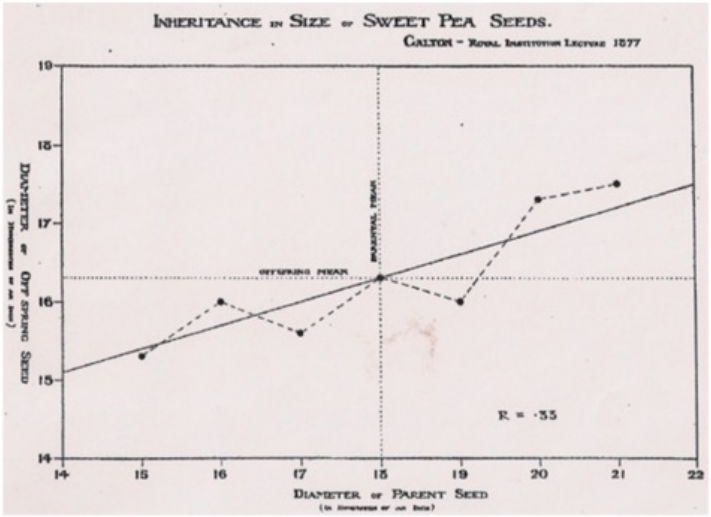
第4回
scikit-learnによる回帰分析

目次

- 回帰とは
- 回帰分析
- 演習で使用するモジュール
- 簡単な回帰分析
  - 必要なモジュールのインポート
  - データのGoogle Colabへのアップロード
  - データの取り込み
  - 2つの量の関係をグラフ化する
- 機械学習のモジュール
  - scikit-learnを用いたパラメータの推定
  - 回帰直線の描画
- 付録
  - 都道府県名の表示
  - 課題

回帰とは

- 回帰は実数値を予測する教師あり機械学習
  - 語源：ダーウィンの弟子ゴルトン
    - 大きい（小さい）種から育ったスイートピーの種からは大きい（小さい）種が得られるはずと考えたが、実際に得られた種の大きさの平均はある傾向の直線に向かって帰っていった。
    - 回帰直線

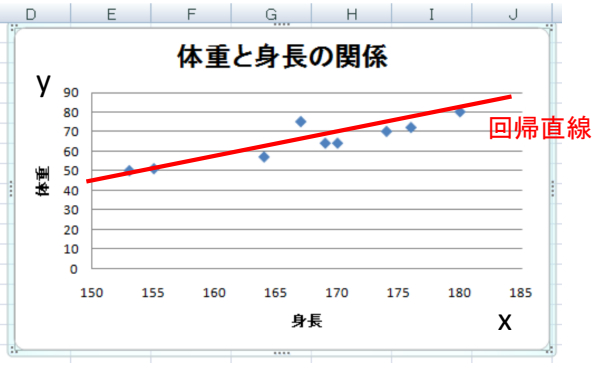


ゴルトンによる世界で最初の回帰直線（1877年）
出典：https://rss.onlinelibrary.wiley.com/doi/full/10.1111/j.1740-9713.2009.00379.x#sign379-sec-0050-title (Royal Statistical Society)

回帰分析

- 回帰分析とは、ある2つの変数にどのような関係があるのかを推定する機械学習による分析手法です。
  - 例えば、あるクラスの生徒について、身長xと体重yの一覧表があったとします。身長xが高い人ほど体重yが大きいといった関係がある場合に、xからy（またはその逆）を推定する式を求めるような分析が回帰分析です。
- 線形回帰モデル
  - 例えば、身長をx、体重をyとした時に、xからyを推定する式が直線の方程式（$y = ax + b$）で与えられたとします。このような直線のモデル式を線形回帰モデルを呼び、描かれる直線を回帰直線と呼びます。
    - このとき、xを説明変数、yを目的変数と呼び、aは傾きのパラメータ、bは切片のパラメータです。

	A	B	C
1	氏名	身長	体重
2	山田太郎	170	64
3	鈴木陽一	155	51
4	三宅太一	164	57
5	杉浦勉	169	64
6	吉永英輔	180	80
7	加藤肇	153	50
8	大杉美紀久	167	75
9	望月敏夫	176	72
10	牧野翼	174	70



$$y = ax + b$$

線形回帰モデル（直線の方程式）

演習で使用するモジュール

- 下記の表の赤文字のモジュールを使用します。

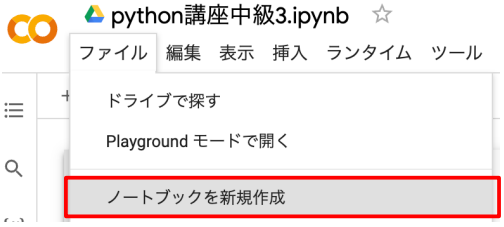
モジュール名	概要
matplotlib	グラフなどの可視化モジュール
seaborn	matplotlibの見栄えをより綺麗にするモジュール
NumPy	計算を効率的に行うためのモジュール
SymPy	数式・記号計算用モジュール
pandas	データ解析支援モジュール（Excelファイルの読み書きが可能）
openpyxl	Excelファイルの読み書きに特化したモジュール
xlwings	Excelアプリを直接制御できるモジュール
scipy	NumPyを利用した数値解析モジュール
scikit-learn	機械学習モジュール
NetworkX	グラフ・ネットワーク計算と可視化モジュール
Basemap	地図描画モジュール
MeCab	形態素解析モジュール
wordcloud	ワードクラウド可視化モジュール

簡単な回帰分析

- 回帰分析の一例として、各都道府県の「日本人宿泊者数」から「外国人宿泊者数」を推定するような回帰直線を求めてみましょう。
- 準備
  - 資料電子データサイトの「kaiki.xlsx」ファイルを使用します。
  - 「宿泊者数」シートには、47都道府県ごとのコロナ禍前のある年度の「日本人宿泊者数」と「外国人宿泊者数」の人数がまとめられています。

	A	B	C
1	都道府県名	日本人宿泊者数	外国人宿泊者数
2	北海道	12068390	4632700
3	青森県	1301130	163130
4	岩手県	2134780	174840
5	宮城県	3158640	130230
6	秋田県	1042670	64440
7	山形県	2104420	110710

  - Google Colabで新しいノートブックを新規作成してください。



必要なモジュールのインポート

- pandasモジュールと、matplotlib.pyplotモジュールをインポートします。また、日本語フォントをインストールします。

```
#データ解析用モジュール
import pandas as pd
#日本語フォントのインストール
!pip install japanize-matplotlib
#グラフ描画用モジュール
import matplotlib.pyplot as plt
import japanize_matplotlib
```

- 実行すると、日本語フォントのインストールが開始されます。

データのGoogle Colabへのアップロード

- kaiki.xlsxをGoogle Colabにアップロードして、読み込めるようにします。

- 手順
  - Google Colab左のフォルダマークをクリックする
  - ファイルツリーが開くので、資料電子データサイトからダウンロードしたkaiki.xlsxをドラッグ＆ドロップしてアップロードする
    - ドラッグ＆ドロップができない場合はアップロードボタンをクリックしてファイルを選択します
  - アップロード後は、再びフォルダマークをクリックしてファイルツリーは閉じてもよい

データの取り込み

- pandasではExcelファイルを直接読み込むことができます。
  - 新しくコードセルを追加し、下記を記述して実行します。

```
#pandasのインポート
import pandas as pd

file = pd.ExcelFile("kaiki.xlsx")
data = file.parse("宿泊者数")
data.head()
```

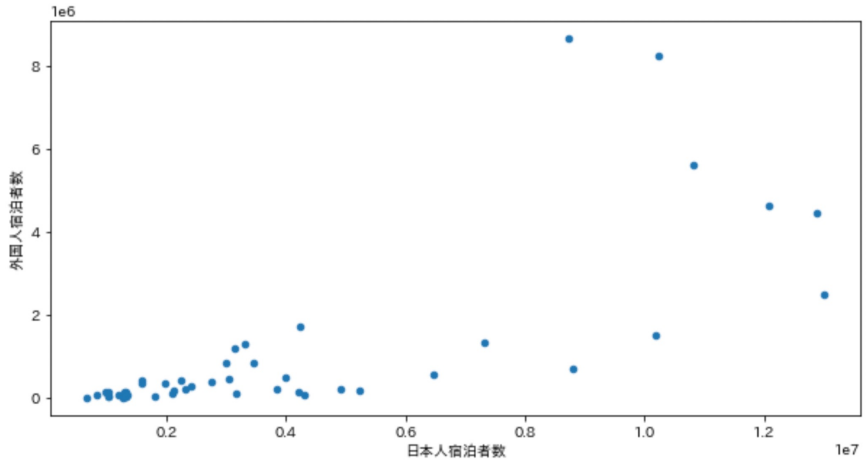
取り込んだデータには自動的に連番のindexが割り当てられます。

	都道府県名	日本人宿泊者数	外国人宿泊者数
0	北海道	12068390	4632700
1	青森県	1301130	163130
2	岩手県	2134780	174840
3	宮城県	3158640	130230
4	秋田県	1042670	64440

2つの量の関係をグラフ化する

- xとyの2つの軸について、データの散らばり具合を描くのが散布図です。散布図を描くことによって、2つの量の関係性を図示することができます。
- 新しいコードセルを追加して、下記を記述して実行しましょう。

```
#plot.scatterメソッドで散布図が描けます。
#xはx軸の列名、yはy軸の列名、figsizeはグラフの横縦のサイズです。
#axにはグラフのオブジェクト（サブプロットと呼ばれる）が代入されます。
ax = data.plot.scatter(x="日本人宿泊者数", y="外国人宿泊者数", figsize=(10, 5))
```



1e7は10^7という意味です。従って、1.2e7は1200万です。

機械学習のモジュール

- Pythonには様々な機械学習のモジュールが用意されています
- モジュールを用いる利点
  - それぞれのアルゴリズムを1から実装する必要がない
  - コードが簡潔に書ける
- 代表的な機械学習モジュール
  - scikit-learn
    - 回帰、ランダムフォレストなど様々なアルゴリズムを備えたモジュール
  - genism
    - 様々なトピックモデルを実装したテキスト解析モジュール
  - Tensor Flow
    - 代表的な深層学習モジュール
  - Keras
    - TensorFlowをベースにより扱いやすくした深層学習モジュール

scikit-learnを用いたパラメータの推定

- scikit-learnモジュールのlinear_modelパッケージのLinearRegressionクラスを使うと回帰直線の傾きaと切片bのパラメータを求めることができます。
  - linear model（和訳：線形モデル）、linear regression（和訳：線形回帰）
- 新しいコードセルを追加して、下記を記述して実行しましょう。

```
#LinearRegressionクラスのインポート
from sklearn.linear_model import LinearRegression

#LinearRegressionオブジェクトの生成
reg = LinearRegression()
#xは日本人宿泊者数のデータ
x = data["日本人宿泊者数"].values.reshape(-1, 1)
#yは外国人宿泊者数のデータ
y = data["外国人宿泊者数"]
#学習してパラメータを計算
reg.fit(x, y)
#傾きと切片の表示
print("傾き", reg.coef_[0])
print("切片", reg.intercept_)
```

【コードの解説】
下方のfitメソッドにx軸の値を与える際に2次元（行列）形式として渡す必要があるため、1次元のベクトル形式から2次元の行列形式に変換している。
.reshape（行数, 列数）として行列形式に変換する際に、引数として-1を指定すると、元のデータから行数（または列数）を自動的に決めてくれる。今回の例では、47次元ベクトルを47行1列の行列に変換している。
x = pd.DataFrame(x) でも可。

【コードの解説】
coefはcoefficient（係数）の略
interceptは日本語で切片のこと

回帰直線の描画(1)

- 求めたパラメータを使って回帰直線を描きましょう。

$$y = ax + b$$

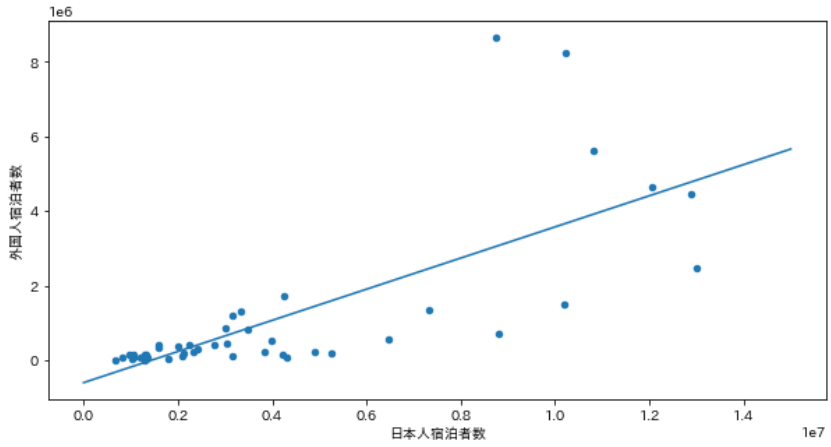
外国人宿泊者数　傾き reg.coef_[0]　日本人宿泊者数　切片 reg.intercept_

  - 新しいコードセルを追加して、下記を記述して実行しましょう。

```
#傾きをa、切片をbに代入
a = reg.coef_[0]
b = reg.intercept_
#xが0でのyの値を計算
x0 = 0
y0 = a * x0 + b
#xが1500万人でのyの値を計算
x1 = 1.5e7
y1 = a * x1 + b
ax = data.plot.scatter(x="日本人宿泊者数", y="外国人宿泊者数", figsize=(10, 5))
ax.plot([x0, x1], [y0, y1]) #xが0とx が1500万人で直線を描く
```


回帰直線の描画(2)

- 日本人宿泊者数から外国人宿泊者数を推定する線形回帰モデルが完成しました。



- 一旦パラメータが求められれば、下記の式で外国人宿泊者数の推定値が計算できます。

日本人宿泊者数

```
print(a * 10000000 + b)
```

```
3576416.479152304
```

付録

(付録)都道府県名の表示(1)

- グラフ中のサンプルポイントに、都道府県名を表示しましょう。
  - p13のコードに下記の赤枠のコードを追記して実行しましょう。

```
#傾きをa、切片をbに代入
a = reg.coef_[0]
b = reg.intercept_
#xが0でのyの値を計算
x0 = 0
y0 = a * x0 + b
#xが1500万人でのyの値を計算
x1 = 1.5e7
y1 = a * x1 + b
ax = data.plot.scatter(x="日本人宿泊者数", y="外国人宿泊者数", figsize=(10, 5))
ax.plot([x0, x1], [y0, y1]) #xが0とx が1500万人で直線を描く
for k, v in data.iterrows():
    ax.annotate(v[0], xy=(v[1], v[2]), size=12)
```

dataの行を1つずつ取り出して、インデックスをkに、各列をvに入れる

字下げ

グラフのannotateメソッドでグラフ内に文字列が表示できます

都道府県名

座標をタプルで指定

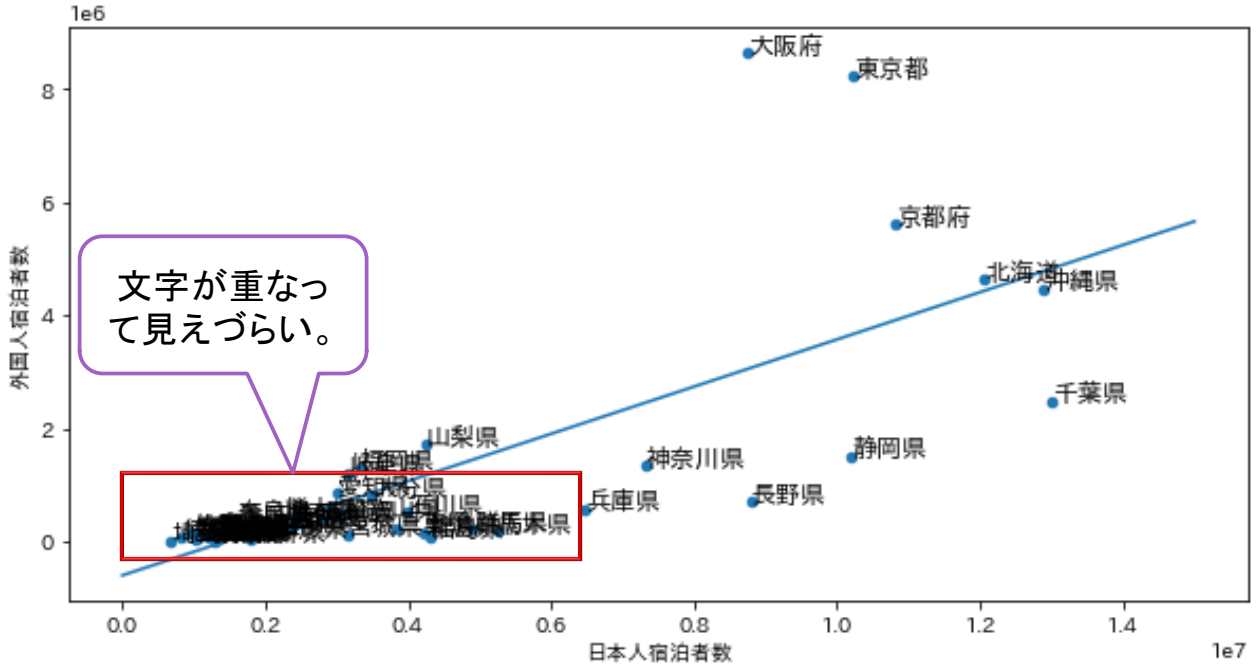
フォントサイズ

k	v[0]	v[1]	v[2]

	都道府県名	日本人宿泊者数	外国人宿泊者数
0	北海道	12068390	4632700
1	青森県	1301130	163130

(付録)都道府県名の表示(2)

- 都道府県名の表示結果



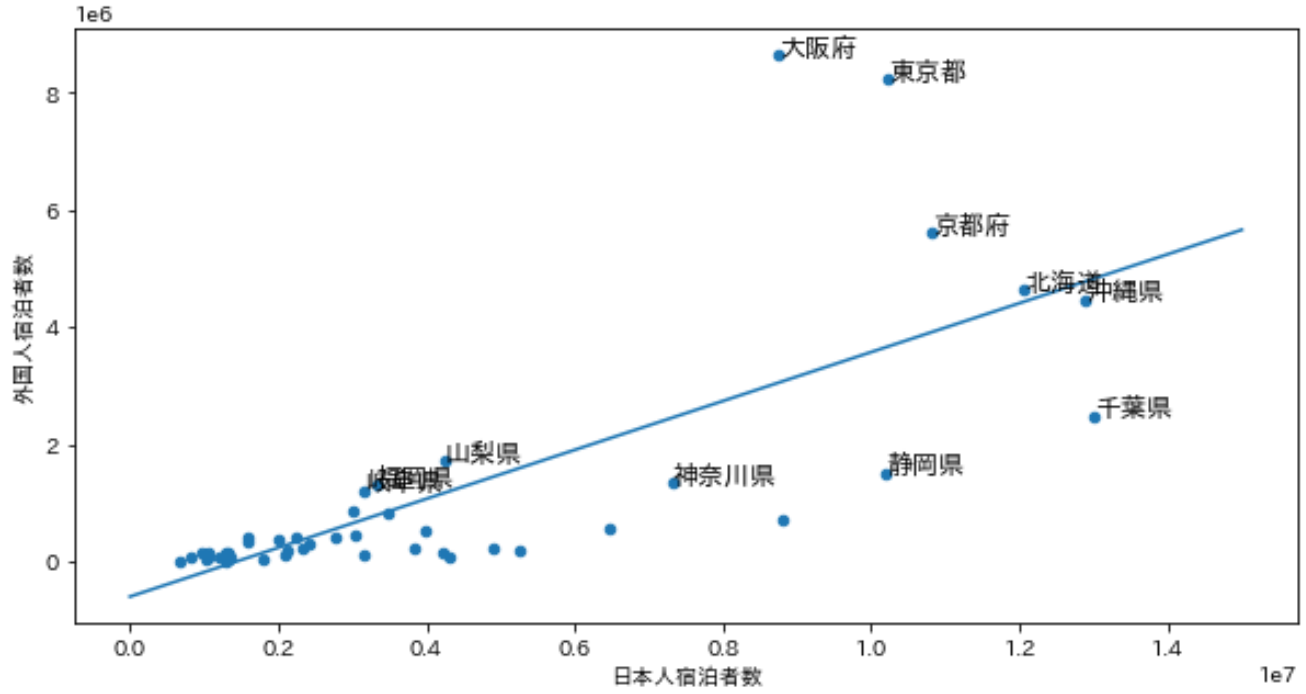
(付録)課題

- 外国人宿泊者数が100万人以下の都道府県については、グラフ中で都道府県名の表示を非表示としてください。
- 解答例：下記の赤枠のif文と、ラベル描画の字下げ
  - p16のコードを下記の赤枠のコードに修正して実行しましょう。

```
#傾きをa、切片をbに代入
a = reg.coef_[0]
b = reg.intercept_
#xが0でのyの値を計算
x0 = 0
y0 = a * x0 + b
#xが1500万人でのyの値を計算
x1 = 1.5e7
y1 = a * x1 + b
ax = data.plot.scatter(x="日本人宿泊者数", y="外国人宿泊者数", figsize=(10, 5))
ax.plot([x0, x1], [y0, y1]) #xが0とx が1500万人で直線を描く
for k, v in data.iterrows():
    if v[2] > 1000000:
        ax.annotate(v[0], xy=(v[1], v[2]), size=12)
```

字下げ

（付録）課題の最終結果


1e6
8
6
4
2
0
外国人宿泊者数
大阪府
東京都
京都府
北海道
沖縄県
千葉県
山梨県
神奈川県
静岡県
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1e7
日本人宿泊者数