Python のすすめ

神戸大学経済学研究科　春山 鉄源　　2020 年 8 月 12 日

１．はじめに

Python はプログラミング言語である。文系の経済学の学生が理系のプログラミングを学ぶとなると「なぜプログラミング？」と思うかも知れない。過去にも同じような質問を問うた経済学部の卒業生はいたと思われる。例えば，Excel のようなスプレッドシートのソフトは 1980 年代からあり，使い方を学ぶ際「なぜ？」と思った学生もいたことだろう。しかし今では Word, Excel, PowerPoint は，大学卒業生にとって当たり前のスキルになっている。同じように，AI（人工知能）やビッグデータが注目を集める社会では，ある程度のプログラミング能力も経済学部卒業生にとって当たり前のスキルになると予測される。実際，文系出身でプログラミングとは縁のなかったある大手新聞社の記者の話では，社内で Python の研修を受けており，仕事に活かすことが期待されているという。また 2020 年度からは小学校においてプログラミング的思考を育成する学習指導要領が実施され，続いて中高教育でもプログラミングに関する内容・科目が充実される予定である。このようにプログラミングのスキルの重要性は益々大きくなると思われる。

「なぜ Python？」プログラミング言語は無数に存在し，それぞれ様々な特徴があり，お互いに影響し合い進化している。その過程で，広く使われ出す言語もあれば廃れていく言語もある。その中で Python は，近年注目を集める言語となっている。Python の人気はどこにあるのだろう。まず最初の理由は無料ということである。経済学研究でよく使われる数十万円するソフトと比べると，その人気の理由は理解できる。しかし計量経済学で広く使われる R を含めて他の多くの言語も無料であり，それだけが理由ではない。人気の第 2 の理由は，汎用性である。Python はデータ分析や科学的数値計算だけではなく，ゲーム（ゲーム理論ではない），画像処理や顔認識にも使われる。また PC やスマートフォンの様々な作業の自動化に使うことも可能である。第 3 の理由は，

学習コストが比較的に低いことである。Python のコードは英語を読む・書く感覚と近いため，他の言語と比較して可読性の高さが大きな特徴である。また頻繁に出てくるコードの中で日本語の助詞「の」や動詞「する」と対応している箇所もある。もちろん，Python の文法や基本的な関数を覚える必要があるが，相対的に最も初心者に易しい言語と言われる。他にも理由はあるが，Python は IT 産業だけではなく金融・コンサルティング・保険・医療などの幅広い分野で使われており，データ分析の重要性が増すごとにより多くの産業で使われると思われる。卒業後，経済学部の大多数の卒業生は幅広い産業で働くことになる。社会全体で注目され，今後より多くの産業で使われることが予測される言語を学ぶことは経済学部生にとって有意義ではないだろうか。

本稿の目的は，簡単な回帰分析を通して Python の使い方を紹介することである。ページ数の制限があるため必要な部分だけを説明するが，詳細は参考文献として挙げた「Python で学ぶ入門計量経済学」を参考にしてほしい。また本稿で使うデータやコードは全て最後に挙げた GitHub のレポジトリーで公開している。以下ではコードをブロック毎に説明するが，Jupyter Notebook のコード・セルで実行することを想定している。[1]

２．オークンの法則

回帰分析の例として，日本の 1994 年 Q1 から 2020 年 Q1 の四半期データを使い，次式で与えられるオークンの法則を考察する。

GDP の前期比成長率（%） $= a - b \times$ 失業率の変化（%ポイント）　　（式１）

ここで$a, b > 0$。この式は GDP 成長率と失業率の変化には負の関係があることを示している。労働者が生産投入に使われることを考えると，両変数の負の関係は直感的に理解できる。例えば，失業率が上昇すると雇用者数は減少し産出

[1] Python と Jupyter Notebook のインストールには Anaconda を勧めする。https://www.anaconda.com/products/individual からインストールできる。また Jupyter Notebook の基本的な使い方の説明については，インターネットで検索すると多くのサイトが出てくるので，そちらに譲ることにする。

量も縮小するため成長率は下落する。では失業率の変化が 1%ポイントの場合，GDP の成長率は平均で何%になるのだろうか。この問いに答えるためにはパラメータ a と b を推計する必要がある。

３．データの読み込み

使用するデータはウェブ上に準備してあるので次のコードで読み込むことができる。

```
1. import pandas as pd
2. from statsmodels.formula.api import ols
3.
4. url = ‘https://bit.ly/2XsaUwK’
5. df = pd.read_csv(url, ,index_col='index',parse_dates=True)
```

Python には機能を拡張するための「プラグイン」のようなものがあり，それらはパッケージ，サブパッケージ，モジュールなどの名前で呼ばれる。[2] 1 行目では，エクセルのようにデータを扱うための pandas というパッケージを pd として（as）読み込む（import）。コードを英語のように読むことができることに気付くのではないだろうか。2 行目では，回帰分析に使う statsmodels というパッケージを読み込む。この行を理解するために「.」を助詞「の」に読み替えると分かりやすい。statsmodels のサブ・パッケージである formula のモジュール api から（from）ols を読み込む（import）。名前が示すように ols を使って最小二乗法の自動計算を行う。「神戸大学」のような単なる文字列を作る場合は対象を ‘’もしくは“”で囲む。この方法を利用して，4 行目はファイルがあるウェブ・アドレスを文字列にし変数 url に割り当てる。[3]「=」は左辺を右辺の変数に「割り当てる」という意味である。[4] 5 行目は，pd（pandas）

[2] パッケージはある目的のための機能からなる「プラグイン」のようなものだが，サブ・パッケージとはその一部であり，モジュールはさらに細分化された一部と思えば良いだろう。
[3] ファイルは GitHub 上にありアドレスが長いため URL を短くするサービスを使って表示している。
[4] 「割り当てる」という意味を簡単に説明すると，右辺の文字列が PC のメモ

の関数 read_csv()を使いウェブ上にある.csv ファイルを読み込み，データを変数 df に割り当てている。この関数には「オプション」の役割をする引数と呼ばれるものが追加されている。index_col=index は csv ファイルにある index という列を df の行ラベルに指定しており，その行ラベルを時系列データ用に設定するための引数が parse_dates=True である。

変数 df には次の 2 変数が含まれており，161x2 のスプレッド・シートを想像すれば良いだろう。

- gdp_growth：実質 GDP の前期比成長率（%;（式 1）の左辺）
- u_deviation：失業率の変化（%ポイント;（式 1）の右辺）

内容を確認するには次のコードが便利である。

```
df.head()
```

df の最初の 5 行（head()）を表示するコードであり，右のような出力となる。行ラベルは毎四半期の最後の日付となっている。また最初の行の NaN（Not a Number の略）は，成長率と失業の変化を計算した際に発生した欠損値を示す。

index	gdp_growth	u_deviation
1994-03-31	NaN	NaN
1994-06-30	-0.487152	-0.026667
1994-09-30	1.133380	0.143333
1994-12-31	-0.412296	-0.053333
1995-03-31	1.149186	0.086667

4．図示と簡単な計算

「Python の全てはオブジェクトである」と言われる。オブジェクトとは様々なデータとしての情報や関数からなる「集合体」であり，正確には，この場合の関数はメソッドと呼ばれる。df を使って説明すると，標本の観測値だけではなく，変数の数や標本の大きさがデータである。メソッドの例としては，上で使った head()がある。head は「最初の 5 行の表示」を意味し，() の部分は

リー上に保存される実体であり，左辺の変数 url は単なる参照記号である。従って，同じ実体に複数の参照記号を割り当てることも可能となる。

「する」と理解すれば良い。

メソッドのもう１つの例として mean()があり平均を簡単に計算できる。

```
df.mean()
```

右のような出力となる。gdp_growth の値は前期比の平均成長率であるため，年率換算すると GDP は平均$(1+0.0021)^4-1\approx 0.00844$（0.844%）で成長している。同じようなメソッドに標準偏差を計算する std()がある。

```
gdp_growth      0.210336
u_deviation    -0.004135
```

```
df.std()
```

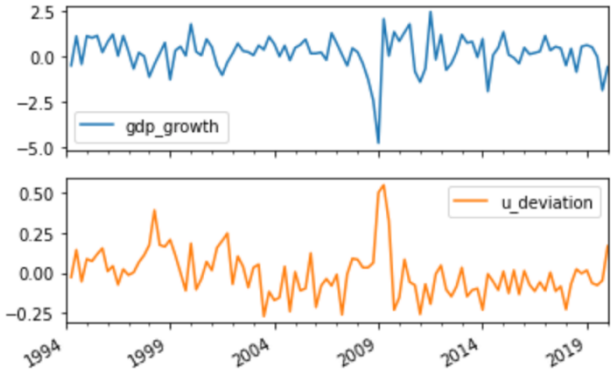
右の計算結果からわかるように，失業率の変化は比較的に小さい。これは日本の雇用制度を反映しており，aとbの推定値に影響を及ぼすことになる。

```
gdp_growth      0.961647
u_deviation     0.145540
```

また図示するためのメソッド plot()も非常に便利である。

```
df.plot(subplots=True);
```

引数の subplots=True は図を上下に配置することを指定しており，右の２段の図となる。２つの図を比べると２変数は概ね逆に動いていることに気付く。これを確かめるために，df に備えられているメソッド corr()を使い相関係数を計算しよう。




```
df.corr()
```

出力は右のようになり，右上と左下の値が２変数の相関係数である。値は約-0.17 であり逆相関が確認できる。

	gdp_growth	u_deviation
gdp_growth	1.00000	-0.17249
u_deviation	-0.17249	1.00000

また上の図から persistence，即ち，正（もしくは負）であれば正（も

しくは負）の期間が続く傾向にあるようにも見える。この特徴を確認する方法として自己相関係数があり，df を使って計算しよう。まず 1 つの変数を選択し，メソッド autocorr()を使う。

```
df['gdp_growth'].autocorr()
```

df の後に['gdp_growth']を追加することにより，その変数名がある列を選択することになる。列ラベルは‘’で囲み文字列で指定する。このコードは「df の列 gdp_growth の相関係数を計算する」と読める。結果は約 0.13 であり，同様に u_deviation の相関係数を計算すると約 0.43 であることが簡単に確認できる。失業率の変化の persistence が高いと言える。

５．最小二乗法による推定

２変数の特徴が確認できたので回帰分析を行うが，４つのステップに分けて説明する。

ステップ１：推定式の設定

推定式は次の形式で書き‘’で囲むことにより文字列として設定する。

‘非説明変数 ~ 定数項以外の説明変数’

定数項は自動的に挿入される。また定数項以外の説明変数が複数ある場合は「+」でつなげる。今回は単回帰となるため，次のコードを実行し推定式を変数 formula に割り当てる。

```
formula = 'gdp_grwoth ~ u_deviation'
```

ステップ２：自動計算の準備

ols を使って計算する対象を準備し，変数 model に割り当てる。

```
model = ols(formula, data=df)
```

ols の中に２つ引数があるが，１つ目は推定式である formula を指定し，２つ

目は推定に使うデータ（data）に df を指定する。

ステップ３：自動計算

変数 model には多くのメソッドが備わっており，その１つが fit()である。これを使うことにより様々な計算を一瞬で行うことが可能となる。

```
result = model.fit()
```

右辺は model の関数 fit()を使い計算した結果を返し，それを左辺の変数 result に割り当てる。

ステップ４：結果の表示

result は計算結果が詰まったオブジェクトであり，その中にメソッド summary()が用意されている。これを使うことにより，基本的な推定結果を表示することができる。

```
result.summary()
```

このコードを実行すると表１が表示される。[5] 表１は３つのセクションから構成される。

表１：

OLS Regression Results

Dep. Variable:	gdp_growth	R-squared:	0.030
Model:	OLS	Adj. R-squared:	0.020
Method:	Least Squares	F-statistic:	3.128
Date:	Wed, 12 Aug 2020	Prob (F-statistic):	0.0800
Time:	11:10:37	Log-Likelihood:	-141.43
No. Observations:	104	AIC:	286.9
Df Residuals:	102	BIC:	292.1
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.2056	0.093	2.202	0.030	0.020	0.391
u_deviation	-1.1397	0.644	-1.769	0.080	-2.418	0.139

Omnibus:	32.541	Durbin-Watson:	1.843
Prob(Omnibus):	0.000	Jarque-Bera (JB):	94.440
Skew:	-1.063	Prob(JB):	3.11e-21
Kurtosis:	7.157	Cond. No.	6.90

[5] 厳密には表１は print(result.summary())の出力である。

- 上段には OLS 推定の基本的な情報が表示される。例えば、左側にある No. Observation は標本の大きさ，右にある R-squared は決定係数である。
- 中段には定数項（Intercept）とbの推定値や t 値、p 値（P>|t|）が表示される。
- 下段には様々な検定統計量が並んでいる。例えば、Durbin-Watson はダービン・ワトソン検定統計量である。

推定結果から次のことがわかる。まず Durbin-Watson 統計量は 1.843 なので誤差項に系列相関はないと判断できる。また，定数項とスロープ係数の符号は期待通りである。定数項は有意水準 5%で$\hat{a} = 0$の帰無仮説を棄却できる。一方，スロープ係数は有意水準 10%で$\hat{b} = 0$の帰無仮説を棄却できる。

定数項の解釈を考えよう。0.21%の値は失業率の変化がない場合の平均成長率を示しており，長期的な成長率と考えて良いだろう。この値は前期比で計算した成長率であるため，コードを書いて年率換算してみる。まず変数 result には推定結果の様々な情報が詰まっていることを思い出そう。特に，係数の推定値は result.params でアクセスできる。また 2 つの推定値があるため，定数項は result.params[0]，スロープ係数は result.params[1]で抽出できる。[6] 従って，年率換算の長期的成長率は次のコードで計算できる（*は乗算，**は累乗を表す）。

```
(1+0.01*result.params[0])**4-1
```

結果は約 0.83%。

次に，失業率の変化が 1%の場合の平均成長率を考えよう。まず前期比での成長率を変数 growth_quarter に割り当てる。

```
growth_quarter = result.params[0]+result.params[1]
```

次のコードで年率換算する。

[6] Python での順番は 1,2,3…ではなく 0,1,2,…と数える。

```
(1+0.01*growth_quarter)**4-1
```

結果は約-3.68%となり，絶対値で考えると非常に大きな値となっている。この結果は，失業率の変化が小さいためであり，少しの変化が成長率に大きな影響があることを意味している。

６．おわりに

本稿では，オークンの法則を例として Python の使い方を説明した。冒頭でも書いたが，本稿に関連する次のファイルは参考文献に挙げた GitHub のレポジトリーからダウンロードできる。

① 実質 GDP と失業率のデータが含まれる 2 つの csv ファイル（内閣府のサイトからダウンロード）

② 実質 GDP の成長率と失業率の変化のデータが含まれる csv データ（①のデータを使って作成）

③ ②のデータを作成するための Python コードと本稿で説明したコードが含まれる Jupyter Notebook

この機会に是非 Python にチャレンジしてはどうだろう。

参考文献

[1] 春山鉄源:『Python で学ぶ入門計量経済学』, https://py4etrics.github.io

[2] GitHub のレポジトリー（関連ファイルのダウンロード用）: https://github.com/Haruyama-KobeU/gakushu2021