pyeviews: Python + EViews

WHITEPAPER AS OF 5/27/2022

Introduction

The purpose of the **pyeviews** package is to make it easier for EViews and Python to talk to each other, so Python programmers can use the econometric engine of EViews directly from Python. The Python package we've written uses COM to transfer data between Python and EViews. (For more information on COM and EViews, look at our whitepaper on the subject.)

Example

Here's a simple example going from Python to EViews. We're going to use the popular Chow-Lin interpolation routine in EViews using data created in Python. Chow-Lin interpolation is a regression-based technique to transform low-frequency data (in our example, annual) into higher-frequency data (in our example, quarterly) by using a higher-frequency series as a pattern for the interpolated series to follow. The quarterly interpolated series is chosen to match the annual benchmark series in one of four ways: first (the first quarter value of the interpolated series matches theannual series), last (same, but for the fourth quarter value), sum (the sum of the first through fourth quarters matches the annual series), and average (the average of the first through fourth quarters matches the annual series).

We're going to create two series in Python using the time series functionality of the **pandas** package, transfer it to EViews, perform Chow-Lin interpolation on our series, and bring it back into Python. The data are taken from Bloem *et al* in an example originally meant for Denton interpolation.

1. If you don't have Python, we recommend using miniforge, a Python installer created by the conda-forge community. After installation, open a miniforge command window and install **pyeviews** with the command:

   ```
   conda install pyeviews
   ```

   (In order to follow the following example you'll also need to install the plotting package **matplotlib** with `conda install matplotlib`.) Alternatively, go to the Python Package Index and get the **pyeviews** package by opening a Windows command line program and using the command:

   ```
   pip install pyeviews
   ```

   or by downloading the package, navigating to your installation directory, and issuing:

   ```
   python setup.py install
   ```

   Other options include miniconda or the Anaconda distribution itself, which will include most of the packages we'll need. After installing Anaconda, open a Windows command line

program (e.g., Command Prompt or PowerShell) and use the command:

```
conda install -c bexer pyeviews
```

to download and install **pyeviews**. Note that packages installed using `python setup.py` or the system `pip` instead of Anaconda's `pip` may not be accessible from within an Anaconda environment.

2. Start python and create two time series using pandas. We'll call the annual series "benchmark"and the quarterly series "indicator":

```
>>> import numpy as np

>>> import pandas as pa

>>> dtsa = pa.date_range('1998', periods = 3, freq = 'A')

>>> benchmark =
pa.Series([4000.,4161.4,np.nan],index=dtsa,
name = 'benchmark')

>>> dtsq = pa.date_range('1998q1', periods = 12, freq
='Q')

>>> indicator = pa.Series([98.2, 100.8, 102.2, 100.8, 99.,
101.6, 102.7, 101.5, 100.5, 103., 103.5, 101.5], index =
dtsq, name = 'indicator')
```

3. Load the **pyeviews** package and create a custom COM application object so we can customizeour settings. Set `showwindow` (which displays the EViews window) to True. Then call the `PutPythonAsWF` function to create pages for the benchmark and indicator series:

```
>>> import pyeviews as evp

>>> eviewsapp =
evp.GetEViewsApp(instance='new',
showwindow=True)

>>> evp.PutPythonAsWF(benchmark, app=eviewsapp)

>>> evp.PutPythonAsWF(indicator,
app=eviewsapp,newwf=False)
```

Behind the scenes, **pyeviews** will detect if the DatetimeIndex of your **pandas** object (if you have one) needs to be adjusted to match EViews' dating customs. Since EViews assigns dates to be the beginning of a given period depending on the frequency, this can lead to misalignment issues and unexpected results when calculations are performed. For example, a DatetimeIndex with an annual'A' frequency and a date of 2000-12-31 will be assigned an internal EViews date of 2000-12-01. In this case, **pyeviews** will adjust the date to 2000-01-01 before pushing the data to EViews.

4. Name the pages of the workfile:

```
>>> evp.Run('pageselect Untitled', app=eviewsapp)

>>> evp.Run('pagerename Untitled annual', app=eviewsapp)

>>> evp.Run('pageselect Untitled1', app=eviewsapp)

>>> evp.Run('pagerename Untitled1
quarterly',app=eviewsapp)
```

5. Use the EViews "copy" command to copy the benchmark series in the annual page to the quarterly page, using the indicator series in the quarterly page as the high-frequency indicatorand matching the sum of the benchmarked series for each year (four quarters) with the matching annual value of the benchmark series:

```
>>> evp.Run('copy(rho=.7, c=chowlins, overwrite)
annual\\benchmark quarterly\\benchmarked
@indicatorindicator', app=eviewsapp)
```

6. Bring the new series back into Python:

```
>>> benchmarked = evp.GetWFAsPython(app=eviewsapp,
pagename='quarterly', namefilter='benchmarked')

>>> print benchmarked

              BENCHMARKED

1998-01-01   867.421429

1998-04-01  1017.292857

1998-07-01  1097.992857

1998-10-01  1017.292857

1999-01-01   913.535714

1999-04-01  1063.407143

1999-07-01  1126.814286

1999-10-01  1057.642857

2000-01-01  1000.000000

2000-04-01  1144.107143

2000-07-01  1172.928571

2000-10-01  1057.642857
```

7. Release the memory allocated to the COM process (this does not happen automatically ininteractive mode). This will close down EViews:

```
>>> eviewsapp.Hide()

>>> eviewsapp = None

>>> evp.Cleanup()
```

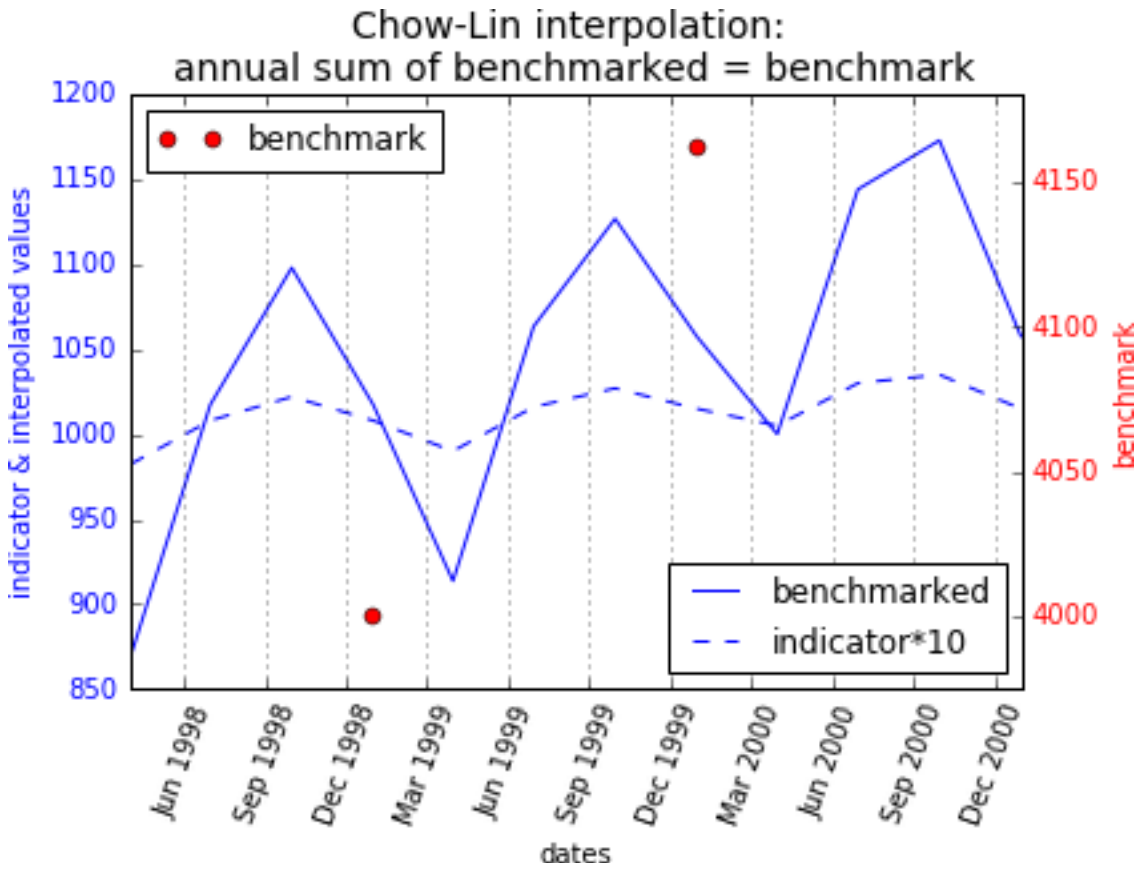
Note that if you choose not to create a custom COM application object (the

`GetEViewsApp`function), you won't need to use the first two lines in the last step. You only need to call `Cleanup()`. If you create a custom object but choose not to show it, you won't need to usethe first line (the `Hide()` function).

8. If you want, plot everything to see how the interpolated series follows the indicator series:

```
>>> # load the matplotlib package to plot

>>> import matplotlib.pyplot as plt

>>> # reindex the benchmarked series to the end of the
quarter so the dates match those of the indicator series


>>> benchmarked_reindexed =
pa.Series(benchmarked.values.flatten(), index =
benchmarked.index + pa.DateOffset(months = 3, days = -1))

>>> # plot

>>> fig, ax1 =

plt.subplots()

plt.xticks(rotation=70)

ax1.plot(benchmarked_reindexed, 'b-', label='benchmarked')

# multiply the indicator series by 10 to put it on the
samescale as the benchmarked series

ax1.plot(indicator*10, 'b--',

label='indicator*10')ax1.set_xlabel('dates')

ax1.set_ylabel('indicator & interpolated
values',color='b')

ax1.xaxis.grid(True)

for tl in

    ax1.get_yticklabels():

    tl.set_color('b')

plt.legend(loc='lower

right')ax2 = ax1.twinx()

ax2.set_ylim([3975, 4180])

ax2.plot(benchmark, 'ro',

label='benchmark')

ax2.set_ylabel('benchmark', color='r')
```

```
for tl in

    ax2.get_yticklabels():

    tl.set_color('r')

plt.legend(loc='upper left')

plt.title("Chow-Lin interpolation: \nannual sum of
benchmarked = benchmark", fontsize=14)

plt.show()
```



References

Bloem, A.M, Dippelsman, R.J. and Maehle, N.O. 2001 Quarterly NationalAccounts Manual–Concepts, Data Sources, and Compilation.
IMF. http://www.imf.org/external/pubs/ft/qna/2000/Textbook/index.htm

List of Functions

Public:

pyeviews.GetEViewsApp(version='EViews.Manager', instance='either', showwindow=False)

Define a custom EViews COM application object with specified options.

Parameters:

version: {'EViews.Manager', 'EViews.Manager.9', 'EViews.Manager.8','EViews.Manager.1'}, optional

Select the version of EViews to be used. 'EViews.Manager' will use the latest installed version of EViews, 'EViews.Manager.9' will use version 9, 'EViews.Manager.8' will use version 8, and 'EViews.Manager.1' will use version7.

instance: {'new', 'either', 'existing'}, optional

The instance type for the EViews COM application. 'new' opens a new EViews application, 'either' uses an existing application, or, if none exists, opens a newone, and 'existing' uses an existing application.

showwindow: bool, optional

Display the EViews window.

Returns:

out: EViews COM application

A user-defined COM application object.

pyeviews.PutPythonAsWF(object, app=None, newwf=True)

Determine the type of object and push into EViews with specified options. Calls _BuildFromPython or _BuildFromPandas.

Parameters:

object: pandas DataFrame, Series, MultiIndex, DatetimeIndex, or RangeIndex; list, dict, or numpy array

The Python or pandas object to be pushed into EViews.

app: EViews COM application,

optionalCOM application

object

newwf: bool, optional

If False, creates a new page in an already existing workfile or a new workfile ifnone exists.

Returns:

out: EViews series, panel, or an empty workfile with the appropriate Range set (for a pandas Index)

Pandas Series and DataFrame attributes are automatically copied into EViews series attributes.

pyeviews.GetWFAsPython(app=None, wfname='', pagename ='', namefilter='*')

Pull data from EViews into Python with specified options.

Parameters:

app: EViews COM application, optional
A user-defined COM application object.

wfname: string, optional

Name of the EViews workfile to pull data from. Must be the full path name. Ifno workfile is specified, the currently open workfile will be used.

pagename: string, optional

Name of the EViews workfile page to be created.

namefilter: string, optional

Base name for series to be pulled.

Returns:

out: pandas DataFrame

A pandas DataFrame containing the series objects pulled from EViews. EViews series attributes are automatically copied into the attributes of the DataFrame.

pyeviews.Run(command, app=None)

Run an EViews command directly from Python.

Parameters:

command: string

The full command to be passed to EViews.

app: EViews COM application, optional

A user-defined COM application object.

pyeviews.Get(objname, app=None)

Return single data values from an EViews workfile.

Parameters:

objname: string

A single piece of EViews data (e.g., a scalar value or string value such as"@pagename."

app: EViews COM application, optional

A user-defined COM application object.

Returns:

out: string

pyeviews.Cleanup(app=None)

Clear the memory allocated to the COM process. This is not done automatically in interactive mode.

Parameters:

app: EViews COM application, optional

COM application object with memory to be released. If no app is specified, the global app is substituted.

Private:

pyeviews._BuildFromPython(objectlength, newwf=True)

Creates the CREATE or PAGECREATE command for a new compatible EViews workfile.

Parameters:

objectlength: integer

The length of the Python object (list, dict, or numpy array) to be pushed toEViews.

newwf: bool, optional

If False, creates a new page in an already existing workfile or a new workfile ifnone exists.

Returns:

out: string

A string with the create command for a workfile or page.

pyeviews._BuildFromPandas(object, newwf=True)

Creates the CREATE or PAGECREATE command for a new compatible EViews workfile.

Parameters:

object: pandas object

The Python pandas object (series, DataFrame, MultiIndex, DatetimeIndex, or RangeIndex) to be pushed to EViews.

newwf: bool, optional

If False, creates a new page in an already existing workfile or a new workfile ifnone exists.

Returns:

out: string

A string with the create command for a workfile or page.

pyeviews._CheckReservedNames(names)

Check that none of the data structure names being pushed to EViews are the reserved names“c” or “resid.”

Parameters:

names: list of object names

pyeviews._GetApp(app=None)

Determine the use of either the user-defined EViews COM application object or the globalapplication object.

Parameters:

app: EViews COM application,

optionalCOM application

object

Returns:

app: EViews COM

application COM

application object

Frequency conversions

Python pandas frequency	EViews frequency
AS, A, BAS,BA *	A
QS, Q, BQS, BQ	Q
MS, M, BMS, BM, CBMS, CBM	M
W	W
D	D7
B	D5
C	D(*day begin, day end*)
H, BH *	H(*day begin-day end, time min-time max*)
T, min *	Min(*day begin-day end, time min-time max*)
S *	Sec(*day begin-day end, time min-time max*)
L, ms, U, us, N	Not supported

* = Includes custom frequencies (2A, 6H, 5min, 30S, etc.). See EViews documentation for full list.