

分析設計

《学修項目》

- ◎データ分析の進め方、仮説検証サイクル、分析目的の設定
- データの収集、加工、統合
- 様々なデータ可視化手法（量的データを中心として）
- 基本統計量と相関係数

《キーワード》

仮説検証サイクル、分析目的の設定、データの収集・加工・分割／統合データの集計、前処理、ソート、欠損値の扱い、外れ値処理、比較対象の設定、さまざまなデータ可視化手法、量的データ、ヒストグラム、箱ひげ図、散布図、ヒートマップ、分割表とクロス集計、代表値、相関係数

《参考文献、参考書籍》

- [1] 東京大学MIセンター公開教材 「1-2 分析設計」 《利用条件CC BY-NC-SA》
- [2] 応用基礎としてのデータサイエンス（講談社 データサイエンス入門シリーズ）
- [3] データサイエンスの考え方 社会に役立つAI×データ活用のために（オーム社）
- [4] Pythonによるあたらしいデータ分析の教科書 第2版（翔泳社）
- [5] 数理・データサイエンス・AI公開講座（放送大学）

1. データ分析の進め方 [1]

1.1 仮説検証サイクル [1]

データ分析によって全てがわかるわけではなく、仮説を立て、実験計画を立てて検証し、正しい場合は価値を見極め、新しい仮説を生み出すことが重要である。これは科学研究と同じ方法であり、データ分析の結果だけに頼らず、継続的に検証と改善を行う必要があるということである。

仮説検証サイクルは、経営学でよく使われる PDCAサイクル（Plan-Do-Check-Action）と同じであり、計画、実行、評価、対策のプロセスを繰り返していくことが重要である。データ分析を行った結果、自分が考えたこととは異なる結果になることが多いため、仮説を修正したり、新たな仮説を立てたりする必要がある。

- [NRI | PDCAサイクル](<https://www.nri.com/jp/knowledge/glossary/lst/alphabet/>)

1. データ分析の進め方

1-1. 仮説検証サイクル

データ分析の目的は大きく分けて次の2つにあります：

1. 傾向や仮説の候補の発見
2. 仮説の検証

発見しただけでは傾向や仮説が正しいかどうかが言えず、**検証**のためには良い仮説が無いといけません。また、2の結果によっては仮説の候補を再検討する必要があります。

そこで通常は1→2→1→2…という反復（仮説検証サイクル）を満足な仮説とその検証が得られるまで繰り返すことになります。

仮説検証

- 例えば「3つ以上同じ作品を視聴しているユーザー同士は、お互いまだ見ていない作品を推薦すると視聴することが多い」という仮説を思いついたとしましょう。
 - 実際のシステムはこれよりはるかに複雑です。
- この際に仮説をすぐサイトで運用するのではなくランダムにユーザーを分けて仮説と従来手法を比較する比較実験を行います。
 - ABテストやランダム化比較試験と呼ばれます。簡単に見えますが他の要因を取り除けるので非常に強力な手法です。
- 仮説検証をすることで本当にうまくいく仮説を残すことができます。

仮説検証（実験）

<https://www.youtube.com/watch?v=0CGQvdAbNcc> の4:05

- Caitlin Smallwood (Netflix) | How Netflix Data Science Powers Global Entertainment (タイムライン4:05から)

1.2 分析目的の設定 [1]

1. データ分析の進め方

1-2. 分析目的の設定

データ分析の各ステップにおいては、そこでの主要な目的が発見にあるのか
検証にあるのか明確にしておいた方が良いでしょう。おおざっぱに言って、
発見のためには可視化や特徴量の比較、クラスタリングなどの手法が、
検証には相関分析や回帰、仮説検定などの手法が関係してきます。

どちらの場合も、分析対象や分析データの特徴、検証しようとしている仮説について良く考えることが必要です。

AIや深層学習への期待で誤解されがちですが、「よく考えないままにデータを投げ入れると答えを出してくれる」と思って良いということはありません。

東京大学 数理・情報教育研究センター 島田 尚 2021 CC BY-NC-SA

5

- KUROCO | データ分析とは？目的や正しいプロセス、手法、活用事例を解説（データ分析は「目的設定」が9割）

2. データの収集、加工、統合

2.1 データの種類、データのオーダ、前処理、欠損値の処理、外れ値の処理

2.1.1 データの種類

構造化データ

- 分析できる形に構造化されたデータ
- 表形式で整理されたデータセット

→量的データ、質的データに分類される

非構造化データ

- まだ構造化されていないデータ

→画像、音声、文章データなど *DE概論/データ表現の章を参照

2.1.2 データのソート

- 順序付けの基準（オーダ）

- 量的変数の場合：科目的得点など（間隔尺度（例：西暦年など）、数値尺度（例：重量など）の別がある）

- 質的変数の場合：名義尺度（学生名など），順序尺度（学籍番号など。順序が存在し、数量によるソートが可能）
- 昇順と逆順
 - 数値の大小で順序付ける：質的変数も、順序尺度であれば可能
 - 名義尺度も あいうえお順、ABC順などで順序付け ($aab < ac < adbc$ など文字列にもオーダがつけられる)

3. データの収集、加工、統合

分析に進む際に、扱うデータがそれぞれどんな性質のものかについて把握しておく必要があります。データの種類には大別して以下があります：

- **量的変数**

自然に数値で扱うことができるもの（金額、人数、長さ、視聴率、…）

- **質的変数**

大きさを表す数値として扱えないもの。さらに次の2つに大別できます：

- **順序尺度**：区分の間に順番はつけられるもの

（Jリーグの年間順位、和牛の等級、癌のステージ、…）

- **名義尺度**（カテゴリー変数）：順番もつけられない、互いに違うもの

（居住地区、職業、支持政党、「上司にしたい芸能人」、…）

→ カテゴリー変数は、便宜上整数を割り当てて扱うことがあります。

ダミー変数と呼ばれるこの数には当然、大きさも順序も意味がありません

6

2.1.3 データの前処理

データクレンジング、データクリーニング

- 破損したデータ、不正確なデータ、無関係のデータなどの処理
- 住所や名簿などをデータ分析可能な状態にすること

主なデータ前処理

- 欠損値処理
- 外れ値処理、異常値処理
- ノイズの除去

データの変換

- 前処理としてのデータの変換
- 型変換やスケーリング
- 正規化、標準化

3. データの収集、加工、統合

データには欠損値、外れ値、データの間での書式の不統一、例外、…などがあることが普通です。計算機で処理する際にはこれらの値を統一の書式に揃えたり統合したりする必要があります。

このような加工や統合には手間がかかることが多く、場合によってはデータ収集や入力の仕方の再考をした方が良いこともあるでしょう。

また、どのようなデータを取るべきかや欠損や例外の処理の仕方は仮説によって変わることがあります。

データクレンジング

- 汚れたデータを特定して除去・訂正することです
 - 汚れの例：破損したデータ、不正なデータ、例外的なデータ
 - 統計処理の観点では、外れ値・異常値・欠測値への対処が主です
- ◆ 外れ値：他の測定値から大きく外れた値
- 観測上の異常です
 - 一般に原因は分かりません
- ◆ 異常値：外れ値の中で異常さの原因が解明されているもの
- 本質的な異常です
 - 誤りや故障によって、異常であるべくして異常になったものです
- ◆ 欠測値：測定データに含まれていない、本来測定されているべき値
- 例えば、アンケートの「無回答」や記入漏れなどで生じます

2.1.4 欠損値の扱い

欠損値を処理する一般的な方法

- 削除：列や行ごと削除する
- 平均置換：平均値で置き換える（当該レコードの信頼性が毀損する）
- 最頻値で置き換え：質的データに有効
- ダミー置換：ダミー変数（Unknown, Zeroなど）で置き換える

完全データ：全ての値が観測されているデータ 不完全データ：欠損が含まれているデータ

頻繁に欠損値が出現するが、量的データのモデルが既知の場合には、欠損データを予測可能な回帰モデルを作成し、逐次代入する方法がとられる（統計的数値補完）

- 参考：note.nkmk.me | pandasで欠損値NaNを除外（削除）・置換（穴埋め）・抽出

欠測値への対処

- 欠測値は統計処理を不可能にします
 - 欠測値が入ると算術平均すら定義されません
→ 算術平均に基づく統計量が全て定義されません
 - 欠測値への典型的な対処法が2つあります
- ◆ 欠測値を含むケースを丸々除去する
- つまり、欠測したものは存在しないものとして扱う
- ◆ 欠測値に非欠測ケースの平均値を代入する
- つまり、欠測値は平均値に対して無害な値であると仮定する
- これらは統計処理を可能にしますが、統計量（統計モデル）に対してバイアスを生みます

2.1.5 外れ値処理、異常値処理

- 外れ値：通常とはかけ離れた値
 - 異常の大きい、小さいなど
- 外れ値処理
 - 一般的には「残す」

+異常値：存在し得ない値など（例：非常に小規模な図書館の入場者が100万人であることを示すデータ、など） * 欠損値と同じ扱いとする * 個別に吟味し、置き換えや削除を検討する

一般的に、データ件数、レコード数が非常に大きくなると、欠損値・外れ値・異常値を含む可能性が高まる

外れ値の検定

- 外れ値を特定する検定には様々な方法があります
- 単純かつ代表的な検定は次の2つです
 - ◆ 第1と第3四分位数の区間から外れる
 - つまり上位25%と下位25%を外れ値と見做します
 - ◆ 平均からの乖離が 2σ ～ 3σ に達する
 - σ は標準偏差を意味します

2.2 Pythonによるデータ抽出の実装例

Pythonによる実装については、クラウド環境であればGoogle colab.が即時使える。ローカルPC上の環境であれば、Windows, macOS, Linuxなどで各々別途インストール作業が必要となる。詳細は[こちらの参考書籍付属の公開資料](#)を参照されたい。

- [Pythonで学ぶアルゴリズムとデータ構造（講談社） 公開資料](#)
- [Pythonチュートリアル | 3. 形式ばらない Python の紹介](#)

2.2.1 サンプルデータ

ChickWeight : 鶏の体重 weight[g]、日齢 Time (0..42日) 、鶏のID Chick (1..50) 、餌の種類 Diet (1..4) (n=577)

体重・日齢は量的変数、鶏のID・餌の種類は質的変数/名義尺度

3. データの収集、加工、統合

餌と鶏の成長についてのデータの例

データ行	鶏体重[g]	日齢	鶏のID	餌の種類
1	42	0	1	1
2	51	2	1	1
3	59	4	1	1
4	64	6	1	1
5	76	8	1	1
6	93	10	1	1
7	106	12	1	1
8	125	14	1	1
9	1490	16	1	1
10	171	18	1	1
11	199	20	1	1
12	205	21	1	1
13	40	0	2	1
14	NA	2	2	1
15	58	4	2	1

測定できなかった等

量的変数

量的変数

名義尺度のダミー変数

(<https://stat.ethz.ch/R-manual/R-devel/library/datasets/html/ChickWeight.html> を加工)



東京大学 数理・情報教育研究センター 島田 尚 2021 CC BY-NC-SA

8

- [The R Datasets Package | ChickWeight - Weight versus age of chicks on different diets](#)
- [ChickWeight.csv download](#)

2.2.2 欠損値のあるレコードの検出

```
In [16]: # オリジナルのCSVデータを カレントディレクトリ直下のフォルダ(一時作業領域)へダウンロードする
!wget -nc https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/

# 欠損値,外れ値あり加工済CSVファイルもダウンロードする
!wget -nc https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/

# wgetしなくても,Google colab. の左メニュー [ファイル] アイコンをクリックして,ブラウザへファイルを

# ファイル (ChickWeight_xxxxxxx.csv) がダウンロード・配置できたことを確認する
##!ls -al ./

# オリジナルと加工済ファイルの相違点を diff で確認しておく
!diff ChickWeight_original.csv ChickWeight_error.csv
```

```
--2023-04-02 06:34:08-- https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/resources/ChickWeight_original.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6406 (6.3K) [text/plain]
Saving to: 'ChickWeight_original.csv'

ChickWeight_origina 100%[=====] 6.26K --.-KB/s in 0s
```

```
2023-04-02 06:34:08 (55.6 MB/s) - 'ChickWeight_original.csv' saved [6406/6406]
```

```
--2023-04-02 06:34:08-- https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/resources/ChickWeight_error.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6408 (6.3K) [text/plain]
Saving to: 'ChickWeight_error.csv'
```

```
ChickWeight_error.c 100%[=====] 6.26K --.-KB/s in 0s
```

```
2023-04-02 06:34:08 (73.7 MB/s) - 'ChickWeight_error.csv' saved [6408/6408]
```

```
9c9
< 149,32,1,1
---
> 1490,32,1,1
14c14
< 49,4,2,1
---
> NaN,4,2,1
```

```
In [17]: # 欠損値,外れ値あり加工済CSVファイルをpandasで読み込んでデータフレームdfに格納
import pandas as pd
from IPython.display import display

df = pd.read_csv('ChickWeight_error.csv')
df.columns = ['weight','Time','Chick','Diet']

# インタラクティブ表示にして, [Filter] > [Search by all fields:] に NaN で検索すると,欠損
# 外れ値(異常値)は、weightの下限に 1000 とか入れてやると探せるかも?
display(df)
```

	weight	Time	Chick	Diet
0	51.0	4	1	1
1	59.0	8	1	1
2	64.0	12	1	1
3	76.0	16	1	1
4	93.0	20	1	1
...
572	175.0	28	50	4
573	205.0	32	50	4
574	234.0	36	50	4
575	264.0	40	50	4
576	264.0	42	50	4

577 rows × 4 columns

```
In [18]: # 行・列ごとに欠損値の個数をカウント
print(df.isnull().sum())
```

```
weight      1
Time       0
Chick      0
Diet       0
dtype: int64
```

```
In [19]: # 欠損値NaNが一つでも含まれる行を抽出
print(df[df.isnull().any(axis=1)])
```

	weight	Time	Chick	Diet
12	NaN	4	2	1

2.2.3 外れ値のあるレコードの検出

```
In [ ]: # numexprのversion 2.8.5はqueryにバグがあるので、バージョン指定でインストール場合 (2023)
# 参考:https://github.com/pandas-dev/pandas/issues/54449
# !pip install numexpr==2.8.4
```

```
In [20]: # 分位数を指定する-df.quantile()
```

```
q = df['weight'].quantile(1 - 0.001) # 分位数 上位0.1%に属するもの(調整してみよ)
print(q)
```

```
# local variable substitution syntaxを用いた方法(numexprのversion 2.8.5はquery)
# df_quan = df.query('weight > @q') # queryを使って外れ値となっているレコードを確認する
```

```
# dataframe自体の行抽出処理を使う方法 (2023.8.31 YM)
df_quan2 = df[df['weight'] > q] # queryを使って外れ値となっているレコードを確認する
df_quan2
```

847.7250000000762

```
Out[20]:   weight  Time  Chick  Diet
7    1490.0     32      1      1
```

3. 基本的なデータ可視化手法（量的データを中心として） [2]

可視化の主な目的は「比較する」「構成を見る」「分布を見る」「変化を見る」の4つに分けらる。表3.1は、可視化の目的とデータの尺度に応じて基本的なグラフを分類したものであり、それぞれのグラフ単独での目的に基づいて分類していることが明記されている（同じグラフを複数並べての比較は含まれていない） [2]。

表3.1 データの量質 × 目的ごとの可視化法 ([2]より引用)

	比較	構成	分布	変化
量的データ	レーダーチャート		箱ひげ図 ヒストグラム バイオリンプロット 散布図 散布図行列	折れ線グラフ
質的データ	棒グラフ	円グラフ 帯グラフ	モザイクプロット ヒートマップ	
	パレート図 積み上げ棒グラフ 層別帯グラフ			

統計グラフを利用する際には、データの尺度や変数の数に留意する必要がある。表3.2では、これらに応じた可視化法が区分されている。量的データにはヒストグラムや箱ひげ図、質的データには円グラフや棒グラフがよく用いられる。また、量的データには散布図、質的データにはモザイクプロットやヒートマップがよく用いられる [2]。

表3.2 データの量質 × 変数の数ごとの可視化法 ([2]より引用)

	1変数データ	2変数データ	多変数データ
量的データ	箱ひげ図 ヒストグラム バイオリンプロット 折れ線グラフ	散布図	散布図行列 レーダーチャート
質的データ	棒グラフ パレート図 円グラフ 帯グラフ	積み上げ棒グラフ 層別帯グラフ モザイクプロット ヒートマップ	モザイクプロット行列

以下では、量的データを主な対象とし、可視化・観察・分析する手法のいくつかについて説明していく。

3.1.1 変数 量的データの可視化

3.1.1.1 ヒストグラム

ヒストグラムは度数分布表を棒グラフで表現したものであり、横軸の棒が階級を表し、幅が階級幅、面積が度数を表す。通常、棒の幅は等しく設定されるが、場合によっては異なる

幅を用いる場合もある。ヒストグラムを作成する際には、適切な階級数の設定が必要であり、階級数が少ないと全体像を理解しにくいが、逆に多すぎても全体像を把握しにくくなる。適切な階級数を決定するためには、スタージェスの公式がよく用いられる [2]。ここで n を対象の数であるとすると、階級の数 k は以下で求めることができます。

$$k = \log_2 n + 1$$

4. 基本的なデータ可視化手法：ヒストグラム

量的変数データの可視化の方法として、ヒストグラムがあります（参照→1-3）。

これは、変数を適当な範囲で区切ってその範囲に入る数を集計した度数分布表を棒グラフで表したもので、範囲の幅は、基本的には分析者が決めます。

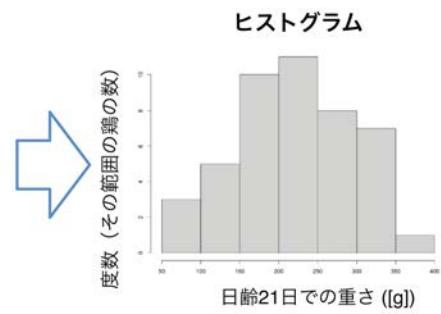
ヒストグラムに見られるようなデータの分布の位置と広がり具合については、それぞれ代表値と標準偏差によって定量的に表すことができます（後述）。

(日齢21日の鶏体重の元データ)

重さ[g]	日齢	個のID	個の確率
205	21	1	1
315	21	2	1
202	21	3	1
157	21	4	1
223	21	5	1
157	21	6	1
305	21	7	1
98	21	9	1
124	21	10	1
175	21	11	1
205	21	12	1
96	21	13	1
266	21	14	1
142	21	17	1
157	21	19	1
117	21	20	1
331	21	21	2
162	21	22	2
175	21	23	2
74	21	24	2
265	21	25	2
251	21	26	2
192	21	27	2
233	21	28	2
309	21	29	2
153	21	30	2
256	21	31	2
305	21	32	3
147	21	33	3
341	21	34	3

度数分布表

重さの階級	度数
0g~50g	0
50g~100g	3
100g~150g	5
150g~200g	10
200g~250g	11
250g~300g	8
300g~350g	7
350g~400g	1
400g 以上	0



21日目の鶏の重さが大体どれくらいで、どのように散らばっているかが把握できる

9

3.1.2 Pythonによるヒストグラム生成の実装例

```
In [21]: # オリジナルのCSVファイルをpandasで読み込んでデータフレームdfに格納
import pandas as pd
df = pd.read_csv('ChickWeight_original.csv')
df.columns = ['weight', 'Time', 'Chick', 'Diet']
## df.head()
```

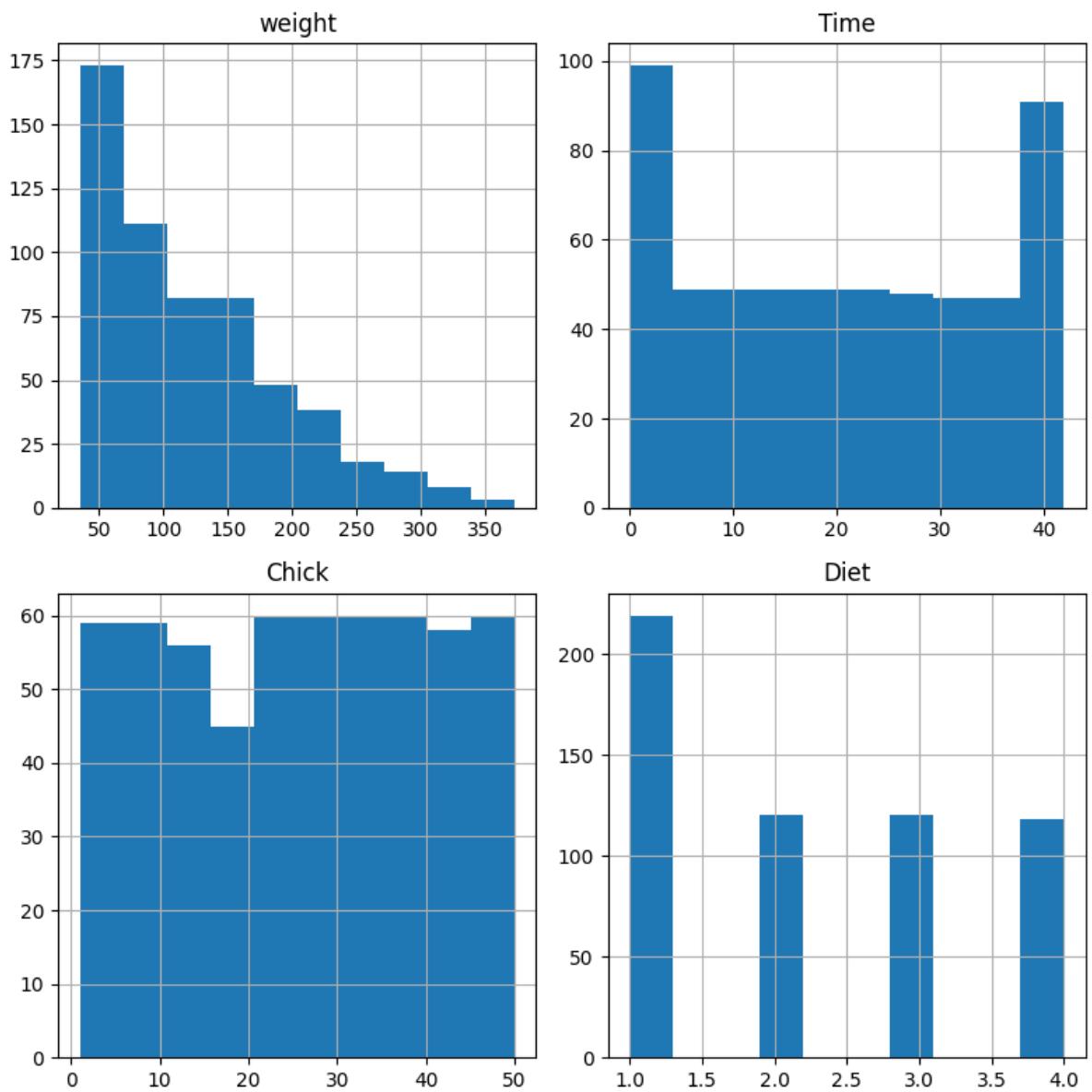
```
In [22]: # スタージェスの公式によって、サンプルデータ ChickWeight の 階級数 $k$ を求めてみる
import math # mathライブラリをインポート

n = len(df) # データの個数を取得 i.e. データフレームdfの行数
print(n) # データの個数を表示

k = math.log(n, 2) # 2を底にした対数関数の値を求める
print('{:.3f}'.format(k)) # 小数点以下3位まで出力
```

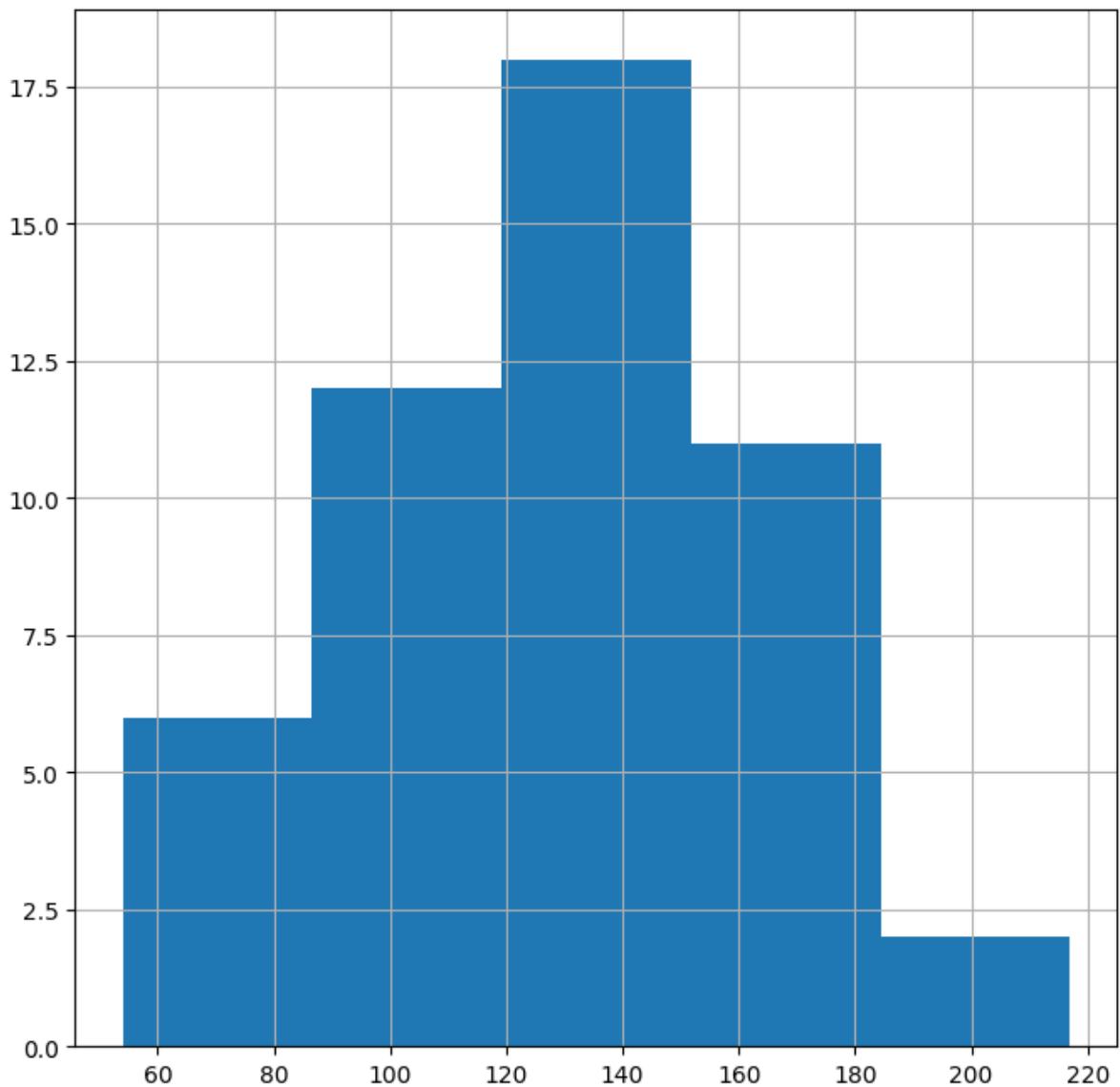
577
9.172

```
In [23]: # 4パラメータ全てで度数を観察してみる
import matplotlib.pyplot as plt
from pylab import rcParams
rcParams['figure.figsize'] = 8, 8
df.hist(); # 一括でヒストグラムを描画する(説明変数の指定無し)
plt.tight_layout() # グラフ同士が重ならないようにする関数
plt.show() # グラフの表示
```



```
In [24]: # 階級数(基數)の数をkに変更し、具体的な日齢で抽出した度数を見てみる
```

```
df_select = df[df['Time'] == 24] # 日齢 24日の場合(変更してみよ 4の倍数であることに注意)
df_select['weight'].hist(bins=int(math.log(len(df_select), 2))); # 階級数k, 基數2
plt.show()
```



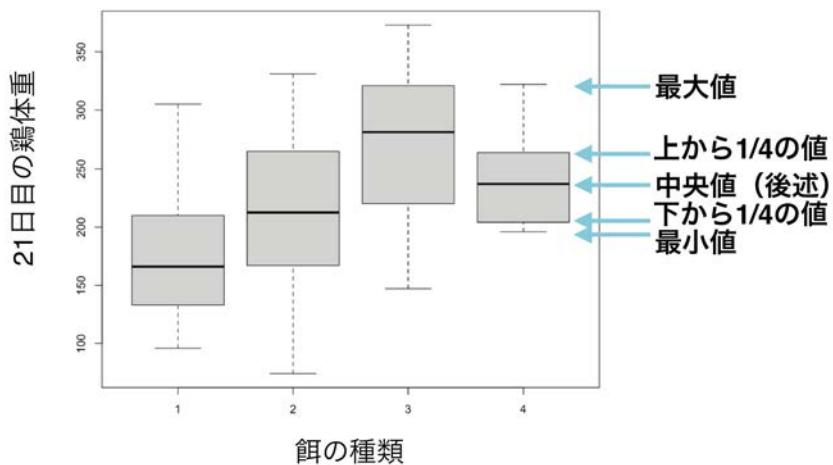
3.1.3 箱ひげ図

箱ひげ図は、データの最小値、第1四分位数、中央値、第3四分位数、最大値の5つの情報を可視化したグラフで、通常はひげが最小値と最大値を表し、箱の両端が四分位数を表す。ひげの長さは箱の長さの1.5倍として、ひげの外側のデータは外れ値として点で表されることもある。箱の中に中央値を示すことが一般的であるが、平均値を追加して表示することもある[2]。

4. 基本的なデータ可視化手法：箱ひげ図

分布同士を比較するのには、箱ひげ図が便利です。（参照→1-5）

これら各条件間に違いがあるといって良い（有意である）かどうかについて検証するには、**区間推定や仮説検定**の考え方を理解する必要があります（参照→1-6）。



東京大学 数理・情報教育研究センター 島田 尚 2021 CC BY-NC-SA

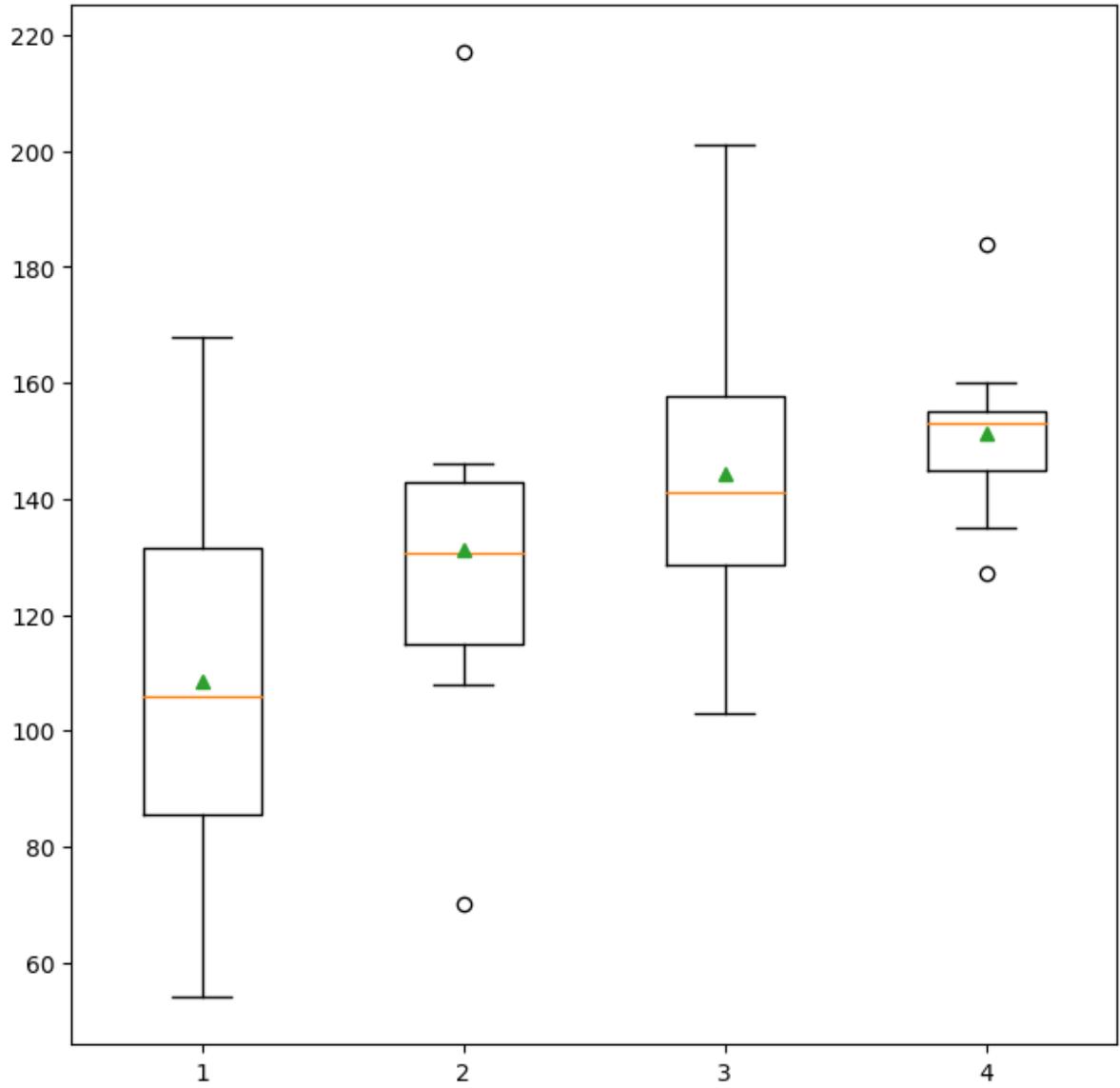
10

3.1.4 Pythonによる箱ひげ図生成の実装例

```
In [25]: # 日齢で抽出したdf_selectを使って、具体的な餌の種類 Diet毎の箱ひげ図を描画してみる(Diet毎に
df_Diet1 = df_select[df_select['Diet'] == 1]
df_Diet2 = df_select[df_select['Diet'] == 2]
df_Diet3 = df_select[df_select['Diet'] == 3]
df_Diet4 = df_select[df_select['Diet'] == 4]

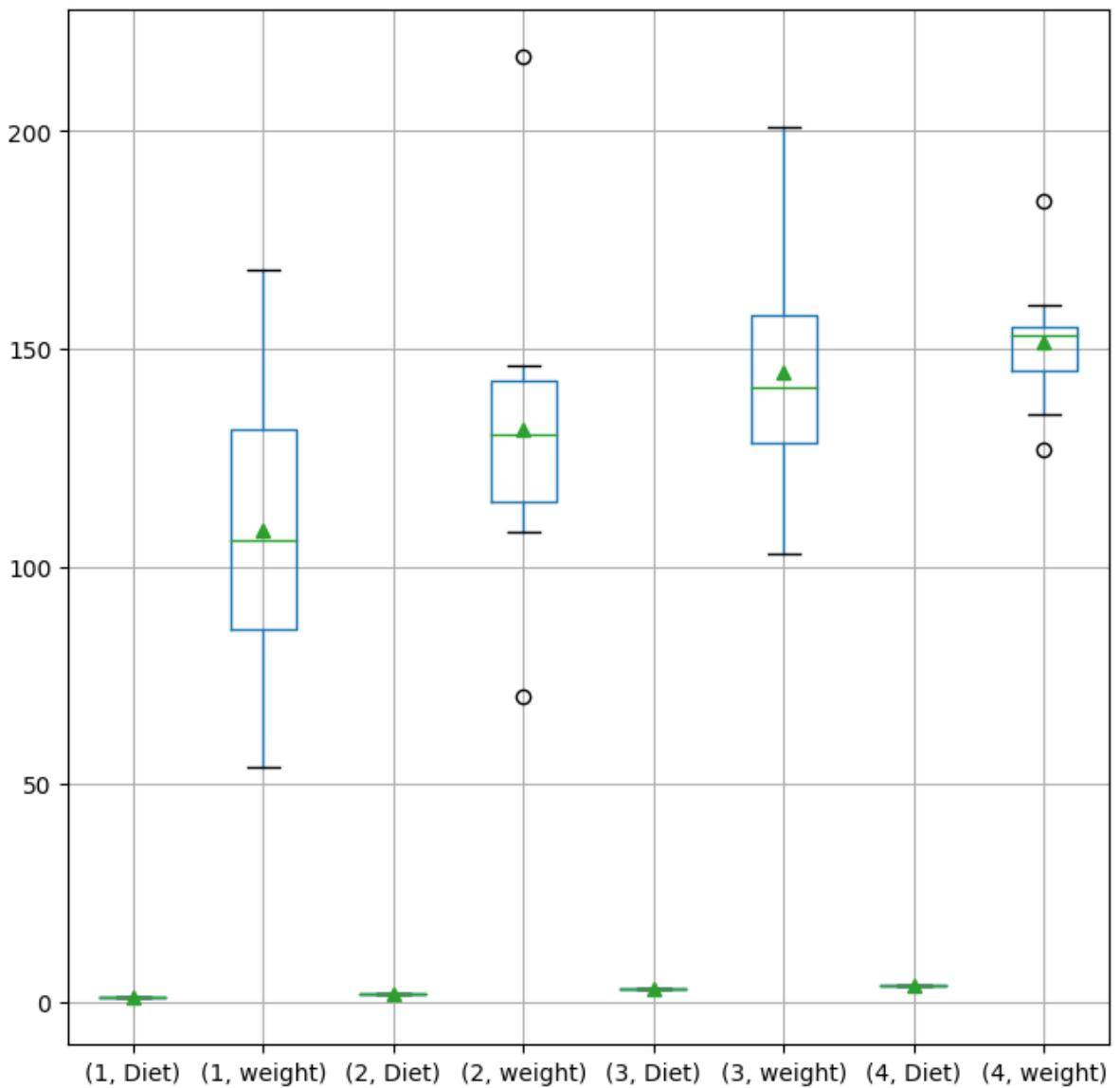
plt.boxplot([df_Diet1['weight'], df_Diet2['weight'], df_Diet3['weight'], df_Diet4['weight']])
```

```
Out[25]: {'whiskers': [<matplotlib.lines.Line2D at 0x7ff0787bb520>,
 <matplotlib.lines.Line2D at 0x7ff078759880>,
 <matplotlib.lines.Line2D at 0x7ff078754e80>,
 <matplotlib.lines.Line2D at 0x7ff0787a9ee0>,
 <matplotlib.lines.Line2D at 0x7ff0787a3640>,
 <matplotlib.lines.Line2D at 0x7ff0787a0400>,
 <matplotlib.lines.Line2D at 0x7ff0787b2eb0>,
 <matplotlib.lines.Line2D at 0x7ff0787b6580>],
 'caps': [<matplotlib.lines.Line2D at 0x7ff078759a90>,
 <matplotlib.lines.Line2D at 0x7ff078759d30>,
 <matplotlib.lines.Line2D at 0x7ff0787a9190>,
 <matplotlib.lines.Line2D at 0x7ff0787a97f0>,
 <matplotlib.lines.Line2D at 0x7ff0787a0820>,
 <matplotlib.lines.Line2D at 0x7ff0787a0ee0>,
 <matplotlib.lines.Line2D at 0x7ff0787b61c0>,
 <matplotlib.lines.Line2D at 0x7ff0787b6430>],
 'boxes': [<matplotlib.lines.Line2D at 0x7ff0787bbdc0>,
 <matplotlib.lines.Line2D at 0x7ff078754af0>,
 <matplotlib.lines.Line2D at 0x7ff0787a3dc0>,
 <matplotlib.lines.Line2D at 0x7ff0787b2490>],
 'medians': [<matplotlib.lines.Line2D at 0x7ff078754dc0>,
 <matplotlib.lines.Line2D at 0x7ff0787a9e80>,
 <matplotlib.lines.Line2D at 0x7ff0787a0d60>,
 <matplotlib.lines.Line2D at 0x7ff0787b67f0>],
 'fliers': [<matplotlib.lines.Line2D at 0x7ff078754580>,
 <matplotlib.lines.Line2D at 0x7ff0787a39a0>,
 <matplotlib.lines.Line2D at 0x7ff0787b20d0>,
 <matplotlib.lines.Line2D at 0x7ff0787cd880>],
 'means': [<matplotlib.lines.Line2D at 0x7ff078754130>,
 <matplotlib.lines.Line2D at 0x7ff0787a9640>,
 <matplotlib.lines.Line2D at 0x7ff0787b2b80>,
 <matplotlib.lines.Line2D at 0x7ff0787cd100>]}
```



```
In [26]: # 日齢で抽出したdf_selectを使って、餌の種類 Diet毎にgroupbyを使って自動的に生成
grouped = df_select[['Diet','weight']].groupby('Diet')
##grouped.groups
grouped.boxplot(subplots=False, showmeans=True)
```

Out[26]: <Axes: >



3.2 2変数・多変数量的データの可視化

3.2.1 散布図

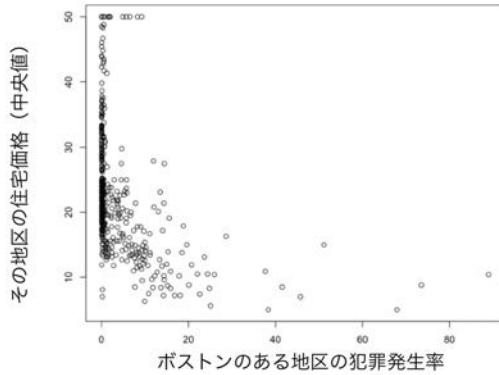
量的な変数の可視化には、散布図または散布図行列がよく使われる。散布図は、2つの量的データを2次元空間にプロットしたもので、2つの変数間の関係を把握するために使用される。散布図は相関係数と密接な関係があり、主に相関関係の確認に使用されるが、外れ値の存在などデータの特徴を把握するためにも非常に有用である。

散布図行列は、複数の変数の組み合わせに対して散布図を作成する方法で、2次元空間に収まらない多次元データを可視化するために使用される。散布図行列には、各変数のヒストグラムが表示され、各変数間の相関関係を一度に確認することができる。

これらの可視化法は、データの特徴を直感的に理解するために非常に有用であり、データ解析の初期段階でよく使われまる。ただし、これらの手法には欠点もあり、例えば、散布図では非線形な関係を捉えることができないといった点がある。

4. 基本的なデータ可視化手法：散布図

量的変数項目同士の関係の可視化の方法として、散布図があります。これは関係を見たい二つの項目の値をそれぞれの軸にとってプロットしたものです。（参照→1-3, 1-5）例えば下の図の例では犯罪発生率が高い地区ほど住宅価格が低い傾向が見て取れますが、このような相関は後述の相関係数などで定量的に評価できます。



(元データ) Harrison, D. and Rubinfeld, D.L. (1978) Hedonic prices and the demand for clean air. *J. Environ. Economics and Management* 5, 81–102.
Belsley D.A., Kuh, E. and Welsch, R.E. (1980) Regression Diagnostics. Identifying Influential Data and Sources of Collinearity. New York: Wiley.

東京大学 数理・情報教育研究センター 島田 尚 2021 CC BY-NC-SA

11

3.2.2 Pythonによる scikit-learn カリフォルニア住宅価格データセットの散布図生成

【参考】 [scikit-learn カリフォルニア住宅価格データの前処理に関する見解](#)

In [27]: # scikit-learnからカリフォルニア住宅価格データ ($n = 20640$) を取得し、DESCRキーの情報を確認

```
import pandas as pd
import seaborn as sns
from sklearn.datasets import fetch_california_housing
dataset = fetch_california_housing()
print(dataset.DESCR)
```

```
.. _california_housing_dataset:  
California Housing dataset  
-----  
**Data Set Characteristics:**  
:Number of Instances: 20640  
:Number of Attributes: 8 numeric, predictive attributes and the target  
:Attribute Information:  
- MedInc median income in block group  
- HouseAge median house age in block group  
- AveRooms average number of rooms per household  
- AveBedrms average number of bedrooms per household  
- Population block group population  
- AveOccup average number of household members  
- Latitude block group latitude  
- Longitude block group longitude  
:Missing Attribute Values: None  
  
This dataset was obtained from the StatLib repository.  
https://www.dcc.fc.up.pt/~ltorgo/Regression/cal\_housing.html  
  
The target variable is the median house value for California districts,  
expressed in hundreds of thousands of dollars ($100,000).  
  
This dataset was derived from the 1990 U.S. census, using one row per censu  
s  
block group. A block group is the smallest geographical unit for which the  
U.S.  
Census Bureau publishes sample data (a block group typically has a populati  
on  
of 600 to 3,000 people).  
  
A household is a group of people residing within a home. Since the average  
number of rooms and bedrooms in this dataset are provided per household, th  
ese  
columns may take surprisingly large values for block groups with few househ  
olds  
and many empty houses, such as vacation resorts.  
  
It can be downloaded/loaded using the  
:func:`sklearn.datasets.fetch_california_housing` function.  
  
.. topic:: References  
  
- Pace, R. Kelley and Ronald Barry, Sparse Spatial Autoregressions,  
  Statistics and Probability Letters, 33 (1997) 291–297
```

In [28]: # DataFrameで実データを確認

```
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)  
df['Price'] = dataset.target  
df.head()
```

Out [28]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

In [29]: # 基本情報を確認

```
df.describe()
```

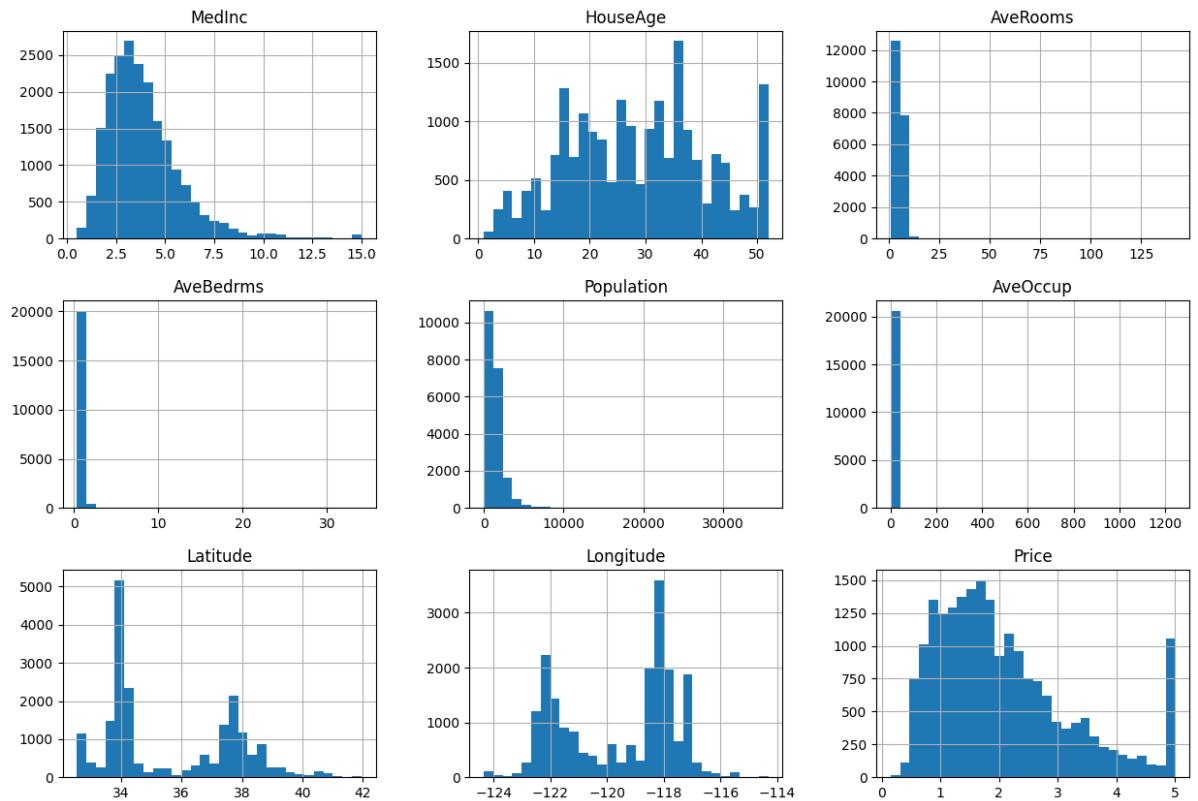
Out [29]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070000
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.380000
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.690000
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.420000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.810000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.280000
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.330000

In [30]: # 各カラムの分布をヒストグラムで確認

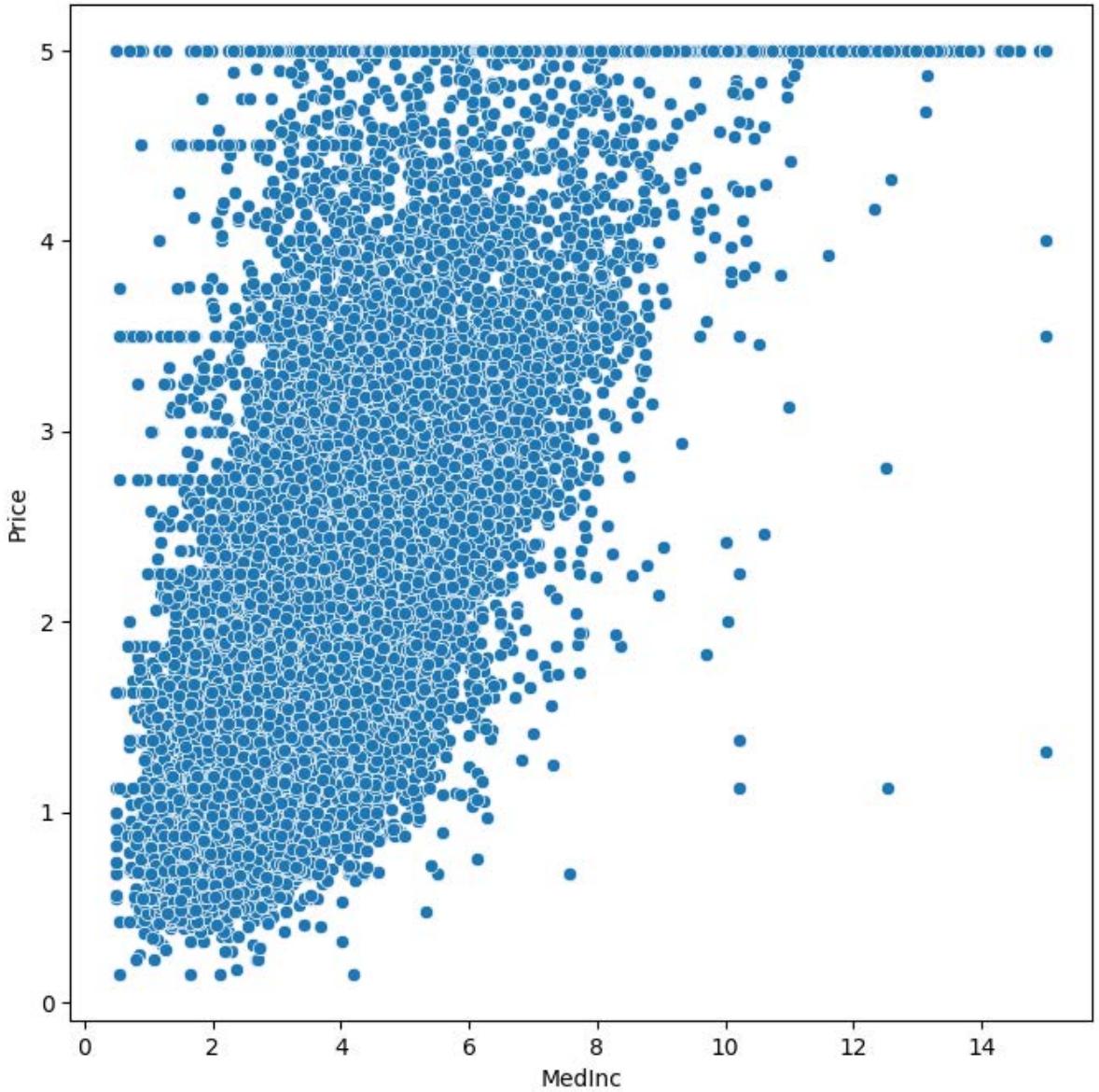
```
df.hist(bins=30, figsize=(15, 10))
```

Out [30]: array([[<Axes: title={'center': 'MedInc'}>,<Axes: title={'center': 'HouseAge'}>,<Axes: title={'center': 'AveRooms'}>], [<Axes: title={'center': 'AveBedrms'}>,<Axes: title={'center': 'Population'}>,<Axes: title={'center': 'AveOccup'}>], [<Axes: title={'center': 'Latitude'}>,<Axes: title={'center': 'Longitude'}>,<Axes: title={'center': 'Price'}>]], dtype=object)



```
In [31]: # MedInc(プロックの所得中央値), Price(住宅価格) の間で散布図を生成してみる (Seabornライブラリ)
sns.scatterplot(data=df, x='MedInc', y='Price')
```

```
Out[31]: <Axes: xlabel='MedInc', ylabel='Price'>
```



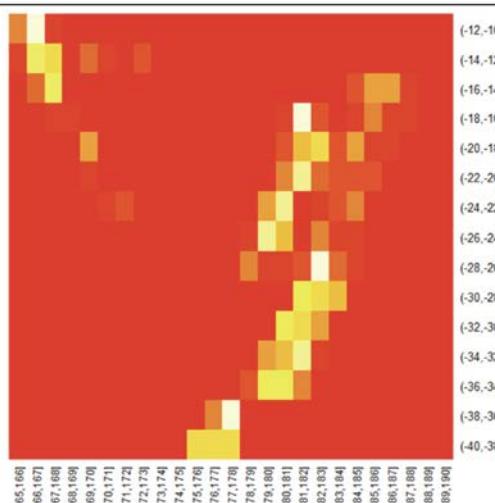
3.2.3 ヒートマップ

ヒートマップは、データの値を色や濃淡で表現したもので、クロス集計されたデータや行列の形で表される。また、地図などの空間に置かれた値の大きさを色や濃淡で表す場合もある。色を使って表現することで、数値で並べるよりも直感的に把握しやすくなる。

4. 基本的なデータ可視化手法：

ヒートマップ (参照→1-5)

- 散布図においてデータの密度を色を変えて表現したものをヒートマップといいます。
- 右図はフィジーの地震データを緯度と経度をもとに、地震の発生頻度を色を分けて表現しています。
※白に近い色がより地震の頻度が高いです。



東京大学 数理・情報教育研究センター 荻原哲平 2020 CC BY-NC-SA

12

3.2.4 Pythonによる住宅価格を推定する問題のトレーニングデータ(Kaggle)のヒートマップ図生成

【参考】 [Kaggle | House Prices - Advanced Regression Techniques](#)

```
In [32]: # CSVデータを カレントディレクトリ直下のフォルダ(一時作業領域)へダウンロードする
!wget -nc https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/
# wgetしなくても,Google colab.の左メニュー [ファイル] アイコンをクリックして,ブラウザへファイルを
# ファイル (house_prices_train.csv) がダウンロード・配置できることを確認する
##!ls -al ./
--2023-04-02 06:34:14-- https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/resources/house_prices_train.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 460676 (450K) [text/plain]
Saving to: 'house_prices_train.csv'

house_prices_train. 100%[=====] 449.88K --.-KB/s    in 0.03s

2023-04-02 06:34:15 (15.1 MB/s) - 'house_prices_train.csv' saved [460676/460676]
```

```
In [33]: # CSVファイルをpandasで読み込んでデータフレームdfに格納
import pandas as pd
df_house = pd.read_csv('house_prices_train.csv', index_col=0)

# pandas.DataFrameのメソッドcorr()を使うと, pandas.DataFrameの各列の間の相関係数を算出で
df_house_corr = df_house.corr()
```

```
print(df_house_corr.shape)
print(df_house_corr.head())
```

(37, 37)

```
MSSubClass  1.000000  -0.386347 -0.139781  0.032628 -0.059316 \
LotFrontage -0.386347  1.000000  0.426095  0.251646 -0.059213
LotArea     -0.139781  0.426095  1.000000  0.105806 -0.005636
OverallQual  0.032628  0.251646  0.105806  1.000000 -0.091932
OverallCond  -0.059316 -0.059213 -0.005636 -0.091932  1.000000

... \
MSSubClass  0.027850  0.040581  0.022936 -0.069836 -0.065649
...
LotFrontage  0.123349  0.088866  0.193458  0.233633  0.049900
...
LotArea     0.014228  0.013788  0.104160  0.214103  0.111170
...
OverallQual  0.572323  0.550684  0.411876  0.239666 -0.059119
...
OverallCond -0.375983  0.073741 -0.128101 -0.046231  0.040229
...

... \
WoodDeckSF   OpenPorchSF EnclosedPorch 3SsnPorch ScreenPorch
\
MSSubClass -0.012579 -0.006100 -0.012037 -0.043825 -0.026030
LotFrontage  0.088521  0.151972  0.010700  0.070029  0.041383
LotArea     0.171698  0.084774 -0.018340  0.020423  0.043160
OverallQual  0.238923  0.308819 -0.113937  0.030371  0.064886
OverallCond -0.003334 -0.032589  0.070356  0.025504  0.054811

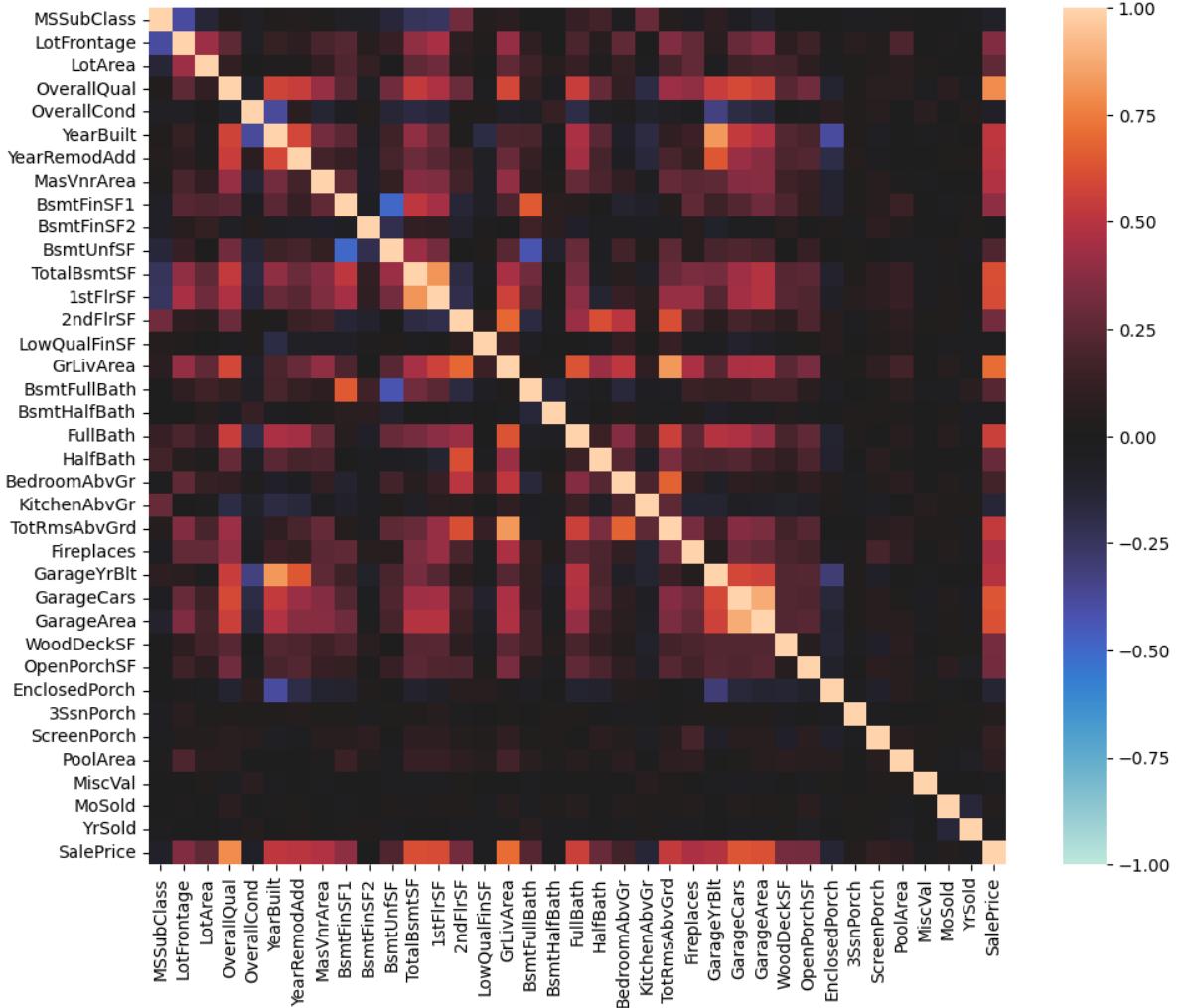
PoolArea    MiscVal    MoSold   YrSold  SalePrice
MSSubClass  0.008283 -0.007683 -0.013585 -0.021407 -0.084284
LotFrontage  0.206167  0.003368  0.011200  0.007450  0.351799
LotArea     0.077672  0.038068  0.001205 -0.014261  0.263843
OverallQual  0.065166 -0.031406  0.070815 -0.027347  0.790982
OverallCond -0.001985  0.068777 -0.003511  0.043950 -0.077856
```

[5 rows x 37 columns]

In [34]: # 各説明変数の相関係数のプロットをヒートマップで表現してみる (Seabornライブラリ利用)。相関が大きい

```
fig, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(df_house_corr, square=True, vmax=1, vmin=-1, center=0)
##plt.savefig('seaborn_heatmap_house_price.png')
```

Out[34]: <Axes: >



3.3 分割表とクロス集計

3.3.1 分割表とクロス集計について

分割表とは、2つの変数の交差表を1つの変数で分割することで、より詳細な情報を得るための表のことである。例えば、男女と年齢層をクロス集計した場合、男女別、さらに年齢層別に分割して集計することができる。このようにすることで、より詳細な情報を得ることができる。

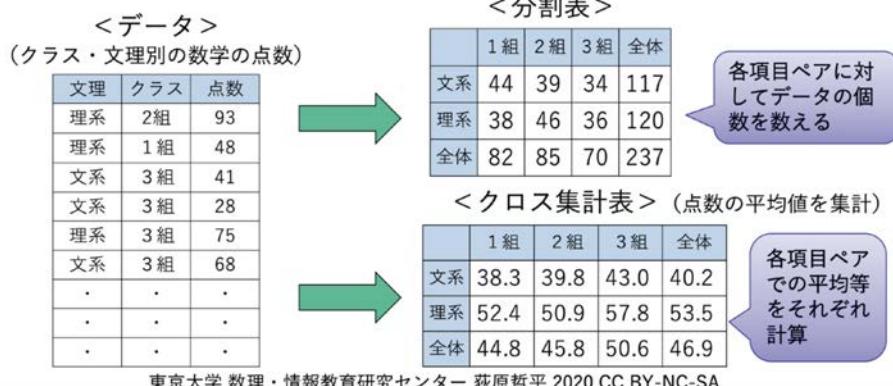
一方、クロス集計は、2つ以上の変数の関係を分析するために用いられる方法である。たとえば、商品の売り上げデータを地域別、性別別に分類した場合、地域別と性別別の2つの要因を同時に分析することができる。このとき、クロス集計によって、地域と性別の2つの変数が交差した集計表が作成される。

4. 基本的なデータ可視化手法：分割表

質的変数項目同士の関係性をつかむためには分割表とクロス集計表が便利です。

(参照→1-3)

- データの2種類の項目について、項目の値のペア毎にデータの個数を数え、表にまとめたものが「分割表」、さらに2種類の項目の値のペア毎（例えば右下表で「文系」と「2組」のペア等）に別の項目（右下表では「点数」）の合計、平均、標準偏差等を集計したものが「クロス集計表」と呼ばれます。



13

3.3.1 Python crosstabを使用した分割表、クロス集計の実装例

`pandas.crosstab` は、クロス集計表を作成するための関数である。

【参考】 [pystyle | pandas – crosstab を使ってクロス集計表を作成する方法](#)

```
In [35]: # サンプルデータ「飲食店でのチップの支払い額を記録したデータ」を読み込み (n = 244)
import pandas as pd

# CSVデータを カレントディレクトリ直下のフォルダ(一時作業領域)へダウンロードする。
!wget -nc https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/

# wgetしなくとも、以下のGitHubから直接読み込んでも可
#df = pd.read_csv("https://git.io/JJlHw")

df = pd.read_csv('JJlHw.csv')
df.head()

--2023-04-02 06:34:16-- https://raw.githubusercontent.com/MDASH-shinshu/MDASH-T-DS/main/2/resources/JJlHw.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 7943 (7.8K) [text/plain]
Saving to: 'JJlHw.csv'

JJlHw.csv          100%[=====] 7.76K --.-KB/s   in 0s

2023-04-02 06:34:16 (81.9 MB/s) - 'JJlHw.csv' saved [7943/7943]
```

```
Out[35]:    total_bill    tip      sex smoker   day    time  size
0        16.99  1.01  Female     No  Sun  Dinner     2
1        10.34  1.66   Male     No  Sun  Dinner     3
2        21.01  3.50   Male     No  Sun  Dinner     3
3        23.68  3.31   Male     No  Sun  Dinner     2
4        24.59  3.61 Female     No  Sun  Dinner     4
```

```
In [36]: # 性別:時間帯別の分割表を作成
```

```
ret = pd.crosstab(df["sex"], df["time"], margins=True)
ret
```

```
Out[36]:    time  Dinner  Lunch   All
              sex
Female      52      35    87
Male       124      33   157
All        176      68   244
```

```
In [37]: # 性別:時間帯に加えて曜日別の分割表を作成
```

```
ret = pd.crosstab(df["sex"], [df["day"], df["time"]], margins=True)
ret
```

```
# 土日はDinner、平日はランチでのチップ支払いが多い
```

```
Out[37]:    day          Fri   Sat    Sun        Thur   All
              time  Dinner  Lunch  Dinner  Dinner  Dinner  Lunch
              sex
Female      5      4    28    18      1    31    87
Male       7      3    59    58      0    30   157
All        12     7    87    76      1    61   244
```

```
In [38]: # normalize=Trueまたはnormalize='all'とすると、全体を合計すると1になるように規格化する
```

```
ret = pd.crosstab([df['sex'], df['smoker']], [df['day'], df['time']], margins=True)
```

```
# 正規化したクロス集計表によると、土日のDinnerで全体の66%以上のチップが得られている。
```

```
# さらに、土日のDinnerで男性の非喫煙者の割合払ったチップの割合が相当大きい(全体の30%以上)であ
```

Out [38] :		day	Fri	Sat	Sun	Thur	All	
	time	Dinner	Lunch	Dinner	Dinner	Dinner	Lunch	
sex	smoker							
Female	No	0.004098	0.004098	0.053279	0.057377	0.004098	0.098361	0.221311
	Yes	0.016393	0.012295	0.061475	0.016393	0.000000	0.028689	0.135246
Male	No	0.008197	0.000000	0.131148	0.176230	0.000000	0.081967	0.397541
	Yes	0.020492	0.012295	0.110656	0.061475	0.000000	0.040984	0.245902
All		0.049180	0.028689	0.356557	0.311475	0.004098	0.250000	1.000000

4. 基本統計量と相関係数 [2]

4.1 代表値

基本統計量は、データの集合を要約するために使用される統計学的手法のことである。の中でも代表値は、データ全体を一つの数値で代表するための方法で、平均値、中央値、最頻値が代表的なものとなる。

代表値は、単独で使用するよりも、複数の代表値を比較することでデータの傾向をより正確に分析することができる。また、分布の偏りを評価する際にも使用される。

4.1.1 平均値、中央値、最頻値

- 平均値は、全てのデータを合計して、データ数で割った値である。すべてのデータを均等に扱うため、外れ値に弱いという特徴がある。
- 中央値は、データを大きさの順に並べ、中央に位置する値である。データの大きさに関係なく、データ全体の真ん中の値を代表することができる。
- 最頻値は、データの中で最も頻繁に出現する値である。頻度分布表を作成することで、最頻値を求めることができる。

5. 基本的なデータ分析手法：代表値

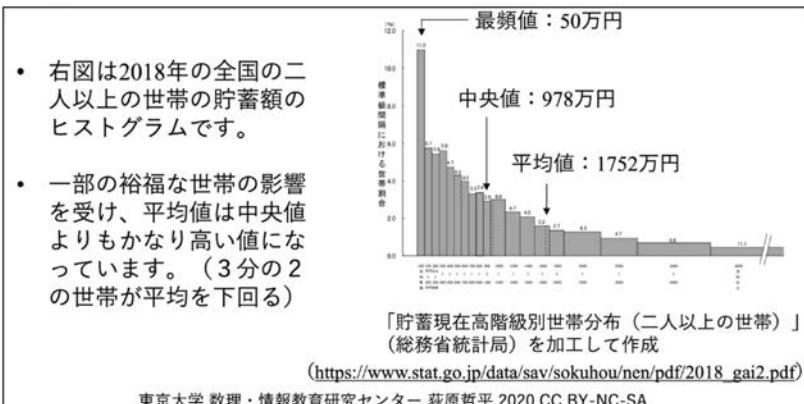
データの分布の位置を表す基本的な量として、可視化の説明で既に用いた平均値と中央値に加えて最頻値があり、これらは代表値と呼ばれます。

$$\cdot \text{平均値} : \bar{X} = \frac{X_1 + X_2 + \cdots + X_n}{n}$$

- ・中央値：データを大きい順もしくは小さい順に並べた時に真ん中に位置する値（データ数が偶数なら真ん中2つの値の平均）
- ・最頻値：データの中で最も多く現れた値。変数が連続値の場合や、整数でもデータ数に比べて範囲が広い場合などは度数が最大の範囲の中央の値。

5. 基本的なデータ分析手法：代表値

分布が平均値まわりで大体左右対称な山をつくっているような形状の場合は三つの代表値は同じような値になりますが、下の例のように現実の分布などではそれぞれ異なる場合があることに注意が必要です。はじめて解析するデータについては、代表値の違いが大きいかどうかや分布の形状そのものを確認してみましょう。前述の箱ひげ図は、代表値に加えてこのような分布形状の情報を残すための可視化手法と言えます。



- e-Stat | 家計調査 貯蓄・負債編 二人以上の世帯 詳細結果表(2021)
- 統計WEB | ジニ係数

4.1.2 分散、標準偏差、偏差値（特殊）

5. 基本的なデータ分析手法：分散と標準偏差

データの分布の広がりを表す量としては、分散と標準偏差が基本です。

$$\text{・分散} : \sigma^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n} = \frac{(X_1 - \bar{X})^2 + (X_2 - \bar{X})^2 + \cdots + (X_n - \bar{X})^2}{n}$$

$$\text{・標準偏差} : \sigma = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n}}$$

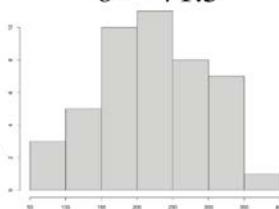
標準偏差は元のデータの次元の量で、ヒストグラムがひと山の形の場合は右下の鶏体重の例のように、代表値まわりの分布の幅に対応します。 (鶏の重さの例)

分布が多数の山を持つ場合や、貯蓄金額の例のように

裾が広い形の場合はこの限りではありません。

また、分布の中でのデータの相対的位置の指標として

$$\text{・偏差値} : 50 + 10 \left(\frac{X_i - \bar{X}}{\sigma} \right) \text{が使われることがあります。}$$



東京大学 数理・情報教育研究センター 島田 尚 2021 CC BY-NC-SA

16

4.1.3 ChickWeightサンプルデータの要約統計量

```
In [39]: # オリジナルのCSVファイルをpandasで読み込んでデータフレームdfに格納
import pandas as pd
df = pd.read_csv('ChickWeight_original.csv')
df.columns = ['weight', 'Time', 'Chick', 'Diet']
##df.head()
```

```
In [40]: # 日齢でデータ抽出し、要約統計量を表示してみる
```

```
df_select = df[df['Time'] == 24] # 日齢 24日の場合(変更してみよ 4の倍数であることに注意)
df_select.describe()
# count:個数、mean:算術平均、std:標準偏差、min-max:最小値-最大値、25%-50%-75%:各四分位
```

```
Out[40]:
```

	weight	Time	Chick	Diet
count	49.000000	49.0	49.000000	49.000000
mean	129.244898	24.0	25.653061	2.224490
std	34.119600	0.0	14.687794	1.177308
min	54.000000	24.0	1.000000	1.000000
25%	108.000000	24.0	13.000000	1.000000
50%	134.000000	24.0	26.000000	2.000000
75%	152.000000	24.0	38.000000	3.000000
max	217.000000	24.0	50.000000	4.000000

4.2 相関と相関係数

相関とは、二つの変数の間に関係があることを示す統計学上の用語で、一方の変数が変化すると他方の変数も変化する傾向があることを指す。相関係数は、相関の強さと方向性を数値

化したもので、-1から1の範囲の値をとる。ピアソンの相関係数は、二つの変数が線形関係にある場合に用いられる相関係数で、広く使われている。

相関があるからといって必ずしも因果関係があるわけではない。相関は単に関連性を示すものであり、因果関係を示すものではないからである。相関がある場合でも、それが偶然の一一致である可能性もある。

因果関係を明らかにするためには、実験的手法を用いて要因と結果の因果関係を（科学的に）証明する必要がある。因果関係を示すためには、相関だけでなく、その他の要因も含めた網羅的な分析が必要なのである。

4.2.1 ピアソンの相関係数

ピアソンの相関係数は、統計学において2つの変数間の線形関係の強さを表す指標の1つであり、-1から1までの値を取る。この指標は、カール・ピアソンによって開発され、広く使用されている（定義は下のスライド参照）。

ピアソンの相関係数は、以下のような性質を持ちます。

- $r \geq 1$ に近づくほど、 X と Y の間には正の相関がある。つまり、 X が増加すると Y も増加する傾向がある
- $r \leq -1$ に近づくほど、 X と Y の間には負の相関がある。つまり、 X が増加すると Y が減少する傾向がある
- $r \approx 0$ に近づくほど、 X と Y の間には相関がなく、2つの変数は独立している

ピアソンの相関係数は、2つの変数間の線形関係を表すため、非線形関係がある場合には適切な指標ではない。データに外れ値がある場合には、相関係数の値が歪んでしまう可能性がある。

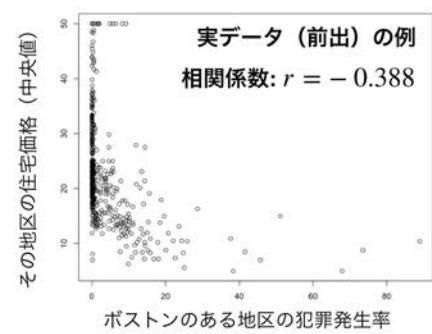
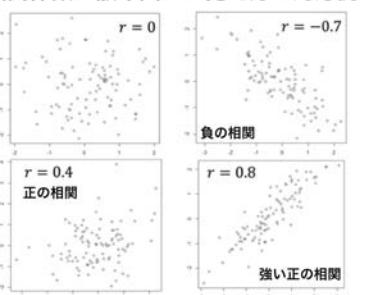
5. 基本的なデータ分析手法：相関と相関係数

散布図で可視化したような二つの項目の相関関係を表す量として、

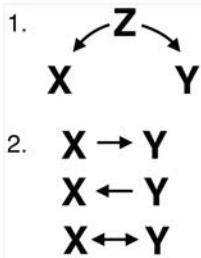
$$\text{相関係数} \text{ (ピアソンの相関係数)} : C_{XY} = \frac{\sum_{i=1}^n (X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^n (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^n (Y_i - \bar{Y})^2}}$$

が使われます。この相関係数は-1から1までの値をとり、正の相関（散布図で右肩上がりの関係）のとき正の値、負の相関（散布図で右肩下がり）のとき負の値をとります。（参照→1-6）

相関係数と散布図での見え方の対応例



5. 基本的なデータ分析手法：相関と因果

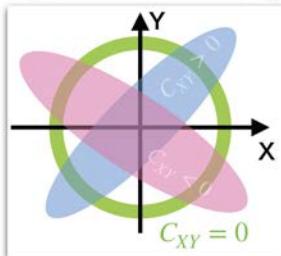


2つの量に相関があるからといってそれらの間に因果関係があるとはいえないことに注意が必要です。

一方が直接的に影響を及ぼしているとは限らず、

(1.) 共通の要因 Z (交絡要因) があるのかもしれません。そのような場合の相関を擬似相関といいます。

(2.) また、相関係数では影響の方向を測ることはできません。影響の方向を決定するには他の基準やよく設計された実験が必要です。

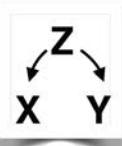


さらに、相関係数が0に近いからといって、関係性が無いとも言えないことに注意しましょう。左図の例のように、 XY の間に明らかな関係性があっても、線形に相関していない場合は相関係数は0になります。 (参照→1-6)

(*発展的) 左図のような関係性は、**相互情報量**によって検出することができます

5. 基本的なデータ分析手法：偏相関係数

交絡要因 Z について心当たりがある場合、その交絡による擬似相関を取り除いた上での2変数データ X と Y の相関関係を**偏相関係数**によって評価することができます：



$$C_{XY|Z} = \frac{C_{XY} - C_{XZ} \cdot C_{YZ}}{\sqrt{1 - C_{XZ}^2} \sqrt{1 - C_{YZ}^2}}$$

0でない偏相関係数が得られたとしても、想定していない**交絡要因**がまだ他にあるかもしれないという点には注意が必要です。

また、偏相関係数も、因果性や XY 間の影響の方向についての情報とはなりません。

4.2.2 カリフォルニア住宅価格データセット（再掲）の相関係数の計算

【参考】 [scikit-learn カリフォルニア住宅価格データの前処理に関する見解](#)

In [41]: # scikit-learnからカリフォルニア住宅価格データ ($n = 20640$) を取得

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_california_housing
```

```
dataset = fetch_california_housing()
##print(dataset.DESCR)
```

In [42]: # DataFrameで実データを確認

```
df = pd.DataFrame(dataset.data, columns=dataset.feature_names)
df['Price'] = dataset.target
df.head()
```

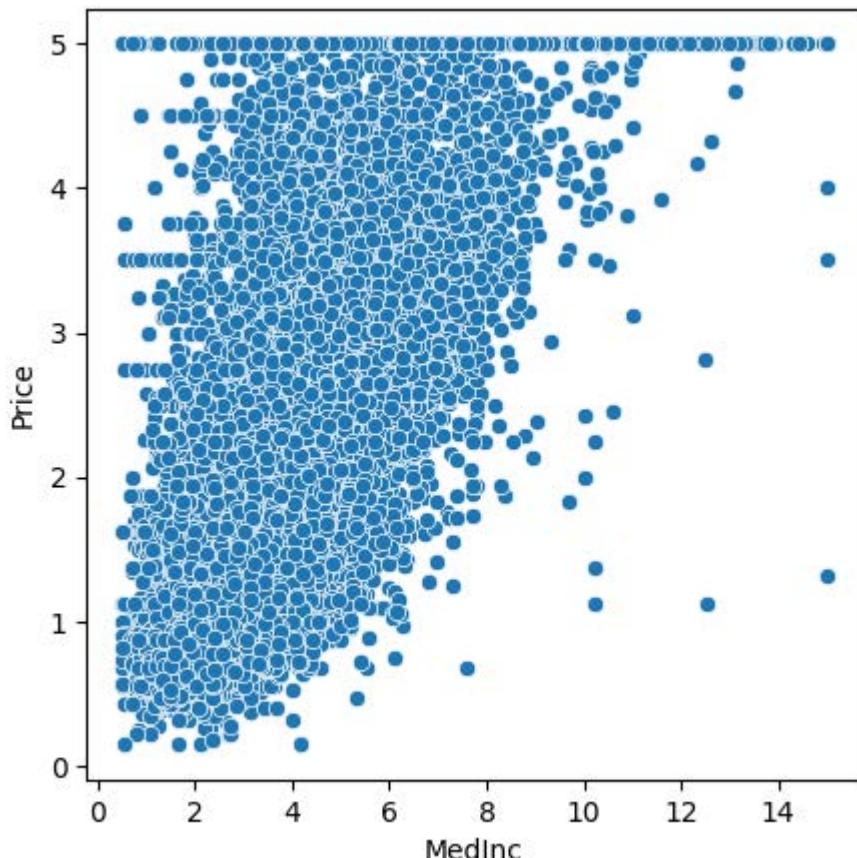
Out[42]:

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

In [43]: # MedInc(プロックの所得中央値), Price(住宅価格) の間で散布図を生成してみる (Seabornライブラリ)

```
plt.rcParams["figure.figsize"] = (5, 5)
sns.scatterplot(data=df, x='MedInc', y='Price')
```

Out[43]: <Axes: xlabel='MedInc', ylabel='Price'>



In [44]: # Pearsonの相関係数を計算(pandas corr) 2ペアのみ

```
df_house_corr=df.loc[:,['MedInc','Price']].corr()
df_house_corr
```

```
Out[44]:
```

	MedInc	Price
MedInc	1.000000	0.688075
Price	0.688075	1.000000

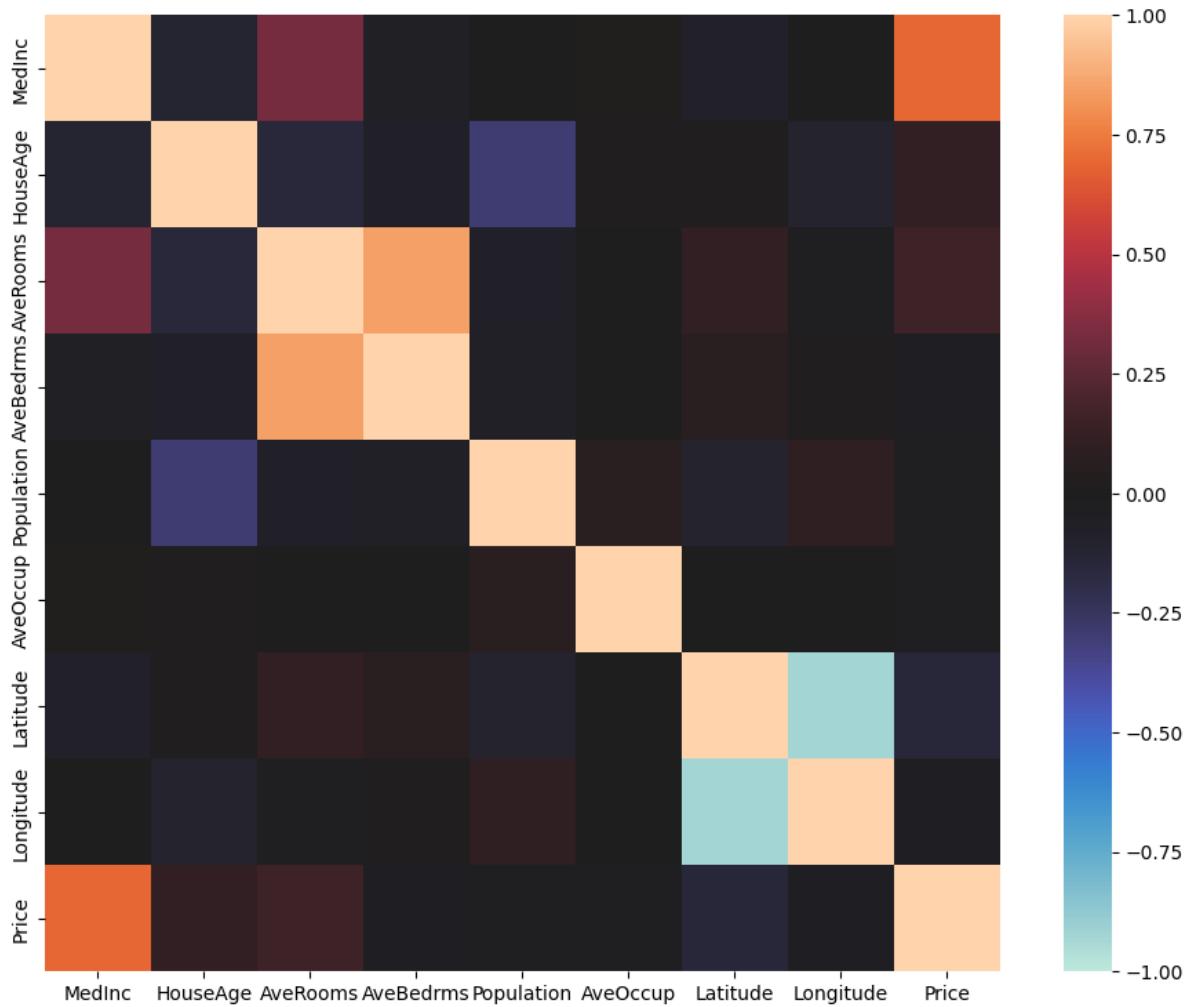
```
In [45]: # Pearsonの相関係数を計算(pandas corr) 全てのペア  
df_house_corr=df.corr()  
df_house_corr
```

```
Out[45]:
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude
MedInc	1.000000	-0.119034	0.326895	-0.062040	0.004834	0.018766	-0.079809
HouseAge	-0.119034	1.000000	-0.153277	-0.077747	-0.296244	0.013191	0.011173
AveRooms	0.326895	-0.153277	1.000000	0.847621	-0.072213	-0.004852	0.106389
AveBedrms	-0.062040	-0.077747	0.847621	1.000000	-0.066197	-0.006181	0.069721
Population	0.004834	-0.296244	-0.072213	-0.066197	1.000000	0.069863	-0.108785
AveOccup	0.018766	0.013191	-0.004852	-0.006181	0.069863	1.000000	0.002366
Latitude	-0.079809	0.011173	0.106389	0.069721	-0.108785	0.002366	1.000000
Longitude	-0.015176	-0.108197	-0.027540	0.013344	0.099773	0.002476	-0.924664
Price	0.688075	0.105623	0.151948	-0.046701	-0.024650	-0.023737	-0.144160

```
In [46]: # 各説明変数の相関係数のプロットをヒートマップで表現してみる (Seabornライブラリ利用)。相関が大きい  
fig, ax = plt.subplots(figsize=(12, 9))  
sns.heatmap(df_house_corr, square=True, vmax=1, vmin=-1, center=0)  
##plt.savefig('seaborn_heatmap_house_price.png')
```

```
Out[46]: <Axes: >
```



Price (住宅価格) と正の相関が強いのは、MedInc(ブロックの所得中央値)ぐらいしかなさそうだ。

memo