B-2) 主成分分析（Principal Component Analysis : PCA）

社会システム科学（12/11）

主成分分析

- 多数の変数で説明されるデータ
  → 変数を合成したより少ない変数（＝主成分）でデータを説明
  ＝データの次元圧縮

例1）身長＋体重 → 身体の大きさ

例2）勤務先＋役職＋年収 → 信用度

- 主成分の意味
  → どの変数がどの程度合成されているかによって利用者が推定

［演習］Google ColabでPCAを使ってみる

［準備］演習用データの準備

- まずは手元で演習用データを準備します

演習用のデータ

- 下記のようなデータを分析する（テストの点数のつもり）

Name	Math	Sci	Lang	Eng	Soc
Tanaka	89	90	67	46	50
Sato	57	70	80	85	90
Suzuki	80	90	35	40	50
Honda	40	60	50	45	55
Kawata	78	85	45	55	60
Yoshida	55	65	80	75	85
Saito	90	85	88	92	95

データの準備

1. 前の表をExcelなどで入力し保存（ファイル名は pca_sample.xlsx など）
2. これをCSV形式で保存する（ファイル → 名前をつけて保存）

2-1. ファイル名は「pca_sample.csv」

2-2. ファイル形式は「CSV UTF-8」

3. 保存するときに下のようなダイアログが出たら「はい」を選択

Google ColabでPCAを使ってみる

- 以降はGoogle Colab上で作業します。

必要なパッケージの読み込み

1. まずプロットに日本語を表示するためのパッケージをインストール

```
!pip install japanize-matplotlib
```

2. 次に必要なパッケージを読み込む

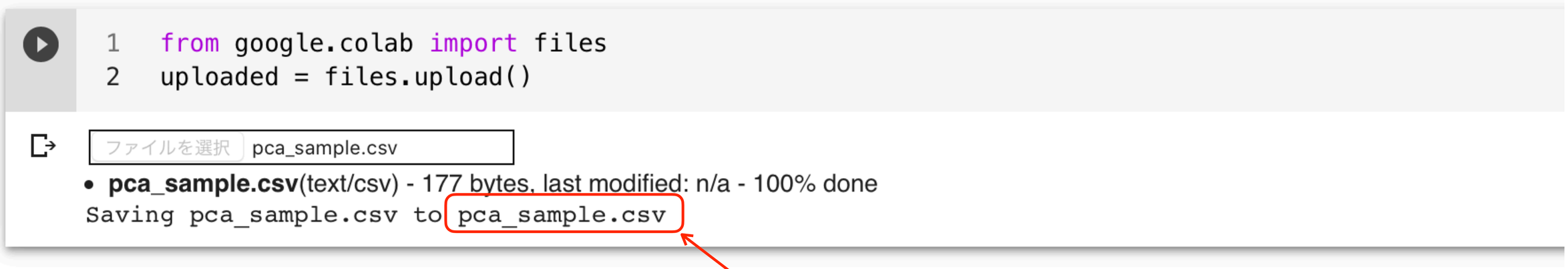
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import japanize_matplotlib
from sklearn.decomposition import PCA ←――― PCAを読み込む

japanize_matplotlib.japanize()
```

CSVファイルをGoogle Colabにアップロード

3. 以下を実行するとファイルアップロード用のダイアログが開く

```
from google.colab import files
uploaded = files.upload()
```

4. pca_sample.csv をアップロード



これがアップロードされたファイル名

CSVファイルの読み込み

5. pandasを使ってCSVファイルを読み込む

```
scores = pd.read_csv("pca_sample.csv", index_col=0, header=0)
```

アップロードされたファイル名を指定する

6. データのサイズを確認

```
scores.shape
```

```
(7, 5)
```

← データ数(学生7人)×特徴数(科目5科目)

7. データの確認（実行例は右）

```
scores
```

```
1  scores
```

Name	Math	Sci	Lang	Eng	Soc
Tanaka	89	90	67	46	50
Sato	57	70	80	85	90
Suzuki	80	90	35	40	50

PCAの実行

8. PCAオブジェクトの生成

```
pca = PCA(n_components = 'mle', whiten = False)
```

9. PCAの実行

```
pca.fit(scores)
```

［文法］PCAオブジェクトの生成

［書式］

```
pca = sklearn.decomposition.PCA(n_components = 整数,
                                whiten = True or False,
                                random_state = None or 整数)
```

［主なオプション］

- n_components
  - 主成分をいくつ求めるか（個数）
  - 'mle' を指定すると最尤推定により個数を自動的に求める
  - 0～1の実数を与えると累積寄与率（後述）がその値になるまで求める
- whiten：白色化（規格化）を行うかどうか
- random_state：ランダムの種は None か整数で指定

［文法］PCAの実行（主成分を求める）

［書式］

```
pca.fit(input)
```

［入力データ］

- 種類：numpy.array または numpy.matrix
- サイズ：データ数×特徴数

結果の確認（1/2）

10.主成分の個数

```
pca.n_components_
```

```
3
```

求められた主成分の数

11.寄与率の確認

```
pca.explained_variance_ratio_
```

```
array([0.6688013 , 0.28791087, 0.04119209])
```

寄与率

- 寄与率：端的に言えば各主成分の重要性を表す
  - 各主成分によって説明できるデータの割合を示す

```
array([ 0.6688013 , 0.28791087, 0.04119209])
```
← 第1主成分の寄与率

- 累積寄与率：主成分の寄与率を足し合わせたもの
  - 選択した複数の主成分によって説明できるデータの割合を表す
    → これが十分に大きくなるまで主成分を選択する

［累積寄与率の計算の仕方］

```
np.cumsum(pca.explained_variance_ratio_)
```

```
array([0.6688013 , 0.95671218, 0.99790427])
```
← 第2主成分までで95.67%

結果の確認（2/2）

12.主成分負荷量（因子負荷量という場合も）の確認

```
pca.components_
```

```
array([[-0.04318455, -0.11661043,  0.55136578,  0.60073709,  0.56537406],
       [-0.84543226, -0.51948621, -0.08791982, -0.08720053,  0.00667425],
       [ 0.02427423, -0.12015885,  0.8167789 , -0.37523513, -0.4207653 ]])
```

主成分負荷量

- 各特徴量の主成分への影響力 → ここから主成分の意味を推定する

		第1変数（算数）	第2変数（理科）	第3変数（国語）	第4変数（英語）	第5変数（社会）	
第1主成分	array([[	-0.04318455,	-0.11661043,	0.55136578,	0.60073709,	0.56537406	],
第2主成分	[	-0.84543226,	-0.51948621,	-0.08791982,	-0.08720053,	0.00667425	],
第3主成分	[	0.02427423,	-0.12015885,	0.8167789 ,	-0.37523513,	-0.4207653	]])

国語，英語，社会の負荷量が高い → 第1主成分は文系科目に関係ありそう

データの変換（次元圧縮）

13. データを主成分空間で表現＝次元圧縮

```
x = pca.transform(scores)
x
```

圧縮後のデータを確認

```
array([[-21.21097689, -21.47715546,  16.13893274],
       [ 35.71460142,  11.68959258,  -3.08132203],
       [-42.0704435 , -10.53162768,  -7.96504946],
       [-22.74370588,  37.14882026,   2.94042802],
       [-21.22256751,  -8.3637958 ,  -9.08119457],
       [ 27.54978153,  16.81652223,   3.32710151],
       [ 43.98331082, -25.28235614,  -2.27889621]])
```

1人目：5次元を3次元で表現

データをプロットしてみる

13.プロット

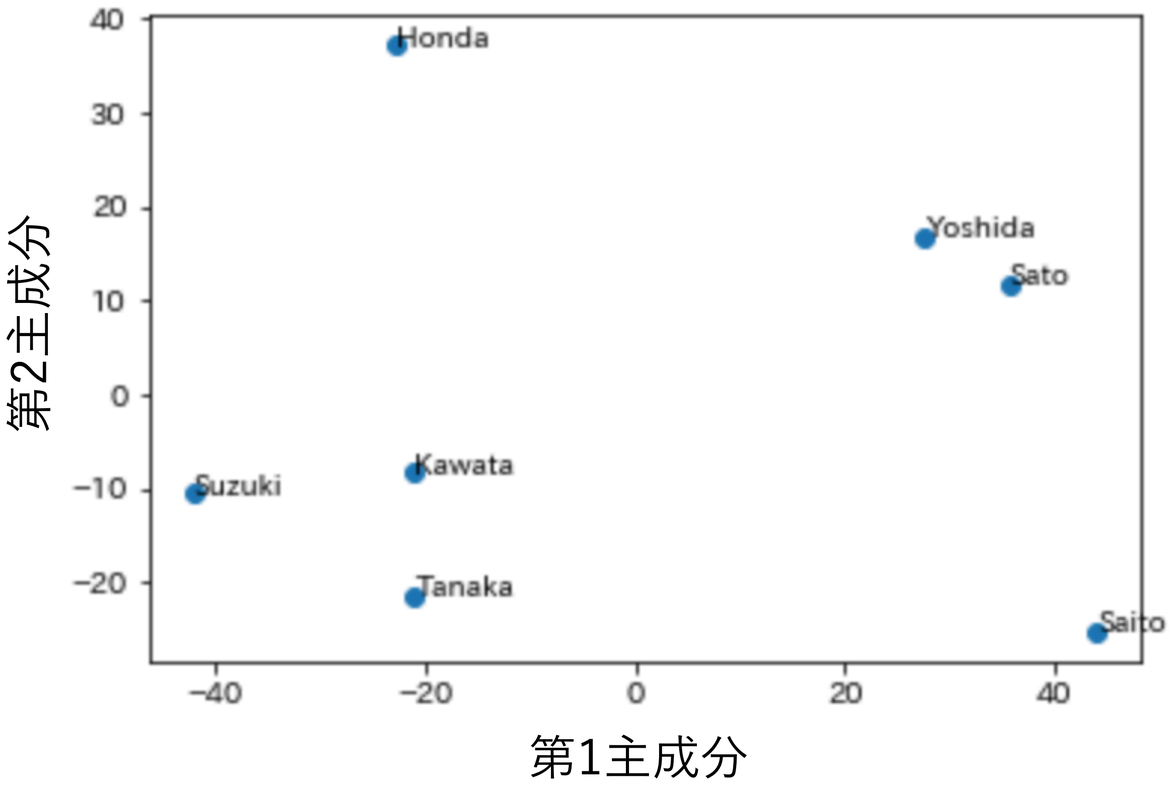
```
plt.scatter(x[:,0],x[:,1])
for d, l in zip(x, scores.index.values):
    plt.text(d[0], d[1], l)
```

第1主成分と第2主成分をプロット

ここの空白が抜けるとエラーになるので注意

プロットの解釈

- 主成分負荷量から意味づけ
  - 第1主成分：文系科目
  - 第2主成分：理系科目（逆向き）



  - どちらもいい：Saito
  - 文系強い：Yoshida, Sato
  - 理系が強い：Suzuki, Kawata, Tanaka
  - どちらもいまいち：Honda

［演習］2023年日本の野球チームのデータを分析してみよう

- BEEFにある 2023_japan_baseball.csv にPCAを適用してみよう。