Mixtape Chapter 1: Probability and Regression Review

Introduction of Python

Hiroki Kato

2021/06/24

Probability Review

いくつかの確率の概念

- 確率過程（**random process**）：毎回異なるアウトカムが何度も繰り返されるプロセス
- 標本空間（**sample space**）：確率過程で得られるすべてのアウトカムの集合
  - e.g.) ガソリン価格：$\Omega = \{p : p \geq 0\}$
  - e.g.) 12 面サイコロ：$\Omega = \{1, 2, 3, \ldots, 11, 12\}$

独立（independence）の定義

1. logical independence：二つのイベントが生じたが、互いに影響を与えると考えられる理由がない
   - 前後即因果の誤謬（post hoc ergo propter hoc）：独立であるにも関わらず、時間の前後関係から二つのイベントは影響しあっていると考えてしまうもの
2. statistical independece

統計的独立の二つの定義

1. $P(A|B) = P(A) \leftrightarrow$ 二つのイベント（A, B）は統計的に独立である
   - sample with/without replacement の例
   - ランダムシャッフルしたトランプから一枚引いたとき、エースである（A）確率は $P(A) = 4/52$
   - 一枚目のカードがエース（B）であり、もう一度カードを一枚引いた（sample w/o replacement）とき、エースである（A）確率は $P(\text{Ace}|\text{Card 1} = \text{Ace}) = 3/51$
   - 一枚目のカードがエース（B）であり、エースを戻してもう一度カードを一枚引いた（sample w/ replacement）とき、エースである（A）確率は $P(\text{Ace}|\text{Card 1} = \text{Ace}) = 4/52$
2. $P(A, B) = P(A)P(B) \leftrightarrow$ 二つのイベント（A, B）は統計的に独立である
   - テキサスポーカーの例（自分にしか分からない二枚のカードと公開されている三枚のカードで役を競う）
   - 自分にしか分からないカード（poket cards）が同じ数字（two of a kind）である確率 $\rightarrow 4/52 \times 3/51 = 0.0045$

イベントに関する Notation の整理

あるイベントを A とし、他のイベントを B とする

- A and B: Both A and B occur
- $\sim A$ and B: A does not occur, but B occurs
- A and $\sim B$: A occurs, but B does not occur
- $\sim A$ and $\sim B$: Neither A and B occurs

樹形図（Probability tree）

樹形図の特徴

あるノードから出発するすべてのブランチについて、

- あるブランチの確率を合計すると 1 になる
- 同時確率を合計すると出発地点のノードに達する確率と一致する（**the law of total probability**）：$P(A) = \sum_n P(A \cap B_n)$
- 筆記試験に合格したことを条件としたとき、運転試験に不合格となる確率は $P(\mathsf{Fail}|\mathsf{Pass}) = 0.45/0.75 = 0.6$（条件付き確率の概念）

ベン図（Venn diagram）

集合の性質

U is the universal set of which A and B are subsets

- $A+ \sim A = B+ \sim B = U$
- $A = B+ \sim B- \sim A$ and $B = A+ \sim A- \sim B$
- $A = A \cap B + A \cap \sim B \rightarrow A \cap \sim B = A - A \cap B$
- $A \cup B = A \cap \sim B+ \sim A \cap B + A \cap B$

テキサス大学のフットボールチームのコーチの危機 (1)

Longhorns（テキサス大学のフットボールチーム）がボールゲームに招待されないと、コーチは解雇される可能性が高い。

- イベント A：Longhorns がボールゲームに招待される（$P(A) = 0.6$）
- イベント B：コーチが雇用継続される（$P(B) = 0.8$）
- 二つのイベントが同時に生じる確率は $P(A, B) = 0.5$

このとき、Longhorn がボールゲームに招待され、コーチが雇用継続されないというイベントが同時に生じる確率は $P(A, \sim B) = P(A) - P(A, B) = 0.1$

テキサス大学のフットボールチームのコーチの危機 (2)

確率はある集合を形成する部分集合が集合に占める割合で計算される

- イベント A が生じた世界でイベント B が生じる確率は
  $A \cap B/A = 0.5/0.6 = 0.83$
- イベント B が生じた世界でイベント A が生じる確率は
  $P(A|B) = P(A, B)/P(B) = 0.5/0.8 = 0.63$

分割表（Contingency tables）

Event labels	Coach is not rehired ($\sim B$)	Coach is rehired (B)	Total
(A) Bowl game	$P(A, \sim B) = 0.1$	$P(A, B) = 0.5$	$P(A) = 0.6$
($\sim A$) no Bowl game	$P(\sim A, \sim B) = 0.1$	$P(\sim A, B) = 0.3$	$P(\sim A) = 0.4$
Total	$P(\sim B) = 0.2$	$P(B) = 0.8$	$P(U) = 1$

条件付き確率の定義

- $P(A|B) = P(A,B)/P(B) \rightarrow P(A,B) = P(A|B)P(B)$
- $P(B|A) = P(B,A)/P(A) \rightarrow P(B,A) = P(B|A)P(A)$

Naive version of Bayes' rule

$$P(A|B) = \frac{P(A)}{P(B)} P(B|A)$$

- これは $P(A|B)P(B) = P(A,B) = P(B,A) = P(B|A)P(A)$ で得られる
- このテキストは原因から結果を推定するが、ベイズの法則は結果から原因に関する合理的な信念を作れることも伝えている

Bayesian decomposition version of Bayes' rule

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B| \sim A)P(\sim A)}$$

- これは $P(B) = P(A, B) + P(\sim A, B)$、
  $P(\sim A, B) = P(B, \sim A) = P(B| \sim A)P(\sim A)$ で得られる
- テキサス大学のコーチが雇用継続されたならば、Longhorn がボールゲームに招待される確率は
  - $P(A|B) = (0.83 \cdot 0.6)/(0.83 \cdot 0.6 + 0.75 \cdot 0.4) = 0.624$

モンティホール・ジレンマ (1)

- 三つのドア（D_i, $i = 1, 2, 3$）があり、一つのドアの裏に百万ドルがあり、残りのドアの裏に羊がいる
- ホストは参加者に好きなドアを一つ選んでもらう。そのあと、ホストは羊がいるドアを一つ開き、ドアを選びなおすかどうかを尋ねる。
- 大半の参加者の反応は「ドアを変えない」（なぜなら、各ドアの裏に百万ドルがある確率は等しいはずだから）
  - この反応は正しくない。それはなぜか？

モンティホール・ジレンマ (2)

- 参加者は最初にドア 1 を選んだと仮定する
- イベント A_i（ドア i の裏に百万ドルがある）の確率は $P(A_i) = 1/3$
  - 全確率の法則より, $P(\sim A) = 2/3$
- ホストは羊がいるドア 2 を開けた（イベント B_i）と仮定する

Q：ドア 2 が開けられたとき、ドア 1 と 3 の裏に百万ドルがある確率はそれぞれいくらか？

モンティホール・ジレンマ (3)

$$P(A_1|B) = \frac{P(B|A_1)P(A_1)}{P(B|A_1)P(A_1) + P(B|A_2)P(A_2) + P(B|A_3)P(A_3)}$$

- イベント A_i の事前確率（**prior probability**, **prioir belief**, **unconditional probability**）は $P(A_i) = 1/3$
- $P(B|A_1) = 1/2$（ドア 2 と 3 の裏に百万ドルはないので、ホストはどちらとも開けられるから）
- $P(B|A_2) = 0$（ホストは百万ドルがあるドアを絶対に開けないから）
- $P(B|A_3) = 1$（参加者はすでにドア 1 を選び、ドア 3 の裏に百万ドルがあるから）

モンティホール・ジレンマ (4)

$$P(A_1|B) = 1/3, \quad P(A_3|B) = 2/3$$

- A_3 の事前確率 $P(A_3) = 1/3$ が **updateing** とよばれる過程を通じて、事後確率（**posterior probability**, **posterior belief**）$P(A_3|B) = 2/3$ に上昇した

Whereas most of this book has to do with estimating effects from known causes, Bayes' rule reminds us that we can form reasonable beliefs about causes from known effects.

Summation operator (1)

$$\sum_{i=1}^{n} x_i = x_1 + x_2 + ... + x_n$$

基本的な性質

- $\sum_{i=1}^{n} c = nc$ for any constant c
- $\sum_{i=1}^{n} cx_i = c\sum_{i=1}^{n} x_i$ for any constant c
- $\sum_{i=1}^{n} (ax_i + by_i) = a\sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} y_i$ for any constant a and b
- $\sum_{i=1}^{n} x_i/y_i \neq \sum_{i=1}^{n} x_i / \sum_{i=1}^{n} y_i$
- $\sum_{i=1}^{n} x_i^2 \neq (\sum_{i=1}^{n} x_i)^2$

Summation operator (2)

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

基本的な性質

- $\sum_{i=1}^{n} (x_i - \bar{x}) = 0$
- $\sum_{i=1}^{n} (x_i - \bar{x})^2 = \sum_{i=1}^{n} x_i^2 - n\bar{x}^2$
- $\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) = \sum_{i=1}^{n} x_i y_i - n(\bar{x}\bar{y})$

期待値

変数 X は確率 $f(x_1), f(x_2), \ldots, f(x_k)$ によって、$x_1, x_2, \ldots, x_k$ という値を取る X の期待値（**expected value**）は

$$E(X) = \sum_{j=1}^{k} x_j f(x_j)$$

基本的な性質

- $E(c) = c$ for any constant c
- $E(aX + b) = aE(X) + b$ for any two constants a and b
- $E(\sum_{i=1}^{n} a_i X_i) = \sum_i a_i E(X_i)$ where $a_1, \ldots, a_n$ are numerous constants, and $X_1, \ldots, X_n$ are random variables
- $E(X - E(X)) = 0$
- $E(\cdot)$ は母集団に関する概念 (サンプルではなく、関心のあるグループ全体)

分散

X の分散は

$$V(X) = \sigma^2 = E[(X - E(X))^2] = E(X^2) - E(X)^2$$

基本的な性質

- $V(c) = 0$ for any constant c
- $V(aX + b) = a^2 V(X)$ for any two constants a and b
- $V(X + Y) = V(X) + V(Y) + 2(E(XY) - E(X)E(Y))$
  - If two random variables are independent, then $E(XY) = E(X)E(Y)$ and $V(X + Y) = V(X) + V(Y)$

不偏分散

母集団の概念である $V(X)$ の不偏推定量は

$$\hat{S}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2$$

自由度：なぜ $(n-1)$ で割るのか？

標本平均 $\bar{x}$ に分散があるから（recall 中心極限定理）

- 平均 μ と分散 σ^2 の IID サンプル x_i を考える
- 標本分散は $s^2 = n^{-1}\sum_i (x_i - \bar{x})^2 = n^{-1}\sum_i (x_i - \mu)^2 - (\bar{x} - \mu)^2$
- この期待値は $E(s^2) = \sigma^2 - V(\bar{x}) = \sigma^2(n-1)/n$
- よって、$E(S^2) = E(s^2 \cdot n/(n-1)) = \sigma^2$
- 標本が十分に大きくなると、$E(s^2)$ は σ^2 に収束する

共分散

二つのランダム変数の共分散は

$$\mathrm{Cov}(X, Y) = E(XY) - E(X)E(Y)$$

- $V(X + Y) = V(X) + V(Y) + 2\mathrm{Cov}(X, Y)$
  - X と Y が独立ならば, $\mathrm{Cov}(X, Y) = 0$ (sufficiency)
- $\mathrm{Cov}(a_1 + b_1X, a_2 + b_2Y) = b_1b_2\mathrm{Cov}(X, Y)$

相関

$$\mathrm{Corr}(X,Y) = \frac{\mathrm{Cov}(X,Y)}{\sqrt{V(X)V(Y)}} = \mathrm{Cov}\left(\frac{X-E(X)}{\sqrt{X}}, \frac{Y-E(Y)}{\sqrt{Y}}\right)$$

Regressin Review

母集団モデルの前提

- 関心のある母集団（population of intereset）から無作為標本を得ている
- 変数は x と y の二つ
- x の変化による y の変動を知りたい（not causal language）

母集団モデルの構築

母集団では以下のモデルが成立していると仮定する

$$y = \beta_0 + \beta_1 x + u$$

- 誤差項（error term）と呼ばれる変数を含んでいるので、このモデルは x 以外の要素が y に影響を与えることを許容している
- y が x に線形依存しているという仮定によって、関数形を明示的にモデル化している
- β_0 と β_1 は母集団を記述していて、これらのパラメーターのデータはないので、直接観察することはできない
  - 私たちにできることは、データと仮定を用いて、これらのパラメーターを推定すること
  - データを用いてパラメーターを正確に推定するために、信頼できる (credible) 仮定を置く必要がある

誤差項の仮定：標準化

$$E(u) = 0$$

- これは標準化の仮定であり、重要なものではない
- 仮に $E(u) = \alpha_0$ としても、切片 β_0 を調整すればよい。これは β_1 に影響を与えない。→ y = (_0 + _0) + _1 x + (u - _0)$

誤差項の仮定：平均独立（mean independence）

$$E(u|x) = E(u) \text{ for all values } x$$

- y を賃金、x を学歴、u を観察できない能力とする
- 平均独立の仮定は、異なる教育レベルの母集団で平均的な能力は同じことを課す $\rightarrow$ 能力に応じて教育投資を決めているならば、この仮定は成立していない

Zero conditional mean assumption

$$E(u|x) = 0 \text{ for all values } x$$

- 回帰分析モデルにおける重要な識別仮定（identification assumption）
- この仮定のもとで、$E(y|x) = \beta_0 + \beta_1 x$ を得る。
  - 母集団回帰モデル（**population regression function**）は x について線形である
  - Angrist and Pischke (2009, MHE) では、条件つき期待値関数（**condition expectation function**）
  - この関係は β_1 を因果パラメーター（**causal parameter**）として解釈するために重要なものである

最小二乗法（Ordinary least squares）の準備

パラメータ β_0 と β_1 を推定するために、u に関する仮定を用いる。

- 標準化仮定：$E(u) = 0$
- zero conditional mean assumption：$E(u|x) = 0 \rightarrow E(xu) = E(xE(u|x)) = 0$

これらの仮定に母集団モデルを挿入すると、

$$E(y - \beta_0 - \beta_1 x) = 0$$

$$E(x[y - \beta_0 - \beta_1 x]) = 0$$

- これらは β_0 と β_1 を決める母集団における仮定である。

最小二乗法の準備

母集団からのランダムサンプルを $\{(x_i, y_i) : i = 1, \ldots, n\}$ とする。私たちは母集団に関する情報にアクセスできないが、その標本対応（sample counterparts・sample analog）を知ることができる。

$$\frac{1}{n}\sum_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$$

$$\frac{1}{n}\sum_i x_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$$

- $\hat{\beta}_0$ と $\hat{\beta}_1$ は推定値（**estimates**）である

最小二乗法の推定量の導出

第一式より、

$$\bar{y} - \hat{\beta}_0 - \bar{x}\hat{\beta}_1 \to \hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}$$

これを第二式に代入すると、

$$\sum_i x_i[y_i - (\bar{y} - \hat{\beta}_1\bar{x}) - \hat{\beta}_1 x_i] = 0$$

$$\sum_i x_i(y_i - \bar{y}) = \hat{\beta}_1 \sum_i x_i(x_i - \bar{x})$$

$$\sum_i (x_i - \bar{x})(y_i - \bar{y}) = \hat{\beta}_1 \sum_i (x_i - \bar{x})^2$$

最小二乗法の推定量

$\hat{\beta}_1$ の推定量（**estimator**）は

$$\hat{\beta}_1 = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{\text{Sample covariance}(x, y)}{\text{Sample variance}(x)}$$

- この推定量は一般に **OLS** 推定量（**OLS estimator**）と呼ばれる
- x_i の標本分散が 0 でないとき、この推定値を得られる
  - x_i の変動が y_i にへの影響を識別するもの
  - 研究者が関心のある説明変数（causal variable）について同じ値を持っているサンプルを観察しているならば、y と x の関係における傾き具合を決められない
- $\hat{\beta}_0$ の OLS 推定量は $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$
  - $\hat{\beta}_1$ の推定値を得られたら、$\hat{\beta}_0$ の推定値も得られる

予測値と残差

ある observation i の予測値（fitted value）は

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x$$

ある observation i の残差（residual）は

$$\hat{u}_i = y_i - \hat{y}_i = y_i - \hat{\beta}_0 - \hat{\beta}_1 x$$

- 誤差項と残差の違い
  - 残差は予測誤差なので、データを用いて計算することができる → 回帰分析によってデータセットの中に現れる
  - 誤差項は定義（モデルによって捉えられない要素）より観察できない → データセットに現れることはない

予測誤差のサイズ

予測誤差のサイズは残差の二乗和（**the sum of squared residuals**）で計測できる

$$\sum_i \hat{u}^2 = \sum_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x)^2$$

- この予測誤差のサイズを最小化するような $\hat{\beta}_0$ と $\hat{\beta}_1$ の推定量は先のものと一致する

Python で OLS 推定したい：モジュールの導入

```
# モジュールの導入
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import plotnine as p

# モジュールにあるメソッドを導入
from itertools import combinations
from see import see  # 属性を調べるパッケージ
```

Python で OLS 推定したい：DGP

$$
\begin{aligned}
y_i &= 5.5x_i + 12u_i \\
x &\sim N(0,1) \\
u &\sim N(0,1)
\end{aligned}
$$

Python で OLS 推定したい：DGP の生成コード

```
np.random.seed(1)

tb = pd.DataFrame({
    "x": np.random.normal(size = 10000),
    "u": np.random.normal(size = 10000)
})
tb["y"] = 5.5 * tb["x"].values + 12 * tb["u"].values
```

pandas.DataFrame (1)

- Series と呼ばれる pandas の基本構造の一元配置をラベリングしたうえで、辞書型で集めたものが DataFrame(二次元配置)
- Series は Numpy の ndarray の一元配置
  - インデックスに番号以外のラベルを付けられる、オブジェクトそのものに名前をつけられる、時間データを格納できるという点で異なる
  - R の c() に近いもの

pandas.DataFrame (2)

R によるデータフレームの作成と直感的に似ている二つの方法

1. 2 次元配列：2 次元配列を作って、それを pandas.DataFrame に渡す

```
twod_array = [[0, 1, 2], [3, 4, 5]]
pd.DataFrame(twod_array)
```

pandas.DataFrame (3)

R によるデータフレームの作成と直感的に似ている二つの方法

2. 辞書表記：辞書型のオブジェクトを作成して、それを pandas.DataFrame に渡す。このとき、辞書のキーがカラムのラベルになり、その内容がカラムに入る

```
dic_arr = {"a": [1, 2, 3], "b": [4, 5, 6]}
pd.DataFrame(dic_arr)

state = pd.Series(["Tokyo", "Osaka", "Aichi"])
population = pd.Series([13515, 8839, 2844])
pd.DataFrame({"state": state, "pop": population})
```

pandas.DataFrame (4)

copy という概念

- 引数で `copy = False` としたとき、元のデータが変更されたら、新しい DataFrame オブジェクトにも反映される。`copy = True` は反映されない。
- コピー元の内容と関連付けるならば、`copy = False`。コピー元の内容と関連付けることをやめて、新しいオブジェクトとして認識させたいなら、`copy = True` を使う。

pandas.DataFrame (5)

- 新しいカラムは `DataFrame[" 新しいカラム名"] = データ`で追加できる（R と同じ）
- カラムの削除は `del DataFrame[" カラム名"]`
- インデックスとカラムに対応する値が存在しない場合が発生するとき、DataFrame は NaN でその値を補完

pandas.DataFrame (6)

値の参照は R と同じ方法だが、二種類のやり方がある

- `DataFrame.loc[" 行ラベル", " 列ラベル"]`：指定しない場合は`:`を使う
- `DataFrame.iloc[行番号, 列番号]`：指定しない場合は`:`を使う

pandas.DataFrame (7)

重要な属性（Attributes）- 属性とは、各オブジェクト固有のデータ保持領域にある要素のこと（R では、`lm` オブジェクトに `residuals` などの様々な要素がリストとして保存されていて、`lm()$residuals` でアクセスできる）- `DataFrame.colums` → カラム名の一覧を取得（R では `names(DataFrame)`）- `DataFrame.values` → データ部分のみを返す - `DataFrame.size` → データ部分のアイテム数を返す - `.describe()` → 各カラムの記述統計量を計算するメソッド

Python で OLS 推定したい：OLS 推定

```
reg_tb = smf.ols("y ~ x", data = tb).fit()
print(reg_tb.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                      y   R-squared:                       0.183
## Model:                            OLS   Adj. R-squared:                  0.183
## Method:                 Least Squares   F-statistic:                     2237.
## Date:                Thu, 24 Jun 2021   Prob (F-statistic):               0.00
## Time:                        02:16:51   Log-Likelihood:                -39049.
## No. Observations:               10000   AIC:                         7.810e+04
## Df Residuals:                    9998   BIC:                         7.812e+04
## Df Model:                           1
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## Intercept      0.1114      0.120      0.927      0.354      -0.124       0.347
## x              5.6887      0.120     47.293      0.000       5.453       5.924
## ==============================================================================
## Omnibus:                        0.640   Durbin-Watson:                   2.050
## Prob(Omnibus):                  0.726   Jarque-Bera (JB):                0.672
## Skew:                          -0.012   Prob(JB):                        0.715
## Kurtosis:                       2.968   Cond. No.                         1.01
## ==============================================================================
##
## Notes:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

statsmodels

Python の基本的な統計解析パッケージで、以下の三つに分かれている

- `statsmodels.api`：クロスセクション用のモデルとメソッドがある → `import statsmodels.api as sm`
- `statsmodels.tsa.api`：時系列データ用のモデルとメソッドがある → `import statsmodels.tsa.api as tsa`
- `statsmodels.formula.api`：解析モデルを formula(R と同じ) で指定できる便利 API → `import statsmodels.formula.api as smf`

statsmodels.api

statsmodels.api の関数は sm. 手法（endog，exog，...）

- OLS ならば、sm.OLS(Y，X1，X2，....，hasconst = bool) で解析できる。hasconst は切片を含めるかどうか（具体的には 1 で構成される配置を作成して、それを exog に入れるみたい）
- WLS ならば、sm.WLS()
- probit や logit はそれぞれ sm.Probit() と sm.Logit()
- formula でモデルを指定するばらば、sm. 手法.from_formula(formula string，data) でいける

statsmodels.formula.api

- R の formula と同様の書き方でモデルを定義できる
- statsmodels.formula.api の関数は statsmodels.api の手法をすべて小文字に変えたもの

.fit() メソッド

- この関数はあくまで解析手法を定義するだけのものなので、実際に実行するためには.fit() というメソッドを付ける必要がある
- .fit() を入れたオブジェクトにはいくつかの重要な Attribute がある
  - params → 係数を返す
  - resid → 残差を返す
  - nobs → 観測数を返す
  - sigma → 分散共分散行列を返す
  - wexog → デザイン行列（X) を返す
  - wendog → 被説明変数を配置で返す
  - .predict() → 予測値を計算するメソッド
  - .summary() → 結果の概要を示すメソッド

Python で OLS 推定したい：予測値と残差の計算

```
# 係数を取り出したい
reg_tb_b0 = reg_tb.params["Intercept"]
reg_tb_b1 = reg_tb.params["x"]

# 予測値を計算したい
tb["yhat1"] = reg_tb.predict(tb)
tb["yhat2"] = reg_tb_b0 + reg_tb_b1 * tb["x"]

# 残差を計算したい
tb["uhat1"] = reg_tb.resid
tb["uhat2"] = tb["y"] - tb["yhat2"]

tb.describe()
```

```
##                   x             u  ...         uhat1         uhat2
## count  10000.000000  10000.000000  ...  1.000000e+04  1.000000e+04
## mean       0.009773      0.009435  ...  7.219114e-16  7.219114e-16
## std        0.998836      1.001239  ...  1.201339e+01  1.201339e+01
## min       -3.656440     -3.451403  ... -4.142425e+01 -4.142425e+01
## 25%       -0.662925     -0.672294  ... -8.199882e+00 -8.199882e+00
## 50%        0.008454      0.018803  ...  4.497835e-02  4.497835e-02
## 75%        0.671809      0.688683  ...  8.147307e+00  8.147307e+00
## max        4.026849      4.168118  ...  5.000751e+01  5.000751e+01
##
## [8 rows x 7 columns]
```

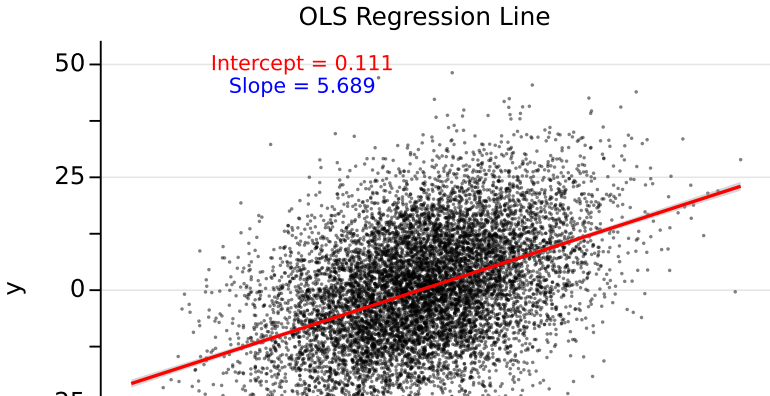
Python で OLS 推定したい：plotnine の準備

```
my_theme = (
    p.theme_minimal() +
    p.theme(

        # setting: plot
        panel_grid_major_x = p.element_blank(),
        panel_grid_minor_x = p.element_blank(),
        panel_grid_minor_y = p.element_blank(),

        # setting: axis
        axis_line = p.element_line(size = 1, color = "black"),
        axis_text = p.element_text(color="black",size=13),
        axis_title = p.element_text(size=13),
        axis_ticks = p.element_line(),
        axis_ticks_length = 6
    )
)
```

Pythonで OLS 推定したい・結果をプロットしたい

```
p.ggplot(tb, p.aes(x = "x", y = "y")) +\
    p.ggtitle("OLS Regression Line") +\
    p.geom_point(size = 0.05, color = "black", alpha = 0.5) +\
    p.geom_smooth(method = "lm", color = "red") +\
    p.annotate("text", x = -1.5, y = 50, color = "red", label = "Intercept = {:.3f}".format(reg_tb_b0)) +\
    p.annotate("text", x = -1.5, y = 45, color = "blue", label = "Slope = {:.3f}".format(reg_tb_b1)) +\
    my_theme
```

```
## <ggplot: (-9223372036820592152)>
```



plotnine という神モジュール

R ユーザーにとって神パッケージ。ggplot2 の文法でグラフが描ける（+ は +\で指定する）。あとは ggplot2 と同じ。すごい。

- annotate の中にある label について解説
  - `.format()` メソッドは文字列の書式設定を行える
  - `print("名前 {name:8s} です。{old:4d} 歳。".format(name="Tanaka", old=24))` → 名前 Tanaka　です。　24 歳。
  - 書式指定についての詳細はここを確認

OLS の代数的性質：残差の和はゼロ

$$\sum_i \hat{u}_i = 0$$

- 切片を含めたとき、$n^{-1}\sum_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$ を仮定した。
- OLS 推定量は残差の合計がゼロになる最適なものである
- 予測値の平均と観測値の平均は一致する（$n^{-1}\sum_i y_i = n^{-1}\sum_i \hat{y}_i + n^{-1}\sum_i \hat{u}_i$ からわかる）

OLS の代数的性質：残差と説明変数の共分散はゼロ

$$\sum_i x_i \hat{u}_i = 0$$

- OLS 推定量を導出するとき、$n^{-1} \sum_i x_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i) = 0$ を用いた。
- この結果より、残差と予測値の共分散（相関）は 0 であることを導ける $\rightarrow$ $\sum_i \hat{y}_i \hat{u}_i = 0$

OLS の代数的性質：回帰直線は平均を通る

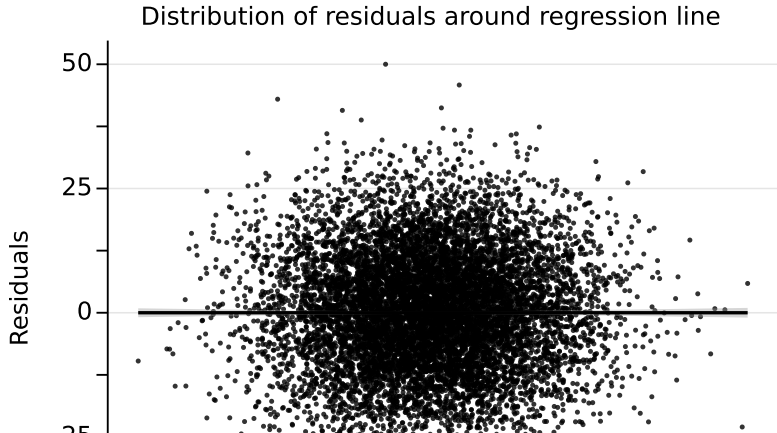
OLS の回帰直線は $(\bar{x}, \bar{y})$ を通る

$$\bar{y} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x}$$

Pythonでの OLS 推定したい・残差と予測値の散布図を描きたい

```
p.ggplot(tb, p.aes(x = "yhat1", y = "uhat1")) +\
    p.geom_point(size = 0.5, color = "black", alpha = 0.8) +\
    p.geom_smooth(method = "lm", color = "black") +\
    p.labs(title = "Distribution of residuals around regression line", x = "Fitted values", y = "Residuals") +\
    my_theme
```

```
## <ggplot: (-9223372036815294344)>
```



Python で OLS 推定したい：他の代数的性質を確認したい

```
tb["x_uhat"] = tb["x"].values * tb["uhat1"].values
tb["yhat_uhat"] = tb["yhat1"].values * tb["uhat1"].values

tb.sum()
```

```
## x             9.772657e+01
## u             9.435413e+01
## y             1.669746e+03
## yhat1         1.669746e+03
## yhat2         1.669746e+03
## uhat1         7.219114e-12
## uhat2         7.219114e-12
## x_uhat       -3.160494e-11
## yhat_uhat    -1.800800e-10
## dtype: float64
```

回帰分析を説明力に関する三つの指標

総平方和（**total sum of squares・SST**）、回帰平方和（**explained sum of squares・SSE**）、残差平方和（**residuals sum of squares・SSR**）を以下のように定義する

$$\mathrm{SST} = \sum_i (y_i - \bar{y})^2$$

$$\mathrm{SSE} = \sum_i (\hat{y}_i - \bar{y})^2$$

$$\mathrm{SSR} = \sum_i \hat{u}_i^2$$

- これらを $n-1$（自由度は 1）で割ると、SST は y_i の標本分散、SSE は $\hat{y}_i$ の標本分散、SSR は $\hat{u}_i$ の標本分散となる

SST・SSE・SSR の関係

$$
\begin{aligned}
\mathrm{SST} &= \sum_i (y_i - \bar{y})^2 \\
&= \sum_i [(y_i - \hat{y}_i) + (\hat{y}_i - \bar{y})]^2 \\
&= \sum_i [\hat{u}_i + (\hat{y}_i - \bar{y})]^2 \\
&= \mathrm{SSR} + \mathrm{SSE} + 2\sum_i \hat{u}_i \hat{y}_i - 2\bar{y}\sum_i \hat{u}_i \\
&= \mathrm{SSR} + \mathrm{SSE}
\end{aligned}
$$

Goodness-of-fit

$$R^2 = \frac{\mathrm{SSE}}{\mathrm{SST}} = 1 - \frac{\mathrm{SSR}}{\mathrm{SST}}$$

- x_i で説明できる y_i の変動の割合（**R-squared**）
- この値が大きいほど、y_i は OLS 回帰直線上に近づく
- この値は y_i と $\hat{y}_i$ の相関の二乗と一致する
- 因果効果を推定することを目標とするならば、R^2 の値に固執する必要はない（この値は因果関係について何も教えてくれない）

Python で OLS 推定したい：R2 値を計算したい

```
tb["ST"] = (tb["y"].values - tb["y"].mean())**2
tb["SE"] = (tb["yhat1"].values - tb["y"].mean())**2
tb["SR"] = (tb["uhat1"].values)**2

sst, sse, ssr = tb["ST"].sum(), tb["SE"].sum(), tb["SR"].sum()
r2_1, r2_2, r2_3 = sse/sst, 1 - ssr/sst, tb.corr().yhat1.y**2
r2_correct = reg_tb.rsquared

print("SSE/SST = {}".format(r2_1))
```

```
## SSE/SST = 0.18281009620183802
```

```
print("1 - SSR/SST = {}".format(r2_2))
```

```
## 1 - SSR/SST = 0.18281009620183786
```

```
print("squared corr(y, yhat1) = {}".format(r2_3))
```

```
## squared corr(y, yhat1) = 0.1828100962018387
```

```
print("OLS of statsmodels = {}".format(r2_correct))
```

```
## OLS of statsmodels = 0.18281009620183786
```

OLS の統計的性質：不偏性（unbiasedness）

$$E(\hat{\beta}) = \beta$$

- 異なるランダムサンプルを用いるとき、OLS 推定量はどのような値を吐き出すか？
  - ランダムサンプルを繰り返したならば、我々は平均的に正しい推定値を得られるか？
- $\hat{\beta}$ は特定のランダムサンプルによって得られる推定量の値
  - 異なるサンプルを用いれば、異なる推定値を得る
- 不偏性 $\rightarrow$ 母集団から多くのランダムサンプルを取り出し、推定量を計算したならば、推定値の平均は真値 β と等しくなる

不偏性の証明の仮定

1. linear in parameters：母集団モデルは $y = \beta_0 + \beta_1 x + u$ と仮定する
   - x と u はある data-genearting process (DGP) によって生成されたランダム変数の結果とみなせる
   - ならば、y も同様にランダム変数である
2. random sampling：母集団モデルにしたがって、サイズ n のランダムサンプル $\{(x_i, y_i) : i = 1, \dots, n\}$ を持っている
   - 母集団から取り出したものなので、各 i について $y_i = \beta_0 + \beta_1 x_i + u_i$ と書ける
3. sample vairation in the explanatory variables：x_i のサンプルアウトカムは同じ値を取らない
   - そもそもこれを満たしていないと、OLS 推定量は計算できない $\rightarrow$ 仮定とは言えない
4. zero conditional mean assumption：$E(u|x) = E(u) = 0$
   - 不偏性の証明で一番大事な仮定
   - 期待値がゼロという仮定は重要ではない。大切な部分は $E(u|x) = E(u)$

不偏性の証明 (1)

$$
\begin{aligned}
\hat{\beta}_1 &= \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \\
&= \frac{\sum_i (x_i - \bar{x}) y_i}{\sum_i (x_i - \bar{x})^2} \\
&= \frac{\beta_0 \sum_i (x_i - \bar{x}) + \beta_1 \sum_i (x_i - \bar{x}) x_i + \sum_i (x_i - \bar{x}) u_i}{\sum_i (x_i - \bar{x})^2} \\
&= \beta_1 + \frac{\sum_i (x_i - \bar{x}) u_i}{\sum_i (x_i - \bar{x})^2}
\end{aligned}
$$

- $\hat{\beta}_1$ は（a）母集団モデルの真値 β_1 と（b）u_i を x_i で OLS 回帰したときに得られる傾きの和

不偏性の証明 (2)

$w_i = (x_i - \bar{x}) / \sum_i (x_i - \bar{x})^2$ とする

$$
\begin{aligned}
\hat{\beta}_1 &= \beta_1 + \sum_i w_i u_i \\
E(\hat{\beta}_1) &= \beta_1 + \sum_i E(w_i u_i) \\
&= \beta_1 + \sum_i E[w_i E(u_i | x_1, \dots, x_n)] \\
&= \beta_1
\end{aligned}
$$

Python で不偏性のシミュレーションをしたい：DGP

$$
\begin{aligned}
y &= 3 + 2x + u \\
x &\sim N(0, 9) \\
u &\sim N(0, 36)
\end{aligned}
$$

Python で不偏性のシミュレーションをしたい

```
coefs = np.zeros(1000)

np.random.seed(3)
for i in range(1000):
    N = 10000
    sd_x, sd_u = np.sqrt(9), np.sqrt(36)
    tb = pd.DataFrame({
        "x": np.random.normal(scale = sd_x, size = N),
        "u": np.random.normal(scale = sd_u, size = N)
    })
    tb["y"] = 3 + 2 * tb["x"].values + tb["u"].values

    reg_tb = smf.ols("y ~ x", data = tb).fit()
    coefs[i] = reg_tb.params["x"]
```

Python で不偏性のシミュレーションをしたい (2)

```
p.ggplot() +\
    p.geom_histogram(p.aes(x = coefs), binwidth = 0.005, fill = "grey", color = "black") +\
    p.annotate("text", x = 2.04, y = 100, color = "red", label = "Obs = {}".format(coefs.size)) +\
    p.annotate("text", x = 2.04, y = 95, color = "red", label = "Mean = {:.3f}".format(np.mean(coefs))) +\
    p.annotate("text", x = 2.04, y = 90, color = "red", label = "Std.Dev. = {:.3f}".format(np.std(coefs))) +\
    my_theme
```

```
## <ggplot: (-9223372036815294344)>
```