Pythonによる統計入門2

松田　史生[1]*・川瀬　雅也[2]

前回に続いて，Pythonを使った基本的な統計処理法を解説したいと思う．χ^2検定，重回帰分析を復習して，最後はニューラルネットワークを作ってみよう．

χ^2検定

X教授：まず，研究室メンバーの血液型の分布が，日本人の典型的な構成による分布と同じとみていいかどうかを検定しよう（表1）．帰無仮説は「日本人の典型的な構成と同じ分布である」で，対立仮説は「日本人の典型的な構成と同じ分布であるとは言えない」となる．度数(n_i)はメンバーの総計(N)50名の内訳で，期待確率(p_i)は日本人の構成比というのは分かるね．期待度数は，もし，標準的な構成比通りなら，各血液型の度数は幾らくらいかを計算したもの，つまり，Np_iの値だ．

表1．研究室メンバーの血液型の構成

	A型	B型	AB型	O型	合計
度数	15	9	10	16	50
期待確率	0.38	0.22	0.09	0.31	1
期待度数	19	11	4.5	15.5	50

B君：つぎはχ^2検定ができるモジュールを探すんでしたね．前回使ったscipy.statsモジュールにあるかな？

Aさん：“Python　カイ2乗検定　scipy.stats”で検索してみると，scipy.stats.chisquare()を使っているものがたくさん見つかりました．

X教授：じゃ次は“scipy.stats.chisquare”で検索したらScipyのホームページが見つからないかね？

B君：ありました．“scipy.stats.chisquare — SciPy v1.2.1 Reference Guide”って，英語のページが見つかりました．

X教授：そのページには使い方が説明されている．たとえば，“Parameters:”というところに

f_obs : array_like
Observed frequencies in each category.
f_exp : array_like, optional
Expected frequencies in each category. By default the categories are assumed to be equally likely.
ddof : int, optional
“Delta degrees of freedom”: adjustment to the degrees of freedom for the p-value. 以下略..
axis : int or None, optional. 以下略..

とあるのを解読してみよう．

B君：英語なんですけど……

Aさん：（無視して）一つ目の引数f_obsは，観察された度数みたいですね．array_likeというのは，リスト形式でいいってことでしょうか？となると[15,9,10,16]になるんですかね．

X教授：次のf_expは期待度数のことだね．optionalというのは，指定しなくてもいい．ということだ．もしf_expを指定しなかった場合は，期待度数を均等とする，と書いてある．リスト形式で[19,11,4.5,15.5]と指定しよう．残りのddofとaxisはoptionalだから，とりあえず無視してやってみよう．

```
In[1] : from scipy import stats #モジュールの読み込み
In[2] : stats.chisquare([15,9,10,16],[19,11,4.5,15.5])
Out[2] : Power_divergenceResult(statistic=
7.944092881274545, pvalue=0.047181378964290865)
```

Aさん：出力の意味は同じページのReturns:というところにあるみたいですけど，今回はわかりやすいですね．このp値は，前の連載でRを使った時の値と同じになってますね．

B君：機能ごとに，使い方が違ったり，出力の見方が違ったり，混乱しますね．

Aさん：Rにもそういうところはありましたし，無料で使わせてもらっているんだから，文句は言えないかも．自分がしっかりしていれば，問題ないということですね．

X教授：その通りだね．それが嫌なら，自分で，自分に合ったものを作るしかないね．Pythonを勉強しているから，できるようになるかもしれないね．

B君：できるまでに，地球が滅びそうですね．

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

Aさん：大げさじゃないですか.

X教授：では，独立性の検定もやってみよう．ここでも，前のデータを使おう（表2）．この表は2行2列であるので2 × 2分割表とよばれる．この例では，新薬Aは偽薬Bより効果があったかということを調べるんだったね．あと，帰無仮説は，行項目と列項目は独立している，つまり，関連性はないんだ.

表2．新薬Aと偽薬Bの効果

	症状の改善	
	有り	無し
新薬A	200	125
偽薬B	56	250

X教授：じゃ，どうやって実行するか調べてみようか.

B君：“Python　独立性検定”で検索してみると，scipy.stats.chi2_contingencyを使うみたいですね.

Aさん：scipy.stats.chi2_contingency”を調べると，Scipyのページがありました．Parameters:には下記のように書いてあります.

observed : array_like
The contingency table. The table contains the observed frequencies (i.e. number of occurrences) in each category. In the two-dimensional case, the table is often described as an“R x C table”.
correction : bool, optional　略
lambda_ : float or str, optional.　略

“The contingency table”（「分割表」）をリスト形式で指定するって，どうすればいいんですか？

B君：二次元の配列は，[[200,56],[125,250]]だから……

```
In[9] : stats.chi2_contingency([[200,56],[125,250]])
Out[9] :
 (120.41864493230997,
 5.122501872713255e-28,
 1,
 array([[131.85419968, 124.14580032],
        [193.14580032, 181.85419968]]))
```

B君：うまくいったけど，またまた，謎の数字が……

Aさん：さっきのページのReturns:をみてみましょ．前から順番にchi2（カイ2乗値），p（p値），dof（自由度），expected（期待値）の順のようですから，2つ目の数字がp値ですね．この場合は関連性アリ，ですね.

相関係数

X教授：回帰分析と相関係数は，覚えているかな.

B君：もちろんです.

X教授：2つのデータ間の類似性の指標の1つが相関係数だ．回帰分析には，説明変数と従属変数があって，単回帰分析と重回帰分析に分けられる．単回帰分析は重回帰分析の特別な形（説明変数が1つの場合）と考えれば，多変量解析に分類できる.

Aさん：回帰分析はExcelで行うのが，簡単だったんですよね.

X教授：その通りだが，いろいろな関数で回帰することを考えると，やはり，Pythonでもできるようになっているほうがいい.

Aさん:そうですね．でも，直線で回帰する以外は，やったことはないですよ.

B君：僕も，そうですが.

X教授:でもまあやってみよう．C直下の¥Pydataにデータが入っているとして，データを読み込もう．今回使うデータファイルのmetabolome.csvは生物工学会のHPからダウンロードしよう．30種類の日本酒の官能試験値（taste）と10種類の代謝物のメタボローム分析結果のデータだ．このデータは阪大の福崎教授からご提供いただいた.

Aさん，B君：先生ありがとうございます！

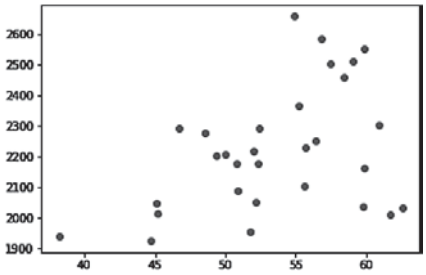
```
In[1] : import pandas as pd　#Pandasの読み込み
In[2] : data=pd.read_csv('C:¥pydata¥metabolome.csv') #ファイル読み込み
```

X教授：dataが読み込まれたか確認してみよう.

```
In[3] : data.head()
    taste   glutamine  methionine     ...
0    59.9      2162.3       300.0     ...
1    38.2      1939.8       243.8     ...
2    55.6      2101.8       357.2     ...
3    59.9      2552.3       301.2     ...
4    52.2      2051.8       285.7     ...
```

Aさん：まずは，データを散布図で確認してみます．tasteとglutamineの比較がいいですかね.

In[4] : import matplotlib.pyplot as plt # pyplotをインポート
In[5] : plt.scatter(data[‘taste’], data[‘glutamine’])) #散布図をプロット



Aさん：うーん．あまりきれいな関係ではないですね．次は，相関係数を求めてみましょう．
B君：“Python 相関係数”で調べてみたら，NumPyに関数numpy.corrcoefがありました．

In[6] : import numpy as np　#NumPyの読み込み
In[7] : np.corrcoef(data[‘taste’], data[‘glutamine’])
Out[7] :
array([[1.　　, 0.40942512],
　　[0.40942512, 1.　　]])

B君：また，謎の数字が，なにこれ？
Aさん：0.40942512が相関係数のようですね．
B君：最後の出力が謎なの以外は，案外，簡単にできるんですね．
X教授：どのプログラムを使っても，まあ，こんなもんだよ．

scikit-learnで単回帰分析

Aさん：次は，単回帰分析のやり方を調べてみました．“python単回帰分析”で検索すると，scikit-learnというモジュールを使っている例がたくさん出てきますね．
X教授：scikit-learnというのは機械学習分野で用いる機能を集めたモジュールだね．最近の機械学習ブームで注目されている．あと，使い方もPythonっぽいので順番に説明しよう．まずはモジュールを読み込もう．

In[1] : from sklearn.linear_model import LinearRegression

X教授：次の呪文で，LinearRegression()というクラスのインスタンスlrを生成する．

In[2] : lr = LinearRegression()

Aさん：ラミパスラミパスるるるる，との違いがわかりません．
X教授：オブジェクト指向はC君が詳しいんじゃないかな．ここではこういう使い方がPythonっぽいってことだけ覚えておいて．で，このlrっていうインスタンスにいろいろ仕事をお願いしていく．
まず，目的変数のtasteをyという変数に，説明変数のglutamineをXいう変数にそれぞれ代入しよう．

In[3] : y = data[‘taste’]
In[4] : X = data[[‘glutamine’]]

このとき，Xだけ[[]]が二重になっている点に注意，1重だとglutamineデータを抜きだしたリスト．2重だとglutamineデータを抜きだした新たなデータフレームが作られる．
最後にfitメソッドで回帰分析を行う．すると，lr.coef_，lr.intercept_という属性ができてそこに係数と切片の値が収納される．アンダースコアは属性だ．ということを明示するためについているのだと思う．

In[5] : lr.fit(X,y) #回帰分析を実行
In[6] : lr.coef_#係数を表示
Out[6] : array([0.01192606])
In[7] : lr.intercept_ #切片を表示
Out[7] : 27.008750955430767

単回帰分析の結果[taste] = 0.0119 * [glutamine] + 27.0087となった．決定係数を計算するには，

In[8] : lr.score(X,y) #決定係数を計算
Out[8] : 0.16762892498873228

で計算できる．
B君：data[[‘glutamine’]]という指定と，係数が配列で表示されるのがややこしいですね．なんでなんでしょう？
X教授：次の重回帰分析でありがたみがわかると思う．

重回帰分析

X教授：では，重回帰分析に進もうか．ためしに，glu-

tamineとmethionineの2変数でやってみよう.

In[9] : X = data[['glutamine', 'methionine']] #glutamine, methionineのデータフレーム
In[10] : X.head()#できたか確認
Out[10] :

	glutamine	methionine
0	2162.3	300.0
1	1939.8	243.8
2	2101.8	357.2
3	2552.3	301.2
4	2051.8	285.7

In[11] : lr.fit(X,y) #回帰分析を実行
In[12] : lr.score(X,y) #決定係数を計算
Out[12] : 0.5746138651263601

Bさん：決定係数がかなり向上しましたね.
X教授：つぎは10変数全部で重回帰分析しよう.

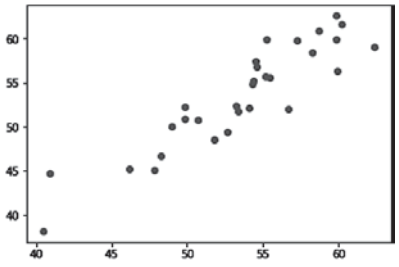
In[13] : X = data.drop("taste", 1)

この命令は，taste列以外をdrop（落とした）データフレームを作る．最後の1は列方向という意味だ.

In[14] : lr.fit(X,y) #回帰分析を実行
In[15] : lr.score(X,y) #決定係数を計算
Out[15] : 0.8313844202072022

lr.predict(X)でこの回帰式でyを予測できる．X軸を予測値，Y軸を官能試験値として比較するには,

In[16] : pyplot.scatter(lr.predict(X),y)



Aさん：かなり合ってきましたね．あと，fit()，score()を使うのは単回帰分析，重回帰分析でも同じなんですね. 重回帰分析をしたら，次は変数選択，でしたっけ？たしか係数がゼロ，という帰無仮説検定のp値を調べればいいんですが……
B君：いくら検索して調べても出てこないね……あれ？
X教授：（調べ物をして）……なるほど……ゴホン. scikit-learnのLinearRegressionにはp値を調べる機能がないみたいだな．scikit-learnの目的は官能試験値をよく予測する回帰式を作ることなんだが，予測に寄与する説明変数を選んで，生物学的に解釈，という作業が機械学習分野ではあまりやらないのかもしれない．ちなみにStatsModelsというモジュールを使うとRっぽい解析ができるから，やってみよう.

In[17] : import statsmodels.api as sm #StatModelsのインポート
In[18] : model = sm.OLS(y,X) #y, Xはそのまま使う
In[19] : results = model.fit()#フィティング実行
In[20] : results.summary()#結果表示

OLS Regression Results

Dep. Variable:	taste	R-squared:	0.998
Model:	OLS	Adj. R-squared:	0.997
Method:	Least Squares	F-statistic:	1003.
Date:	Fri, 12 Apr 2019	Prob (F-statistic):	9.66e-25
Time:	22:29:06	Log-Likelihood:	-68.830
No. Observations:	30	AIC:	157.7
Df Residuals:	20	BIC:	171.7
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
glutamine	0.0098	0.004	2.498	0.021	0.002	0.018
methionine	0.0796	0.026	3.061	0.006	0.025	0.134
alanine	0.0011	0.002	0.672	0.509	-0.002	0.005
tryptophan	-0.0412	0.031	-1.334	0.197	-0.106	0.023
lysine	9.821e-05	0.006	0.017	0.987	-0.012	0.012
leucic_acid	0.0644	0.033	1.934	0.067	-0.005	0.134
malate	-0.0139	0.003	-4.084	0.001	-0.021	-0.007
citrate	0.0025	0.003	0.728	0.475	-0.005	0.010
pyruvate	-0.0018	0.005	-0.380	0.708	-0.012	0.008
succinate	0.0003	0.002	0.160	0.874	-0.003	0.004

Omnibus:	1.219	Durbin-Watson:	2.272
Prob(Omnibus):	0.544	Jarque-Bera (JB):	0.914
Skew:	-0.087	Prob(JB):	0.633
Kurtosis:	2.163	Cond. No.	371.

B君：これRっぽいな．昔に戻ったような気がします. P>]t[がp値なのでこれが0.05より小さいglutamine, methionine，malateが採用できるのかな.
Aさん：確かめてみましょう.

In[21]X = data[['glutamine', 'methionine', 'malate']] #glutamine, methionine, malateのデータフレーム
In[22]lr.fit(X,y)
In[23]lr.score(X,y)　#決定係数
Out[23] : 0.758102800533946

Aさん：3つの代謝物だけで決定係数0.75の重回帰モデルができました．

ニューラルネットワーク

X教授：では，scikit-learnで遊んでみよう．ここでは，細かい説明はともかく，作業の流れを見てほしい．日本酒のデータは数が少なすぎるので，アヤメのデータセット（Iris）を使う．3種のアヤメの花びら，がくの幅と長さのデータが50個体ずつ，計150個体分ある．このテスト用データとしてscikit-leranモジュールが用意してくれているから，それを使おう．

In[17] : from sklearn.datasets import load_iris #Irisを読み込むためのモジュール
In[18]irisd = load_iris() #Irisをirisdに読み込む．

X教授：つぎに花びら，がくの幅と長さのデータから，アヤメの種類を分類するニューラルネットワークを作ってみよう．
まず，モデル作成用の訓練（training）データと，モデルの性能テスト用のデータに分割する．scikit-learnにはデータ分割作業用の機能train_test_splitがある．下記ではirisd.data（説明変数），irisd.target（従属変数）をランダムに分割してくれる．

In[19] from sklearn.model_selection import train_test_split
In[20]X_train, X_test, y_train, y_test = train_test_split(irisd.data, irisd.target, test_size = 0.3) #テスト：訓練 = 3:7に分割

X教授：次にMLPClassifierというクラス分類用ニューラルネットワークを使う

>>> from sklearn.neural_network import MLPClassifier
>>> clf = MLPClassifier(hidden_layer_sizes = (100,100,),solver = “sgd”,max_iter = 10000) #隠れ層が2層のニューラルネットワークを生成
clf.fit(X_train, y_train) #訓練データでトレーニング
clf.score(X_test,y_test) #テストデータでテスト
Out[244] : 0.9777777777777777 #97%で一致．

Aさん：わ！重回帰分析とほぼ同じ手順で，fit(), score()を使ってできちゃうんですね．
X教授：ニューラルネットワークといっても基本は，重回帰分析と同じだよ．hidden_layer_sizesを変えて試してみると，勉強になる．次回はあたらしく「多変量解析」に進むことにしよう．
Aさん：楽しみです．

参考文献

1) 谷合廣紀 著，辻　真吾 監修：Pythonで理解する統計解析の基礎，技術評論社 (2018).