第5章 基本的回帰分析

Python `statmodels` を使ったモデル

HDS 中村 知繁

T. Nakamura | Juntendo Univ.
2025/02/08

本章の目標

1. **データモデリング**の基本的な考え方を理解する。
    - 目的変数と説明変数（予測変数）
    - 説明のためのモデリング vs 予測のためのモデリング
2. **線形回帰**の基礎を学ぶ。
    - 1つの数値型説明変数を持つ**単純線形回帰**
    - 1つのカテゴリ型説明変数を持つ回帰
3. 線形回帰モデルの**構築**と**解釈**をPythonで行う。
4. **最小二乗法**の原理と、回帰係数の導出過程を理解する。
5. 「**相関は必ずしも因果を意味しない**」という重要な注意点を学ぶ。

はじめに (1/2)

- **データモデリング**: 目的変数 y と説明変数 x の関係性を明らかにする。
  - **目的変数 (Target Variable)** y: 予測・説明したい変数 (従属変数, 応答変数)
  - **説明変数 (Explanatory Variable)** x: y を説明・予測する変数 (独立変数, 共変量)
- 数学的には、y を x の「関数として」モデル化: $y = f(x)$
- **モデリングの主な目的**:
  1. **説明のためのモデリング**: y と x の関係を記述・定量化、有意性評価、因果の探求。
  2. **予測のためのモデリング**: x の情報から y を精度良く予測。

はじめに (2/2)

- 本書では主に**説明のためのモデリング**に焦点。
- 扱うモデリング手法: **線形回帰 (Linear Regression)**
  - 目的変数 y: 数値型
  - 説明変数 x: 数値型 または カテゴリ型
  - 仮定: y と x の関係は**線形 (直線的)**
- **本章の構成**: 説明変数が1つのモデル
  - 5.1節: 1つの数値的説明変数 (単純線形回帰)
  - 5.2節: 1つのカテゴリ的説明変数
- **キーワード**: 相関係数、相関と因果、最適な当てはめ直線

必要なパッケージ (Python)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from IPython.display import display # Colab等でDataFrameを綺麗に表示
```

- `seaborn` ライブラリからデータセットを読み込むため、事前のファイルダウンロードは不要です。

5.1 1つの数値的説明変数

リサーチクエスチョン:
レストランでの食事の総支払額 (`total_bill`) とウェイター/ウェイトレスへのチップの額 (`tip`) にはどのような関係があるか？

- **目的変数** y: チップの額 (`tip`) (数値型)
- **説明変数** x: 食事の総支払額 (`total_bill`) (数値型)

使用データ: `seaborn` の `tips` データセット

5.1.1 探索的データ分析 (EDA)

EDAの主なステップ:

1. データフレームの基本情報確認、実際のデータ値の確認。
2. 記述統計量（平均値、中央値など）の計算。
3. データの可視化。

1. データ読み込みと基本情報の確認

```
# seabornからtipsデータセットを読み込む
tips = sns.load_dataset('tips')

# データの最初の5行
print("レストランのチップデータの最初の5行:")
display(tips.head())

# データの構造確認
print("\nデータの構造:")
tips.info()
```

5.1.1 EDA (続き)

2. 記述統計量の計算

```
# 'tip'と'total_bill'の記述統計量
print("\n'tip'と'total_bill'の記述統計量:")
display(tips[['tip', 'total_bill']].describe())

# チップ額と総支払額の相関係数
correlation_tips = tips['tip'].corr(tips['total_bill'])
print(f"\nチップ額と総支払額の相関係数: {correlation_tips:.3f}")
# 出力例: チップ額と総支払額の相関係数: 0.676
```

- 相関係数約0.676は、比較的強い正の線形関係を示唆。

5.1.1 EDA (続き)

3. データの可視化 (散布図)

```
plt.figure(figsize=(8, 5)) # スライド用に調整
sns.scatterplot(x='total_bill', y='tip', data=tips, alpha=0.6)
plt.xlabel('総支払額 (total_bill)')
plt.ylabel('チップ額 (tip)')
plt.title('チップ額と総支払額の関係')
plt.grid(True, alpha=0.3)
plt.show() # このコードを実行するとColab等で図が表示されます
```

- 点が右上がりの傾向を示し、正の相関と整合的。

5.1.1 EDA (続き)

3. データの可視化 (散布図 + 回帰直線)

```
plt.figure(figsize=(8, 5)) # スライド用に調整
sns.regplot(x='total_bill', y='tip', data=tips,
            scatter_kws={'alpha':0.4, 's':30}, # 点のサイズ調整
            line_kws={'color':'blue'}, ci=None)
plt.xlabel('総支払額 (total_bill)')
plt.ylabel('チップ額 (tip)')
plt.title('チップ額と総支払額の関係 (回帰直線付き)')
plt.grid(True, alpha=0.3)
plt.show() # このコードを実行するとColab等で図が表示されます
```

- 回帰直線がデータ全体の傾向を捉えている。

5.1.2 単純線形回帰

モデル式: $\hat{y} = b_0 + b_1 x$

- b_0: 切片 (Intercept)
- b_1: 傾き (Slope)
- $\hat{y}$: 目的変数の予測値

`statsmodels` によるモデル構築:

```
model_tips_vs_bill = smf.ols('tip ~ total_bill', data=tips).fit()
# print(model_tips_vs_bill.summary()) # 全詳細表示
```

回帰結果の要約 (主要部分): (`model_tips_vs_bill.summary()` の `coef` 部分より)

- 切片 b_0 (`Intercept`): 約 0.9203
- 傾き b_1 (`total_bill`): 約 0.1050

当てはめられた回帰直線の方程式:

$$\widehat{\text{tip}} = 0.9203 + 0.1050 \cdot \text{total\_bill}$$

5.1.2 単純線形回帰 (係数の解釈)

- **切片 ($b_0 \approx 0.9203$)**:
  - 数学的解釈: `total_bill` (総支払額) が0ドルの場合の予測チップ額。
  - 実用的解釈: この文脈では `total_bill` = 0 は現実的でないため、実用的な意味は薄い。モデルの当てはまりに必要な項。
- **傾き ($b_1 \approx 0.1050$)**:
  - 解釈: `total_bill` が1ドル増加するごとに、 `tip` (チップ額) は**平均して**約0.105ドル (10.5セント) 増加する**関連がある**。
  - **注意点**:
    - 「関連がある」であり、因果関係ではない (5.3節で詳述)。
    - 「平均して」の傾向であり、個々のケースで必ずしもこの通りではない。

5.1.2 単純線形回帰 (回帰表)

```
regression_table_tips_vs_bill = pd.DataFrame({
    'term': ['Intercept', 'total_bill'],
    'estimate': model_tips_vs_bill.params.values,
    'std_error': model_tips_vs_bill.bse.values,
    't_value': model_tips_vs_bill.tvalues.values,
    'p_value': model_tips_vs_bill.pvalues.values,
    'conf_int_lower': model_tips_vs_bill.conf_int().iloc[:, 0].values,
    'conf_int_upper': model_tips_vs_bill.conf_int().iloc[:, 1].values
})
print("\n回帰表 (tip ~ total_bill):")
display(regression_table_tips_vs_bill) # Colab等で表示
```

(表示される表の例)

term	estimate	std_error	t_value	p_value	conf_int_lower	conf_int_upper
Intercept	0.920270	0.159735	5.76123	2.43e-08	0.606030	1.23451
total_bill	0.105023	0.007365	14.2605	6.69e-34	0.090522	0.119524

- `p_value` や `conf_int` (信頼区間) の解釈は後の統計的推論の章で。

5.1.3 観測値, 予測値, 残差

1. **観測値 (y_i)**: 実際のデータ値 (例: 実際のチップ額)
2. **予測値 ($\hat{y}_i$)**: 回帰モデルによる予測値 (回帰直線上の値)
    - $\hat{y}_i = b_0 + b_1 x_i$
3. **残差 (e_i)**: 観測値と予測値の差 (モデルの誤差)
    - $e_i = y_i - \hat{y}_i$

例: 最初のデータ点 `total_bill` = 16.99, `tip` = 1.01

- $\widehat{\text{tip}} = 0.9203 + 0.1050 \cdot 16.99 \approx 2.7043$
- 残差 $e = 1.01 - 2.7043 \approx -1.6943$

```
# 予測値と残差をデータフレームに追加
tips['tip_hat_vs_bill'] = model_tips_vs_bill.predict(tips)
tips['residual_vs_bill'] = model_tips_vs_bill.resid # または tips['tip'] - tips['tip_hat_vs_bill']
# display(tips[['total_bill', 'tip', 'tip_hat_vs_bill', 'residual_vs_bill']].head())
```

5.1.3 残差プロットと残差ヒストグラム

残差プロット: 横軸に予測値 $\hat{y}$、縦軸に残差 e

```
plt.figure(figsize=(8,5)) # スライド用に調整
sns.scatterplot(x='tip_hat_vs_bill', y='residual_vs_bill', data=tips, alpha=0.6)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('予測チップ額 (ŷ)'); plt.ylabel('残差 (y - ŷ)')
plt.title('残差プロット (tip ~ total_bill)'); plt.grid(True, alpha=0.3)
plt.show()
```

- 理想: 点が $y = 0$ の周りにランダムに分布。パターンがない。
- この例: 予測値大でばらつき増？(不等分散性の可能性)

5.1.3 残差プロットと残差ヒストグラム (続き)

残差のヒストグラム: 残差の分布を確認

```
plt.figure(figsize=(8,5)) # スライド用に調整
sns.histplot(tips['residual_vs_bill'], kde=True, bins=20)
plt.xlabel('残差 (y - ŷ)'); plt.ylabel('度数')
plt.title('残差のヒストグラム (tip ~ total_bill)'); plt.grid(True, alpha=0.3)
plt.show()
```

- 理想: 0を中心に対称的な分布 (例: 正規分布)。
- この例: 0中心だが、やや右に裾が長い（正の歪み）かも。

5.1.4 最適な当てはめ直線と最小二乗法

回帰直線は「最適な当てはめ直線」であり、**残差平方和 (Sum of Squared Residuals, SSR)** を最小にする直線です。

$$SSR = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} e_i^2$$

この SSR を最小にする b_0, b_1 を求める方法が **最小二乗法 (Method of Least Squares)** です。

b_0, b_1 の導出 (アイデア):
モデル: $\hat{y}_i = b_0 + b_1 x_i$
SSR: $SSR(b_0, b_1) = \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_i))^2$
目標: $SSR(b_0, b_1)$ を最小にする b_0, b_1 を見つける。
方法: SSR を b_0 と b_1 でそれぞれ偏微分し、0 とおく。

5.1.4 最小二乗法による b_0, b_1 の導出 (1/4)

1. SSR を b_0 で偏微分し、0 とおく:

$$\frac{\partial SSR}{\partial b_0} = -2\sum_{i=1}^{n}(y_i - b_0 - b_1 x_i)$$

$\frac{\partial SSR}{\partial b_0} = 0$ より、

$$\sum(y_i - b_0 - b_1 x_i) = 0$$

$$\sum y_i - nb_0 - b_1\sum x_i = 0$$

整理して、**第一正規方程式**:

$$nb_0 + b_1\sum x_i = \sum y_i \quad \cdots(1)$$

5.1.4 最小二乗法による b_0, b_1 の導出 (2/4)

2. SSR を b_1 で偏微分し、0 とおく:

$$\frac{\partial SSR}{\partial b_1} = -2\sum_{i=1}^{n} x_i(y_i - b_0 - b_1 x_i)$$

$\frac{\partial SSR}{\partial b_1} = 0$ より、

$$\sum x_i(y_i - b_0 - b_1 x_i) = 0$$

$$\sum x_i y_i - b_0 \sum x_i - b_1 \sum x_i^2 = 0$$

整理して、**第二正規方程式**:

$$b_0 \sum x_i + b_1 \sum x_i^2 = \sum x_i y_i \quad \cdots (2)$$

5.1.4 最小二乗法による b_0, b_1 の導出 (3/4)

3. **正規方程式 (1), (2) を解く**:
   (1)式 $nb_0 + b_1 \sum x_i = \sum y_i$ より、b_0 について解くと:

$$b_0 = \frac{\sum y_i - b_1 \sum x_i}{n} = \bar{y} - b_1 \bar{x} \quad \cdots (3)$$

   ここで、$\bar{x} = \frac{\sum x_i}{n}, \bar{y} = \frac{\sum y_i}{n}$ は平均値。
   (この式は、回帰直線が点 $(\bar{x}, \bar{y})$ を通ることを示す)

   (3)式を(2)式 $b_0 \sum x_i + b_1 \sum x_i^2 = \sum x_i y_i$ に代入:

$$(\bar{y} - b_1 \bar{x}) \sum x_i + b_1 \sum x_i^2 = \sum x_i y_i$$

   b_1 について整理すると:

$$b_1 \left(\sum x_i^2 - \bar{x} \sum x_i \right) = \sum x_i y_i - \bar{y} \sum x_i$$

5.1.4 最小二乗法による b_0, b_1 の導出 (4/4)

$$b_1 \left(\sum x_i^2 - \frac{(\sum x_i)^2}{n} \right) = \sum x_i y_i - \frac{(\sum y_i)(\sum x_i)}{n}$$

よって、傾き b_1 は:

$$b_1 = \frac{n \sum x_i y_i - (\sum x_i)(\sum y_i)}{n \sum x_i^2 - (\sum x_i)^2}$$

または、偏差表記 $S_{xy} = \sum (x_i - \bar{x})(y_i - \bar{y}), S_{xx} = \sum (x_i - \bar{x})^2$ を用いて:

$$b_1 = \frac{S_{xy}}{S_{xx}}$$

そして、b_1 が求まれば、(3)式から切片 b_0 は:

$$b_0 = \bar{y} - b_1 \bar{x}$$

- これらの計算はソフトウェアが自動で行います。原理の理解が重要です。

5.1.4 最適な当てはめ直線 (SSRの確認)

```
# statsmodels が計算したモデルの残差平方和
ssr_tips_vs_bill = model_tips_vs_bill.ssr
print(f"このモデルの残差平方和 (SSR): {ssr_tips_vs_bill:.3f}")
# 出力例: このモデルの残差平方和 (SSR): 252.788
```

`statsmodels` が出力した係数 b_0, b_1 は、このSSRを最小にする値です。

5.2 1つのカテゴリ的説明変数

リサーチクエスチョン:
レストランで支払われるチップの額 (`tip`) は、曜日 (`day`) によって異なるか？

- **目的変数** y: チップの額 (`tip`) (数値型)
- **説明変数** x: 曜日 (`day`) (カテゴリ型: 木, 金, 土, 日)

引き続き `tips` データセットを使用。

5.2.1 探索的データ分析 (EDA)

1. `day` 変数の確認と記述統計量

```
print("曜日のカテゴリとそれぞれの件数:")
display(tips['day'].value_counts())

print("\n曜日ごとのチップ額の記述統計量:")
tip_by_day_summary = tips.groupby('day')['tip'].agg(
    ['mean', 'median', 'std', 'min', 'max', 'count']
).sort_values('mean', ascending=False)
display(tip_by_day_summary)
```

- 平均チップ額: 日曜 > 土曜 > 木曜 > 金曜 の順。

5.2.1 EDA (続き)

2. データの可視化 (箱ひげ図)

```
day_order = tips['day'].cat.categories
plt.figure(figsize=(8, 6)) # スライド用に調整
sns.boxplot(x='day', y='tip', data=tips, order=day_order)
plt.xlabel('曜日 (day)'); plt.ylabel('チップ額 (tip)')
plt.title('曜日別のチップ額の分布'); plt.grid(True, axis='y', alpha=0.3)
plt.show()
```

- 曜日によってチップ額の分布に違いが見られる。

(バイオリンプロットも同様に提示可能ですが、スライド枚数を考慮し省略。講義では口頭で触れるか、追加スライドで)

5.2.2 線形回帰 (カテゴリ型説明変数)

`statsmodels` で `C(day)` のように指定すると、カテゴリ変数を適切に扱ってくれます。

```
model_tips_vs_day = smf.ols('tip ~ C(day)', data=tips).fit()
# print(model_tips_vs_day.summary())
```

回帰結果の要約 (主要部分): (`model_tips_vs_day.summary()` の `coef` 部分より)

- `Intercept`: 約 2.771 (木曜日の平均チップ額)
- `C(day)[T.Fri]`: 約 -0.030 (金曜 vs 木曜の差)
- `C(day)[T.Sat]`: 約 0.212 (土曜 vs 木曜の差)
- `C(day)[T.Sun]`: 約 0.478 (日曜 vs 木曜の差)

基準カテゴリ: `tips['day']` の最初のカテゴリ (`Thur`: 木曜日) が基準。
他の曜日の係数は、木曜日との平均チップ額の差 (オフセット) を表す。

5.2.2 線形回帰 (係数の解釈)

- `Intercept` **(**$b_0 \approx 2.771$**)**:
  基準カテゴリである**木曜日 (**`Thur`**)** の平均チップ額の推定値。
- `C(day)[T.Fri]` **(**≈ -0.030**)**:
  金曜日 (`Fri`) の平均チップ額は、木曜日の平均チップ額より約0.030ドル低い。
  (金曜日の平均チップ額 $\approx 2.771 - 0.030 = 2.741$)
- `C(day)[T.Sat]` **(**≈ 0.212**)**:
  土曜日 (`Sat`) の平均チップ額は、木曜日より約0.212ドル高い。
  (土曜日の平均チップ額 $\approx 2.771 + 0.212 = 2.983$)
- `C(day)[T.Sun]` **(**≈ 0.478**)**:
  日曜日 (`Sun`) の平均チップ額は、木曜日より約0.478ドル高い。
  (日曜日の平均チップ額 $\approx 2.771 + 0.478 = 3.249$)

5.2.2 線形回帰 (回帰表)

```
base_category = tips['day'].cat.categories[0] # 'Thur'
terms_day = [f'Intercept ({base_category})']
for term_name in model_tips_vs_day.params.index[1:]:
    day_name = term_name.split('[T.')[1][:-1]
    terms_day.append(f"{day_name} (vs {base_category})")

regression_table_tips_vs_day = pd.DataFrame({ # (以下略, 前のMD参照)
    'term': terms_day,
    'estimate': model_tips_vs_day.params.values, # ...
    # ... (std_error, t_value, p_value, conf_int)
})
print(f"\nカテゴリ変数を用いた回帰の回帰表 (基準: {base_category}):")
display(regression_table_tips_vs_day)
```

(表示される表の例は前のMarkdown資料を参照)

- 各曜日の平均チップ額推定値が、EDAで計算したグループ平均と一致することを確認。

5.2.3 観測値, 予測値, 残差

- **予測値 ($\hat{y}_i$)**: 各データ点が属する曜日の平均チップ額の推定値。
  (例: 木曜日のデータなら $\hat{y}_i = 2.771$, 金曜日なら $\hat{y}_i = 2.741$)
- **残差 ($e_i = y_i - \hat{y}_i$)**: 個々のチップ額と、その曜日の平均チップ額推定値との差。

```
tips['tip_hat_vs_day'] = model_tips_vs_day.predict(tips)
tips['residual_vs_day'] = model_tips_vs_day.resid
# display(tips[['day', 'tip', 'tip_hat_vs_day', 'residual_vs_day']].head())
```

曜日別残差プロット (箱ひげ図):

```
plt.figure(figsize=(8, 6)) # スライド用に調整
sns.boxplot(x='day', y='residual_vs_day', data=tips, order=day_order)
plt.axhline(y=0, color='red', linestyle='--'); plt.xlabel('曜日 (day)')
plt.ylabel('残差 (tip - tip_hat_vs_day)'); plt.title('曜日別の残差の分布')
plt.grid(True, axis='y', alpha=0.3); plt.show()
```

- 各曜日の平均からのばらつき具合を示す。

5.3 相関は必ずしも因果を意味しない

重要: 2つの変数 X と Y に統計的な関連 (相関や回帰係数) が見られても、
必ずしも「X が Y を引き起こす」という**因果関係**を意味するわけではない。

例: 靴を履いて寝る (X) と 頭痛 (Y)

- 観察: X と Y に正の相関。
- 結論の誤り: 「靴を履いて寝ると頭痛になる！」
- **交絡変数 (Confounding Variable) Z**: 前夜の飲酒量
  - 飲酒 (Z) $\rightarrow$ 靴を履いて寝る (X)
  - 飲酒 (Z) $\rightarrow$ 頭痛 (Y)

- XとYの見かけ上の関連は、Zによって説明されるかもしれない。
- 因果関係の主張には、ランダム化比較試験 (RCT) や高度な統計手法が必要。

まとめ (1/2)

本章で学んだこと:

1. **モデリングの目的**: 説明 vs 予測。
2. **単純線形回帰 ($\hat{y} = b_0 + b_1 x$)**:
   - 数値型説明変数 (`total_bill` $\rightarrow$ `tip`)
   - 係数 b_0 (切片), b_1 (傾き) の解釈。
3. **最小二乗法による係数導出**:
   - 残差平方和 (SSR) $\sum (y_i - \hat{y}_i)^2$ を最小化。
   - 偏微分を用いた正規方程式とその解法（b_0, b_1 の公式）。
4. **カテゴリ型説明変数を持つ回帰**:
   - (`day` $\rightarrow$ `tip`)
   - 基準カテゴリとオフセットとしての係数の解釈。

まとめ (2/2)

5. **探索的データ分析 (EDA)**: 可視化や記述統計の重要性。
6. **観測値、予測値、残差**:
    - $e_i = y_i - \widehat{y}_i$
    - 残差プロットやヒストグラムによるモデル評価の初歩。
7. **相関と因果**: 「相関は因果ではない」。交絡変数の可能性。

使用ツール: Python (`pandas`, `matplotlib`, `seaborn`, `statsmodels`)

次章予告: 複数の説明変数を持つ**重回帰分析**へ。

(付録) Pythonコード一括実行用

このスライドで使用したPythonコードの多くは、以下のセルにまとめてあります。
Google Colab等で実行して、結果やグラフを実際に確認してみてください。

```
# 必要なライブラリのインポート
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
import statsmodels.formula.api as smf
from IPython.display import display
```

```
# --- 5.1節用コード ---
print("=== 5.1 1つの数値的説明変数 ===")
tips = sns.load_dataset('tips')
# (EDAコード ... display(tips.head()), tips.info(), display(tips.describe()...))
# (可視化コード ... sns.scatterplot, sns.regplot)
# (モデル構築 ... smf.ols('tip ~ total_bill', data=tips).fit())
# (回帰表作成 ... pd.DataFrame(...))
# (予測値・残差計算 ... tips['tip_hat_vs_bill'] = ...)
# (残差プロット等 ... sns.scatterplot(x='tip_hat_vs_bill',...))

# --- 5.2節用コード ---
print("\n=== 5.2 1つのカテゴリ的説明変数 ===")
# (EDAコード ... tips['day'].value_counts(), tips.groupby('day')...)
# (可視化コード ... sns.boxplot(x='day', y='tip', ...))
# (モデル構築 ... smf.ols('tip ~ C(day)', data=tips).fit())
# (回帰表作成 ... pd.DataFrame(...))
# (予測値・残差計算 ... tips['tip_hat_vs_day'] = ...)
# (残差プロット等 ... sns.boxplot(x='day', y='residual_vs_day',...))
```