Pythonによる統計入門1

松田　史生[1]・川瀬　雅也[2]*

前回，基本的な文法を解説したので，今回と次回で，Pythonを使った基本的な統計処理法を解説したい．

さて，海外出張（海外逃亡？）していたX教授が戻ったとのことで，例の二人組が早速，X教授のもとを訪れた．

何故，Pythonで統計を？

Aさん：X教授，お帰りなさい．

B君：どこへ行ってらっしゃったのですか．

X教授：やあ，二人とも久しぶり．うーんとだね，ヨーロッパから雄琴，アジアあたりを回って有意義だったよ．

ところで，Pythonを勉強中ってH君から聞いたよ．

B君：そうなんです．ただ，ちょっと新鮮なだけで統計処理ならRでいいのかなって気もしますけど．

X教授：なるほど．データサイエンスの世界でPythonがよく使われる理由は，何といってもPythonという言語の分かりやすさだね．これは，納得してもらえると思うが．

Aさん：はい．十分，納得です．

X教授：あと，Pythonは苦手なことが少ないんだ．無料で，数値計算も得意で，わかりやすいから，深層学習用フレームワークはPython用が多いね．だからここ数年大人気なんだ．プログラミングの情報も，充実している．強いて欠点をあげると，ちょっと遅いことくらいかな．

Aさん：そこで，以前の連載で教わった統計処理の基礎をPythonで復習してるんです．以前の連載は，https://www.sbj.or.jp/sbj/sbj_tokei_kaiseki.htmlからダウンロードできるって便利な世の中ですね．

B君：でも，Rで簡単にできたことをPythonでどうすればいいのかがわからなくて，うまくいかないんです．

Pythonを使う＝モジュールを使う

Aさん：前連載の第1回目のデータを生物工学会誌のHPからダウンロードしてきました．CSVファイルで列方向にデータが並んでます．一行目が見出し行です．1列目（ID）が見出し列，2（A），3（B）列目にそれぞれAとBのデータが20個ずつ並んでいます．データは，C：直下のpydataにtest1.csvという名前で置いてみました．

X教授：ふむふむ．

B君：Hさんから教わったように，左下のWindows＝＞Anaconda＝＞Spyderを立ち上げました．次に，右下のコンソールにコマンドを入力して，ファイルを読み込みたいんです．で，Pythonのドキュメントを調べたんですが……

X教授：載ってなかったんだね．

表1．実験のコロニー数（前連載の第1回目で使ったデータ）

Aさんのコンピテントセル	B君のコンピテントセル
15, 13, 11, 13, 18, 19, 22, 21, 20, 16, 11, 16, 18, 10, 11, 12, 11, 14, 12, 11	15, 17, 21, 23, 28, 19, 22, 24, 30, 26, 21, 19, 18, 20, 21, 22, 21, 24, 22, 21

Aさん：そうなんです．先輩いきなりやる気ゼロです．

B君：Rだとすぐできたのに……

X教授：Pythonはモジュールでいろんな機能が追加できる，って聞いているよね．でも，Rと違って，すっぴんのPythonだけではほとんど何もできないんだ．つまり，モジュールを使いこなすことが秘訣なんだよ．統計処理で必要なモジュールは，numpy，pandas，scipy，pyplot，Scikit-learnくらいだと思う．

B君：でも，使えるモジュールをどこで探すんですか？

X教授：検索エンジンで“python csv 読み込み”で調べてみよう．

B君：ほんとだ，いろいろ出てきました．このページには“csv”ってモジュールの使用法が書いてあります．

Aさん：こっちのページは“NumPy”モジュールを使ったCSVファイルの読み込みの説明がありました．

X教授：PythonをRっぽく使うための“pandas”っていうモジュールでもCSVファイルの読み込み機能がある．

B君：うーん．ややこしい．では，どれを使えばいいんですか？

著者紹介　[1]大阪大学大学院情報科学研究科（教授）
[2]長浜バイオ大学（教授）　E-mail: m_kawase@nagahama-i-bio.ac.jp

pandasを使ってみる

X教授：どれでもいいんだけど，今回はpandasを使おう．Rにデータフレームってあったよね．Pythonでデータフレームを使えるようにしたのがpandasだ．便利なのでデータの入出力によく使われる．データ解析に必要な関数も入っている．Anacondaに含まれているから，インストールも不要だ．

B君：じゃあ早速やってみよう．確か……

```
In[1]: import pandas
```

ってすればモジュールが読み込まれるんだよね．

Aさん：では，検索で見つけたこっちのページの記述をまねして，read_csvでtest1.csvを読み込んでみよっか．ファイル名は'/pydata/test1.csv'で見出し列をindex_col='ID'で指定したらいいみたい．

```
In[2]: df=pd.read_csv('/pydata/test1.csv',index_col='ID')
Traceback (most recent call last):
  （略）
NameError: name 'pd' is not defined
```

Aさん：あれ，なにやらエラーが出ましたよ，先輩．

B君：なんで？書いてある通りにやったのに．

Aさん：あ，これは名前空間ってやつですね．こないだC君が教えてくれました．

X教授：その通り．Aさんが参考にしたページではpandasモジュールを

```
In[1]: import pandas as pd
```

として読み込んでないかい？

Aさん：確かにその通りです．：pdがpandasの代わりの名前になるんですよね．次のスクリプトと

```
import pandas as pd
df=pd.read_csv('/pydata/test1.csv',index_col='ID')
```

と，次のものは同じことをしていますね．

```
import pandas
df= pandas.read_csv('/pydata/test1.csv',index_col='ID')
```

あと，pandasからread_csvの機能だけを読み込むには

```
from pandas import read_csv
df= read_csv('/pydata/test1.csv',index_col='ID')
```

というのもアリだとC君が言ってました．

X教授：ふーん．C君って学生はなかなか詳しいね．read_csvという機能に違う名前（空間）を付けて呼び出すことができるんだ．検索で見つかる例をいくつか見るとわかるけど，

```
import pandas as pd
```

がpandasを呼び出すときによく使われているね．

B君：うーん．これまたややこしいですね．Pythonのどこがいいんだろ？

X教授：じゃあpandasの便利機能をいろいろ試してみよう．

```
In[1]: df.head() #最初の5つのデータを表示
     A   B
ID
1   15  15
2   13  17
3   11  21
4   13  23
5   18  28
```

読み込まれたデータの先頭後5行が確認できる．データを読み込んだデータフレームdfにhead()という命令を出す．という感じかな．このようなお作法をオブジェクト指向という（データフレームオブジェクトのインスタンスであるdfのhead()メソッドを呼び出している）．

B君：ところで，データの確認は，いつも行うんですね．

X教授：その通りだね．面倒でも必ず，データを正しく読み込んだかどうかを確認する癖をつけるように．B君のデータが，確か正規分布に近かったので，こちらのデータを使って基本統計量を出してみよう．

B君：Aさん，昔は，実験下手だったよね．

Aさん：今は，私の方が，いいデータを出しています．先輩，後で，お話をしましょうね．

B君：…….まずい！！
X教授：相変わらずで，楽しいね．平均の計算法は“pandas平均”で検索するとみつかるよ．あと，各列のデータは列の名前のdf['A']とdf['B']で抜き出せる．

```
In[14]:df['B'].mean()
Out[14]: 21.7
```

df['B']というデータフレームにmean()という命令を出して平均の計算をやってもらっている．同じように，中央値や，最頻値は，標本分散，不偏分散，四分位なども計算できるよ．

```
In[15]: df['B'].var() #不偏分散の計算
Out[15]: 12.642105263157896
In [16]: df['B'].describe() #四分位を調べる
count    20.000000
mean     21.700000
std       3.555574
min      15.000000
25%      19.750000
50%      21.000000
75%      23.250000
max      30.000000
```

NumPy

X教授：Pythonは数値計算が遅いんだ．そこで数値演算を高速に実行可能なNumPyというモジュールが作られている．NumPyは商用数値演算ソフトウェアのMATLABを意識して作られたフシがある．NumPyにも平均を計算する機能が，関数として提供されている．df['B']の平均を求めるには

```
In[23]: import numpy as np    #numpyをnpとして読み込む
In[24]: np.mean(df['B'])
Out[24]: 21.7
```

とする．df['b']の平均値の計算をnp.mean()にお願いする．というイメージだ．こういうのを関数型という．pandasとは機能を利用するときのスタイルが違うんだ．
B君：これまたいろいろあってややこしいですね．

matplotlib

X教授：では，ヒストグラムを書いてみよう．まず，作図に使うmatplotlibというモジュールをimportする．MatplotlibはPythonとNumPyのためのグラフ描画モジュールで，さまざまな種類のグラフを描画できる．RやMATLABのようにグラフを書くことができる．
　下記の例ではMatplotlibのなかのpyplotというサブモジュールをpltという名前で読み込んでいる．Matplotlibは巨大なので，必要な一部だけ読み込んだほうがいい．

```
In[1]: import matplotlib.pyplot as plt
In[2]: plt.hist(B_scores,range(15,30,3))
In[3]: plt.show()
```

とすると，図1のようにヒストグラムも，階級の幅を自由にできる．

```
In[4]: plt.hist(B_scores,range(10,40,2))
In[5]: plt.show()
```

では，図2ようになる．Rで，自由に作図するのは勉強が必要だ．こういった操作は，Pythonの方が簡単だと思う．
B君：なるほど．
Aさん：自分の目的に合った図を描けるのがいいですね．
B君：説明だと簡単にできそうですが，実際，自分でやってみると訳が分からなくなるような気がする…….
Aさん：先輩，練習あるのみです．
X教授：まずは今，説明したコマンドをまねることが大事だね．それと，失敗することだ．失敗して自分で調べて解決することで，勉強になることも多いからね．では，箱ひげ図も書いてみよう．

```
In[6]: plt.boxplot(B_scores)
In[7]: plt.show()
```

とすればいい（図3）．

```
In[8]: points=(A_scores, B_scores)
In[9]: fig, ax=plt.subplots()
In[10]: bp=ax.boxplot(points)
```

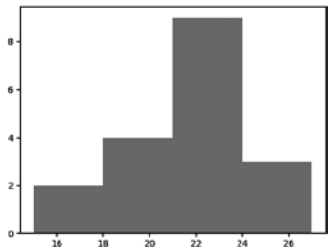

図1．B君のデータのヒストグラム

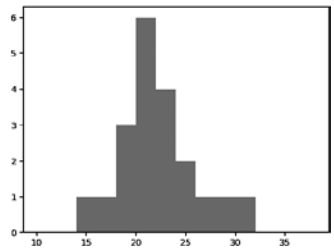

図2．階級の幅を変えたヒストグラム

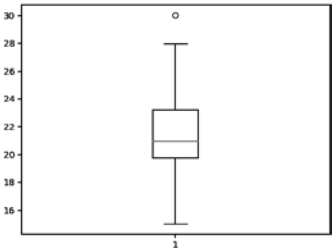

図3．B君のデータの箱ひげ図

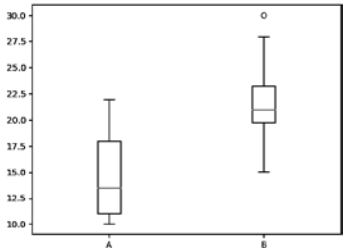

図4．AさんとB君のデータの箱ひげ図

```
In[11]: ax.set_xticklabels(['A','B'])
In[12]: plt.show()
```

とすると，図4のように，並べて表示もできる．

定番のt-検定

X教授：次は，t-検定の「2群間の平均値の差の検定」をやってみよう．t-検定の手順を覚えているかな．

Aさん：まず，データの正規性を確認して，正規性があれば，分散が同じだとしていいかを見るんでしたね．

B君：分散が同じと見てよければ，Studentのt-検定で，そうでなければ，Welchのt-検定でしたよね．

Aさん：最近は，分散の結果によらずWelchのt-検定が奨励されているんでしたよね．

X教授：その通り．これは，実験で得られるデータ数が多くないことが理由の一つだ．分散は同じ経験的に分かっている場合は，Studentのt-検定を使うとされる．

Aさん：思い出しました．

X教授:まず，正規性をShapiro-Wilk検定で確認しよう．データが，正規分布からサンプリングされたことを帰無仮説とする検定だ．一番簡単な検定法なので，覚えておくといいと思う．統計関連の機能はScipyのstatsというモジュールを使うのが定番だ．ScipyはMATLABが得意とする科学（工学も含む）計算用の機能を集めたものだ．

```
>>> from scipy import stats
```

のようにすれば，statsとして読み込める．

Aさん：検索したら，Shapiro-Wilk検定はstats.shapiro()を使うとできるみたいです．

```
In[5]: stats.shapiro(df['A'])
Out[5]: (0.9004665613174438, 0.04207809269428253)
In[6]: stats.shapiro(df['B])
Out[6]: (0.9600358605384827, 0.5445652604103088)
```

B君：また，謎の数字が……

Aさん：記事によると，2番目の数字が「正規分布に従う」という帰無仮説のp値ですね．

B君：Aさんのデータは，有意水準（α）を0.05とした場合，正規分布に従うとは言えないんだったね．

Aさん：……．先輩，昔は昔って，こないだ言いましたよね？

X教授：練習だし，p値も0.05に近いから正規性ありとして進めよう．続けるよ．次は，等分散性を見るために，stats.bartlett()を使う．

```
In[7]: stats.bartlett(df['A'], df['B']) # Bartlett検定
Out[7]:BartlettResult(statistic=0.09019087382806061,
                      pvalue=0.7639346392064993)
```

pvalue > 0.05となったので，等分散とみていいという結果を得た．では，Studentのt-検定を行ってみよう．

```
In[7]: stats.ttest_ind(df['A'], df['B']) # Studentのt検定（両側）
Out[7]:Ttest_indResult(statistic=-6.004774759060393,
                       pvalue=5.607530447664291e-07)
```

となって，二人の結果には有意差があるとでたね．

Aさん：ありますね．言っときますけど，C君と同じことをやったときは，私のほうがいい結果でした．

X教授：Welchのt-検定をするには，equal_var=Falseを追加する．

```
In[8]: stats.ttest_ind(df['A'], df['B'], equal_var=False)
# Welchのt検定（両側）
Out[8]:Ttest_indResult(statistic=-6.004774759060393,
                       pvalue=5.708138841750333e-07)
```

だから，結果は有意差ありで同じだね．Rと違って，PythonのstatsではStudentのt-検定がデフォルトだということだ．

B君：要注意ですね．あと，対応のある場合の検定は，どうやればいいんですか．

X教授：ちょうど，二人のデータ数が同じなので，データに対応があるとして検定してみよう．

```
In[9]: stats.ttest_rel(A_scores,B_scores)
Out[9]:Ttest_relResult(statistic=-7.252507846014176,
                       pvalue=6.968927124146987e-07)
```

となって，有意差があるね．

Aさん：stat.ttestの後，indとするかrelとするかの違いですね．Rよりも簡単かもしれませんね．

X教授：その通りだ．「有意差がある」=「差がある」と考える人は，まだ，多いと思う．有意差の意味をしっかりと理解して，統計処理の結果を使うようにしなければならないことを，いつも，頭の中に持っておいてほしいな．

参考文献

1) 谷合廣紀：Pythonで理解する統計解析の基礎，技術評論社 (2018).
変数名などは，本書に準じている．