Manuel d'introduction à Stata

Simon Briole

Manuel d'introduction à Stata

Simon Briole*

Mis à jour le : 15/01/2024

*Centre d'Économie de l'Environnement - Montpellier, Université de Montpellier & Paris School of Economics. Contact: simon.briole@umontpellier.fr

Ce manuel d'introduction à Stata s'est notamment appuyé sur l'*Introduction au logiciel Stata* d'Antoine Bozio ainsi que sur la *Short introduction to Stata* de Marion Leturcq. Je remercie également chaleureusement Hélène Le Forner et Sarah Flèche d'avoir accepté de partager leurs ressources pédagogiques sur Stata.

Table des matières

1 Présentation du logiciel

1.1 L'environnement de Stata

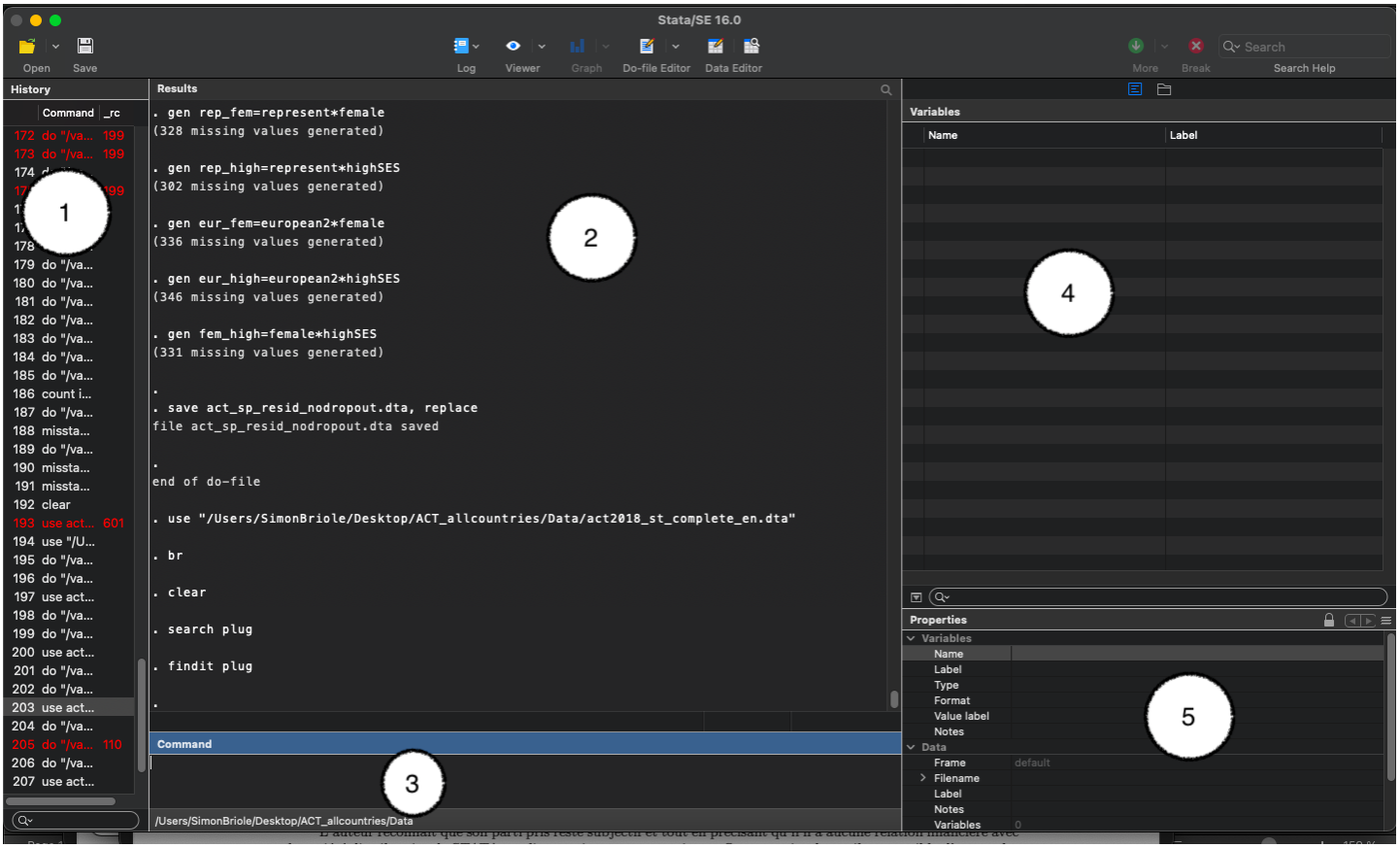
L'environnement de Stata comprend 5 fenêtres principales, visibles en permanence (voir Figure 1 ci-dessous)[1] :

1. La fenêtre de Commandes passées (*History* ou *Review*): récapitule les commandes soumises pendant toute la session ouverte de Stata
2. La fenêtre de Résultats (*Results*) : affiche les commandes soumises et les résultats des opérations effectués par Stata
3. La fenêtre de Commandes (*Command*): permet de taper des commandes qui peuvent être exécutées immédiatement au moyen de la touche « Entrée »
4. La fenêtre de Variables (*Variables*): liste les variables de la base des données avec leur « label »
5. La fenêtre de Propriétés (*Properties*): permet d'obtenir plus de détails sur la base de données et les variables

A ces 5 fenêtres principales s'ajoutent les fenêtres suivantes, qui n'apparaissent que lorsqu'une action précise déclenche leur ouverture :

- La fenêtre *Viewer* : apparaît lors d'une demande d'aide ou lors de la visualisation des fichiers « log », qui enregistrent les commandes soumises et les résultats obtenus (voir plus bas)
- La fenêtre *Do-file Editor* : éditeur de fichier de codes (les *do-files*) de Stata
- Les fenêtres *Data Editor* ou *Data Browser* : il s'agit de l'éditeur de données ; le premier permet l'observation et la modification des données alors que le deuxième permet seulement l'observation des données mais pas leur modification
- La fenêtre *Variables Manager* : permet de créer et gérer les variables (noms, labels, formats, ...)
- La fenêtre *Graph Editor* : permet d'éditer manuellement les graphiques générés par Stata

Figure 1: Environnement Stata



La barre d'outils de Stata (voir Figure 2) permet de gérer facilement et rapidement les actions de base. De gauche à droite les icones représentent: ouverture d'un fichier de données Stata, sauvegarde du fichier, création/ouverture d'un fichier log, affichage de la fenêtre Viewer, affichage du dernier graphique commandé, ouverture de la fenêtre Do-file Editor, ouverture des fenêtres du Data Editor et du Data Browser.

Figure 2: Barre d'outils de Stata

[1]Notez que l'apparence de cet environnement peut changer d'une version à l'autre, selon que l'on utilise Stata sur Windows ou Mac, mais le contenu des fenêtres reste identique.

1.2 Les différents types de fichier

Il existe 4 principaux types de fichier gérés par Stata, chacun d'entre eux correspondant à un suffixe spécifique :

1. Les fichiers de données au format Stata (identifiés par le suffixe .dta),
2. Les fichiers de code/programmation appelés ***do-files*** (identifiés par le suffixe .do). Ils sont générés par l'éditeur de texte (*Do-file Editor*) et contiennent un ensemble de commandes et de commentaires.
3. Les fichier de précédures *ado-files* (identifiés par le suffixe .ado). Ces fichiers – fournis ou à télécharger-, contiennent des routines écrites dans le langage Stata et permettent de réaliser des traitements spécifiques (ex: exécution d'une commande)
4. Les *log-files*, fichiers qui servent à enregistrer les résultats du travail effectué dans Stata au cours d'une session (identifiés par le suffixe .smcl, mais il est possible de changer le format des fichiers vers un format texte, par exemple, soit en identifiant le fichier par le suffixe .log, soit à l'aide de la commande `translate`).

En plus de ces 4 principaux types de fichiers Stata, deux autres types méritent d'être brièvement mentionnés : les fichiers d'aide en ligne (.hlp) ou les graphiques (.gph).

À son installation, Stata crée de façon automatique un répertoire appelé ≪ado ≫ dans C: (avec des sous-repertoires ≪ personal ≫ et ≪ plus ≫). Ce répertoire n'est que la bibliothèque des procédures (les .ado), celles qui sont mises à jour régulièrement, ayant été corrigées, homogénéisées ou complétées par le groupe industriel StataCorp (aidés par la communauté des utilisateurs), et celles crées où installées par les utilisateurs (vous). Lorsque vous installez Stata sur votre ordinateur, un certain nombre de procédures sont déjà installées et vous permettent de réaliser les commandes de base. Néanmoins, certaines procédures correspondant à des commandes plus récentes peuvent ne pas être installées. Dans ce cas, la commande ne s'exécutera pas et Stata affichera un message d'erreur. Pour pouvoir exécuter la commande, il est nécessaire de télécharge un package (voir encadré ci-dessous).

Encadré 1 : mise à jour et installation de nouvelles procédures avec les *packages* de Stata

Pour exécuter une nouvelle commande qui est indisponible sur votre répertoire, il faut installer le package correspondant à cette commande, qui contient le ou les ado-files nécessaires à l'exécution de cette commande ainsi que les fichiers d'aides associés à la commande. Ces packages sont disponibles sur l'archive SSC (Boston College Statistical Software Components). Pour installer un package, il suffit d'exécuter la commande suivante :

```
ssc install newcommand
```

où *newcommand* doit être remplacé par le nom de la nouvelle commande que vous souhaiter exécuter.

1.3 Organiser son travail sur Stata

1.3.1 Structurer ses répertoires

Pour bien organiser son travail sur Stata, il est impératif de créer un ensemble structuré de répertoires, contenant de manière séparée les données, les fichiers de programmation (do-files), et les résultats des analyses. Ci-dessous, vous trouverez un exemple de structuration des répertoire[2] :

Répertoire de base :

/Users/SimonBriole/Desktop/Cours_STATA_UM

Les bases de données sont stockées dans le répertoire :

/Users/SimonBriole/Desktop/Cours_STATA_UM/data

Les programmes que nous allons rédiger seront stockés dans :

/Users/SimonBriole/Desktop/Cours_STATA_UM/dofiles

Et nos résultats (graphiques, logfiles, ...) dans :

/Users/SimonBriole/Desktop/Cours_STATA_UM/results

1.3.2 Sauvegarder son travail avec les do-files

Pour travailler sous Stata, il est impératif d'utiliser des *do-files*. Ces fichiers permette d'écrire un programme que l'on peut sauvegarder et réutiliser la fois suivante. Il sont donc nécessaires afin d'assurer la réplicabilité des analyses réalisées et pour ne pas repartir à zéro d'une session de travail à l'autre.

[2]Note : cet exemple est rédigé à partir d'une utilisation de Stata sur Mac. Sur Windows, les / sont à remplacer par des \ pour les chemins d'acccès

L'utilisation de la fenêtre *Command* sert plutôt à tester rapidement des variantes ou à s'exercer au début avec les commandes.

Créer un do-file

Pour créer un do-file, il faut utiliser l'éditeur intégré de Stata, le *Do-file Editor*. Pour cela, vous pouvez soit cliquer sur l'icône Do-File Editor de la barre d'outils (voir Figure 2), soit utiliser le menu déroulant (File → New → Do-file). Un nouveau fichier s'ouvre alors dans une fenêtre séparée, dédiée aux do-files.

Structurer et rédiger un do-file

Un do-file doit être clair afin de pouvoir être utilisé longtemps après sa réalisation ou par des tiers. Pour cela il est utile d'avoir en mémoire quelques bons réflexes de programmation. En voici quelques uns :

1. Préciser le répertoire de travail. Pour commencer un do file il faut préciser dans quel dossier/répertoire on travaille, à l'aide de la commande `cd` ou avec l'aide du menu déroulant (File → Change working directory, puis choisir le dossier). La syntaxe de la commande `cd` est la suivante :

   ```
   cd "/Users/SimonBriole/Desktop/Cours_STATA_UM"
   ```

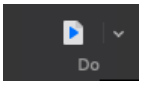
2. Préciser les dates de création et mise à jour du do-file : Indiquer la date de création du do-file ainsi que toutes les dates de révision, avec éventuellement un bref descriptif des changements apportés.

3. Commenter ses codes : Stata traite chaque ligne qui commence par un astérisque (*) comme un commentaire. Vous pouvez écrire des commentaires aussi avec 2 slashes (//) au début ou à la fin d'une ligne (toujours précédé d'un espace), ou s'ils occupent plusieurs lignes, en plaçant une barre oblique et un astérisque (/*) au début de ce commentaire et un astérisque et une barre oblique à la fin (*/).

Exécuter un do-file

Une fois les commandes saisies dans l'éditeur, elles peuvent être exécutées en les sélectionnant et en utilisant l'icône de l'éditeur, dernier icône sur la barre des menus, correspondant à la commande « Execute (do) » (voir Figure 3).

Figure 3: Exécuter un do-file



Stata execute les commandes ligne par ligne et chaque passage à la ligne correspond à un nouvel ensemble de commandes. Si vous voulez rédiger un ensemble de commandes de longues lignes (par exemple lister les nombreuses variables que vous voulez conserver dans l'enquête) vous pouvez le répartir sur plusieurs lignes en écrivant simplement 3 slashes /// (précédées d'un espace) en fin de ligne et en ne mettant rien en début de ligne.

Encadré 2 : Un exemple de début de do-file

```
/* COURS STATA UNIVERSITÉ DE MONTPELLIER - EXEMPLE DE DO-FILE
Auteur: Simon Briole (SB)
Derniere mise à jour: 24/01/2021 (SB)
Description du do-file:
Ce dofile nous montre comment commencer un fichier .do et quelquescommandes de base de STATA.
Afin de rendre les programmes lisibles, il est important de lescommenter, sans abuser des commentaires afin de ne pas nuire à la lisibilité */

* DEBUT :
   clear all
   set more off

* REPERTOIRE DE TRAVAIL :
   cd "/Users/SimonBriole/Desktop/Cours_STATA_UM"

* OUVERTURE BASE DONNEES :
   use ./data/com_info.dta, clear
/* NB : Une fois qu'on a indiqué à Stata le chemin vers notre répertoire de travail (grâce à la commande cd), au lieu de devoir le réécrire à chaque fois, il est possible de résumer et substituer le chemin connu du répertoire par un point */

/* SAUVEGARDE DONNEES : Afin de ne pas modifier les données d'origine, il est impératif de sauvegarder le fichier de données sous un autre nom avant de commencer à modifier la base de données */
   save ./data/com_info.dta, replace
```

1.4 Où trouver de l'aide ?

Il existe différents manuels Stata officiels disponibles en ligne, où l'on peut trouver de l'aide sur l'utilisation du logiciel ainsi qu'une description détaillée des commandes existantes (cf. Appendix A). Si vous souhaitez obtenir des informations sur une commande, la façon la plus rapide de faire est de lancer la commande `help` suivi du nom de la commande qui vous intéresse :

```
help mean
```

Lorsque vous ignorez le nom précis de la commande, vous pouvez toujours utiliser la commande `search` suivie d'un mot-clé en anglais qui précise ce que l'on cherche, pour trouver des suggestions de commandes (ou des programmes Stata et sites référencés sur internet)[3] :

```
search word [word ...]  [,all | local | net author entry exact faq historical or manual
sj ]
```

[3] Si vous utilisez une version équivalente ou ultérieure à Stata 15, les commandes `findit` et `net search` sont équivalentes aux options `[,all]` et `[,net]` de la commande `search`.

2 Gérer une base de données

2.1 Ouvrir, importer et sauvegarder une base de données

*2.1.A Ouvrir une base de données au format Stata (.dta) : la commande **use***

Si vous disposez de données sous format Stata (.dta) alors vous pouvez ouvrir votre base de données en saisissant la commande `use` dans la fenêtre *Stata Command*, ou dans votre dofile :

```
use /Users/SimonBriole/Desktop/Cours_STATA_UM/data/com_info.dta, clear
```

Une autre option est de se placer au préalable dans le répertoire choisi (avec la commande cd) puis d'ouvrir la base dans un second temps :

```
cd /Users/SimonBriole/Desktop/Cours_STATA_UM/data
use com_info.dta, clear
```

Vous pouvez également cliquer sur l'icône Open de la barre d'outils et rechercher le fichier dont vous avez besoin ou bien utiliser le menu déroulant (File → Open).

> Encadré 3 : les options des commandes
>
> Les options liées aux commandes de Stata sont généralement situées après les commandes et après une virgule. Les fichiers d'aide liés à une commande donnée précise généralement les options possibles pour cette commande. Dans l'exemple ci-dessus avec la commande `use`, l'option `clear` permet d'ouvrir un nouveau fichier de données lorsqu'un autre fichier est déjà ouvert.

NB : Stata met à disposition des exemples de bases de données disponibles sur internet pour pouvoir expliquer les différentes commandes. Elles sont accessibles à travers la commande `webuse`.

*2.1.B Importer des données dans un autre format : les commandes **import** et **infile***

Plusieurs commandes sont disponibles afin d'importer des données provenant d'autres sources en fonction de leur format[4] :

- La commande `import excel` permet d'importer des fichiers de données au format Excel. L'option firstrow après virgule permet de traiter la première ligne du fichier Excel comme contenant les noms ou les labels des variables. Exemple :

[4] Le séparateur des décimales sous Stata est le point (.). Avant l'importation d'un fichier, il faudra penser à transformer toutes les virgules en points dans le fichier Excel (sinon les variables avec des virgules seront lues comme des variables textuelles –*string*- par Stata)

```
import excel /Users/SimonBriole/Desktop/Cours_STATA_UM/data/com_info.xlsx, sheet("Sheet1")
firstrow clear
```

Stata peut importer des données des formats autres que Excel (voir par exemple les commandes `import sasxport, import dbase, spshape2dta`, etc.).

- La commande `import delimited` permet d'importer des fichiers de données au format texte (ASCII), dans lesquelles il n'y a qu'une seule observation par ligne et où les variables sont séparées par des tabulations ou des virgules. Exemple :

```
import delimited /Users/SimonBriole/Desktop/Cours_STATA_UM/data/com_info.txt, clear
import delimited /Users/SimonBriole/Desktop/Cours_STATA_UM/data/com_info.csv, clear
```

- La commande `infile` permet d'importer des fichiers de données au format texte plus complexes, dans lesquels il peut y avoir plusieurs observations par ligne et où les variables sont séparées par des tabulations, des virgules ou des espaces. Elle est la commande la plus évoluée pour importer des données mais aussi la plus complexe à manipuler.

2.1.C Sauvegarder une base de données

Si vous effectuez des changements dans un fichier de données STATA ouvert et que vous souhaitez sauvegarder ces changements, vous pouvez utiliser la commande `save` :

```
save /Users/SimonBriole/Desktop/Cours_STATA_UM/data/exemple.dta
```

Si vous êtes déjà dans le répertoire de travail, il suffit d'écrire :

```
save exemple.dta
```

Si vous souhaitez écraser une base de données existante, il faut ajouter l'option `replace` :

```
save exemple.dta, replace
```

⚠**Attention** ⚠: pour préserver les données initiales, on n'oublie pas de donner un nom différent à la base sur laquelle on va effectuer les traitements de celui de la base initiale.

2.1.D Le logiciel Stat Transfert

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

2.1.E Problème de mémoire insuffisante

Si vous n'avez pas assez de mémoire vive disponible pour Stata (le message d'erreur est : no room for more observations) alors il faut utiliser la commande `set memory XXm` pour préciser le nombre de megabytes[5] que vous souhaitez allouer à Stata.

2.2 Analyser et modifier la structure d'une base de données

2.2.1 Visualiser et éditer le jeu de données : *browse* et *edit*

Une fois votre base de données ouverte, il peut s'avérer utile de visualiser les données. Pour ce faire, il suffit d'exécuter la commande `browse`, qui va ouvrir la fenêtre *Data Browser*. Notez qu'il est possible d'ouvrir cette fenêtre en demandant à Stata de ne montrer que certaines variables. Pour cela, il suffit d'écrire la commande `browse` suivie des variables que vous souhaitez voir (ex: `browse pop1999`).

Si vous souhaitez éditer les données manuellement, vous pouvez ouvrir le *Data Editor* (la même fenêtre que le *Data Browser*, mais avec des droits de modification des données) en exécutant la commande `edit`.

Encadré 4 : les données sur Stata

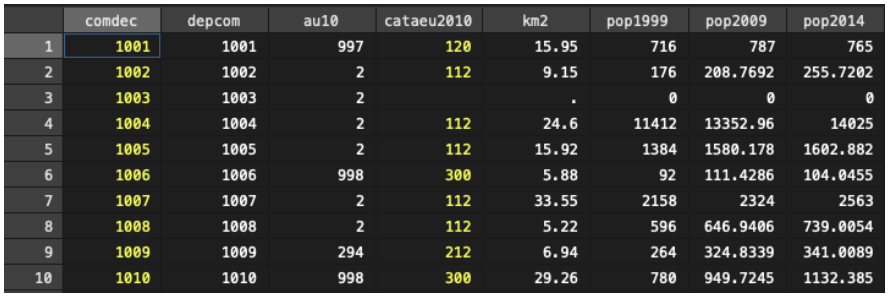
Les données sur Stata apparaissent sous la forme d'une matrice correspondant aux variables (colonnes) et aux observations (lignes). Par exemple, pour la base *com_info*, les colonnes peuvent correspondre à la population en 1999 (*pop1999*) et à la superficie des communes (*km2*), et les lignes aux communes présentes dans la base. Les cellules correspondent donc à la valeur d'une variable précise pour un individu (dans notre exemple, une commune) précis, comme dans la Figure 4. Il existe deux types principaux de données:

1. **Numérique**. Ex : age, année, superficie, etc.
2. **Texte**. Ex : nom d'une commune, identifiant personnel, etc. Ces données sont encadrées par des guillemets (") au début et à la fin de chaque cellule.

⚠ NB: pour les variables numériques, les valeurs manquantes sont indiquées par un point (.) alors que pour les variables textes elles sont indiquées soit par des guillemets vides (" ") soit par un point entre guillemets (".")

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

Figure 4: Visualisation des données



	comdec	depcom	au10	cataeu2010	km2	pop1999	pop2009	pop2014
1	1001	1001	997	120	15.95	716	787	765
2	1002	1002	2	112	9.15	176	208.7692	255.7202
3	1003	1003	2		.	0	0	0
4	1004	1004	2	112	24.6	11412	13352.96	14025
5	1005	1005	2	112	15.92	1384	1580.178	1602.882
6	1006	1006	998	300	5.88	92	111.4286	104.0455
7	1007	1007	2	112	33.55	2158	2324	2563
8	1008	1008	2	112	5.22	596	646.9406	739.0054
9	1009	1009	294	212	6.94	264	324.8339	341.0089
10	1010	1010	998	300	29.26	780	949.7245	1132.385

2.2.2 Décrire les variables et leur contenu

Pour visualiser toutes les variables d'une base de données, vous pouvez utiliser la commande `describe`. Pour visualiser seulement une variable ou une liste de variables, utilisez la commande `describe` ou `codebook` suivie du nom des variables :

```
describe
codebook pop1999
```

La commande `count` permet de connaître le nombre d'observations dans la base de données. Par défaut, elle indique le nombre total d'observations, mais si vous utlisez une condition, elle permet de connaître le nombre d'observations *vérifiant cette condition*. Par exemple :

`count` (indique le nombre total d'observations)

`count if au10==2` (indique le nb d'observations pour lesquelles la variable *au10* est égale à 2).

Afin de décrire les données (valeurs) contenues dans une ou plusieurs variables, vous pouvez utiliser la commande `list`, suivie du nom des variables :

```
list pop1999
```

2.2.3 Réorganiser les observations : *sort* et *gsort*

La commande `sort` permet de classer les observations de manière croissante, en fonction de la valeur prise par une ou plusieurs variables. Pour ce faire, il faut donc préciser sur quelle(s) variable(s) on veut réaliser le classement :

```
sort pop1999 km2
```

La commande ci-dessus va classer les observations de la plus petite à la plus grande valeur de *pop1999*. Lorsque plusieurs observations ont la même valeur de *pop1999*, alors ces observations vont être classées

de la plus petite à la plus grande valeur de *km2*.

La commande `gsort` permet de trier les variables soit par ordre ascendant (nom de la variable précédé du signe +), soit par ordre descendant (nom de la variable précédé du signe -) :

```
gsort +pop1999 -km2
```

La commande ci-dessus va classer les observations de la plus petite à la plus grande valeur de pop1999. Lorsque plusieurs observations ont la même valeur de *pop1999*, alors ces observations vont être classées de la plus grande à la plus petite valeur de *km2*.

2.2.4 Formats wide et long et transposition des données : *reshape*

Si vous avez un jeu de données qui contient une variable x pour les individus (i) au cours de plusieurs périodes (t), il est possible d'organiser ces données de telle sorte que chaque ligne correspond à l'ensemble des observations pour un individu (format *wide*), ou bien de telle sorte que chaque ligne correspond à un individu i au cours d'une date t (format *long*). Dans le format *long*, vous aurez donc $i \times t$ lignes et une seule colonne par ligne qui correspond à la variable x ; dans le format *wide* vous aurez une ligne par individu i mais $x \times t$ colonnes. La Figure 7 ci-dessous illustre ces deux formats.

Figure 5: Données *wide* (gauche) et *long* (droite)

	comdec	pop1999	pop2009	pop2014
1	1001	716	787	765
2	1002	176	208.7692	255.7202
3	1003	0	0	0
4	1004	11412	13352.96	14025

	comdec	annee	pop
1	10001	1999	0
2	10001	2009	0
3	10001	2014	0
4	10002	1999	244
5	10002	2009	284
6	10002	2014	265
7	10003	1999	2180
8	10003	2009	2417.92
9	10003	2014	3587.623
10	10004	1999	276
11	10004	2009	241.1206
12	10004	2014	242.0708

Pour passer d'un format à l'autre, il faut utiliser la commande `reshape`. Par exemple, dans le fichier ci-dessus :

```
reshape long pop, i(comdec) j(annee) /*pour passer du wide au long*/
reshape wide pop, i(comdec) j(annee) /*pour passer du long au wide*/
```

NB : si vous passez d'un format *wide* au format *long* (ou inversement), il est toujours possible de revenir au premier format avec la commande complémentaire.

2.2.5 Transformer la base : *collapse*

La commande `collapse` transforme la base de données en mémoire en statistiques essentielles sur celle-ci (`sum`=somme ; `mean`=moyenne ; `sd`=écart-type; `median`=médiane). Par exemple, si votre base de données comprend le salaire d'individus répartis dans plusieurs pays, vous pouvez transformer votre base individuelle en une base "pays", avec une seule observation par pays contenant le salaire moyen des indivius de ce pays :

```
collapse (mean) salaire, by(pays)
collapse (mean) salaire, by(pays) cw
```

La commande `collapse` traite les observations manquantes comme des zéros dans le calcul des statistiques. L'inclusion de l'option `cw` permet de ne pas utiliser ces données manquantes. Il est toujours préférable de s'assurer du nombre d'observations réellement utilisés dans le calcul des moyennes.

⚠ ATTENTION ⚠ la commande `collapse` remplace les données en mémoire. Il faut donc bien vérifier que le fichier de données utilisé jusqu'à cette commande a bien été sauvegardé avant.

2.2.6 Tronquer la base : *keep* et *drop*

Il arrive parfois de n'être intéressé que par une partie des observations contenues dans la base de données sur laquelle on travaille. Dans ce cas, il est possible de tronquer cette base de données en supprimant la partie des observations qui ne nous intéresse pas ou, de manière équivalente, en ne gardant que la partie qui nous intéresse. Les commandes permettant de réaliser cette opération sont `drop` et `keep`. Par exemple, dans la base com_info, si l'on travaille sur un phénomène urbain, on peut ne vouloir garder que les villes de plus de 10000 habitants en 1999. Alernativement, si l'on s'intéresse aux zones rurales, on pourrait vouloir supprimer de la base toutes les villes de plus de 2000 habitants. Voici les commandes correspondantes :

```
keep if pop1999 > 10000
drop if pop1999 > 2000
```

⚠ Attention, la suppression des données est définitive et il n'est pas possible de revenir à la base complète une fois celle-ci tronquée, à moins de recharger à nouveau le fichier de données initial. Lorsque vous

souhaitez tronquer votre base de données de manière temporaire, il est conseillé d'utiliser les commandes `preserve` et `restore` (voir encadré 5 ci-dessous).

Encadré 5 : faire des modifications temporaires avec Stata

Les commandes `preserve` et `restore` permettent de réaliser des modifications temporaires sur les données. Elles sont particulièrement utiles pour réaliser des opérations dont les effets sont irréversibles, comme tronquer ou transformer la base. La syntaxe est très simple, il suffit d'entourer l'ensemble des commandes correspondant aux modifications que vous souhaitez effectuer de manière temporaire par `preserve` (en préambule) et `restore` (en conclusion) dans votre do-file. Ex :

```
preserve
keep if pop1999 > 10000
sum pop1999
drop if pop1999 > 2000
sum pop1999
restore
```

2.3 Gérer plusieurs bases de données

2.3.1 Empiler des bases de données: *append*

La commande `append` permet de concaténer plusieurs bases de données, c'est à dire d'ajouter des observations à la fin d'un fichier en mémoire. Les variables doivent avoir le même nom dans les différents fichiers pour un empilage parfait. Par exemple, si l'on dispose de deux bases de données identiques sur l'emploi en France, l'une concernant l'année 2020 (*emploi2020*) et l'autre l'année 2021 (*emploi2021*), alors on peut les empiler (les "concaténer") dans une seule base de la manière suivante :

```
use emploi2020.dta, clear
append using emploi2021.dta
```

Dans l'exemple ci-dessus, si la variable permettant de distinguer l'année d'observation n'est pas disponible dans les fichiers avant la concaténation, alors il est indispensable de la créer au préalable. Lorsque les bases sont parfaitement identiques (les variables sont les mêmes) alors l'empilage est parfait. En revanche, lorsqu'une variable n'est disponible que dans la base de départ, alors cette variable aura une valeur manquante pour toutes les observations ajoutées à partir de la seconde base (et vice-versa).

2.3.2 Apparier des bases de données: *merge*

La commande merge permet de faire la jointure entre deux fichiers, c'est à dire de rajouter des variables (colonnes) pour les observations (lignes) d'une base de départ. L'appariement est réalisé selon une variable clé, l'identifiant d'appariement, qui est commun aux deux bases. Il est indispensable que les deux bases à apparier (le fichier *master* ou base principale et le fichier *using* ou base ajoutée) contiennent l'identifiant choisi pour que l'appariement soit réussi.

Prenons un exemple concret. Supposons que l'on dispose de deux bases individuelles, l'une portant sur le statut d'emploi des individus (*emploi*) et l'autre portant sur leur situation familiale (*sitfam*). Chacune des deux bases contient les identifiants individuels uniques *id_indiv*. La syntaxe pour apparier les deux bases est la suivante :

```
use emploi.dta, clear
merge 1:1 id_indiv using sitfam.dta
```

La commande `merge` permet d'effectuer différents types d'appariements :

- lorsque chaque observation de la base *master* et de la base *using* correspond à un identifiant unique, on utilise `merge 1:1` (= merge one observation to one observation)
- lorsque chaque observation de la base *master* a un identifiant unique mais correspond à plusieurs observations de la base *using*, on utilise `merge 1:m` (= merge one observation to many observations)
- lorsque plusieurs observations de la base *master* correspondent à un même identifiant unique mais à une seule observation de la base *using*, on utilise `merge m:1` (= merge many observations to one observation)
- lorsque les identifiants correspondent à plusieurs observations dans les deux bases, on utilise `merge m:m` (= merge many observations to many observations)

NB: la commande `merge` crée la variable *_merge*, qui permet d'évaluer si l'appariement a été réalisé correctement. Il faut toujours vérifier que l'opération d'appariement s'est bien déroulée. La variable *_merge* prendra la valeur 1 pour les observations de la base principale non retrouvées dans la base ajoutée, 2 pour les observations de la base ajoutée non retrouvées dans la base principale et 3 pour les observations retrouvées dans les deux bases et parfaitement appariées.

2.4 Gérer les variables et leur contenu

2.4.1 Supprimer des variables: *keep* et *drop*

De la même manière qu'il est possible de supprimer des observations (ou ne garder que certaines observations), il est possible de supprimer ou ne garder que certaines variables. Les commandes sont identiques, et il suffit d'ajouter la liste des variables que l'on souhaite garder/supprimer après la commande. Ex :

```
keep comdec pop1999 km2
drop pop2014 km2
```

2.4.2 Créer de nouvelles variables: *gen* et *egen*

La commande `generate` (abrégée en `gen`) permet de créer de nouvelles variables. On peut par exemple vouloir créer une variable égale à une constante (ex: 2) que l'on va appeler *cst* ou bien une autre variable égale à la somme de deux variables existantes (*salaire1* et *salaire2*) que l'on va appeler *salairetot* . La syntaxe est la suivante :

```
gen cst = 2
gen salairetot = salaire1 + salaire2
```

La commande `gen` est la commande la plus courante pour créer des variables de base. Mais vous pouvez créer des variables plus sophistiquées avec la commande `egen`. Cette commande introduit une fonction mathématique ou logique et peut être combinée avec la commande `by`. Par exemple, si vous voulez créer une variable qui donne la moyenne du salaire par entreprise :

```
bysort entreprise :  egen salaire_moyen=mean(salaire)
```

Il existe une très grande variété d'applications possibles pour la commande `egen`. Il est conseillé au lecteur de lire le fichier d'aide de cette commande, en exécutant la commande suivante : `help egen`.

Générer des dummies: Il est très courant de vouloir créer des variables *dummies* (indicatrices en français). Il s'agit de variables numériques prenant la valeur 1 lorsqu'une condition est vérifiée et 0 lorsque la condition n'est pas vérifiée. Pour créér ce type de variable, il suffit d'écrire la condition entre parenthèse après le signe =. Exemples :

```
gen jeune = (age <= 25)
gen femme = (sexe=="F")
```

La variable jeune va donc prendre la valeur 1 pour tous les individus dont la variable age est inférieure ou égale à 25; la variable femme va prendre la valeur 1 pour tous les individus dont la variable sexe (textuelle) est égale à "F".

Créer des variables textuelles: les commandes `gen` et `egen` peuvent également être utilisées avec des variables textuelles. Par exemple, si vous voulez extraire une partie du texte dans une variable textuelle, vous pouvez utiliser la fonction `substr`. L'exemple ci-dessous crée une variable appelée *pays3l* qui vous donne les trois premières lettres du nom des pays[6] :

```
gen pays3l=substr(pays,1,3)
```

Encadré 6 : les conditions et principaux opérateurs Stata

Avec n'importe quelle commande, vous pouvez ajouter une condition, par exemple si vous ne prenez en compte qu'un sous-échantillon de vos données. Il suffit d'écrire la condition à la fin de la commande à l'aide de `if` afin d'indiquer quel sous-échantillon vous prenez en compte. La condition `if` introduit une expression logique. L'opérateur le plus courant est `==`, qui signifie égal à. Par exemple, si vous voulez résumer la variable de population *pop1999* sur le sous-échantillon dont la catégorie d'aire urbaine *cataeu2010* est égale à 111 (les villes situées dans des grandes aires urbaines) :

```
sum pop1999 if cataeu2010=="111"
```

Il est bien évidemment possible d'ajouter d'autres conditions, grâce au symbole `&`. Il vous suffit d'écrire les conditions :

```
sum pop1999 if cataeu2010=="111" & km2 >=10
```

Voici quelques **opérateurs de base** : `&` (et) `|` (ou) `!` (non, n'est pas) `>` (supérieur à) `<` (inférieur à) `>=` (supérieur ou égal à) `<=` (inférieur ou égal à) `==` (égal à) `!=` (différent de) `+` (addition) `-` (soustraction) `*` (multiplication) `/` (division) `^` (puissance).

Par exemple, si vous voulez voir les observations pour lesquelles la population en 1999 (*pop1999*) n'est pas manquante : `browse if pop1999!=.`

[6] Pour plus d'information sur la commande `substr`, exécutez la commande `help substr`

2.4.3 Modifier les variables existantes

Renommer une variable : la commande `rename` permet de renommer les variables. Elle doit être suivie du nom actuel de la variable puis du nouveau nom que l'on souhaite lui attribuer. La syntaxe est la suivante : `rename km2 sqkm` (la variable *km2* va être renommée *sqkm*).

Changer le label de la variable : la commande `label` permet d'attribuer un label à une variable ou de changer le label existant. Par exemple, si on veut attribuer le label "kilometres carres" à la variable *km2*, la syntaxe est la suivante : `label var km2 "kilometres carres"`

Il est également possible d'assigner un label aux modalités d'une variable catégorielle. Pour cela, il faut d'abord définir les valeurs du label, puis assigner ce label à la ou aux variables. Par exemple, pour assigner un label à la variable *cataeu2010* :

```
label define cataeu2010label 1 "Petite aire urbaine" 2 "Grande aire urbaine"
label values cataeu2010 cataeu2010label
```

Modifier le contenu de la variable : Le codage des variables n'est pas toujours optimal et peut parfois contenir des erreurs. La commande `replace` permet de modifier les valeurs d'une variable existante. Il est notamment possible de remplacer les valeurs d'une variable donnée pour une sous-population en utilisant la condition `if`. Par exemple, si vous souhaitez attribuer le salaire moyen dans l'entreprise aux individus présents dans votre base et dont vous connaissez l'entreprise mais pour lesquels le salaire est manquant :

```
bysort entreprise :  egen salaire_moy=mean(salaire)
replace salaire=salaire_moy if salaire==.
```

Il est également possible de recoder les variables numériques avec la commande `recode`. Par exemple, si la situation familiale (*sitfam*) célibtaire est codée 3 et que vous préférez la coder 0 :

```
recode sitfam (3=0)
```

Modifier le format de la variable : Parfois, des variables de nature numérique (ex : âge, salaire) peuvent nénamoins être stockées comme variable textuelle, de sorte que Stata refuse de calculer la moyenne ou de donner un résumé de la distribution. Pour modifier le format de la variable, utilisez la commande `destring`, en spécifiant si vous voulez remplacer la variable précédente, avec l'option `replace`, ou en créer une nouvelle, avec l'option `gen()` :

```
destring age, replace
```

```
destring age, gen(agenum)
```

Même si c'est moins souvent utile, vous pouvez aussi changer une variable numérique en une variable textuelle avec la commande `tostring`, qui a exactement la même syntaxe que la commande `destring`

L'autre cas de figure est une variable textuelle par nature (ex : pays) que vous souhaitez coder en numérique. Dans ce cas, les commandes `encode`, pour passer en numérique, ou `decode` pour revenir en textuel, seront nécessaires. Pour ces commandes, vous êtes obligés de créer une nouvelle variable :

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

2.4.4 Changer l'ordre d'apparition des variables dans la base: *order*

Les commandes `order`, `aorder` et `move` permettent de trier les variables de la base de données. La commande `order` est suivie de la liste de variables à ordonner. La commande `aorder` réorganise toutes les variables (ou bien uniquement celles citées après la commande) par ordre alphabétique. La commande `move` envoie la première variable à la place de (juste devant) la deuxième variable. Exemples :

```
order comdec pop1999 pop2014 km2
aorder
move km2 pop1999
```

3 Explorer les données

3.1 Statistiques descriptives

3.1.1 Résumer les variables numériques : *summarize*

La commande `summarize` (abrégée en `sum`) permet d'afficher dans la fenêtre de résultats des statistiques de base sur une ou plusieurs variables numériques de la base de données, notamment la moyenne, l'écart-type et les valeurs extrêmes (min et max). L'option detail (abrégée en `d`, après la virgule) permet d'afficher des statistiques plus précises sur la distribution de la ou des variables, notamment la médiane et les autres centiles. La syntaxe est la suivante :

```
sum pop1999 pop2014
```

Figure 6: Summarize

Variable	Obs	Mean	Std. Dev.	Min	Max
pop1999	38,261	1522.531	13615.64	0	2125017
pop2014	38,261	1664.975	14398.33	0	2220809

```
sum pop1999, d
```

Figure 7: Sumarize avec l'option detail

	Percentiles	Smallest		
1%	0	0		
5%	0	0		
10%	64	0	Obs	38,261
25%	152	0	Sum of Wgt.	38,261
50%	352		Mean	1522.531
		Largest	Std. Dev.	13615.64
75%	872	390174		
90%	2356	444852	Variance	1.85e+08
95%	4707	796525	Skewness	108.0403
99%	20723	2125017	Kurtosis	15847

Il est possible de combiner la commande `summarize` avec la commande `bysort` afin d'afficher des statistiques par sous-groupe. Le groupe doit alors être défini par une variable catégorielle (ex : pays, sexe, tranche d'âge, etc.). La commande permet d'afficher les statistiques de base pour chaque valeur de cette variable catégorielle (chaque groupe ou catégorie). Par exemple, pour afficher la population moyenne en 1999 (*pop1999*) par catégorie d'aire urbaine (*cataeu2010*) :

```
bysort cataeu2010 :  sum pop1999
```

On peut aussi combiner la commande `summarize` avec la commande `if` pour afficher des statistiques sur un sous-échantillon particulier. Par exemple, pour afficher la population moyenne en 1999 (*pop1999*) pour les communes avec une superficie supérieure à 3 kilomètres carrées (*km2*) :

```
sum pop1999 if km2 >3
```

3.1.2 Résumer les variables textuelles : *tabulate* et *table*

La commande `tabulate` (abrégée en `tab`) permet d'afficher les tableaux de fréquences (i.e., la distribution) des variables numériques catégorielles ou des variables textuelles[7]. Par exemple, pour afficher la distribution de la variable *cataeu2010*, pour l'échantillon total et pour le sous-échantillon des communes avec une superficie supérieur à 2 km2 :

```
tab cataeu2010
tab cataeu2010 if km>2
```

La commande `tabulate` peut aussi être utilisée pour créer un tableau à double entrée :

```
tab cataeu2010 grandeville
```

Figure 8: Tabulate

categorie aire urbaine	Grande ville (dummy) 0	1	Total
111	2,344	833	3,177
112	8,498	3,276	11,774
120	2,726	1,083	3,809
211	238	201	439
212	505	267	772
221	397	435	832
222	375	165	540
300	4,357	2,296	6,653
400	3,634	3,297	6,931
Total	23,074	11,853	34,927

Une autre commande utile est `table`. Cette commande permet de combiner les fonctiones de tabulate et de summarize, et présente une grande flexibilité dans les statistiques qu'elle permet d'afficher. Par exemple, pour afficher la moyenne et l'écart type de la population moyenne en 1999 et en 2014 selon qu'il s'agit d'une grande ville ou non :

```
table grandeville, c(mean pop1999 sd pop1999 mean pop2014 sd pop2014)
```

[7] Cette commande peut également fonctionner pour une variable numérique mais les résultats affichés seront difficile à interpréter car il y aura autant de fréquences affichées que de valeurs possibles pour la variable

Encadré 7 : les valeurs manquantes et la commande `misstable`

Par défaut, la commande `tabulate` affiche les fréquences pour les valeurs non manquantes. Il suffit de rajouter l'option `m` après la virgule (ex: `tab cataeu2010,m`) pour afficher également le pourcentage de valeurs manquantes. De même, pour les variables numériques, la commande `summarize` ne peut calculer des statistiques que pour les valeurs non manquantes. La commande `misstable` permet de connaître le pourcentage de valeur manquantes pour ce type de variable. La syntaxe est la suivante :

```
misstable sum pop1999
```

3.2 Analyser les données

3.2.1 Corrélation entre les variables et alpha de Cronbach : *correlate*, *pwcorr* et *alpha*

La commande `correlate` (abregée en `corr`) permet d'obtenir le coefficient de corrélation (ou la covariance, avec l'option `cov` après la virgule) entre deux ou plus variables. Exemple :

```
corr pop1999 km2
corr pop1999 km2, cov
```

Figure 9: Correlate

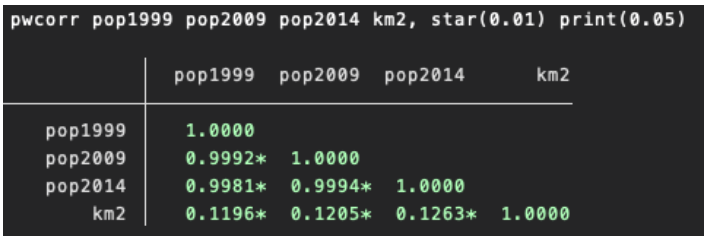
```
. corr pop1999 km2
(obs=34,920)

             | pop1999      km2
-------------+------------------
    pop1999  |   1.0000
        km2  |   0.1196   1.0000


. corr pop1999 km2, cov
(obs=34,920)

             | pop1999      km2
-------------+------------------
    pop1999  |  2.0e+08
        km2  |  28014.5  270.434
```

La commande `pwcorr` permet de tester la significativité des coefficients de corrélation et de n'afficher que les coefficients significatifs à un certain seuil que l'on indique au préalable. L'hypothèse testée est H0 : absence de corrélation. Exemple :

```
pwcorr pop1999 pop2009 pop2014 km2, star(0.01) print(0.05)
```

Dans cet exemple, Stata affichera les corrélations significatives au seuil de 5% et marquera avec une étoile celles significatives au seuil de 1%.

Figure 10: pwcorr



La commande `alpha` permet de calculer des *alphas de Cronbach* et de générer une variable résumant un groupe de variables. Cette commande est très utile lorsque vos données contiennent des "échelles", c'est à dire un ensemble de variables visant à mesurer le même concept. Par exemple, lorsque vous cherchez à mesurer l'estime de soi des individus dans le cadre d'une enquête, vous allez souvent utiliser l'échelle de Rosenberg (Rosenberg, 1965). Cette échelle est composée de plusieurs questions visant toutes à mesurer le même concept (l'estime de soi). La validité de l'échelle repose sur l'hypothèse que les questions qui la composent mesurent bien la même chose. Pour s'assurer de la validité de cette hypothèse, on calcule l'*alpha de Cronbach*, qui mesure de manière synthétique les corrélations entre les réponses aux différentes questions de l'échelle. On considère généralement qu'une échelle est valide si son *alpha de Cronbach* est supérieur à 0.8 (parfois 0.7). La commande suivante permet de calculer l'*alpha de Cronbach* :

```
alpha estime1 estime2 estime3 estime4
```

On peut ensuite générer une variable *estime_idx* résumant les variables *estime1* à *estime4* avec l'option `gen()` : `alpha estime1 estime2 estime3 estime4, gen(estime_idx)`

3.2.2 Tests de comparaison : *ttest*

Les commandes `ttest` et `sdtest` permettent respectivement de faire des test d'égalité de moyennes et des tests d'égalité de variances (écart-types). Formellement, on teste l'hypothèse nulle H0 : la différence des moyennes -ou écart types- de chaque groupe d'une variable est nulle. Par exemple, pour tester l'égalité des moyennes ou écart-type de la superficie (*km2*) entre les grandes communes et les petites communes :

```
ttest km2, by(grandeville)
sdtest km2, by(grandeville)
```

Figure 11: ttest

```
. ttest km2, by(grandeville)

Two-sample t test with equal variances

   Group |     Obs        Mean    Std. Err.   Std. Dev.   [95% Conf. Interval]
---------+--------------------------------------------------------------------
       0 |  23,074    7.981994    .0229024    3.478898    7.937104    8.026884
       1 |  11,846    29.63202    .1978951    21.53877    29.24411    30.01992
---------+--------------------------------------------------------------------
combined |  34,920    15.32639    .0880021    16.44487     15.1539    15.49887
---------+--------------------------------------------------------------------
    diff |           -21.65002    .1453507               -21.93492   -21.36513
------------------------------------------------------------------------------
    diff = mean(0) - mean(1)                                      t = -1.5e+02
Ho: diff = 0                     degrees of freedom =    34918

    Ha: diff < 0                 Ha: diff != 0                 Ha: diff > 0
 Pr(T < t) = 0.0000         Pr(|T| > |t|) = 0.0000          Pr(T > t) = 1.0000
```

3.2.3 Analyse en Composante Principale (ACP) : *pca*

L'Analyse en Composantes Principales (ACP) est une méthode très courante d'analyse des données, qui a pour objectif de résumer l'information contenue dans un jeu de données composé d'un nombre élevé d'observtions (lignes) et de variables (colonnes). Il s'agit d'un outil de synthèse permettant de traiter des données **quantitatives**.[8] La commande `pca`, suivie du nom des variables sur lesquelles on veut réaliser l'ACP, permet de réaliser ce type d'analyse. Par exemple :

```
pca x1 x2 x3 x4
```

Cette commande produit deux tableaux de résultats (cf. Figure 12). Le premier tableau permet notamment de connaître la *valeur-propre* de chaque composante créée par l'ACP, et de savoir quelle est la part de l'*inertie* des données (i.e., de la variation de l'ensemble des variables sélectionnées pour l'ACP) qui est résumée par cette composante. Le second tableau renseigne sur la position des variables sur chaque axe (ou composante).

[8] Concrètement, la méthode consiste à transformer des variables corrélées entre elles en de nouvelles variables décorrélées les unes des autres. Ces nouvelles variables sont nommées « composantes principales », ou axes principaux. Cette méthode permet donc de réduire le nombre de variables et de rendre l'information moins redondante.

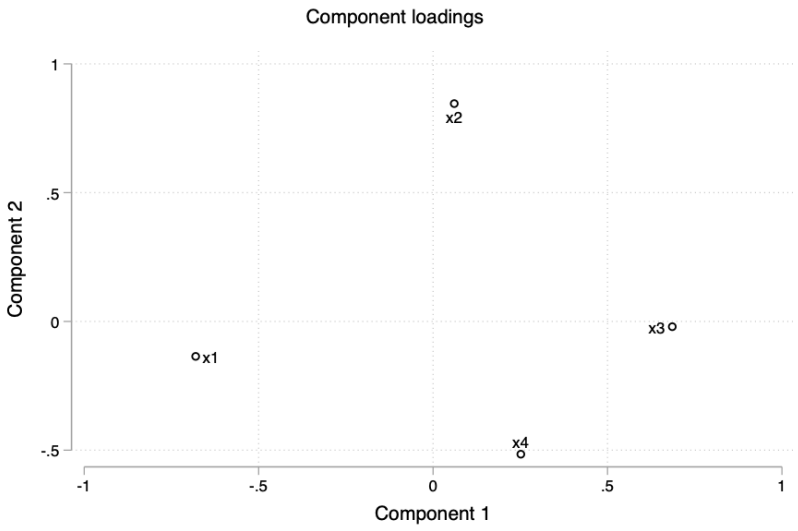
Figure 12: ACP - tableaux

```
. pca x1 x2 x3 x4

Principal components/correlation                  Number of obs    =     83,008
                                                  Number of comp.  =          4
                                                  Trace            =          4
    Rotation: (unrotated = principal)             Rho              =     1.0000

    Component |   Eigenvalue   Difference        Proportion   Cumulative
--------------+------------------------------------------------------------
        Comp1 |      1.33409       .31741            0.3335       0.3335
        Comp2 |      1.01668     .0537745            0.2542       0.5877
        Comp3 |      .962901      .276563            0.2407       0.8284
        Comp4 |      .686338            .            0.1716       1.0000

Principal components (eigenvectors)

    Variable |    Comp1     Comp2     Comp3     Comp4  Unexplained
-------------+--------------------------------------------------------
          x1 |   -0.6805   -0.1359    0.1662    0.7006            0
          x2 |    0.0607    0.8457    0.5208    0.0994            0
          x3 |    0.6856   -0.0205   -0.1811    0.7048            0
          x4 |    0.2515   -0.5156    0.8176   -0.0497            0
```

Il est souvent préférable de représenter ces résultats à l'aide d'un graphique à deux dimensions (les deux composantes principales). Pour produire ce graphique, on utilise la commande `loadingplot`, en précisant le nombre de dimensions du graphique[9] :

```
loadingplot, factors(2)
```

Figure 13: ACP - graphique



[9]Par défaut, le graphique produit est en 2 dimensions. On peut spécifier plus de dimensions, mais le graphique sera plus difficile à interpréter.

3.3 Graphiques

Il existe une grande variété de graphiques possibles sous Stata et d'options pour ces graphiques. Nous présentons ici les commandes nécessaires pour générer les graphiques les plus courants, mais les fichiers d'aides (`help graph`) seront très utiles dès lors que vous souhaiterez générer un type de graphique ou utiliser une option plus spécifique.[10]

Tout d'abord, il faut distinguer entre les représentations graphiques d'une seule variable et pour lesquelles on utilise la commande `graph`, et celles mettant en relation deux (ou plus) variables dont la ligne de commande commence par `twoway`. La syntaxe générale de tout graphique sera :

`graph type_graphique Y, options`	pour un graphique unidimensionnel
`twoway type_graphique Y X, options`	pour un graphique bi-dimensionnel.

Les options rajoutées à la fin de la commande après la virgule (`title(), ytitle(), xtitle(), legend(), note(), xlabel(), ylabel()`, ...) permettent d'organiser et d'embellir les représentations de base.

3.3.1 Graphiques unidimensionnels : *graph*

Les diagrammes en secteurs ou "camemberts" permettent de représenter des variables qualitatives nominales. Pour les générer, on utilise la commande `graph pie`. Exemple (cf. Figure 14) :

```
graph pie, over(sex) ///
plabel(_all percent, color(white) size(medlarge)) ///
title(Population par sexe du répondant) ///
note(Source:  WVS data)
```

Les diagrammes en "tuyaux d'orgues" permettent de représenter des variables qualitatives ordinales ou les variables quantitatives discrètes. Pour les générer, on utilise la commande `graph bar` ou `graph hbar`. Exemple (cf. Figure 15) :

```
graph bar (count) , over(x1) ///
outergap(30) blabel(bar) title(Effectifs par niveau de bien-être déclaré) ///
note(Source:  WVS data)
```

[10] Il est aussi possible de générer des graphiques à partir du menu déroulant. Si cette méthode peut servir afin de se familiariser avec la production de graphique sous Stata, elle est néanmoins nettement moins pratique et moins rapide, notamment lorsqu'il faut répliquer différents graphiques sur de nouvelles variables ou modifier à la marge des graphiques obtenus.

Figure 14: Diagramme en secteurs avec *graph pie*

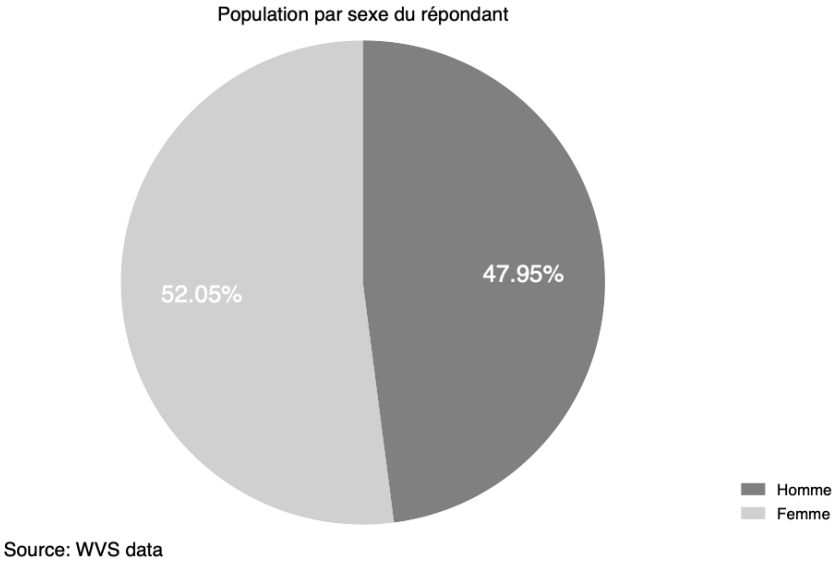
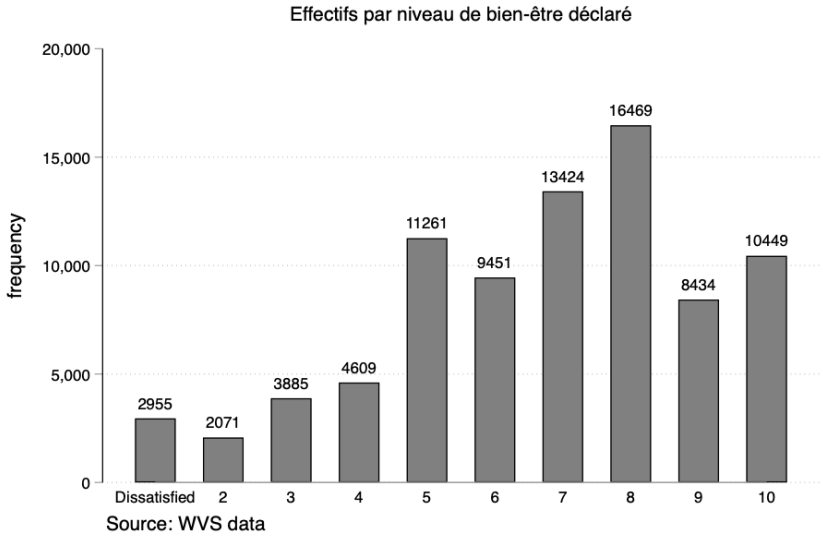


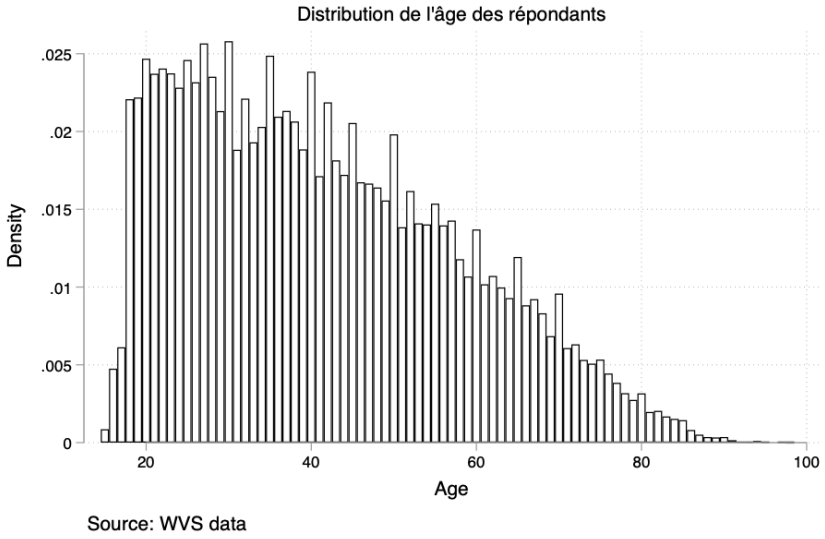
Figure 15: Diagrammes en tuyaux d'orgues avec *graph bar*



Pour les variables quantitatives continues, on préfère généralement les histogrammes. Pour les générer, on utilise la commande `histogram` ou `graph hbar`. Exemple :

```
histogram age, ///
title(Distribution de l'âge des répondants) note(Source: WVS data) d
```

Figure 16: Histogrammes



3.3.2 Graphiques bi-dimensionnels : *twoway*

On représente généralement la distribution jointe de deux variables continues à l'aide d'un nuage de points. Pour générer ce type de graphique, on utilise la commande `twoway scatter Y X`, où Y est la variable à représenter en ordonnées et X celle à représenter en abscisses. Exemple (cf. Figure 17) :

```
twoway scatter SACSECVAL age , ///
title(Importance des valeurs traditionnelles par âge) note(Source:  WVS data)
```

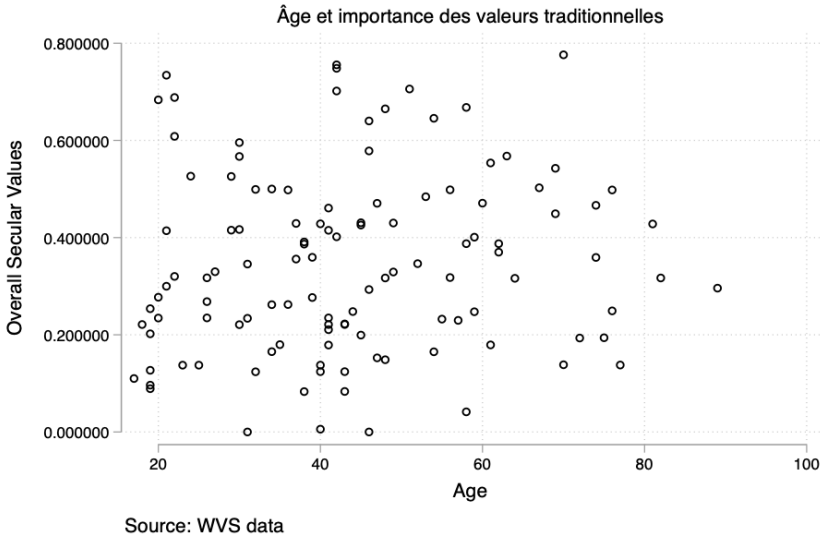
Des options sont disponibles si on veut représenter, par exemple, non plus des points mais une courbe qui relie ces points (utile lorsque seule une observation de Y est disponible pour chaque valeur de la variable X). Exemples :

```
twoway line Y X, sort
twoway connect Y X, sort
```

Il est aussi possible de représenter plusieurs courbes sur un même graphique. Par exemple, si on veut représenter Y1 vs X et Y2 vs le même axe X, on écrira :

```
twoway (line Y1 X) (line Y2 X)
```

Figure 17: Graphiques bi-dimensionnels avec *twoway scatter*



3.3.3 Sauvegarder ses graphiques : *graph save* et *graph export*

Pour sauvegarder un graphique après l'avoir créé, on utilise la commande suivante (on peut aussi utiliser l'interface graphique) :

```
graph save nom_graph, replace
```

Le graphique sera sauvegardé au format Stata .gph. Pour sauvegarder dans un autre format :

```
graph export nom_graph.ext, replace
```

Le graphique sera sauvegardé au format indiqué par l'extension .ext. Si le graphique est destiné à apparaître dans un document Microsoft word, il est préférable de l'enregistrer au format .wmf ou au format .pdf.

4 Économétrie

Stata peut effectuer un grand nombre de commandes de régression, générales ou très spécifiques : `anova`, `ivreg`, `qreg`, `reg`, `tobit`, `xtreg`, etc. La syntaxe de base de ces commandes est la suivante :

```
regress depvar [varlist] [if exp] [weight], options
```

où *depvar* doit être remplacée par la variable dépendante (le "y") et *varlist* par les variables indépendantes (les "x"). Comme pour les autres commandes, on peut rajouter une condition de restriction à un certain sous-échantillon en combinant la commande `if` avec des expression logiques (ex : `if age < 25`). Si l'on souhaite pondérer les observations (cf. encadré 8) à partir d'une variable de la base nommée *pond*, il faut l'indiquer entre crochets (ex : `[aweight=pond]`). Enfin, il est possible de rajouter des options, notamment afin de modifier la manière de traiter la constante du modèle (ex : `noconstant`) et les termes d'erreurs (ex : `vce(robust)`).

4.1 Régression linéaire (MCO)

4.1.1 La commande *regress*

Supposons que l'on cherche à estimer le vecteur de paramètres β dans l'équation suivante :

$$y = X\beta + \epsilon$$

où y est le vecteur des observations de la variable à expliquer, X la matrice de variables explicatives $(x_1, x_2, ...x_p)$ et ϵ un vecteur d'aléas ou résidus.

La commande `regress` (abrégée en `reg`) permet d'estimer un modèle linéaire par la méthode des moindres carrés ordinaires (MCO). Par exemple, si on cherche à expliquer le bien-être déclaré des individus (*satisfaction*) par leur revenu (*income*), leur âge (*age*) et leur nombre d'enfants (*children*), la syntaxe est la suivante (cf. Figure 18 pour le tableau de résultat) :

```
reg satisfaction income age children
```

On peut rajouter une condition pour n'estimer cette équation que pour les femmes de notre échantillon :

```
reg satisfaction income age children if sex==2
```

4.1.2 Gérer les variables indicatrices dans les régressions

Il est très courant de souhaiter inclure des indicatrices dans notre modèle. Par exemple, si on dipose d'une variable indiquant le satut d'emploi (*employment*), on peut vouloir inclure une indicatrice pour chaque valeur possible de cette variable. Pour ce faire, il existe 3 solutions :

1. La plus longue, créer à la main toutes les indicatrices avec `generate` et `replace` :

```
gen employ1 = 0

replace employ1= 1 if employment == 1

gen employ2 = 0

replace employ2=1 if employment ==2

...
```

2. Créer toutes les indicatrices possibles en une seule commande en combinant `tabulate` avec l'option `gen`. Ex :

```
tabulate employment, gen(employment)
```

3. La plus rapide, sans créer les variables indicatrices, en rajoutant `i.` devant la variable dont vous souhaitez extraire les indicatrices, directement dans votre commande de régression :

```
reg satsfaction income i.employment
```

Figure 18: Régression linéaire avec *reg*

```
. reg satisfaction income age children

      Source |       SS           df       MS      Number of obs   =    83,008
-------------+----------------------------------   F(3, 83004)     =    934.45
       Model |  14805.0017         3  4935.00058   Prob > F        =    0.0000
    Residual |  438357.804    83,004  5.28116482   R-squared       =    0.0327
-------------+----------------------------------   Adj R-squared   =    0.0326
       Total |  453162.806    83,007  5.45933242   Root MSE        =    2.2981

------------------------------------------------------------------------------
satisfaction |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
      income |   .1145848   .0027787    41.24   0.000     .1091386     .120031
         age |    .007205   .0005179    13.91   0.000     .0061899    .0082201
    children |  -.1274245   .0038304   -33.27   0.000     -.134932   -.1199169
       _cons |   6.141255   .0248343   247.29   0.000      6.09258    6.18993
------------------------------------------------------------------------------
```

Encadré 8 : les poids dans les régression

Dans des données administratives ou issues d'enquêtes, on trouve souvent une variable de pondération (ex : *pond*) qui correspond au poids de chaque observation (i.e., combien de personnes la personne interrogée représente dans la population totale). Il est possible d'utiliser cette pondération dans les régressions en utilisant l'option `weight`. Il existe quatre types de poids (weights) :

- **fweight** pour *frequency weights* : ce sont les observations qui sont dupliquées en autant de fois que la pondération l'indique. Si *pond*=127 pour une observation, Stata considère qu'il y a en fait 127 observations identiques dans la population. L'option fweight peut être adaptée lorsque vous travaillez sur des données administratives agrégées (ex : au niveau région ou commune). En revanche, elle est déconseillée pour des données d'enquêtes, car vous risquez d'obtenir une très forte significativité de vos coefficients alors que vous ne disposez pas de ces données sur toute la population.
- **pweight** pour *probability weights* : indique l'inverse de la probabilité que cette observation soit inclue dans l'échantillon (i.e., la probabilité de tirage). Par exemple, si *pond*=100 pour une observation, cela signifie que cette observation a une probabilité 1/100 d'être dans l'échantillon.
- **aweight** pour *analytic weights* : ce sont des poids qui sont inversement proportionnels à la variance d'une observation, c'est-à-dire que la variance de la $j$ème observation est supposée être $\frac{\sigma^2}{w_j}$ où w_j est le poids. Ces poids sont par exemple adaptés lorsque les observations représentent des moyennes et les poids sont le nombre d'éléments qui ont donné lieu à la moyenne.
- **iweight** pour *importance weights* : pas de définition statistique formelle.

4.2 Modélisation des variables qualitatives dichotomiques : *logit* et *probit*

Il est très courant de chercher à expliquer la réalisation d'un évènement ou d'une condition. Par exemple, on peut chercher à estimer l'effet du diplôme et de l'âge sur la probabilité d'être en emploi. La variable à expliquer est alors qualitative dichotomique. Concrètement, il s'agit d'une indicatrice qui prend les

valeurs 1 ou 0 (ex : 1 pour les individus en emploi et 0 pour les autres), et dont on veut expliquer les réalisations par une série de variables.

Deux méthodes courantes pour ce type d'analyses reposent sur les modèles *logit* et *probit*[11]. Pour estimer ces modèles, Stata dispose de commandes spécifiques. Ces commandes et leurs options sont très similaires à la commande `regress`. Exemples :

```
logit employed i.diplome age
probit employed i.diplome age
```

Figure 19: Estimation d'un modèle *logit*

```
. logit employed i.diplome age

Iteration 0:   log likelihood = -34891.241
Iteration 1:   log likelihood = -32870.971
Iteration 2:   log likelihood = -32839.802
Iteration 3:   log likelihood = -32839.725
Iteration 4:   log likelihood = -32839.725

Logistic regression                             Number of obs     =     53,870
                                                LR chi2(5)        =    4103.03
                                                Prob > chi2       =     0.0000
Log likelihood = -32839.725                     Pseudo R2         =     0.0588

------------------------------------------------------------------------------------------------------------------
                                        employed |      Coef.   Std. Err.      z    P>|z|     [95% Conf. Interval]
-------------------------------------------------+----------------------------------------------------------------
                                         diplome |
                      Incomplete primary school  |  -.2576057   .0457836    -5.63   0.000    -.3473398   -.1678716
                        Complete primary school  |   .3452976   .0360842     9.57   0.000     .2745739    .4160214
Complete secondary school: technical/ vocatio..  |   1.144103   .0341993    33.45   0.000     1.077074    1.211133
Complete secondary school: university-prepara..  |   1.074425   .0348327    30.85   0.000     1.006154    1.142696
                                                 |
                                             age |  -.0129557   .0005865   -22.09   0.000    -.0141053   -.0118061
                                           _cons |   -.761399   .0396681   -19.19   0.000    -.8391471   -.6836509
------------------------------------------------------------------------------------------------------------------
```

[11] Voir par exemple Crépon & Jacquemet (2018) ou Wooldridge (2018) pour des manuels de référence en économétrie et pour plus d'informations sur les modèles *logit* et *probit*

4.3 Commandes post-estimation : *predict* et *test*

Il est possible de récupérer les valeurs prédites et les résidus d'une régression en utilisant la commande `predict` après avoir lancé une régression. Par exemple :

```
reg employed i.diplome age
predict emphat
predict empresid, resid
```

La deuxième ligne de commande ci-dessus crée la variable *emphat* qui est équivalente au $\hat{y}$ de votre modèle estimé par MCO (i.e., le taux d'emploi prédit par les variables explicatives et les coefficients estimés par la régression MCO), alors que la troisième ligne de commande crée une variable égale au résidu de cette prédiction (dans le cas du modèle MCO, ce résidu est égal à la différence entre y, la valeur observée du statut d'emploi, et $\hat{y}$, la valeur estimée par le modèle).

Il est aussi possible d'effectuer des tests sur les coefficients après l'estimation de la regression. Par exemple, pour tester si le coefficient associé à l'âge est différent de 0, ou pour tester s'il y a une différence significative entre le fait d'avoir un diplome de niveau 2 ou de niveau 3 :

```
reg employed i.diplome age
test age=0
test 2.diplome = 3.diplome
```

Figure 20: Tests post-estimation

```
. test age=0

 ( 1)  age = 0

       F(  1, 53864) =  483.16
            Prob > F =    0.0000

. test 2.diplome = 3.diplome

 ( 1)  2.diplome - 3.diplome = 0

       F(  1, 53864) =  190.37
            Prob > F =    0.0000
```

Pour voir la liste complète des commandes de post-estimation possibles, vous pouvez regarder la liste des possibilités dans le Menu/Statistics/Postestimation, après le lancement d'une commande d'estimation.

4.4 Stocker et exporter les résultats

Il est possible de stocker les résultats d'une estimation dans Stata avec la commande `estimates store` :

```
reg employed i.diplome age
estimates store reg1
```

La deuxième ligne de commande ci-dessus sauvegarde les résultats de la régression (notamment la taille de l'échantillon, les coefficient estimés, leur écart-type, etc.) sous le nom *reg1*. L'intérêt est de pouvoir réutiliser les résultats de cette régression, sans relancer la commande de régression. Par exemple, si vous souhaitez réutiliser ces résultats dans un calcul, il suffit de faire référence aux variables `_b[nomdevariable]` qui sont les coefficients de la régression précédente et aux variables `_se[nomdevariable]` qui sont les écarts-type. Pour la constante, il suffit de faire référence à `_b[_cons]`. Ex :

```
gen coeff1 = _b[age] + _b[_cons]
```

Vous pouvez également afficher les résultats avec la commande `estimates table`. Cette commande permet d'afficher simultanément les résultats de plusieurs régressions. Ex :

```
reg employed i.diplome age
estimates store reg1
reg employed i.diplome gender
estimates store reg2
estimates table reg1 reg2
```

Pour voir la liste des estimations stockées dans la mémoire de Stata : `estimates dir`

4.4.1 Exporter les résultats

Il est possible d'exporter les résultats obtenus dans Stata dans des documents externes, généralement dans un format word, excel ou LaTeX. Ils existent plusieurs commandes permettant de réaliser ce type d'export. Nous présentons ici la commande `outreg2`. Afin d'utiliser cette commande, il sera généralement nécessaire d'installer le package correspondant au préalable (cette commande n'est nécessaire qu'une seule fois) : `ssc install outreg2`

La syntaxe pour exporter les résultats est la suivante :

```
reg employed i.diplome age
outreg2 using "filename.xls", nocons label replace
```

Les options `nocons`, `label` et `replace` permettent respectivement de ne pas afficher la constante, d'utiliser le label des variables (plutôt que leur nom) dans le fichier créé et de remplacer le fichier du même nom si celui-ci existe. Si vous souhaitez exporter d'autres résultats dans le même fichier :

```
reg employed i.diplome gender
outreg2 using "filename.xls", nocons label append
```

L'option `append` permet d'ajouter les résultats au fichier existant (plutôt que d'écraser le fichier, avec l'option `replace`).

Lorsque vous avez stocké une ou plusieurs estimations au préalable avec la commande `estimates store`, alors il vous suffit d'écrire :

```
outreg2 reg1 using "filename.xls", nocons label replace
outreg2 reg2 using "filename.xls", nocons label append
```

où reg1 et reg2 doivent être remplacés par les noms sous lesquels vous avez stockés les résultats.

Appendix A Ressources et références

- User Manuel, pdf which describes the main features and commands
  - http://www.Stata.com/manuals13/u.pdf (version 13)
- Stata Reference Manual: Stata dictionary (book)
  - https://www.Stata.com/manuals/r.pdf
- Stata Graphics Manual: to do graphs
  - http://www.Stata.com/manuals13/g.pdg (version 13)

- Cameron and Trivedi (2010), « Microeconometrics using Stata », Stata Press, 706p.
- Crépon, B., Jacquemet, N. (2018). Économétrie: méthode et applications. De Boeck Supérieur.
- Rosenberg , M. (1965) *Society and the adolescent self-image*, Princeton university press.
- Wooldridge, J. (2018). Introduction à l'économétrie: une approche moderne. De Boeck Supérieur.