Programmation avec Python

Table of contents

Il s’agit d’une courte présentation de l’utilisation de Python sous Stata pour générer des graphiques. Après quelques éléments sur l’installation de Python et son utilisation avec Stata, 3 exemples de graphiques sont données avec les librairies **seaborn**, **plotnine** (wrapper de R/ggplot2) et **plotly** (graphiques intéractifs).  
Dans un futur plus ou moins proche, des graphiques tirés du (trè attendu chapitre 4) seront reproduits avec des fonctions de ces librairies Python.

**Quelques généralités sur l’utilisation de Python avec Stata**

* Depuis la version 16 de Stata (printemps 2019) il est possible d’utiliser Python de manière intéractive avec l’application.
* Une librairie Python SFI (https://www.stata.com/python/api16/Data.html) est mise à disposition pour favoriser cette interactivé. J’en ai pas eu besoin pour ce document, le code étant par ailleurs assez complexe.
* Niveau édition, une librairie Python officielle a été mise à disposition pour utiliser Stata sous **Jupyter** et **Spyder** si on utilise une version **Conda** de Python. Un noyau Stata [**Statakernel**] est disponible depuis de nombreuses années pour le notebook **Jupyter** et fonctionne très bien.
* Dans Stata, plusieurs commande on été implémentée pour gérer l’utilisation de python: **help python**
* Des éléments de correspondance Stata-Python pour la manipulation des données (librairiepandas) sont disponibles sur <https://pandas.pydata.org/docs/getting_started/comparison/comparison_with_stata.html#compare-with-stata>. Comme dans R, une fonction permet d’importer une base Stata en python.

|  |
| --- |
| Tip |
| Les macros Stata sont reconnues dans un code Python et appelées dans le code de manière classique avec des quotes pour les macro temporaire ou $ pour les macro en dur. |

Dans ce support, seul les graphiques seront exécutés avec Python, les manipulations en amont étant faite avec Stata. Seuls le chargement des bases seront effectués avec la fonction read\_Stata de la librairie **pandas**.

# 1. Installation de Python et des librairies [Windows]

## 1.1 Installation de python

Je ***déconseille vivement*** d’installer Python via une distribution *conda* pour générer seulement des graphiques. Un bug systématique se produira lorsque qu’un script *matplotlib* sera exécuté, faisant crasher la session de Stata. On peut remédier à ce problème facilement via une ligne de programme à insérer dans un fichier .profile, mais autant éviter un problème que d’y remédier.

**WARNING** Pour utiliser Python, vous devez avoir le même type de built pour les deux applications:

* Stata 64 bits <=> Python 64 bits
* Stata 32 bits <=> Pyhon 32 bits

A l’Ined, la version Stata est en 64b

* *Python 64* Installation Standard: <https://www.python.org/downloads/windows/>
* *Python 32* Installation Standard: <https://www.python.org/downloads/>

|  |
| --- |
| Installation de Python |
| * Cocher *“Add Python 3.10 to PATH”* (si version 3.10)  |  | | --- | | ***Variable environnement*** |  * Après avoir cliquer sur *Install Now* conserver l’installation de *pip* qui permet de gérer les librairies, en particulier de les installer et de les mettre à jour  |  | | --- | | ***Installation de pip***\* | |

**Vérification de la reconnaissance de Python sous Stata**

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

**Test exécution de python dans un .do**

python:   
print("Ined forever!!!!")  
1+1  
end

----------------------------------------------- python (type end to exit) -----  
>>> print("Ined forever!!!!")  
Ined forever!!!!  
>>> 1+1  
2  
>>> end   
-------------------------------------------------------------------------------

**Test de reconnaisance des macros Stata avec Python**

local x=5  
global m "Salut"  
  
python:   
print("$m")  
`x'/2  
end

----------------------------------------------- python (type end to exit) -----  
>>> print("$m")  
Salut  
>>> `x'/2  
2.5  
>>> end   
-------------------------------------------------------------------------------

## 1.2 Installation des libraries

**Vérification des librairies installées**

Dans le terminal windows pour garder le prompteur ouvert. [[1]](#footnote-36)

pip list

Installation d’une librairie. De préférence dans le terminal

pip install plotnine

Ou dans Stata:

! pip install plotnine

Librairies nécessaire pour la suite de cette documentation: **pandas** (installe numpy), **matplotlib**, **seaborn**, **plotnine**, **plotly**

pip install pandas  
pip install matplotlib  
pip install seaborn  
pip install plotnine  
pip install plotly

# 2. Deux exemples de graphiques

### 2.0.1 Exemple 1 avec seaborn

* Exécution d’un graphique avec la librairie seaborn. La fonction pairplot(), affiche une matrice de distribution croisée entre les variables de la base. L’option hue="forein" (comprendre *couleur*), permet de stratifier les distributions par
* Le graphique est enregistré avec la fonction savefig().
* Le graphique est affiché avec la fonction plt.show() de matplotlib.

\* Stata  
qui sysuse auto, clear  
qui keep price mpg displacement gear\_ratio turn foreign  
qui save auto, replace  
  
\* Python  
qui python:  
import pandas as pa   
import seaborn as sns  
import matplotlib.pyplot as plt  
  
df = pa.read\_stata('auto.dta')  
  
g = sns.pairplot(df, hue="foreign")  
g.savefig("D:/D/Marc/SMS/STATA INED/Graphiques/programmation/python/img/graph1.png")  
end  
  
\* Affichage du graphique sous commande windows  
\* ! "D:/D/Marc/SMS/STATA INED/Graphiques/programmation/python/img/graph1.png"

|  |
| --- |
| seaborn: fonction pairplot |

### 2.0.2 Exemple 2 avec plotly

* Un graphique intéractif avec la fonction **px.violin** de la librairie plotly
* Avec Stata, on a sélectionné la variable avec une macro temporaire, qui est reconnu tel quel dans le code python: y="`x'". Le graphique se mettra donc à jour automatiquement en changeant le nom de la variable dans la définition de la macro.

sysuse auto, clear  
local x price  
gen rep78b= rep78>3  
save auto, replace  
  
python:   
import pandas as pa   
import plotly.express as px  
  
df = pa.read\_stata('auto.dta')  
g = px.violin(df, y="`x'",   
 x="rep78b",   
 color="foreign", box=True, points="all")  
g.update\_layout(  
 title="Violin plot avec plotly",  
 xaxis\_title="REP78b",  
 yaxis\_title="`x'",  
 legend\_title="Foreign",  
 font=dict(  
 family="Arial",  
 size=24,  
 color="black"  
 ))   
g.write\_html("D:/D/Marc/SMS/STATA INED/Graphiques/programmation/python/img/graph2.html")   
fig.show()  
  
end

Warning: le package 'htmltools' a été compilé avec la version R 4.2.3

1. touche windows + R et taper cmd dans la fenêtre qui s’ouvre / taper cmd dans la fenêtre de recherche de la barre de tache. On peut attacher le prompteur à la barre de tâche pour l’ouvrir directement par la suite [↑](#footnote-ref-36)