

Pense bête de quelques commandes sous Stata

(Eric Cahuzac, cahuzac@toulouse.inra.fr)

- **describe** pour avoir la liste des variables du fichier (avec les labels)
ds pour une liste succincte
codebook tax détail de cette variable (missing, valeurs uniques, bornes,...)
lookfor tax pour lister tous les noms des variables contenant 'tax'
 - **rename sex sexe** renomme la variable sex en sexe
 - **more** dans un fichier do : arrête un prog
pause donne la main, pour la reprendre taper q.
 - **scalar** define a=1 pour définir le scalaire **a** et lui donner la valeur **1**
 - **gen rm=real(rmi)** transforme une variable caractère en numérique
gen str4 zau=substr(zemp,1,1) créé une variable caractère à partir d'une autre
gen jeune = inrange(age,18,25) jeune=1 si $18 \leq \text{age} \leq 25$ et jeune=0 sinon
gen id = _n créé un identifiant individuel
gen age21=(age>21) age21=1 si age>21 (ou missing) et 0 sinon
gen taillegr = recode(taille,150,160,170,180,190,200) regroupe la taille en 6 classes
- ⓘ **Attention** : Stata considère le "." comme la valeur la plus élevée pour une variable numérique et le "" comme la valeur plus faible pour une variable caractère.
- **egen** taillegr = **cut**(taille), group(5) découpe taille en 5 groupes
egen avg = **mean**(chol) aussi autres calculs min, max, std,
egen newid = **group**(oldid) créé un identifiant pour chaque obs de oldid
egen tzau = **group**(taille zau) créé autant de modalités à tzau que de couples
egen abc = **concat**(a b c) concatène les variables **a b** et **c**
egen zau = **eqany**(zemp), v(3/5) crée zau=1 si zemp vaut 3, 4 ou 5 et 0 sinon
 - **encode** sex, gen(genre) crée genre (*numérique*) à partir de sex (*caractère*)
destring, replace change toutes les variables *caractère* en *numérique*
tostring sexe transforme une variable numérique en chaîne (Stb 56)
 - **replace** zau = 4 if zau == 2 | zau == 3 recode 2 et 3 en 4 pour zau (| = ou, & = et)
replace b = 3 in 100/110 recode b à 3 pour les observations 100 à 110
 - **by** id, sort : **replace** sex = sex[_n-1] if sex == . voir aussi **stfill** sex, **forward**
 - **mvencode** _all, mv(999) recode les missing de toutes les variables à 999
mvdecode sexe, mv(0 -1) recode les 0 et les -1 de la variable sexe à missing
 - **list** A pour lister une variable
list if _n==3 pour lister certaines observations
list if _n == _N

- `print @Result` imprime la fenêtre de résultat
- `print @Graph` imprime la fenêtre de graphique
- `label define sexe 0 "Homme" 1 "Femmes"` définit un label de modalités
si beaucoup de modalités :
 `label define depart 1 "Grand bassin parisien"`
 `label define depart 2 "Nord Ouest",add`
 ...
puis `label value sex sexe, nofix` pour assigner ce label à la variable sex
- `label list sexe` liste le label sexe
- `label drop sexe` supprime le label sexe
- `label list` liste tous les labels
- `label variable rmi "Beneficiaire du Rmi"` label de variable
for x in 1/12 : `label variable varX "Variable numéro X"` pour chaîner les labels
- `tabulate` pour des stats descriptives sur des variables discrètes
`tab A B, row col cel` tableau croisé avec %tage lignes colonnes et cellules
`tab1 A B` deux tableaux simples
- `tab zau, gen(zau)` créé autant de dichotomiques que de modalité dans zau
- `summarize rev` pour des stats descriptives sur les variables continues
`sum rev, detail` même chose mais avec les centiles
- `tabstat txent, stats(mean count n sum sd)` équivalent mais plus complet sur les stats
- `tabulate foreign, summarize(mpg)` donne des stats des d'une variable continue par modalité d'une variable discrète
- `table zemp cs , c(mean salh n salh)` équivalent mais plus complet sur les stats
- `univar rev` même chose que sum mais avec quartiles
`univar rev, vlabel boxplot` permet d'avoir les labels et sous forme de tableau
`univar rev, by(id)` pour univar le by est une option (faire un sort avant)
- `means age` puis
`scalar moy = r(mean)` stocke dans le scalaire moy la moyenne de age
- `centile price, c(1 3 5 50 95)` calcule les centiles spécifiés
- `ci price, level(95)` donne un intervalle de confiance à 95% pour la variable
- `count if income<0` compte le nombre d'observations où income est <0
`by id, sort: count if income<0`
- `set obs 500` créé un fichier de 500 obs
`range x -5 5` créé x qui va de -5 à 5 sur 500 points

Commandes répétées

- **for** X in 7318 7319 7320 : rep zau="7300" if zau=="X" recode la **zau**
for X in 5/10 : gen varX=X^2 crée des variables indicées
for new dip1-dip25 \ var cse1-cse25: gen X= dip*Y créé une variable croisée dip*cse
for X in var v1 v2 v3: sts graph, na by(X) \ more réalise 2 actions dans la boucle
- **foreach** var of varlist **pri-rep t*** { summarize `var' } identique à for
- **forvalues** identique à for
forval num = 5/13 { count if var`num' > 10 }
forval X = 1(1)10 {for Y in num 10[1]15: di `X' \ scalar define SY`X'=`X'}

ⓘ **Attention** : **foreach** et **forvalue** permettent de réaliser des boucles emboîtées où on exécute plusieurs commandes ce que ne sait pas faire **for**. Attention toutefois à la syntaxe pour les listes et les cotes (voir dernier exemple).

Un exemple de match-merge

- Soit coller dads297 à dads197 selon la clé : zemp cs trage sexe nes12

```
use dads297
sort zemp cs trage sexe nes12
save dads297, replace
use dads197
sort zemp cs trage sexe nes12
merge zemp cs trage sexe nes12 using dads297

tabulate _merge
drop if _merge < 3
drop _merge
save dads, replace
```

Quelques éléments sur les matrices

- **matrix** input A=(1,2\3,4) crée A (2x2), les lignes sont séparée par des \
mkmat V1 V2 V3, matrix(A) crée A (_Nx2), à partir du fichier en mémoire
mat B = **I**(4) crée B (4x4) remplie de 1
mat D = **J**(L,C,1) crée D (LxC) remplie de 1
- mat **list** A affiche A à l'écran
- **display** A[2,2] affiche l'élément (2,2) de A : ici 4
scalar define a=3* A[2,2] crée le scalaire a = 12
- **matfncs** à voir pour des opérations plus complètes
matoper sur les matrices.

Opérations sur les dates

- `gen datedeb = date(debut,"dmy")` si début est une chaîne de la forme dd/mm/yy ou dd-mm-yy fait de datedeb un format date
- `gen duree = datefin-datedeb` donne le **nombre de jours** écoulés entre 2 dates

Quelques éléments sur les modèles de durées

- Toutes les commandes relatives aux durées, commencent par "st".
- Avant de faire un modèle de durées, il faut indiquer à Stata le type de variable utilisée pour décrire les durées (intervalle, date, continue,...) et lui dire quelle est la censure. on le fait avec la commande `stset`
- `stset dur, failure(cens = =1 2)` permet d'initialiser la variable de durée
- `stdes` statistiques descriptives de la variable de durées
- `sts list`
`sts list,na` affiche la statistique de Kaplan Meyer
affiche la statistique de Nelson Alen
- `sts gen km=s` enregistre Kaplan Meyer dans la variable km
- `sts graph, by(jeune)` graphe de la fonction de survie de Kaplan-Meyer
- `stsum, by (fac)` l'équivalent de `summari`
- `sts test nbpub`
`sts test nbpub, wilcoxon` fait un Log-rank test sur la variable nbpub
fait un test de Wilcoxon
- `stcox` pour les modèles à HP semi paramétriques
- `streg` pour les modèles à HP ou AFT paramétriques.