

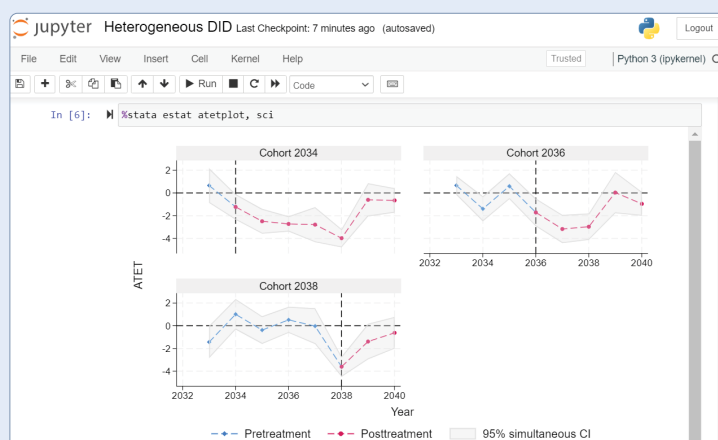
PyStata—Python and Stata integration

Call Stata from Python and call Python from Stata

Call Stata from Python

Access Stata from Python environments such as

- Jupyter Notebook and Lab
- Spyder IDE
- PyCharm IDE
- Windows Command Prompt
- macOS terminal
- Unix terminal



Work with IPython magic commands or API functions

Use magic commands in any IPython kernel-based environment

Run one Stata command with line magic

```
%stata regress y x1 x2 i.a##i.b
```

Or run a block of Stata commands with cell magic

```
%%stata
regress y x1 x2 i.a##i.b
margins a#b
predict yhat
```

Add options to cell magic to transfer data and results and alter settings.

Load Python data into Stata

```
%%stata -d mydata
```

Load Stata data into Python

```
%%stata -doutd newdata
```

Load Stata results into Python

```
%%stata -ret r -eret e
```

Use API functions anywhere you access Python

Load Python data into Stata

```
stata.pdataframe_to_data(mydata)
```

Run Stata commands in Python

```
stata.run("regress y x1 x2 i.a##i.b")
```

Or run a block of Stata commands

```
stata.run('''
regress y x1 x2 i.a##i.b
margins a#b
predict yhat
''')
```

Load Stata data into Python

```
newdata = stata.pdataframe_from_data()
```

Load Stata results into Python

```
r = stata.get_return()
e = stata.get_ereturn()
```

Add options to suppress output, use a subset of observations or variables, and more!

Call Python from Stata

- Use Python packages within Stata
 - Matplotlib for 3D graphs
 - Scrapy for scraping data
 - scikit-learn for machine learning
 - TensorFlow for deep learning
 - Much more

- Use the Stata Function Interface (sfi) Python module
 - Access Stata's core features in Python, including data, frames, macros, scalars, matrices, value labels, characteristics, and more
 - Store Python results back into Stata

Invoke Python interactively

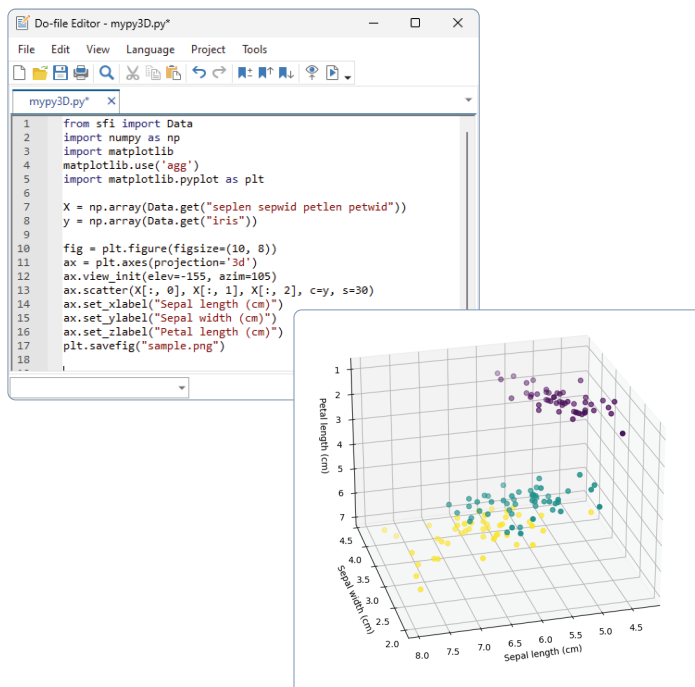
Type **python** in the Stata Command Window to enter the Python environment.

```
. python
python (type end to exit)
>>> import numpy as np
>>> b = np.arange(6).reshape(2,3)
>>> b
array([[0, 1, 2],
       [3, 4, 5]])
>>> b.sum(axis=0)
array([3, 5, 7])
>>> end
```

Run a Python script within Stata

Use the **python script** command to run a Python script directly from Stata.

- use <https://www.stata-press.com/data/r19/iris.dta>
- **python script mpy3D.py**



Embed Python code in a Stata script (do-file) or program (ado-file)

Combine Stata code with Python code in your do-files and ado-files.

```
Do-file Editor - mpy
File Edit View Language Project Tools
1 use https://www.stata-press.com/data/r19/iris, clear
2
3 python:
4 from sfi import Data
5 import numpy as np
6 from sklearn.svm import SVC
7
8 X = np.array(Data.get("seplen sepwid petlen petwid"))
9 y = np.array(Data.get("iris"))
10
11 svc_clf = SVC(gamma='auto')
12 svc_clf.fit(X, y)
13
14 y_pred = svc_clf.predict(X)
15
16 Data.addVarByte("irispr")
17 Data.store("irispr", None, y_pred)
18 end
19
20 tabulate iris irispr
Line: 22, Col: 1 CAP NUM OVR
```

```
Do-file Editor - mysvm
File Edit View Language Project Tools
1 program mysvm
2 version 19.5
3 syntax varlist, predict(name)
4
5 gettoken label feature : varlist
6
7 //call the Python function
8 python: dosvm("`label'", "`feature'", "`predict'")
9 end
10
11 version 19.5
12 python:
13 from sfi import Data
14 import numpy as np
15 from sklearn.svm import SVC
16
17 def dosvm(label, features, predict):
18     X = np.array(Data.get(features))
19     y = np.array(Data.get(label))
20
21     svc_clf = SVC(gamma='auto')
22     svc_clf.fit(X, y)
23
24     y_pred = svc_clf.predict(X)
25
26     Data.addVarByte(predict)
27     Data.store(predict, None, y_pred)
28 end
29
Line: 31, Col: 1 CAP NUM OVR
```

- use <https://www.stata-press.com/data/r19/iris.dta>
- **mysvm iris seplen sepwid petlen petwid, predict(irispr)**
- **tabulate iris irispr**