	02	003	02003020	0200302001	homme	
2	02	003	02003020	0200302002	femme	

La command append ne nécessite pas de trier les observations. Si une variable n'existe que dans une des deux tables, par exemple dans la base table_CommI, elle figurera dans la base finale, avec des valeurs manquantes pour les observations de la commune II.

merge :

La commande merge permet de fusionner "verticalement" deux bases de données contenant des individus en commun et des variables différentes. Supposons que je dispose des données socio-démographiques des individus dans une base de données, et des variables relatives à l'emploi dans une autre base de données. Je veux les assembler en étant sure d'affecter les bonnes valeurs des variables d'emploi aux bons individus. Alors j'utilise la commande merge :

```
use sociodemo.dta, clear
sort idind
save sociodemo.dta, replace
use emploi.dta, clear
sort idind
merge idind using sociodemo.dta
```

Il est nécessaire de créer le même identifiant dans les deux bases et de trier les bases selon cet identifiant.

sociodemo.dta :

idmen	idind	m3
01009025	0100902501	homme
01009025	0100902502	femme
01009025	0100902503	homme
01009034	0100903401	homme
01009034	0100903402	femme

emploi.dta :

idmen	idind	sitac
01009025	0100902501	actif occupé
01009025	0100902502	actif occupé
01009025	0100902503	moins de 10 ans
01009034	0100903401	actif occupé
01009034	0100903402	actif occupé

Résultat de la commande merge :

idmen	idind	m3	sitac	_merge
01009025	0100902501	homme	actif occupé	3
01009025	0100902502	femme	actif occupé	3
01009025	0100902503	homme	moins de 10 ans	3
01009034	0100903401	homme	actif occupé	3
01009034	0100903402	femme	actif occupé	3

La commande merge crée automatiquement une variable nommée _merge qui vaut :
- 1 si l'observation ne se trouvait que dans la base de données "maître" (master data), ici emploi.dta
- 2 si l'observation ne se trouvait que dans la base de données "à utiliser" (using data)
- 3 si l'observation se trouvait dans les deux bases de données

Aussi, pour s'assurer de la qualité de la fusion, il est nécessaire de regarder la distribution de la variable _m. Ici :

```
tabulate _merge
     _merge |      Freq.     Percent        Cum.
------------+-----------------------------------
          3 |          5      100.00      100.00
------------+-----------------------------------
      Total |          5      100.00
```

Ici les 5 observations se trouvaient dans les deux bases de données.

Une fois la qualité de la fusion vérifier, on peut supprimer la variable _merge :

```
drop _merge
```

Autre exemple :

sociodemo.dta :

idmen	idind	m3
01009025	0100902502	femme
01009025	0100902503	homme
01009034	0100903401	homme
01009034	0100903402	femme

emploi.dta :

idmen	idind	sitac
01009025	0100902501	actif occupé
01009025	0100902502	actif occupé
01009034	0100903401	actif occupé
01009034	0100903402	actif occupé

Résultat de la commande merge :

idmen	idind	m3	sitac	_merge
01009025	0100902501	.	actif occupé	2
01009025	0100902502	femme	actif occupé	3
01009025	0100902503	homme	.	1
01009034	0100903401	homme	actif occupé	3
01009034	0100903402	femme	actif occupé	3

On vérifie la qualité de la fusion :

```
tabulate _merge
    _merge |      Freq.     Percent        Cum.
-----------+-----------------------------------
         1 |          1       20.00      100.00
         2 |          1       20.00      100.00
         3 |          3       60.00      100.00
-----------+-----------------------------------
     Total |          5      100.00
```

_merge prend la valeur 1 pour l'individu "0100902503". En effet, il est bien dans la base "maîtresse", sociodemo.dta, mais pas dans la base utilisée, emploi.dta. On ignore donc sa variable de statut vis-à-vis de l'activité dont la valeur est manquante dansla base finale. De la même façon, _merge prend la valeur 2 pour l'individu "0100902501". En effet, il est bien dans la base utilisée, emploi.dta, mais pas dans la base "maîtresse", sociodemo.dta. On ignore donc sa variable de genre, dont la valeur est manquante dansla base finale.

Vérifier la qualité du merge. Pensez à supprimer ensuite la variable _merge

collapse : A UTILISER AVEC PRECAUTION

La commande collapse remplace la base de données utilisée par une table de statistiques descriptives. Supposons que la base de données contienne le revenu de chaque membre des ménages. Si je veux créer une base ne contenant qu'une seule ligne par ménage avec comme variable le revenu total du ménage :

```
. list  idmen idind revind
     +--------------------------------+
     |    idmen        idind   revind |
     |--------------------------------|
  1. | 01009025   0100902502      132 |
  2. | 01009025   0100902503        0 |
  3. | 01009025   0100902501      120 |
  4. | 01009034   0100903401     1584 |
  5. | 01009034   0100903402     1200 |
     |--------------------------------|
  6. | 01009034   0100903403       60 |
  7. | 01009034   0100903404        0 |
  8. | 01009043   0100904302      132 |
  9. | 01009043   0100904301        0 |
 10. | 01009043   0100904304        0 |
     |--------------------------------|
 11. | 01009043   0100904303      396 |
     +--------------------------------+
. sort idmen
. collapse (sum) revind, by(idmen)
. list
     +-------------------+
     |    idmen   revind |
     |-------------------|
  1. | 01009025      252 |
  2. | 01009034     2844 |
  3. | 01009043      528 |
     +-------------------+
```

Cette commande remplaçe la base de données initiale par une base de données contenant des statistiques descriptives (ici le total du revenu). Il convient donc de vérifier qu'on a bien enregistré sa base de données avant d'utiliser collapse. Par prudence, on lui préfère egen, qui calculera aussi le revenu total du ménage, mais en conservant toutes les observations individuelles (voir egen, dans le même chapitre) :

Attention : lorsqu'on débute, egen est préférable à collapse

expand

La commande expand permet de dupliquer les observations :

```
. list
     +------------------+
     |      idind   age |
     |------------------|
  1. | 0100900701    39 |
  2. | 0100901603    20 |
     +------------------+
. expand 3
(4 observations created)
. list
     +------------------+
     |      idind   age |
     |------------------|
  1. | 0100900701    39 |
  2. | 0100901603    20 |
  3. | 0100900701    39 |
  4. | 0100900701    39 |
  5. | 0100901603    20 |
     |------------------|
  6. | 0100901603    20 |
     +------------------+
```

Les variables systèmes _n et _N

Une variable système est une variable crée par Stata mais qui n'est pas présente dans la base de donnée. Elles existent dès lors qu'une base de données est chargée dans Stata. Les deux variables système les plus utilisées sont _n et _N.

_n contient le numéro de l'observation courante. Par exemple, lorsqu'on utilise la commande list, des numéros figurent à gauche de la première variable (en rouge ici, mais en vert dans la fenêtre Results) :

```
. list  idmen idind m3

     +-------------------------------+
     |    idmen        idind      m3 |
     |-------------------------------|
  1. | 01009025   0100902501   homme |
  2. | 01009025   0100902502   femme |
  3. | 01009025   0100902503   homme |
  4. | 01009034   0100903401   homme |
  5. | 01009034   0100903402   femme |
     |-------------------------------|
  6. | 01009034   0100903403   femme |
  7. | 01009034   0100903404   homme |
  8. | 01009034   0100903405   femme |
  9. | 01009052   0100905201   homme |
 10. | 01009052   0100905202   femme |
     +-------------------------------+
```

Ces numéros sont la valeur de la variable _n pour chaque observation. _n est donc la position de l'observation dans la base de données. Si je trie la base de données dans un autre ordre, l'individu "0100902501" n'aura plus la même valeur de _n :

```
. list  idmen idind m3

     +-------------------------------+
     |    idmen        idind      m3 |
     |-------------------------------|
  1. | 01009034   0100903403   femme |
  2. | 01009052   0100905201   homme |
  3. | 01009025   0100902503   homme |
  4. | 01009034   0100903405   femme |
  5. | 01009052   0100905202   femme |
     |-------------------------------|
  6. | 01009025   0100902502   femme |
  7. | 01009034   0100903401   homme |
  8. | 01009034   0100903402   femme |
  9. | 01009034   0100903404   homme |
 10. | 01009025   0100902501   homme |
     +-------------------------------+
```

L'individu "0100902501" a désormais comme valeur de _n le nombre 10 car il est sur la 10ème ligne de la base de données, alors qu'avant, il avait comme valeur de _n le nombre 1 car il était sur la 1ère ligne de la base de données. _n est donc le numéro de la ligne de l'individu dans la base.

La variable _n peut également compter les observations au sein d'un groupe, par exemple du ménage.

```
. sort idmen

. by idmen : list idmen idind m3

-------------------------------------------------------------------------------
-> idmen = 01009025
     +--------------------------------+
     |    idmen        idind      m3 |
     |--------------------------------|
  1. | 01009025   0100902501   homme |
  2. | 01009025   0100902502   femme |
  3. | 01009025   0100902503   homme |
     +--------------------------------+

-------------------------------------------------------------------------------
-> idmen = 01009034
     +--------------------------------+
     |    idmen        idind      m3 |
     |--------------------------------|
  1. | 01009034   0100903401   homme |
  2. | 01009034   0100903402   femme |
  3. | 01009034   0100903404   homme |
  4. | 01009034   0100903405   femme |
  5. | 01009034   0100903403   femme |
     +--------------------------------+

-------------------------------------------------------------------------------
-> idmen = 01009052

     +--------------------------------+
     |    idmen        idind      m3 |
     |--------------------------------|
  1. | 01009052   0100905202   femme |
  2. | 01009052   0100905201   homme |
     +--------------------------------+
```

Supposons que je ne dispose que de l'identifiant du ménage. Je veux numéroter les individus au sein des ménages sans règle particulière (une règle fréquemment utiliser est d'attribuer le numéro 1 au chef du ménage, mais pour l'exemple, on suppose que cela nous est égal).

```
. sort idmen

. by idmen : gen num_individu=_n

. list  idmen num_individu m3
     +-----------------------------+
     |    idmen   num_in~u      m3 |
     |-----------------------------|
  1. | 01009025          1   homme |
  2. | 01009025          2   femme |
  3. | 01009025          3   homme |
  4. | 01009034          1   homme |
  5. | 01009034          2   femme |
     |-----------------------------|
  6. | 01009034          3   homme |
  7. | 01009034          4   femme |
  8. | 01009034          5   femme |
  9. | 01009052          1   femme |
 10. | 01009052          2   homme |
     +-----------------------------+
```

_N contient le nombre total d'observations dans la base de données, c'est-à-dire le maximum de _n. Par exemple, dans la base de données ci-dessus, _N=10.

De la même façon que pour _n, _N peut être calculé par sous-groupe, par exemple par ménage.

```
. sort idmen

. by idmen : generate taille=_N

. list idmen idind taille
     +--------------------------------+
     |    idmen        idind   taille |
     |--------------------------------|
  1. | 01009025   0100902501        3 |
  2. | 01009025   0100902502        3 |
  3. | 01009025   0100902503        3 |
  4. | 01009034   0100903401        5 |
  5. | 01009034   0100903402        5 |
     |--------------------------------|
  6. | 01009034   0100903403        5 |
  7. | 01009034   0100903404        5 |
  8. | 01009034   0100903405        5 |
  9. | 01009052   0100905201        2 |
 10. | 01009052   0100905202        2 |
     +--------------------------------+
```

Un exemple courant d'utilisation de ces deux variables (_n et _N) est la recherche et la suppression de duplications dans une base de données. Il arrive souvent que des erreurs dans la saisie des données conduisent à des duplications d'observation. C'est-à-dire qu'un même individu figurera deux fois dans la base de données. Dans certains cas, ces duplications sont nombreuses. Aussi, lorsqu'on récupère une base de données, il convient de vérifier l'existence de duplications et le cas échéant, de les supprimer.

Si deux lignes sont strictement identiques, elles auront le même identifiant de l'individu

```
. sort idind
. by idind : generate dupli = cond(_N==1,0,_n)
```

dupli vaut 0 si si _N==1, c'est-à-dire si le nombre d'observations par valeur d'identifiant est égal à 1, c'est-à-dire si un seul individu correspond à un seul identifiant. Donc, dupli vaut 1 si l'individu n'est pas dupliqué.

Si _N n'est pas égal à 1, c'est-à-dire s'il est supérieur à 1, cela veut dire que plusieurs observations ont le même identifiant, par exemple _N=2 observations ont le même identifiant. C'est-à-dire qu'un individu figure deux fois dans la base de données. Si plusieurs observations ont le même identifiant, dupli vaut _n.

Exemple :

```
. list
     +--------------------------+
     |      idind      m3   age |
     |--------------------------|
  1. | 0303512904   homme     6 |
  2. | 0201501802   femme    35 |
  3. | 0301709104   homme    25 |
  4. | 0511720405   homme     1 |
  5. | 0112302203   femme    13 |
     |--------------------------|
  6. | 0201501802   femme    35 |
  7. | 0201501802   femme    35 |
  8. | 0511720405   homme     1 |
     +--------------------------+
```

Si on se penche sur la base de données, on s'aperçoit que les lignes 2, 6 et 7 sont les mêmes et que les lignes 4 et 8 sont identiques également. C'est encore plus visible lorsqu'on trie la base selon idind :

```
. sort idind
. list
     +--------------------------+
     |      idind      m3   age |
     |--------------------------|
  1. | 0112302203   femme    13 |
  2. | 0201501802   femme    35 |
  3. | 0201501802   femme    35 |
  4. | 0201501802   femme    35 |
  5. | 0301709104   homme    25 |
     |--------------------------|
  6. | 0303512904   homme     6 |
  7. | 0511720405   homme     1 |
  8. | 0511720405   homme     1 |
     +--------------------------+
```

On crée alors la variable dupli :

```
. by idind : generate dupli = cond(_N==1,0,_n)
. list
     +----------------------------------+
     |      idind      m3   age   dupli |
     |----------------------------------|
  1. | 0112302203   femme    13       0 |
  2. | 0201501802   femme    35       1 |
  3. | 0201501802   femme    35       2 |
  4. | 0201501802   femme    35       3 |
  5. | 0301709104   homme    25       0 |
     |----------------------------------|
  6. | 0303512904   homme     6       0 |
  7. | 0511720405   homme     1       1 |
  8. | 0511720405   homme     1       2 |
     +----------------------------------+
```

dupli vaut 0 lorsque l'observation est unique. dupli vaut 1 pour la première occurrence de l'identifiant "0201501802", 2 pour la deuxième occurrence et 3 pour la troisième. De même, dupli vaut 1 pour la première occurrence de l'identifiant "0511720405" et 2 pour la deuxième occurrence. Pour supprimer les duplications, il suffit alors de taper :

```
. drop if dupli>1
(5 observations deleted)
. list
     +----------------------------------+
     |      idind      m3   age   dupli |
     |----------------------------------|
  1. | 0201501802   femme    35       2 |
  2. | 0201501802   femme    35       3 |
  3. | 0511720405   homme     1       2 |
     +----------------------------------+
```

Attention, il ne faut pas garder uniquement les observations pour lesquels dupli=0, mais bien celles pour qui dupli=0 ou dupli=1. Sinon vous supprimeriez toutes les réplications d'un individu et il disparaîtrait de la base.

Cours VI. Commandes de bases pour obtenir des statistiques descriptives

tabulate :

La commande tabulate sert à calculer les fréquences d'apparition des modalités d'une variable discrète :

```
. tabulate m14b

        type |
enseignement |      Freq.     Percent        Cum.
 ------------+-----------------------------------
   sans educ |      6,492       50.92       50.92
     fonda 1 |      3,077       24.14       75.06
     fonda 2 |      1,604       12.58       87.64
   Lycée Gal |        447        3.51       91.14
    Lycée TP |        660        5.18       96.32
         Sup |        469        3.68      100.00
 ------------+-----------------------------------
       Total |     12,749      100.00
```

L'option nolabel permet d'affichier la valeur des modalités plutôt que leur label. L'option plot représente la répartition des modalités avec des lignes d'étoiles :

```
. tabulate m14b, plot nolabel

        type |
enseignement |      Freq.
-------------+------------+-----------------------------------------------------
           0 |      6,492 |**************************************************
           1 |      3,077 |************************
           2 |      1,604 |*************
           3 |        447 |****
           4 |        660 |*****
           5 |        469 |****
 ------------+------------+-----------------------------------------------------
       Total |     12,749
```

L'option gen est très utile car elle permet de créer une variable indicatrice correspondant à chacune des modalités. Ici, on créé 6 variables nommées niveau1, niveau2,…niveau6 :

```
 . tabulate m14b, gen (niveau)
```

La commande tabulate permet également de créer des tableaux croisés de variables. Ici, on croise la situation dans l'activité du chef de ménage avec le quintile de revenu du ménage.

```
. tabulate sitac quintiles if CDM==1, row col nofreq

+-------------------+
| Key               |
|-------------------|
|  row percentage   |
| column percentage |
+-------------------+

 situation |                5 quantiles of revcap
  activité |         1          2          3          4          5 |     Total
-----------+--------------------------------------------------------+----------
     actif |     10.17      17.36      18.90      24.44      29.13 |    100.00
    occupe |     38.98      78.55      87.65      90.53      92.40 |     77.96
-----------+--------------------------------------------------------+----------
   chomeur |     76.81       8.70       5.80       5.80       2.90 |    100.00
       BIT |     10.82       1.45       0.99       0.79       0.34 |      2.86
-----------+--------------------------------------------------------+----------
   chomeur |     50.00      23.33       3.33      10.00      13.33 |    100.00
 decourage |      3.06       1.69       0.25       0.59       0.68 |      1.25
-----------+--------------------------------------------------------+----------
   inactif |     53.47      17.59      10.42       9.49       9.03 |    100.00
           |     47.14      18.31      11.11       8.09       6.59 |     17.93
-----------+--------------------------------------------------------+----------
     Total |     20.34      17.23      16.81      21.05      24.57 |    100.00
           |    100.00     100.00     100.00     100.00     100.00 |    100.00
```

Les options row et col indiquent qu'on souhaite voir les pourcentages en ligne et en colonne. L'option nofreq précise qu'on ne souhaite pas voir les effectifs.

La commande tabulate permet également d'effectuer un test du χ^2. L'option expected permet de voir les effectifs attendus, si les variables étaient indépendantes.

```
. tabulate sitac quintiles if CDM==1, chi2 expected
+--------------------+
| Key                |
|--------------------|
|     frequency      |
| expected frequency |
+--------------------+

 situation |                5 quantiles of revcap
  activité |         1          2          3          4          5 |     Total
-----------+--------------------------------------------------------+----------
     actif |       191        326        355        459        547 |     1,878
    occupe |     382.0      323.5      315.7      395.2      461.5 |   1,878.0
-----------+--------------------------------------------------------+----------
   chomeur |        53          6          4          4          2 |        69
       BIT |      14.0       11.9       11.6       14.5       17.0 |      69.0
-----------+--------------------------------------------------------+----------
   chomeur |        15          7          1          3          4 |        30
 decourage |       6.1        5.2        5.0        6.3        7.4 |      30.0
-----------+--------------------------------------------------------+----------
   inactif |       231         76         45         41         39 |       432
           |      87.9       74.4       72.6       90.9      106.2 |     432.0
-----------+--------------------------------------------------------+----------
     Total |       490        415        405        507        592 |     2,409
           |     490.0      415.0      405.0      507.0      592.0 |   2,409.0

          Pearson chi2(12) = 597.1354   Pr = 0.000
```

L'hypothèse nulle du test du χ^2 est l'indépendance des deux variables. Ici on a moins de 1 chance sur 1000 de se tromper en rejetant H_0. Donc on la rejette. Les variables sont significativement dépendantes.

table

La commande table sert à créer des tableaux de statistiques descriptives. Ici, on crée une table contenant, pour chaque croisement de la situation dans l'activité et du genre, la moyenne du revenu individuel.

```
. table m3 sitac, contents (mean revind)
----------------------------------------------------------
          |             situation dans activité
    genre |        1         2         3         4         5
----------+-----------------------------------------------
    homme | 362.5652  150.6192  253.1582  229.9465  191.8583
    femme | 295.5595  177.4115  157.0455  240.8907  208.6397
----------------------------------------------------------
```

summarize

La commande summarize sert à calculer des statistiques descriptives d'une variable continue (vue précédemment).

mean

La commande mean calcule la moyenne d'une variable, l'erreur type et l'intervalle de confiance de l'estimation de la moyenne. L'option over permet de faire les calculs séparément sur des groupes :

```
. mean revind, over(m3)
Mean estimation                     Number of obs    =   12748

        homme: m3 = homme
        femme: m3 = femme

--------------------------------------------------------------
        Over |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
revind       |
       homme |   389.0121   21.99022      345.9079    432.1162
       femme |   121.5222   8.247381      105.3561    137.6883
--------------------------------------------------------------
```

total

La commande total calcule le total d'une variable

```
. total unite poids

Total estimation                    Number of obs    =   12749

--------------------------------------------------------------
             |      Total   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
       unite |      12749          0             .           .
       poids |    1142483    5766.12       1131181     1153786
--------------------------------------------------------------

. total unite [pw=poids]

Total estimation                    Number of obs    =   12749
--------------------------------------------------------------
             |      Total   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
       unite |    1142483          0             .           .
--------------------------------------------------------------
```

ratio

La commande ratio calcule le ratio entre deux variable

```
. ratio actif/PAT
Ratio estimation                    Number of obs    =   12749
     _ratio_1: actif/PAT
--------------------------------------------------------------
             |             Linearized
             |      Ratio   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
    _ratio_1 |   .5562444   .0052749      .5459048    .5665839
--------------------------------------------------------------
```

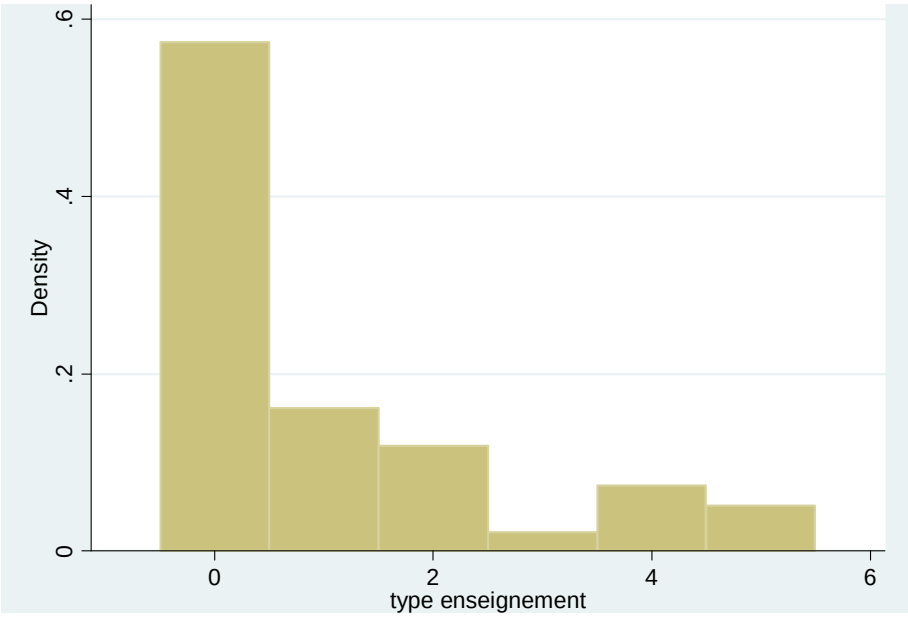
Cette commande donne le même résultat que

```
. mean actif if PAT==1
```

Les graphiques dans Stata

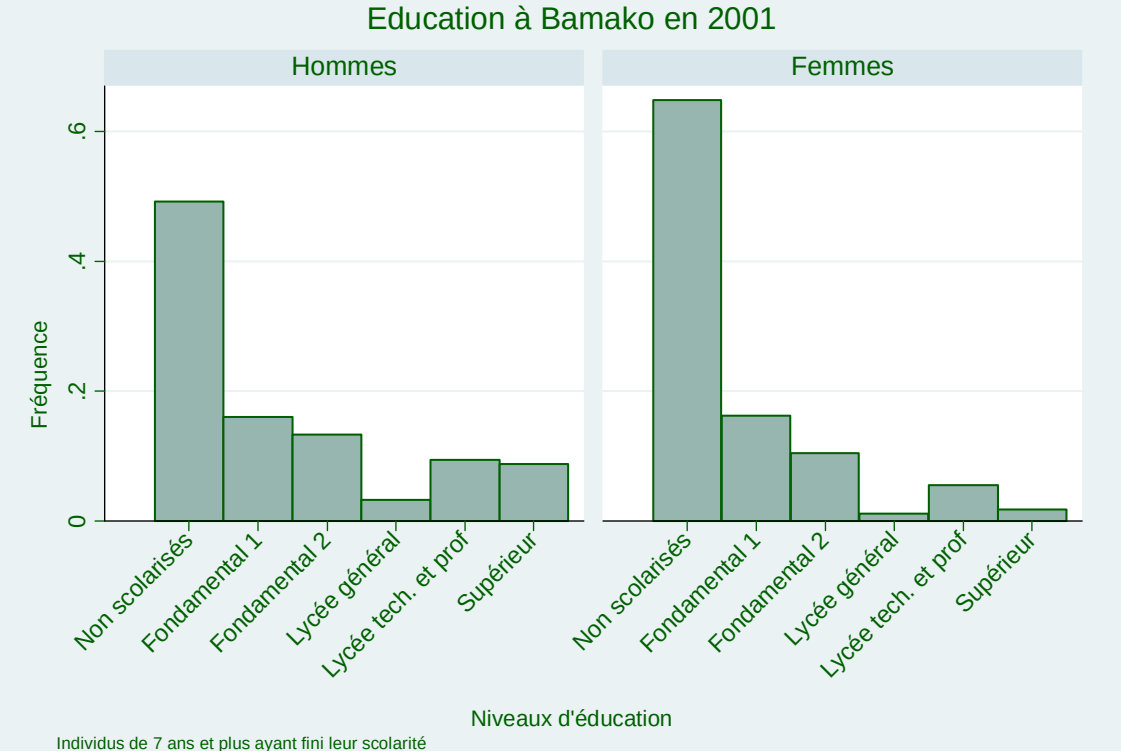
On présente ici quelques commandes permettant de dessiner des graphiques. Stata dispose d'un menu graphique qui évite de programmer les commandes graphiques et leurs options. Ce menu est très pratique car les options "esthétiques" sont très nombreuses (couleurs, police,…). Par exemple, la commande histogram représente l'histogramme d'une variable (catégorielle ou continue).

```
. histogram m14b if m17==2 & age>=7, discrete
```



Au prix de la commande ci-dessous, très longue et incompréhensible, on peut dessiner des graphiques très travaillés. Pour de tels graphiques, il est donc préférables d'utiliser le menu.

```
histogram m14b if age>=7 & m17==2, discrete fcolor(eltgreen) lcolor(dkgreen)/*
*/ ytitle(Fréquence, size(small) color(dkgreen)) ylabel(, tlcolor(dkgreen) /*
*/ labcolor(dkgreen) labsize(medsmall)) xtitle(Niveaux d'éducation, size(small) /*
*/ color(dkgreen)) xlabel( 0 "Non scolarisés" 1 "Fondamental 1" 2 "Fondamental 2"/*
*/ 3 "Lycée général" 4 "Lycée tech. et prof" 5 "Supérieur" , noticks labels /*
*/ valuelabel tlcolor(dkgreen) labcolor(dkgreen) angle(forty_five) /*
*/ labsize(medsmall)) title(, size(medium) color(dkgreen)) /*
*/ subtitle(,color(dkgreen)) by(femme, title(Education à Bamako en 2001, /*
*/ size(medium) color(dkgreen)) subtitle(, size(small) color(dkgreen)) /*
*/ note(Individus de 7 ans et plus ayant fini leur scolarité, /*
*/ size(vsmall)color(dkgreen)) legend(off))
```



La commande quantile représente les quantiles de la distribution d'une variable continue et ceux de la distribution uniforme.

La commande qnorm représente les quantiles de la distribution et les compare à ceux de la loi normale.

La commande graph box représente un box-plot

La commande scatterplot représente un nuage de points

La commande kdensity représente la densité de Kernel de la distribution d'une variable

La commande lowess lisse une distribution

La commande graph twoway représente une fonction quelconque définie par (x,y)

Sauver un graphique :

```
graph save "Distribution des revenus.gph", replace
```

Cours VII. La précision dans Stata : poids, plan de sondage et bootstrap

Il existe 4 types de poids dans Stata :

- **fweights**, pour frequency weights. Ils indiquent le nombre d'individus représentés par une observation. Ils doivent être entiers.
- **pweights**, les poids d'échantillonnage, ou coefficients d'extrapolation, inverses des probabilités d'inclusion.
- **aweights**, pour analytic weights. Ils s'utilisent quand les observations représentent des moyennes et le poids représente le nombre d'éléments ayant servi à calculer ces moyennes. Ils ne changent pas la valeur de la moyenne calculée, mais l'erreur-type, la précision, car il tient compte du fait que chaque observation est déjà issu d'une estimation.
- **iweights**, pour importance weights. Ils indiquent une importance relative des observations. Ils n'ont pas de définition statistique. Les commandes qui supportent les iweights définissent précisément le traitement fait. En fait ils sont utilisés par les programmeurs de commandes.

On précise le type de poids dans la commande :

[by *varlist***]** : **command [***varlist***] [=***exp***] [if** *exp***]** [weights] **[,***options***]**

```
. summarize age [aw=poids]

    Variable |     Obs      Weight        Mean   Std. Dev.       Min        Max
-------------+--------------------------------------------------------------------
         age |   12749  1142483.45    21.07993   16.97981          0         98


. mean age [pw=poids]

Mean estimation                     Number of obs    =   12749

--------------------------------------------------------------
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+------------------------------------------------
         age |   21.07993   .1703419      20.74604    21.41383
--------------------------------------------------------------


. summarize age [iw=poids]

    Variable |     Obs      Weight        Mean   Std. Dev.       Min        Max
-------------+--------------------------------------------------------------------
         age |   12749  1142483.45    21.07993   16.97915          0         98


. summarize revcap [fw=taille1] if CDM==1

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
      revcap |     12749     255.352    819.6166          0      60000
```

Tous les types de poids ne sont pas disponibles avec toutes les commandes. Pour connaître le type de poids supporté par la commande, il faut regarder dans le fichier d'aide.

L'utilisation des poids d'échantillonnage se fait à l'aide de pweight. Cependant un calcul sans biais de la précision des indicateurs calculés nécessite d'utiliser d'autres informations du plan d'échantillonnage : les strates, les unités primaires de sondage dans un sondage à deux degrés,…. Il faut donc donner toutes ces informations à Stata. Le plan d'échantillonnage se définit par la commande svyset :

```
svyset ZD [pw=poids], strata(strate)
```

Une fois le plan de sondage défini, le préfixe svy permet d'effectuer des estimations en estimant correctement la précision : `svy : mean, svy : total, svy : ratio, svy : regress`

La variance des totaux, moyenne et autres statistiques peut être estimée à l'aide de plusieursm éthodes : la linéarisation de la variance, le bootstrap, le jackknife, la méthode des BRR (Balanced Repeated Replications). Il est possible de décrire des plans de sondage très complexes, comme à 5 degrés, stratifiés à tous les degrés…Plusieurs exemples sont présentés dans le fichier d'aide de svyset.

Pour effacer le plan de sondage :

```
svyset, clear
```

Lorsqu'on dispose de peu d'information sur le plan de sondage (ou dans le cas de modèles économétriques, qu'on doute de l'homoscédasticité du modèle ou qu'on suppose un mauvais calcul des erreurs-type) la méthode de bootstrap peut être utile. Largement répandue aujourd'hui, elle a l'avantage de suivre un principe "universel". Le bootstrap consiste à tirer avec remise *R* échantillons de même taille que l'échantillon initial. L'estimateur, quel qu'il soit, est calculé à chaque itération. La précision de l'estimateur bootstrapé est estimée par la variance des R estimateurs.

Le préfixe bootstrap permet de faire des réplications bootstrap d'une estimation. Par défaut, Stata effectue 50 réplications. Ici, on lui demande d'en faire 100.

```
. bootstrap, reps(100) : mean age
(running mean on estimation sample)

Bootstrap replications (100)
----+--- 1 ---+--- 2 ---+--- 3 ---+--- 4 ---+--- 5
..................................................    50
..................................................   100

Mean estimation                   Number of obs    =   12749
                                  Replications     =     100

------------------------------------------------------------
             |   Observed   Bootstrap         Normal-based
             |       Mean   Std. Err.     [95% Conf. Interval]
-------------+----------------------------------------------
         age |   21.69896   .1716595      21.36251     22.0354
------------------------------------------------------------
```

Cours VIII. Le modèle linéaire dans Stata

Supposons qu'on étudie les déterminants du revenu. On commence par étudier des corrélations. La commande correlate calcule le coefficient de corrélation entre deux variables

```
. correlate age revind etudes
(obs=12748)

             |      age   revind   etudes
-------------+---------------------------
         age |   1.0000
      revind |   0.2109   1.0000
      etudes |   0.2544   0.1598   1.0000
```

Il apparaît que le revenu individuel (activités principale et secondaire et autres revenus) est corrélé positivement à l'âge et au nombre d'années d'études réussies. Pour savoir si cette corrélation est significative, on peut procéder à un test de nullité du coefficient de la corrélation, à l'aide de la commande pwcorr et de son option sig :

```
. pwcorr age revind etudes, sig

             |      age   revind   etudes
-------------+---------------------------
         age |   1.0000
             |
      revind |   0.2109   1.0000
             |   0.0000
             |
      etudes |   0.2541   0.1598   1.0000
             |   0.0000   0.0000
```

Les corrélations (revenu, âge) et (revenu, études) sont toutes les deux significatives. (La corrélation entre l'âge et les études aussi). On à moins de 1 chance sur 1000 de se tromper en affirmant que le coefficient de corrélation est différent de 0. On dit que le coefficient de corrélation entre le revenu et le nombre d'années d'études est de 25,4% et qu'il est significatif au seuil de 1‰.

significativité = 0.000	significatif au seuil de 1‰
significativité < 0.010	significatif au seuil de 1%
significativité < 0.050	significatif au seuil de 5%
significativité < 0.100	significatif au seuil de 10%

Le revenu dépend de plusieurs variables en même temps. La corrélation nous permet de tester les varaibles explicatives une par une, mais pas ensemble. Le modèle linéaire, ou modèle des moindres carrés ordinaires permet d'estimer l'effet d'un facteur sur la variable d'intérêt, "toute chose égale par ailleurs", c'est-à-dire en maintenant les autres variables constantes, à leur moyenne pour les varaibles continues, à

leur valeur dite de référence pour les variables catégorielles. Il est préférable de choisir la carégorie la plus représentée come catégorie de référence.

Pour pouvoir assumer que les coefficients de la MCO sont non-biaisés, c'est-à-dire que la valeur prédite par l'estimateur converge vers la valeur dans la population, on doit faire l'hypothèse que les 4 conditions suivantes sont respectées dans notre échantillon :

1. Les paramètres suivent une fonction linéaires : $y = \beta 0 + \beta 1x + u$

2. L'échantillon est identiquement et indépendamment distribué (iid).

3. L'espérance du terme d'erreur sachant x est égale à zéro. $E(u/x) = 0$

4. Pas de multicolinéarité exacte entre les varaibles explicatives x.

La commande regress permet d'estimer un modçle par les moindres carrés ordinaires. Ici on cherche à expliquer le revenu de l'activité principale par le genre, l'âge et son carré, et le niveau d'études. ON a choisi le niveau "sans éducation" comme modalité de référence.

```
. regress lrevAPh femme age age2 nived2 nived3 nived4 nived5 nived6

      Source |       SS       df       MS              Number of obs =    3876
-------------+------------------------------           F(  8,  3867) =  219.16
       Model |   1616.0648     8    202.0081           Prob > F      =  0.0000
    Residual |  3564.32398  3867  .921728466           R-squared     =  0.3120
-------------+------------------------------           Adj R-squared =  0.3105
       Total |  5180.38877  3875  1.33687452           Root MSE      =  .96007

------------------------------------------------------------------------------
     lrevAPh |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       femme |  -.3826071   .0321178   -11.91   0.000    -.4455765   -.3196378
         age |   .0958512   .0053997    17.75   0.000     .0852647    .1064378
        age2 |  -.0009373   .0000689   -13.59   0.000    -.0010724   -.0008021
      nived2 |   .0557579   .0427996     1.30   0.193    -.0281541    .1396699
      nived3 |   .2882051   .0465887     6.19   0.000     .1968644    .3795458
      nived4 |    .525106   .0889622     5.90   0.000     .3506888    .6995233
      nived5 |    .837949   .0573946    14.60   0.000     .7254225    .9504756
      nived6 |   1.203702   .0653987    18.41   0.000     1.075483    1.331921
       _cons |  -3.905688   .1016011   -38.44   0.000    -4.104885   -3.706491
------------------------------------------------------------------------------
```

Stata retourne les coefficients estimés, leur erreur standard et l'intervalle de confiance (en bleu). Les colonnes t et P>|t|, en vert, représentent les résultats des tests de nullité des coefficients (test de Student).

H0 : le coefficient est nul.

Pour la variable nived2 (niveau fondamental 1), P>|t|=0.193, il y a 19% de chance de se tromper en rejetant H0. Donc on ne la rejette pas. On dit alors que le coefficient de nived2 n'est pas significativement différent de 0, ou encore plus simplement qu'il n'est pas significatif.

Généralement, on établit 3 seuils en dessous desquels on considère qu'un coefficient est significatif : 1% d'erreur, 5% d'erreur et 10% d'erreur. Tous les autres coefficients sont significatifs au seuil de 1%

P>|t|=0.000	significatif au seuil de 1‰
P>|t|<0.010	significatif au seuil de 1%
P>|t|<0.050	significatif au seuil de 5%
P>|t|<0.100	significatif au seuil de 10%

Le R^2 (en rouge) représente la part de la variabilité du revenu espliquée par le modèle. Ici, 31,2% de la varaibilité du revenu est expliquée par le genre, l'âge et les études. Cette mesure peut augmenter

artificiellement lorsqu'on ajoute des varaibles explicatives, mais si celle-ci n'explique pas la variable dépendante. Aussi, il est d'usage de lui préférer le R^2 ajusté. Ici ils sont proches car il y a peu de variables explicatives dans notre modèle.

La commande outreg2 permet de créer un fichier de résultat qu'on pourra ouvrir dans Word.

```
outreg2 using "MCO.doc", bdec(3,3,3) adec(3) nor2 addstat(R2a,e(r2_a))
```

L'option bdec permet de préciser le nombre de décimales de coeffiients, l'option nor2 précise qu'on ne veut pas que le R2 apparaisse, l'option addstat permet de demander que d'autres statistiques apparaissent, ici le R^2 ajusté. L'option adec précise le nombre de décimales des statistiques supplémentaires, donc ici du R^2 ajusté. On obtient alors aisément un tableau de la forme (en utilisant la fonction convertir un texte en tableau dans Word) :

	lrevAPh
femme	-0.383***
	(0.032)
age	0.096***
	(0.005)
age2	-0.001***
	(0.000)
nived2	0.056
	(0.043)
nived3	0.288***
	(0.047)
nived4	0.525***
	(0.089)
nived5	0.838***
	(0.057)
nived6	1.204***
	(0.065)
Constant	-3.906***
	(0.102)
Observations	3876
R2a	0.311

Standard errors in parentheses
*** p<0.01, ** p<0.05, * p<0.1

Annexe 1. Quelques fonctions utiles de Stata

Quelques fonctions

Fonctions mathématiques

abs(x) retourne la valeur absolue de x.

exp(x) retourne l'exponentielle de x.

log(x) retourne le logarithme de x si x>0.

max(x1,x2,...,xn) retourne le maximum de x1, x2, ..., xn.

min(x1,x2,...,xn) retourne le minimum de x1, x2, ..., xn.

round(x,y) retourne x arrondi à l'unité de y.

sqrt(x) retourne la racine carrée de x si x > 0.

sum(x) retourne la somme cumulée de x, traitant les valeurs manquantes comme des 0.

Distributions et densités

binomial(n,k,p) retourne la probabilité d'au moins k succès en n tirages avec une probabilité de succès à chaque tirage de p.

normal(z) retourne la fonction de répartition de la loi normale standard.

normalden(z) retourne la fonction de densité de la loi normale standard.

Générations de nombres aléatoires

uniform( ) retourne des nombres uniformément distribué sur l'intervalle [0,1].

invnormal(uniform( ))retourne des nombres normalement distribué, de moyenne 0 et d'écart-type 1. Ces fonctions n'ont pas d'arguments mais il est nécessaire de mettre les parenthèses.

Chaîne de caractères

length(s) retourne la longueur de la chaine de caratère s.

string(n) convertir n en une chaîne de caractère

substr(s,n1,n2) retourne une partie d'une chaîne de caractère s. Elle extrait la partie allant du n1ème caractère au (n1+n2)ème caractère inclus. Si n2 n'est pas renseigné, elle retourne tous les caractères se trouvant à droite de n1, n1 compris.

```
generate s2=substr("CSPouvrier", 4, 7)
generate s3=substr("CSPouvrier", 4)
```

Alors s2=s3="ouvrier"

Fonctions utiles en programmation

cond(x,a,b) retourne a si la condition x est vérifier, b si elle ne l'est pas

```
generate femme=1 if m3==2
replace  femme=0 if m3==1
```

peut s'écrire en une ligne

```
generate femme=cond(m3==2,1,0)
```

inlist(z,a,b,...) retourne la valeur 1 (la valeur vraie) si z appartient à lal iste des autres arguments de la fonction (a, b, …). Sinon elle retourne 0. Le nombre d'arguments peut être compris entre 2 et 255 pour des réels et entre 2 et 10 pour des chaînes de caractères.

inrange(z,a,b) retourne la valeur 1 (vraie) si a < z < b, 0 sinon. Elle retourne 0 si z est manquant

r(name) contient la valeur d'un résultat sauvé

```
. summarize revcap, detail
                          revcap
-------------------------------------------------------------
      Percentiles      Smallest
 1%            0              0
 5%            0              0
10%     17.45455              0       Obs                12749
25%     56.57143              0       Sum of Wgt.        12749

50%          132                      Mean            255.352
                      Largest         Std. Dev.       819.6166
75%        250.6         18484
90%       477.72         18484       Variance        671771.4
95%          741         27864       Skewness        38.65029
99%         2160         60000       Kurtosis        2411.216

. gen ligne2=0.5*r(p50)
```

Pour connaître la liste des résultats de l commandes, on tape :

```
return list
```

e(sample) retourne la valeur 1 si l'observation a été utilisée dans l'échantillon d'estimation de la commande précédente.

```
. regress lrevAPh femme age age2

      Source |       SS       df       MS              Number of obs =    3876
-------------+------------------------------           F(  3,  3872) =  366.78
       Model |  1146.38086     3  382.126953            Prob > F      =  0.0000
    Residual |  4034.00791  3872  1.04184089            R-squared     =  0.2213
-------------+------------------------------           Adj R-squared =  0.2207
       Total |  5180.38877  3875  1.33687452            Root MSE      =  1.0207

------------------------------------------------------------------------------
     lrevAPh |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
       femme |  -.4861283   .0336211   -14.46   0.000    -.5520451   -.4202116
         age |   .1177752   .0056444    20.87   0.000     .1067089    .1288415
        age2 |  -.0011868   .0000722   -16.43   0.000    -.0013284   -.0010452
       _cons |  -4.055482   .1065508   -38.06   0.000    -4.264383   -3.846582
------------------------------------------------------------------------------

. gen R2=e(r2)
. display R2
.22129244
```

Pour connaître la liste des résultats d'estimation, on tape :

```
ereturn list
```