



総務省統計局

1日で学べる！ はじめてのPython

データサイエンス・オンライン講座《特別編》

2023年**12**月**9**日(土)、**23**日(土)

主催：総務省統計局
運営委託：株式会社ネットラーニング

1

自己紹介



佐久間 貴士

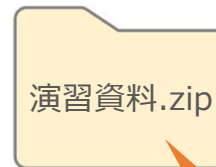
- 千葉県立保健医療大学
健康科学部歯科衛生学科講師
- 立正大学データサイエンス学部 非常勤講師
- 日本大学法学部 非常勤講師

2

本日の講義資料

次の2点をお手元にご用意ください

- ① 20231223_1日で学べる！はじめてのPython.pdf（投影しているPDF資料）
- ② 演習資料.zip

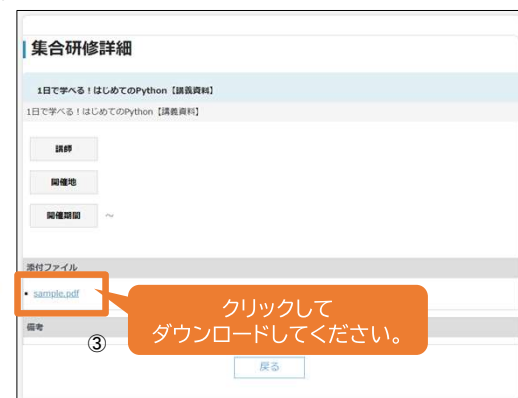


演習資料.zipは解凍し、お手元にcsvファイルが6ファイルあることをご確認ください。

3

講義資料の格納先について（ダウンロード方法）

- ① カリキュラム一覧で ボタンをクリックしてください。
- ② 事前接続テストや講座当日のZoom参加のためのページならびに講義資料等のページが表示されますので、 ボタンの横にあるタイトルをクリックしてください。
- ③ 下記2点の資料をクリックいただきダウンロードしてください。
 - ・ 20231223_1日で学べる！はじめてのPython 講義資料.pdf
 - ・ 演習資料.zip



4

講義のまえに

推奨環境：パソコン（Windows/Mac 問いません）

※ タブレットやスマートフォンでの操作については
ご質問に回答できない場合があります。

5

講義のまえに

Google Colaboratory の起動

<https://colab.research.google.com/> にアクセス



6

講義のまえに

Google Colaboratory の起動



7

講義のまえに

Google Colaboratory の起動



8

講義のまえに

プログラムの保存について

Colaboratoryで作成したノートブックは Google Drive[※] (<https://drive.google.com/>) 内の「Colab Notebooks」に自動で保存されます。

※ Googleが提供しているクラウドストレージで、Googleアカウントでログインします



名前の変更について

Google Drive もしくは Google Colaboratory 、どちらからでも変更することができます。

9

講義のまえに

さまざまなプログラミング言語

- Python
- R
- C言語
- C++
- Java
- PHP



10

講義のまえに

ExcelとPythonのちがい

- Excelの限界
- 少ないコードで実装でき、標準ライブラリやコミュニティから提供されたモジュールが豊富、導入が容易
 - ・ データ量が膨大でも重たくない
 - ・ 再現性の高さ
 - ・ オープンライセンス
 - ・ ライブラリによる効率的なデータ分析
 - ・ 機械学習と深層学習

11

講義のまえに

- **実際に手を動かして、コードを入力してみよう！**
本日作成するコードは、講義終了後にみなさまにお配りします。
講義時間中は、コピー&ペーストではなく、ご自身でコードを打ち込んでみましょう。

12

今日の内容

- 1 データ分析に必要な統計学の基礎を学ぶ
- 2 比較して2変数の関係を考える
- 3 データに基づいて判断を下すための手法を学ぶ
- 4 ビジネスにおける予想と分析結果の報告

13

今日の内容

- 1 データ分析に必要な統計学の基礎を学ぶ
- 2 比較して2変数の関係を考える
- 3 データに基づいて判断を下すための手法を学ぶ
- 4 ビジネスにおける予想と分析結果の報告

14

第1章の内容

1. データの種類とは
2. 1変数の状況と把握
 - (1) 可視化の活用
 - (2) 代表値の活用
3. ビジネスにおける比較
 - (1) 概要
 - (2) 活用

15

1.データの種類とは

分析とは何か

収集した情報の整理，加工，取捨選択を経て分析するプロセスのこと

適切なデータ分析により，数値にもとづく合理的な意思決定が可能となるほか，今まで気づけなかった課題やチャンスに気づきやすくなる

16

1.データの種類とは

データ分析のメリット

- データドリブン（Data Driven）が可能になる
- 迅速な意思決定が可能になる
- 新たなビジネスチャンスを発見できる

17

1.データの種類とは

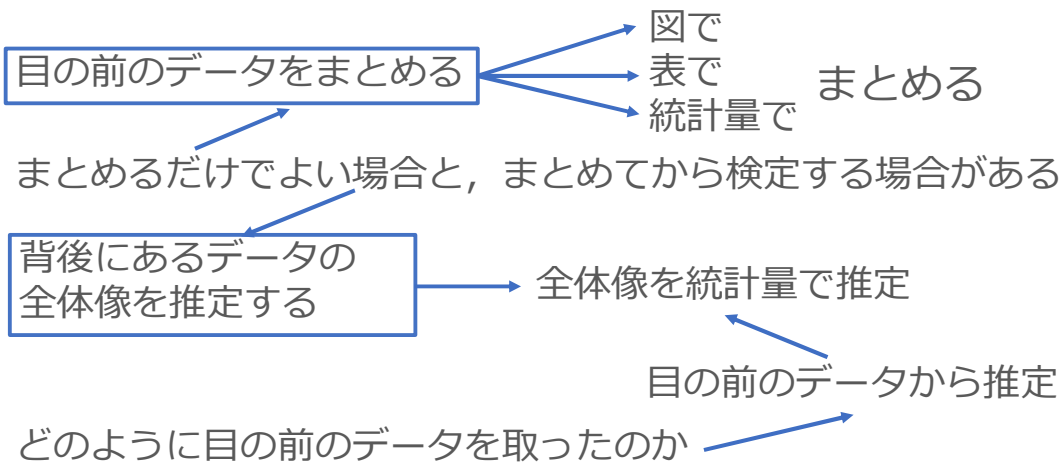
データ分析に用いられる主な10の手法

- | | |
|--------------|---------------|
| • バスケット分析 | • 決定木分析 |
| • アソシエーション分析 | • ABC分析 |
| • クロス集計 | • ロジスティック回帰分析 |
| • 因子分析 | • 主成分分析 |
| • クラスタ分析 | • グレイモデル |

18

1. データの種類とは

統計的処理の方針



19

2. 1変数の状況と把握 (1) 可視化の活用

1次元データを扱う

- 体位のデータ（体重，身長，その他）
- 医学データ（血圧，血糖値，〇〇値）
- 国民の所得
- 試験の点数
- 工業製品，農業生産物の質量など
- その他，数値の集合が統計の中心

押さえるポイント

全体のようなすをつかむ データの可視化 → 分布のかたち
統計量を知る 代表値：平均，分散，四分位数

20

2. 1変数の状況と把握 (1) 可視化の活用

データの整理

- 性質に注目すると以下の3種類に分けられる

1. カテゴリカルデータ（名義データ）
2. 順位データ（順序データ）
3. 計算データ（数量データ）

ア. 体重 イ. 身長 ウ. 性別 エ. A・B・C・Dの成績
 オ. 100点満点のテストの得点 カ. あなたは走るのは速いですかという質問に対して、速い・ふつう・遅いの回答選択肢が用意されている場合 キ. 100m走のタイム ク. 給料 ケ. ある道路を自動車が通った台数 コ. 年齢
 サ. 体重を重い・普通・軽いに分けた場合 シ. 好きな食べ物を聞いた場合

21

2. 1変数の状況と把握 (1) 可視化の活用

データの整理

- データの性質を基礎にした分類以外の分類でよく使われるもの

1. 時系列データ：時間の流れとともに観測して得られるデータ
2. 横断面データ：一時点のみにおけるデータ

縦断的研究, 横断的研究

ア. あるクラスで一斉に行ったテストの結果 イ. A君の前期, 後期の成績
 ウ. A君の前期の統計学, 環境学, 論理学の成績 エ. 東京における1年間365日の気温
 オ. 平成28年4月1日の全国各地の気温
 カ. 4月に行った学生健康診断の結果 キ. ある人の過去10年間の健康診断の結果

22

2. 1変数の状況と把握 (1) 可視化の活用

重要な統計量

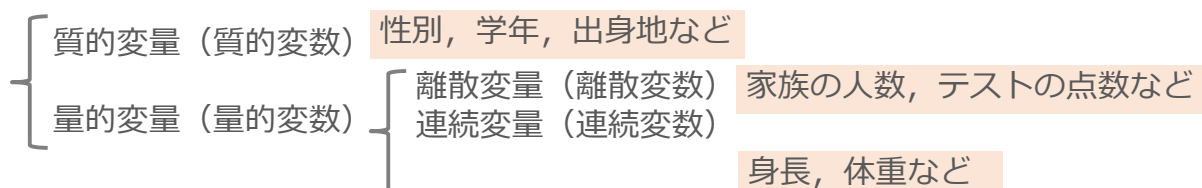
- 平均 (mean) : $\mu = \frac{1}{n} \sum_{i=1}^n x_i$
- 分散 (variance) : $\sigma^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2$
- 標準偏差 (standard deviation, sd) : $\sigma = \sqrt{\sigma^2}$
- 四分位数 (Quantile) : Q_1, Q_2, Q_3
- メジアン, 中央値 (median) : Q_2

Averageも平均を表す概念だが、意味がやや広く、メジアン等の「真ん中」をも指す

23

2. 1変数の状況と把握 (1) 可視化の活用

データの種類



質的変量と量的変量

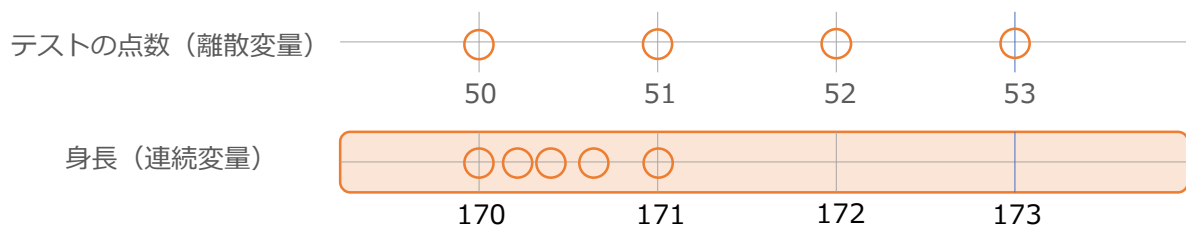
- 数値が量的な変量を持つ変量を量的変量, 意味を持たないものを質的変量という。
- 質的変量か量的変量かを見分けるには, 「**算術平均をとって意味があるか**」を考えてみる方法がある。

24

2. 1変数の状況と把握 (1) 可視化の活用

離散変量と連続変量

- 離散変量は家族の人数やテストの点数など、とびとびの値しかとらない変数である。
- 一方、身長と体重などは正確に測ろうとする場合、無限に細かい数値になる。(身長171.2865...cm) このような変量は連続変量である。



データの種類によって、まとめ方が異なる

25

2. 1変数の状況と把握 (1) 可視化の活用

データの尺度

- データの分類方法としては、**尺度**による分類方法もある。



26

2. 1変数の状況と把握 (1) 可視化の活用

データの尺度

- i. 名義尺度（性別，出身地など）
データ同士を区別するためにつけたもの。性別で，男-1,女-2などとしているが，男女を入れ替えても問題ない。
- ii. 順序尺度（テストの順位，成績評価など）
テストの順位や成績評価など，順番に意味があるものである。これは，入れ替えることはできない。
- iii. 間隔尺度（テストの点数，日付など）
テストの点数のように，順番に意味があり，さらにそれが等間隔に並んでいるもの。比例尺度との違いは，ゼロが絶対的な意味を持つかどうか。
- iv. 比例尺度（身長，体重，家族の人数など）
比例尺度ともいう。体重40kgは20kgの2倍というように，比にも意味がある。

27

2. 1変数の状況と把握 (2) 代表値の活用

ライブラリのインポート

ライブラリ（モジュール）
プログラムファイルで、複数の関数が
定義されている。

- numpy
高度な数値計算と科学技術計算を支援するために設計された強力なライブラリ
(多次元配列, ブロードキャスト, 数学関数, など)
- pandas
データ操作と分析のための効果的なデータ構造を提供するライブラリ
(データフレーム, データ操作, 時系列データ処理, など)

28

2. 1変数の状況と把握 (2) 代表値の活用

ライブラリのインポート

```
# pandas を pd という名前でインポート
import pandas as pd
```

```
# numpy を np という名前でインポート
import numpy as np
```

(ハッシュ記号)をつけることで、プログラムの内容を説明する「コメント」を記述できます

以降は“pd.関数名”, “np.関数名”と入力すればそれぞれの機能が使える

基本的に「import ライブラリ名」のように読み込む
「from ライブラリ名 import モジュール名.関数名」でもOK

29

2. 1変数の状況と把握 (2) 代表値の活用

ライブラリのインポート

```
import numpy as np
import pandas as pd
```

```
# google.colabからファイルをインポート
from google.colab import files
```

```
# uploadするファイルを選択できるインタフェースを表示させる
uploaded = files.upload()
```

ファイル選択 選択されていません

演習資料.zipの中の
sample_cross.csvを
選択してください

30

2. 1変数の状況と把握 (2) 代表値の活用

ライブラリとデータのインポート

```
import io
```

```
# Pandasのread_csv関数を使って、csvファイルを読み込み、  
変数ad_dfに格納
```

```
ad_df =  
pd.read_csv(io.BytesIO(uploaded['sample_cross.csv']))
```

```
# ad_dfをprint関数で表示させる  
# print関数は()内を表示させる関数  
print(ad_df)
```

```

  広告  購入  性別  年齢
0  B  しなかった  男性  31
1  B  しなかった  女性  28
2  A  しなかった  女性  25
3  A  しなかった  男性  31
4  B  しなかった  男性  33
... ..
988 B  しなかった  女性  27
989 B  しなかった  男性  25
990 B  しなかった  男性  32
991 B  しなかった  女性  32
992 B  しなかった  女性  28
[993 rows x 4 columns]
```

31

2. 1変数の状況と把握 (2) 代表値の活用

ライブラリとデータのインポート

```
# ageという変数を用意  
# numpyの中にあるarray関数（配列）を使って、先ほど定義した  
ad_dfの中の[年齢]というカラムから100件データを並べる
```

```
age = np.array(ad_df['年齢'][:100])  
age
```

```
array([31, 28, 25, 31, 33, 31, 30, 30, 24, 30, 22, 23, 31, 25, 22, 23, 27,  
       27, 27, 27, 29, 32, 32, 32, 26, 26, 29, 26, 25, 25, 28, 30, 24, 30,  
       28, 30, 27, 25, 32, 32, 28, 29, 30, 30, 32, 23, 31, 25, 32, 25, 31,  
       31, 30, 26, 31, 28, 28, 27, 28, 29, 28, 23, 24, 23, 27, 23, 28, 32,  
       30, 24, 23, 29, 24, 31, 28, 25, 31, 33, 31, 30, 30, 24, 30, 22, 23,  
       31, 25, 22, 23, 27, 27, 27, 29, 32, 32, 26, 26, 29])
```

32

2. 1変数の状況と把握 (2) 代表値の活用

ライブラリとデータのインポート

```
# 変数age_df にPandasのDataFrameというデータ構造で格納
age_df = pd.DataFrame({'年齢':age})
```

```
# age_df を表示
age_df
```

DataFrameは二次元配列でデータを表示させる記述です
各レコードには「インデックス」という番号が表示されます

	年齢
0	31
1	28
2	25
3	31
4	33
...	...
95	32
96	32
97	26
98	26
99	29

100 rows × 1 columns

33

2. 1変数の状況と把握 (2) 代表値の活用

データの中心の指標 (平均値 : mean)

```
# sum関数で求めた (合計) をlen関数で求めた配列数で割る = 算術平均の計算
sum(age) / len(age)
```

```
# 上記と同じ作業をNumpyのmean関数でも求めることができる
```

```
np.mean(age)
```

```
# 先ほどデータフレームage_dfに入れた関数を平均するというコード
age_df.mean()
```

```
27.8
```

```
27.8
```

```
年齢 27.8
dtype: float64
```

34

2. 1変数の状況と把握 (2) 代表値の活用

データの中心の指標 (中央値 : median)

データを大きさの順に並べた時にちょうど中央に位置する値

```
# Numpyのsort関数 (小さい順に並ぶ) でageを読み込み、変数sorted_ageに格納
```

```
sorted_age = np.sort(age)
```

```
# print()関数と同じ sorted_ageを表示させる
```

```
sorted_age
```

```
↳ array([22, 22, 22, 22, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 23, 24, 24, 24, 24,
        24, 24, 25, 25, 25, 25, 25, 25, 25, 25, 25, 25, 26, 26, 26, 26, 26, 26,
        27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28, 28, 28, 28, 28, 28,
        28, 28, 28, 28, 29, 29, 29, 29, 29, 29, 29, 29, 30, 30, 30, 30, 30, 30,
        30, 30, 30, 30, 30, 30, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31,
        31, 31, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33])
```

35

2. 1変数の状況と把握 (2) 代表値の活用

データの中心の指標 (中央値 : median)

```
n = len(sorted_age) # 変数nにsorted_ageの個数を入れる
```

```
# ifはexcelのifと同義で、論理式を分岐させる指示
```

```
# == 2つの値を比較して等しいかどうかを調べる
```

```
if n % 2 == 0: # %は余りを表示させる数式
```

```
    m0 = sorted_age[n//2 - 1] # //は切り捨ての除算
```

```
    m1 = sorted_age[n//2] # 偶数、奇数の判定
```

```
    median = (m0 + m1) / 2
```

```
else:
```

```
    median = sorted_age[(n+1)//2 - 1]
```

```
median
```

```
↳ 28.0
```

36

2. 1変数の状況と把握 (2) 代表値の活用

データのばらつきの指標（分散と標準偏差）

平均が40歳でも、全員が40歳の場合のデータと、0歳が半分、80歳が半分のデータでは全く異なる。ばらつきを求めるために**偏差(deviation)**を計算する。

```
# Numpyのmean（平均）を求める
```

```
mean = np.mean(age)
```

```
# ageから平均を引き、平均との差=ばらつき（偏差）を求める
```

```
deviation = age - mean
```

```
deviation
```

```
array([ 3.2,  0.2, -2.8,  3.2,  5.2,  3.2,  2.2,  2.2, -3.8,  2.2, -5.8,
        -4.8,  3.2, -2.8, -5.8, -4.8, -0.8, -0.8, -0.8, -0.8,  1.2,  4.2,
         4.2,  4.2, -1.8, -1.8,  1.2, -1.8, -2.8, -2.8,  0.2,  2.2, -3.8,
         2.2,  0.2,  2.2, -0.8, -2.8,  4.2,  4.2,  0.2,  1.2,  2.2,  2.2,
         4.2, -4.8,  3.2, -2.8,  4.2, -2.8,  3.2,  3.2,  2.2, -1.8,  3.2,
         0.2,  0.2, -0.8,  0.2,  1.2,  0.2, -4.8, -3.8, -4.8, -0.8, -4.8,
         0.2,  4.2,  2.2, -3.8, -4.8,  1.2, -3.8,  3.2,  0.2, -2.8,  3.2,
         5.2,  3.2,  2.2,  2.2, -3.8,  2.2, -5.8, -4.8,  3.2, -2.8, -5.8,
        -4.8, -0.8, -0.8, -0.8, -0.8,  1.2,  4.2,  4.2,  4.2, -1.8, -1.8,
         1.2])
```

37

2. 1変数の状況と把握 (2) 代表値の活用

データのばらつきの指標（分散と標準偏差）

```
# ageのdfのコピーを作る
```

```
summary_df = age_df.copy()
```

```
# int型とタイプを指定
```

```
summary_df['偏差'] = deviation.astype(int)
```

```
summary_df
```

シングルクォーテーションを記載
することで日本語で記述できる！

	年齢	偏差
0	31	3
1	28	0
2	25	-2
3	31	3
4	33	5
...	...	...
95	32	4
96	32	4
97	26	-1
98	26	-1
99	29	1

100 rows × 2 columns

38

2. 1変数の状況と把握 (2) 代表値の活用

データのばらつきの指標 (分散と標準偏差)

```
summary_df.mean()
np.mean(deviation **2)
```

```
↳ 9.7
```

```
# Numpyのvar関数でも同様に分散を求めることができる。
np.var(age)
```

```
↳ 9.7
```

39

2. 1変数の状況と把握 (2) 代表値の活用

データのばらつきの指標 (分散と標準偏差)

```
# Pandasのvarを使って分散を計算してみると…
age_df.var()
```

```
年齢      9.79798
dtype: float64
```

40

2. 1変数の状況と把握 (2) 代表値の活用

データのばらつきの指標 (分散と標準偏差)

```
summary_df['偏差二乗'] = np.square(deviation)
summary_df

summary_df.mean()
```

```
📄 年齢    27.80
   偏差    0.25
   偏差二乗  9.70
   dtype: float64
```

	年齢	偏差	偏差二乗
0	31	3	10.24
1	28	0	0.04
2	25	-2	7.84
3	31	3	10.24
4	33	5	27.04
...	...	...	...
95	32	4	17.64
96	32	4	17.64
97	26	-1	3.24
98	26	-1	3.24
99	29	1	1.44

100 rows × 3 columns

41

2. 1変数の状況と把握 (2) 代表値の活用

データのばらつきの指標 (分散と標準偏差)

```
import statistics
import math

print(statistics.mean(age))
print(sum(age) / len(age))
print(statistics.median(age))
print(statistics.mode(age))
print(statistics.variance(age))
print(statistics.pstdev(age))
```

```
📄 27
   27.8
   28.0
   30
   9
   3.0
```

42

3. ビジネスにおける比較 (1) 概要

ABテスト

- AパターンとBパターンでどちらが効果があるのかをテストする
= 複数のものを比較するテスト
- Webマーケティングの領域で非常に頻繁に使われている

例) Webサイトのデザイン, インターネット広告,
ランディングページ最適化, メール配信のセグメント

43

3. ビジネスにおける比較 (1) 概要

カイ二乗検定 (独立性の検定 : test for independence)

- 2つの変数XとYについて, 関係があるのか, それとも独立であるのか
 - 帰無仮説 H_0 : 属性間には関係がない「XとYは独立である」
 - 対立仮説 H_1 : 属性間には関係がある「XとYは独立ではない」
- 独立性の検定にはカイ二乗分布が使われるのでカイ二乗検定 (chi-square test) と呼ばれる

大前提: カテゴリカルデータ (名義データへの適用)

44

3. ビジネスにおける比較 (1) 概要

帰無仮説と対立仮説

- 帰無仮説 (null hypothesis) H_0 :
「有意差がない」という仮説
「無に帰すことも予定している」仮説であり、通常は
否定したい仮説を設定する
- 対立仮説 (alternative hypothesis) H_1 :
「有意差がある」という仮説
帰無仮説が間違っていると確信されたとき (棄却された) に
採用される

45

3. ビジネスにおける比較 (2) 活用

ABテスト (仮説を立てる)

例 1

ある商品の広告プランとしてAとBがあり、どちらが
より購買意欲につながっているのか

- もし広告の種類と購入の有無が独立なら購入の割合に変化はない
- そうでないなら、購入の割合に差が出るはず

 独立性の検定が使える

46

3. ビジネスにおける比較 (2) 活用

ABテスト

例 1

ある商品の広告プランとしてAとBがあり, どちらがより購買意欲につながっているのか

	購入した	購入しなかった	合計
広告A	60	1,000	1,060
広告B	40	400	440
合計	100	1,400	1,500

47

3. ビジネスにおける比較 (2) 活用

ABテスト (クロス集計表を作成する)

```
# pandas を pd という名前でインポート
import pandas as pd
# scipy から 一部分 (stats) のみインポート

from scipy import stats
# カラム (columns。縦列) のタイトルを付与。
df = pd.DataFrame([[60, 1000], [40, 400]],
                   index=['A', 'B'], columns=['購入した', '購入していない'])
df
```

	購入した	購入していない
A	60	1000
B	40	400

48

3. ビジネスにおける比較 (2) 活用

ABテスト (検定を行う)

#カイ二乗検定で検定していくため、以下のとおり変数を指定
`chi2, p, dof, exp = stats.chi2_contingency(df, correction=False)`

```
print("期待度数", "\n", exp)
print("自由度", "\n", dof)
print("カイ二乗値", "\n", chi2)
print("p値", "\n", p)
```

＼マークには注意！

Windowsでは円記号「¥」で、
Macは「\」記号で表示されます

```
期待度数
[[ 70.66666667 989.33333333]
 [ 29.33333333 410.66666667]]
自由度
1
カイ二乗値
5.880911541288903
p値
0.015305895674955605
```

結果はカイ二乗値(chi2), p値(p), 自由度(dof), 期待度数(exp)で出力される

標準だとイエイツの修正が入るので、
correction=Falseにして、補正が入らないように設定

49

3. ビジネスにおける比較 (2) 活用

ABテスト (期待度数について)

例 1

ある商品の広告プランとしてAとBがあり、どちらがより購買意欲につながっているのか

来客数の合計が1,500名

100名が商品を購入している

広告AもBも1/15の割合で購買意欲につながっている

期待値

A: $1060 * (1/15) = 70.7$
 B: $440 * (1/15) = 29.3$

$$\frac{(\text{観測データ} - \text{期待度数})^2}{\text{期待度数}}$$

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

50

3. ビジネスにおける比較 (2) 活用

ABテスト (実行結果の確認)

例 1

ある商品の広告プランとしてAとBがあり、どちらがより購買意欲につながっているのか

<pre>chi2, p, dof, exp = stats.chi2_contingency(df, corr</pre>	自由度 = (行数-1) * (列数-1)
<pre>print("期待度数", "\n", exp) print("自由度", "\n", dof) print("カイ二乗値", "\n", chi2) print("p値", "\n", p)</pre>	自由度1の時のカイ二乗分布の値 5%で3.84, 1%で6.63, 0.5%で7.88
<pre>期待度数 [[70.66666667 989.33333333] [29.33333333 410.66666667]] 自由度 1 カイ二乗値 5.880911541288903 p値 0.015305895674955605</pre>	5.88は3.84を上回っているが6.63を下回っている AとBが同じ前提の時 この事象は5%以下の確率でしか起きない

51

3. ビジネスにおける比較 (2) 活用

ABテスト (実行結果の確認)

例 1

ある商品の広告プランとしてAとBがあり、どちらがより購買意欲につながっているのか

<pre>chi2, p, dof, exp = stats.chi2_contingency(df, correction=False) print("期待度数", "\n", exp) print("自由度", "\n", dof) print("カイ二乗値", "\n", chi2) print("p値", "\n", p)</pre>	p値 (p-value) 帰無仮説を考えた時にその結果が出る確率 (有意水準と照らし合わせるための数値)
<pre>期待度数 [[70.66666667 989.33333333] [29.33333333 410.66666667]] 自由度 1 カイ二乗値 5.880911541288903 p値 0.015305895674955605</pre>	有意水準 > p値 = 有意差がある

52

3. ビジネスにおける比較 (2) 活用

ABテスト

例 2

Webサイトから商品購入を促すバナー（画像）を
2種類用意し、どちらが良いかテストすることにした

	バナーA	バナーB
クリック数	10,000	10,000
コンバージョン数	400	340
コンバージョン率	???	???

※コンバージョン率：アクセスしてきたユーザーのうち、どのくらいがコンバージョンに至ったかを示す数値
コンバージョン：（訪問者がWebサイトの）目標としているアクションを起こしてくれた状態のこと

53

3. ビジネスにおける比較 (2) 活用

ABテスト（仮説を立てる）

例 2

Webサイトから商品購入を促すバナー（画像）を
2種類用意し、どちらが良いかテストすることにした

- H_0 : バナーAとバナーBにはコンバージョン数の差異を決定づける明らかな差が「ない」（独立である：関連がない）
- H_1 : バナーAとバナーBにはコンバージョン数の差異を決定づける明らかな差が「ある」（独立ではない：関連がある）

仮説：両バナーにおいてコンバージョン数の差異を決定づける明らかな差が「ある」に違いない

54

3. ビジネスにおける比較 (2) 活用

ABテスト (ABテストができるようクロス集計を行う)

例 2

Webサイトから商品購入を促すバナー（画像）を
2種類用意し、どちらが良いかテストすることにした

	good click	no good click	total
バナーA	?	?	?
バナーB	?	?	?
合計	?	?	?

※コンバージョンがあったクリックとなかったクリックに分ける必要がある

55

3. ビジネスにおける比較 (2) 活用

ABテスト

例 2

Webサイトから商品購入を促すバナー（画像）を
2種類用意し、どちらが良いかテストすることにした

	good click	no good click	total
バナーA	400	9,600	10,000
バナーB	340	9,660	10,000
合計	740	19,260	20,000

56

3. ビジネスにおける比較 (2) 活用

ABテスト (クロス集計表を作成する)

```
import pandas as pd
from scipy import stats

df = pd.DataFrame([[400, 9600], [340, 9660]],
                  index=['バナーA', 'バナーB'], columns=['goog click', 'no
good click'])
df
```



	goog click	no good click
バナーA	400	9600
バナーB	340	9660




57

3. ビジネスにおける比較 (2) 活用

ABテスト (検定を実施し実行結果を確認する)

```
chi2, p, dof, exp = stats.chi2_contingency(df, correction=False)

print("期待度数", "%n", exp)
print("自由度", "%n", dof)
print("カイ二乗値", "%n", chi2)
print("p値", "%n", p)
```



```
期待度数
[[ 370. 9630.]
 [ 370. 9630.]]
自由度
1
カイ二乗値
5.051780752715333
p値
0.02460064285694141
```

58

今日の内容

- 1 データ分析に必要な統計学の基礎を学ぶ
- 2 比較して2変数の関係を考える
- 3 データに基づいて判断を下すための手法を学ぶ
- 4 ビジネスにおける予想と分析結果の報告

59

第2章の内容

1. クロス集計の軸設定と見方
2. 散布図と相関の調べ方
3. 相関係数と因果関係の違い
4. 時系列データの見方、分析方法

60

1.クロス集計の軸設定と見方

クロス集計とは

数量データの関係性を見る際は相関係数が利用できるが、カテゴリカルデータの関係性を見る際には**クロス集計**を用いるのが便利。分割表とも言う場合もある。

単純にはカテゴリごとの度数を記録した表、ただし、2つ以上の変数を対象とし、その組み合わせで度数を求める。

縦軸：表側（＝原因）どのような視点から調査結果を見たいか ※分析軸
横軸：表頭（＝結果）どのような結果を見たいか

61

1.クロス集計の軸設定と見方

CSVファイルのマウント

```
import numpy as np
import pandas as pd

from google.colab import files

uploaded = files.upload()
```

62

1.クロス集計の軸設定と見方

CSVファイルのマウント

```
import pandas as pd
import io
```

```
df = pd.read_csv('sample_cross.csv')
print(df)
```

```
  広告  購入  性別  年齢
0   B  しなかった  男性  31
1   B  しなかった  女性  28
2   A  しなかった  女性  25
3   A  しなかった  男性  31
4   B  しなかった  男性  33
... ..
988  B  しなかった  女性  27
989  B  しなかった  男性  25
990  B  しなかった  男性  32
991  B  しなかった  女性  32
992  B  しなかった  女性  28
[993 rows x 4 columns]
```

63

1.クロス集計の軸設定と見方

CSVファイルのマウント

```
n=len(df)
print(n)
df.head()
```

```
993
  広告  購入  性別  年齢
0   B  しなかった  男性  31
1   B  しなかった  女性  28
2   A  しなかった  女性  25
3   A  しなかった  男性  31
4   B  しなかった  男性  33
```

64

1.クロス集計の軸設定と見方

CSVファイルのマウント

```
ad_cross=pd.crosstab(df['広告'], df['性別'])
ad_cross
```

性別	女性	男性
広告		
A	200	199
B	297	297

65

1.クロス集計の軸設定と見方

CSVファイルのマウント

```
ad_cross=pd.crosstab(df['購入'], df['広告'])
ad_cross
```

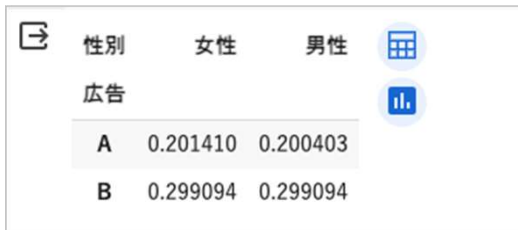
広告	A	B
購入		
した	41	68
しなかった	358	526

66

1. クロス集計の軸設定と見方

CSVファイルのマウント

```
ad_cross=pd.crosstab(df['広告'], df['性別'], normalize=True)
ad_cross
```



性別	女性	男性
広告		
A	0.201410	0.200403
B	0.299094	0.299094

pandas.crosstab()
 第一引数：indexに結果の行見出し
 第二引数：columnsに結果の列見出し
 引数normalizeで全体・行ごと・列ごとに規格化（正規化）

pandas.DataFrame を返す

67

2. 散布図と相関の調べ方

散布図とは

2つの要素からなる1組のデータが得られたときに、**2つの要素の間にある関係（相関関係）**を見るためのグラフ。因果関係（どちらかが原因となって、もう一方が起こる）を示すものではない。QC7つ道具の一つ。

例1) 数学の点数と英語の点数：数学の点数が高い生徒は英語の点数も高い傾向がある（反対も含める）。

例2) 高齢者の運動量と体力：運動量が増えると体力も増える（〃）。

68

2. 散布図と相関の調べ方

相関分析

2つの要素（2変数）間の関係を数値で表現する分析方法。

「相関」とは2つ以上の変数があるときに、「**どれくらい類似しているのか**」という「類似度」を意味する。2つの変量の強弱を数値化したものを「相関係数」という。

「類似度」の強さを「-1から1」までの範囲で表現される。
正（または負）の相関関係が強いほど1（または-1）に近く、相関関係が弱いほど0に近くなる。

69

2. 散布図と相関の調べ方

CSVファイルのマウント

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

```
import pandas as pd  
import io  
  
df = pd.read_csv(io.BytesIO(uploaded['sample_covid19.csv']))  
df
```

70

2. 散布図と相関の調べ方

CSVファイルのマウント

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

sample_covid19.csv
• sample_covid19.csv(text/csv) - 42033 bytes, last modified: 2023/12/17 - 100% done
Saving sample_covid19.csv to sample_covid19.csv

import pandas as pd
import io
df = pd.read_csv(io.BytesIO(uploaded['sample_covid19.csv']))
df
```

	Date	ALL	Hokkaido	Aomori	Iwate	Miyagi	Akita	Yamagata	Fukushima	Ibaraki	...	Ehime	Kochi	Fukuoka	Saga	Nagasaki	Kumamoto	Oi
0	2021/1/1	3249	98	10	4	30	3	3	12	42	...	6	6	158	1	20	30	
1	2021/1/2	3055	77	4	2	4	0	1	13	20	...	6	7	124	5	28	26	
2	2021/1/3	3136	68	10	3	20	3	6	14	52	...	7	11	104	3	30	22	
3	2021/1/4	3333	93	10	0	18	3	5	25	32	...	9	2	128	23	24	34	
4	2021/1/5	4936	79	7	6	37	8	5	25	67	...	25	7	187	10	55	63	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
268	2021/9/26	2146	54	14	2	16	0	1	9	47	...	10	0	88	4	8	14	
269	2021/9/27	1159	19	5	1	6	1	1	3	31	...	2	3	46	12	9	10	
270	2021/9/28	1743	26	25	0	12	6	1	3	25	...	10	4	43	8	5	17	
271	2021/9/29	1980	45	23	1	21	0	1	4	44	...	7	7	44	8	7	19	
272	2021/9/30	1571	26	23	0	14	2	1	5	20	...	14	3	37	6	6	15	

273 rows x 49 columns

71

2. 散布図と相関の調べ方

CSVファイルのマウント

変数dfに格納したデータの何番目を
取得するか指定

df.iloc[:,9]

df.iloc[:,14]

df['Ibaraki']

df['Tokyo']

```
df.iloc[:,9]
```

0	42
1	20
2	52
3	32
4	67
...	...
268	47
269	31
270	25
271	44
272	20

Name: Ibaraki, Length: 273, dtype: int64

```
df.iloc[:,14]
```

0	793
1	829
2	826
3	905
4	1315
...	...
268	302
269	155
270	250
271	268
272	219

Name: Tokyo, Length: 273, dtype: int64

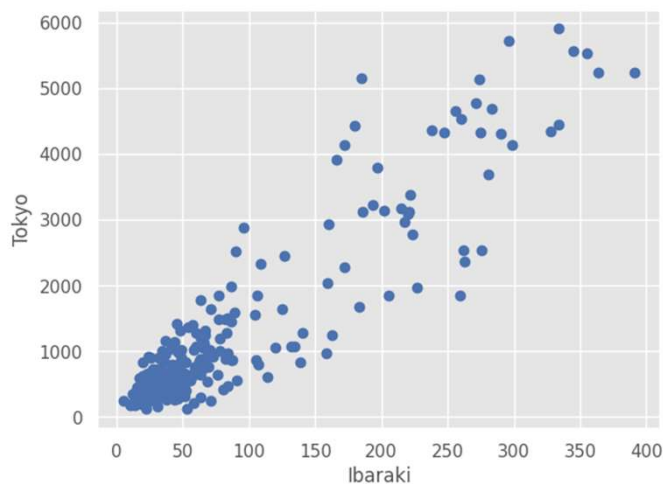
72

2. 散布図と相関の調べ方

プロット（東京都と茨城県の新規感染者数）

```
%matplotlib inline
# グラフツール matplotlibをインポート
import matplotlib.pyplot as plt
# カラーリングするツールをインポート
import warnings

# xlabel、ylabelはX軸、Y軸を指定する
plt.style.use('ggplot')
plt.plot(df.iloc[:,9],
df.iloc[:,14], 'bo')
plt.xlabel('Ibaraki')
plt.ylabel('Tokyo')
plt.show()
```



73

2. 散布図と相関の調べ方

プロット（東京都と茨城県の新規感染者数）：相関係数

```
y_colum='Tokyo'
```

```
# 全体の相関係数を計算し、corrの
matrix上に表示させる
corr_matrix = df.corr()
# 変数y_corrにy_colum(つまりTokyo)
との相関係数を格納
y_corr = corr_matrix[y_colum]
y_corr
```

```
ALL 0.933407
Hokkaido 0.562570
Aomori 0.597597
Iwate 0.720486
Miyagi 0.694157
Akita 0.658737
Yamagata 0.599073
Fukushima 0.857339
Ibaraki 0.919476
Tochigi 0.916396
Gunma 0.889860
Saitama 0.959624
Chiba 0.911269
Tokyo 1.000000
Kanagawa 0.932625
Niigata 0.853796
Toyama 0.835855
Ishikawa 0.793950
Fukui 0.736283
Yamanashi 0.875588
Nagano 0.794002
Gifu 0.681034
Shizuoka 0.848566
Aichi 0.634805
Mie 0.738161
Shiga 0.838877
Kyoto 0.852544
Osaka 0.757124
Hyogo 0.740625
Nara 0.711916
Wakayama 0.751228
Tottori 0.797398
```

```
Ishikawa 0.793950
Fukui 0.736283
Yamanashi 0.875588
Nagano 0.794002
Gifu 0.681034
Shizuoka 0.848566
Aichi 0.634805
Mie 0.738161
Shiga 0.838877
Kyoto 0.852544
Osaka 0.757124
Hyogo 0.740625
Nara 0.711916
Wakayama 0.751228
Tottori 0.797398
Shimane 0.708764
Okayama 0.744315
Hiroshima 0.665415
Yamaguchi 0.670897
Tokushima 0.490078
Kagawa 0.824499
Ehime 0.733994
Kochi 0.660480
Fukuoka 0.882209
Saga 0.786857
Nagasaki 0.789520
Kumamoto 0.856704
Oita 0.722250
Miyazaki 0.761319
Kagoshima 0.828517
Okinawa 0.893809
Name: Tokyo, dtype: float64
```

74

2. 散布図と相関の調べ方

プロット（東京都と茨城県の新規感染者数）：相関係数

```
y_colum='Tokyo'

corr_matrix = df.corr()['Ibaraki']
y_corr = corr_matrix[y_colum]
y_corr
```

0.9194759486275046

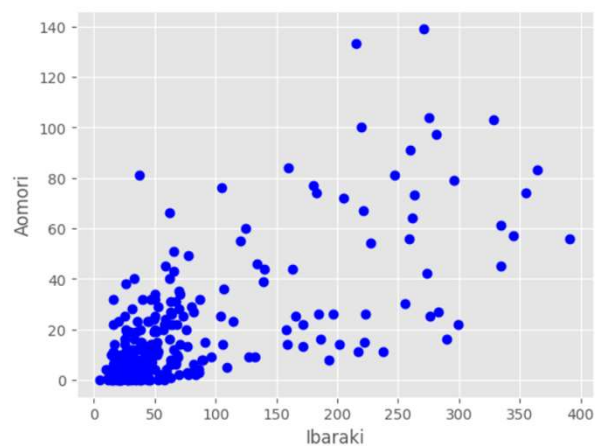
75

2. 散布図と相関の調べ方

プロット（青森県と茨城県の新規感染者数）

```
plt.style.use('ggplot')
plt.plot(df.iloc[:,9],
df.iloc[:,3], 'bo')
plt.xlabel('Ibaraki')
plt.ylabel('Aomori')
plt.show()
```

0.6844695489050351



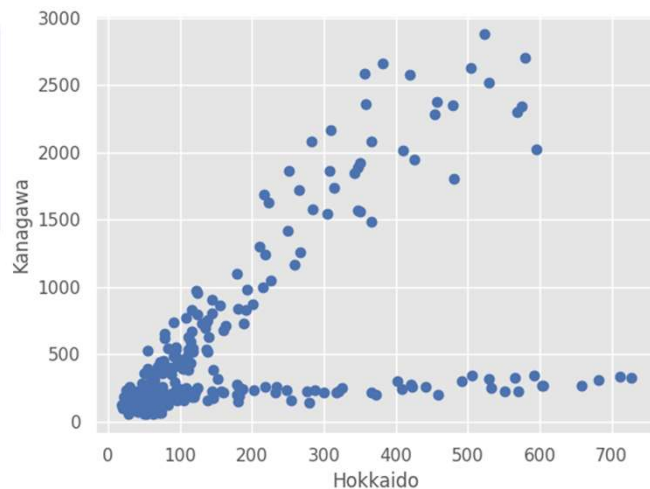
76

2. 散布図と相関の調べ方

プロット（神奈川県と北海道の新規感染者数）

```
plt.style.use('ggplot')
plt.plot(df.iloc[:,2],
df.iloc[:,15],'bo')
plt.xlabel('Hokkaido')
plt.ylabel('Kanagawa')
plt.show()
```

0.5503568815983438

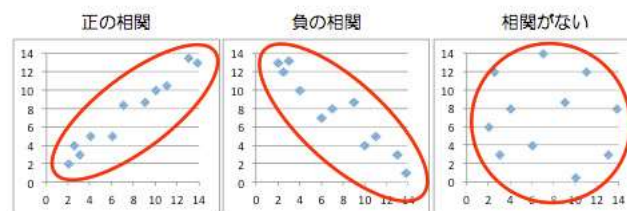


77

3. 相関係数と因果関係の違い ※セクション3の紹介

相関係数

2つの変量の強弱を数値化したものを「相関係数」という。「類似度」の強さを「-1から1」までの範囲で表現される。正（または負）の相関関係が強いほど1（または-1）に近く、相関関係が弱いほど0に近くなる。



なるほど統計学園 : https://www.stat.go.jp/naruhodo/10_tokucho/hukusu.html

78

2. 散布図と相関の調べ方- iris dataset -

irisデータセット

```
import pandas as pd
import seaborn as sns
sns.set()
iris = sns.load_dataset('iris')
titanic = sns.load_dataset('titanic')
```

79

2. 散布図と相関の調べ方- iris dataset -

データセットの利用

```
print(iris.head())
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

80

2. 散布図と相関の調べ方 - iris dataset -

データセットの利用

```
print(titanic.head())
```

```
survived  pclass  sex  age  sibsp  parch  fare  embarked  class
0         0      3  male  22.0    1     0   7.2500        S  Third
1         1      1  female  38.0    1     0  71.2833        C  First
2         1      3  female  26.0    0     0   7.9250        S  Third
3         1      1  female  35.0    1     0  53.1000        S  First
4         0      3  male   35.0    0     0   8.0500        S  Third

who  adult_male  deck  embark_town  alive  alone
0   man        True  NaN  Southampton    no  False
1  woman       False   C   Cherbourg   yes  False
2  woman       False  NaN  Southampton   yes   True
3  woman       False   C   Southampton   yes  False
4   man        True  NaN  Southampton    no   True
```

81

2. 散布図と相関の調べ方 - iris dataset -

データセットの利用

```
print(iris.head())
print(iris.info())
print(iris.shape)
print(iris.ndim)
print(iris.columns)
```

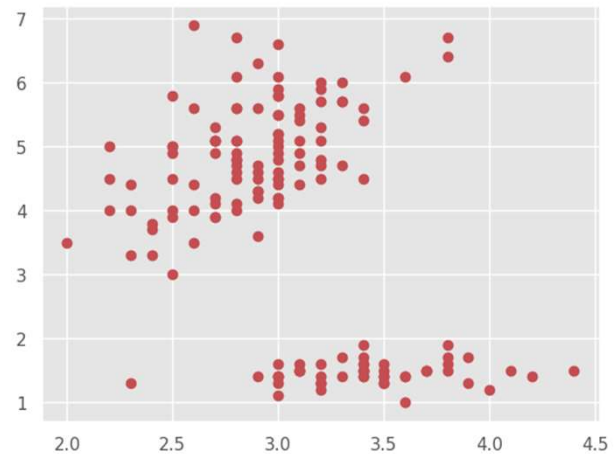
```
sepal_length  sepal_width  petal_length  petal_width  species
0         5.1         3.5         1.4         0.2  setosa
1         4.9         3.0         1.4         0.2  setosa
2         4.7         3.2         1.3         0.2  setosa
3         4.6         3.1         1.5         0.2  setosa
4         5.0         3.6         1.4         0.2  setosa
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
#   Column      Non-Null Count  Dtype
---  -
0  sepal_length  150 non-null    float64
1  sepal_width   150 non-null    float64
2  petal_length  150 non-null    float64
3  petal_width   150 non-null    float64
4  species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None
(150, 5)
2
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width',
      'species'],
      dtype='object')
```

82

2. 散布図と相関の調べ方- iris dataset -

データセットの利用

```
%matplotlib inline
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
plt.plot(iris.iloc[:,1], iris.iloc[:,2],
'ro')
plt.show()
```



83

2. 散布図と相関の調べ方- iris dataset -

データセットの利用

```
iris.corr()
```

	sepal_length	sepal_width	petal_length	petal_width
sepal_length	1.000000	-0.117570	0.871754	0.817941
sepal_width	-0.117570	1.000000	-0.428440	-0.366126
petal_length	0.871754	-0.428440	1.000000	0.962865
petal_width	0.817941	-0.366126	0.962865	1.000000

84

2. 散布図と相関の調べ方- iris dataset -

データセットの利用

```
y_colum='sepal_width'
```

```
corr_matrix = iris.corr()
y_corr = corr_matrix[y_colum]
y_corr
```

```
sepal_length -0.117570
sepal_width  1.000000
petal_length -0.428440
petal_width  -0.366126
Name: sepal_width, dtype: float64
```

85

2. 散布図と相関の調べ方- iris dataset -

データセットの利用

```
print(iris.describe())
```

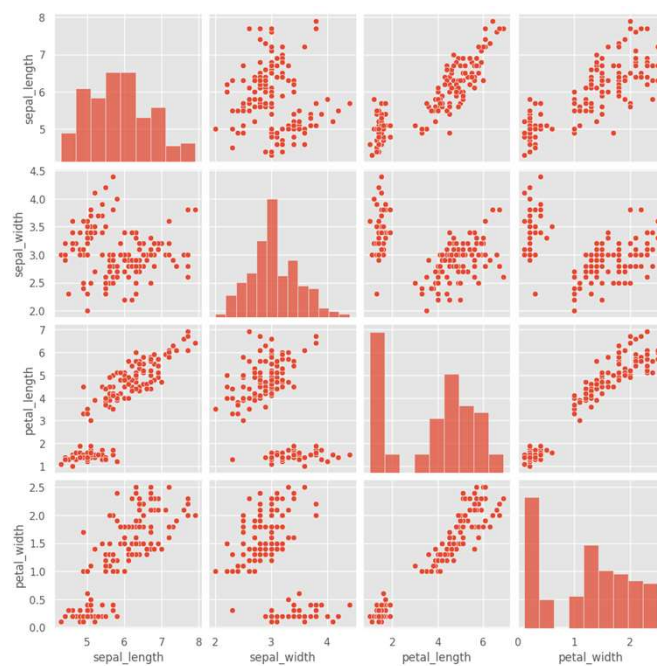
	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

86

2. 散布図と相関の調べ方 - iris dataset -

```
pg=sns.pairplot(iris)
plt.show(pg)
```

87

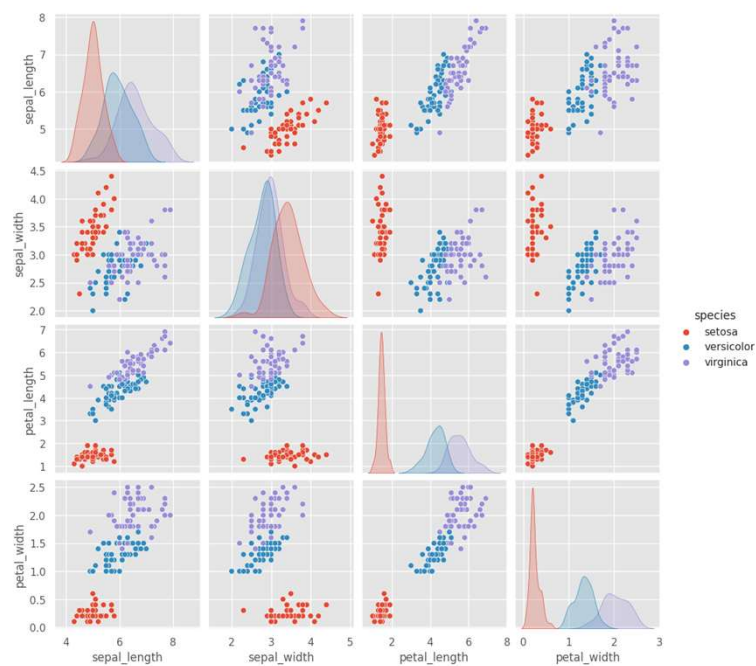


88

2. 散布図と相関の調べ方 - iris dataset -

```
pg=sns.pairplot(iris,hue='species')
plt.show(pg)
```

89

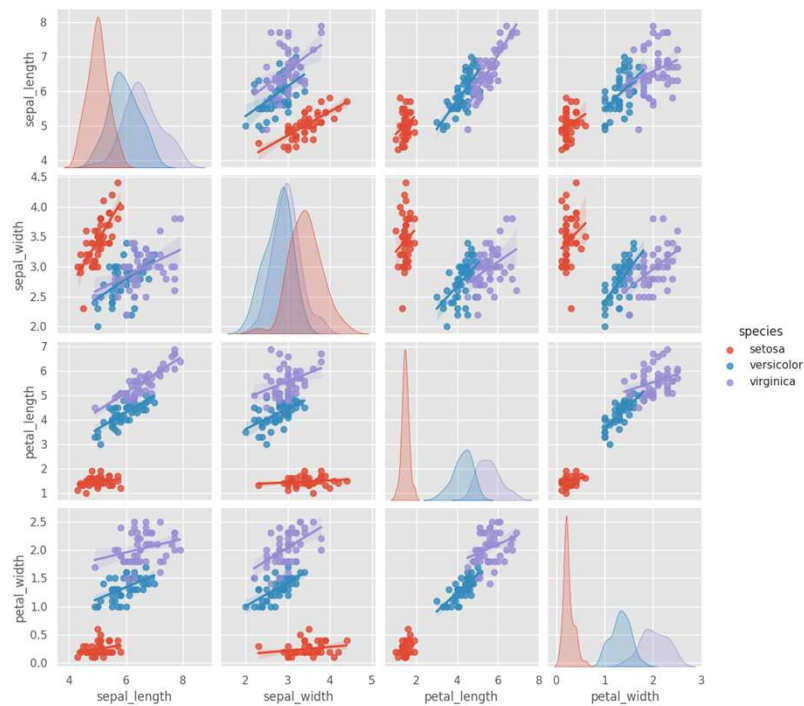


90

2. 散布図と相関の調べ方 - iris dataset -

```
pg=sns.pairplot(iris,hue='species', kind='reg')
plt.show(pg)
```

91

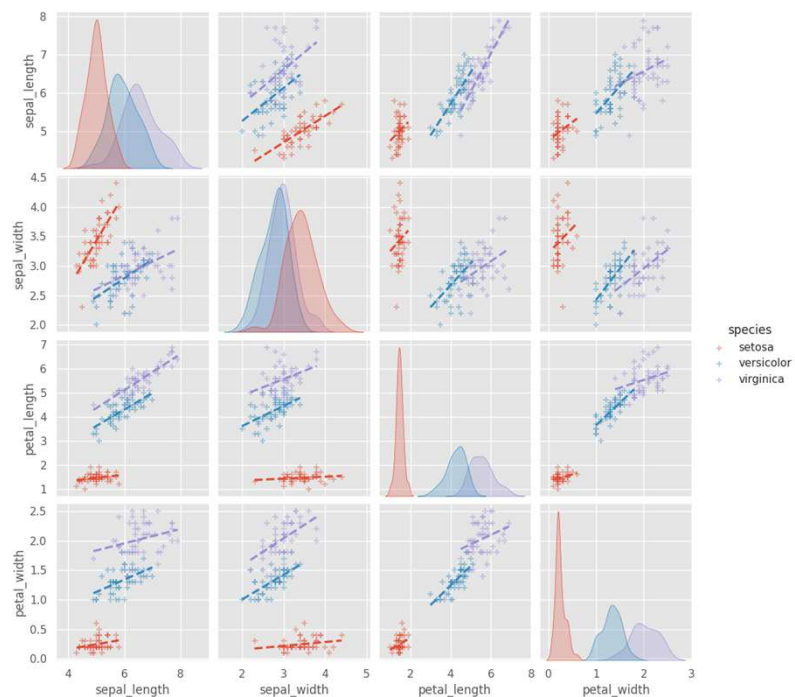


92

2. 散布図と相関の調べ方 - iris dataset -

```
pg=sns.pairplot(iris,hue='species', kind='reg' ,plot_kws={'ci':
None,'marker': '+','scatter_kws': {'alpha': 0.4},'line_kws':
{'linestyle': '--'}})
plt.show(pg)
```

93

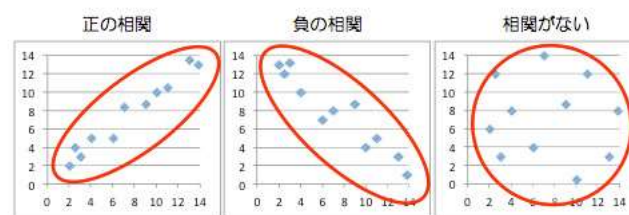


94

3.相関係数と因果関係の違い

相関係数

2つの変量の強弱を数値化したものを「相関係数」という。「類似度」の強さが「-1から1」までの範囲で表現される。正（または負）の相関関係が強いほど1（または-1）に近く、相関関係が弱いほど0に近くなる。



なるほど統計学園 : https://www.stat.go.jp/naruhodo/10_tokucho/hukusu.html

95

3.相関係数と因果関係の違い

因果関係

「原因とそれによって生じる結果との関係」（広辞苑、第6版）を**因果関係**という。要因とアウトカムの間において、関連はみられるが要因が結果を導く関係（真の因果関係）になっていないこともあるため、その判断には注意が必要である。

（一般社団法人日本疫学会 : <https://jeaweb.jp/glossary/glossary015.html>）

相関関係 : AとBの事柄になんらかの関連性があるもの
因果関係 : Aを原因としてBが変動すること

96

4.時系列データの見方、分析方法

時系列データ

時間的な順序をとめないながら観測されるデータのこと。
ある一定の時間間隔で観測されたデータや、イベントが発生した時刻・頻度などが含まれる。

時系列解析：時系列データに潜む傾向や特徴を把握したり、
時系列データの将来の値を予測したりする際に有効な技術

例) 株価データ, 天気予報の気温や降水確率などの気象データ,
人口統計データ, センサーデータ, 販売数データ, など

97

4.時系列データの見方、分析方法

時系列データがもつ情報

- 長期（傾向）変動（トレンド）
時系列の長期的傾向、時間の経過とともに増加・減少する傾向
- 循環変動（サイクル）
傾向変動より短期的で、周期的に繰り返される変動
- 季節変動（シーズナル）
（通常）1年を周期とする規則的な変動
- 不規則変動（ノイズ）
上記の変動で説明できないような、短期的かつ不規則な変動

98

4.時系列データの見方、分析方法

CSVファイルのマウント

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

```
import pandas as pd
import io
```

※Date列を日付データとして解析する

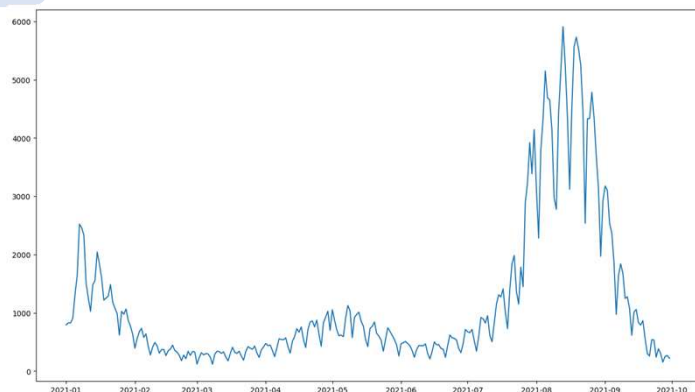
```
df = pd.read_csv(io.BytesIO(uploaded['sample_covid19.csv']),
parse_dates=['Date'])
df
```

99

4.時系列データの見方、分析方法

折れ線グラフで表現

```
import matplotlib.pyplot as plt
plt.plot(df['Date'], df['Tokyo'])
```

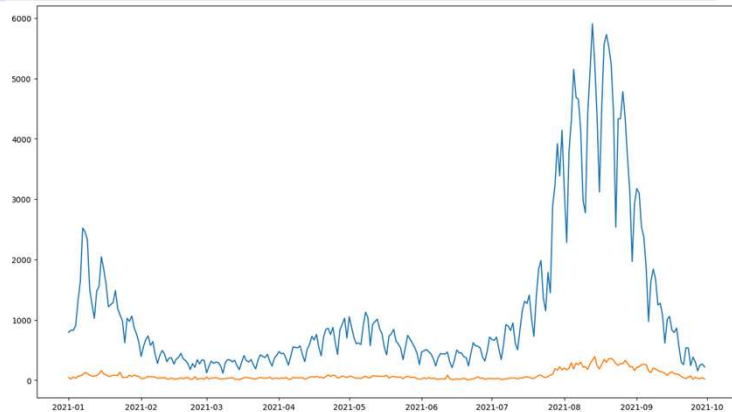


100

4.時系列データの見方、分析方法

折れ線グラフで表現

```
plt.plot(df['Date'], df['Tokyo'])
# 要素を追加すると、棒グラフが重なって表示される
plt.plot(df['Date'], df['Ibaraki'])
```



101

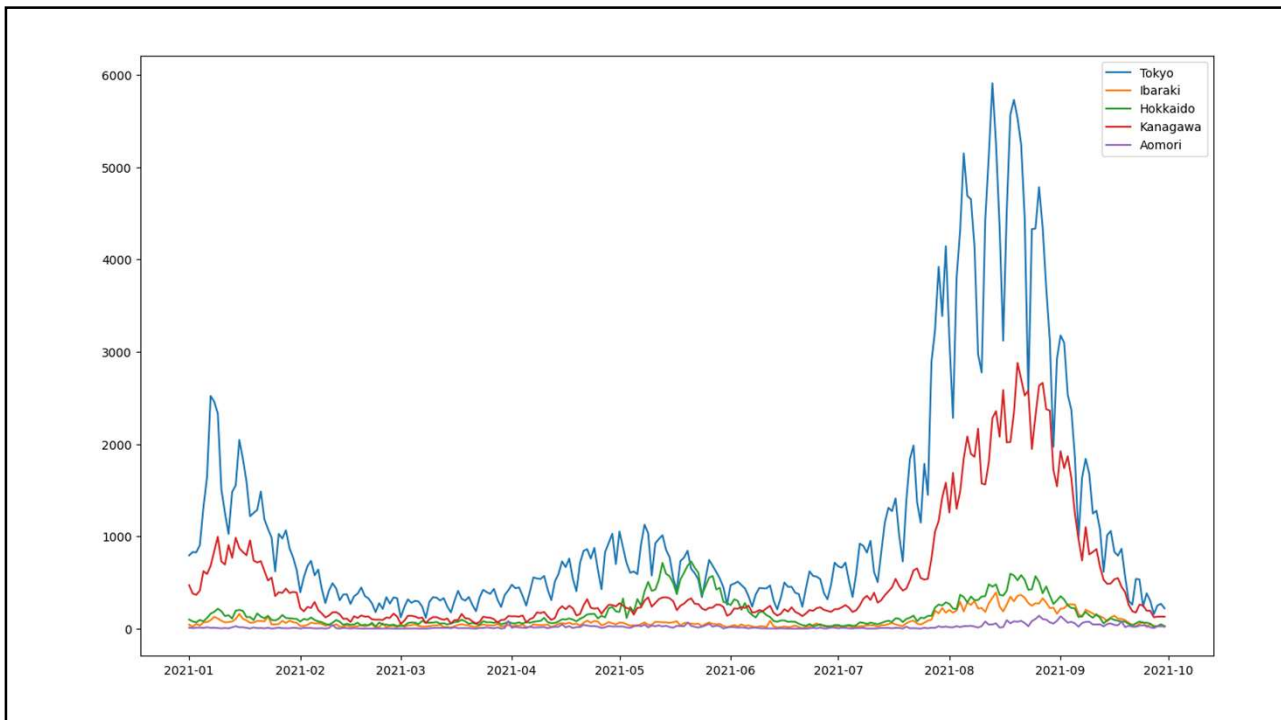
4.時系列データの見方、分析方法

折れ線グラフで表現

```
plt.plot(df['Date'], df['Tokyo'], label='Tokyo')
plt.plot(df['Date'], df['Ibaraki'], label='Ibaraki')
plt.plot(df['Date'], df['Hokkaido'], label='Hokkaido')
plt.plot(df['Date'], df['Kanagawa'], label='Kanagawa')
plt.plot(df['Date'], df['Aomori'], label='Aomori')
plt.legend()
```

※要素にラベルを作成し、凡例に反映させる

102



103

今日の内容

- 1 データ分析に必要な統計学の基礎を学ぶ
- 2 比較して2変数の関係を考える
- 3 データに基づいて判断を下すための手法を学ぶ
- 4 ビジネスにおける予想と分析結果の報告

104

第3章の内容

1. はじめに・推測統計の基本
2. 統計的推定（点推定・区間推定）
3. 統計的仮説検定
 - 基本的な仮説検定 –
 - 2つの標本問題に関する仮説検定 –

105

1.はじめに・推測統計の基本

推測統計とは

母集団から抽出した標本の情報を用いて母集団の情報を推測すること。つまり、観測対象全体の統計的性質を、その観測対象の一部分のみを使って推測する。

母集団（population）：推測したい観測対象全体のこと

標本（sample）：推測に使う観測対象の一部

標本抽出（sampling）：母集団から標本を取り出すこと

推定量（estimator）：推定に用いられる統計量

推定値（estimate）：標本データを用いて計算した結果、推定量の実現値

106

2.統計的推定（点推定・区間推定）

基本的な分布

● 正規分布

- ・ 統計解析において最もよく使われ、重要な推定や検定の理論は全て正規分布を基礎にしていると言っても過言ではない。
- ・ 自然界の多くの現象を表現できる連続分布（確率分布）。
- ・ 18世紀にガウス（Gauss）によって誤差の研究から誘導されたものでガウス分布（Gaussian distribution）とも呼ばれる。

107

2.統計的推定（点推定・区間推定）

基本的な分布

● 確率密度関数（probability density function, pdf）

- ・ 連続型確率変数がある値をとるという事象の確率の密度を表す関数

ある事象が起きる確率が決まっているという性質をもつ変数

→確率変数（random variable）

確率変数がどのような値になるかという法則性を与えるもの

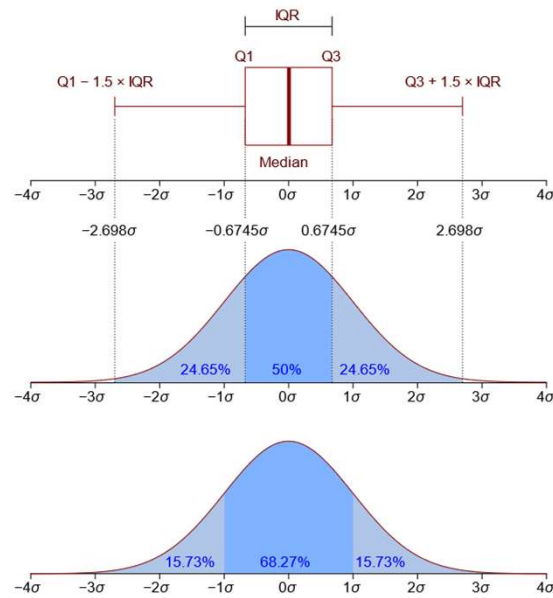
→確率分布（probability distribution）

※母集団のばらつき具合を確率分布としてとらえる

108

2.統計的推定（点推定・区間推定）

基本的な分布



出典：Wikipedia（確率密度関数）

109

2.統計的推定（点推定・区間推定）

基本的な分布（正規分布：確率密度関数）

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm
```

※必要なパッケージ等の読み込み

```
norm.pdf(0)
```

```
0.3989422804014327
```

0が発生する確率

110

2.統計的推定（点推定・区間推定）

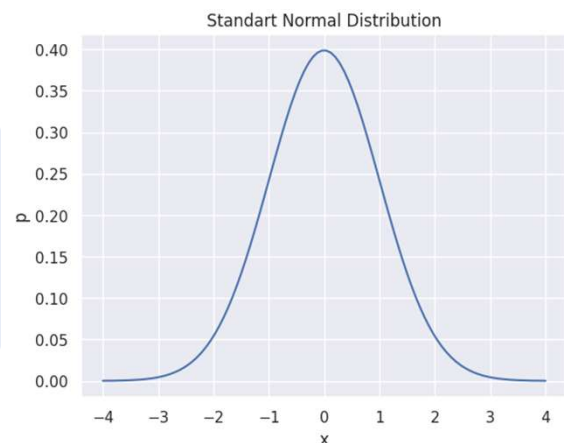
基本的な分布（正規分布：確率密度関数）

```
x = np.linspace(-4, 4, 100)
```

-4から4までの区間で100個の値で近似

```
y_pdf = norm.pdf(x)
```

```
plt.plot(x, y_pdf)
plt.xlabel('x')
plt.ylabel('p')
plt.title('Standart Normal Distribution')
pass
```



111

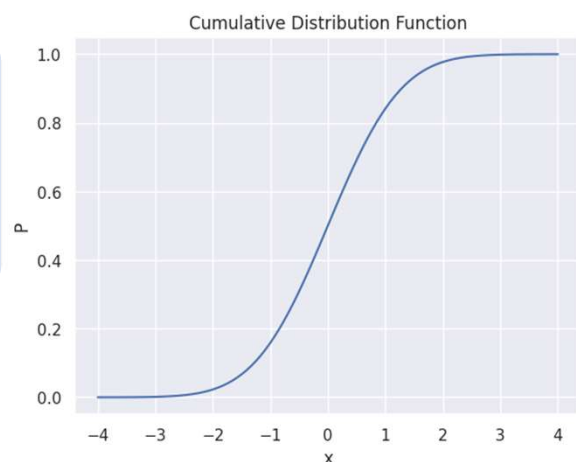
2.統計的推定（点推定・区間推定）

基本的な分布（正規分布：累積分布関数）

```
y_cdf = norm.cdf(x)
```

Cumulative Distribution Function

```
plt.plot(x, y_cdf)
plt.xlabel('x')
plt.ylabel('P')
plt.title('Cumulative Distribution Function')
pass
```



112

2.統計的推定（点推定・区間推定）

基本的な分布（正規分布：累積分布関数）

```
norm.cdf(0)
```

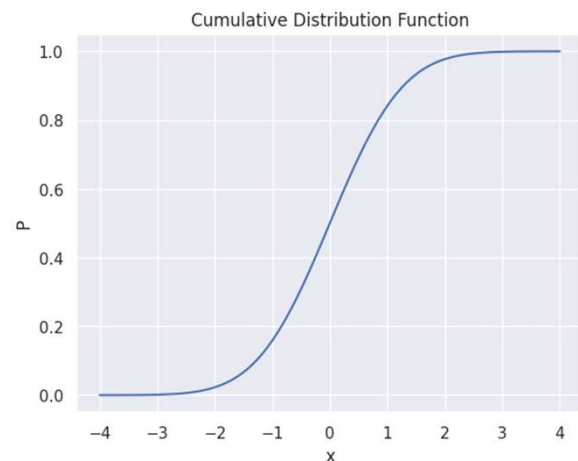
0以下を取る確率
(平均0を中心に左右対称のため)

```
0.5
```

```
norm.cdf(-4)
```

xが-4以下の場合

```
3.167124183311986e-05
```



113

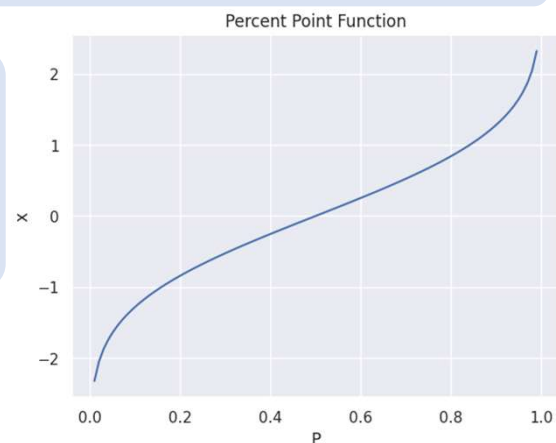
2.統計的推定（点推定・区間推定）

基本的な分布（正規分布：パーセント・ポイント関数）

```
p = np.linspace(0,1,100)
y_ppf = norm.ppf(p)
```

Percent Point Function

```
plt.plot(p,y_ppf)
plt.xlabel('P')
plt.ylabel('x')
plt.title('Percent Point Function')
pass
```



114

2.統計的推定（点推定・区間推定）

基本的な分布（正規分布：パーセント・ポイント関数）

```
norm.ppf(0.5)
```

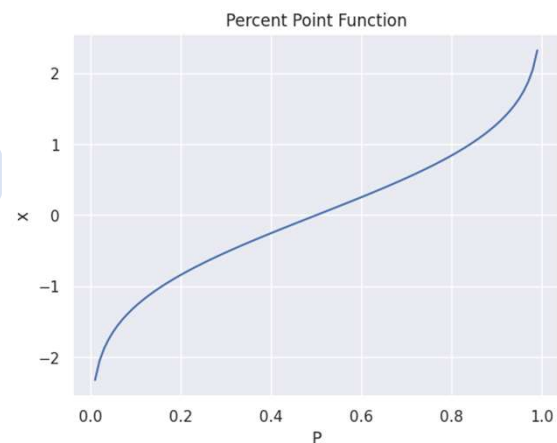
P=0.5の場合

```
0.0
```

```
norm.ppf(0.025)
```

P=0.025の場合

```
-1.9599639845400545
```



115

2.統計的推定（点推定・区間推定）

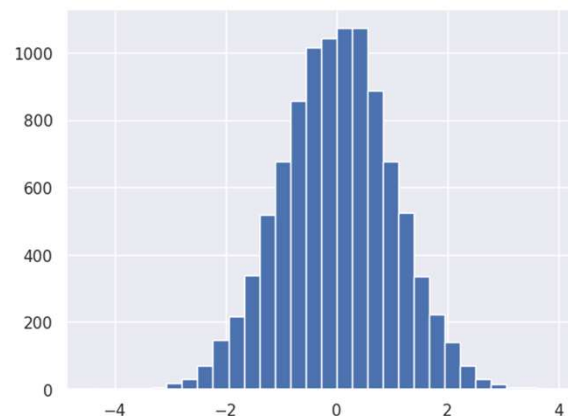
基本的な分布（正規分布：ランダム変数生成関数）

```
y_rvs = norm.rvs(size=10_000)
```

10,000個のランダム変数を生成

```
plt.hist(y_rvs, bins=30)
```

pass bins=表示する棒の数



116

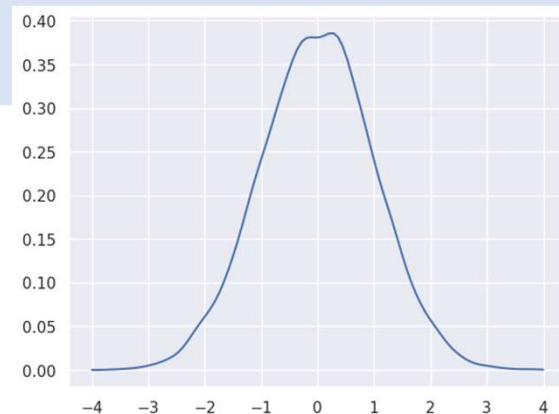
2.統計的推定（点推定・区間推定）

基本的な分布（正規分布：ランダム変数生成関数）

```
from scipy.stats import gaussian_kde
kde = gaussian_kde(y_rvs)
plt.plot(x, kde(x))
pass
```

サブパッケージの読み込みと
y_rvsから確率密度関数を推定

カーネル密度推定：
統計学において、確率変数の確率密度関
数を推定するノンパラメトリック手法の
ひとつ



117

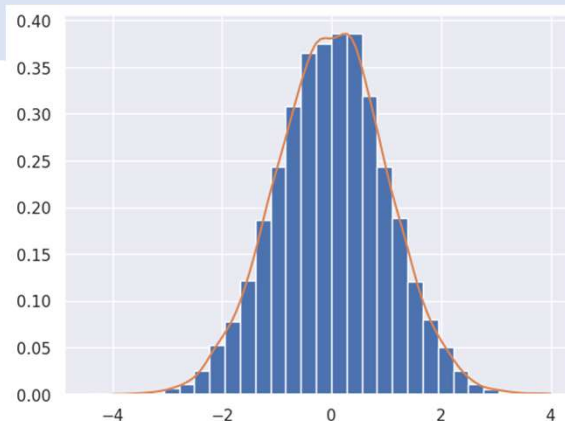
2.統計的推定（点推定・区間推定）

基本的な分布（正規分布：ランダム変数生成関数）

```
plt.hist(y_rvs, bins=30, density=True)
plt.plot(x, kde(x))
pass
```

density=Trueでplt.histとplt.plotの
カーネル密度関数が同じスケールに

推定なので標準正規分布の確率密度と
全く同じにならないが、非常に近い



118

2.統計的推定（点推定・区間推定）

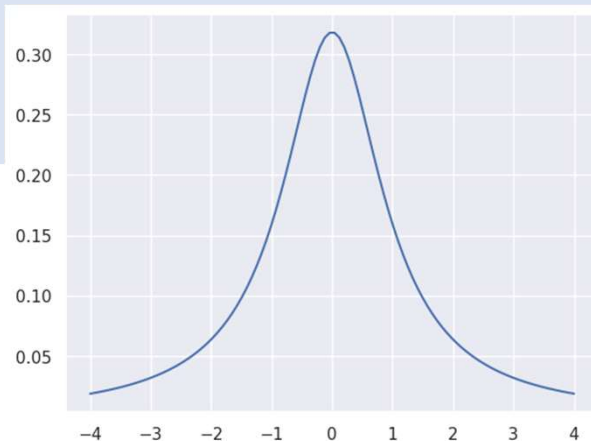
基本的な分布（t 分布）

```
from scipy.stats import t
x = np.linspace(-4, 4, 100)
y = t.pdf(x, df=1)
plt.plot(x,y)
pass
```

母分散が未知の場合の母平均に関する推論

- ・母集団が正規分布に従う
- ・ σ が未知
- ・ $n < 30$

dfn:自由度（degree of freedom）



119

2.統計的推定（点推定・区間推定）

基本的な分布（t 分布）

```
t.cdf(-3, df=1)
```

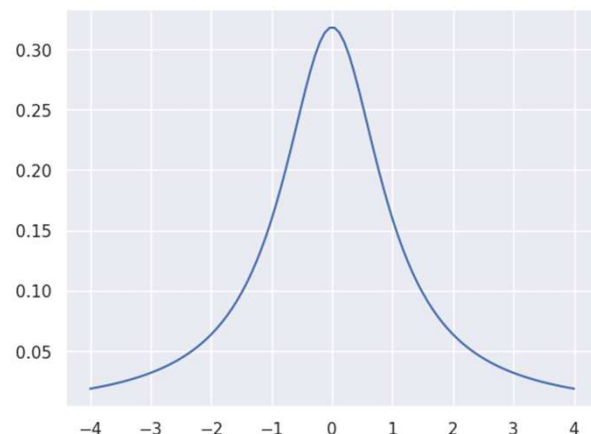
```
0.10241638234956672
```

自由度1の場合にxが-3以下となる確率

```
1-t.cdf(3, df=1)
```

```
0.10241638234956672
```

自由度1の場合にxが3以上となる確率



120

2.統計的推定（点推定・区間推定）

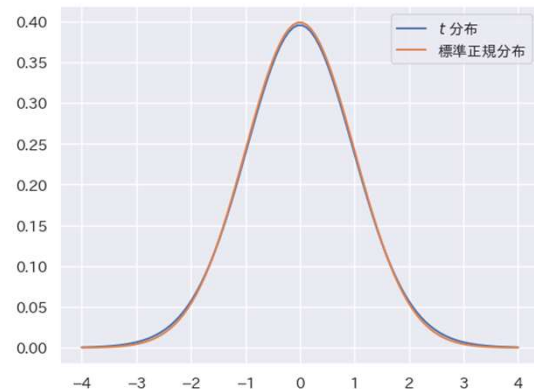
基本的な分布（t分布）

`!pip install japanize-matplotlib` [japanize-matplotlibのインストール](#)

`import japanize_matplotlib` インポート

```
x = np.linspace(-4, 4, 100)
plt.plot(x, t.pdf(x, 30), label=r'$t$ 分布')
plt.plot(x, norm.pdf(x), label='標準正規分布')
plt.legend()
pass
```

自由度30で標準正規分布と重ねてプロット

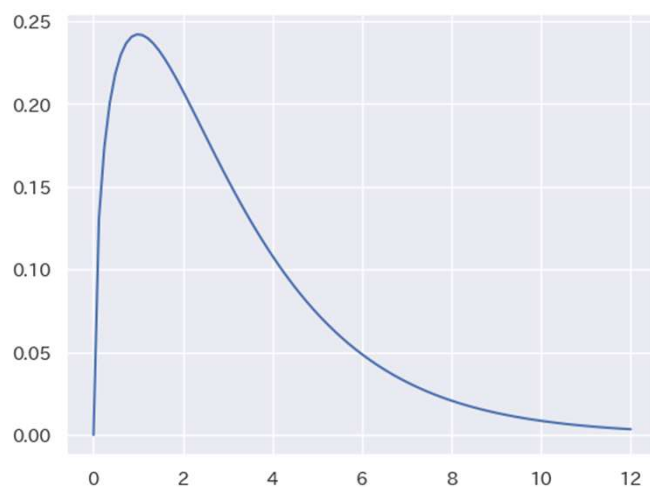


121

2.統計的推定（点推定・区間推定）

基本的な分布（ χ^2 分布）

```
from scipy.stats import chi2
x = np.linspace(0,12,100)
y = chi2.pdf(x, df=3)
plt.plot(x,y)
pass
```



122

2.統計的推定（点推定・区間推定）

基本的な分布（ χ^2 分布）

```
chi2.cdf(1, df=3)
```

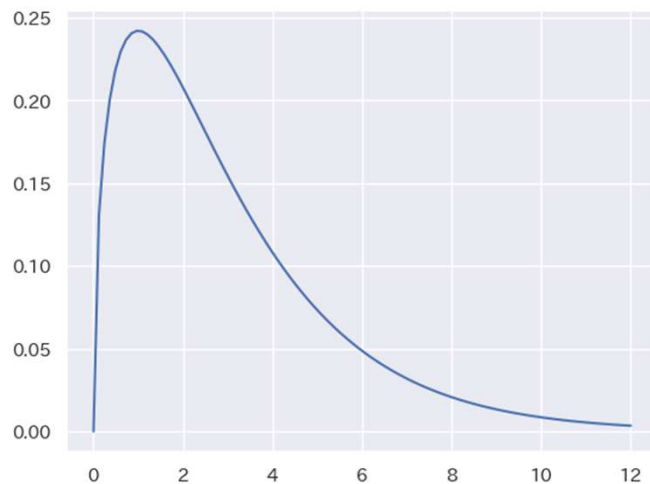
```
0.19874804309879915
```

自由度3の場合にxが1以下の確率

```
1-chi2.cdf(10, df=3)
```

```
0.0185661354630432
```

自由度3の場合にxが10以上となる確率



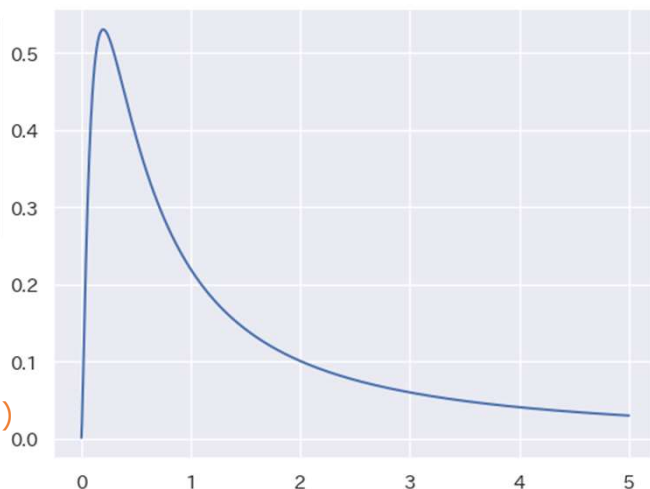
123

2.統計的推定（点推定・区間推定）

基本的な分布（F分布）

```
from scipy.stats import f
x = np.linspace(0.001, 5, 1000)
y = f.pdf(x, dfn=5, dfd=1)
plt.plot(x, y)
pass
```

dfn:分子の自由度
(numerator degree of freedom)
dfd:分母の自由度
(denominator degree of freedom)



124

2.統計的推定（点推定・区間推定）

基本的な分布（ χ^2 分布）

`f.cdf(0.1, dfn=5, dfd=1)`

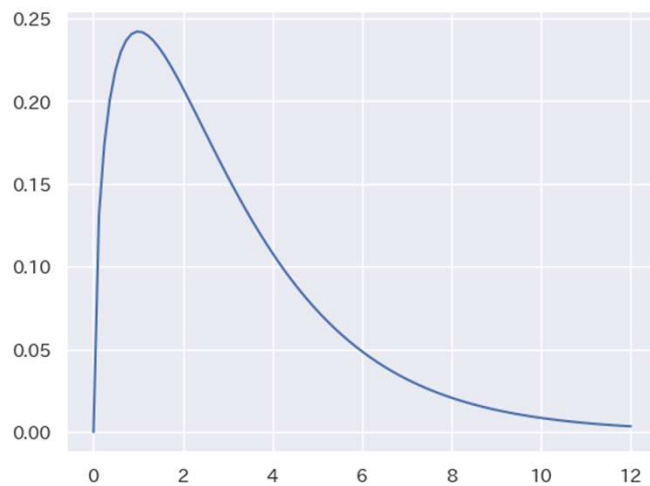
0.02503101581845294

dfn=5, dfd=1の場合にxが0.1以下の確率

`1-f.cdf(5, dfn=5, dfd=1)`

0.32657156446244606

dfn=5, dfd=1の場合にxが5以上の確率



125

2.統計的推定（点推定・区間推定）

点推定

- 母集団分布のパラメータである1つの値として指定する推定方法

区間推定

- 推定値に幅を持たせた推定方法

126

2.統計的推定（点推定・区間推定）

点推定

- 架空のデータのサイズ10, このデータは正規母集団からの無作為標本であると仮定する

```
weight = [8.033, 7.298, 6.223, 7.538, 2.546, 9.251, 5.006, 5.769, 9.628, 6.512]
```

- 母平均を推定する場合は標本平均を, 母分散を推定する場合は不偏分散を, 推定量として扱う

127

2.統計的推定（点推定・区間推定）

点推定

- 母平均を推定する場合は標本平均を, 母分散を推定する場合は不偏分散を, 推定量として扱う

```
x_bar = np.mean(weight)
u2 = np.var(weight, ddof=1)

print('標本平均 : ', round(x_bar, 3))
print('不偏分散 : ', round(u2, 3))
```

```
☞ 標本平均: 6.78
   不偏分散: 4.345
```

128

2.統計的推定（点推定・区間推定）

区間推定（用語の整理）

- 信頼係数
 - ・ 区間推定の幅における信頼の度合いを確率で表現したもの
- 信頼区間
 - ・ ある信頼係数を満たす区間
- 信頼限界
 - ・ 信頼区間の下限値（下側信頼限界）と上限値（上側信頼限界）

129

2.統計的推定（点推定・区間推定）

区間推定（手順）

- 信頼区間95%として、母平均の区間推定
 - ・ 母分散が明らかであれば標準正規分布が利用できるが普通はない
→ t 分布を活用する

130

2.統計的推定（点推定・区間推定）

区間推定（手順）

1. 標本平均 $\bar{X}$ と標準誤差 SE を計算
2. サンプルサイズを n とすると、自由度 $n-1$ の t 分布における2.5%点と97.5%点を計算する
 - ・ t 分布における2.5%点を $t_{0.025}$ と表記
 - ・ t 分布における97.5%点を $t_{0.975}$ と表記
 - ・ t 分布における従う確率変数が $t_{0.025}$ 以上 $t_{0.975}$ 以下になる確率は95%
 - ・ このとき95%が信頼係数となる
3. $\bar{X} - t_{0.975} * SE$ が下側信頼限界となる
4. $\bar{X} - t_{0.025} * SE$ が上側信頼限界となる

131

2.統計的推定（点推定・区間推定）

区間推定

```
n = len(weight)
df = n - 1
u = np.std(weight, ddof=1)
se = u / np.sqrt(n)

print('サンプルサイズ : ', n)
print('自由度          : ', df)
print('標準偏差        : ', round(u, 3))
print('標準誤差        : ', round(se, 3))
print('標本平均        : ', round(x_bar, 3))
```

区間推定に必要なのは
自由度，標本平均，標準誤差
標本平均は計算済み

📄	サンプルサイズ : 10
	自由度 : 9
	標準偏差 : 2.085
	標準誤差 : 0.659
	標本平均 : 6.78

132

2.統計的推定（点推定・区間推定）

区間推定

```
t_025 = stats.t.ppf(q=0.025, df=df)
t_975 = stats.t.ppf(q=0.975, df=df)

print('t分布の2.5%点 : ', round(t_025, 3))
print('t分布の97.5%点 : ', round(t_975, 3))
```

自由度n-1の t 分布における
2.5%点と97.5%点を計算

⇒ t分布の2.5%点 : -2.262
t分布の97.5%点 : 2.262

133

2.統計的推定（点推定・区間推定）

区間推定

```
lower_mu = x_bar - t_975 * se
upper_mu = x_bar - t_025 * se

print('下限信頼区間 : ', round(lower_mu, 3))
print('上限信頼区間 : ', round(upper_mu, 3))
```

t 分布は左右対称なので,
 $t_{0.025} - t_{0.975}$ で信頼区間を計算

⇒ 下限信頼区間: 5.289
上限信頼区間: 8.272

母平均の95%信頼区間は, 5.289から8.272となった

134

2.統計的推定（点推定・区間推定）おまけ問題

区間推定

1. 標本 $n = 10$

405g, 395g, 374g, 410g, 417g, 426g, 383g, 398g, 390g, 402g

母標準偏差：15g

信頼水準95%で信頼区間を推定せよ

2. 工場で生産している製品Aがある

以下のデータがわかっている

標本 $n = 100$, 標本平均=150g, 母分散 = 15^2 g

母平均 μ を信頼水準95%で信頼区間を推定せよ

135

3.統計的仮説検定

統計的仮説検定

- データを使って何かを判断したいときに使われる手法
- さまざまな種類があり判断する対象も手法によってさまざま、単に検定と呼ぶ場合もある

- ・ 統計的推定 → 母集団分布のパラメータを言い当てる試み
- ・ 統計的検定 → 母集団のパラメータについて判断を下す
例) 「母平均が50か、あるいは50ではないか」を判断

136

3.統計的仮説検定 - 基本的な仮説検定 -

t検定（母平均に関する1標本のt検定）

- データは正規母集団からの無作為標本，と仮定
- 平均値が「ある値」と異なると言えるかどうか，を判断

例

内容量が135gと書かれたスナック菓子，測ってみると134gの場合もあれば，136gの場合もある。工場の機械に問題がないか検査する必要がある。

「スナック菓子の内容量の母平均が135gと異なっていると言えるかどうか」という判断をサポート。

137

3.統計的仮説検定 - 基本的な仮説検定 -

仮説検定の流れ

1. 仮説を立てる
2. 有意水準を決める
3. 検定統計量を計算
4. p値を計算
5. p値より有意水準が大きいか
 - （Yes）帰無仮説を採択
 - （No）帰無仮説を棄却

138

3.統計的仮説検定 - 基本的な仮説検定 -

仮説検定の流れ

1. 仮説を立てる
2. 有意水準を決める
3. 検定統計量を計算
4. p値を計算
5. p値より有意水準が大きいのか
 - (Yes) 帰無仮説を採択
 - (No) 帰無仮説を棄却

139

3.統計的仮説検定 - 基本的な仮説検定 -

1. 仮説を立てる

- 帰無仮説 H_0 ：スナック菓子の母平均は135gである
- 対立仮説 H_1 ：スナック菓子の母平均は135gと異なる
- 帰無仮説が棄却されたならば、有意差あり、つまり「スナック菓子の母平均は135gと異なる」と判断する
(135gより大きい or 少ない, ではない)

140

3.統計的仮説検定 - 基本的な仮説検定 -

仮説検定の流れ

1. 仮説を立てる
2. 有意水準を決める
3. 検定統計量を計算
4. p値を計算
5. p値より有意水準が大きいのか
 - (Yes) 帰無仮説を採択
 - (No) 帰無仮説を棄却

141

3.統計的仮説検定 - 基本的な仮説検定 -

有意水準を決める

- 有意である：偶然ではなく、何か意味があるということ
- 帰無仮説が間違っていると判断（棄却）する区間のことを棄却域（rejection region）、採択される区間を採択域（acceptance region）といい、この境の基準となる確率のこと
- つまり、帰無仮説を棄却する基準
- 伝統的に α と表記し、5%や1%が使われることが多い

142

3.統計的仮説検定 - 基本的な仮説検定 -

有意水準を決める（危険率）

- 第一種の過誤：帰無仮説が正しいのに誤って帰無仮説を棄却してしまうこと
- 第二種の過誤：帰無仮説が間違っているのに、誤って帰無仮説を採択してしまうこと
- 有意水準は第一の過誤を許容できる確率

143

3.統計的仮説検定 - 基本的な仮説検定 -

仮説検定の流れ

1. 仮説を立てる
2. 有意水準を決める
3. 検定統計量を計算
4. p値を計算
5. p値より有意水準が大きいか
 - （Yes）帰無仮説を採択
 - （No）帰無仮説を棄却

144

3.統計的仮説検定 - 基本的な仮説検定 -

検定統計量を計算（準備）

```
import numpy as np
import pandas as pd
from scipy import stats
```

```
food = [111, 124, 125, 126, 127, 134, 135, 136, 139, 141]
```

- 帰無仮説 H_0 ：スナック菓子の母平均は135gである
- 対立仮説 H_1 ：スナック菓子の母平均は135gと異なる
- 有意水準は5%

145

3.統計的仮説検定 - 基本的な仮説検定 -

標本平均

```
x_bar = np.mean(food)
round(x_bar, 3)
```

 129.8

自由度

```
n = len(food)
df = n - 1
df
```

 9

146

3.統計的仮説検定 - 基本的な仮説検定 -

標準誤差（標準偏差 / サンプルサイズの平方根）

```
u = np.std(food, ddof=1)
se = u / np.sqrt(n)
round(se, 3)
```

2.839

t値の計算

```
t_sample = (x_bar - 135) / se
round(t_sample, 3)
```

-1.831

147

3.統計的仮説検定 - 基本的な仮説検定 -

棄却域の計算

```
round(stats.t.ppf(q=0.025, df=df), 3)
```

-2.262

```
round(stats.t.ppf(q=0.975, df=df), 3)
```

2.262

$-t_{0.025} < |t_{\text{sample}}| < t_{0.975}$ の範囲が採択域となる,
よって帰無仮説を棄却することはできない

148

3.統計的仮説検定 - 基本的な仮説検定 -

仮説検定の流れ

1. 仮説を立てる
2. 有意水準を決める
3. 検定統計量を計算
4. p値を計算
5. p値より有意水準が大きいのか
 - (Yes) 帰無仮説を採択
 - (No) 帰無仮説を棄却

149

3.統計的仮説検定 - 基本的な仮説検定 -

p値の計算

```
p_value = stats.t.cdf(-np.abs(t_sample), df=df) * 2
round(p_value, 3)
```

p値が有意水準0.05を上回っているので、帰無仮説を支持する確率が高い

スナック菓子の平均重量は135gと有意に異なっていないと判断することができる

150

3.統計的仮説検定 - 基本的な仮説検定 -

stats.ttest_1samp関数（1標本のt検定）

```
stats.ttest_1samp(food, 135)
```

```
TtestResult(statistic=-1.831369433567421, pvalue=0.10027730072021666, df=9)
```

statisticがt値, pvalueがp値

151

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

2群のデータに対するt検定

- 2つの変数の間で平均値に差があるかどうか, を判断

例

あるボディビルにより体重の増減がおこるかどうか調べるために, 10人についてその効果を調べる場合など,
「同じ対象を異なった条件で2回測定して, その違いをみる」
とした場合

152

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

2群のデータに対するt検定

	正規分布を仮定できる	正規分布を仮定できない
対応あり	対応のある t 検定※1	ウィルコクソンの符号付き順位検定
対応なし	対応のない t 検定※2	マン・ホイットニーのU検定

※1 「データの差」を取ってから母平均に関する1標本のt検定

※2 「平均値の差」に注目する

153

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

1. 仮説を立てる

- ・ 帰無仮説 H_0 : ボディビルにより体重の増減は起こっていない
- ・ 対立仮説 H_1 : ボディビルにより体重の増減は起こっている

2. 有意水準を決める

- ・ 有意水準5%とし, p値が0.05を下回れば, 帰無仮説が棄却され, ボディビルでの体重増減への有意な変化が認められると主張できる

154

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

3. 検定統計量を計算する

No.	1	2	3	4	5	6	7	8	9	10	平均
前	57.6	88.5	73.5	77.1	64.9	93.0	76.2	79.4	89.4	61.7	
後	61.2	90.7	73.4	82.6	66.7	93.7	78.0	84.4	88.0	64.0	
後-前	3.6	2.2	-0.1	5.5	1.8	0.7	1.8	5	-1.4	2.3	2.14

155

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

3. 検定統計量を計算する（ライブラリの読み込み等）

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

```
import numpy as np
import pandas as pd
from scipy import stats
```

```
bodybuilding = pd.read_csv('sample_t-
test.csv')
print(bodybuilding)
```

```
person bodybuilding weight
0 1 before 57.6
1 2 before 88.5
2 3 before 73.5
3 4 before 77.1
4 5 before 64.9
5 6 before 93.0
6 7 before 76.2
7 8 before 79.4
8 9 before 89.4
9 10 before 61.7
10 1 after 61.2
11 2 after 90.7
12 3 after 73.4
13 4 after 82.6
14 5 after 66.7
15 6 after 93.7
16 7 after 78.0
17 8 after 84.4
18 9 after 88.0
19 10 after 64.0
```

156

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

3. 検定統計量を計算する

```
before = bodybuilding.query('bodybuilding == "before"')['weight']
after = bodybuilding.query('bodybuilding == "after"')['weight']
```

ボディビル前と
後の標本平均

```
before = np.array(before)
after = np.array(after)
```

アレイに変換

```
diff = after - before
diff
```

差を計算

```
array([ 3.6, 2.2, -0.1, 5.5, 1.8, 0.7, 1.8, 5., -1.4, 2.3])
```

157

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

対応のない t 検定（不等分散）

```
#平均値
x_bar_bef = np.mean(before)
x_bar_aft = np.mean(after)
```

```
#分散
u2_bef = np.var(before, ddof=1)
u2_aft = np.var(after, ddof=1)
```

```
#サンプルサイズ
m = len(before)
n = len(after)
```

```
#t値
t_value = (x_bar_aft - x_bar_bef) / np.sqrt((u2_bef/m + u2_aft/n))
round(t_value, 3)
```

```
0.406
```

158

3.統計的仮説検定 - 2つの標本問題に関する仮説検定 -

対応のない t 検定 (不等分散)

```
df = (u2_bef / m + u2_aft / n)**2 / ((u2_bef / m)**2 / (m-1) + (u2_aft / n)**2 / (n-1))
round(df, 3)
```

```
17.963
```

```
p_value = stats.t.cdf(-np.abs(t_value), df=df)*2
round(p_value,5)
```

```
0.68988
```

```
stats.ttest_ind(after, before, equal_var=False)
```

```
TtestResult(statistic=0.4055321209937369, pvalue=0.6898758363640115, df=17.963028980728833)
```

159

今日の内容

- 1 データ分析に必要な統計学の基礎を学ぶ
- 2 比較して2変数の関係を考える
- 3 データに基づいて判断を下すための手法を学ぶ
- 4 ビジネスにおける予想と分析結果の報告

160

第4章の内容

1. 回帰分析による予想

2. 決定木による予想

3. クラスター分析

- その1 –
- その2 –

161

1.回帰分析による予想

回帰分析とは

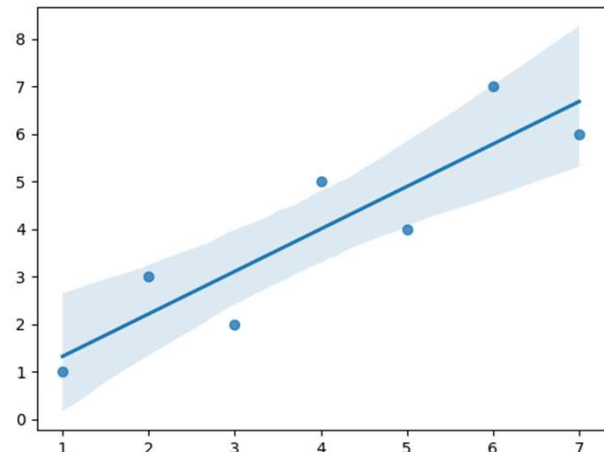
- 因果関係が疑われる複数の変数を使って、ある変数から他の変数の値を予測する手法
- 原因となる変数のことを説明変数、結果となる変数のことを応答変数という
- 説明変数と応答変数はそれぞれ独立変数と従属変数という
- 説明変数が数量データであるモデルを用いた分析手法
(説明変数が複数あるものを重回帰分析と呼ぶ)

162

1. 回帰分析による予想

回帰分析

```
import seaborn as sns
x = [1, 2, 3, 4, 5, 6, 7]
y = [1, 3, 2, 5, 4, 7, 6]
sns.regplot(x=x, y=y) 散布図
```



163

1. 回帰分析による予想

回帰分析

!pip install pingouin Pingouin (ペンギン) というパッケージをインストール

```
import pingouin as pg
pg.corr(x, y)
```

		R:相関係数		p値		
	n	r	CI95%	p-val	BF10	power
pearson	7	0.892857	[0.43, 0.98]	0.006807	8.961	0.854032

95%信頼区間

```
pg.linear_regression(x, y)
```

	names	coef	se	T	pval	r2	adj_r2	CI[2.5%]	CI[97.5%]
0	Intercept	0.428571	0.900680	0.475831	0.654258	0.797194	0.756633	-1.886700	2.743843
1	x1	0.892857	0.201398	4.433293	0.006807	0.797194	0.756633	0.375147	1.410568

傾き p値 R² (相関係数の二乗) 95%信頼区間

164

1. 回帰分析による予想

回帰分析と結果の可視化

	A	B
1	temperature	sales
2	4.7	507
3	5.4	416
4	11.5	607
5	17	746
6	19.8	894
7	22.4	1021
8	28.3	1506
9	28.1	1443
10	22.9	861
11	19.1	640
12	14	492
13	8.3	537
14	5.6	494
15	7.2	423
16	10.6	542
17	13.6	667
18	20	1000
19	21.8	991
20	24.1	1236
21	28.4	1513
22	25.1	996
23	19.4	724
24	13.1	531
25	8.5	584
26	7.1	510
27	8.3	482
28	10.7	610

既知のデータ
sales_data.csv

	A	B
1	temperature	
2	12.6	
3	18.7	
4	10.2	
5	15.1	
6	20.5	
7		
8		
9		
10		
11		
12		
13		
14		
15		
16		
17		
18		
19		
20		
21		
22		
23		
24		
25		
26		
27		
28		

予測対象データ
sales_future.csv

一般社団法人 日本アイスクリーム協会

(<https://www.icecream.or.jp/about/>) と

国土交通省気象庁

(https://www.data.jma.go.jp/obd/stats/etrn/view/monthly_s3.php?prec_no=44&block_no=47662) のデータを合わせて作成した

sales_data.csvに基づいて回帰分析を行い、
sales_future.csvに対して売上予測

165

1. 回帰分析による予想

回帰分析と結果の可視化

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

ファイルの読み込み等

```
import pandas as pd
```

```
df = pd.read_csv('sample_sales_data.csv')
df_test = pd.read_csv('sample_sales_future.csv')
```

166

1.回帰分析による予想

回帰分析と結果の可視化

```
x_name = "temperature"
y_name = "sales"
x = df[x_name]
y = df[y_name]

import statsmodels.api as sm
model = sm.OLS(y, sm.add_constant(x))
result = model.fit(dis=0)
print(result.summary())
```

求めたいもの（目的変数）が「y」で、その計算に使うもの（説明変数）が「x」
「add_constant」は切片を使う場合に指定
つまり、 $y=ax + b$ のbの項を使う場合に付ける

167

1.回帰分析による予想

回帰分析と結果の可視化

OLS Regression Results

Dep. Variable:

sales

R-squared:

0.832

Model:

OLS

Adj. R-squared:

0.829

Method:

Least Squares

F-statistic:

287.1

Date:

Fri, 17 Nov 2023

Prob (F-statistic):

3.96e-24

Time:

08:27:53

Log-Likelihood:

-380.86

No. Observations:

60

AIC:

765.7

Df Residuals:

58

BIC:

769.9

Df Model:

1

Covariance Type:

nonrobust

	coef	std err	t	P> t	[0.025	0.975]
const	155.0243	44.387	3.493	0.001	66.175	243.874
temperature	41.4521	2.447	16.943	0.000	36.555	46.349

Omnibus:

0.628

Durbin-Watson:

1.324

Prob(Omnibus):

0.731

Jarque-Bera (JB):

0.706

Skew:

0.048

Prob(JB):

0.703

Kurtosis:

2.477

Cond. No.

44.5

Notes:

[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

coefとconstが変わった所が切片
「155.0243」

R-squaredがR²値
「0.832」

coefとtemperatureが変わった所
が傾き「41.4521」

アイスクリームの売上をy,
気温をx,
 $y = 41.45x + 155.02$
という関係があるっぽい

168

1. 回帰分析による予想

回帰分析と結果の可視化（未知のデータの推測）

```
y_result=[]
for i in range(len(df_test.index)): Xの傾きが格納
    y_tmp = result.params.const + result.params[x_name] *
    df_test[x_name][i]
    y_result.append(y_tmp) 切片が格納
print(y_result)
```

```
[677.3210908376445, 930.1790552581247, 577.8359900820456, 780.9514041247264, 1004.7928808248239]
```

169

1. 回帰分析による予想

回帰分析と結果の可視化（未知のデータの推測）

```
df_test_y = pd.DataFrame(y_result,columns=["y"])
df_result = pd.concat([df_test,df_test_y],axis=1)
df_result.to_csv("result.csv")
df_result
```

	temperature	y
0	12.6	677.321091
1	18.7	930.179055
2	10.2	577.835990
3	15.1	780.951404
4	20.5	1004.792881

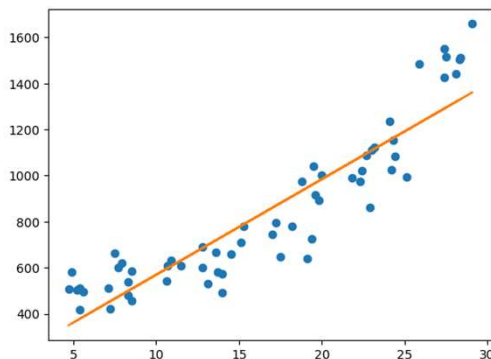
読み込んだファイル（sales_future.csv）に計算結果（y列）を追記し、csvファイルとして出力することも可能

170

1. 回帰分析による予想

回帰分析と結果の可視化（プロット）

```
import matplotlib.pyplot as plt
plt.plot(x, y, 'o')
plt.plot(x, result.params.const+result.params[x_name]*x)
plt.show()
```



そこそこ良い感じの分析精度

171

2. 決定木による予想

決定木とは（Decision Tree Analysis:DCA）

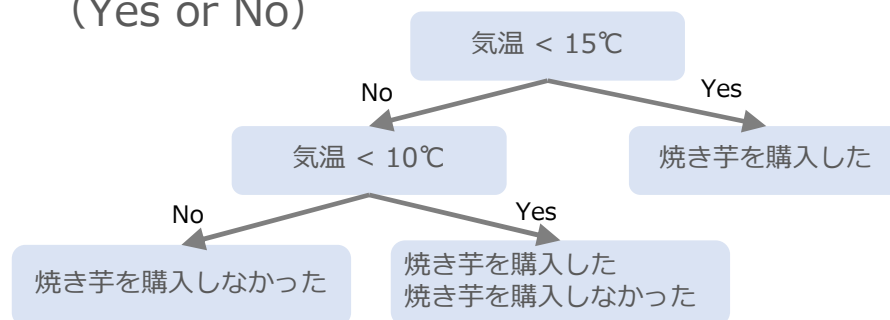
- データから分類・判別のために作られる決定木と呼ばれる樹形図を作成し、予測や検証をする分析（機械学習の1つ）
- 購買情報やアンケート結果等のさまざまなデータに対して実施することが可能
- 目的変数の予測や目的変数に影響している因子の検証等に活用

172

2.決定木による予想

決定木分析と回帰分析の違い

- どちらも目的変数を予想するモデル、ただプロセスが違う
- 計算式などを使わずにシンプルな分岐のみで予想
(Yes or No)



173

2.決定木による予想

流れ

- 事前準備
- 決定木の前処理
 - 説明変数 (x) と目的変数 (y) に分割
 - ダミー変数処理 (文字列→数値)
 - 学習用-テスト用に分割
- 決定木モデルの作成と予想
 - パラメータ: max_depth (最大深度)
- 決定木の可視化 (plot_tree)
 - ツリーの見方
 - パラメータ: 説明変数の名前 (feature_names)
 - パラメータ: 目的変数の名前 (class_names)
 - パラメータ: 色 (filled)

174

2. 決定木による予想

事前準備（タイタニック号データ）

```
import seaborn as sns
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import tree
df = sns.load_dataset('titanic')
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

175

2. 決定木による予想

事前準備（タイタニック号データ）

survived	生存フラグ
pclass	チケットクラス
sex	性別
age	年齢
sibsp	兄弟・配偶者の数
parch	親・子の数
fare	料金

embarked	出航地
class	チケットクラス
who	性別
adult_male	成人男性かどうか
deck	乗船していたデッキ
embark_town	出航地名
alive	生存
alone	一人だったか

176

2.決定木による予想

決定木の前処理：説明変数（x）と目的変数（y）に分割

```
df_x = df[['sex','pclass','fare']]
df_y = df['survived']
```

事前に定義したタイタニック号のデータの変数（df）から，説明変数（df_x）と目的変数（df_y）に分ける

決定木の可視化を見やすくするために，性別・チケットクラス・運賃に限定

177

2.決定木による予想

決定木の前処理：ダミー変数処理（文字列→数値）

```
df_x = pd.get_dummies(df_x, drop_first=True)
```

↖ drop_first カラム数を減らすパラメータ

性別のカラムに「male / female」の文字列が格納されているので，数値列に変換
get_dummies

	pclass	fare	sex_male
0	3	7.2500	1
1	1	71.2833	0
2	3	7.9250	0
3	1	53.1000	0
4	3	8.0500	1

178

2.決定木による予想

決定木の前処理：学習用-テスト用に分割

```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y =
train_test_split(df_x,df_y,random_state=1)
```

学習用とテスト用にデータを分割
train_test_split

説明変数 (df_x)			目的変数 (df_y)	学習データ (train)
sex_male	pclass	fare	survived	
1	3	7.2500	0	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	テストデータ (test)
0	1	55.008	0	

179

2.決定木による予想

決定木の前処理：学習用-テスト用に分割

```
from sklearn.model_selection import train_test_split
train_x, test_x, train_y, test_y =
train_test_split(df_x,df_y,random_state=1)
```

学習用とテスト用にデータを分割
train_test_split

説明変数 (df_x)			目的変数 (df_y)	学習データ (train)
sex_male	pclass	fare	survived	
1	3	7.2500	0	
⋮	⋮	⋮	⋮	
⋮	⋮	⋮	⋮	テストデータ (test)
0	1	55.008	0	

180

2. 決定木による予想

決定木モデルの作成と予測

```
from sklearn import tree
model = tree.DecisionTreeClassifier(max_depth=2, random_state=1)
```

決定木のモデルを作成

scikit-learnというpython用機械学習ライブラリの中から決定木モデルのtreeを拝借

sklearn.tree.DecisionTreeClassifierというクラスに決定木が実装されている

max_depth : 分類枝の最大深さ

random_state : 学習時の乱数シード, 常に同じ結果を得たい場合は適当な整数を指定

181

2. 決定木による予想

決定木モデルの作成と予測

```
model.fit(train_x, train_y)    fitメソッドで学習を行う
```

fit(x, y) : 特徴量 x, クラス y を教師データとして学習する

182

2. 決定木による予想

決定木モデルの作成と予測

`model.predict(test_x)` モデルに対してpredictfitで予想を実施

`predict (x)` : 特徴量 `x` に対するクラスの予想結果を返す

```
array([1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0,
       0, 0, 0])
```

183

2. 決定木による予想

決定木モデルの作成と予測

`model.score(test_x, test_y)` `score`メソッドでスコア（正解率）を算出

`score (x, y)` : 決定係数を出力, 予想値 `x`と正解値 `y`の相関を測る

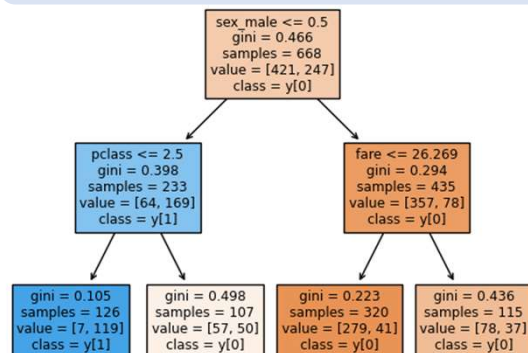
```
0.7533632286995515
```

184

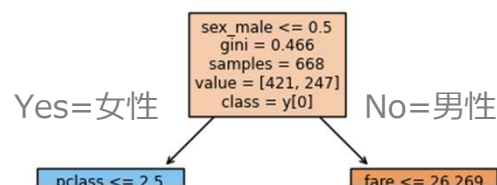
2. 決定木による予想

決定木の可視化 (plot_tree)

```
from sklearn.tree import plot_tree
plot_tree(model, feature_names=train_x.columns, class_names=True,
          filled=True)
```



性別（男性=1）が0.5より小さい



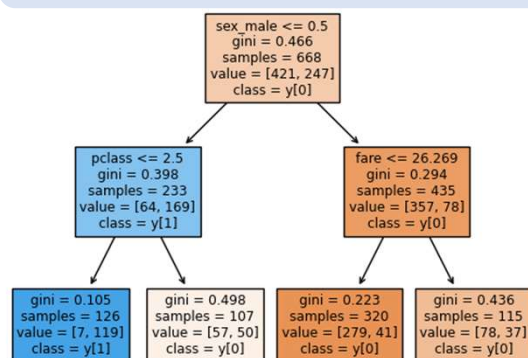
185

2. 決定木による予想

決定木の可視化 (plot_tree)

```
from sklearn.tree import plot_tree
plot_tree(model, feature_names=train_x.columns, class_names=True,
          filled=True)
```

class_name : 目的変数の名前
filled : 色



gini（ジニ係数）：不純度
各葉（ノード）にどれくらい間違いが含まれているのか
不純度が低い（正しい）と値は「0」
不純度が高い（間違い）と「1」に近づく

色合い：濃いほど目的変数に近い
（ジニ係数も低い）
sample：ノードの個数
value：その条件に当てはまる数

186

3. クラスター分析

クラスター分析とは（クラスタリング）

- 個々のデータから似ているデータ同士をグルーピングする分析手法
- クラスタ（群）ごとに分けることで、データの中身を理解しやすくしたり、群ごとに施策を行うことができる

187

3. クラスター分析

流れ

- 事前準備
- k-meansの前処理
 - データを限定する
 - 欠損値を処理する
 - ダミー変数処理（文字列→数値）
 - データを標準化する
- k-meansのクラスタリングを実行
- クラスタリングの結果を確認
 - 主成分分析でグラフ化

188

3. クラスター分析

事前準備 (タイタニック号データ)

```
import seaborn as sns
import pandas as pd
df = sns.load_dataset('titanic')
df.head()
```

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male	deck	embark_town	alive	alone
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True	NaN	Southampton	no	False
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False	C	Cherbourg	yes	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False	NaN	Southampton	yes	True
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False	C	Southampton	yes	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True	NaN	Southampton	no	True

189

3. クラスター分析

事前準備 (タイタニック号データ)

survived	生存フラグ
pclass	チケットクラス
sex	性別
age	年齢
sibsp	兄弟・配偶者の数
parch	親・子の数
fare	料金

embarked	出航地
class	チケットクラス
who	性別
adult_male	成人男性かどうか
deck	乗船していたデッキ
embark_town	出航地名
alive	生存
alone	一人だったか

190

3. クラスター分析

k-meansの前処理：データを限定する

```
columns = ['survived', 'pclass', 'sex', 'age', 'fare']
df = df[columns]
```

生存・チケットクラス・性別・年齢・運賃に絞込む

191

3. クラスター分析

k-meansの前処理：欠損値を処理する

```
df.isnull().sum() isnull と sum の組み合わせ
```

年齢に177件の欠損を確認，これを今回は平均で補う

```
survived    0
pclass      0
sex         0
age        177
fare        0
dtype: int64
```

```
mean = df['age'].mean()
df['age'] = df['age'].fillna(mean) fillna を使う
```

192

3. クラスター分析

k-meansの前処理 : ダミー変数処理 (文字列→数値)

```
df = pd.get_dummies(df, drop_first=True)
df.head()
```

isnull と sum の組み合わせ

データに文字列が含まれているのでダミー変数処理を行う

	survived	pclass	age	fare	sex_male
0	0	3	22.0	7.2500	1
1	1	1	38.0	71.2833	0
2	1	3	26.0	7.9250	0
3	1	1	35.0	53.1000	0
4	0	3	35.0	8.0500	1

193

3. クラスター分析

k-meansの前処理 : データの標準化

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
df_sc = sc.fit_transform(df)
df_sc = pd.DataFrame(df_sc, columns=df.columns)
df_sc.head()
```

StandardScalerを使用

年齢は22・26..., チケットクラスは1・3..., スケールの違いを合わせる : 標準化

	survived	pclass	age	fare	sex_male
0	-0.789272	0.827377	-0.592481	-0.502445	0.737695
1	1.266990	-1.566107	0.638789	0.786845	-1.355574
2	1.266990	0.827377	-0.284663	-0.488854	-1.355574
3	1.266990	-1.566107	0.407926	0.420730	-1.355574
4	-0.789272	0.827377	0.407926	-0.486337	0.737695

194

3. クラスター分析

k-meansのクラスタリングを実行

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=4, random_state=1)
model.fit(df_sc)      n_clustersは分割するクラスタ数
```

sklearnからKMeansをインポートし、モデルを作成

```
cluster = model.labels_
cluster      labels_ でクラスタ番号を取得
```

```
array([3, 1, 2, 1, 3, 3, 0, 3, 2, 2, 2, 1, 3, 3, 2, 2, 3, 0, 2, 2, 0, 0,
       2, 0, 2, 2, 3, 1, 2, 3, 0, 1, 2, 0, 0, 0, 3, 3, 2, 2, 2, 2, 3, 2,
       2, 3, 3, 2, 3, 2, 3, 3, 1, 2, 0, 0, 2, 3, 2, 3, 3, 1, 0, 3, 0, 3,
       2, 3, 2, 3, 3, 2, 3, 3, 3, 3, 3, 3, 2, 3, 3, 2, 0, 2, 2, 3, 3,
       1, 3, 3, 3, 0, 3, 0, 3, 0, 0, 2, 3, 2, 3, 0, 3, 3, 3, 2, 3, 3, 2,
       0, 2, 3, 2, 2, 3, 0, 3, 1, 2, 3, 3, 3, 2, 0, 3, 3, 3, 2, 3, 3, 3,
```

195

3. クラスター分析

k-meansのクラスタリングを実行

```
df['cluster'] = cluster
df      DataFrame の新しいカラムに
        クラスタ番号を追加
```

	survived	pclass	age	fare	sex_male	cluster
0	0	3	22.000000	7.2500	1	3
1	1	1	38.000000	71.2833	0	1
2	1	3	26.000000	7.9250	0	2
3	1	1	35.000000	53.1000	0	1
4	0	3	35.000000	8.0500	1	3
...	...	...	...	...	...	...
886	0	2	27.000000	13.0000	1	3
887	1	1	19.000000	30.0000	0	2

196

3. クラスター分析

クラスタリングの結果を確認

```
# style.bar で DataFrame にカラーバーを追加
df.groupby('cluster').mean().style.bar(axis=0)
```

読み込ませたデータの平均値をクラスターごとに確認

cluster	survived	pclass	age	fare	sex_male
0	0.280000	1.320000	42.981210	37.686277	0.986667
1	0.925926	1.009259	33.376021	123.153357	0.148148
2	0.650000	2.645455	25.380733	18.052823	0.000000
3	0.138015	2.828087	26.213949	13.968097	1.000000

197

3. クラスター分析

クラスタリングの結果を確認

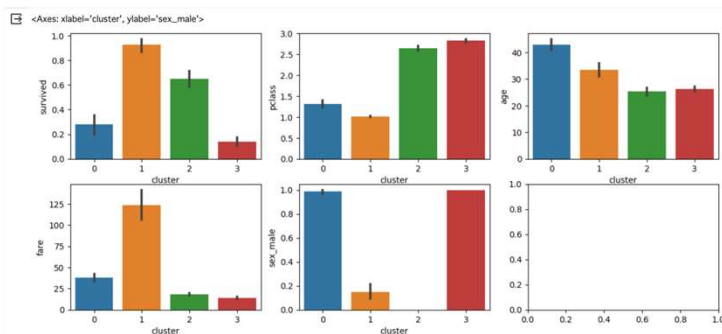
```
import matplotlib.pyplot as plt
fig, axes = plt.subplots(2,3, figsize=(14, 6))
sns.barplot(ax=axes[0,0], data=df, x='cluster', y='survived')
sns.barplot(ax=axes[0,1], data=df, x='cluster', y='pclass')
sns.barplot(ax=axes[0,2], data=df, x='cluster', y='age')
sns.barplot(ax=axes[1,0], data=df, x='cluster', y='fare')
sns.barplot(ax=axes[1,1], data=df, x='cluster', y='sex_male')
```

198

3. クラスター分析

クラスタリングの結果を確認

クラスタ0：男性で高齢，良い客室に泊まっており，生存率が低い
 クラスタ1：女性で良い客室に泊まり運賃が最も高く，生存率も一番高い
 クラスタ2：女性で安い客室に泊まっている
 クラスタ3：男性で若年，安い客室に泊まっており，一番生存率が低い



199

3. クラスター分析

クラスタリングの結果を確認：主成分分析でグラフ化

```
df_sc['cluster'] = cluster
```

```
from sklearn.decomposition import PCA  PCAをインポートし学習させる
pca = PCA(n_components=2, random_state=1)
pca.fit(df_sc)
feature = pca.transform(df_sc)
feature
```

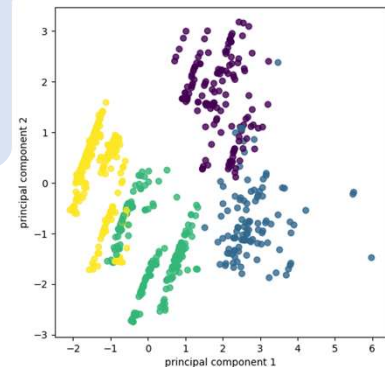
```
array([[ -1.79546026,  0.2666309 ],
       [  2.61164075, -0.78970203],
       [  0.02143814, -1.73958101],
       ...,
       [-0.43069953, -0.47767583],
       [  2.14671007,  0.27131272],
       [-1.59197757,  0.65939251]])
```

200

3. クラスター分析

クラスタリングの結果を確認：主成分分析でグラフ化

```
import matplotlib.pyplot as plt  散布図で可視化
plt.figure(figsize=(6, 6))
plt.scatter(feature[:, 0], feature[:, 1], alpha=0.8,
            c=cluster)
plt.xlabel('principal component 1')
plt.ylabel('principal component 2')
plt.show()
```



201

3. クラスター分析（その2）

階層的クラスタリング（Hierarchical Clustering）

- 最も類似する（または最も類似しない）サンプルデータの組み合わせを見つけ出し、順番にグループ分けしていく手法
- 樹形図（デンドログラム）をプロットできる
- 二分木で階層的クラスタリングを可視化したもの

202

3. クラスター分析（その2）

計算の流れ

- 各サンプルを単一のクラスターとみなし，全てのクラスター間のユークリッド距離を計算
- クラスター間の距離に基づき，クラスターを連結
- クラスター情報を更新し，ユークリッド距離を再計算
- クラスターが最終的に1つになるまで繰り返す

203

3. クラスター分析（その2）

流れ

- 事前準備
- 最適なクラスターの数を見つけるための樹形図の作成
- クラスター数の決定・モデル学習
- クラスター可視化

204

3. クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

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

```
import pandas as pd
df = pd.read_csv('sample_7.csv')
df.head()
```

サンプルデータをインポート

統計センターのSSDSE（教育用標準データセット）を利用
<https://www.nstac.go.jp/use/literacy/ssdse/>

205

3. クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

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

```
import pandas as pd
df = pd.read_csv('sample_7.csv')
df.head()
```

サンプルデータをインポート

統計センターのSSDSE（教育用標準データセット）を利用
<https://www.nstac.go.jp/use/literacy/ssdse/>

	Unnamed: 0	総人口	幼稚園の数(10万人あたり)	就業時間の平均(時間/月)	科学技術振興費の研究費(万円/10万人あたり)	情報通信業の企業数
0	北海道	5250000	7.695238	12.0	95.238095	143
1	青森県	1246000	7.062600	10.5	0.000000	30
2	岩手県	1227000	7.497963	11.5	2086.389568	36
3	宮城県	2306000	10.320902	12.5	1036.426713	51
4	秋田県	966000	4.037267	9.0	5621.118012	16

206

3. クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

```
import numpy as np
df.describe()
```

基本統計量を確認

	総人口	幼稚園の数(10万人あたり)	残業時間の平均(時間/月)	科学技術振興費の研究費(万円/10万人あたり)	情報通信業の企業数
count	4.700000e+01	47.000000	47.000000	47.000000	47.000000
mean	2.684404e+06	8.555025	11.553191	1584.176816	117.425532
std	2.779720e+06	3.126166	1.372217	1573.513577	419.861610
min	5.560000e+05	3.597122	8.500000	0.000000	9.000000
25%	1.075500e+06	6.671337	10.500000	807.490314	26.500000
50%	1.602000e+06	7.796773	11.500000	1142.857143	34.000000
75%	2.693500e+06	10.313550	12.500000	1660.102424	57.500000
max	1.392100e+07	16.758242	15.000000	7591.145833	2899.000000

207

3. クラスター分析（その2）

事前準備（SSDSE：教育用標準データセット）

```
import scipy.cluster.hierarchy as sch
# import matplotlib.pyplot as plt
X = df.iloc[:, [1, 3]].values
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
```

クラスタリング軸設定（人口vs残業時間の平均）

樹形図作成用インスタンス

scipy.cluster.hierarchyクラスからインスタンスを生成することで、樹形図を作成
インスタンス内のlinkage()メソッドはデータ間の距離を計算しつながりを作る。
連結方法としてウォード連結法で距離を計算。

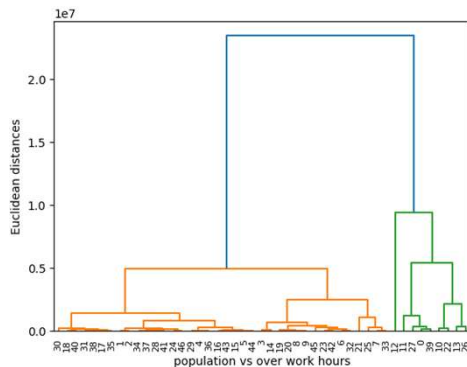
208

3. クラスター分析（その2）

最適なクラスターの数を見つけるための樹形図の作成

```
plt.xlabel('population vs over work hours')
plt.ylabel('Euclidean distances')
plt.show()
```

ラベルを貼りながら出力



この結果から2つのクラスターで分割するのが良いかもしれない。

209

3. クラスター分析（その2）

モデル学習

```
from sklearn.cluster import AgglomerativeClustering
```

モデルの訓練

```
hir_clus = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean',
linkage = 'ward')
y_hir_clus = hir_clus.fit_predict(X)
```

第一引数：クラスター数（n_clusters）デフォルト値は2

第二引数：距離のパラメータ（affinity）

第三引数：クラスター連結法（linkage）

210

3. クラスター分析（その2）

モデル学習によって予想されたクラスターの結果

```
print(y_hir_clus)
```

➡ [01111111002011111101110011111111
1101111111]

211

3. クラスター分析（その2）

クラスターの可視化

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm

cluster_labels = np.unique(y_hir_clus)
n_clusters = cluster_labels.shape[0]

for i in range(len(cluster_labels)):
    color = cm.jet(float(i) / n_clusters)
    plt.scatter(X[y_hir_clus == i, 0], X[y_hir_clus == i, 1], s = 50, c = color, label =
'Cluster'+str(i))

plt.title('Clusters of population')
plt.xlabel('population')
plt.ylabel('over work hours')
plt.legend(loc="best")
plt.show()
```

クラスタの配列情報
一意なクラスタ要素
配列の長さ

可視化

グラフに関する情報

212

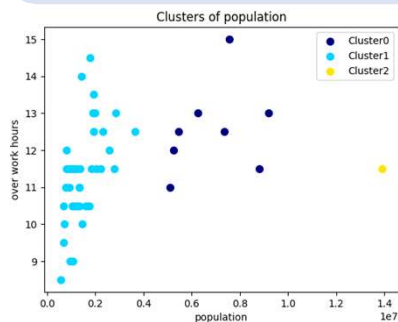
3. クラスター分析（その2）

クラスターの可視化

（以下再掲）

```
plt.title('Clusters of population')
plt.xlabel('population')
plt.ylabel('over work hours')
plt.legend(loc="best")
plt.show()
```

グラフに関する情報



213

参考文献（継続して学習したい方向け）

- 谷合廣紀（2018）
『Pythonで理解する統計解析の基礎』技術評論社
- 塚本邦尊（著）、山田典一（著）、大澤文孝（著）、中山浩太郎（監修）（その他）、松尾豊（監修）（2019）
『東京大学のデータサイエンティスト育成講座 Pythonで手を動かして学ぶデータ解析』マイナビ出版
- 馬場真哉（著）（2022）
『Pythonで学ぶあたらしい統計学の教科書 第2版』翔泳社

214

データサイエンス・オンライン講座のご紹介

「誰でも使える統計オープンデータ」受講者募集中！

講座の目的：e-Stat、jSTAT MAP、API機能等を使い、
統計オープンデータを活用したデータ分析の
基本的な知識を習得する

開 講 期 間：令和6年1月16日（火）～3月19日（火）

学 習 時 間：1回10分程度×5～7回程度（1週間）×4週

課 題：各週の確認テストと最終課題の実施

講 師：西内啓氏（統計家）ほか



申込はこちらから！



週	各週のテーマ	内 容
1	e-Statを使った データ分析	e-Statの統計データを活用したデータ分析の事例、基本的な活用方法を学ぶ (e-Statの機能紹介、活用事例紹介等)
2	公的統計データの 使い方	公的統計データの基本事項及び読み方を学ぶ (公的統計の種類と体系、労働力調査・家計調査の基礎知識及び利用の際のポイント等)
3	地図で見る統計 (jSTAT MAP) の 活用	統計データと地図を組み合わせた活用方法を学ぶ (地図で見る統計 (jSTAT MAP) の機能紹介、簡単にできるレポート作成、活用事例紹介等)
4	統計オープンデータ の高度利用	統計API機能の仕組みや具体的な活用事例等の統計オープンデータの高度な活用方法を学ぶ (統計APIの仕組み、統計オープンデータの活用事例、講座のまとめ等)