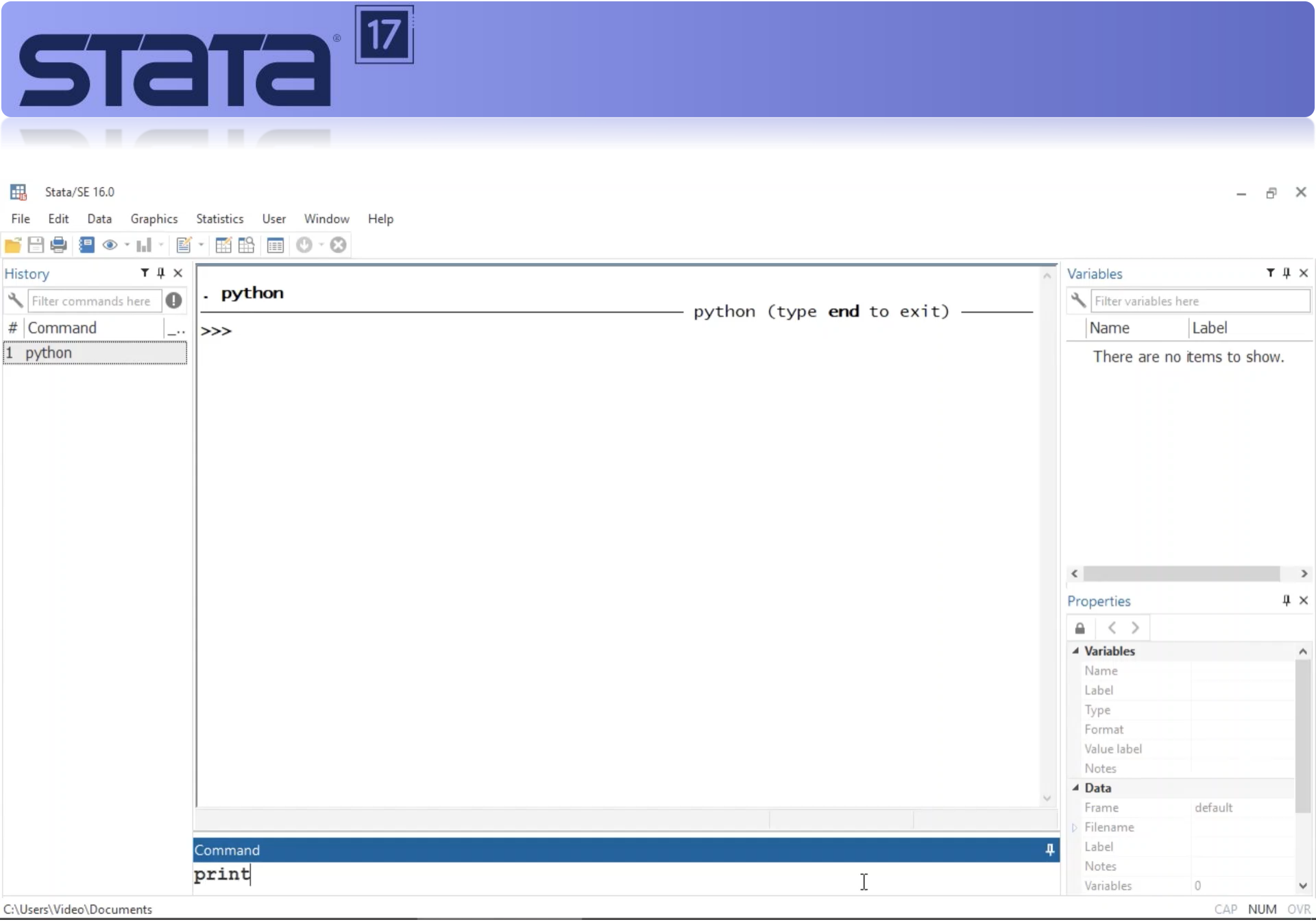
Using Stata in Python/ Jupyter Notebook

Meghan Cain | October 5th, 2021

You can download the slides and other materials here:
https://tinyurl.com/PyStata21


STATA 17
Stata/SE 16.0
File Edit Data Graphics Statistics User Window Help
History
Filter commands here
Command _...
1 python
. python
python (type end to exit)
>>>
Variables
Filter variables here
Name Label
There are no items to show.
Properties
Variables
Name
Label
Type
Format
Value label
Notes
Data
Frame default
Filename
Label
Notes
Variables 0
Command
print
C:\Users\Video\Documents
CAP NUM OVR

The **pystata** Python package

API functions

```
stata.run("didregress (satisfaction_score) (procedure), group(hospital_id) time(month)")
```

The magic commands

```
%stata didregress (satisfaction_score) (procedure), group(hospital_id) time(month)
```

The **pystata** Python package

API functions

```
stata.run("didregress (satisfaction_score) (procedure), group(hospital_id) time(month)")
```

- Slightly more functionality

The magic commands

```
%stata didregress (satisfaction_score) (procedure), group(hospital_id) time(month)
```

- Easier to use

The **pystata** Python package

API functions

```
stata.run("didregress (satisfaction_score) (procedure), group(hospital_id) time(month)")
```

- Slightly more functionality
- Available anywhere you can access Python

The magic commands

```
%stata didregress (satisfaction_score) (procedure), group(hospital_id) time(month)
```

- Easier to use
- Only available in IPython kernel-related environments

Use Stata in …

- Any interactive Python (IPython) kernel-related environment
  - Jupyter Notebook/ JupyterLab
  - Spyder IDE
  - PyCharm IDE
- Any Shell that can access Python
  - Windows Command Prompt
  - macOS terminal
  - Unix terminal

Use Stata in …

- Any interactive Python (IPython) kernel-related environment
  - Jupyter Notebook/ JupyterLab
  - Spyder IDE
  - PyCharm IDE
- Any Shell that can access Python
  - Windows Command Prompt
  - macOS terminal
  - Unix terminal

magic or API

Use Stata in …

- Any interactive Python (IPython) kernel-related environment
  - Jupyter Notebook/ JupyterLab
  - Spyder IDE
  - PyCharm IDE

magic or API

- Any Shell that can access Python
  - Windows Command Prompt
  - macOS terminal
  - Unix terminal

API

For more information

https://www.stata.com/python/pystata/

https://github.com/StataMeghan/PyStata/blob/main/JupyterPractice.ipynb

Workflow

1. Installation
2. Configuration
3. Launch Jupyter Notebook, etc.
4. Magic commands
5. API functions
6. Resources for advanced usage

Installation

- Stata 17.0+
- Python 3.4+
- Python packages: pandas, NumPy, IPython, Jupyter Notebook

In Command Prompt/Terminal:

```
> pip install pandas numpy ipython notebook
```

Installation

- Stata 17.0+
- Python 3.4+
- Python packages: pandas, NumPy, IPython, Jupyter Notebook

In Command Prompt/Terminal:

```
> py -m pip install pandas numpy ipython notebook
```

Launch Jupyter Notebook

```
> jupyter notebook
```

Configuration

- First, you need to know where Stata is on your computer.

```
. display c(sysdir_stata)
C:\Program Files\Stata17/
```

- We refer to this as STATA_SYSDIR

Configuration

1. Use the Python module stata_setup

Configuration

1. Use the Python module stata_setup
2. Add pystata's location to Python's module search path

Configuration

1. Use the Python module stata_setup
2. Add pystata's location to Python's module search path
3. Change the current working directory to pystata's location

Configuration

1. Use the Python module stata_setup
2. Add pystata's location to Python's module search path
3. Change the current working directory to pystata's location
4. Permanently add pystata's location to Python's module search path

The magic commands

`%pystata`

configure the system and display current system information and settings

`%stata`

execute Stata commands

`%mata`

execute Mata code

PyStata magic

```
%pystata status

%pystata set graph_show True|False [, perm]

%pystata set graph_size w|h #[in|px|cm] [, perm]

%pystata set graph_format svg|png|pdf [, perm]
```

Stata magic

`%stata` *command*

- Executes one line of Stata code, as it would in Stata's Command window

`%%stata`
commands

- Executes a block of Stata code, as it would in a Stata do-file

Stata magic

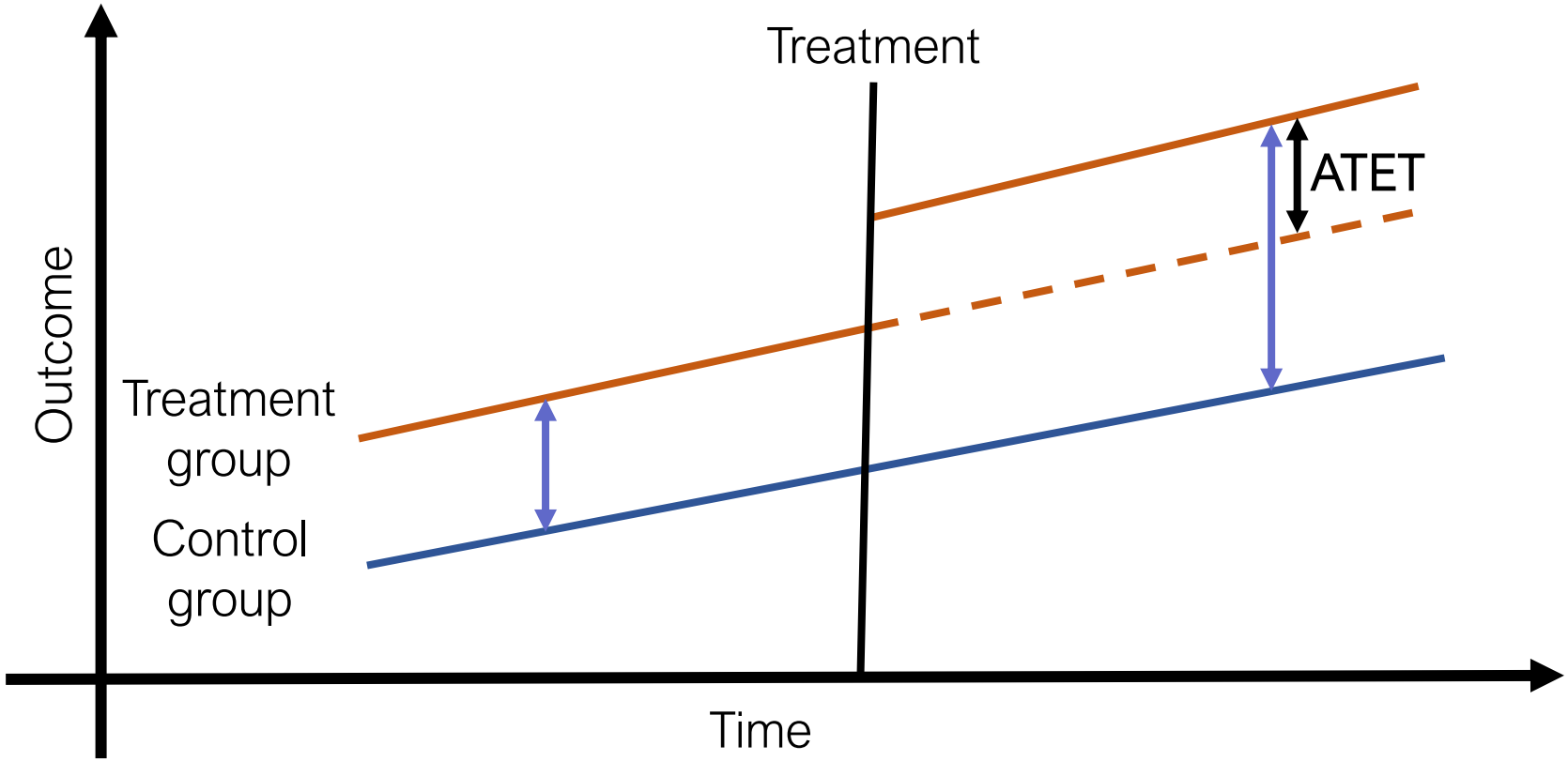
%%stata [-d DATA] [-f DFLIST|ARRLIST] [-force]
[-doutd DATAFRAME] [-douta ARRAY] [-foutd FRAMELIST]
[-fouta FRAMELIST] [-ret DICTIONARY] [-eret DICTIONARY]
[-sret DICTIONARY] [-qui] [-nogr] [-gw WIDTH] [-gh HEIGHT]

Python → Stata

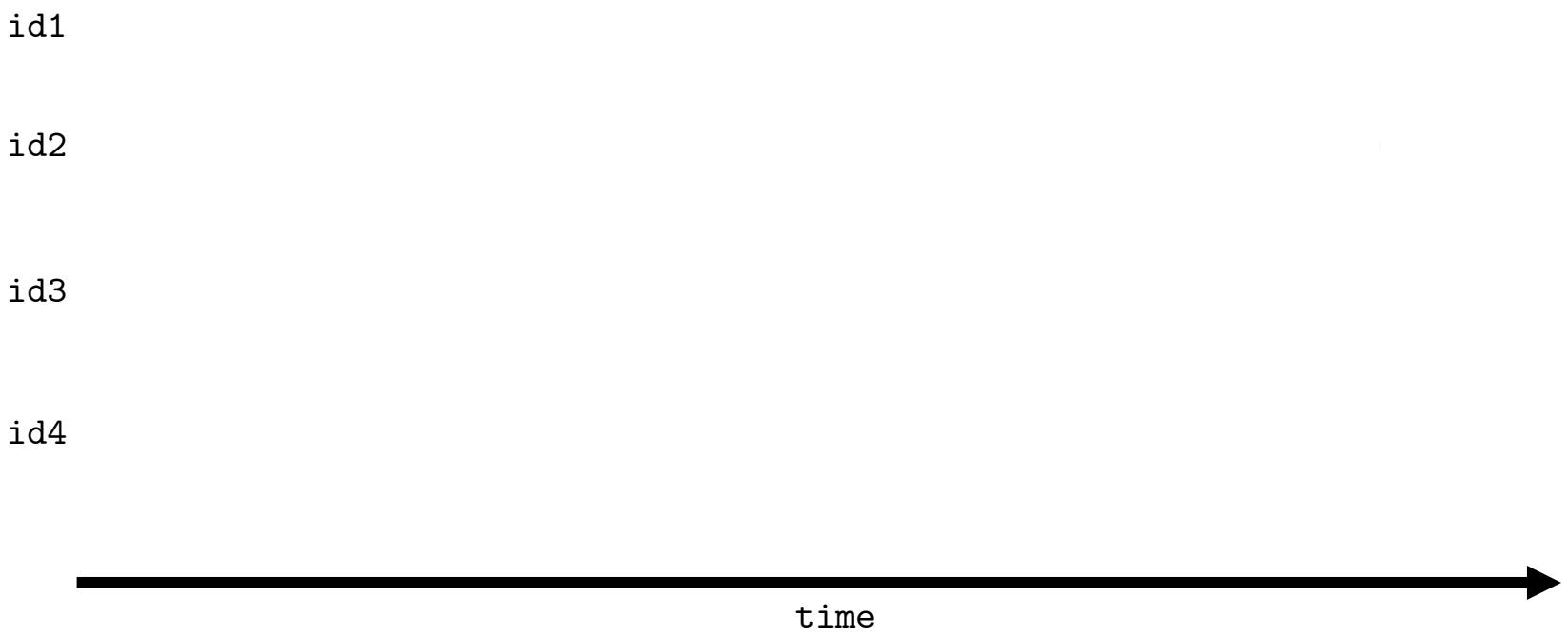
Stata → Python

Output

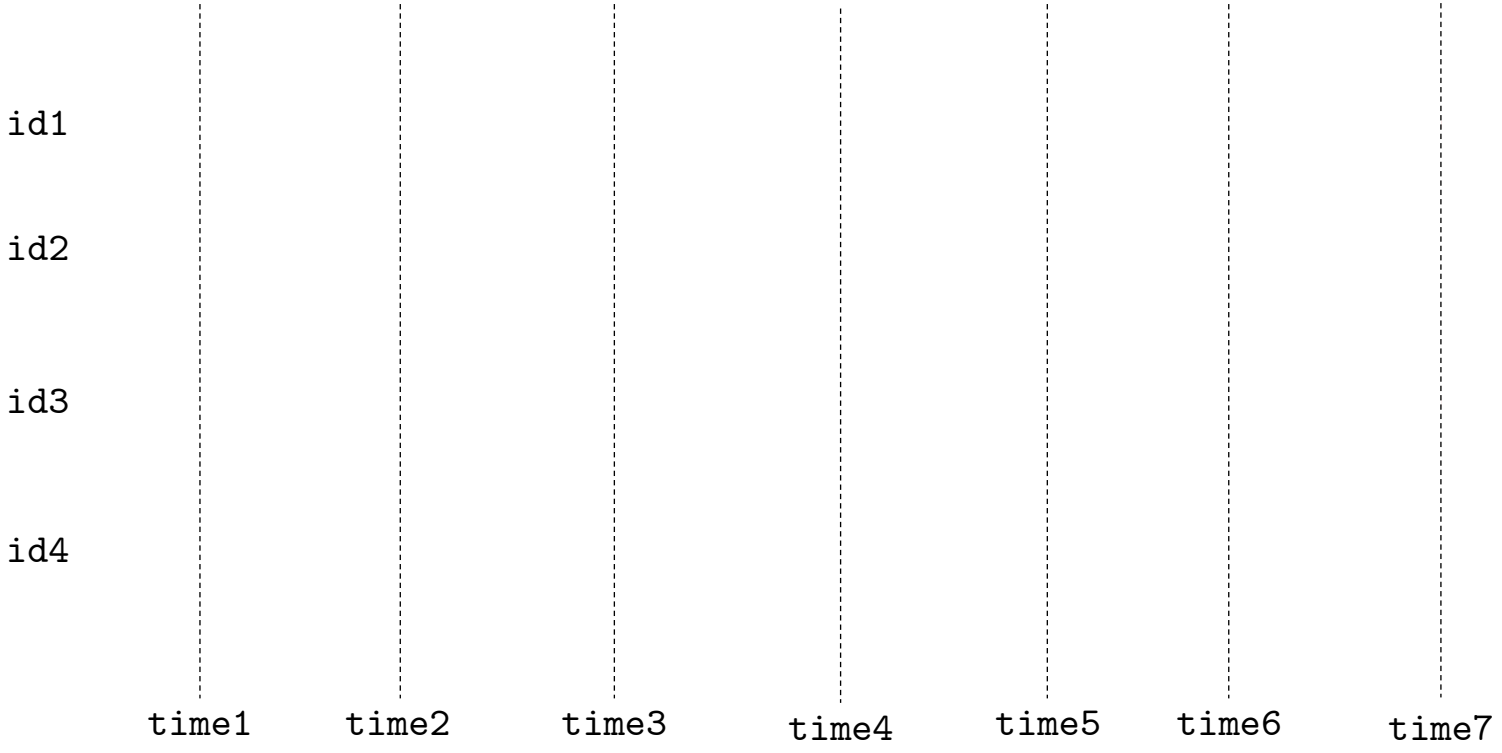
Difference in differences (DID)



Survival analysis



Interval-censored Cox models



$$h(t;\boldsymbol{x}) = h_0(t)\exp(\boldsymbol{x}\boldsymbol{\beta})$$

Mata magic

`%mata` *command*

- Executes one line of Mata code, similar to specifying `mata: istmt` within Stata

`%%mata` [`-m` ARRAYLIST] [`-outm` MATLIST] [`-qui`] [`-c`] *commands*

- Executes a block of Mata code, as it would in a Stata do-file

API functions

Two modules

`config`

- Configure the system and display current system information and settings

`stata`

- Interact with Stata

config

init(*edition*)
is_stata_initialized()
status()
set_graph_format(svg | png | pdf [, perm])
set_graph_size([*width*, *height*, perm])
set_graph_show(True | False [, perm])
set_output_file(*filename* [, replace])
close_output_file()

stata

run(*cmd* [, *quietly*, *echo*, *inline*])
nparray_to_data(*arr* [, *prefix*, *force*])
pdataframe_to_data(*df* [, *force*])
nparray_to_frame(*arr*, *stfr* [, *prefix*, *force*])
pdataframe_to_frame(*df*, *stfr* [, *force*])
nparray_from_data([*var*, *obs*, *selectvar*, *valuelabel*, *missingval*])
pdataframe_from_data([*var*, *obs*, *selectvar*, *valuelabel*, *missingval*])
nparray_from_frame(*stfr* [, *var*, *obs*, *selectvar*, *valuelabel*, …])
pdataframe_from_frame(*stfr* [, *var*, *obs*, *selectvar*, *valuelabel*,…])
get_return()
get_ereturn()
get_sreturn()

`nparray_from_data(`[*var*, *obs*, *selectvar*, *valuelabel*, *missingval*]`)`
`pdataframe_from_data(`[*var*, *obs*, *selectvar*, *valuelabel*, *missingval*]`)`

`var=None`, *integer*, *string*, or *list*

- Variables to access.

`obs=None`, *integer*, or *list*

- Observations to access.

`selectvar=None`, *integer*, or *string*

- Observations for which *selectvar!=0* will be selected.

`valuelabel=False` or `True`

- Use the value label when available.

`missingval=` *default_mis*, *number* or *string*

- If *missingval* is specified, all the missing values in the returned list are replaced by this value.

Resources for advanced usage

- The **Stata Function Interface (sfi)** module allows users to interact Python's capabilities with core features of Stata.
- Many of these functions are used in the background for the PyStata package.
- This will be your main tool if you would like to write wrapper modules or packages.
- For more information: https://www.stata.com/python/api17/

Thank you!

Questions?

You can download the slides and other materials here:
https://tinyurl.com/PyStata21

You can contact tech support at tech-support@stata.com