

見えないデータの関係を可視化する

多次元尺度構成法（Multi Dimensional Scaling:MDS）

社会システム科学 (B-1)

多次元尺度構成法

- ・ ベクトル空間の可視化（低次元化）に使われる手法の1つ
 - ・ 位相を保ったまま低次元化 = 遠い/近いベクトルを互いに遠く/近く配置
- ・ 探索的データ分析（EDA : Exploratory Data Analysis）に利用される
※データの概要を把握するための分析ステージ

PythonにおけるMDSの利用

必要なパッケージ

- scikit-learn : 機械学習用パッケージ ← MDSのツールが入っている
 - scipy : 科学技術計算用パッケージ
 - numpy : 数値演算用パッケージ
- pandas : データ処理支援パッケージ

※Google Colabには全て入っているのでインストールする必要はない。

手順1：MDSオブジェクトの生成

[書式]

```
mds = sklearn.manifold.MDS(n_components = 整数,  
                           metric = True or False,  
                           dissimilarity = "precomputed" or "euclidean",  
                           random_state = None or 整数)
```

[オプション]

- n_components : 圧縮後の次元数を指定
- metric : 計量多次元尺度構成法 or 非計量多次元尺度構成法
 - ※ 計量MDS : 相対距離を保持 ⇔ 非計量MDS : 距離の大小のみ保持
- dissimilarity : ベクトル間の距離は予め計算済み (precomputed) or 否か
 - ※ euclideanを指定するとユークリッド距離を計する
- random_state : ランダムの種はランダム (None) or 指定する (整数)
 - ※ ランダムの種を固定すると同じ結果が得られる

手順2：MDSの実行（データの次元圧縮）

[書式]

```
output = mds.fit_transform(input)
```

[入力データ]

- ・ 種類：numpy.array（ベクトル）または numpy.matrix（行列）
- ・ サイズ：データ数×データ数またはデータ数×特徴数

[出力データ（変換後のデータ）]

- ・ 種類：データ数×圧縮後の次元数

1. Google ColabでMDS

必要なパッケージの読み込み

1. まず必要なパッケージを読み込む

```
import sklearn.manifold as skm ← MDSが含まれるサブパッケージを読み込む
import numpy as np
import matplotlib.pyplot as plt
```

データの設定

2. ラベル（後の描画用）と距離マトリックスを用意

```
labels = ["Hyogo", "Wakayama", "Osaka", "Nara", "Shiga", "Kyoto"]  
data = np.array([[ 0, 134,  85, 116, 118,  60],  
                [134,   0,  68,  66, 145, 141],  
                [ 85,  68,   0,  32,  83,  75],  
                [116,  66,  32,   0,  79,  95],  
                [118, 145,  83,  79,   0,  63],  
                [ 60, 141,  75,  95,  63,   0]])
```

データ数 (6)

データ数 (6)

3. データのサイズを確認

```
data.shape
```

(6, 6) ← データ数×データ数になってる

MDSオブジェクトの生成

4. 続いてMDSのオブジェクトを生成

```
mds = skm.MDS(n_components=2, metric=True, dissimilarity="precomputed", random_state=0)
```

二次元に圧縮

データ数×データ数でデータ間の距離行列の場合は precomputed

データの次元圧縮

5. MDSによるデータの次元圧縮

```
data_transformed = mds.fit_transform(data)
```

6. 変換後のデータのサイズ確認

```
data_transformed.shape
```

変換後のデータのプロット

7. 散布図 (scatter plot) の作成

```
plt.scatter(data_transformed[:,0], data_transformed[:,1])
```

x座標

y座標

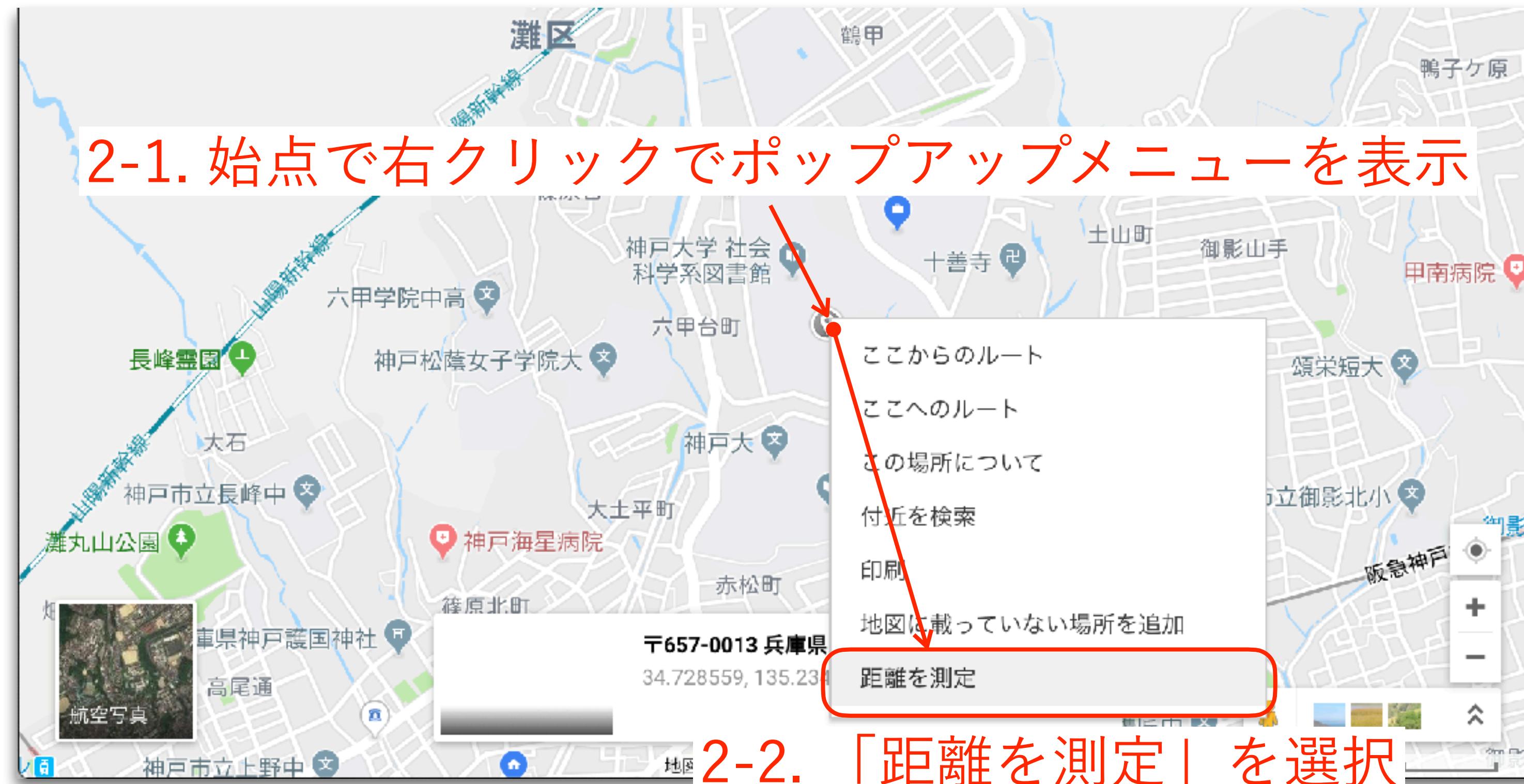
8. ラベルの付与

```
for d, l in zip(data_transformed, labels):  
    plt.text(d[0], d[1], l)
```

2. データを作ってMDSしてみる

Google Mapで距離を測る（1/2）

1. Google Map (<https://maps.google.com>) を開く
2. 始点で右クリックしてポップアップメニューを表示
→ 「距離を測定」を選択



Google Mapで距離を測る (2/2)

3. 始点からの距離を測りたい場所を左クリック（距離が表示される）
※ このまま左クリックしていくと次々と距離が測れる
4. 3で置いたマーカーを移動させて色々な場所の始点からの距離を測る

3-1. 始点からの距離を測りたい場所を左クリック



3-2. 距離が表示される

例) 適当な5カ所から阪急/JR/阪神の駅までの距離を計測 (1/3)

1. データ数 (駅の数) = 7, 特徴数 (= 計測した場所) = 5

```
labels=["Rokko", "Ojikoen", "Rokko-michi", "Maya", "Oishi", "Shinzaike", "Ishiyagawa"]  
data = np.array([[ 517, 1200, 1020, 1070, 1170, 1420, 2040],  
                [ 981, 2140, 328, 1630, 1160, 340, 718],  
                [1220, 614, 1300, 229, 635, 1390, 2240],  
                [1110, 2600, 712, 2130, 1720, 900, 608],  
                [1030, 1040, 1530, 1190, 1480, 1900, 2550]])  
  
←—————データ数 (7)—————→  
↑ 特徴数 (5)
```

例) 適当な5カ所から阪急/JR/阪神の駅までの距離を計測 (2/3)

2. データの形を確認

```
data.shape
```

(5, 7) ← 特徴数×データ数になってる！（データ数×特徴数になってない！）

3. データを転置

```
data = data.T  
data.shape
```

(7, 5) ← データ数×特徴数になっている

MDSオブジェクトの生成

4. 続いてMDSのオブジェクトを生成

```
mds = skm.MDS(n_components=2, metric=True, dissimilarity="euclidean", random_state=None)
```

データ数×特徴数の場合はほぼ euclidean

5. 以下の手順は同じ

3. CSVファイルからデータを読み込んでMDS

CSVファイル

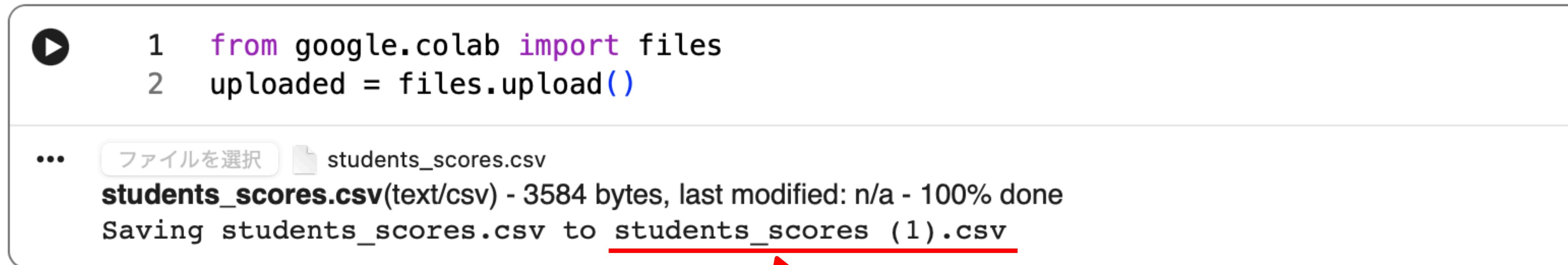
- ・ CSV : Comma-Separated Values = コンマ区切りのテキストファイル
- ・ 様々なツール (Excelなどの表計算ソフトやエディタ) で作成・編集できる
→ データ配布用の形式としてよく利用される

[準備] CSVファイルをGoogle Colabにアップロード

1. BEEF+から students_scores.csv を手元のPCにダウンロード
2. 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()
```

3. sekseki.csv をアップロード



The screenshot shows a Google Colab notebook interface. On the left, there is a code editor with two lines of Python code:

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

On the right, there is a file upload dialog box. It has a '選択' button, a file input field containing 'students_scores.csv', and a status message: 'students_scores.csv(text/csv) - 3584 bytes, last modified: n/a - 100% done'. Below the status message, it says 'Saving students_scores.csv to students_scores (1).csv'. A red arrow points from the text 'このファイル名を後で使うのでチェックしておく' to the underlined part of the file name in the status message.

このファイル名を後で使うのでチェックしておく

Google ColabでのCSVの読み込み

- データ処理用パッケージ pandas インポートする

```
import pandas as pd
```

- CSVファイルを読み込む（read_csvについては次ページ）

```
seiseki = pd.read_csv("students_scores (1).csv", index_col=0, header=0)
```

先ほどチェックしたファイル名

pandasによるCSVファイルの読み込み

[書式]

```
data = pandas.read_csv(ファイル名, index_col=整数, header=整数 or None)
```

[オプション]

- `index_col` : 項目名の書いてある列を指定 (0~)
※指定が無い場合は0列目からデータとして読み込む
- `header` : 見出し行を指定 (0~ or None)
Noneを指定した場合は見出し行がない (0行目からデータ) とする
※指定がない場合は `header=0` と同じ (0行目を見出し行)

読み込んだデータの確認

3. データの行列サイズを確認

```
seiseki.shape
```

(100, 7) ← 学生数(100) × 科目数(7)

4. データを確認

```
seiseki
```

1 seiseki

	Math	Japanese	SocialStudies	English	Science	Arts	PE
Name							
Hana_Ando	75	69	76	100	69	66	61
Keita_Ando	88	63	57	56	69	90	100
Mei_Ando	79	47	64	74	77	53	60
Naoki_Ando	66	66	94	100	73	73	70
Daichi_Aoki	70	86	71	83	38	62	59
...
Taro_Yamamoto	66	82	78	73	92	36	100
Taiki_Yamazaki	63	79	64	67	68	73	47
Yoko_Yamazaki	63	79	100	62	44	74	70
Mao_Yoshida	67	84	90	64	56	49	64
Miku_Yoshida	74	71	43	82	70	57	80

100 rows × 7 columns

次のステップ: [seiseki を使用してコードを生成](#) [New interactive sheet](#)

MDSの実行

5. MDSオブジェクトの生成

```
mds = skm.MDS(n_components = 2, metric = True, dissimilarity="euclidean", random_state=None)
```

6. データの変換

```
data_transformed = mds.fit_transform(seiseki.drop(["Gender"], axis=1))
```

7. プロット

```
colors = seiseki["Gender"].map({"M": "blue", "F": "red"})
plt.scatter(data_transformed[:,0], data_transformed[:,1])
for d, l in zip(data_transformed, seiseki.index.values):
    plt.text(d[0], d[1], l)
```

Genderの列以外

CSVデータの行見出しをラベルとして利用