Le langage Stata

Table of contents

| SECTION | COMMANDES ET EXPRESSIONS |
| --- | --- |
| **Opérateurs** | = == < <= > >= != & | + - \* / ^ |
| **Valeurs manquantes** | . **mdesc** **mvpatterns** **misschk** |
| **Suppression de l’output, et affichage d’une expression** | quietly display |
| **Sélection groupées** | *drop* *keep* \* - |
| **Macros et répétition** | local global foreach forvalue return list i *regress* |

* **En gras, commandes externes**
* *En italique, commandes associées à un chapitre ultérieur*

# 1. **La syntaxe générique**

**Remarque sur les crochets**

|  |
| --- |
| […] expression entre crochets |
| A ma connaissance, à l’exception des pondérations il n’y a pas d’utilisation de crochets dans la syntaxe des commandes usines. Dans les fichiers d’aide et pour cette formation, ils indiquent les expressions optionnelles d’une ligne d’instructions. |

Forme concise de la syntaxe STATA pour une instruction portant sur des variables:

nom\_command varlist [, options ]

***Exemple*** : **tabulate var1 var2, nofreq row**

Ce qui ce traduit par: produire un tableau croisé entre *var1* (en ligne) et la *var2* (en colonne) en affichant la répartition en % de *var2* pour chaque valeur de *var1* et pour l’ensemble des donnnées renseignées .

(1978 automobile data)

tabulate rep78 foreign, nofreq row

Repair |  
 record | Car origin  
 1978 | Domestic Foreign | Total  
-----------+----------------------+----------  
 1 | 100.00 0.00 | 100.00   
 2 | 100.00 0.00 | 100.00   
 3 | 90.00 10.00 | 100.00   
 4 | 50.00 50.00 | 100.00   
 5 | 18.18 81.82 | 100.00   
-----------+----------------------+----------  
 Total | 69.57 30.43 | 100.00

* Forme générique de la syntaxe STATA

[prefix:] command varlist [ [type\_weight=var] if/in, options]

Expression conditionnelle (sélection): if (sélection de valeurs) ou in (sélection d’observations)

Exemple : **bysort var2: summarize var1 if var4!=1, detail**  
Ce qui ce traduit par: pour chaque valeur de *var2*, des statistiques descriptives détaillées pour la variable *var1* si la valeur de *var3* est différente de 1.

bysort foreign: summarize price if rep78!=1, detail

-> foreign = Domestic  
  
 Price  
-------------------------------------------------------------  
 Percentiles Smallest  
 1% 3291 3291  
 5% 3667 3299  
10% 3892 3667 Obs 50  
25% 4181 3799 Sum of wgt. 50  
  
50% 4782.5 Mean 6132.74  
 Largest Std. dev. 3143.481  
75% 6303 13466  
90% 11441 13594 Variance 9881473  
95% 13594 14500 Skewness 1.717717  
99% 15906 15906 Kurtosis 4.857758  
  
-------------------------------------------------------------------------------  
-> foreign = Foreign  
  
 Price  
-------------------------------------------------------------  
 Percentiles Smallest  
 1% 3748 3748  
 5% 3798 3798  
10% 3895 3895 Obs 22  
25% 4499 3995 Sum of wgt. 22  
  
50% 5759 Mean 6384.682  
 Largest Std. dev. 2621.915  
75% 7140 9690  
90% 9735 9735 Variance 6874439  
95% 11995 11995 Skewness 1.215236  
99% 12990 12990 Kurtosis 3.555178

Si la commande implique une base de données, le nom de la base est généralement précédée de **`using``** (sauf pour les instructions officielle d’ouverture/sauvegarde d’une base):

command using nom\_base [,options]

# 2. **Autres langages pris en charge**

*interne*

* Langage matriciel (MATA)
* Editeur de texte (SMCL) pour rédiger les aides ou paramétrer la forme des outputs (c’est une une horreur)
* Création de boites de dialogue (proche du Java)
* La programmation de commande dispose d’un certain nombre d’éléments de langage dédié. Pour une réutilisation ultérieure automatisée, le programme est enregistré dans un fichier *.ado*

*Externe*

* Via une commande externe (rsource), on peut exécuté du R. R doit ête bien évidemment installé.

rsource, terminator(END\_OF\_R)   
  
library(readr)  
trans <- read.csv("https://raw.githubusercontent.com/mthevenin/analyse\_duree/master/bases/transplantation.csv")  
  
head(trans)  
table(trans$died)  
  
END\_OF\_R

id year age died stime surgery transplant wait mois compet  
1 15 68 53 1 1 0 0 0 1 1  
2 43 70 43 1 2 0 0 0 1 1  
3 61 71 52 1 2 0 0 0 1 1  
4 75 72 52 1 2 0 0 0 1 1  
5 6 68 54 1 3 0 0 0 1 2  
6 42 70 36 1 3 0 0 0 1 1  
  
table(trans$died)  
  
 0 1   
28 75

* Depuis la version 16 on peut programmer intéractivement en Python avec Stata. Cette intégration permet à Python de reconnaitre les macros stata dans son code…c’est très utile. Python doit néanmoins être installé manuellement.

*Vérification de l’installation*

python query

Python Settings  
 set python\_exec C:/Users/thevenin\_m/AppData/Local/Programs/Python/Py  
> thon310/python.exe  
 set python\_userpath   
  
 Python system information  
 initialized yes  
 version 3.10.5  
 architecture 64-bit  
 library path C:\Users\thevenin\_m\AppData\Local\Programs\Python\Py  
> thon310\python310.dll

*Utilisation de python*

python:   
  
a = 4  
b = 2  
a\*b  
   
end

----------------------------------------------- python (type end to exit) -----  
>>>   
>>> a = 4  
>>> b = 2  
>>> a\*b  
8  
>>>   
>>> end  
-------------------------------------------------------------------------------

python:  
import pandas as pd  
trans = pd.read\_csv("https://raw.githubusercontent.com/mthevenin/analyse\_duree/master/bases/transplantation.csv")  
  
trans.head(10)  
trans.info()  
end

----------------------------------------------- python (type end to exit) -----  
>>> import pandas as pd  
>>> trans = pd.read\_csv("https://raw.githubusercontent.com/mthevenin/analyse\_du  
> ree/master/bases/transplantation.csv")  
>>>   
>>> trans.head(10)  
 id year age died stime surgery transplant wait mois compet  
0 15 68 53 1 1 0 0 0 1 1  
1 43 70 43 1 2 0 0 0 1 1  
2 61 71 52 1 2 0 0 0 1 1  
3 75 72 52 1 2 0 0 0 1 1  
4 6 68 54 1 3 0 0 0 1 2  
5 42 70 36 1 3 0 0 0 1 1  
6 54 71 47 1 3 0 0 0 1 1  
7 38 70 41 1 5 0 1 5 1 1  
8 85 73 47 1 5 0 0 0 1 1  
9 2 68 51 1 6 0 0 0 1 1  
>>> trans.info()  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 103 entries, 0 to 102  
Data columns (total 10 columns):  
 # Column Non-Null Count Dtype  
--- ------ -------------- -----  
 0 id 103 non-null int64  
 1 year 103 non-null int64  
 2 age 103 non-null int64  
 3 died 103 non-null int64  
 4 stime 103 non-null int64  
 5 surgery 103 non-null int64  
 6 transplant 103 non-null int64  
 7 wait 103 non-null int64  
 8 mois 103 non-null int64  
 9 compet 103 non-null int64  
dtypes: int64(10)  
memory usage: 8.2 KB  
>>> end  
-------------------------------------------------------------------------------

* Depuis la version 17 de Stata, on peut également programmer intéractivement en *Java*… mais là je n’y connais absolument rien.
* Intégration de l’édition en *markdown* pour produire des rapports en html ou pdf. Egalement possibilité de générer des documents word ou excel avec des commandes dédiées. L’intégration de *latex* est également possible via une commande externe. Toutes ces possibilité me semble néanmoins nettement en deçà de ce qui est réalisable actuellement avec **RStudio** en particulier avec le nouvel outil Quarto associé à Statamarkdown…qui sont utilisés pour faire ce support.

# 3. **Les opérateurs**

| **Opérateurs** |
| --- |
| **&** **Et** |
| **|** [Alt+Gr6] **Ou** |
| **Opérateurs d’affectation** |
| **=** |
| **Opérateurs pour expressions conditionnelles** |
| **==** |
| **!=** |
| **>** |
| **>=** |
| **<** |
| **<=** |
| **Opérateurs arithmétique** |
| + , - , / , ^ (puissance) |
| **Opérateur chaîne de caractères** |
| + |

[1] *+* concatène des variables caractères qui n’ont pas de valeur manquante. Sinon utiliser **concat** associée à la commande **egen** (voir chapitre sur la création de variables).

# 4. **Les valeurs manquantes**

|  |
| --- |
| Statut des valeurs manquantes |
| La valeur d’une observation manquante dépasse la plus grande valeur observée d’une variable. Ceci doit être pris en compte dans les expressions conditionnelles impliquant par exemple des regroupement de variables ordinales ou mésurées (âge, revenus…): Si une variable numérique a des observations manquantes, la condition if X>valeur conservera ces informations.  Si :   * ...if x>4 conserve * ...if x>4 & x<. ou ...if x>4 & x!=. regroupe seulement |

**Les valeurs manquantes utilisateurs**  
On peut rendre la valeur manquante informative en lui ajoutant une lettre: **.a** , **.b**, **.c** etc….

Exemple:

* *.a* = Ne sait pas.
* *.b* =Refus.
* *.c* =Pas de réponse.

**Repérage des valeurs manquantes**

Des commandes, comme tabulate avec l’option mis pour les variables catégorielles, permettent de repérer et d’afficher le nombre d’observations manquantes.

Il y a aussi plusieurs commandes qui permettent d’analyser ce type d’observations observations globalement.

* Commande externe **mdesc**: affiche pour chaque variable de la base ou une sélection de celle, le nombre et le % d’observations manquantes.
  + installation: ssc install mdesc
  + syntaxe: mdesc [varlist]

sysuse auto.dta, clear  
  
mdesc

(1978 automobile data)  
  
 Variable | Missing Total Percent Missing  
----------------+-----------------------------------------------  
 make | 0 74 0.00  
 price | 0 74 0.00  
 mpg | 0 74 0.00  
 rep78 | 5 74 6.76  
 headroom | 0 74 0.00  
 trunk | 0 74 0.00  
 weight | 0 74 0.00  
 length | 0 74 0.00  
 turn | 0 74 0.00  
 displacement | 0 74 0.00  
 gear\_ratio | 0 74 0.00  
 foreign | 0 74 0.00  
----------------+-----------------------------------------------

* commandes externes mvpatterns et misschk pour analyser les différents patterns de valeurs manquantes (une même observation peut avoir des valeurs manquantes sur plusieurs variables). misschk ne scanne que les variables de type numérique, et permet de générer deux variables pour indiquer le nombre et le pattern de valeurs manquantes pour chaque observation.
  + installation: ssc install mvpatterns et ssc install misschk
  + syntaxe: mvpatterns [varlist] et misschk [varlist], gen(nom)

mvpatterns

variables with no mv's: make price mpg headroom trunk weight length turn  
 displacement gear\_ratio foreign  
  
Variable | type obs mv variable label  
-------------+---------------------------------------  
rep78 | int 69 5 Repair record 1978  
-----------------------------------------------------  
  
Patterns of missing values  
  
 +------------------------+  
 | \_pattern \_mv \_freq |  
 |------------------------|  
 | + 0 69 |  
 | . 1 5 |  
 +------------------------+

misschk

Variables examined for missing values  
  
 # Variable # Missing % Missing  
--------------------------------------------  
 1 price 0 0.0  
 2 mpg 0 0.0  
 3 rep78 5 6.8  
 4 headroom 0 0.0  
 5 trunk 0 0.0  
 6 weight 0 0.0  
 7 length 0 0.0  
 8 turn 0 0.0  
 9 displacement 0 0.0  
 10 gear\_ratio 0 0.0  
 11 foreign 0 0.0  
  
Warning: this output does not differentiate among extended missing.  
To generate patterns for extended missing, use extmiss option.  
  
 Missing for |  
 which |  
 variables? | Freq. Percent Cum.  
---------------+-----------------------------------  
 \_\_3\_\_ \_\_\_\_\_ \_ | 5 6.76 6.76  
 \_\_\_\_\_ \_\_\_\_\_ \_ | 69 93.24 100.00  
---------------+-----------------------------------  
 Total | 74 100.00  
  
Missing for |  
 how many |  
 variables? | Freq. Percent Cum.  
------------+-----------------------------------  
 0 | 69 93.24 93.24  
 1 | 5 6.76 100.00  
------------+-----------------------------------  
 Total | 74 100.00

# 5. **Casse et troncature**

## 5.1 Sensibilité à la casse

Comme R ou Python, Stata est intégralement sensible à la casse pour les instructions, seulement en minuscules. Par exemple TABULATE X renverra un message d’erreur.

## 5.2 Troncature des instructions et des options

Dans le fichier d’aire d’une commande usine ou externe, le niveau de troncature est indiqué par un soulignement dans l’instruction: par exemple **tabulate** est souligné au niveau de **ta** : **tabulate** = **tabulat** = **tabula** = **tabul** = **tabu** = **tab** = **ta**. On utilise généralement tabulate ou tab.

A manier avec une certaine précaution car le programme peut devenir rapidement incompréhensible, surtout s’il est partagé entre personnes dont la pratique diffère à ce niveau:

tabulate var1 var2, nofreq row miss   
  
\* est équivalent à:   
  
ta var1 var2, nof r m

# 6. **Suppression de l’output, et affichage d’une expression**

On peut rendre le résultat d’une commande invisible dans la fenêtre output avec quietly (qui).

tab rep78  
qui tab rep78

Repair |  
record 1978 | Freq. Percent Cum.  
------------+-----------------------------------  
 1 | 2 2.90 2.90  
 2 | 8 11.59 14.49  
 3 | 30 43.48 57.97  
 4 | 18 26.09 84.06  
 5 | 11 15.94 100.00  
------------+-----------------------------------  
 Total | 69 100.00

Remarque: ne fonctionne pas avec les graphiques où l’on doit utiliser l’option nodraw

**display (di)**

Ce n’est pas une commande à proprement parler, mais l’instruction display (di) permet d’afficher dans l’output, entres autres, des opérations arithmétiques (c’est donc une calculatrice).

di exp(1)/(1+exp(1))  
di "SALUT!!!!"

.73105858  
  
SALUT!!!!

Elle est également utilisé pour vérifier le contenu d’une macro variable, de préférence lorsque cette macro implique des valeurs.

# 7. **Sélection groupées de variables**

Commandes associées pour filtrer: **keep**, **drop** [pour sélectionner des obseravations: keep if, drop if]

On peut sélectionner un ensemble de variables qui ont une racine commune, par exemple **c**, en écrivant : \*c\*.  
Exemple:  
*television*, *telephone*, *table* ont comme racine **t**. Pour supprimer ces variables, on peut exécuter drop t\* au lieu de drop television telephone table.  
Si on souhaite supprimer *television* et *telephone* seulement : drop tele\*.

Si on veut sélectionner des variables occurencées ou comme dans la base *auto* les 5 variables qui se suivent [*headroom*, *trunk*, *weight*, *length*, *turn*]: keep headroom-turn . Pour des variables occurencées de x1 à x5: keep x1-x5.

sum t\*

Variable | Obs Mean Std. dev. Min Max  
-------------+---------------------------------------------------------  
 trunk | 74 13.75676 4.277404 5 23  
 turn | 74 39.64865 4.399354 31 51

# 8. **Macros et répétition**

## 8.1 Introduction au macros

Juste une introduction…Vu la simplicité du langage Stata, il est conseiller de se mettre rapidement à la manipulation des expressions dites macro.

* Une macro, dans sa version la plus simple, est une expression qui est utilisée une ou plusieurs fois dans un programme. Elle sont de type temporaire (local) ou enregistré en dur (global).
* Les commandes, en particulier sur les opérations statistiques, enregistre un certains nombre d’objet de type macro qui peuvent être utilisés ultérieurement. On peut récupérer leur liste à la fin du fichier d’aide, et les visualiser les valeurs enregistrées après avoir exécuté une commande avec return list, ereturn list…
* Un autre type d’objet, appelé **scalar** ressemble à une macro mais n’en est pas. Il s’agit de pseudo variables. Ils ne seront pas traités.
* Le contenu d’une macro peut être affiché avec display ou macro list (mac list)
  + privilégier display pour afficher le contenu d’une macro de type valeur
  + privilégier mac list pour afficher le contenu d’une macro de type chaîne de caractère lorsque des doubles quotes (“) doivent rester apparentes. La gestion des” dans les macros peut s’avérer particulièrement retord.
    - macro *local*: mac list \_macroname
    - macro *global*: mac list macroname

**Macro temporaire**

L’instruction local permet de définir des macros variables temporaire (disparaissent après l’exécution du programme):

local nom\_macro expression  
\* ou  
local nom\_macro = expression numérique  
  
\* ou  
  
local nom\_macro : fonction macro // hors contenu de la formation

local a = 2  
  
di `a'  
  
di 5^(`a')

2  
  
25

local var mpg foreign  
  
di "`var`"  
mac list \_var  
  
sum `var'  
regress price `var`

`var`  
  
\_var: mpg foreign  
  
 Variable | Obs Mean Std. dev. Min Max  
-------------+---------------------------------------------------------  
 mpg | 74 21.2973 5.785503 12 41  
 foreign | 74 .2972973 .4601885 0 1  
  
` invalid name  
r(198);  
  
end of do-file  
r(198);

On en voit tout de suite l’utilité: si je veux changer la liste de variables pour les instructions sum (troncature de summarize) et pour regress, je le fais une fois (dans la définition de la macro) au lieu de deux.

Avec Stata peut définir également des macros dites *global* qui sont sauvegardées et s’appliqueront à tous les programmes (on peut les supprimer). Leur utilisation est moins courante: global nom\_macro expression, le nom de la macro dans l’expression s’écrira $nom\_macro.  
Il est conseillé de les supprimer en fin de programme avec macro drop + noms des macros.

global var mpg weight length turn  
  
di "$var"  
mac list var  
  
sum $var  
reg price $var  
  
macro drop var

mpg weight length turn  
  
var: mpg weight length turn  
  
 Variable | Obs Mean Std. dev. Min Max  
-------------+---------------------------------------------------------  
 mpg | 74 21.2973 5.785503 12 41  
 weight | 74 3019.459 777.1936 1760 4840  
 length | 74 187.9324 22.26634 142 233  
 turn | 74 39.64865 4.399354 31 51  
  
 Source | SS df MS Number of obs = 74  
-------------+---------------------------------- F(4, 69) = 12.13  
 Model | 262166817 4 65541704.3 Prob > F = 0.0000  
 Residual | 372898579 69 5404327.23 R-squared = 0.4128  
-------------+---------------------------------- Adj R-squared = 0.3788  
 Total | 635065396 73 8699525.97 Root MSE = 2324.7  
  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 mpg | -94.651 80.879 -1.17 0.246 -256.000 66.697  
 weight | 5.030 1.154 4.36 0.000 2.728 7.332  
 length | -73.147 40.212 -1.82 0.073 -153.368 7.074  
 turn | -323.861 126.882 -2.55 0.013 -576.983 -70.738  
 \_cons | 19581.418 6005.223 3.26 0.002 7601.327 31561.509  
------------------------------------------------------------------------------

La liste des macro enregistrées en dur est donnée par l’instruction: macro dir

mac dir

GRSTYLE\_SN0: s2color  
GRSTYLE\_SN: \_GRSTYLE\_  
GRSTYLE\_FN: D:\D\ado\personal/scheme-\_GRSTYLE\_.scheme  
Rterm\_options: --vanilla  
Rterm\_path: C:\Program Files\R\R-4.1.2\bin\R.exe  
tmp: D:\D\stata\_temp\  
user: C:\Users\thevenin\_m\  
F1: help advice;  
F2: describe;  
F7: save  
F8: use  
S\_ADO: BASE;SITE;.;PERSONAL;PLUS;OLDPLACE  
S\_StataSE: SE  
S\_OS: Windows  
S\_OSDTL: 64-bit  
S\_MACH: PC (64-bit x86-64)  
S\_level: 95  
S\_MODE: batch  
S\_FN: C:\PROGRA~1\Stata17\ado\base/a/auto.dta  
S\_FNDATE: 13 Apr 2020 17:45

## 8.2 Objets sauvegardés lors der l’exécution d’une commande

* Ces objets de type macro ne sont conservés en mémoire qu’entre 2 commandes exécutés
* On peut donc les manipuler qu’à ce moment là, en particulier les enregistrer sous forme de macro standard pour les utiliser ultérieurement.

qui sum price  
  
return list  
  
local mprice = r(mean)  
  
di `mprice'

scalars:  
 r(N) = 74  
 r(sum\_w) = 74  
 r(mean) = 6165.256756756757  
 r(Var) = 8699525.974268788  
 r(sd) = 2949.495884768919  
 r(min) = 3291  
 r(max) = 15906  
 r(sum) = 456229  
  
  
6165.2568

|  |
| --- |
| Note |
| Une application typique est la normalisation d’une pondération brute (somme des poids = nombre d’observation dans l’échantillon). Si *wb* est la pondération brute (somme des poids = population cible), et *wn* les poids que l’on souhaite normaliser:  qui sum wb gen wn = wb/`r(mean)'  Tout changement de la variable *wb* modifiera automatiquement la normalisation.  Pour la commande \*\*gen (ou generate) se reporter au chapitre 5. |

## 8.3 Répétition avec des boucles

* **`forvalues**: valeurs occurencées, compteur

for num 1/n: commande est de plus en plus abandonnée (aide Stata supprimée). On lui préfère maintenant l’instruction forvalues pour effectuer des boucles sur des occurences numériques.

Si l’on souhaite par exemple changer le nom des variables x1 à x9 en var1, var2,…., var9:

forvalues i=1/9 {  
   
rename x`i' var`i'   
   
}

* **foreach**: termes d’une expression enregistrée sous la forme d’une macro

Par l’exemple, et juste une petite introduction. On veut faire une régression linéaire entre la variable *price* et la variable *foreign* en ajoutant une seule autre variable dans cette liste: *mpg*, *headroom*, *trunk*.

Au lieu d’exécuter:

regress price foreign mpg   
regress price foreign headroom   
regress price foreign trunk

On génère une macro variable temporaire qui liste ces 3 variables , et on exécute une boucle avec l’instruction foreach.

local var mpg headroom trunk  
  
foreach x of local var {  
  
regress price foreign `x', noheader  
}

price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 foreign | 1767.292 700.158 2.52 0.014 371.217 3163.368  
 mpg | -294.196 55.692 -5.28 0.000 -405.242 -183.149  
 \_cons | 11905.415 1158.634 10.28 0.000 9595.164 14215.667  
------------------------------------------------------------------------------  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 foreign | 577.812 787.566 0.73 0.466 -992.549 2148.174  
 headroom | 491.575 428.405 1.15 0.255 -362.641 1345.791  
 \_cons | 4522.071 1412.097 3.20 0.002 1706.430 7337.711  
------------------------------------------------------------------------------  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 foreign | 1190.155 760.805 1.56 0.122 -326.847 2707.157  
 trunk | 262.772 81.852 3.21 0.002 99.564 425.980  
 \_cons | 2196.541 1267.857 1.73 0.088 -331.494 4724.576  
------------------------------------------------------------------------------

….et on peut aller plus loin… Juste pour information car cela se complique (et pas qu’un peu), avec une technique de macro empilée, on ajoute les les variables une à une au modèle.

local j mpg weight length turn headroom trunk  
  
foreach j2 of local j {  
local x `x' `j2'  
   
di "covariables introduites = `x'" // pour afficher ce qui est lu dans la macro  
   
regress price `x' , noheader  
}

4.   
covariables introduites = mpg  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 mpg | -238.894 53.077 -4.50 0.000 -344.701 -133.088  
 \_cons | 11253.061 1170.813 9.61 0.000 8919.088 13587.033  
------------------------------------------------------------------------------  
covariables introduites = mpg weight  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 mpg | -49.512 86.156 -0.57 0.567 -221.302 122.278  
 weight | 1.747 0.641 2.72 0.008 0.468 3.025  
 \_cons | 1946.069 3597.050 0.54 0.590 -5226.245 9118.382  
------------------------------------------------------------------------------  
covariables introduites = mpg weight length  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 mpg | -86.789 83.943 -1.03 0.305 -254.209 80.630  
 weight | 4.365 1.167 3.74 0.000 2.036 6.693  
 length | -104.868 39.722 -2.64 0.010 -184.090 -25.646  
 \_cons | 14542.434 5890.632 2.47 0.016 2793.940 26290.929  
------------------------------------------------------------------------------  
covariables introduites = mpg weight length turn  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 mpg | -94.651 80.879 -1.17 0.246 -256.000 66.697  
 weight | 5.030 1.154 4.36 0.000 2.728 7.332  
 length | -73.147 40.212 -1.82 0.073 -153.368 7.074  
 turn | -323.861 126.882 -2.55 0.013 -576.983 -70.738  
 \_cons | 19581.418 6005.223 3.26 0.002 7601.327 31561.509  
------------------------------------------------------------------------------  
covariables introduites = mpg weight length turn headroom  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 mpg | -96.145 80.259 -1.20 0.235 -256.300 64.010  
 weight | 5.015 1.145 4.38 0.000 2.730 7.300  
 length | -60.922 40.791 -1.49 0.140 -142.319 20.476  
 turn | -332.592 126.045 -2.64 0.010 -584.111 -81.073  
 headroom | -538.252 373.149 -1.44 0.154 -1282.859 206.356  
 \_cons | 19317.280 5961.554 3.24 0.002 7421.185 31213.375  
------------------------------------------------------------------------------  
covariables introduites = mpg weight length turn headroom trunk  
------------------------------------------------------------------------------  
 price | Coefficient Std. err. t P>|t| [95% conf. interval]  
-------------+----------------------------------------------------------------  
 mpg | -94.063 80.371 -1.17 0.246 -254.484 66.357  
 weight | 5.079 1.148 4.42 0.000 2.788 7.371  
 length | -73.487 43.011 -1.71 0.092 -159.338 12.364  
 turn | -327.070 126.311 -2.59 0.012 -579.188 -74.952  
 headroom | -731.292 427.369 -1.71 0.092 -1584.324 121.740  
 trunk | 98.275 105.721 0.93 0.356 -112.745 309.295  
 \_cons | 20447.251 6090.068 3.36 0.001 8291.424 32603.078  
------------------------------------------------------------------------------