高校情報I
「データサイエンス」
～データの分析～
Python & Google Colaboratory演習

高知工科大学 情報学群・データ＆イノベーション学群

本日の予定

- 1時間目：8:45-9:35
  - データサイエンス・仮説検定
- 2時間目：9:45-10:35
  - 演習： t 検定，Google Colaboratory と js-STAR を使用
- 3時間目：10:45-11:35
  - 統計解析，回帰分析
- 4時間目：11:45-12:35
  - 演習：回帰分析

本日の演習で操作などで
分からないことがあったら，
手を挙げて各教室にいる
アシスタント（情報学群学生
＆大学院生）に遠慮なく
質問してください

データサイエンス

データサイエンス

- データから有益な情報を取り出す方法論
  - データの設計，取得，管理，解析，データからの推論 [De Veaux他 2017]
- データの処理・分析を行う
  - 確率・統計の考え方を使った解析
  - データ処理のための様々な情報技術（アルゴリズムやシステム）

- 社会の様々な場所でデータが蓄積されている　　ビッグデータ
  - デジタル化，オンラインサービス，SNS，スマートフォン，etc.
- 理系・文系の様々な分野で必要とされる知識
  - 経済学，経営学，政治，金融，戦略，医学，心理学，薬学，農学，etc.

統計的なデータ解析

- データサイエンスの中心的な処理
- 可視化（グラフ化）
  - データの中の関係性や違いを見出す
    - 散布図
    - 箱ひげ図
    - 棒グラフ
- 統計分析
  - 相関分析 → データの変数間の相関（関係性）の大小を分析する
  - 回帰分析 → データ中のある変数の値から別の変数の値を予測する
- 仮説検定
  - 「見出したいこと」について，統計的に意味が有る（有意）と主張する
    - 科学的に物事を捉える（客観的に物事を捉える）ために広く使われる

仮説検定

仮説検定 (平均値の例)

- 目的：Aの平均値とBの平均値には差があることを示したい
- データ：A と B からいくつかの標本（サンプル）データ D_A, D_B を取得

- 帰無仮説 H_0： A, B の母集団（全体）での平均の差は無い
- 対立仮説 H_1： A, B の母集団（全体）での平均の差が有る

- H_0 が成り立つと仮定したとき， D_A, D_B のようなデータが得られる確率 p （p 値と呼ぶ）を求める
- $p < 0.05, 0.01$ or 0.001 ⇨ 確率が低い → 偶然にしては珍しすぎる

⇨ 帰無仮説 H_0 が間違っていると解釈 → 棄却する
対立仮説 H_1 を支持する → 目的を達成する

数学の背理法の
証明に似た論理構成

様々な仮説検定

- 正規分布を持つと仮定する2つの母集団の平均値の差
- 相関係数が（有意に）あるか （無相関ではないか）
- 回帰分析の係数が（有意に）あるか（係数が0でないか）
  - t 検定 (Student の t 検定，Welchの t 検定)

- 3つ以上の母集団 (正規分布) の平均値の差
  - 分散分析

- その他
  - 中央値の差の検定，母比率の差の検定（適合度検定），etc.

t 検定を行う

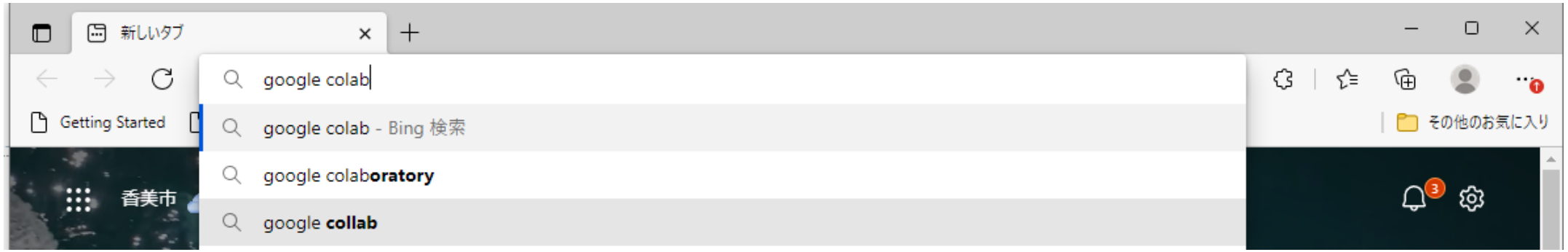
- t 検定で，2つの群からなるデータについて，
  2群間で平均値に差が有るかを調べる

- プログラム言語 Python ＋ Google colaboratory
  - データの生成を行う

- js-STAR
  - Web ブラウザから統計処理を行う

t 検定を Google Colab で行う準備

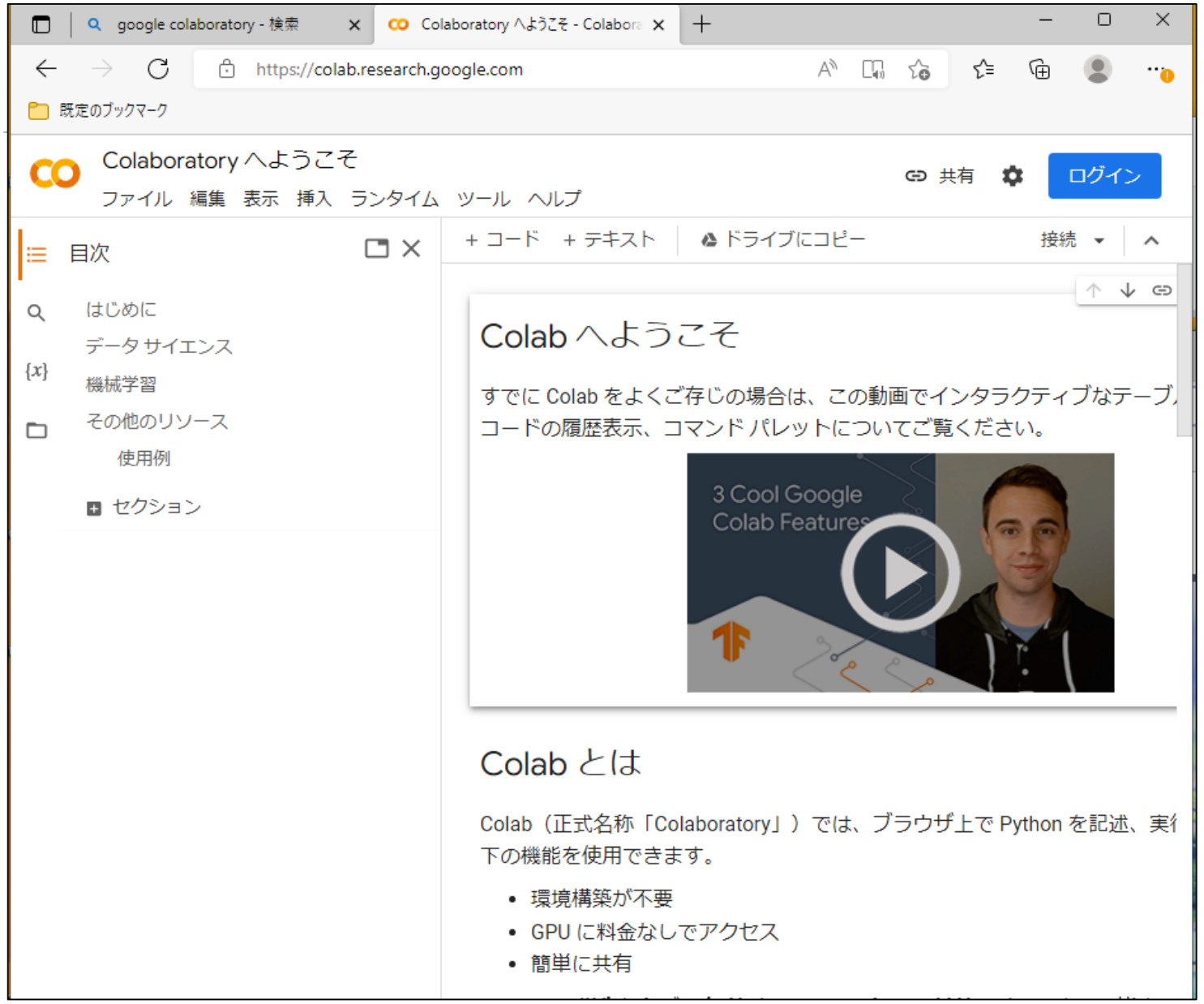
- Chromebook へログイン
- Google colaboratory へアクセス



クリック

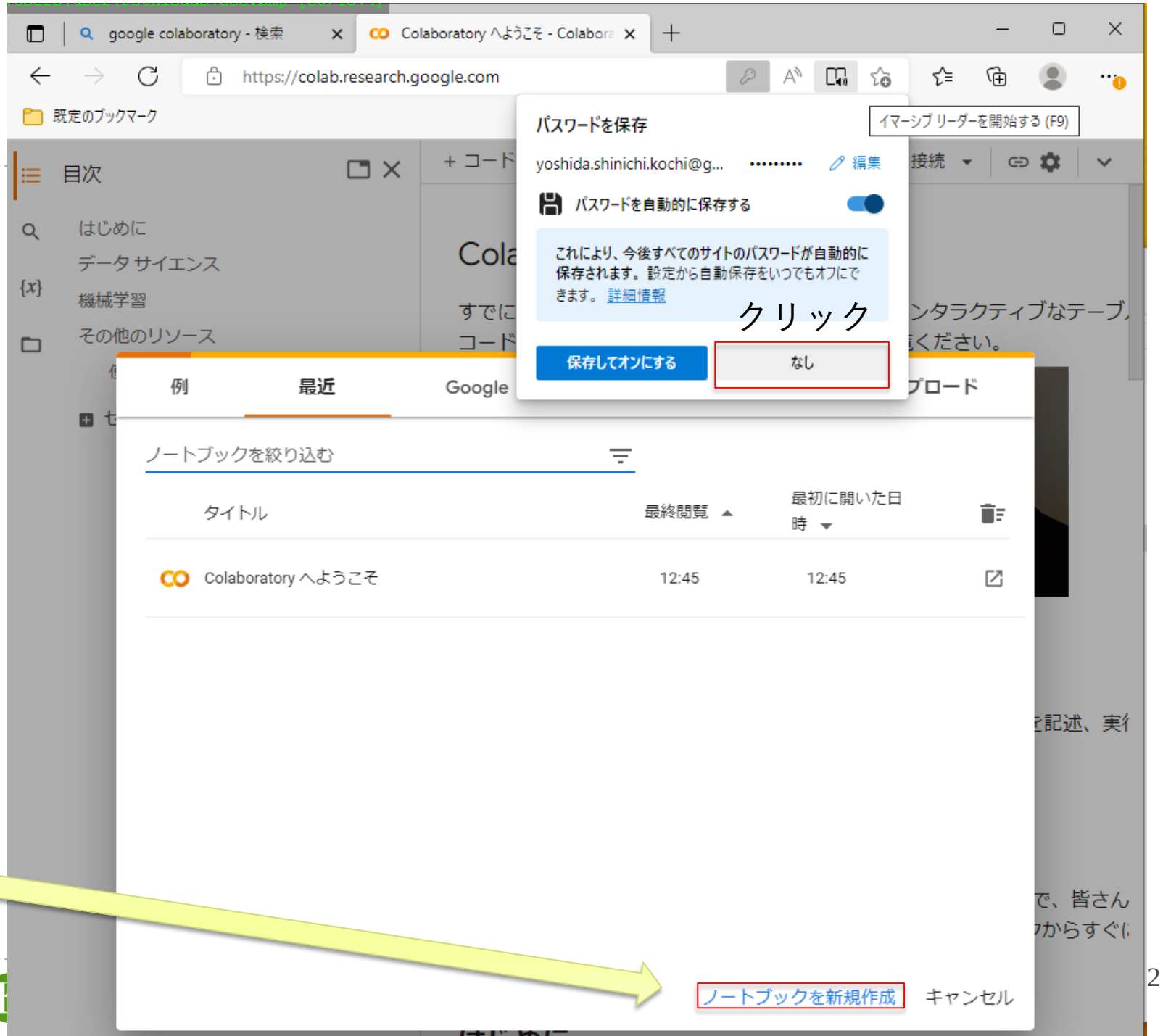
Google Colab

- 右上「ログイン」
  ボタンがある場合は，
  Chromebookで
  使う Google の
  アカウントでログイン



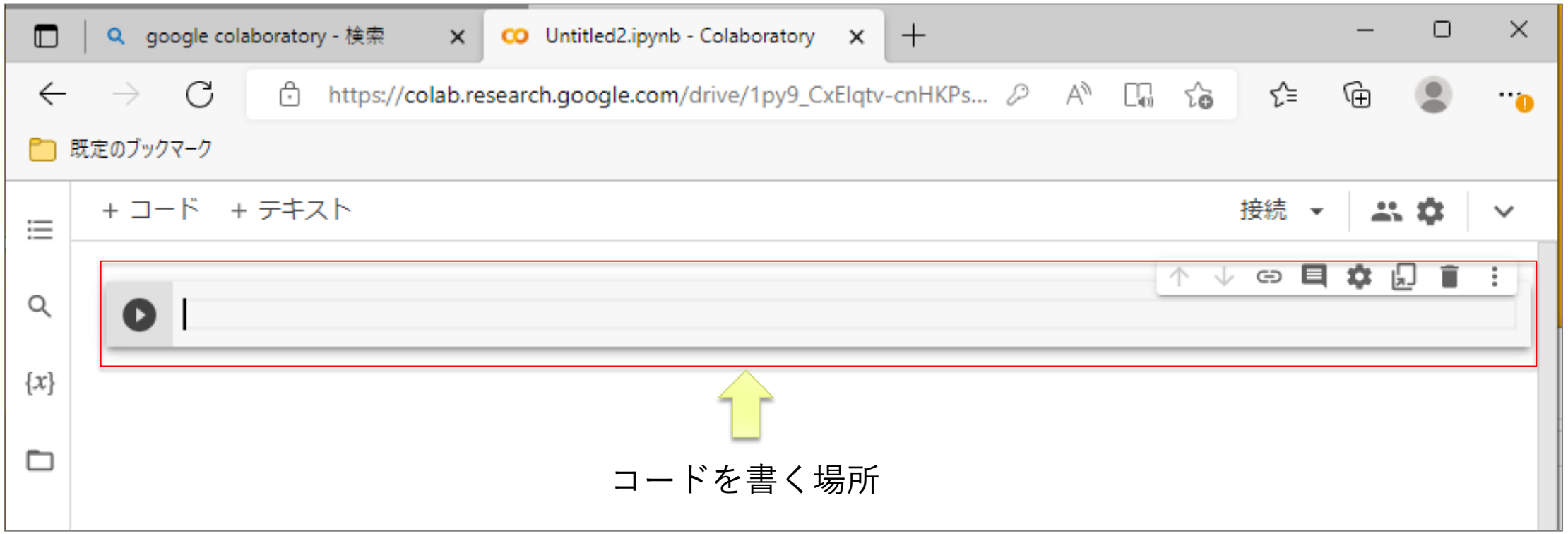
- パスワードは
  保存しない
  (しても問題ない)
- 「ノートブック」
  を新規作成

クリック



Google Colaboratory ノートブック

- ノートブックは，文章(テキスト)，Pythonプログラム(コード)を自由に書き込み保存できる．
- コードを実行することもできる．



t 検定を試してみる

- 乱数を使って，それぞれ標本数13（サンプル）の2つのデータを作成
  - A： 平均値40, 標準偏差13
  - B： 平均値60, 標準偏差13

下記ホームページからプログラムコードを取得してください

```
import numpy as np

np.random.seed(seed=1) # 乱数のシード(種)

# 正規分布の乱数を13個取得
# 平均40, 標準偏差10, サンプル数13
sample1 = np.random.normal(40, 10, 13)
# 平均60, 標準偏差10, サンプル数13
sample2 = np.random.normal(60, 10, 13)

#縦に値を順番に表示
for i in range(len(sample1)):
  print(sample1[i])
for i in range(len(sample2)):
  print(sample2[i])
```

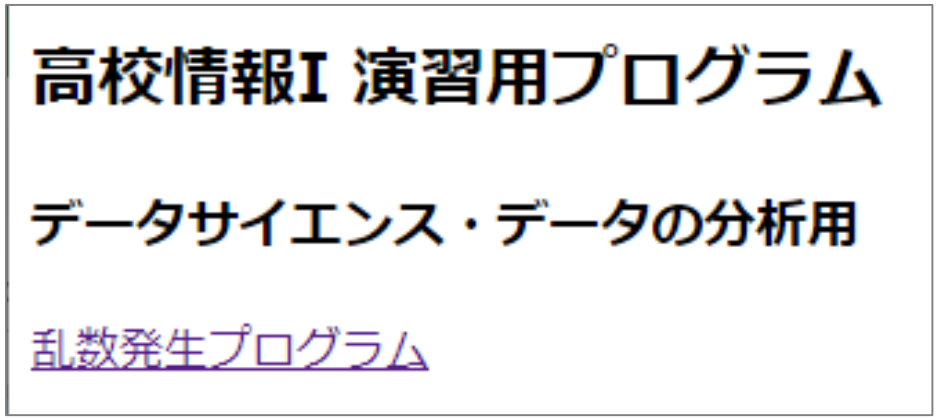
- ホームページ
  - Google検索「高知工科大学 吉田研」

クリック

プログラムのダウンロード

- 上部「高校情報I」 をクリック

- 「乱数発生プログラム」をクリック



プログラムのダウンロード

- 全て選択する

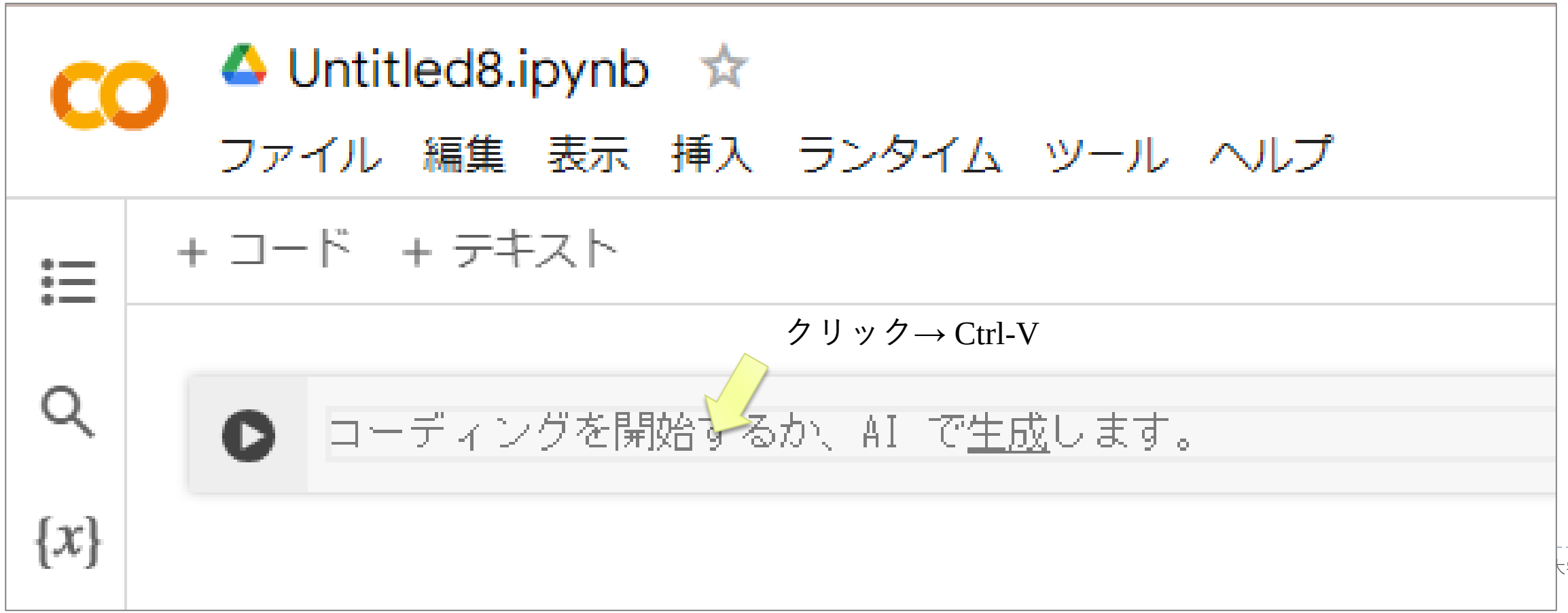
```
import numpy as np

np.random.seed(seed=1) # 乱 数 の シ ー ド (種 )

# 正 規 分 布 の 乱 数 を 13個 取 得
# 平 均 40, 標 準 偏 差 10, サ ン プ ル 数 13
sample1 = np.random.normal(40, 10, 13)
# 平 均 60, 標 準 偏 差 10, サ ン プ ル 数 13
sample2 = np.random.normal(60, 10, 13)

#縦 に 値 を 順 番 に 表 示
for i in range(len(sample1)):
  print(sample1[i])
for i in range(len(sample2)):
  print(sample2[i])
```

画面内を**マウスで全選択するか**,
画面内でマウスをクリックし,
Ctrl - A
（Ctrl (コントロールキー)を
押しながら A を押す）
（青く選択される）

次に，**マウスで右クリック→コピーするか**,
Ctrl – C
（Ctrl キーを押しながら C）

Google Colaboratory ノートブックにペースト

- グレーの枠内（「コーディングを開始するか」のところ）をクリックし，ペーストする（「**編集」→「貼り付け」するか，Ctrl – V** (Ctrl キーを押しながら V)



乱数13個を2つ出力

- 実行（再生ボタン）
- 13個×2で，26個が縦に出力される

- 上13個：データA
- 下13個：データB

- データA, B で，平均値の差はあるか？

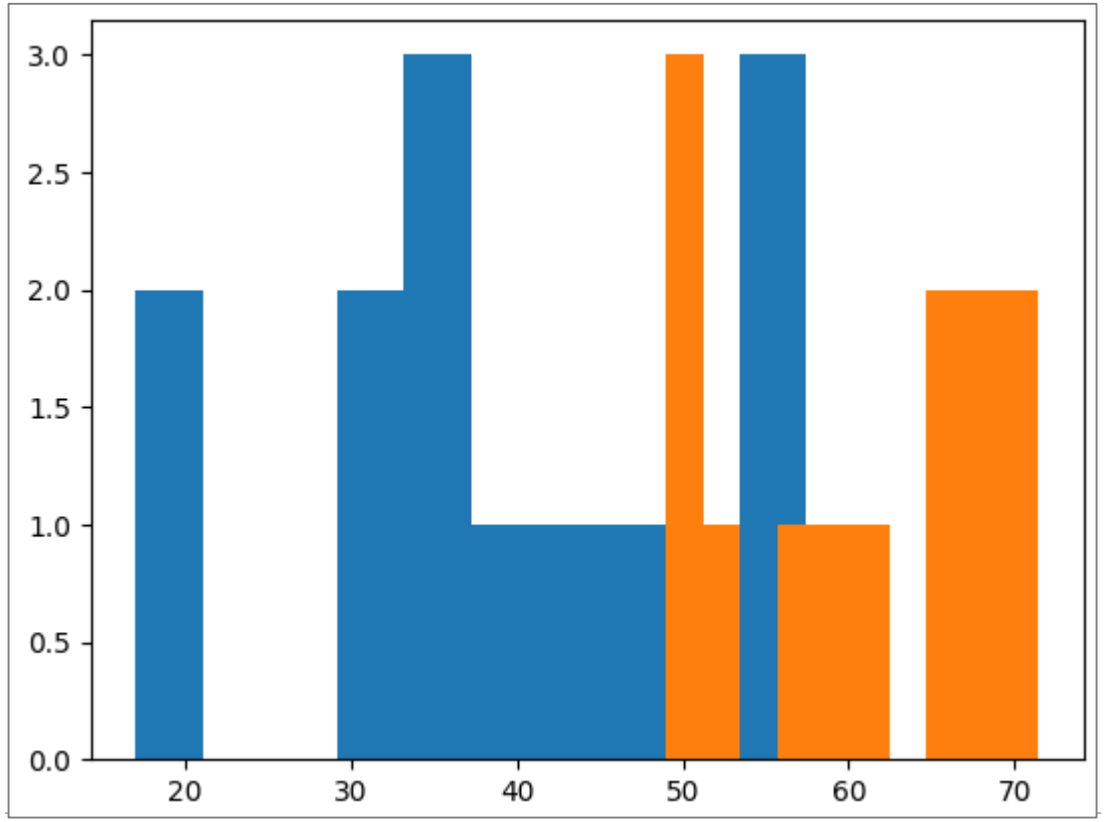
- t 検定を行う

```python
import numpy as np

np.random.seed(seed=1) # 乱数のシード(種)

# 正規分布の乱数を13個取得
# 平均40，標準偏差10，サンプル数13
sample1 = np.random.normal(40, 10, 13)
# 平均60，標準偏差10，サンプル数13
sample2 = np.random.normal(60, 10, 13)

#縦に値を順番に表示
for i in range(len(sample1)):
  print(sample1[i])
for i in range(len(sample2)):
  print(sample2[i])
```

```
56.24345363663242
33.882435863499246
34.71828247736544
29.270313778438293
48.65407629324679
16.984613031197174
57.4481176421648
32.387930991048975
43.19039096057099
37.5062962452259
54.62107937044974
19.39859290502346
36.77582795986493
56.15945645331584
71.33769442335438
49.00108732685969
58.275717924495645
51.221415820786284
60.42213746715593
65.82815213715823
48.99380822787079
71.44723709839614
69.01590720592796
```

sample1 と sample2 のヒストグラムを見る

- 標本数がそれぞれ13なので，正規分布の形状は見られないが，sample1 (青)の方がsample2(オレンジ)より小さいことが分かる



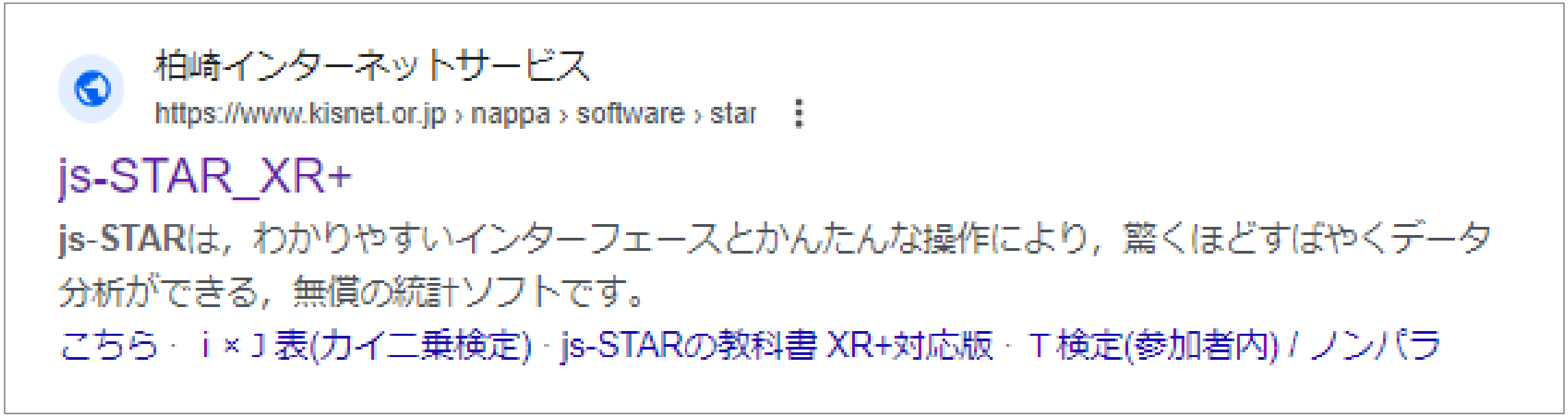

```
import matplotlib.pyplot as plt

# ヒストグラムの作成
plt.hist(sample1)
plt.hist(sample2)

# グラフの表示
plt.show()
```

統計解析Webサイト js-STAR

- ブラウザのタブを一つ新しく開いて
  「js-star」を検索

無償統計解析Webサイト

- js-STAR

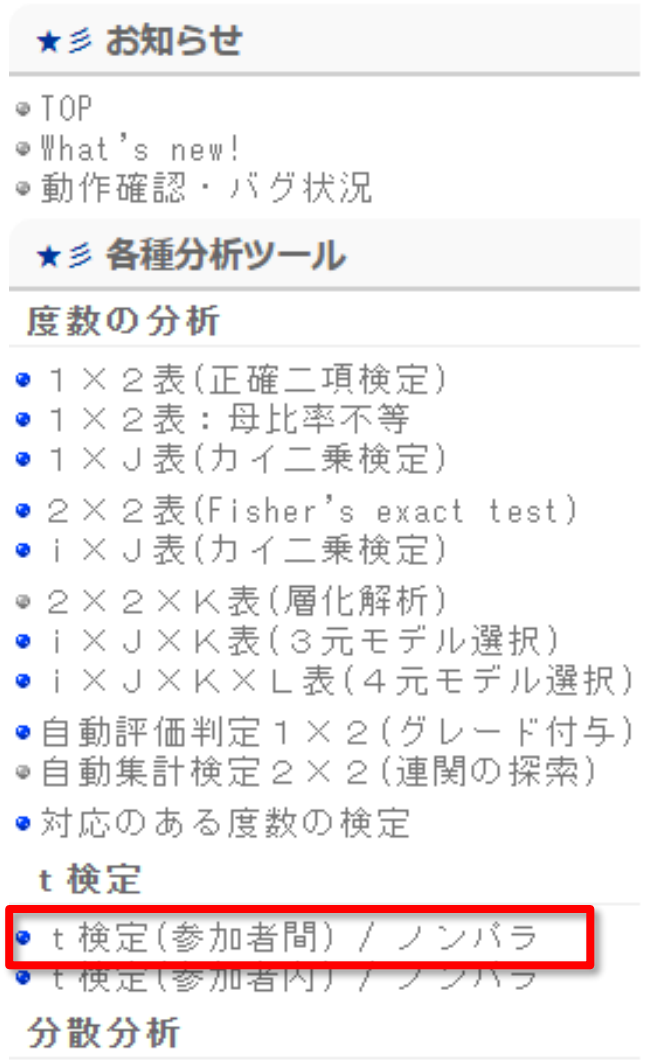


js-STAR

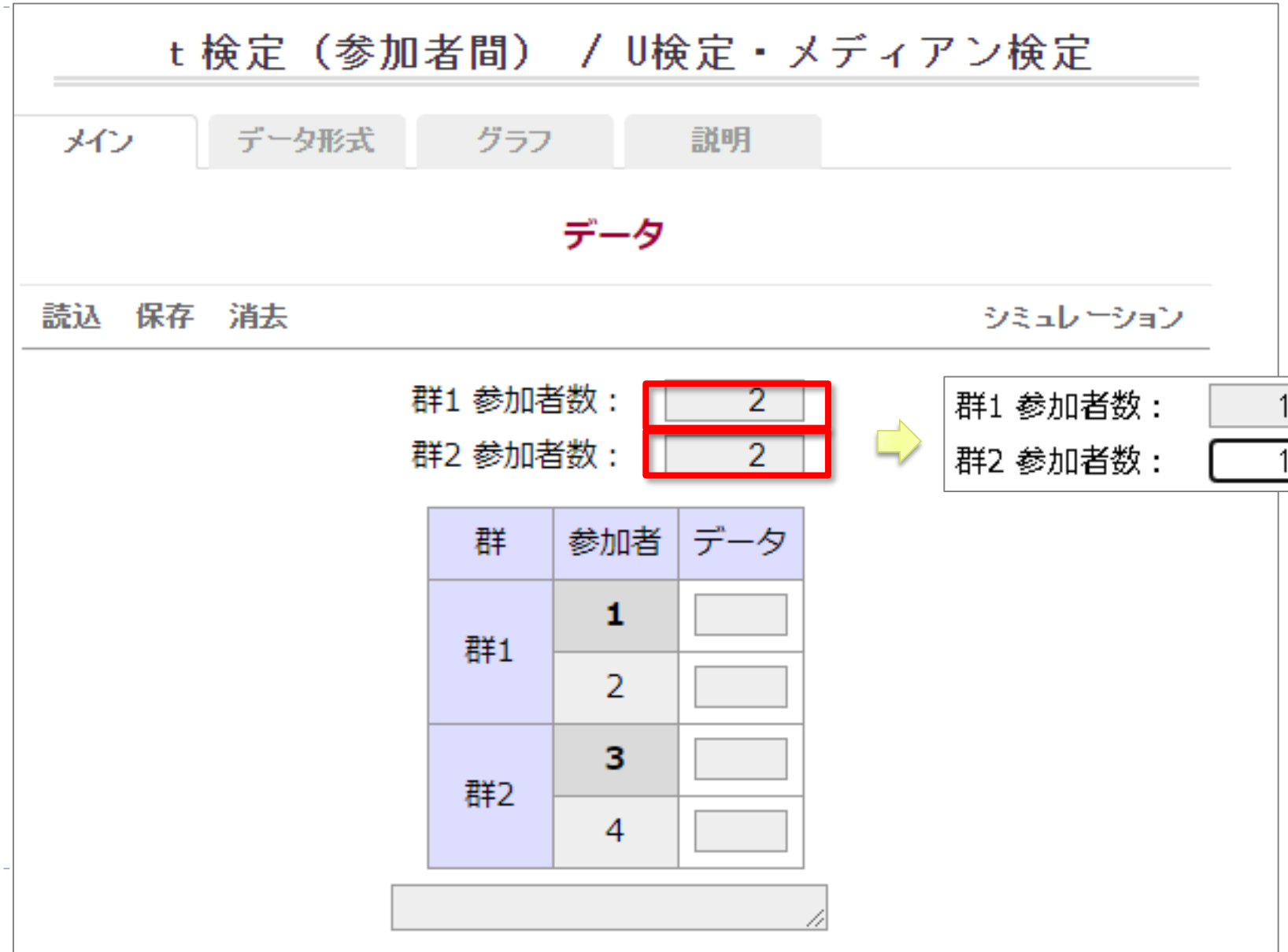
- 様々な統計解析がブラウザから行えるサイト

t 検定を選択

- 左メニューから t 検定を選択
- t 検定（参加者間）／ノンパラ

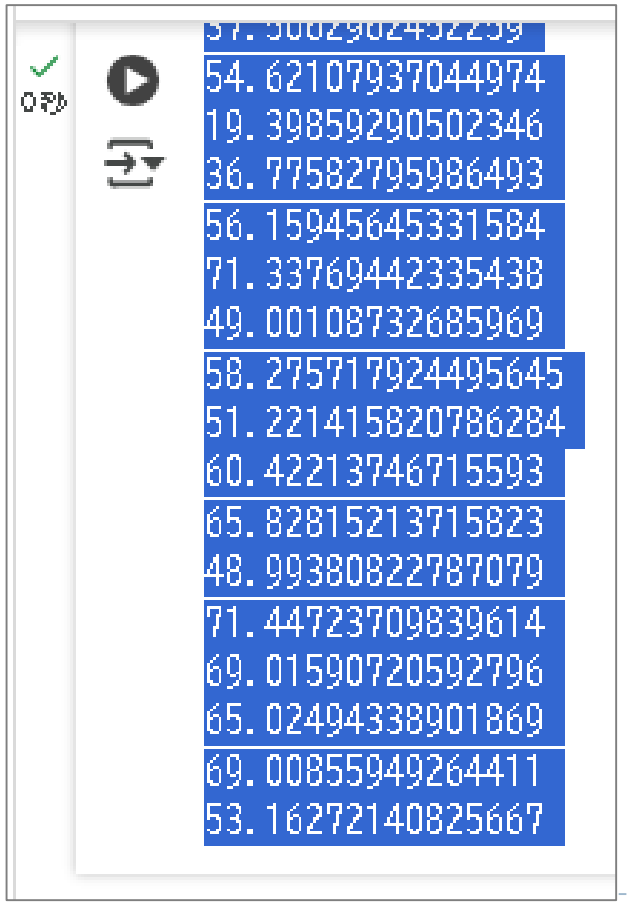


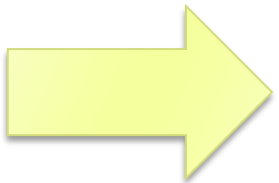
データの標本数を13ずつに増やす

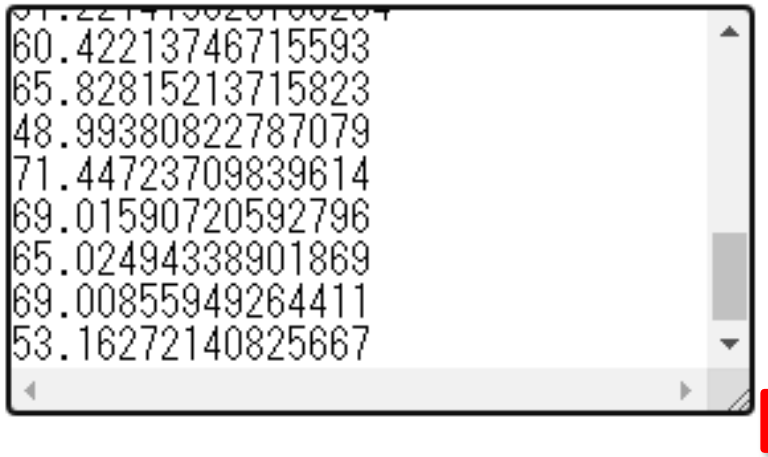


データを入力

- 先ほどの Google Colab のデータ
  26個をコピー＆ペーストし，「代入」クリック







ここをクリックし
下へ引き伸ばすと
データ入力エリアが
広がる

「代入」をクリック

「計算！」

- 「計算！」をクリックするとｔ検定が行われる

- Mean (平均),
  S.D. (標本標準偏差)
  t 値が表示される
- 結果
  - p<.01 (**)
  - **p値 0.01より小さい**
  → **有意な差がある**

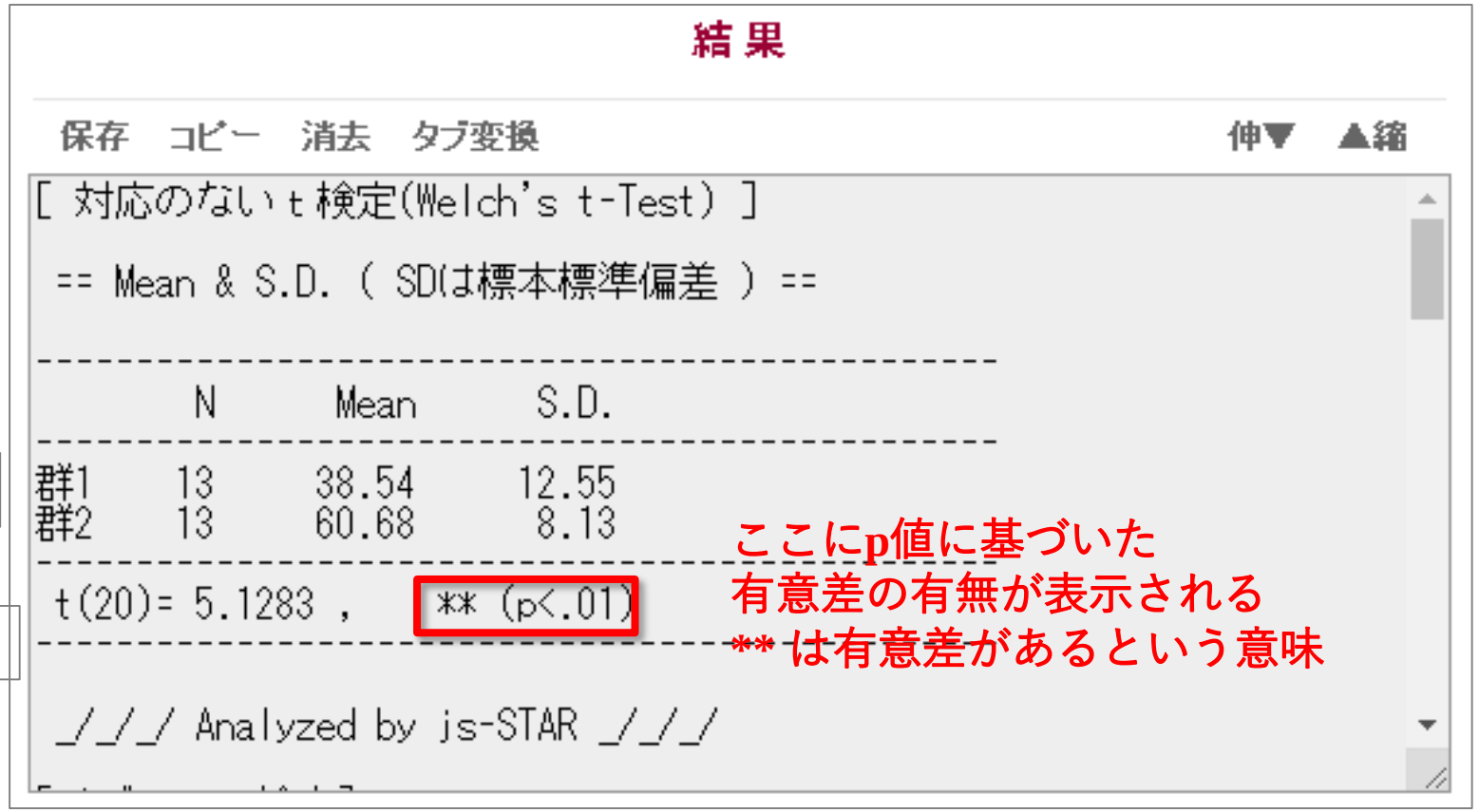
結果

- *, ** → 有意差有り
- ns → 有意差無し (not significant)
- + → 差がある傾向

p<0.05, p<0.01

p > 0.1

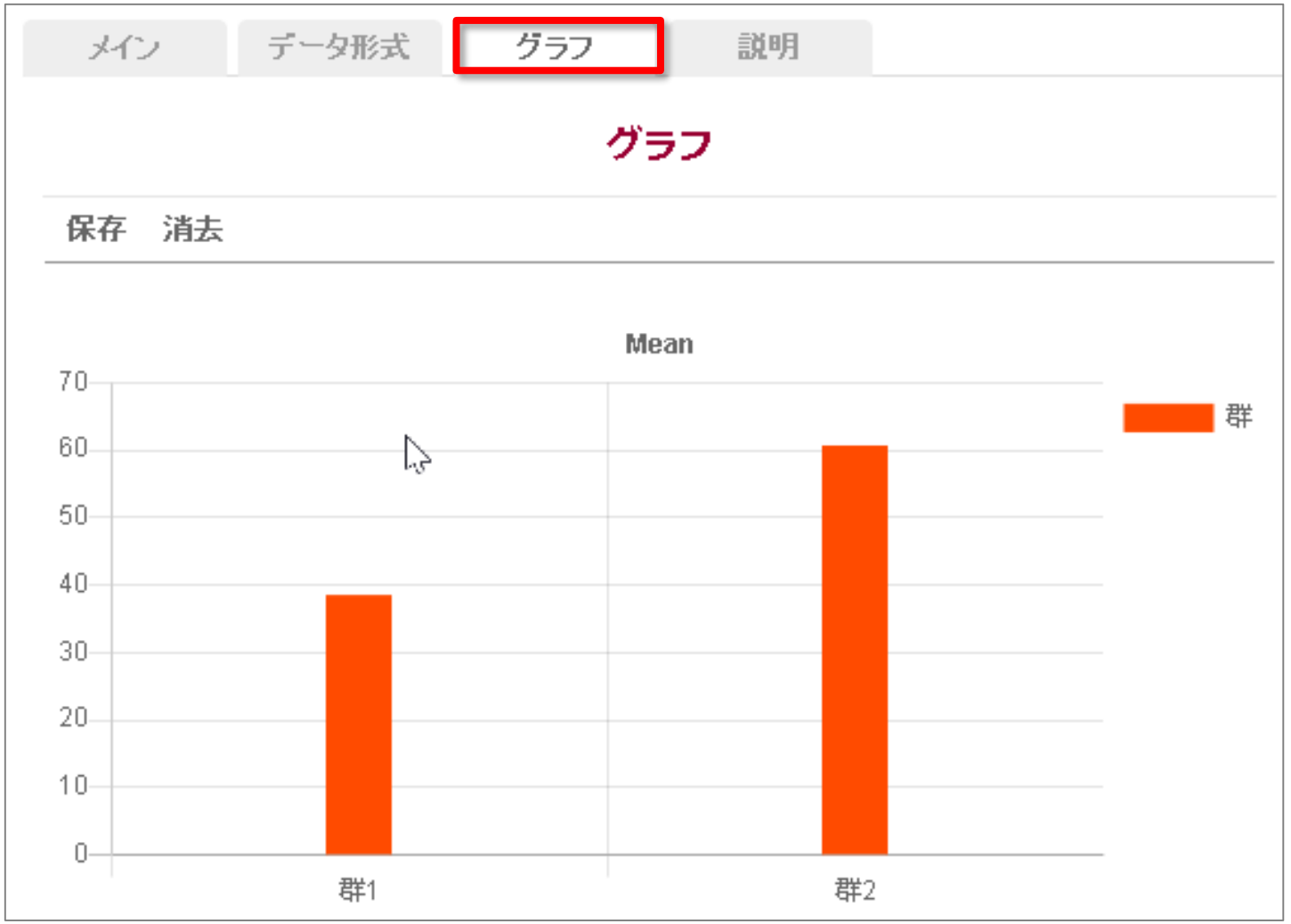
0.01 < p < 0.1



データのグラフを見る

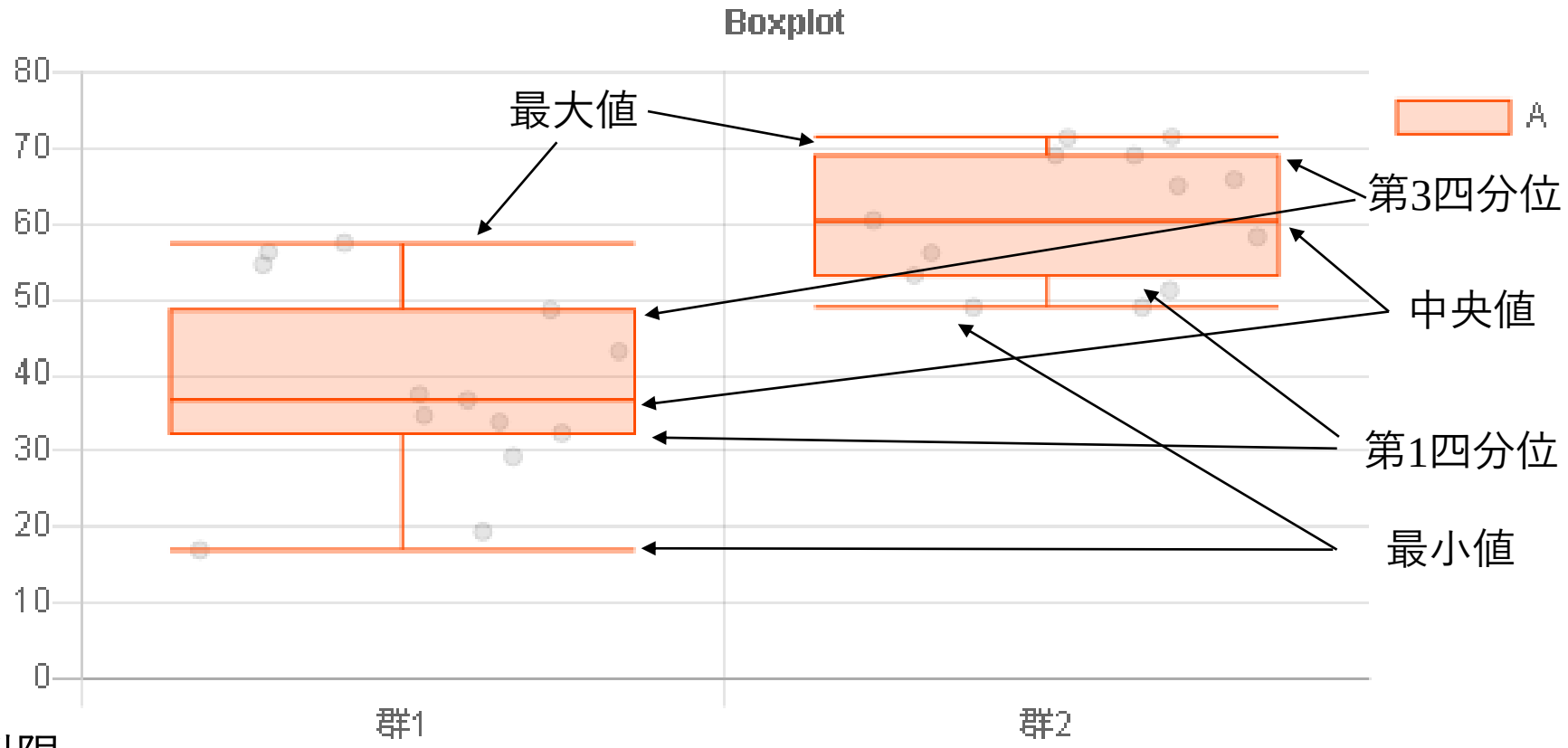
- 棒グラフはデータの大きさ（平均値）を比較する

- 誤差棒(エラーバー)により，標準誤差，標準偏差，信頼区間を表示することもできる
（データがどの程度ばらついているか，信頼できるか）



箱ひげ図

- 箱ひげ図は，正規分布でないデータ（ノンパラメトリック）も比較できる．上から，最大値，第３四分位数，中央値，第１四分位数，最小値を示す．



- データ点も重ねられる
  - 最大，最小よりも上，下にデータ点があれば外れ値
    - 箱の高さの1.5倍にひげの長さを制限

発展編

①この行削除

①乱数発生のシード（種）を削除してみる
- 実行ごとに異なる値(隣の人とも異なる)

②乱数発生の平均値を変えてみる (Python)
- 40 と 60 → それぞれ，40 と 50 に変更してみる

③標準偏差 (SD) を変えてみる
- 10 → 20

④データの数を変えてみる
- 13個のデータ → 100個，あるいは，10個

```python
import numpy as np

np.random.seed(seed=1) # 乱数のシード(種)

# 正規分布の乱数を13個取得
# 平均40，標準偏差10，サンプル数13
sample1 = np.random.normal(40, 10, 13)
# 平均60，標準偏差10，サンプル数13
sample2 = np.random.normal(60, 10, 13)

#縦に値を順番に表示
for i in range(len(sample1)):
  print(sample1[i])
for i in range(len(sample2)):
  print(sample2[i])
```

②この値を50に

③この値を20に

補足：コンピュータの乱数は疑似乱数で，シード(種)を固定すると同じ値が出力される
通常は種も適当に異なる値を割り当て，ランダムな値を得る

発展編で分かること

- ②平均値が離れているほど，有意差は出やすい
- ③ばらつき（標準偏差）が大きいほど，有意差は出にくい
- ④データの数が多いほど，有意差は出やすい

- ①乱数を毎回変えて実行すれば，p が約0.05なら，20回に1回は有意差が出ない(ns) 結果となる（母集団では，本当は有意差があるのに）

仮説検定のまとめ

- 仮説検定は，調べたいこと（見出したいこと）に対して，標本データから統計的に，その通りと言えるか（有意であるか）否かを確認する方法
- 何らかの主張をデータから結論づける際に，科学研究，経営戦略等，幅広く使われる
  - データサイエンスの中心的手法

- 例：
  - 2群の標本に差が有るのか → t 検定

回帰分析

統計分析

- データから統計的に現実世界や事象を説明するモデル（模型）を作る
- モデルを使って，未来や未知の場面で，未知のデータが入力されたときの出力を予測することができる
- 統計的な手法によるモデルを統計モデルと呼ぶ
- 回帰分析は代表的な統計モデル
  - 一次関数（直線）で未知を予測する

所持金額予測

- ある国のある店の前で道行く人に聞く
  「今財布にいくらありますか？」

1人目	2人目	3人目
40ドル	60ドル	50ドル

4人目?
ドル

見た目でおおよそ年齢が分かるので、それをヒントに予測

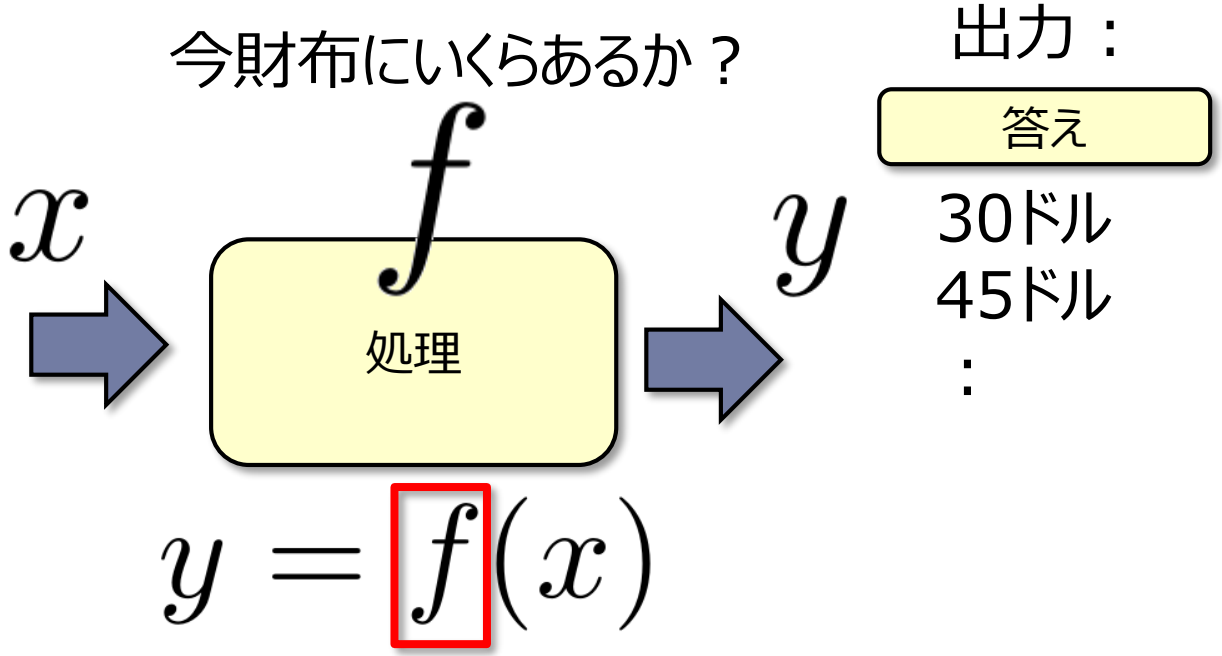
回帰分析

- 次に来る人の年齢から所持金を予測する**「関数」**を作る

入力：年齢　30歳
40歳
：

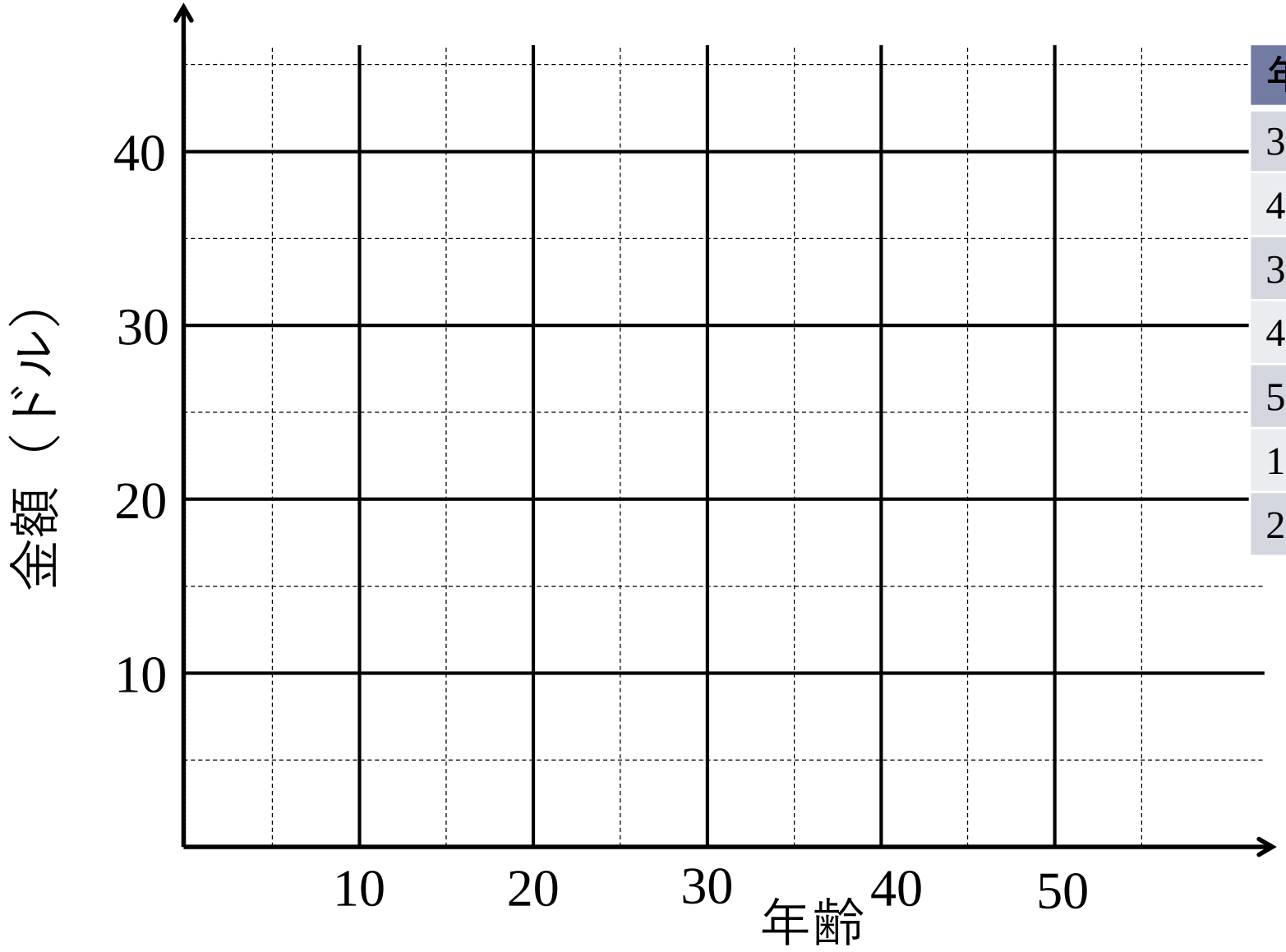
7人分のデータ

年齢	金額
30	30
40	35
30	40
40	30
50	45
10	10
20	15



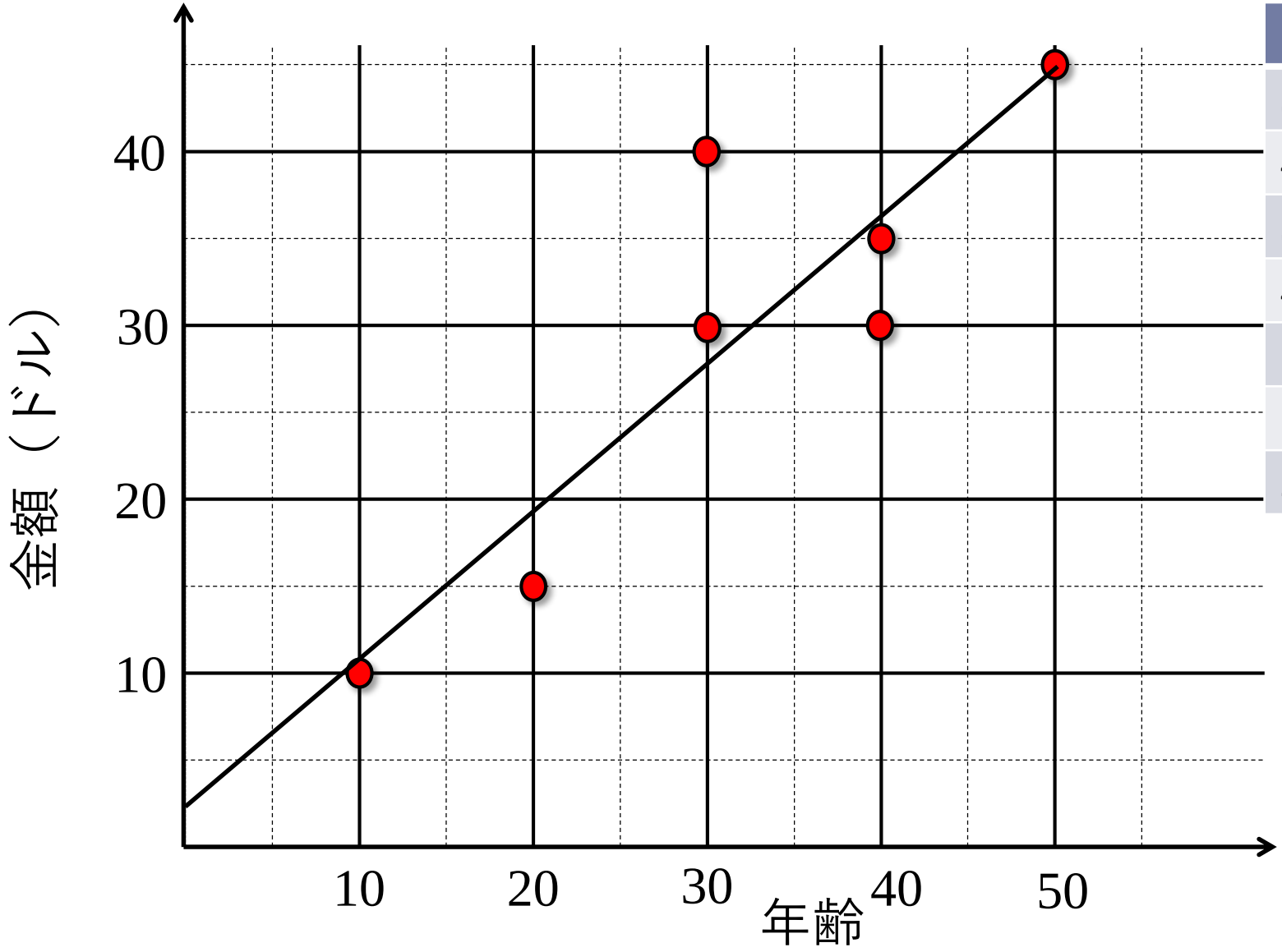
問題：
8人目(40歳)はいくら？

年齢と金額の関連を調べてみる → 散布図を描く



年齢	金額
30	30
40	35
30	40
40	30
50	45
10	10
20	15

散布図の各点のなるべく近くを通る直線を引く



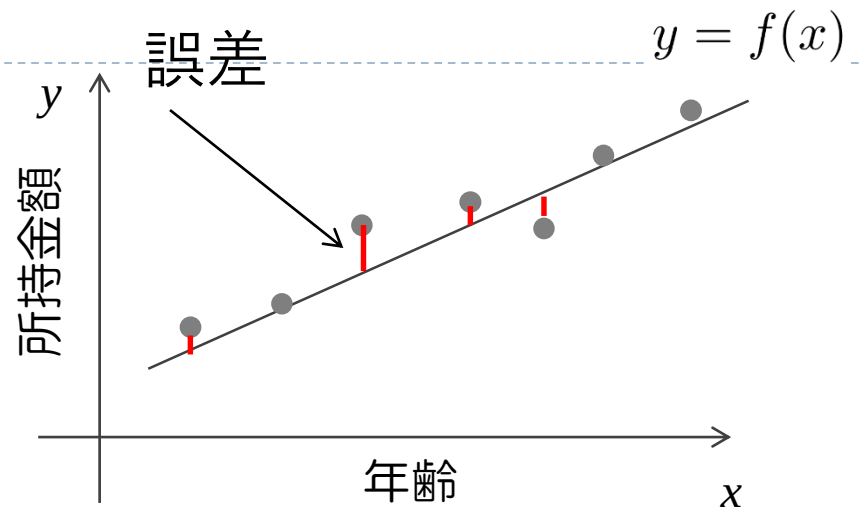
年齢	金額
30	30
40	35
30	40
40	30
50	45
10	10
20	15

直線の方程式

$$y = \frac{8}{10}x + 2.5\ ?$$

回帰分析

- 最も各点の近くを通る
  ＝点との誤差が最小
  の直線の方程式を求める



直線の方程式

傾き　切片

教師データ(金額) ⟶ $y = \boldsymbol{a}\,x + \boldsymbol{b}$ ⇨ これが関数

入力データ(年齢)

過去のデータ
x, y と誤差がなるべく少ない
$\boldsymbol{a}, \boldsymbol{b}$
を探す→ **予測モデル**となる

$\boldsymbol{a}$ と $\boldsymbol{b}$ を変化させると，
様々な直線に変化する
$\boldsymbol{a}, \boldsymbol{b}$ → **パラメータ**

回帰分析の演習

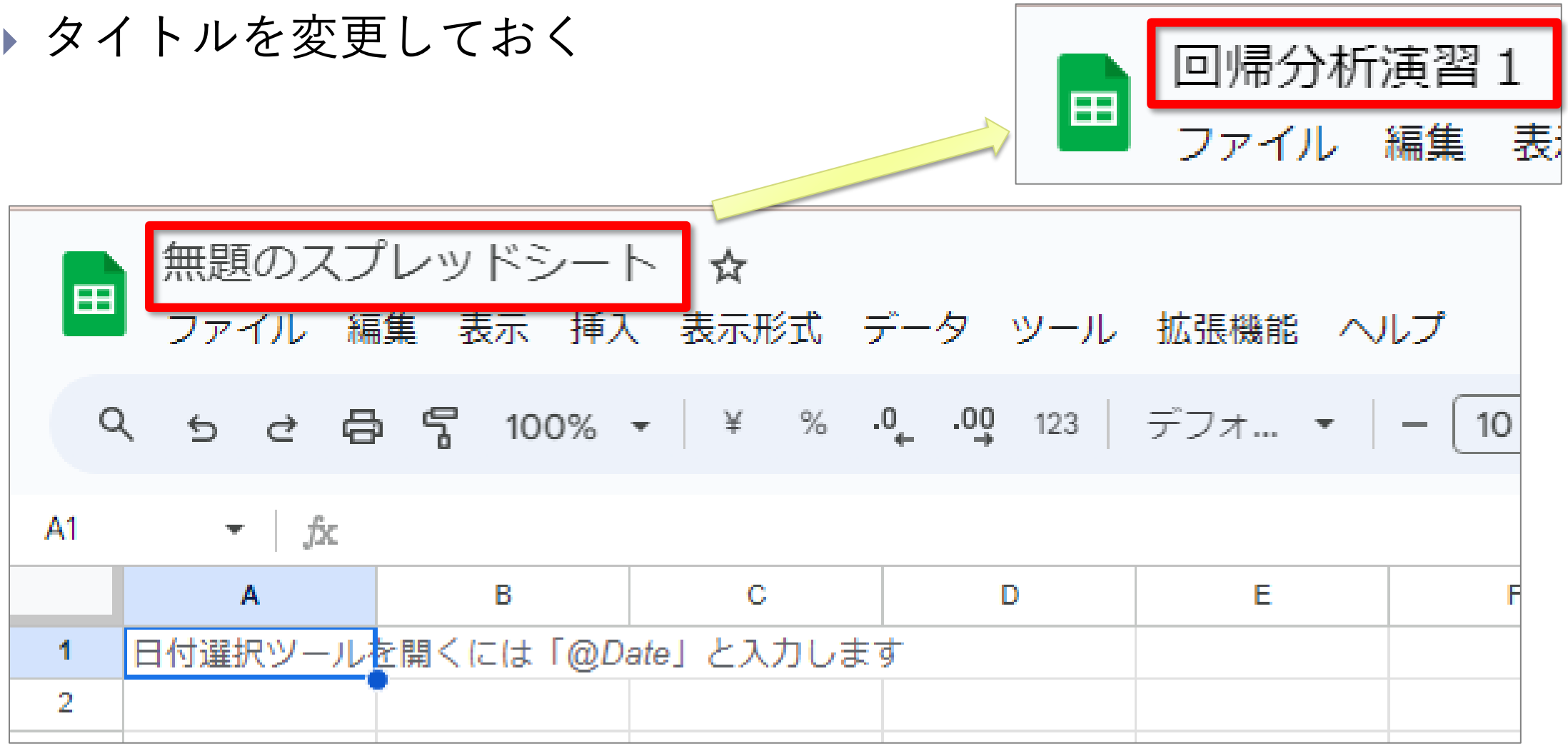
- コンピュータの解（最適な予測モデルである直線）を求める
- Google スプレッドシートを使う
- パソコンのブラウザから「Google Drive」へアクセス

回帰分析の演習

- 「+ 新規」でGoogleスプレッドシート → 空白のスプレッドシートを作成

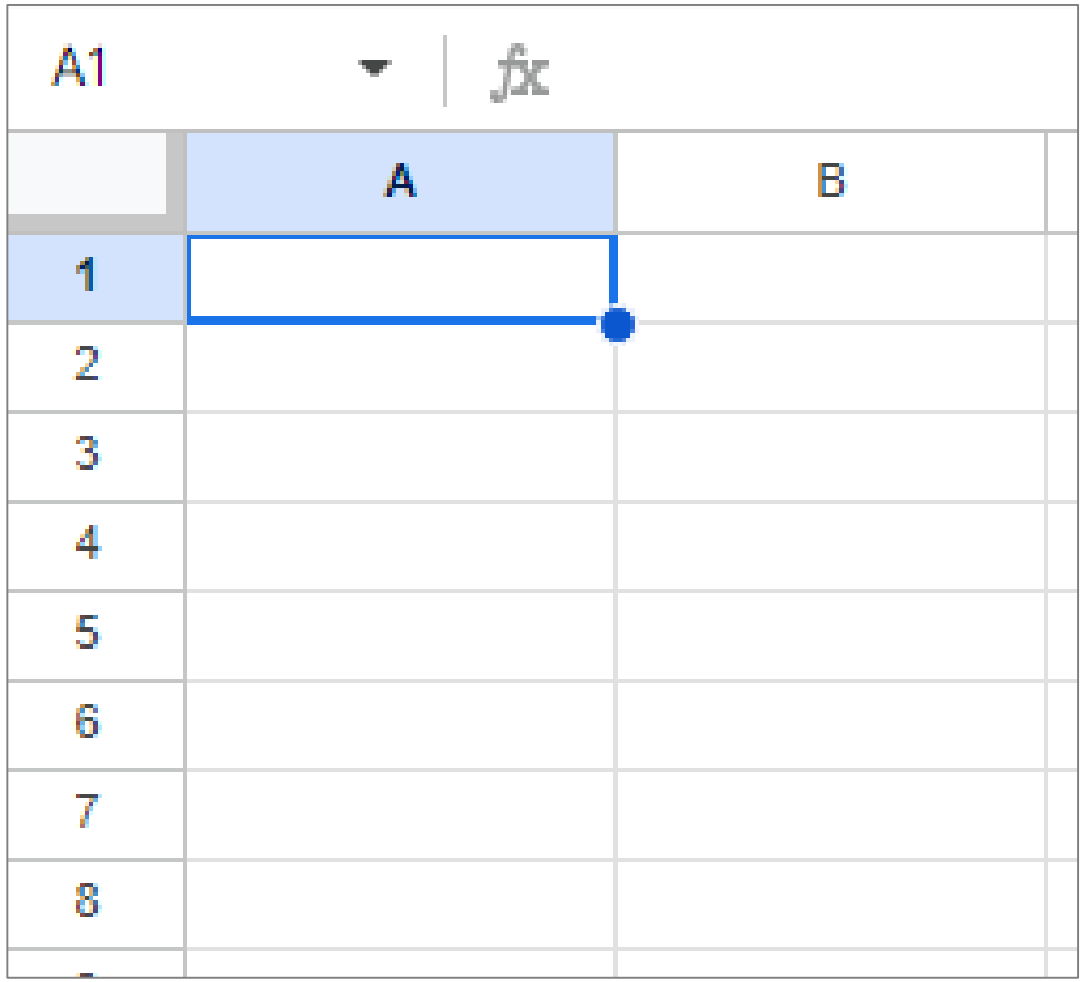
「無題のスプレッドシート」が開く

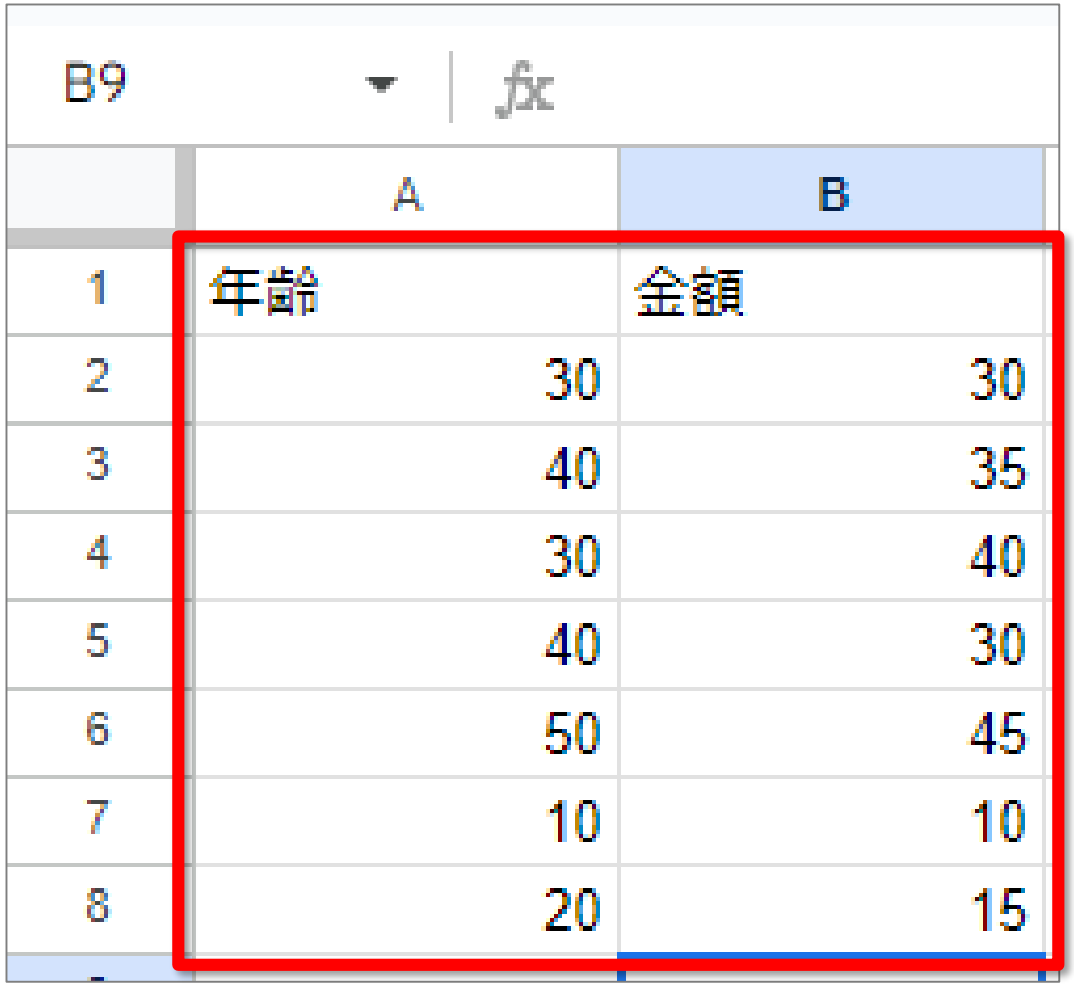
- タイトルを変更しておく



先ほどの7つのデータを入力する

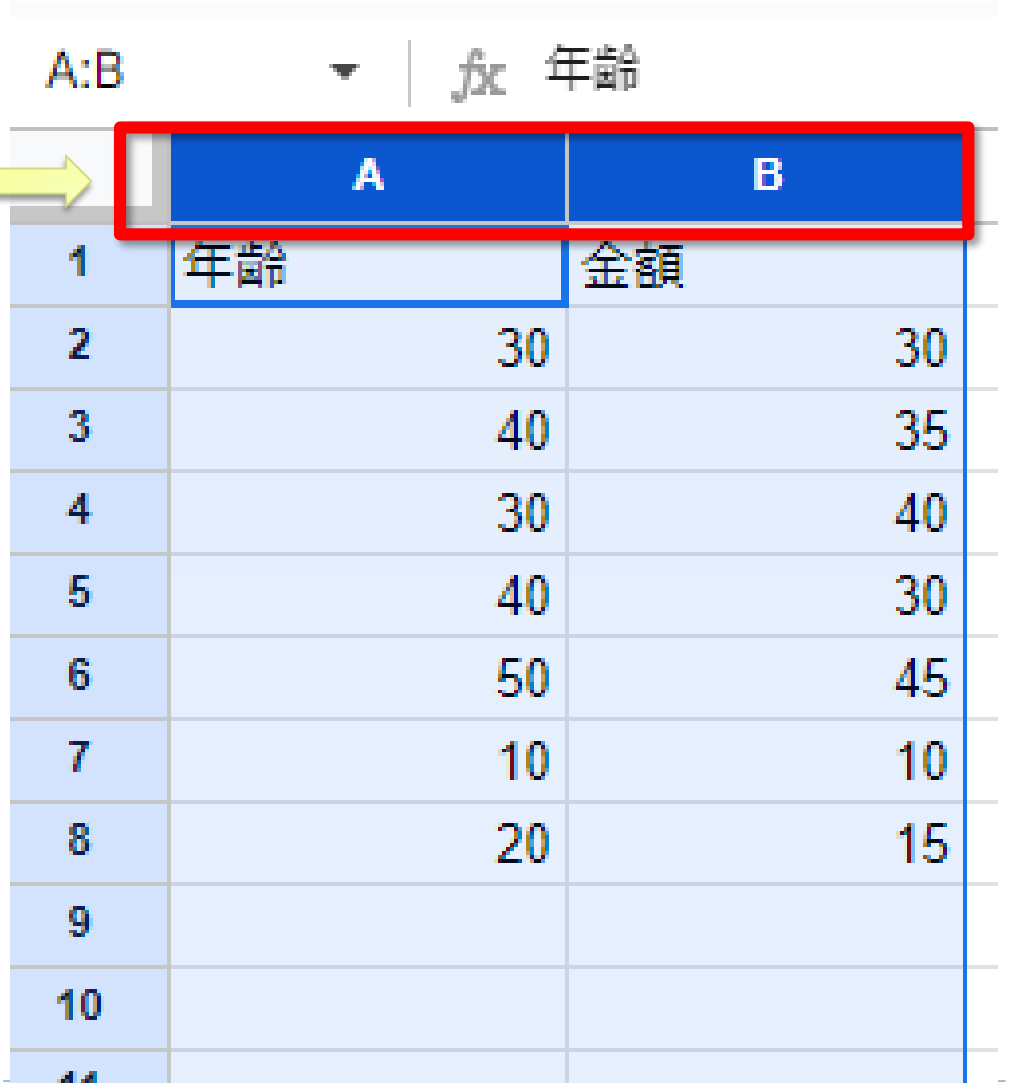
- 1行目には「年齢」，「金額」と入力





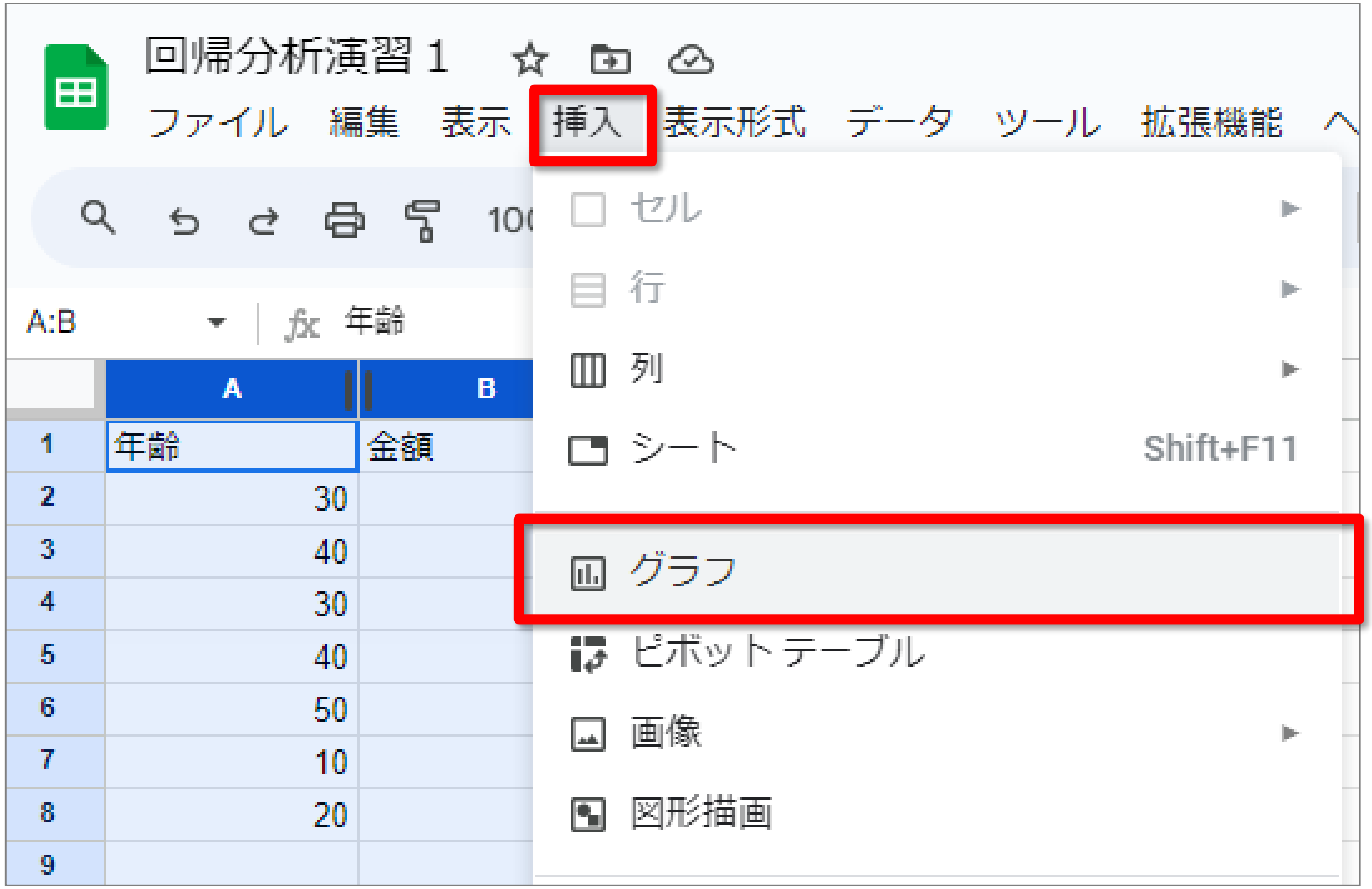
A列・B列を選択

マウスで「A」と「B」 の上を
Ctrl キーを押しながらクリック
（複数選択）

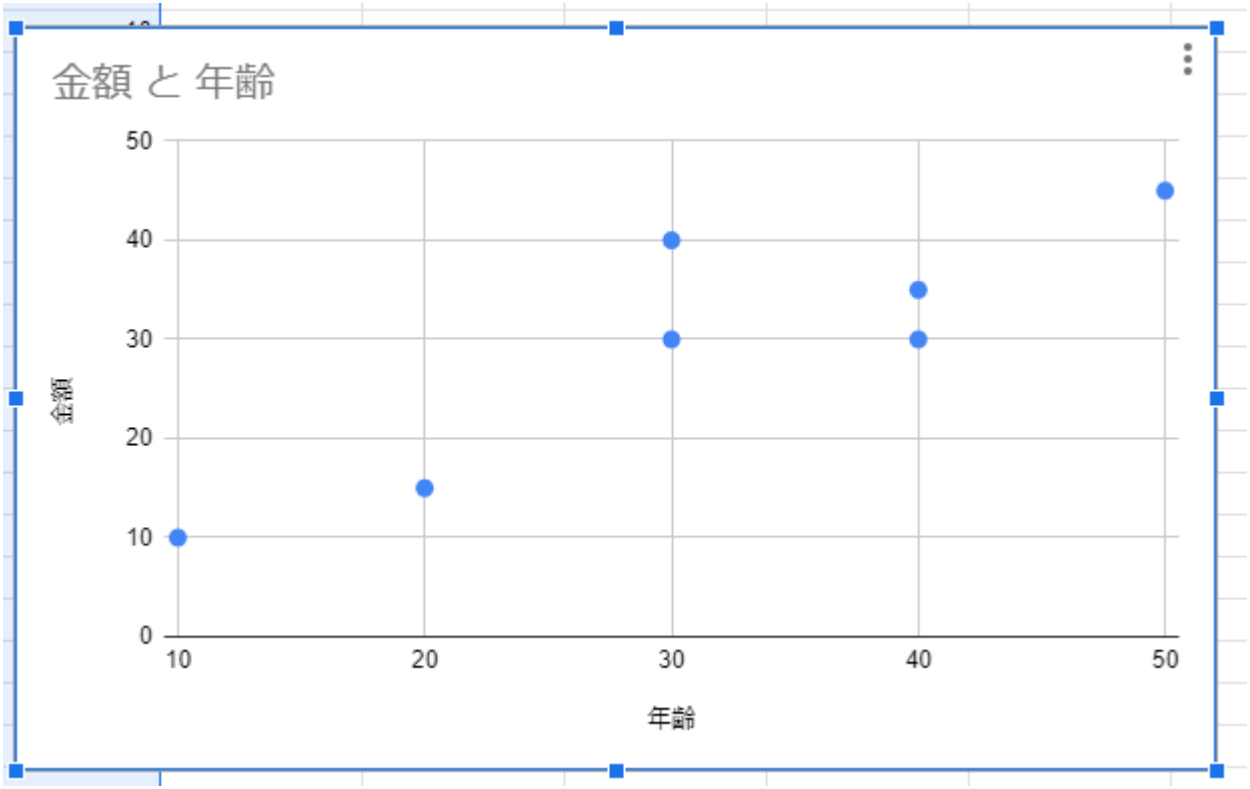


散布図を作成する

- 「挿入」
  →「グラフ」
  を選択



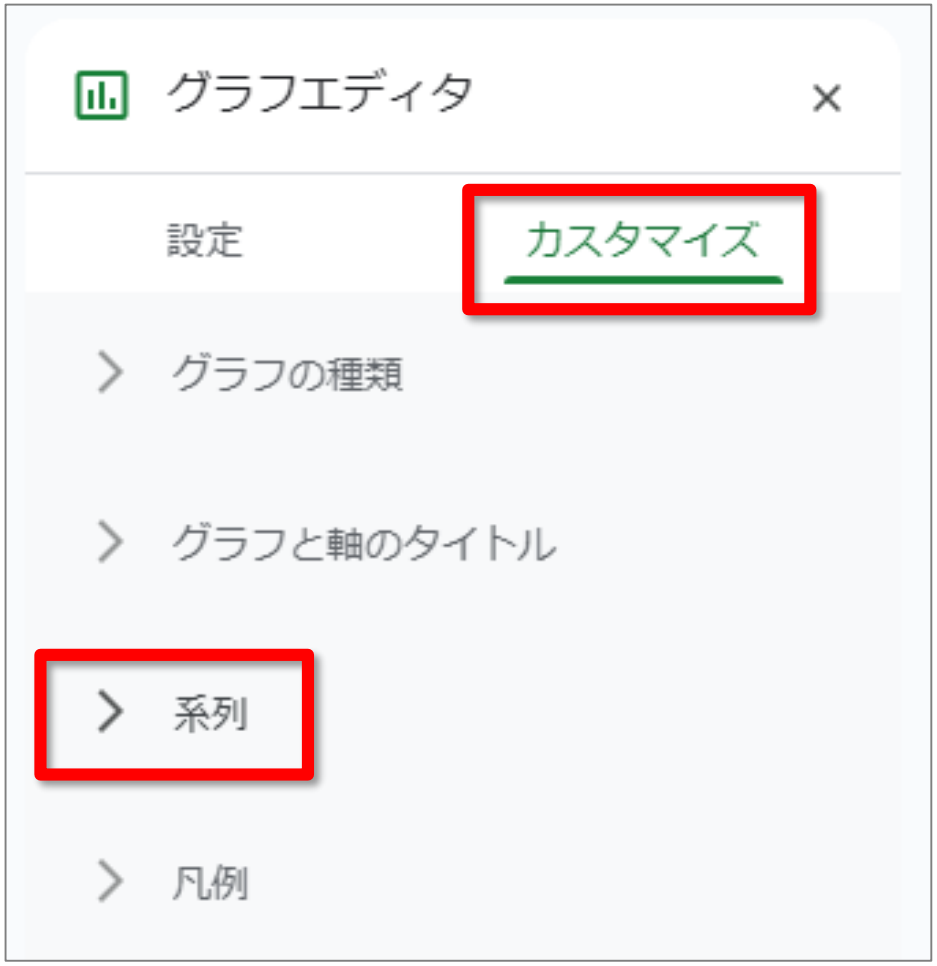
散布図を作成する



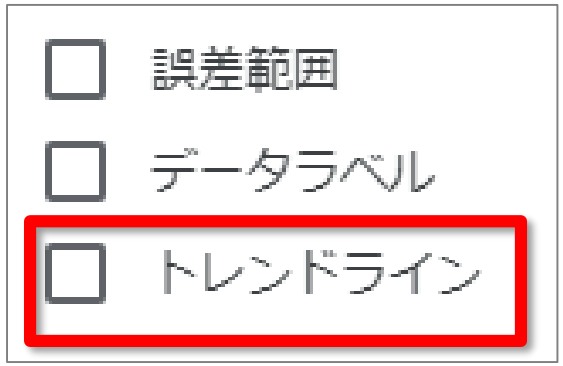
もし，左図のように散布図が作成されなければ，右のグラフエディタの「グラフの種類」から「散布図」を選択する

直線の方程式を求める

- グラフエディタから「カスタマイズ」→「系列」を選択



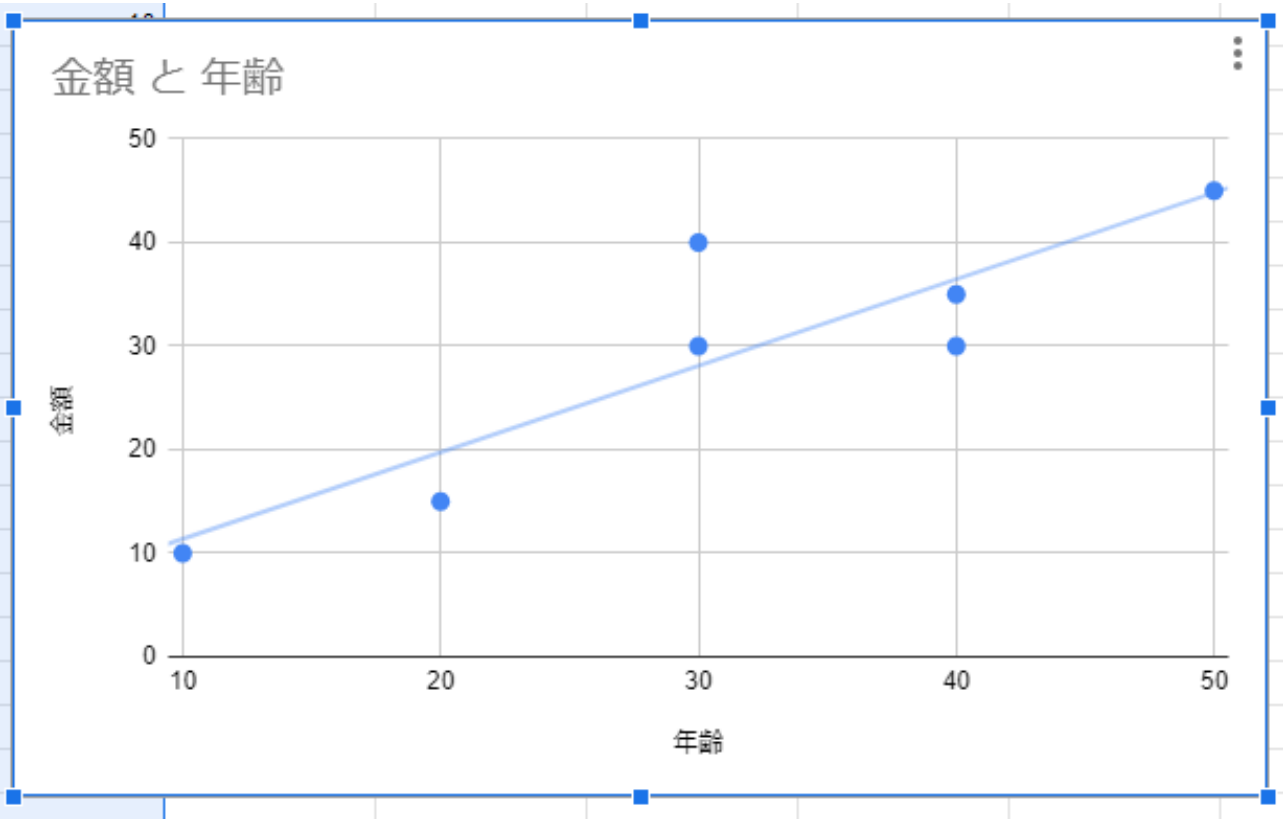
下へ
スクロール

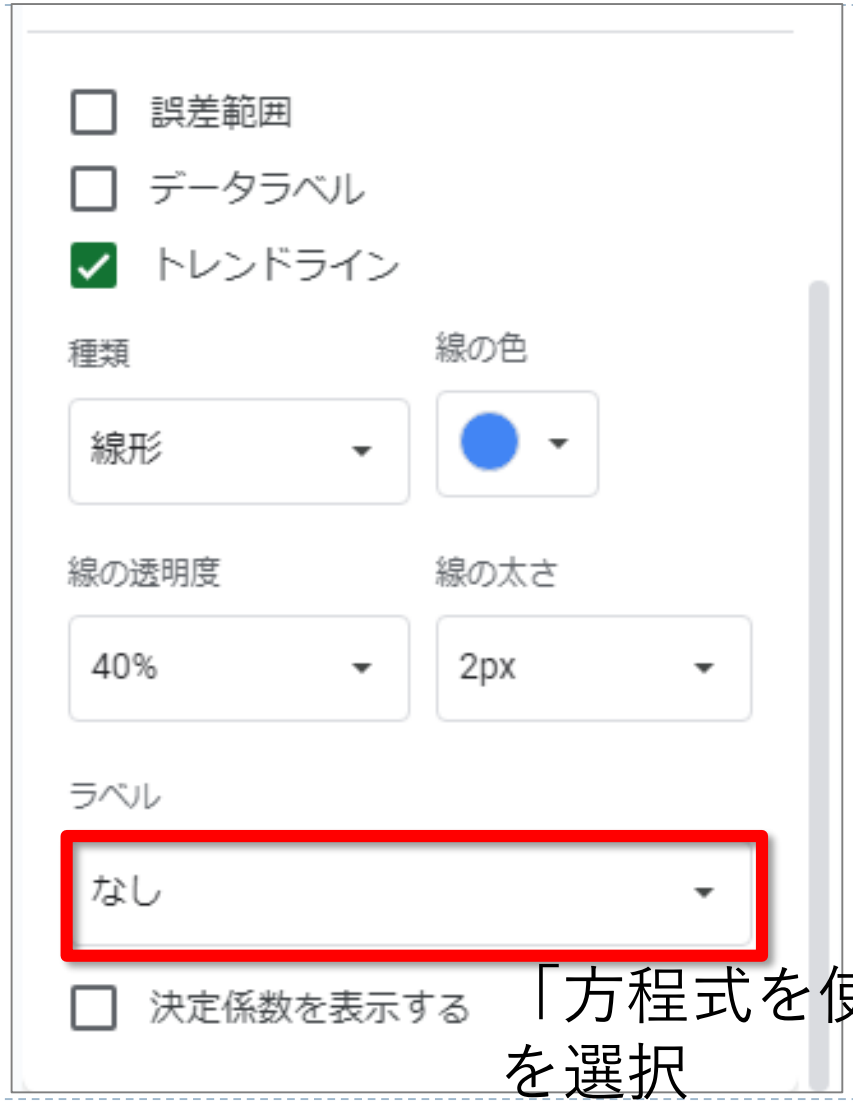


「トレンドライン」を選択

直線の方程式を表示

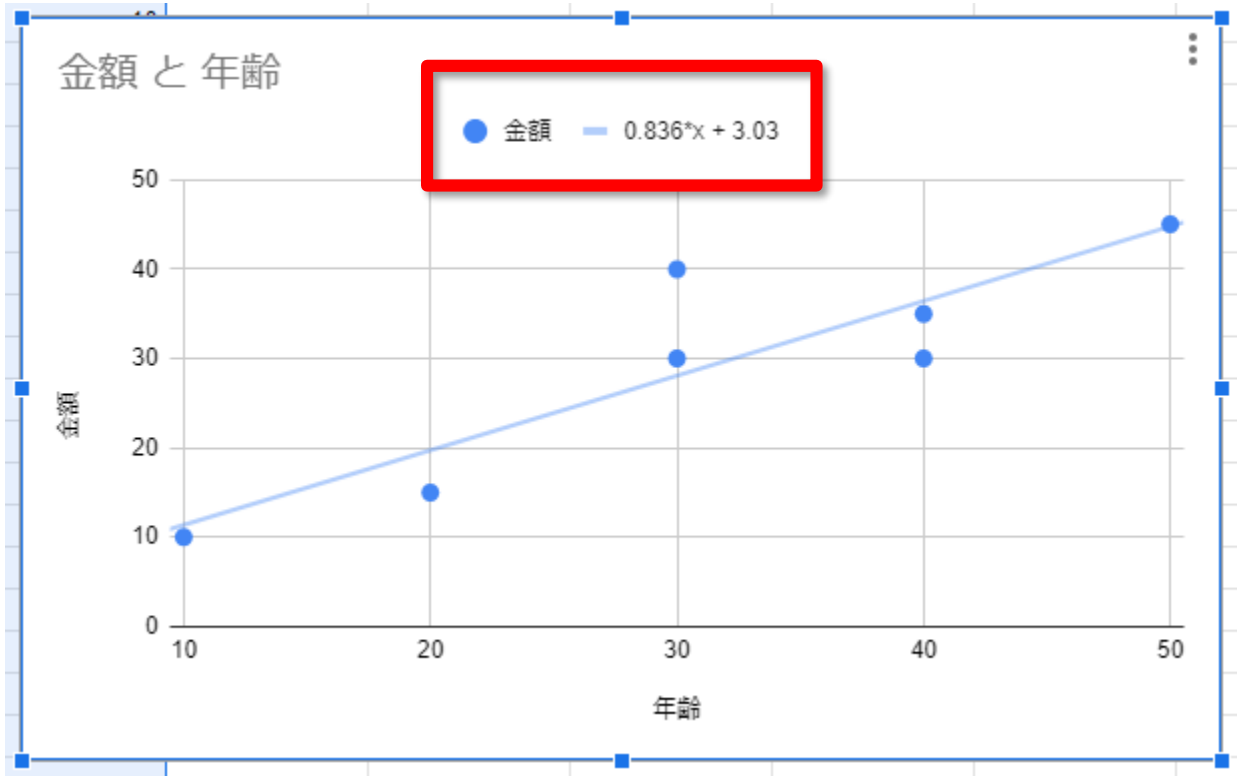
- 散布図に直線が引かれたことを確認



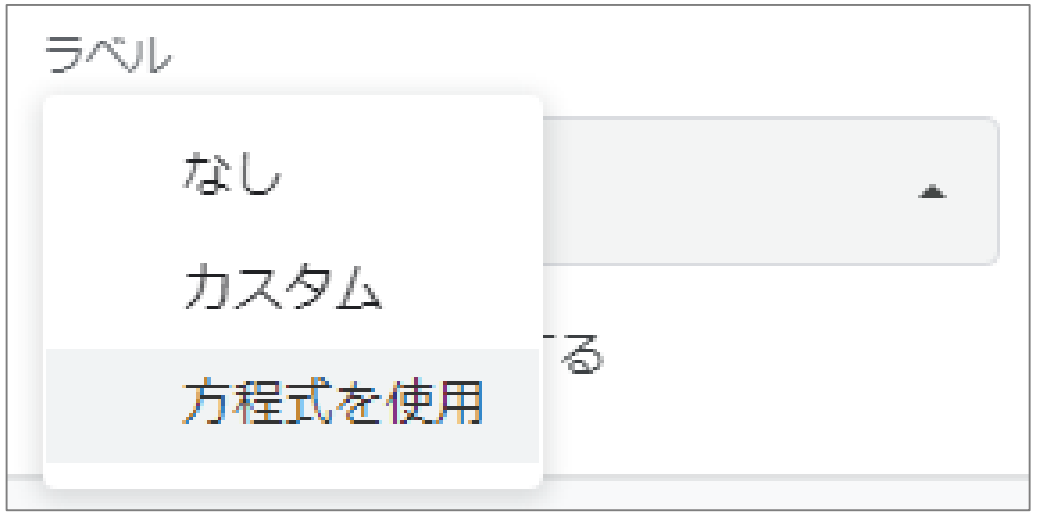


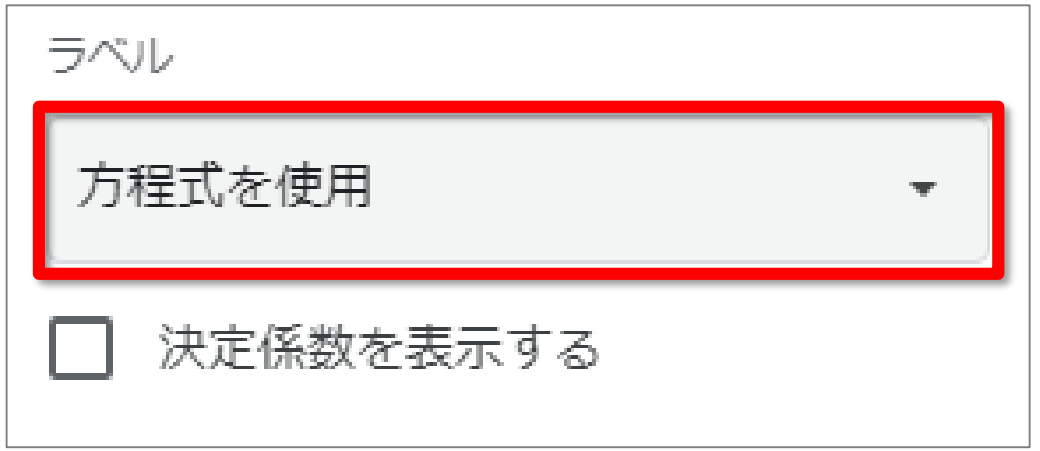
「方程式を使用」を選択

直線の方程式を表示



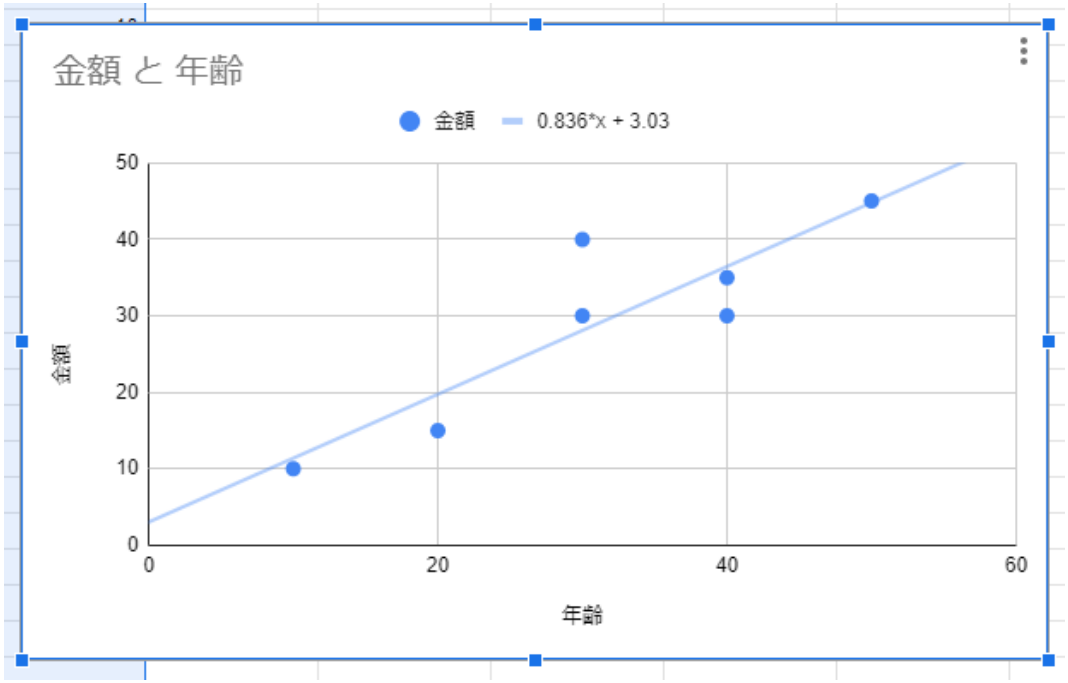
$$y = 0.836\,x + 3.03$$





グラフを調整

- 切片が見えるようグラフの x 軸(横軸)範囲を調整



【発展編】実際のデータ分析例

- 2019年新型コロナウイルス感染症
  COVID-19
- 各都道府県の「飲食・宿泊業の比率」から
  ➡「10万人あたりの感染者数」
  を予測

- 演習
  - 四国4県のデータは入っていない
  - 他の都道府県のデータから回帰分析によるモデルを構築し，それを使って**四国4県の「10万人あたりの感染者数」を「飲食・宿泊業の比率」から予測する**

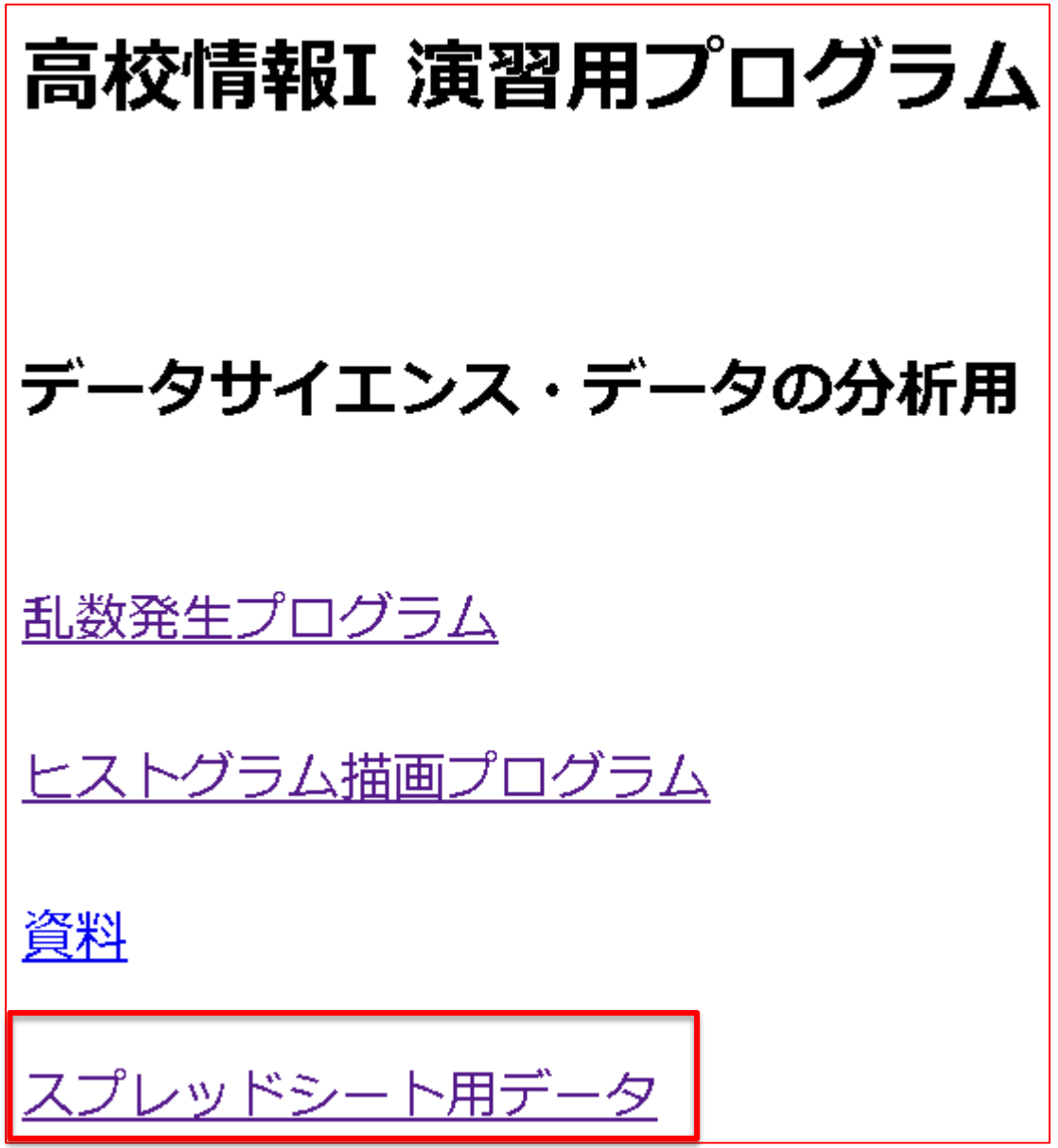
既存のデータから予測モデルを構築し，未知のデータに適用

⬇

機械学習(現代のAI)の基本的な考え方

COVID-19 演習

- Google スプレッドシートを使う
- Google スレッドシートに
  データ読み込み
- 「スプレッドシート用データ」
  のリンクをクリック



データ

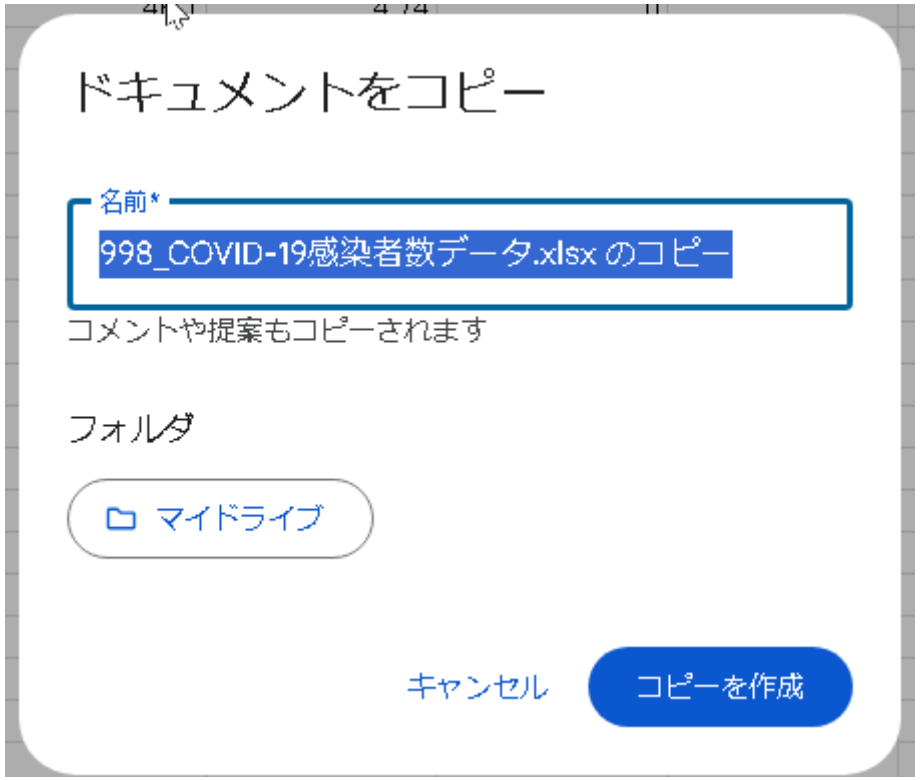
- 都道府県データと10万人あたり感染者数

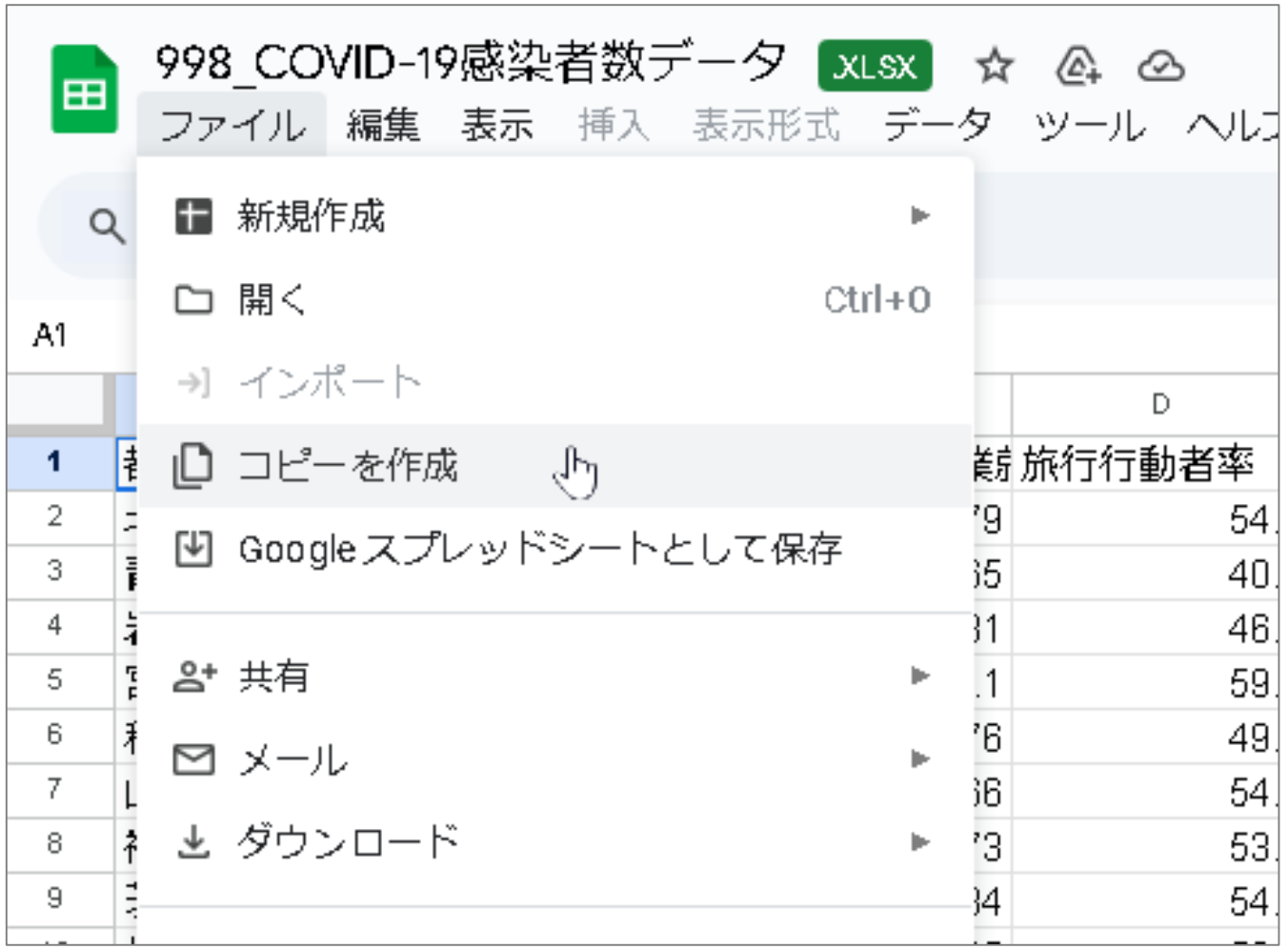
都道府県	就業者比率	飲食店・宿泊業	旅行行動者率	外国人宿泊者比	人口10万人当たり感染者率
北海道	45.25	5.79	54.6	25.29	24.12
青森県	47.85	4.65	40.6	8.01	2.14
岩手県	49.73	4.81	46.1	4.74	0
宮城県	46.19	5.1	59.7	4.21	4.06
秋田県	47.2	4.76	49.2	3.76	1.63
山形県	50.01	4.66	54.4	3.07	6.33
福島県	48.18	4.73	53.6	1.71	4.4
茨城県	48.02	4.34	54.6	4.24	6.05
栃木県	48.83	5.05	53.5	3.15	3.91
群馬県	48.96	5.02	54.8	3.92	7.79
埼玉県	47.95	4.52	65.7	3.93	15.89
千葉県	46.28	5.06	64.8	17.22	15.19
東京都	43.35	6.35	69.2	36.48	46.34
神奈川県	45.16	5.23	66.5	13.4	16.67

下記記事を参考にデータを作成
https://qiita.com/y_itoh/items/b6254ca2acf8b875b593
政府統計ポータルサイト「政府統計の総合窓口e-Stat」都道府県データ
https://www.e-stat.go.jp/regional-statistics/ssdsview/prefectures
厚生労働省「新型コロナウイルス感染症」各都道府県の検査陽性者の状況
https://www.mhlw.go.jp/content/10906000/000646813.pdf

自分の「マイドライブ」にコピー

- 「ファイル」→「コピーを作成」→「コピーを作成」(マイドライブ)





回帰分析 (直線)

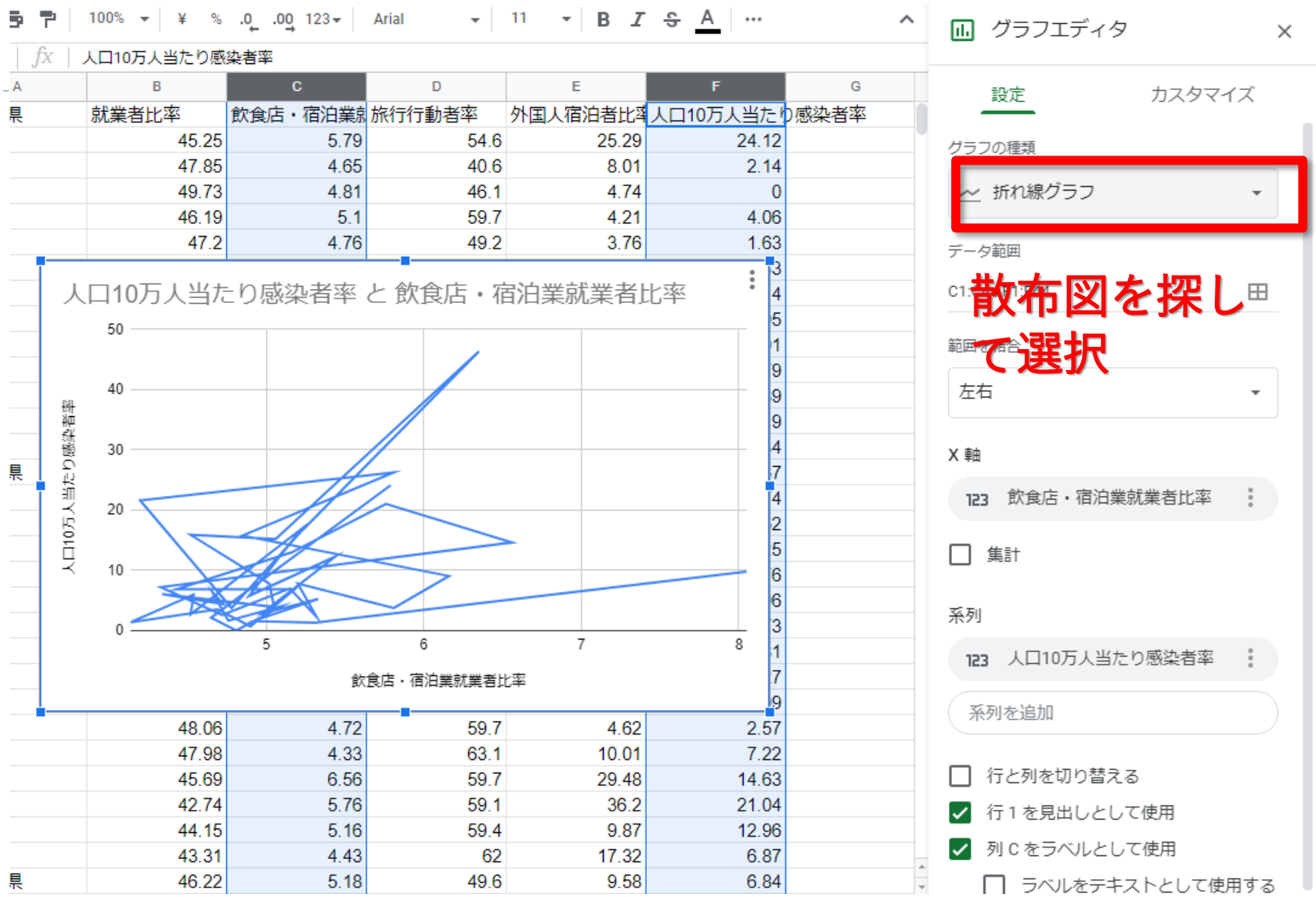
- C列とF列を選択
  - 「C」をクリックし，コントロール[Ctrl]を押しながら「F」をクリック

	A	B	C	D	E	F	G
	都道府県	就業者比率	飲食店・宿泊業	旅行行動者率	外国人宿泊者比	人口10万人当たり感染者率	
	北海道	45.25	5.79	54.6	25.29	24.12	
	青森県	47.85	4.65	40.6	8.01	2.14	
	岩手県	49.73	4.81	46.1	4.74	0	
	宮城県	46.19	5.1	59.7	4.21	4.06	
	秋田県	47.2	4.76	49.2	3.76	1.63	
	山形県	50.01	4.66	54.4	3.07	6.33	
	福島県	48.18	4.73	53.6	1.71	4.4	
	茨城県	48.02	4.34	54.6	4.24	6.05	
	栃木県	48.83	5.05	53.5	3.15	3.91	
	群馬県	48.96	5.02	54.8	3.92	7.79	

散布図作成

- 「挿入」→「グラフ」をクリック

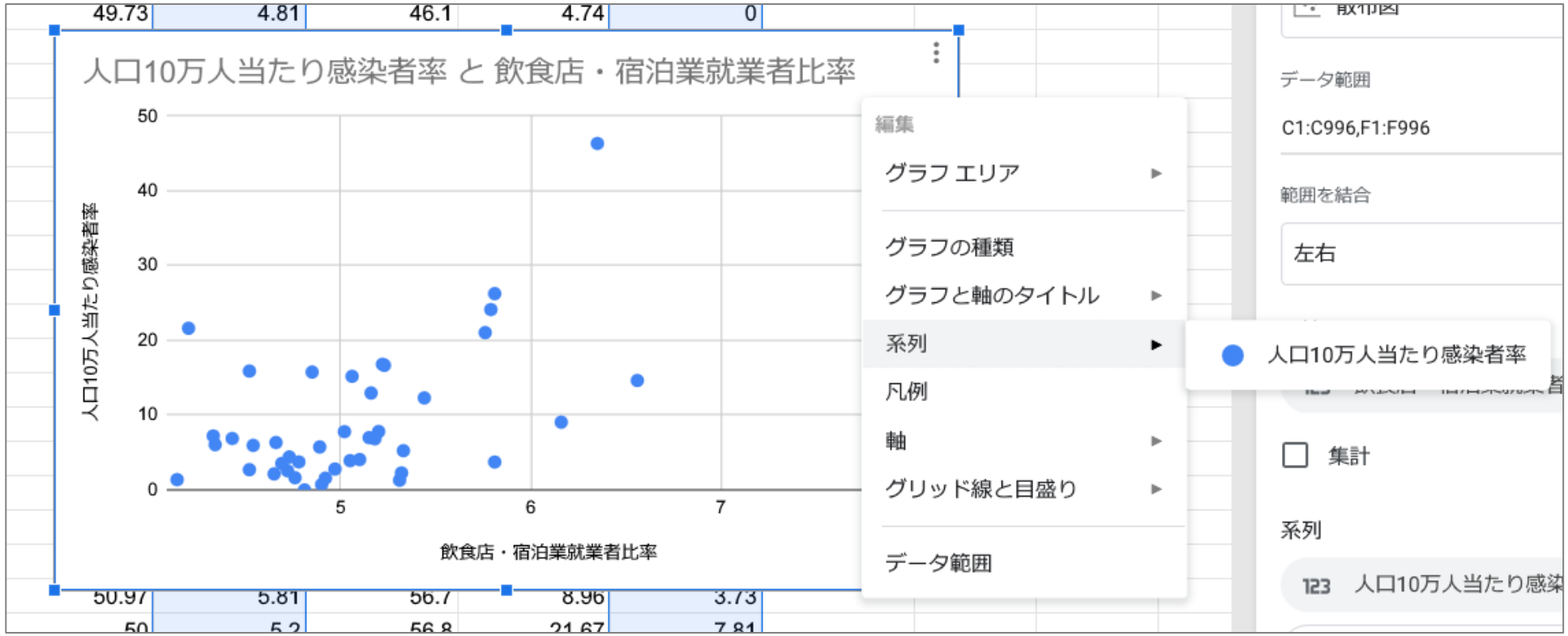
散布図を作成



散布図を探して選択

直線を求める

- グラフを右クリック
  「系列」→「人口10万人当たり感染者率」



直線を求める

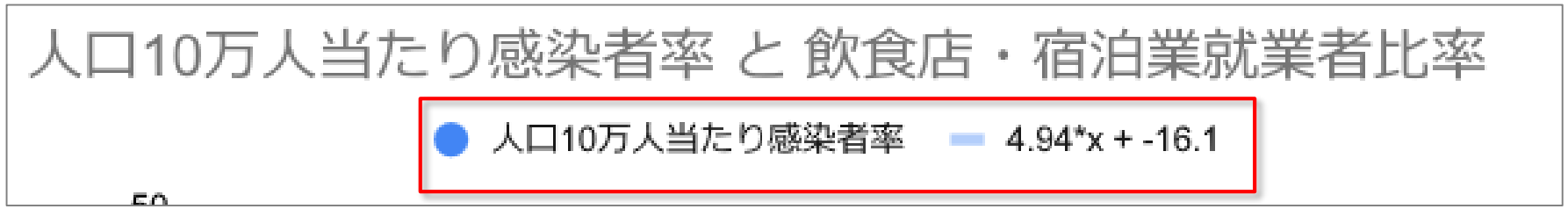
- 下の方,

  トレンドライン
  をチェック

  ラベルで,
  「方程式を使用」
  を選択

直線の方程式

- 回帰か直線

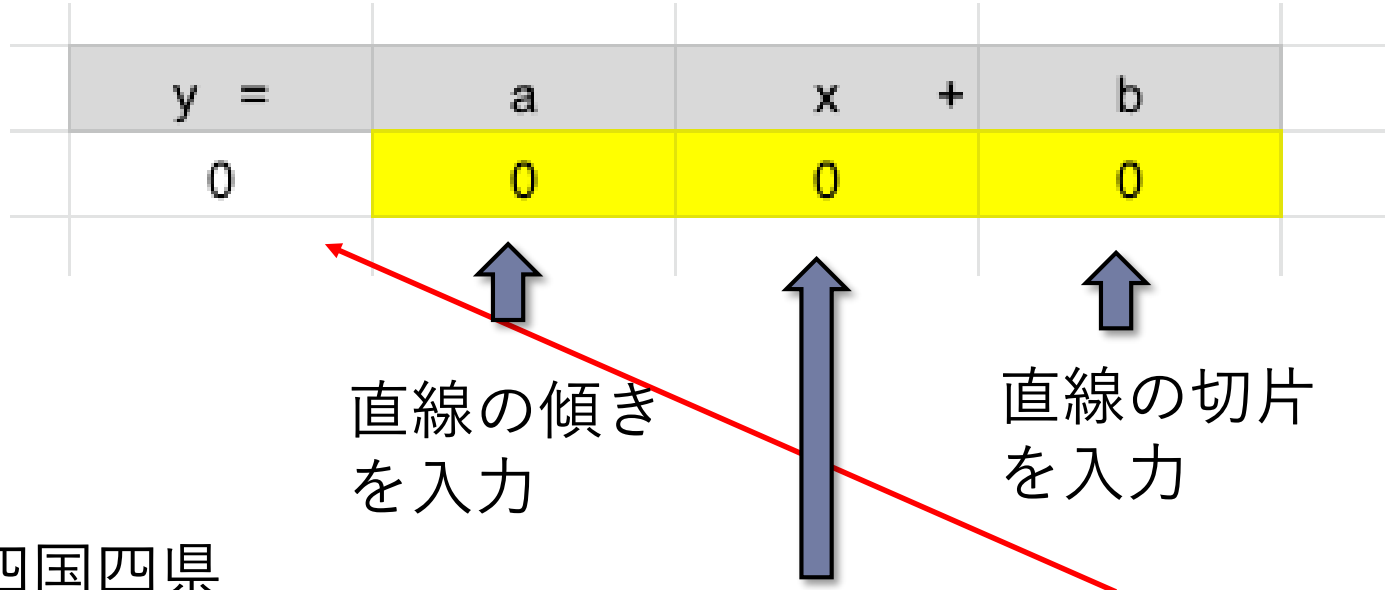


予測をしてみる

操作

① a の下の黄色のセル(I26) に,
回帰直線の傾き 4.94 を入力
② b の下の黄色のセル(K26) に,
回帰直線の切片 -16.1 を入力
③ x の下の黄色のセル(J26) に,
高知県の飲食店・宿泊業の比率
5.37 を入力
④ 計算された y の値 (H26) を確かめる
実際の高知県の
10万人あたりの感染者数 10.48
とどのくらい離れているか？

- 表の黄色のセルI26～K26



四国四県

県	飲食店・宿泊業比率	10万人あたり感染者数
徳島県	4.22	0.82
香川県	4.5	2.91
愛媛県	4.42	6.07
高知県	5.37	10.48

回帰分析のまとめ

- 未来や未知のことを予測するモデル
- データから統計的に作成される
  - 一次関数（直線）を使う

- 代表的な統計モデルで，医学，薬学，経済・経営，心理学，農学その他，幅広く使われている

- AI (人工知能) の予測も，考え方は回帰分析の延長であり，大量の過去のデータに基づいて，複雑な（直線ではない）関数を構築し，予測を行う