

Programmer avec EViews

Formation DCPM

Louis de Charsonville

21 mars 2016

1. Les Objets sous EViews
2. Programmer avec Eviews
3. TD : Créer une courbe de Phillips

Les Objets sous EViews

Les Objets



Figure 1: Les différents objets sous EViews

Les Objets sous EViews

Déclarer et assigner un objet

Déclaration d'un objet

Pour déclarer un objet sous EViews, on utilise la commande

```
type (options) nom
```

Exemple

- Déclarer une série

```
series maSerie
```

- Déclarer une matrice

```
matrix(3,4) maMatrice
```

Après avoir déclaré un objet, il faut lui assigner une valeur. C'est fait avec le signe =

Exemple

- `maSerie=0` → assigne la valeur "0" à tous les éléments de la série.
- `monScalaire=2`

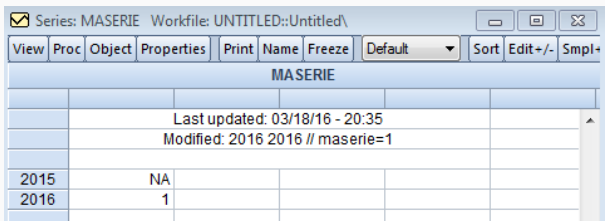
À noter :

- La déclaration d'une série ne dépend pas du **sample** actif. En revanche l'assignation en dépend.

Assigner un objet

Exemple

```
wfcreate Q 2015 2016  
smpl 2016 2016  
series maSerie  
maSerie=1
```



MASERIE	
Last updated: 03/18/16 - 20:35	
Modified: 2016 2016 // maserie=1	
2015	NA
2016	1

Figure 2: EViews output

L'initialisation et la déclaration sont souvent faits avec une seule commande

Exemple

```
equation eq.ls y c x1 x2  
series y=0
```

Les Objets sous EViews

Procédure associée

Procédure (ou méthode, commande) associée

Chaque objet a des procédures qui lui sont associées. On les utilise comme ceci :

```
objet.methode
```

Certaines méthodes commencent par "@" :

```
objet.@methode
```

Exemple

- `monGroupe.@count` → renvoie le nb d'éléments dans le groupe `monGroupe`
- `maSerie.@name` → renvoie le nom de la série `maSerie`

Certaines procédures ont des *arguments* :

```
objet.methode list_args
```

Exemple

- `monGroup.add maSerie maSerie2` → ajoute `maSerie` et `maSerie2` à `monGroupe`
- `monEquation.ls y c x1 x2` → la procédure ".ls" prend comme arguments : l'endogène, un objet coefficient et la liste des exogènes.

La syntaxe complète d'une commande sur EViews est :

```
action(options) nomObjet.commande(options) arguments
```

Il y a quatre types d'actions :

- `show` : affiche la vue d'un objet (par exemple, le résultat d'une équation).
- `do` : exécute l'action sans afficher la vue.
- `freeze` : crée un graphique ou une table à partir de la fenêtre de vue
- `print` : imprime la vue.

À noter

Par défaut, l'action `do` est implicite.

Les Objets sous EViews

Commande auxilliaire

Les commandes auxiliaires sont les commandes qui ne sont pas reliées à un objet particulier. Elles sont généralement utilisées pour manipuler les objets ou les workfiles. La syntaxe est

```
commande(liste_options) liste_args
```

Exemple

```
store(d="V:\databases\prix\insee.edb") frx000000
```

qui sauvegarde dans la database insee la série frx000000.

Quelques commandes auxilliaires utiles

wfcreate, pagecreate, wfopen, wfclose, fetch, store, copy, dbopen, dbclose.

Les Objets sous EViews

Les séries temporelles

Définition : série temporelle

Une série temporelle est caractérisée par :

- une fréquence : annuelle, trimestrielle, mensuelle etc.

Définition : série temporelle

Une série temporelle est caractérisée par :

- une fréquence : annuelle, trimestrielle, mensuelle etc.
- un vecteur de données

Définition : série temporelle

Une série temporelle est caractérisée par :

- une fréquence : annuelle, trimestrielle, mensuelle etc.
- un vecteur de données
- une date de début (et une date de fin)

Commandes associées (1/5)

Commande	Description
$d(x, n)$	$(1 - L)^n X$
$@lag(x, n)$	$X(-n)$
$dlog(x)$	$(1 - L) * \log(X)$
$@pch(x)$	$\frac{\delta X}{X}$
$@pcy(x)$	glissement annuel

Table 1: Quelques commandes utiles

Accéder à un élément d'une série

- `@elem(x, date)` : renvoie l'observation de la série x à la date X
- `x(i)` : renvoie la i ème observation de la série x .

Attention : selon le contexte, `x(i)` renvoie la série avancée d'une période

Créer une dummy

- On peut générer une dummy via **une condition logique** :

```
series dumResPos = (mesResidus > 0)
```

→ la variable `dumResPos` vaut 1 lorsque `mesResidus` est positif.

- ou à partir d'une **date** :

```
series apres2011Q1 = @after("2011Q1")
```

→ la variable `apres2011` vaut 1 à partir de 2011Q1.

Opérations

Les opérations usuelles $+$, $-$, \times , $/$ fonctionnent avec les séries et sont faites éléments par éléments.

```
series brentEuro=brent*tchange
```

- `series maRacine = @sqrt (maSerie)`
- `series monExp = exp (maSerie)`
- `series hicp_log = log (hicp)`

Fonctions mathématiques

- `@mean(maSerie)` : renvoie un scalaire
- `@max(maSerie)`
- `@min(maSerie, smp11)` : renvoie le minimum de `maSerie` sur la *sample* `smp11`
- `@rmse(maSerie, maSerie2)` : renvoie l'erreur quadratique entre les séries `maSerie` et `maSerie2`

⇒ Presque toutes les fonctions usuelles sont *natives* dans EViews, l'aide en fournit une liste exhaustive.

Groupe de séries

Un **groupe** est un objet EViews qui regroupe plusieurs séries.

```
group monGroupe maSerie1 maSerie2
```

Prodécures usuelles

- `monGroupe.@count` : renvoie le nombre d'éléments du groupe
- `monGroupe.@seriesname(i)` : renvoie le nom de la *i*ème variable du groupe

Les Objets sous EViews

Les Graphiques

Graphique

On déclare un graphique en lui assignant un type :

```
graph monGraph.type liste_series
```

Les types les plus courants sont

- lignes : `graph monGraph.line monGroupe`
- diagramme : `graph monGraph.bar(s) monGroupe`
- scatter : `graph monGraph.scat x y`

On peut personnaliser le graphique avec un certain nombre d'options

Procédures associées usuelles

- `monGraph.setelem(1) lpat(1) lcolor(red)`
`legend("Inflation")`
- `monGraph.datelabel format("YY[Q]Q")`
- `monGraph.addtext(t, font(b)) "Inflation en France"`

Utiliser un template

```
monGraph.template(t) nom_du_template
```

Les Objets sous EViews

Les variables muettes / de contrôle

Les variables de contrôle (1/2)

Variables de contrôle

EViews permet de la définition de variables *muettes* ou de contrôle, ce ne sont pas des objets sauvegardés dans le workfile, qui peuvent-être utilisées au sein du programme.

Il y a deux types de variables de contrôle :

- les variables **caractères** : utiles pour définir une date, un chemin

```
%debPrev = "2013m04"
```

```
%data="C:\mon_chemin\vers\les\donnees"
```

- Les variables **numériques** : utiles pour définir un compteur

```
!monCompteur=1
```

Attention : les variables de contrôles sont des variables **globales**.

Les variables de contrôle (2/2)

Remplacer la variable par sa valeur

Pour remplacer une variable de contrôle par sa valeur, on utilise les accolades { }.

Lorsqu'on utilise des accolades autour de la variable de contrôle, EViews remplace la variable par sa valeur.

Exemple

```
%x="gdp"  
series maSerie = %x  
series maSerie = {%x}
```

La deuxième ligne est équivalente à `series maSerie = "gdp"`.

La troisième ligne est équivalente à `series maSerie=gdp`

Programmer avec Eviews

Programmer avec Eviews

Structure d'un programme

Qu'est-ce qu'un programme

Un Programme

Un **programme EViews** est un fichier texte comprenant une liste d'instructions interprétées et exécutées par EViews une à une.

Structurer un programme

- Le programme doit comporter en en-tête le nom de l'auteur, une description ainsi que les dernières modifications apportées.
- le code doit être **aéré**, **indenté** et **commenté**.

Exécuter un programme

Deux modes :

- **verbose** : exécution pas à pas, le workfile et la barre de statut sont mis à jour au fur et à mesure de l'exécution du programme
- **quiet** : exécution en *background* → beaucoup plus rapide.

Programmer avec Eviews

Les Conditions et boucles

La condition `if ... else`

If ... Else

Une condition permet de tester une variable et de n'exécuter une partie de code que si la condition est fausse (ou vraie). La syntaxe d'EVIEWS est la suivante :

```
if <condition> then
    <instructions>
else
    <instructions>
endif
```

Les opérateurs booléens

Les opérateurs sont : =, >=, <, <>

Boucle For

For ... Next

La boucle **For** permet de répéter une opération un *certain* nombre de fois. La variable de contrôle peut-être **numérique** ou **une liste de chaîne de caractères**.

Exemple 1

```
for !i=1 to 10
    series maSerie{!i} = nrnd
next
```

Exemple 2

```
for %i brenteuro tchange eer38
    fetch(c=a,d=%mascotte) {%i}
next
```

Boucle While

While ... Wend

La boucle **while** permet d'exécuter une commande tant qu'une condition est satisfaite :

```
while <condition>  
  <instructions>  
wend
```

Exemple

```
while monGroupe.@count > 1  
  monGroupe.drop monGroupe.@seriesname(2)  
wend
```

TD : Créer une courbe de Phillips

TD : Créer une courbe de Phillips

Exemples de code

Exemple de code (1/3)

```
'Creer un workfile
wfccreate q 1990 2010

'Creer une series
series y=nrnd

'Creer 15 series
for !i=1 to 15
    series x{!i}=nrnd
next

'Regression deux a deux
for !i=1 to 15
    equation eq{!i}.ls y c x{!i}
next
```

Exemple de code (2/3)

```
'Creation d'un workfile
'=====
wfccreate q 1990 2010

'Creation d'un groupe contenant les variables
group xs

'Creation de 5 series
for %i GDP UNEMP INFL CPI M1
    series {%i}=nrnd
    xs.add {%i}
next

'Vecteur pour stocker les R2
vector(10) r2s

'Declaration d'un objet equation
equation eq
```

Exemple de code (3/3)

```
'Compteur du nombre d'equation
!rowcounter=1

'On fait les regressions de chaque variable sur l'autre
for !i=1 to xs.@count-1
    %iname = xs.@seriesname(!i)
    for !j=!i+1 to xs.@count
        %jname = xs.@seriesname(!j)
        eq.ls {%iname} c {%jname}
        r2s(!rowcounter) = eq.@r2
        !rowcounter = !rowcounter+1
    next
next
```

TD : Créer une courbe de Phillips

Instructions

TODO

1. Créer un workfile : `wfcreate`
2. Charger les données : `copy`, `fetch`
3. Estimer l'équation : par OLS `.ls`
4. Faire une prévision `.forecast`
5. Représenter les résultats graphiquement
6. Représenter les contributions économétriques

Questions ?