

Python によるマクロ経済分析

1. はじめに

2020 年度から小学校で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、中学校では 2021 年度から、高校では 2022 年度からプログラミング授業が必修化された。また 2025 年の大学入学共通テストからはプログラミングの科目である「情報 I」が追加される予定である。この新たな政策は、政府だけではなく社会全体がプログラミングの重要性を強く認識している証と言える。

このような社会の趨勢が現れる前から、神戸大学経済学部および経済学研究科では R や Stata, Matlab を使う授業が提供されてきたが、今後更に、プログラミングに基づく授業が拡充されると期待される。

一方、プログラミングを学んだことがない学生にとって、「どのようなものなのか?」、「何ができるのか?」と疑問に感じることだろう。本稿では、マクロ経済学における重要な概念であるオークンの法則を使い、Python の使い方を説明し、上の問いに答えるのが目的である。紙面に限りがあるため、掘り下げた説明はできないが、読者が「こういうものか」というイメージを抱くことができれば幸いである。詳細な説明は参考文献を参照することを勧める。

では、なぜ Python なのか? 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。人気の理由は、(1) 無料、(2) 高い汎用性（機械学習、科学的数値計算、ゲーム、画像処理など）、(3) 低い学習コストが挙げられる。特に、(3) に関して付け加えると、Python のコードは英語を読む・書く感覚と近いだけではなく、頻繁に出てくるコードの中で日本語の助詞「の」や動詞「実行する」と対応する箇所もある。

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いる。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が、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想されるプログラミング言語を学ぶことは経済学部生にとって非常に有意義ではないだろうか。

本稿で使うデータやコードは全て参考文献に挙げる GitHub のレポジトリで公開している。また、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。ⁱ

2. オークンの法則

マクロ経済学で重要な概念にオークンの法則があり、次式で与えられる。

$$\frac{Y_t - \bar{Y}_t}{\bar{Y}_t} = -b \times (u_t - \bar{u}_t), \quad b > 0 \quad (1)$$

Y_t は GDP, \bar{Y}_t は GDP のトレンド（もしくは潜在的 GDP）, u_t は失業率, \bar{u}_t は失業率のトレンド（もしくは自然失業率）である。従って、左辺は GDP のトレンドからの乖離率（%）であり、右辺は失業率のトレンドからの乖離（%ポイント）を表している。式(1)は両変数に負の関係があることを示しており、労働者が生産投入に使われることを考えると、負の関係は直感的に理解できるだろう。例えば、失業率が上昇しトレンドからの乖離が大きくなると、雇用者数が減少することになり、GDP も長期的トレンドを下回ることになる。では、失業率の乖離が 1%ポイント上昇した場合、GDP は平均でトレンドから何%乖離するのだろうか。この問いに答えるためにはパラメータ b を推定する必要がある。以下では、日本の 1994 年 Q1 から 2021 年 Q4 の四半期データを使い、回帰分析をとおしてパラメータ b を推定する。

3. データの読み込み

使用するデータはインターネット上に準備してあり、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/3NkpZhQ'
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、それらはライブラリー、パッケージ、モジュールなどの名前と呼ばれる。パッケージとはライブラリーの一部であり、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。上のコードの 1 行目では、エクセルのようにデータを扱うための `pandas` というライブラリーを `pd` として（`as`）読み込んでいる（`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` というライブラリーを読み込む。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くなる。「`statsmodels` のパッケージである `formula` のモジュール `api` から（`from`）`ols` を読み込む（`import`）。」`ols` は名前が示すように、最小二乗法（Ordinary Least Squares）の自動計算をおこなう関数である。「神戸大学」のような単なる文字列を作る場合は、`' '` 又は `" "` で囲む必要があるが、この方法を利用して、4 行目はファイルが置いてあるウェブ・アドレスを文字列として変数 `url` に割り当てている。「`=`」は右辺を左辺の変数に「割り当てる」という意味である。5 行目は、`pd` (`pandas`) 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、エクセルのスプレッド・シートを想像すれば良いだろう。

- `gdp_kairi`: 実質 GDP のトレンドからの乖離率 (%) であり、式(1)の被説明変数に使う。
 - `u_kairi`: 失業率の変化 (%ポイント) であり、式(1)の説明変数に使う。
- `df` の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行を表示するコードであり、右の出力となる。`index` の下にあるのが行ラベルであり、毎四半期の最後の日付となっている。

	gdp_kairi	u_kairi
index		
1980-03-31	0.752303	-0.087693
1980-06-30	-0.736446	-0.109084
1980-09-30	0.324628	-0.047087
1980-12-31	1.275978	0.031755
1981-03-31	1.122977	0.004260

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や関数から構成される集合体であり、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等がデータである。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「実行する」と理解すれば良いだろう。

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()
```

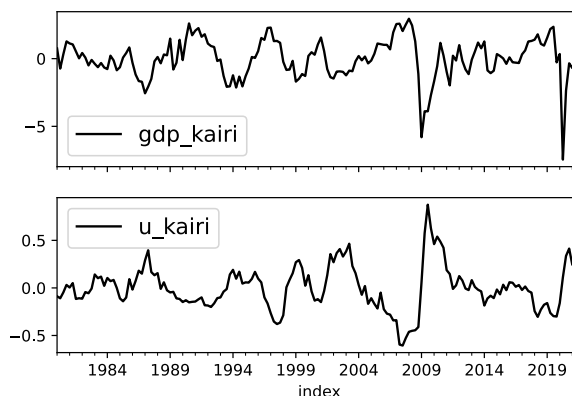
```
gdp_kairi    -7.119104e-03
u_kairi       4.695293e-14
```

右の出力となるが、`e-03` は 10^{-3} を、`e-14` は 10^{-14} を表す科学表記である。つまり、失業率の正と負の乖離は相殺し、平均は概ね0だということである。GDPの乖離率も同様の動きがあることが言える。

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、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だろう。これを確かめるために、dfのメソッドcorr()を使い相関係数を計算してみよう。

```
df.corr()
```

右上もしくは左下の値が2変数の相関係数となる。値は約-0.63であり、逆相関が確認できる。

	gdp_kairi	u_kairi
gdp_kairi	1.000000	-0.630428

また上の図から持続性と呼ばれる特徴も確認できる。持続性とは、正（もしくは負）であれば正（もしくは負）の期間が続く傾向を指しており、自己相関係数を使い数値化することができる。dfを使って計算してみよう。まず1つの変数を選択し、メソッドautocorr()を使う。

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

dfの後に['u_kairi']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ができる。列ラベルは' '又は" "で囲み文字列で指定する。このコードは「dfの列u_kairiの相関係数を計算する」と読める。結果は約0.876であり、図でも数値でも強い持続性が確認できる。同様にgdp_growthの相関係数を計算すると約0.691となる。

5. 最小二乗法による推定

次に回帰分析をおこなうが、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

推定式は次の構文に沿って書き、' '又は" "で囲み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半角の「+」でつなげるだけである。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に割り当てることにする。

```
formula = 'gdp_kairi ~ u_kairi'
```

ステップ2：自動計算の準備

ols関数を使い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（関数）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 `fit()` を使い計算した結果を返し、それを左辺の変数 `result` に割り当てている。

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、推定結果の基本的な情報を表示することができる。

```
result.summary()
```

コードを実行すると、3つのセクションから構成される表1が表示される。

表 1	上段	<pre> ===== OLS Regression Results ===== Dep. Variable: gdp_kairi R-squared: 0.397 Model: OLS Adj. R-squared: 0.394 Method: Least Squares F-statistic: 109.5 Date: Wed, 18 May 2022 Prob (F-statistic): 5.34e-20 Time: 17:57:45 Log-Likelihood: -260.10 No. Observations: 168 AIC: 524.2 Df Residuals: 166 BIC: 530.4 Df Model: 1 Covariance Type: nonrobust ===== </pre>					
	中段	<pre> ===== coef std err t P> t [0.025 0.975] ===== Intercept -0.0071 0.088 -0.081 0.936 -0.182 0.167 u_kairi -4.0174 0.384 -10.464 0.000 -4.775 -3.259 ===== </pre>					
	下段	<pre> ===== Omnibus: 98.256 Durbin-Watson: 1.085 Prob(Omnibus): 0.000 Jarque-Bera (JB): 847.180 Skew: -1.976 Prob(JB): 1.09e-184 Kurtosis: 13.267 Cond. No. 4.35 ===== </pre>					

- 上段には基本的な情報が表示される。例えば、左側にある `Dep. Variable` は被説明変数の変数名、右にある `R-squared` は決定係数を意味する。
- 中段の `coef` は、定数項（`Intercept`）と `u_kairi` の係数である b の推定値を指す。また t 値、 p 値（ $P>|t|$ ）と信頼区間が表示されている。
- 下段には様々な検定統計量が並んでいる。例えば、右にある `Durbin-Watson` はダービン・ワトソン検定統計量である。

表1から次のことがわかる。第一に、定数項の推定値は非常に小さい。また p 値は大きく、通常の有意水準では「定数項の値はゼロ」の帰無仮説を棄却できない。即ち、失業率の乖離がゼロの場合、GDPの乖離率もゼロになることを示しており、定数項がない式(1)の予測と一致する。第二に、

パラメータ b の推定値は負であり，式(1) と整合的である。定数項をゼロとすると，失業率が長期的トレンドを上回る（下回る）場合，平均で GDP 乖離率は負(正)に転じることが確認できる。また p 値の値は非常に小さく，通常の有意水準では「 b の推定値はゼロ」の帰無仮説を棄却できない。第三に， b の推定値は約 -4.02 である。1980 年～2021 年の間，失業率がトレンド（もしくは自然失業率）から 1 %ポイント乖離すると，平均で GDP はトレンド（もしくは潜在的 GDP）から絶対値で 4%乖離したことがわかる。最後に，Durbin-Watson 比は 1.085 であり誤差項に正の自己相関が疑われる。

6. おわりに

Python を使いオークンの法則を検討した。簡単なコードで，マクロ経済分析ができることが分かったのではないだろうか。

本稿を執筆する上で使った次のファイルは参考文献に挙げた GitHub のレポジトリからダウンロード可能となっている。

- ① 実質 GDP と失業率のデータが含まれる 3 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 の乖離率と失業率の乖離のデータが含まれる csv ファイル（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 を使ってマクロ経済分析にチャレンジしてはどうだろう。

参考文献

- [1] 春山鉄源：『経済学のための Python 入門』，
<https://py4basics.github.io>
- [2] 春山鉄源：『Python で学ぶ計量経済学』，<https://py4etrics.github.io>
- [3] 春山鉄源：『Python で学ぶマクロ経済学（中級＋レベル）』，
<https://py4macro.github.io>
- [4] GitHub のレポジトリ（関連ファイルのダウンロード用）：
<https://github.com/Haruyama-KobeU/arukikata2022>

ⁱ Python と Jupyter Notebook のインストールには Anaconda を勧める。
<https://www.anaconda.com/products/individual> からダウンロードできる。
また Jupyter Notebook の基本的な使い方の説明は，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。