

pyeviews: EViews + Python

はじめに

- **pyeviews** パッケージは、EViews と Python の連携を容易にするためのツールです。EViews の計量経済学エンジンを Python から利用できるようになります。
- Python パッケージは、COM を利用してデータを Python と EViews の間でやり取りします。COM についての詳細は下記をご参照ください。

▼EViews COM Automation

http://www.eviews.com/download/whitepapers/EViews_COM_Automation.pdf

環境構築

- Anaconda のインストールを推奨します。下記からダウンロードいただけます。

▼Anaconda Individual Edition

<https://www.anaconda.com/products/individual>

- Anaconda インストール後、Anaconda Prompt など下記のコマンドを実行します。**pyeviews** がダウンロード・インストールされます。

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

補足：

- Anaconda を使用しない場合、下記を実行してパッケージをインストールします。

```
• pip install pyeviews
```

または、パッケージをダウンロード後に、パッケージのディレクトリで下記を実行します。

```
python setup.py install
```

使用例 Chow-Lin interpolation

- Chow-Lin interpolation は、補間された系列などの高頻度系列を使って、低頻度データ（例では年次）を高頻度データ（例では四半期）に変換するための、回帰をベースにした手法です。
- 補間された四半期系列は、以下の 4 つの内のいずれか 1 つの方法で年次のベンチマーク系列と一致するように選択されます。
 - first（補間系列の最初の四半期値を年次系列と一致させる）
 - last（補間系列の 4 番目の四半期値を年次系列と一致させる）
 - sum（1 番目から 4 番目の四半期値の合計を年次系列と一致させる）
 - average（1 番目から 4 番目の四半期値の平均を年次系列と一致させる）

- **pandas** パッケージの時系列機能を使って、2つの系列を Python に作成します。そして、系列を EViews に移して Chow-Lin interpolation を実行し、結果を Python に返します。
- Denton interpolation の例題のデータ (Bloem *et al*) を使用します。
- 本マニュアルのコマンドを記述した pyeviews.ipynb を配布しています。

1. Python を起動して、**pandas** を使用し 2 つの系列を作成します。年次系列を「**benchmark**」、四半期系列を「**indicator**」とします。

```
>>> 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')
```

2. **pyeviews** パッケージをロードし、**custom COM application object** を作成すると、設定をカスタマイズできるようになります。**showwindw** (EViews ウィンドウの表示) の設定を **True** にします。そして、**PutPythonAsWF** 関数を呼び出して、**benchmark** と **indicator** の系列のページを作成します。

```
>>> import pyeviews as evp
>>> evIEWSapp = evp.GetEViewsApp(instance='new', showwindow=True)
>>> evp.PutPythonAsWF(benchmark, app=evIEWSapp)
>>> evp.PutPythonAsWF(indicator, app=evIEWSapp, newwf=False)
```

- **pyeviews** は、**pandas** オブジェクトの **DatetimeIndex** を EViews の日付設定に合わせるために調整する必要があるかどうかを裏で検出しています。
- EViews は、頻度に応じて特定の期間の開始に日付を割り当てます。これにより、予期しない問題や不整合が起こる場合があります。

例：年次頻度 A の日付データ 2000 年 12 月 31 日の **DatetimeIndex** は、EViews 内部の日付 2000 年 12 月 1 日に割り当てられます。この場合、**pyeviews** は、データを EViews に送る前に日付を 2000 年 1 月 1 日に調整します。

3. EViews のワークファイルの各ページに名前を付けます。

```
>>> 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)
```

4. EViews の **copy** コマンドで、annual ページにある **benchmark** 系列を **quaretely** ページにコピーします。**quaretely** ページにある **indicator** 系列を高頻度のインディケータとして使用し、**benchmark** 系列の各年の合計（4 期の四半期）を、**benchmark** 系列の年次の値と一致させます。

```
>>> evp.Run('copy(rho=.7, c=chowlins, overwrite) annual¥¥benchmark quarterly¥¥
benchmarked @indicator indicator', app=evIEWSapp)
```

5. 新しい系列を Python に戻します。

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

- 出力結果

	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

6. COM process に割り当てられたメモリを解放します（インタラクティブモードでは自動解放されません）。EViews が閉じます。

```
>>> evIEWSapp.Hide()
>>> evIEWSapp = None
>>> evp.Cleanup()
```

- custom COM application object（GetEViewsApp 関数）を作成しなかった場合は、終了時に 1-2 行目を実行する必要はありません。**Cleanup()**のみを実行してください。
- カスタムオブジェクトを作成したけれど表示しなかった場合は、1 行目の **Hide()**関数を実行する必要はありません。

7. 結果をグラフにプロットし、補間された系列がどのように **indicator** 系列に追従するのかを表示します。

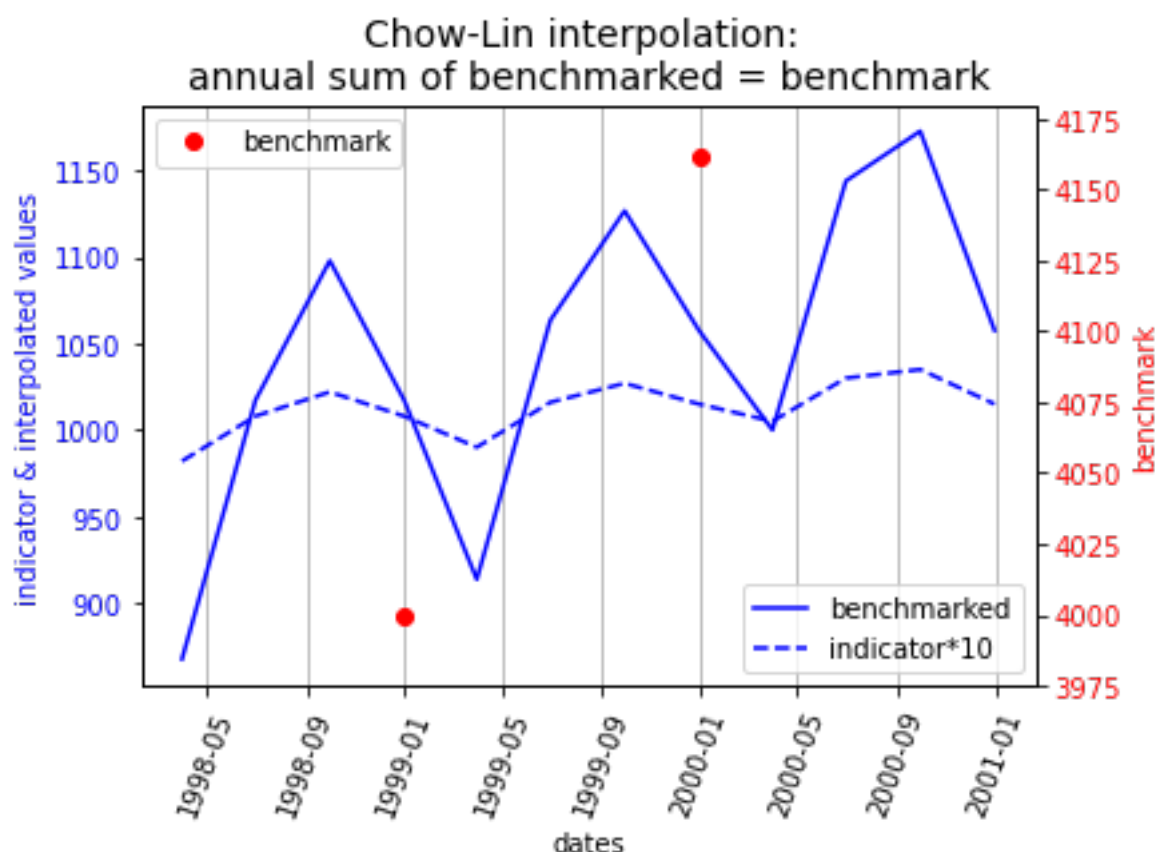
```
>>> # load the matplotlib package to plot
>>> import matplotlib.pyplot as plt

>>> # 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='benchmark')

>>> # multiply the indicator series by 10 to put it on the same scale as the
benchmark 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()
```

- 表示結果



参考文献

Bloem, A.M, Dippelsman, R.J. and Maehle, N.O. 2001 Quarterly National Accounts Manual–Concepts, Data Sources, and Compilation.

IMF. <http://www.imf.org/external/pubs/ft/qna/2000/Textbook/index.html>

機能リスト

Public

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

指定したオプションを使って custom EViews COM application オブジェクトを定義します。

Parameters:

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

使用する EViews のバージョンを選択します。

‘EViews.Manager’: インストールされた最新のバージョンを使用します。

‘EViews.Manager.9’: バージョン 9 を使用します。

‘EViews.Manager.8’: バージョン 8 を使用します。

‘EViews.Manager.1’: バージョン 7 を使用します。

instance: {‘new’, ‘either’, ‘existing’}, optional

EViews COM application のインスタンスの形式です。

‘new’: 新規で EViews アプリケーションを起動します。

‘either’: 現在起動しているアプリケーションを使用します。起動していない場合は、新規で EViews アプリケーションを起動します。

‘existing’: 現在起動しているアプリケーションを使用します。

showwindow: bool, optional

EViews のウィンドウを表示します。

- **Returns:**

out: EViews COM application

ユーザ定義の COM applicatio

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

オブジェクトのタイプを決定し、指定したオプションを使って EViews に送信します。_BuildFromPython または _BuildFromPandas を呼び出します。

- **Parameters:**

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

Python または pandas のオブジェクトを EViews に送ります。

app: EViews COM application, optional

COM application object

newwf: bool, optional

False の場合、起動しているワークファイル（ない場合は新規ワークファイル）に新規ページを作成します。

- **Returns:**

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

Pandas 系列と DataFrame 属性 が自動的に EViews の系列属性にコピーされます。

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

EViews から Python に指定したオプションでデータを引き渡します。

- **Parameters:**

app: EViews COM application, optional

ユーザ定義の COM application object.

wfname: string, optional

データを引き出すために EViews のワークファイルに名前を付けます。フルパス名の必要があります。ワークファイルが指定されない場合、現在開いているワークファイルが使用されます。

pagename: string, optional

作成するワークファイルのページの名前。

namefilter: string, optional

引き渡される系列のベース名。

- **Returns:**

out: pandas DataFrame

EViews から引き渡された系列オブジェクトを含んでいる pandas DataFrame。EViews の系列属性は自動的に DataFrame の属性にコピーされます。

pyeviews.Run(command, app=None)

EViews のコマンドを直接 Python から実行します。

- **Parameters:**

command: string

EViews に送るフルコマンド

app: EViews COM application, optional

ユーザ定義 の COM application object

pyeviews.Get(objname, app=None)

EViews ワークファイルから単一のデータ値を返します。

- **Parameters:**

objname: string

単一の EViews データ（例：スカラー値、「@pagename」のような文字列データ値）

app: EViews COM application, optional

ユーザ定義の COM application object

- **Returns:**

out: string

pyeviews.Cleanup(app=None)

COM proces に割り当てられたメモリを解放します（インタラクティブモードでは自動解放されません）。

- **Parameters:**

app: EViews COM application, optional

解放するメモリのある COM application object。指定しない場合は、global app で代用されます。

Private:

pyeviews._BuildFromPython(objectlength, newwf=True)

互換性のある新規 EViews ファイルの CREATE または PAGECREATE コマンドを作成します。

- **Parameters:**

objectlength: integer

EViews に送る Python オブジェクト(`list`, `dict`, `numpy array` など) の長さ。

newwf: bool, optional

`False` の場合、作成済みのワークファイル（存在しない場合は新規ワークファイル）に新規ページを作成します。

- **Returns:**

out: string

ワークファイルやページを作成するコマンドを含む文字列

pyeviews._BuildFromPandas(object, newwf=True)

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

- **Parameters:**

object: pandas object

EViews に送信される Python pandas オブジェクト(`series`, `DataFrame`, `MultiIndex`, `DatetimeIndex`, `RangeIndex`)。

newwf: bool, optional

`False` の場合、作成済みのワークファイル（存在しない場合は新規ワークファイル）に新規ページを作成します。

- **Returns:**

out: string

ワークファイルやページを作成するコマンドを含む文字列

pyeviews._CheckReservedNames(names)

EViews 送信されるデータ構造名が予約名「c」または「resid」でないことを確認します。

- **Parameters:**

names: list of object names

pyeviews._GetApp(app=None)

ユーザ定義の EViews COM application object または global application object のどちらかの使用を決定します。

- **Parameters:**

app: EViews COM application, optional

COM application object

- **Returns:**
app: EViews COM application
COM application object

頻度の変換

Python pandas の頻度	EViews の頻度
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	サポートしていません

※: カスタムの頻度 (2A, 6H, 5min, 30S など) を含む。すべてのリストは EViews マニュアルをご参照ください。