データサイエンス応用コース
Python 入門

（配列・データ操作）

下川　和郎
大阪大学

今日の予定

- ネットワーク接続確認 （5分）
- Python：他の言語との違い
- 配列データの扱い
- データ可視化
- グループワーク

講義資料のダウンロード

-------- 応用コース資料DLページ --------
https://hram.or.jp/recurrent/rec_documents/2020adv/

ID: student20adv
Pass: Rec2020adv

2021/1/21 スクーリング資料　　をクリック

Python

- 0.9x 1991年～ ABC言語の後継、教育用
- 1.0 1994年～
- 2.x 2000年～
  - 2.7.18 が最終版（サポート終了）
- 3.x 2008年～ 下位互換性は考慮していない.
  - 3.6 これをサポートしたNN関連ライブラリはまだ多い（2021年初時点）.
  - 3.9 最新?

ABC 言語

関数 words の定義

```
HOW TO RETURN words document:
    PUT {} IN collection
    FOR line IN document:
      FOR word IN split line:
        IF word not.in collection:
          INSERT word IN collection
    RETURN collection
```

Python: 高校教科書「情報 I」

高等学校情報科「情報 I」教員研修用教材

◆目　次

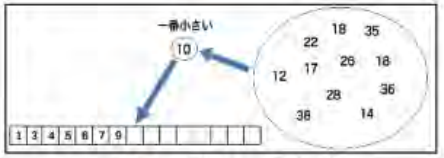
◇第３章　コンピュータとプログラミング
本単元の学習内容
学習 11　コンピュータの仕組み
学習 12　外部装置との接続
学習 13　基本的プログラム
学習 14　応用的プログラム
学習 15　アルゴリズムの比較
学習 16　確定モデルと確率モデル
学習 17　自然現象のモデル化とシミュレーション
全体を通じた学習活動の進め方

（2）ソートアルゴリズム　選択ソートとクイックソート

リストなどの中を昇順，降順に並べ替えることをソートといい，選択ソートやクイックソートなどがある。

■選択ソート

選択ソートはリスト内のデータから最小値を探索し，最小値から順に取り出すことで並べ替えを実現するアルゴリズムである。（図表８参照）



図表８　選択ソートの概念

■プログラム例

コード例を以下の図表９に示す。a はソート対象のリストである。なお a[i] と a[j] の値を入れ替える際には変数 temp を使って以下のように行っている。

1 a[i]の値を一時的に temp に入れておく(temp=a[i])
2 a[i]の値を a[j]の値に置き換える(a[i]=a[j])
3 a[j]の値を temp の値に置き換える(a[j]=temp)

```
1  def selectionsort(a):
2      for i in range(0,len(a),1):
3          for j in range(i+1,len(a),1):
4              if a[j]<a[i]:
5                  temp = a[i]
6                  a[i] = a[j]
7                  a[j] = temp
8
9  a = [7,22,11,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
10 print("ソート前",a)
11 selectionsort(a)
12 print("ソート後",a)
```

図表９　コード

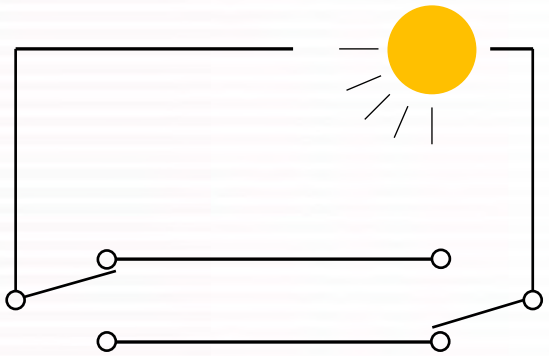
Python 動作環境と学習の進め方

- 多数のモジュールを import して動作させる.
- モジュール間のバージョンの組み合わせが問題.
  - → pip, conda, pyenv
- ある程度まとまった動作環境を提供.
  - → Anaconda, Google Colaboratory, ….
- 仮想環境を提供してその中で安定動作
  - Docker Container, Vmware, VirtualBox ….

今回は Jupiter Notebook (Anaconda) を用いる.
言語の基本的な文法については既に「応用コースⅠ」で
学んでいるため、ここでは Python の特徴を中心に解説.

プログラミングパラダイム

- 宣言型言語
  - 達成目的を指示.
  - 関数型を含む
  - 例：SQL



- 命令型言語
  - 達成方法を指示.
  - オブジェクト指向を含む
  - 例：C

ex. 廊下のスイッチ

宣言型：「廊下の灯りをつけろ」

命令型：「スイッチが下なら上へ、
スイッチが上なら下へ」

Pythonは両方の性質を備えている.

命令型（手続き型）言語はみな同じ？

For 文　　（i の初期化 , 繰り返し条件 , i 値の増減, 処理 ）

Perl

```
for ($i=0; $i<100; $i++){
    ....
}
```

Visual Basic

```
For i = 0 To 99
    ....
Next
```

C

```
for (i=1; i<100; i++){
    ....
}
```

Python

```
for i in range(100):
    ....
```

命令型（手続き型）言語はみな同じ？

While 文　　（i の繰り返し条件, 処理）

Perl

```
while($i > -1){
    ....
}
```

Visual Basic

```
While i > -1
    ....
End While
```

C

```
while (i > -1) {
    ....
}
```

Python

```
while i > -1:
    ....
```

行列、配列に関する記述

ex. 配列の定義 $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$

Python では事実上の標準ライブラリ numpy を用いる.
R では最初から組み込み関数として使える.

C, Perl では正式にサポートがされていないため演算も容易ではない.

Python

```
import numpy as np
na = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(na)

[[1 2 3]
 [4 5 6]
 [7 8 9]]
```

R

```
x <- rbind(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9))
x

 [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

配列操作などは C, Perl より R, Mathematica などに近い
Python の重要な特徴なのでここからは配列を中心に解説

行列、配列に関する記述

ex. 行列の演算

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}^{t} = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$$ 転置

Python

```
import numpy as np
na = np.array([[1,2,3],[4,5,6],[7,8,9]])
nat=np.transpose(na)
print(nat)

[[1 4 7]
 [2 5 8]
 [3 6 9]]
```

R

```
x <- rbind(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9))
t(x)
 [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

配列操作

数値と文字は混在不可

```
list01=[1,4,3,6,5] ←― 入力
list01
[1, 4, 3, 6, 5]      ←― 出力

list01.append(8)
list01
[1, 4, 3, 6, 5, 8]

list02=list01.append(2)
list02
（何も出ない. 定義されていない）

list01
[1, 4, 3, 6, 5, 8, 2]
```

```
list02=list01+[9,3,5]
list02
[1, 4, 3, 6, 5, 8, 2, 9, 3, 5]

lista=('strategy')
lista
‘strategy’

listb=lista+list01
（TypeError）

listb=lista+(' entry')
listb
'strategy entry'
```

配列操作

特定位置の値の抽出

```
list03=list(range(10))
list03
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

list04=range(10)
list04
range(0, 10)

list03[:3]
[0, 1, 2]

list03[:8]
[0, 1, 2, 3, 4, 5, 6, 7]

list03[3:]
[3, 4, 5, 6, 7, 8, 9]
```

```
list03[2]
2

list03[7]
7

list03[-1]
9

list03[-2]
8

list03[-3:]
[7, 8, 9]
```

辞書

```
dict01={'a':"Alpha", 'b':"Bravo", 'c':"Charly"}
dict01['a']
'Alpha‘

dict01['c']
‘Charly’

"c" in dict01
True

“Charly" in dict01
False
```

```
dict02={'a':1, 'b':2, 'c':3}
dict02['c’]
3

"2" in dict02
False

for j in dict02:
    print (j)
a
b
c

for j in dict02:
    print (dict02[j])
1
2
3
```

配列操作 スタックとしての操作

```
import random
list05=list(range(10))
list05
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

random.shuffle(list05)
list05
[9, 7, 1, 5, 3, 4, 0, 8, 6, 2]

list05.pop()
2

list05.pop()
6
```

```
list05
[9, 7, 1, 5, 3, 4, 0, 8]

list05.append(7)
[9, 7, 1, 5, 3, 4, 0, 8, 7]

list05.append(12)
list05
[9, 7, 1, 5, 3, 4, 0, 8, 7, 12]
```

配列操作 多次元配列

```
data_array = np.array(range(12))
data_array
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

data_array.reshape(2,6)
array([[ 0,  1,  2,  3,  4,  5],  [ 6,  7,  8,  9, 10, 11]])

data_array.reshape(3,4)
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])

data_array.reshape(4, -1)
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])
```

配列操作

```
data_array = np.array(range(12))
data_array
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9,
10, 11])

data_array.reshape(2, 3, -1)
array([[[ 0,  1],
        [ 2,  3],
        [ 4,  5]],

       [[ 6,  7],
        [ 8,  9],
        [10, 11]]])
```

多次元も可能

```
data_array2 = np.array(range(24))
data_array2
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,
13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23])

data_array2.reshape(2, 3, 2, -1)
array([[[[ 0,  1],
         [ 2,  3]],

        [[ 4,  5],
         [ 6,  7]],

        [[ 8,  9],
         [10, 11]]],


       [[[12, 13],
         [14, 15]],

        [[16, 17],
         [18, 19]],

        [[20, 21],
         [22, 23]]]])
```

配列操作 行列の転置

```
mtrx = data_array.reshape(4,3)
mtrx
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

mtrx.T                          # 転置
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

```
mtrx = data_array.reshape(4,3)
mtrx
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

mtrx.T                          # 転置
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

配列操作

```
data_array = np.array(range(12))
data_array
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])
mtrx = data_array.reshape(4,3)
mtrx
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

mtrx[1]                 # 特定の部分を取り出す
array([3, 4, 5])

mtrx[3]
array([ 9, 10, 11])

mtrx[2:]
array([[ 6,  7,  8],
       [ 9, 10, 11]])
```

```
mtrx[:2]
array([[0, 1, 2],
       [3, 4, 5]])

mtrx[:,0]
array([0, 3, 6, 9])

mtrx[:,2]
array([ 2,  5,  8, 11])

mtrx[:,[1,2]]
array([[ 1,  2],
       [ 4,  5],
       [ 7,  8],
       [10, 11]])

mtrx[:,[0,2]]
array([[ 0,  2],
       [ 3,  5],
       [ 6,  8],
       [ 9, 11]])
```

配列操作

```
mtrx
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  7,  8],
       [ 9, 10, 11]])

mtrx[2,1]
7

mtrx[2,1]=12
mtrx
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6, 12,  8],
       [ 9, 10, 11]])
```

特定の位置の値を書き換える

```
mtrx[mtrx==7]=0
mtrx
array([[ 0,  1,  2],
       [ 3,  4,  5],
       [ 6,  0,  8],
       [ 9, 10, 11]])
```

特定の値を検索して書き換える

データ操作 政府統計

参 考 表 全 国 人 口 の 推 移
Reference Time Series of Population Estimates

総 人 口 （確 定 値） Total population (Final estimates)

年 月 Year and month	月初人口 (1)(2) Population as of 1st of each month	人口増減 Population change							
		純増減 Net change		自然動態 Natural change			社会動態 Migration change		
		増減数 Number (6)+(9)	増減率(3) Rate (‰)	出生児数 Live Births	死亡者数 Deaths	自然増減 Natural change (4)-(5)	入国者数 Entries	出国者数 Exits	社会増減 Net migration (7)-(8)
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
2010年	128,057,352 (8)								
2011年	127,834,233	-223,119	-1.74	1,073,663	1,256,387	-182,724	2,685,681	2,764,665	-78,984
2012年	127,592,657	-241,576	-1.89	1,046,825	1,248,186	-201,361	2,756,710	2,835,515	-78,805
2013年	127,413,888	-178,769	-1.40	1,044,983	1,276,719	-231,736	2,796,384	2,782,006	14,378
2014年	127,237,150	-176,738	-1.39	1,022,371	1,274,085	-251,714	2,910,793	2,874,407	36,386
2015年	127,094,745 (8)	-142,405	-1.12	1,025,105	1,300,537	-275,432	3,079,784	2,985,346	94,438
2016年	126,932,772	-161,973	-1.27	1,004,068	1,299,933	-295,865	3,361,488	3,227,596	133,892
2017年	126,706,210	-226,562	-1.78	965,289	1,342,578	-377,289	3,615,119	3,464,392	150,727
2018年	126,443,180	-263,030	-2.08	944,146	1,388,632	-424,486	3,848,382	3,686,926	161,456
2019年	126,166,948	-276,232	-2.18	895,844	1,380,859	-485,015	4,181,759	3,972,976	208,783
2018年									
7月 July	126,529,100	-32,798	-0.26	81,397	106,839	-25,442	330,724	338,080	-7,356
8月 Aug.	126,496,302	-79,673	-0.63	82,144	106,739	-24,595	403,704	458,782	-55,078
9月 Sept.	126,416,629	26,551	0.21	78,126	104,894	-26,768	338,667	285,348	53,319
10月 Oct.	126,443,180	9,741	0.08	81,290	113,690	-32,400	309,736	267,595	42,141
11月 Nov.	126,452,921	-18,356	-0.15	76,193	115,227	-39,034	260,505	239,827	20,678
12月 Dec.	126,434,565	-117,397	-0.93	80,047	126,041	-45,994	327,505	398,908	-71,403
2019年									
1月 Jan.	126,317,168	-7,478	-0.06	71,138	138,502	-67,364	409,403	349,517	59,886
2月 Feb.	126,309,690	-61,266	-0.49	65,698	117,871	-52,173	290,764	299,857	-9,093
3月 Mar.	126,248,424	5,228	0.04	70,451	118,778	-48,327	383,957	330,402	53,555
4月 Apr.	126,253,652	-73,009	-0.58	70,257	112,495	-42,238	337,242	368,013	-30,771
5月 May	126,180,643	70,913	0.56	78,097	111,667	-33,570	365,196	260,713	104,483
6月 June	126,251,556	13,375	0.11	72,097	101,928	-29,831	316,473	273,267	43,206
7月 July	126,264,931	-46,075	-0.36	77,763	106,219	-28,456	348,490	366,109	-17,619
8月 Aug.	126,218,856	-87,412	-0.69	77,293	111,071	-33,778	446,608	500,242	-53,634
9月 Sept.	126,131,444	35,504	0.28	75,520	107,370	-31,850	385,880	318,526	67,354
10月 Oct.	126,166,948	-5,767	-0.05	76,128	113,895	-37,767	320,313	288,313	32,000
11月 Nov.	126,161,181	-17,188	-0.14	71,520	119,038	-47,518	283,774	253,444	30,330
12月 Dec.	126,143,993	-155,784	-1.23	77,554	129,482	-51,928	350,482	454,338	-103,856

日 本 人 人 口 （確 定 値） Japanese population (Final estimates)

年 月 Year and month	月初人口 (1)(2) Population as of 1st of each month	人口増減 Population change								
		純増減 Net change		自然動態 Natural change			社会動態 Migration change			国籍の異動(7)による純増減
		増減数 Number (15)+(18)+(19)	増減率(3) Rate (‰)	出生児数 Live Births	死亡者数 Deaths	自然増減 Natural change (13)-(14)	入国者数 Entries	出国者数 Exits	社会増減 Net migration (16)-(17)	Net increase or decrease by change of nationality
	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
2010年	126,381,728 (9)									
2011年	126,209,681	-172,047	-1.36	1,061,961	1,249,754	-187,793	798,792	826,503	-27,711	10,946
2012年	126,022,681	-187,000	-1.48	1,034,111	1,241,606	-207,495	887,670	910,828	-23,158	11,142
2013年	125,802,643	-220,038	-1.75	1,031,738	1,270,004	-238,266	903,217	926,289	-23,072	8,789
2014年	125,561,810	-240,833	-1.91	1,007,995	1,267,329	-259,334	928,759	952,213	-23,454	9,444
2015年	125,319,299 (9)	-242,511	-1.93	1,010,467	1,293,594	-283,127	948,338	949,299	-961	9,066
2016年	125,020,252	-299,047	-2.39	987,747	1,293,340	-305,593	961,652	963,739	-2,087	8,633
2017年	124,648,471	-371,781	-2.97	948,566	1,335,658	-387,092	976,144	971,918	4,226	11,085
2018年	124,218,285	-430,186	-3.45	927,255	1,361,417	-434,162	976,853	980,290	-3,437	7,413
2019年	123,731,176	-487,109	-3.92	878,092	1,373,600	-495,508	990,656	989,665	991	7,408
2018年										
7月 July	124,349,004	3,772	0.03	80,022	106,254	-26,232	124,251	95,006	29,245	759
8月 Aug.	124,352,776	-93,752	-0.75	80,705	106,182	-25,477	86,490	155,536	-69,046	771
9月 Sept.	124,259,024	-40,739	-0.33	76,697	104,394	-27,697	59,439	72,854	-13,415	313
10月 Oct.	124,218,285	-36,418	-0.29	79,792	113,094	-33,302	64,543	68,819	-4,276	1,160
11月 Nov.	124,181,867	-37,429	-0.30	74,773	114,643	-39,870	56,063	54,589	1,474	967
12月 Dec.	124,144,438	43,162	0.40	78,470	125,348	-46,878	144,816	43,938	94,878	1,162
2019年										
1月 Jan.	124,193,600	-135,974	-1.09	69,709	137,784	-68,075	57,438	125,834	-68,396	497
2月 Feb.	124,057,626	-65,085	-0.52	64,340	117,274	-52,934	55,843	68,512	-12,669	518
3月 Mar.	123,992,541	-32,120	-0.26	68,967	118,179	-49,212	84,031	67,499	16,532	560
4月 Apr.	123,960,421	-60,353	-0.49	68,802	111,878	-43,076	78,262	95,944	-17,682	405
5月 May	123,900,068	-26,650	-0.22	76,583	111,104	-34,521	71,146	63,745	7,401	470
6月 June	123,873,418	7,593	0.06	70,699	101,370	-30,671	105,111	67,416	37,695	569
7月 July	123,881,011	-860	-0.01	76,237	105,643	-29,406	124,786	96,854	27,932	614
8月 Aug.	123,880,151	-105,475	-0.85	75,730	110,470	-34,740	86,453	157,651	-71,198	463
9月 Sept.	123,774,676	-43,500	-0.35	73,990	106,813	-32,823	62,164	72,864	-10,700	23
10月 Oct.	123,731,176	-42,463	-0.34	74,472	113,245	-38,773	63,032	67,611	-4,579	889
11月 Nov.	123,688,713	-43,181	-0.35	69,863	118,417	-48,554	57,622	53,668	3,954	1,419
12月 Dec.	123,645,532	42,552	0.34	75,815	128,783	-52,968	149,256	55,124	94,132	1,388

https://www.e-stat.go.jp/

データ操作

```
import pandas as pd
df2 = pd.read_excel('05k2-2.xlsx', usecols=[14, 17, 19, 20, 21, 22, 23],skiprows=13, skipfooter=43,
names='西暦 総人口 増減数 増減率 出生児数 死亡者数 自然増減'.split()).set_index('西暦')

df2
```

西暦	総人口	増減数	増減率	出生児数	死亡者数	自然増減
2011年	126209681	-172047	-1.36	1061961	1249754	-187793
2012年	126022681	-187000	-1.48	1034111	1241606	-207495
2013年	125802643	-220038	-1.75	1031738	1270004	-238266
2014年	125561810	-240833	-1.91	1007995	1267329	-259334
2015年	125319299	-242511	-1.93	1010467	1293594	-283127
2016年	125020252	-299047	-2.39	987747	1293340	-305593
2017年	124648471	-371781	-2.97	948566	1335658	-387092
2018年	124218285	-430186	-3.45	927255	1361417	-434162
2019年	123731176	-487109	-3.92	878092	1373600	-495508

データ操作

日本人人口（確定値） Japanese population (Final estimates)

年月 Year and month	月初人口 [1][2] Population as of 1st of each month	人口増減 [3] Population change								
		純増減 Net change		自然動態 [4] Natural change			社会動態 [5] Migration change			国籍の異動 [7] による純増減
		増減数 Number (15)+(18)+(19)	増減率 [6] Rate (‰)	出生児数 Live Births	死亡者数 Deaths	自然増減 Natural change (13)-(14)	入国者数 Entries	出国者数 Exits	社会増減 Net migration (16)-(17)	Net increase or decrease by change of nationality
	(10)	(11)	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)
2010年	126,381,728 [9]									
2011年	126,209,681	-172,047	-1.36	1,061,961	1,249,754	-187,793	798,792	826,503	-27,711	10,946
2012年	126,022,681	-187,000	-1.48	1,034,111	1,241,606	-207,495	887,670	910,828	-23,158	11,142
2013年	125,802,643	-220,038	-1.75	1,031,738	1,270,004	-238,266	903,217	926,289	-23,072	8,789
2014年	125,561,810	-240,833	-1.91	1,007,995	1,267,329	-259,334	928,759	952,213	-23,454	9,444
2015年	125,319,299 [9]	-242,511	-1.93	1,010,467	1,293,594	-283,127	948,338	949,299	-961	9,066
2016年	125,020,252	-299,047	-2.39	987,747	1,293,340	-305,593	961,652	963,739	-2,087	8,633
2017年	124,648,471	-371,781	-2.97	948,566	1,335,658	-387,092	976,144	971,918	4,226	11,085
2018年	124,218,285	-430,186	-3.45	927,255	1,361,417	-434,162	976,853	980,290	-3,437	7,413
2019年	123,731,176	-487,109	-3.92	878,092	1,373,600	-495,508	990,656	989,665	991	7,408
2018年										
7月 July	124,349,004	3,772	0.03	80,022	106,254	-26,232	124,251	95,006	29,245	759
8月 Aug.	124,352,776	-93,752	-0.75	80,705	106,182	-25,477	86,490	155,536	-69,046	771
9月 Sept.	124,259,024	-40,739	-0.33	76,697	104,334	-27,637	59,439	72,854	-13,415	313
10月 Oct.	124,218,285	-36,418	-0.29	79,792	113,094	-33,302	64,543	68,819	-4,276	1,160
11月 Nov.	124,181,867	-37,429	-0.30	74,773	114,643	-39,870	56,063	54,589	1,474	967
12月 Dec.	124,144,438	49,162	0.40	78,470	125,348	-46,878	144,816	49,938	94,878	1,162
2019年										
1月 Jan.	124,193,600	-135,974	-1.09	69,709	137,784	-68,075	57,438	125,834	-68,396	497
2月 Feb.	124,057,626	-65,085	-0.52	64,340	117,274	-52,934	55,843	68,512	-12,669	518
3月 Mar.	123,992,541	-32,120	-0.26	68,967	118,179	-49,212	84,031	67,499	16,532	560
4月 Apr.	123,960,421	-60,353	-0.49	68,802	111,878	-43,076	78,262	95,944	-17,682	405
5月 May	123,900,068	-26,650	-0.22	76,583	111,104	-34,521	71,146	63,745	7,401	470
6月 June	123,873,418	7,593	0.06	70,699	101,370	-30,671	105,111	67,416	37,695	569
7月 July	123,881,011	-860	-0.01	76,237	105,643	-29,406	124,786	96,854	27,932	614
8月 Aug.	123,880,151	-105,475	-0.85	75,730	110,470	-34,740	86,453	157,651	-71,198	463
9月 Sept.	123,774,676	-43,500	-0.35	73,990	106,813	-32,823	62,164	72,864	-10,700	23
10月 Oct.	123,731,176	-42,463	-0.34	74,472	113,245	-38,773	63,032	67,611	-4,579	889
11月 Nov.	123,688,713	-43,181	-0.35	69,863	118,417	-48,554	57,622	53,668	3,954	1,419
12月 Dec.	123,645,532	42,552	0.34	75,815	128,783	-52,968	149,256	55,124	94,132	1,388

```
import pandas as pd
pd.read_excel
```

```
skiprows=13
```

```
skipfooter=43
```

データ操作

df2[5:]

西暦	総人口	増減数	増減率	出生児数	死亡者数	自然増減
2016年	125020252	-299047	-2.39	987747	1293340	-305593
2017年	124648471	-371781	-2.97	948566	1335658	-387092
2018年	124218285	-430186	-3.45	927255	1361417	-434162
2019年	123731176	-487109	-3.92	878092	1373600	-495508

df2['総人口']

```
西暦
2011年    126209681
2012年    126022681
2013年    125802643
2014年    125561810
2015年    125319299
2016年    125020252
2017年    124648471
2018年    124218285
2019年    123731176
Name: 総人口, dtype: int64
```

df2[['総人口', '増減数', '出生児数', '自然増減']]

西暦	総人口	増減数	出生児数	自然増減
2011年	126209681	-172047	1061961	-187793
2012年	126022681	-187000	1034111	-207495
2013年	125802643	-220038	1031738	-238266
2014年	125561810	-240833	1007995	-259334
2015年	125319299	-242511	1010467	-283127
2016年	125020252	-299047	987747	-305593
2017年	124648471	-371781	948566	-387092
2018年	124218285	-430186	927255	-434162
2019年	123731176	-487109	878092	-495508

データ操作

```
df2['Total'] = df2['出生児数'] - df2['死亡者数']
df2
```

西暦	総人口	増減数	増減率	出生児数	死亡者数	自然増減	Total
2011年	126209681	-172047	-1.36	1061961	1249754	-187793	-187793
2012年	126022681	-187000	-1.48	1034111	1241606	-207495	-207495
2013年	125802643	-220038	-1.75	1031738	1270004	-238266	-238266
2014年	125561810	-240833	-1.91	1007995	1267329	-259334	-259334
2015年	125319299	-242511	-1.93	1010467	1293594	-283127	-283127
2016年	125020252	-299047	-2.39	987747	1293340	-305593	-305593
2017年	124648471	-371781	-2.97	948566	1335658	-387092	-387092
2018年	124218285	-430186	-3.45	927255	1361417	-434162	-434162
2019年	123731176	-487109	-3.92	878092	1373600	-495508	-495508

```
df2[df2['増減率'] < -2.0]
```

西暦	総人口	増減数	増減率	出生児数	死亡者数	自然増減
2016年	125020252	-299047	-2.39	987747	1293340	-305593
2017年	124648471	-371781	-2.97	948566	1335658	-387092
2018年	124218285	-430186	-3.45	927255	1361417	-434162
2019年	123731176	-487109	-3.92	878092	1373600	-495508

データ操作 政府統計

2 平成21年度米麦加工食品生産動向

平成21年の生産量

（参考4） 乾めん類の都道府県別生産量（平成21年）

（単位：小麦粉使用トン）

都道府県	うどん	ひらめん	ひやむぎ	そうめん	手延うどん	手延ひやむぎ	手延そうめん	干中華	日本そば	合計	対前年（%）	そば粉使用量
北海道	1,830	26	1,015	1,897	61	10	2	5,083	1,332	11,255	6.0	544
青森	31	37	37	45	0	0	0	0	51	200	4.2	23
岩手	835	103	201	743	0	0	0	28	1,776	3,686	19.3	849
宮城	1,846	80	1,166	3,538	34	1	157	8	1,033	7,862	−1.5	349
秋田	1,781	26	31	50	243	10	44	0	73	2,258	−2.7	14
山形	1,621	137	1,083	1,205	24	9	0	164	3,464	7,708	−0.7	1,215
福島	1,661	15	616	1,150	94	120	31	10	1,337	5,032	−3.1	490
茨城	3,797	327	1,347	1,439	14	0	0	106	2,728	9,759	−7.5	1,173
栃木	311	12	61	101	0	0	1	2	34	521	−44.9	22
群馬	3,136	206	914	792	2	1	39	259	216	5,563	−10.3	101
埼玉	317	188	199	51	0	0	0	25	112	891	−13.9	44
千葉	767	4	193	297	1	3	2	141	81	1,487	−7.0	34
東京	*	*	*	*	*	*	*	*	*	*	*	*

https://www.e-stat.go.jp/

データ操作 政府統計

```
df = pd.read_excel('i010-21-005.xls', usecols=[0, 1, 2, 3, 5, 7, 10,11], skiprows=4, skipfooter=3,
names='都道府県 うどん ひらめん ひやむぎ 手延うどん 手延そうめん 合計 対前年（%）'
.split()).set_index('都道府県')

df
```

都道府県	うどん	ひらめん	ひやむぎ	手延うどん	手延そうめん	合計	対前年（%）
北海道	1830	26	1015	61	2	11255	6
青　森	31	37	37	0	0	200	4.2
岩　手	835	103	201	0	0	3686	19.3
宮　城	1846	80	1166	34	157	7862	-1.5
秋　田	1781	26	31	243	44	2258	-2.7
山　形	1621	137	1083	24	0	7708	-0.7
福　島	1661	15	616	94	31	5032	-3.1
茨　城	3797	327	1347	14	0	9759	-7.5
栃　木	311	12	61	0	1	521	-44.9
群　馬	3136	206	914	2	39	5563	-10.3
埼　玉	317	188	199	0	0	891	-13.9
千　葉	767	4	193	1	2	1487	-7
東　京	*	*	*	*	*	*	*

データ操作 政府統計

print(df=='*')

栃　木	False	False	False	False	False	False	False
群　馬	False	False	False	False	False	False	False
埼　玉	False	False	False	False	False	False	False
千　葉	False	False	False	False	False	False	False
東　京	True	True	True	True	True	True	True
神奈川	False	False	False	False	False	False	False
新　潟	False	False	False	False	False	False	False
富　山	False	False	False	False	False	False	False
石　川	True	True	True	True	True	True	True
福　井	False	False	False	False	False	False	False
山　梨	False	False	False	False	False	False	False

欠損値の取り扱い
* 印を探して 0 と入れ替える.

df[df=='*']=0

栃　木	311	12	61	0	1	521	-44.9
群　馬	3136	206	914	2	39	5563	-10.3
埼　玉	317	188	199	0	0	891	-13.9
千　葉	767	4	193	1	2	1487	-7
東　京	0	0	0	0	0	0	0
神奈川	595	6	292	0	1	1401	-9.2
新　潟	1319	0	597	1	0	4852	2.1
富　山	83	0	5	65	322	754	-3
石　川	0	0	0	0	0	0	0
福　井	1	0	0	0	0	3	-83.3
山　梨	1824	46	641	0	0	3914	-27.1

データ操作 政府統計

「うどん」についてソート

```
df.sort_values('うどん', ascending=False)[:10]
```

都道府県	うどん	ひらめん	ひやむぎ	手延うどん	手延そうめん	合計	対前年（%）
香　川	12563	64	2229	89	2817	24090	-7.9
茨　城	3797	327	1347	14	0	9759	-7.5
群　馬	3136	206	914	2	39	5563	-10.3
宮　城	1846	80	1166	34	157	7862	-1.5
北海道	1830	26	1015	61	2	11255	6
山　梨	1824	46	641	0	0	3914	-27.1
秋　田	1781	26	31	243	44	2258	-2.7
福　島	1661	15	616	94	31	5032	-3.1
山　形	1621	137	1083	24	0	7708	-0.7
兵　庫	1615	56	1504	52	20117	33951	-2.8

データ操作 政府統計

「合計」についてソート

```
df.sort_values('合計', ascending=False)[:10]
```

都道府県	うどん	ひらめん	ひやむぎ	手延うどん	手延そうめん	合計	対前年（%）
兵　庫	1615	56	1504	52	20117	33951	-2.8
香　川	12563	64	2229	89	2817	24090	-7.9
長　崎	70	1	0	1060	13566	14910	5.7
長　野	372	15	391	0	0	14478	-8.4
北海道	1830	26	1015	61	2	11255	6
茨　城	3797	327	1347	14	0	9759	-7.5
宮　城	1846	80	1166	34	157	7862	-1.5
山　形	1621	137	1083	24	0	7708	-0.7
愛　知	1284	661	1062	331	186	6388	-0.7
群　馬	3136	206	914	2	39	5563	-10.3

データ操作 政府統計

「手延そうめん」についてソー

```
df.sort_values('手延そうめん', ascending=False)[:10]
```

都道府県	うどん	ひらめん	ひやむぎ	手延うどん	手延そうめん	合計	対前年（%）
兵　庫	1615	56	1504	52	20117	33951	-2.8
長　崎	70	1	0	1060	13566	14910	5.7
奈　良	10	1	2	0	3669	3818	-9.9
香　川	12563	64	2229	89	2817	24090	-7.9
徳　島	46	0	1	3	2664	3549	-1.1
岡　山	19	21	226	1559	1636	5426	-0.1
富　山	83	0	5	65	322	754	-3
愛　知	1284	661	1062	331	186	6388	-0.7
三　重	424	69	232	194	179	3360	2.1
宮　城	1846	80	1166	34	157	7862	-1.5

データ可視化 政府統計

日本地図を表示

```
pip install --user japanmap

（再起動）

%matplotlib inline
import matplotlib.pyplot as plt
from japanmap import picture
plt.rcParams['figure.figsize'] = 6, 6
plt.imshow(picture());
plt.savefig('japan.png')
```

DataFrame Column/Index

必要に応じて 確認 / 書き換え

df.index

```
Index(['北海道', '青　森', '岩　手', '宮　城', '秋　田', '山　形', '福　島', '茨　城', '栃　木', '群　馬',
       '埼　玉', '千　葉', '東　京', '神奈川', '新　潟', '富　山', '石　川', '福　井', '山　梨', '長　野',
       '岐　阜', '静　岡', '愛　知', '三　重', '滋　賀', '京　都', '大　阪', '兵　庫', '奈　良', '和歌山',
       '鳥　取', '島　根', '岡　山', '広　島', '山　口', '徳　島', '香　川', '愛　媛', '高　知', '福　岡',
       '佐　賀', '長　崎', '熊　本', '大　分', '宮　崎', '鹿児島', '沖　縄'],
      dtype='object', name='都道府県')
```

```
df.index=['北海道', '青森', '岩手', '宮城', '秋田', '山形', '福島', '茨城', '栃木',
      '群馬','埼玉', '千葉', '東京', '神奈川', '新潟', '富山', '石川', '福井', '山梨',
      '長野', '岐阜', '静岡', '愛知', '三重', '滋賀', '京都', '大阪', '兵庫', '奈良',
      '和歌山','鳥取', '島根', '岡山', '広島', '山口', '徳島', '香川', '愛媛', '高知',
      '福岡', '佐賀', '長崎', '熊本', '大分', '宮崎', '鹿児島', '沖縄']
```

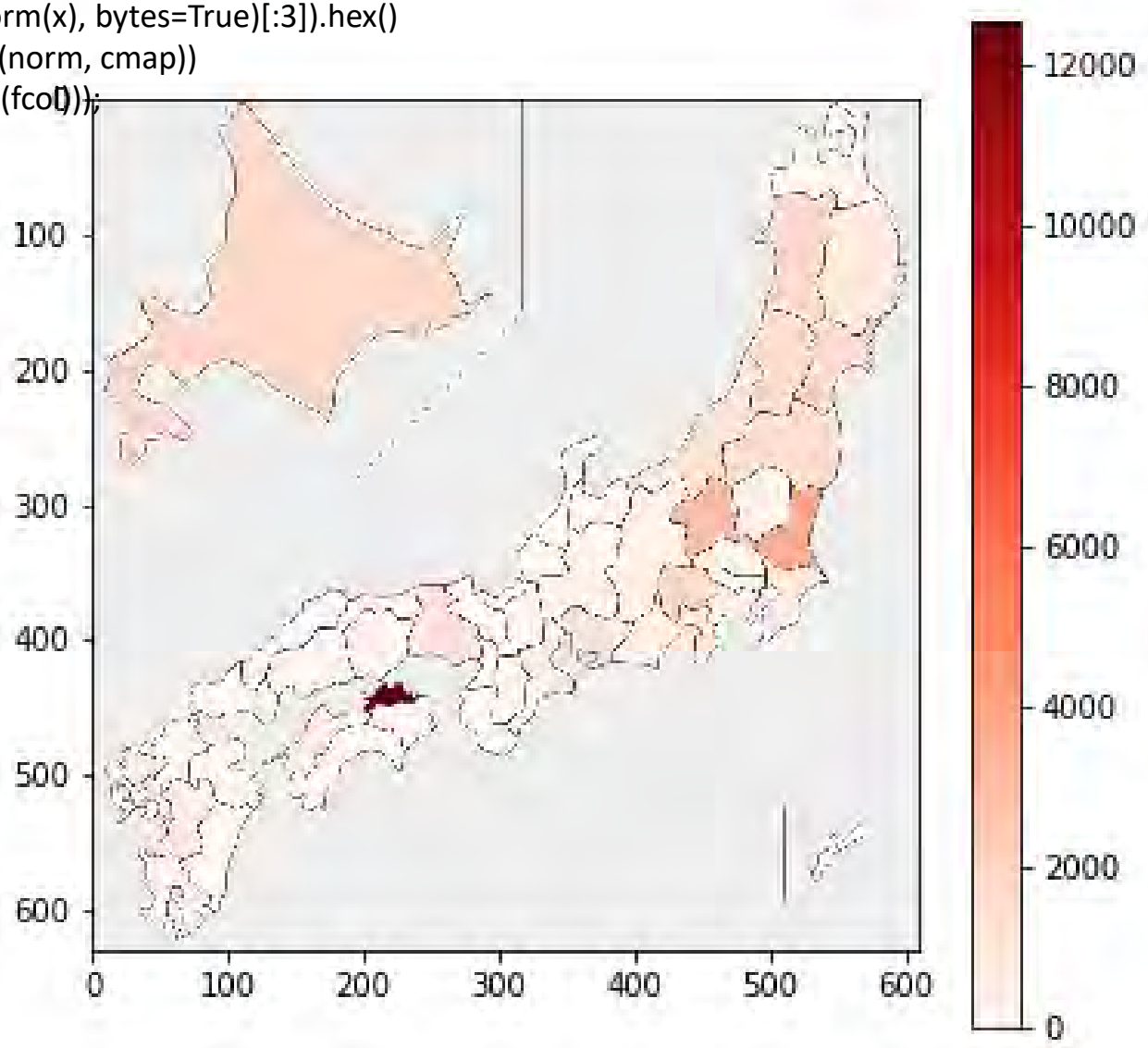
df.columns

```
Index(['うどん', 'ひらめん', 'ひやむぎ', '手延うどん', '手延そうめん', '合計', '対前年
（％）'],
 dtype='object')
```

```
df.columns=['Udon', 'Kishimen', 'Hiyamugi', '....']
```

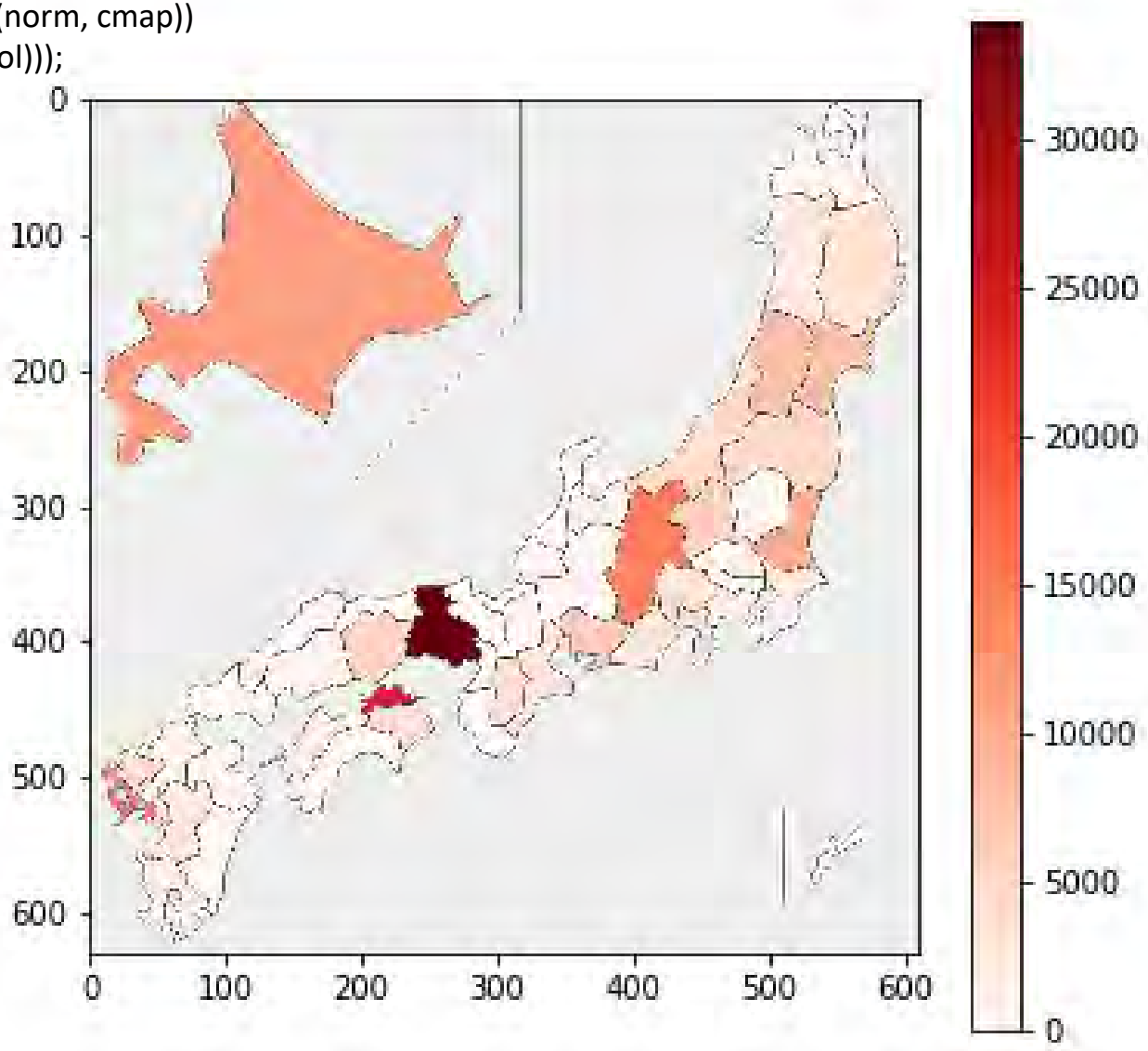
```
cmap = plt.get_cmap('Reds')
norm = plt.Normalize(vmin=df.うどん.min(), vmax=df.うどん.max())
fcol = lambda x: '#' + bytes(cmap(norm(x), bytes=True)[:3]).hex()
plt.colorbar(plt.cm.ScalarMappable(norm, cmap))
plt.imshow(picture(df.うどん.apply(fcol)));
plt.savefig('japan_Udon.png')
```

「うどん」の生産量日本一



```
cmap = plt.get_cmap('Reds')
norm = plt.Normalize(vmin=df.合計.min(), vmax=df.合計.max())
fcol = lambda x: '#' + bytes(cmap(norm(x), bytes=True)[:3]).hex()
plt.colorbar(plt.cm.ScalarMappable(norm, cmap))
plt.imshow(picture(df.合計.apply(fcol)));
plt.savefig('japan_Total.png')
```

「乾めん」の生産量日本一



テストデータ作成

```
from sklearn.model_selection import train_test_split
X_train, x_test, y_train, y_test = train_test_split(df.うどん, df.合計,
test_size=0.3)
```

X_train

都道府県	
岡　山	19
徳　島	46
埼　玉	317
神奈川	595
奈　良	10
秋　田	1781
福　岡	167
岩　手	835
長　崎	70
福　井	1
山　口	0
千　葉	767
福　島	1661
山　形	1621

```
print(len(X_train))
print(len(x_test))

32
15
```

テストデータ
サイズは 30%

テストデータ作成

```
from sklearn.model_selection import train_test_split
X_train, x_test, y_train, y_test = train_test_split(df[['うどん', 'ひらめん', 'ひやむぎ',
"手延うどん"]], df.合計, test_size=0.3)
```

都道府県	うどん	ひらめん	ひやむぎ	手延うどん
神奈川	595	6	292	0
岡　山	19	21	226	1559
山　形	1621	137	1083	24
群　馬	3136	206	914	2
福　岡	167	35	58	0
沖　縄	0	0	0	0
滋　賀	0	0	0	0
兵　庫	1615	56	1504	52
宮　崎	0	0	0	0
愛　媛	405	0	174	0
長　野	372	15	391	0

....

X_train の一部

X_train

都道府県	うどん	ひらめん	ひやむぎ	手延うどん
神奈川	595	6	292	0
岡　山	19	21	226	1559
山　形	1621	137	1083	24
群　馬	3136	206	914	2
福　岡	167	35	58	0
沖　縄	0	0	0	0
滋　賀	0	0	0	0
兵　庫	1615	56	1504	52
宮　崎	0	0	0	0
愛　媛	405	0	174	0
長　野	372	15	391	0
山　梨	1824	46	641	0
秋　田	1781	26	31	243
栃　木	311	12	61	0
青　森	31	37	37	0
大　阪	0	0	0	0
三　重	424	69	232	194
山　口	0	0	0	0
長　崎	70	1	0	1060
岩　手	835	103	201	0
茨　城	3797	327	1347	14
和歌山	0	0	0	0
高　知	0	0	0	0
熊　本	741	24	271	6
大　分	3	23	20	0
東　京	0	0	0	0
広　島	118	0	41	1
岐　阜	178	20	108	0
福　井	1	0	0	0
香　川	12563	64	2229	89
千　葉	767	4	193	1
宮　城	1846	80	1166	34

x_test

都道府県	うどん	ひらめん	ひやむぎ	手延うどん
福　島	1661	15	616	94
新　潟	1319	0	597	1
愛　知	1284	661	1062	331
埼　玉	317	188	199	0
徳　島	46	0	1	3
石　川	0	0	0	0
鳥　取	0	0	0	0
島　根	42	1	0	0
鹿児島	377	3	12	0
佐　賀	978	18	624	25
京　都	0	0	0	0
北海道	1830	26	1015	61
富　山	83	0	5	65
静　岡	477	47	77	0
奈　良	10	1	2	0

y_train

```
都道府県
神奈川     1401
岡　山     5426
山　形     7708
群　馬     5563
福　岡     1010
沖　縄        0
滋　賀        0
兵　庫    33951
宮　崎        0
愛　媛     1492
長　野    14478
山　梨     3914
秋　田     2258
栃　木      521
青　森      200
大　阪        0
三　重     3360
山　口        0
長　崎    14910
岩　手     3686
茨　城     9759
和歌山        0
高　知        0
熊　本     2831
大　分      104
東　京        0
広　島      436
岐　阜      372
福　井        3
香　川    24090
千　葉     1487
宮　城     7862
Name: 合計, dtype: object
```

y_test

```
都道府県
福　島     5032
新　潟     4852
愛　知     6388
埼　玉      891
徳　島     3549
石　川        0
鳥　取        0
島　根      335
鹿児島     1531
佐　賀     4257
京　都        0
北海道    11255
富　山      754
静　岡     3275
奈　良     3818
Name: 合計, dtype: object
```

Python: オブジェクト指向

```
def main():
    a = 3 # assign a value '3' to a variable 'a'
    b = 15
    print(a + b) # addition
    print(a - b) # subtraction

if __name__ == "__main__":
    main()
```

import で呼ばれたときは、ファイル名が格納される.
コマンドラインから直接呼ばれると、main が格納される.

コマンドラインから呼ばれたときのみ実行する.

Class を定義する.

Python の
オブジェクト指向言語
としての側面

```
class Pain:
        def __init__(self):
                self.arg1 = "Body: “

        def ouch1(self):
                print(self.arg1 + "Knee.")

        def ouch2(self):
                print(self.arg1 + "Elbow.")
```

呼び出された時
自動実行

実行

```
inspect=Pain()
inspect.ouch2()
Body: Elbow.
```

__init__が実行

ouch2が実行

```
dir(inspect)

['__class__',
   ....
 '__dir__',
   ....
 '__init__',
   ....
 'arg1',
 'ouch1',
 'ouch2']
```

現在のローカルスコープ
にある名前のリスト

Calass
Object
Method

Group Work

- 現在及びこれからの業務内容から考えて...
  - どのようなデータと関わることになるか？
    - 画像、時系列、学習、統計処理、可視化...
  - プログラミング言語を用いる機会があるか.
    - 簡単なスクリプトを書くような使い方？
  - その時どのような言語を用いる必要があるか.

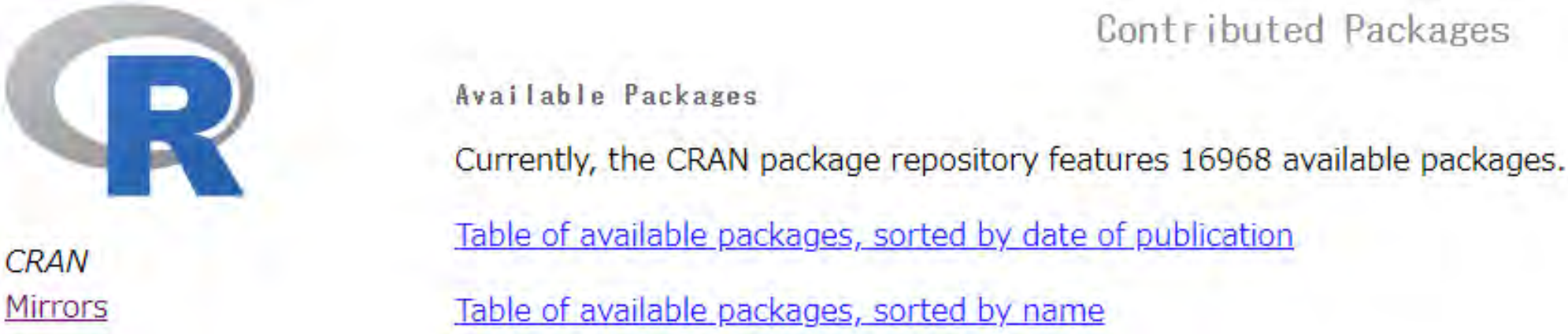
R言語入門

R言語の概要

- C言語等
  - プログラム作成が大変
  - 高性能に分析
- EXCEL, SPSS
  - GUIで操作
  - 手軽に作成することが可能
  - 高価
- R言語
  - オープンソース＆フリー
  - 1995年に開発された統計解析に特化したプログラミング言語
  - データの変換、分析＆グラフィック、レポーティング

R言語の概要

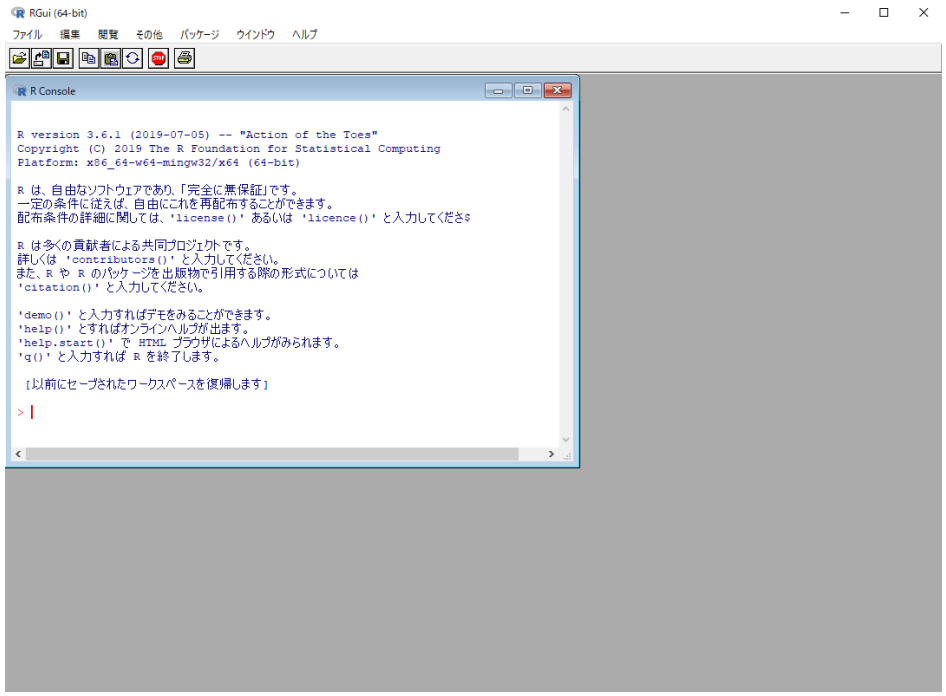
- ユーザーが作成した多数のパッケージ
- 2021年1月21日現在
- Currently, the CRAN package repository features 16957 available packages



https://cran.r-project.org/

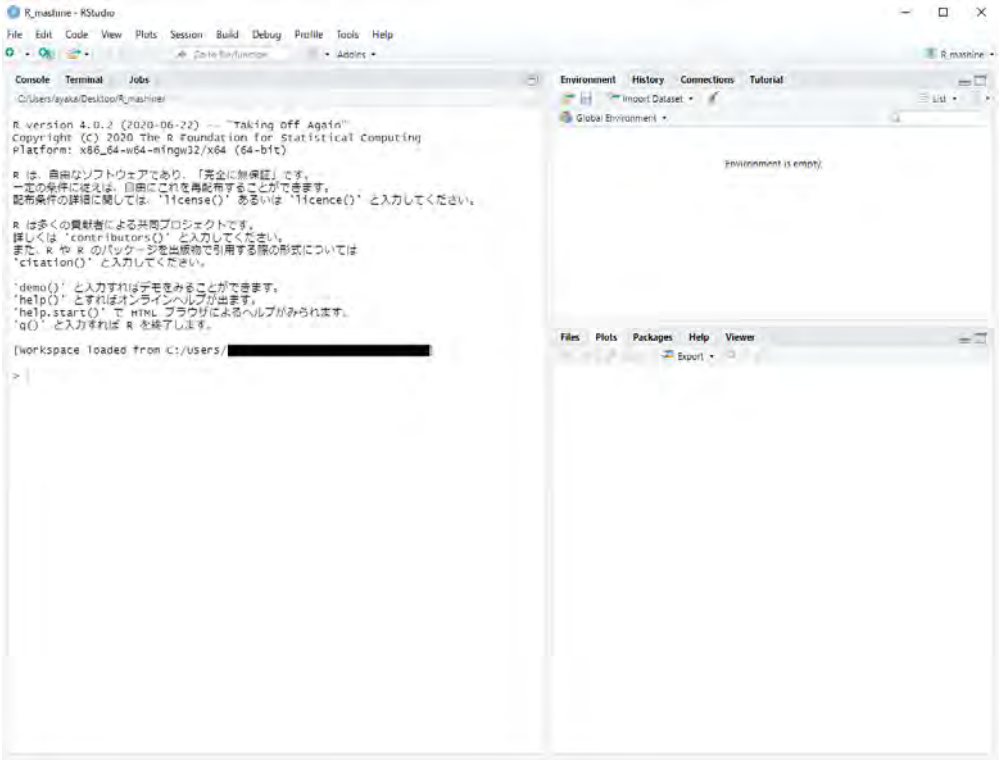
RとRStudio

R



計算を実行するプログラミング言語

RStudio



RStudioを通してRを利用する

RとRStudio

- RStudio
- Rでデータ分析やプログラミングを行うのに便利な機能が豊富に用意されている統合開発環境 (Integrated Development Environment; IDE)
- 多くの便利な機能やツールを追加してインタフェースを提供
- クロスプラットフォームなソフトウェア
  - Windows, Mac, 各種Linux系のOSに対応

RとRStudio

R

車のエンジン

RStudio

車のダッシュボード

(参考) https://education.rstudio.com/learn/beginner/

RとRStudio

- 車を運転するときにエンジンを直接使用するのではなく、ダッシュボードの要素を利用するように、Rを直接使用するのではなく、RStudioのインターフェイスを使用
- スピードメーターやバックミラー、ナビゲーションシステムにアクセスすることで運転が楽になるのと同じように、RStudioのインタフェースを使うことで R の使い方も楽になる

(参考) https://education.rstudio.com/learn/beginner/

RとRStudio

- R と RStudio（デスクトップ版）の両方をコンピュータにダウンロードしてインストールする必要がある
- 最初に R をインストールしてから RStudio をインストールする

(参考) https://education.rstudio.com/learn/beginner/

RとRStudio

- Rのダウンロード：以下のサイトにアクセス

https://cloud.r-project.org/

[windows]

①Download R for Windowsをクリック

②baseをクリック

③ダウンロードリンクをクリック

RとRStudio

- Rstudioのダウンロード：以下のサイトにアクセス

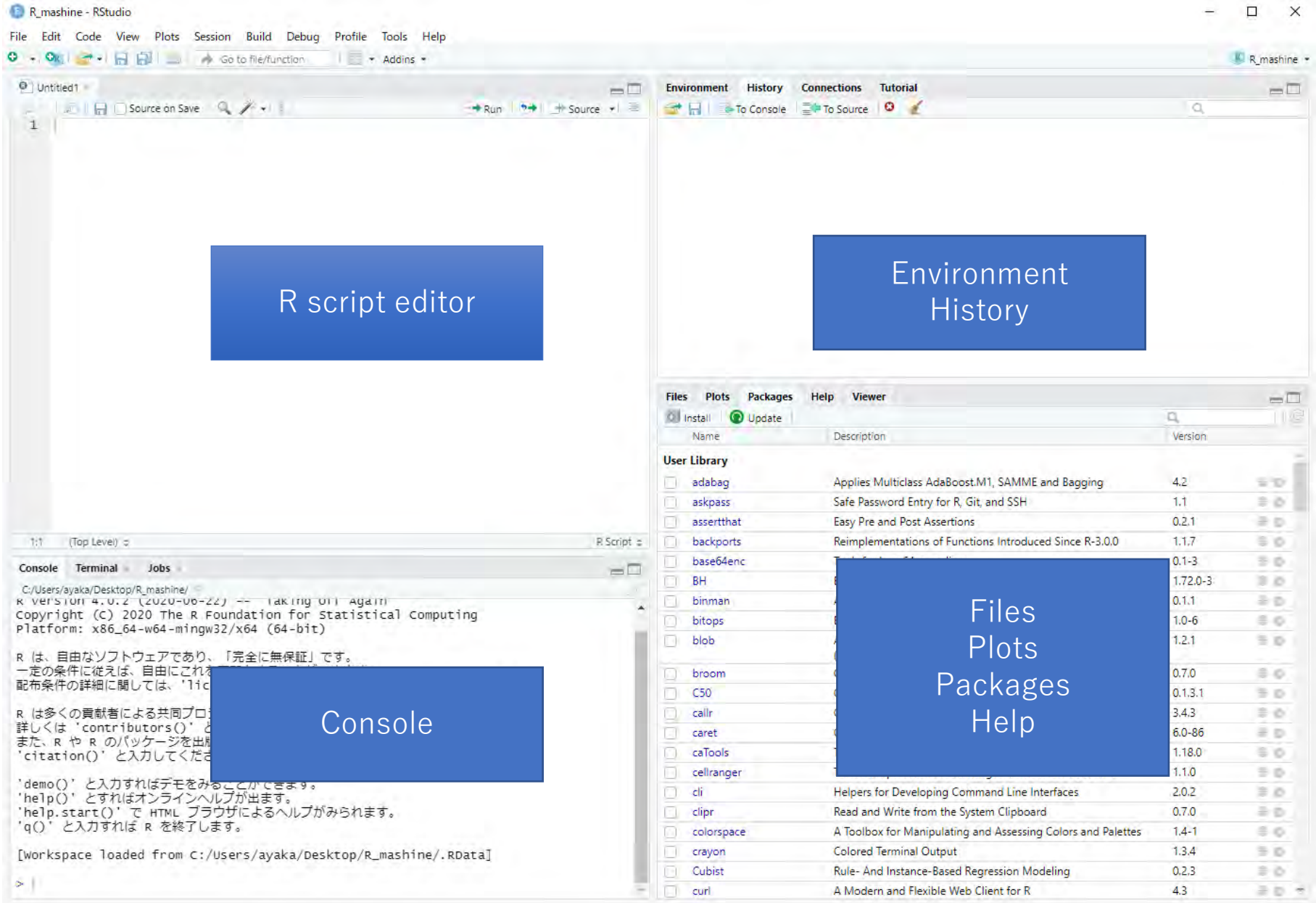
https://rstudio.com/products/rstudio/download/

[windows]

①RStudio DesktopのDOWNLOADをクリック

②OS Windows 10/8/7のダウンロードリンクをクリック

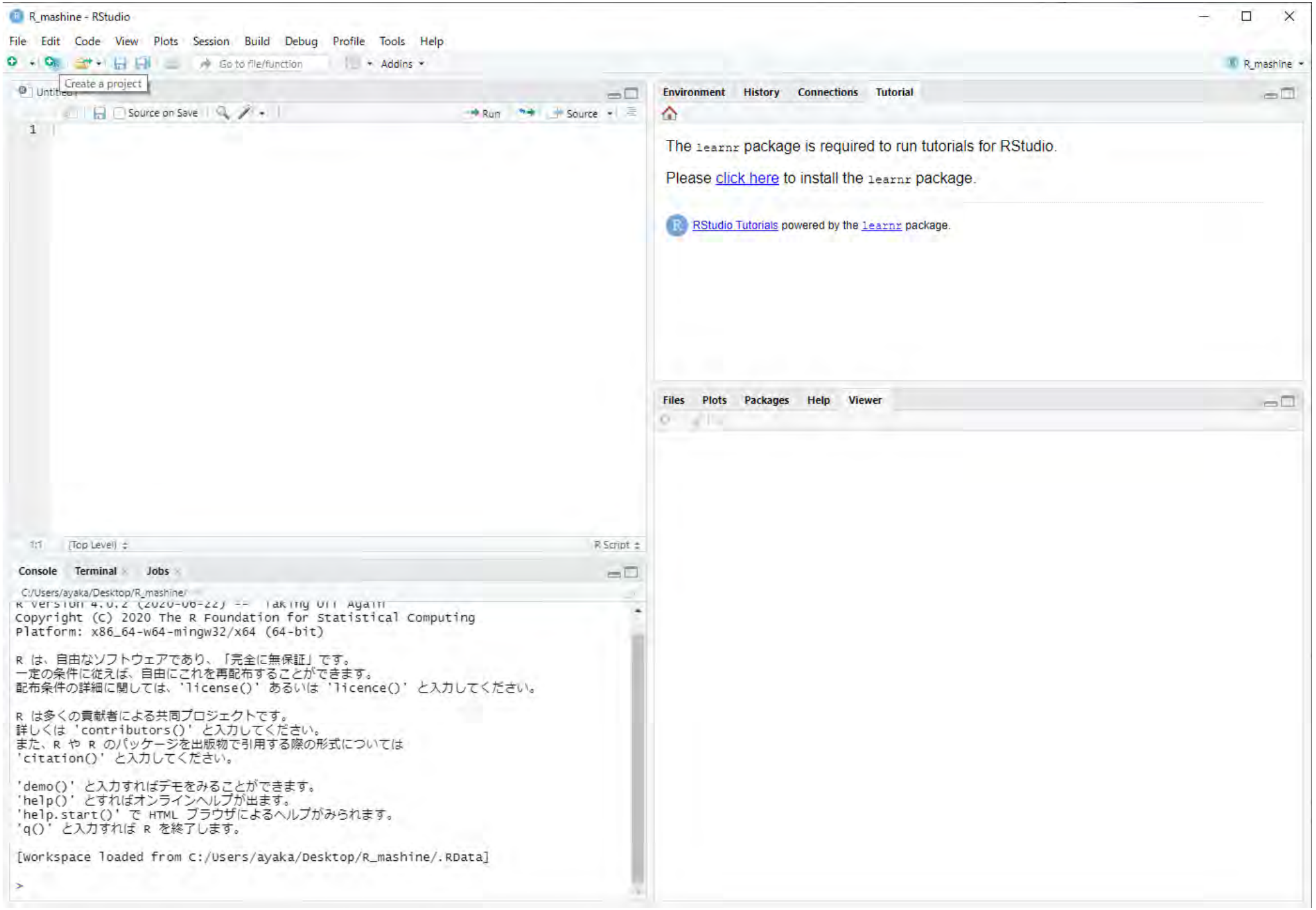
Rstudioの基礎


R_mashine - RStudio
File Edit Code View Plots Session Build Debug Profile Tools Help
Untitled1
Source on Save
Run
Source
R script editor
Environment History Connections Tutorial
To Console
To Source
Environment
History
Files Plots Packages Help Viewer
Install
Update
Name
Description
Version
User Library
adabag Applies Multiclass AdaBoost.M1, SAMME and Bagging 4.2
askpass Safe Password Entry for R, Git, and SSH 1.1
assertthat Easy Pre and Post Assertions 0.2.1
backports Reimplementations of Functions Introduced Since R-3.0.0 1.1.7
base64enc 0.1-3
BH 1.72.0-3
binman 0.1.1
bitops 1.0-6
blob 1.2.1
broom 0.7.0
C50 0.1.3.1
callr 3.4.3
caret 6.0-86
caTools 1.18.0
cellranger 1.1.0
cli Helpers for Developing Command Line Interfaces 2.0.2
clipr Read and Write from the System Clipboard 0.7.0
colorspace A Toolbox for Manipulating and Assessing Colors and Palettes 1.4-1
crayon Colored Terminal Output 1.3.4
Cubist Rule- And Instance-Based Regression Modeling 0.2.3
curl A Modern and Flexible Web Client for R 4.3
Files
Plots
Packages
Help
Console Terminal Jobs
C:/Users/ayaka/Desktop/R_mashine/
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)
R は、自由なソフトウェアであり、「完全に無保証」です。
'citation()' と入力してくだ
'demo()' と入力すればデモをみることができます。
'help()' とすればオンラインヘルプが出ます。
'help.start()' で HTML ブラウザによるヘルプがみられます。
'q()' と入力すれば R を終了します。
[Workspace loaded from C:/Users/ayaka/Desktop/R_mashine/.RData]
Console

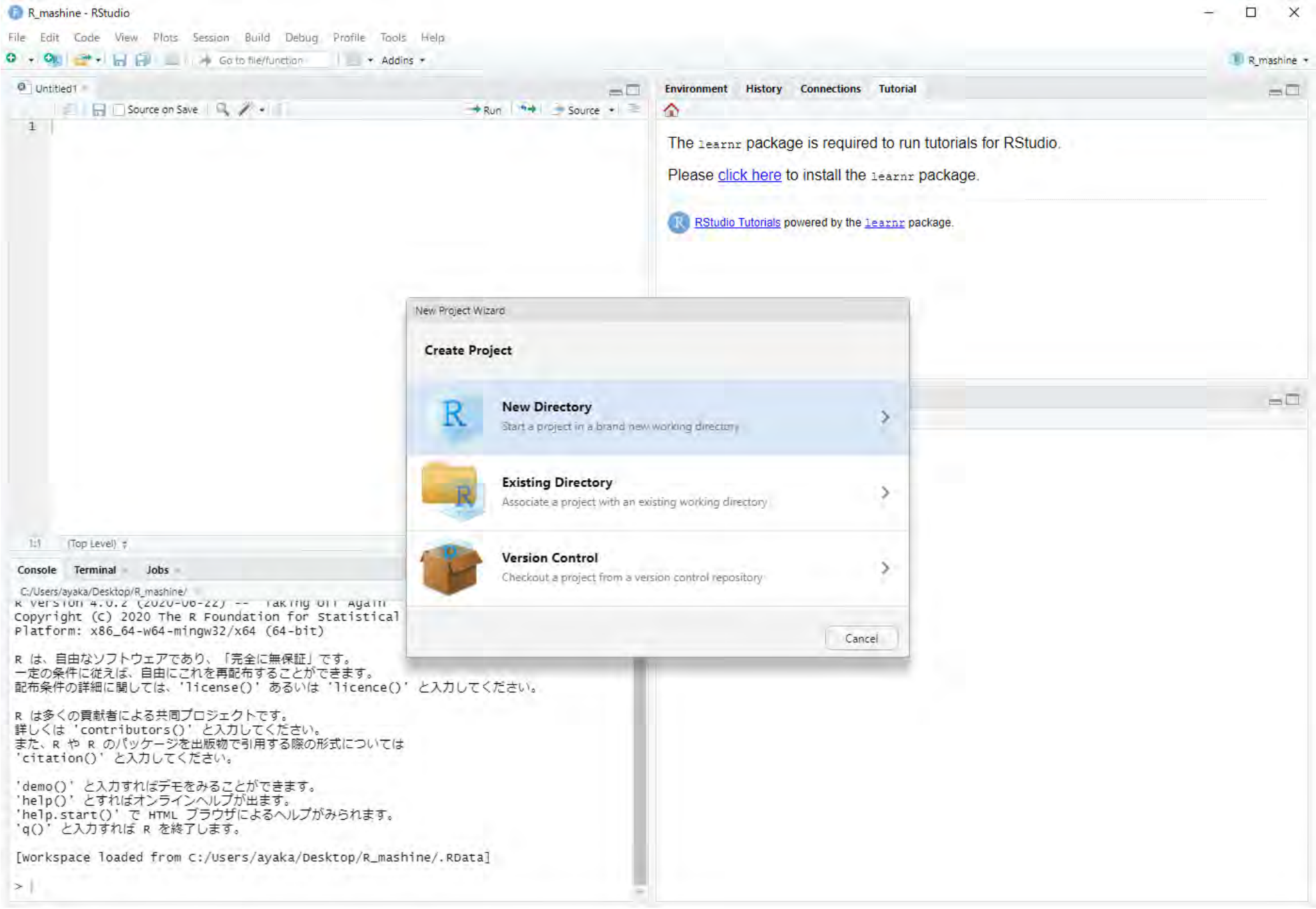
Rstudioの基礎

- プロジェクト機能
  - データや分析コードを案件ごとに管理
- ファイル、バージョンの管理が容易に
- プロジェクトの新規作成

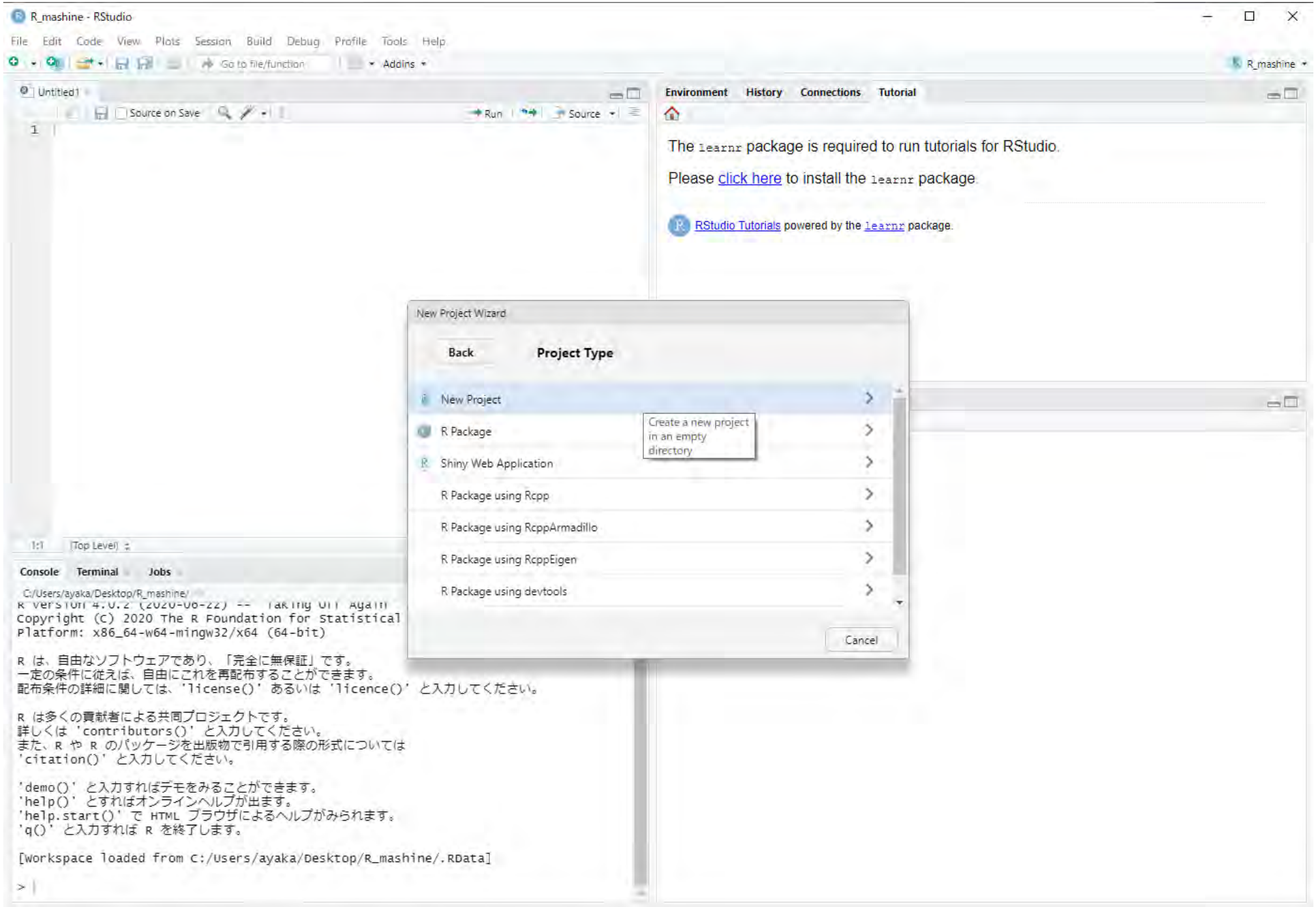
1. Create a project



2. New Directory

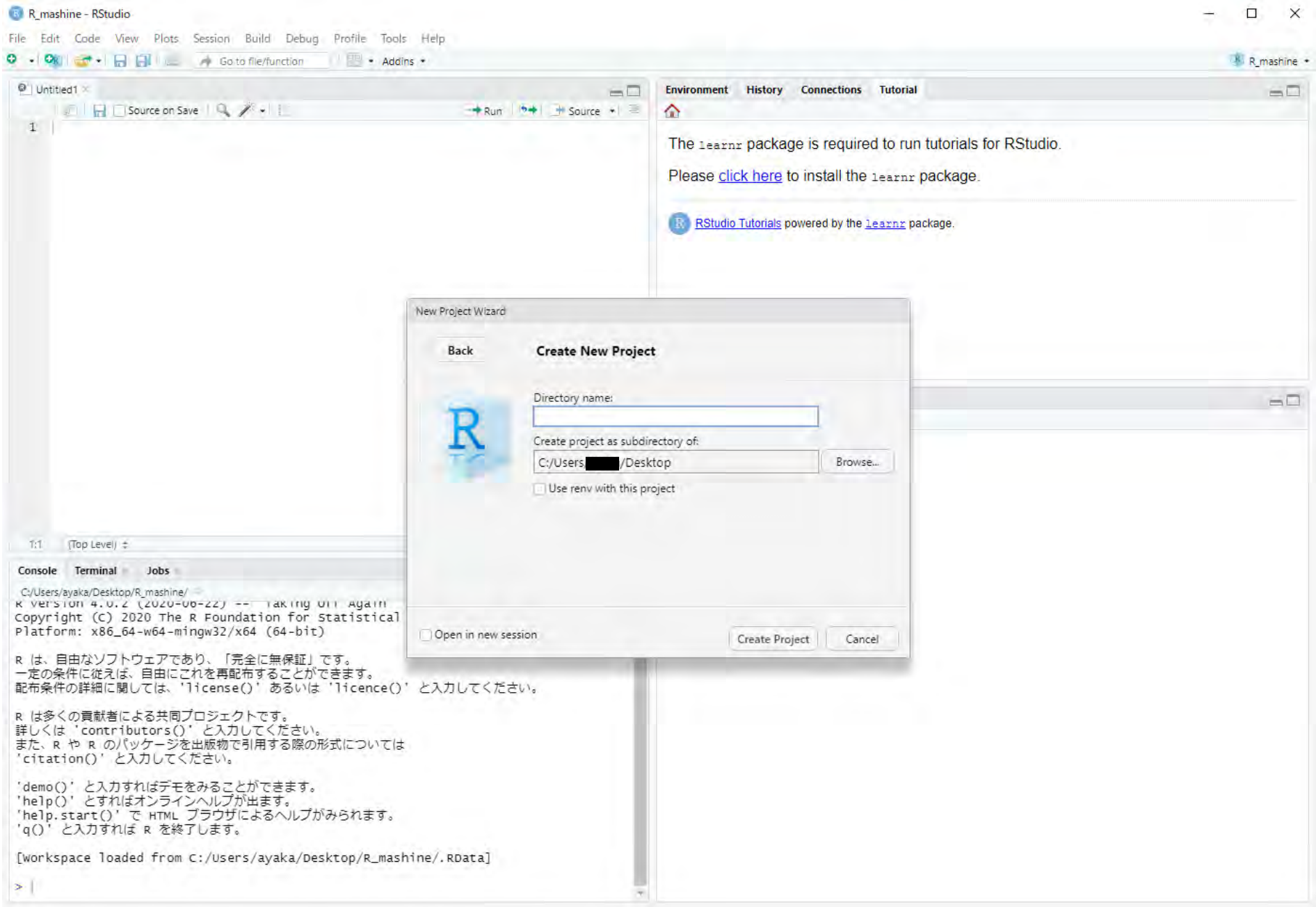


3. New Project

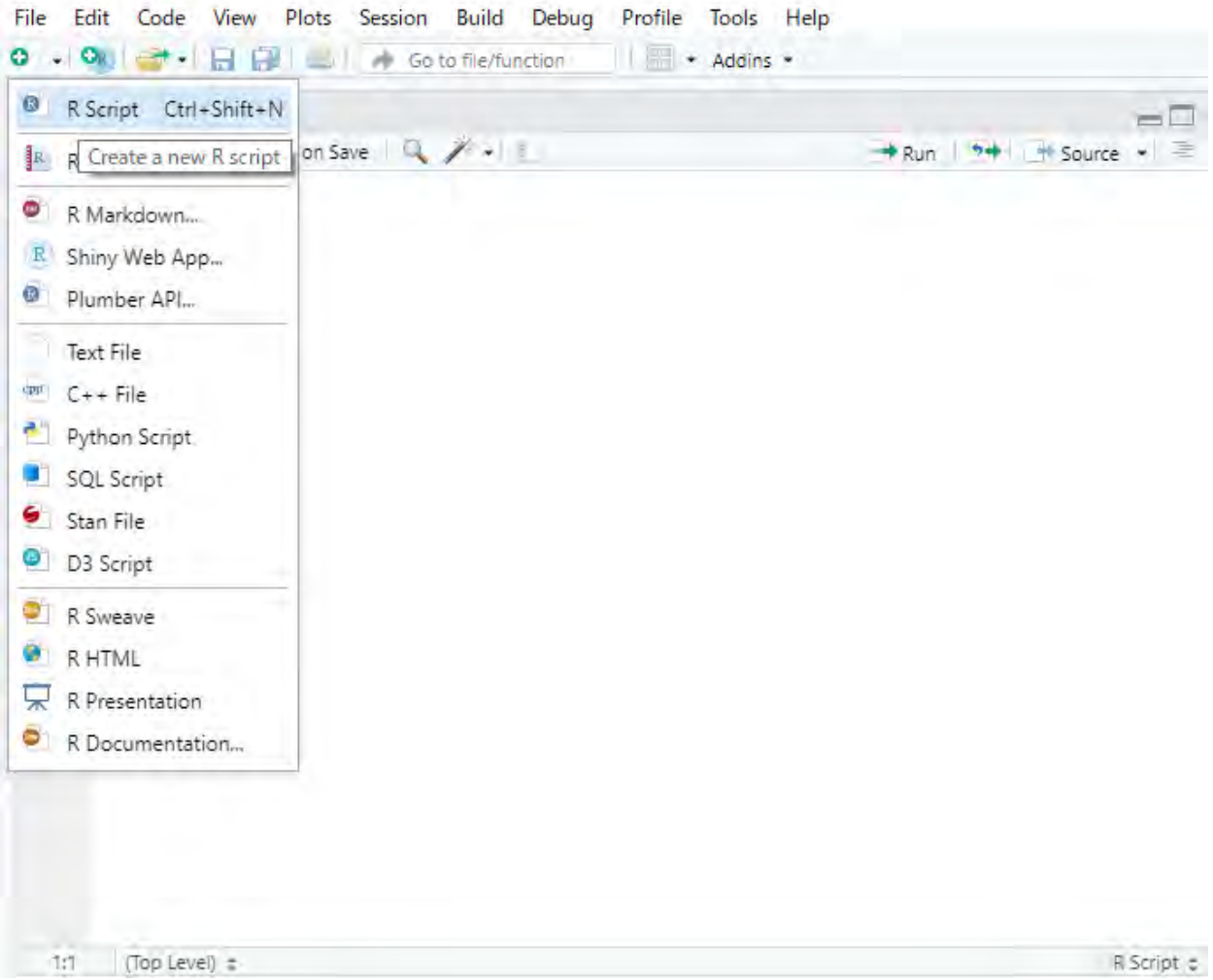


4. Directory name, Browse, Create Project

[プロジェクト名].Rprojという名前のファイルが作成される



Rstudioの基礎



- R Scriptにコマンドを書いていく
- R Script

[File]⇒[New File]⇒[R Script]

[Save current]

Rの基礎

- 基本的な計算

```
> 1 + 1  #足し算
[1] 2
> 5 - 1 #引き算
[1] 4
> 2 * 5  #掛け算
[1] 10
> 10 / 2  #割算
[1] 5
```

Rの基礎

- 計算の順番：PEMDAS
  - Parenthesis(括弧), Exponents(指数), Multiplication(乗算), Division(除算), Addition(加算), Subtraction(減算)

P	Parentheses	括弧	( )
E	Exponents	指数	a^2
MD	Multiplication Division	乗算 除算	×,÷
AS	Addition Subtraction	加算 減算	+, −

Rの基礎

- 計算の順番を確認

```
> 2 * 3 + 4
[1] 10
> 2 * (3 + 4)　#括弧内の計算が優先される
[1] 14
```

Rの基礎

- 変数は関数、分析結果、プロット等のRオブジェクトを保持することができる
- 変数に値を代入する方法
- xに5, yに6を代入してみる

```
> x <- 5
> x
[1] 5
> y = 6
> y
[1] 6
```

Rの基礎

```
> 7 -> z #矢印演算子は逆方向を指すこともできる
> z
[1] 7
> a <- b <- 8 #複数の変数に同値を代入することもできる
> a
[1] 8
> b
[1] 8
```

Rの基礎

```
> a + b    #代入した変数同士の計算
[1] 16
```

- 変数名
  - 英数字、ピリオド(.)、アンダースコア(_)を含むことができる
  - ただし、数字とアンダースコアからは始めることはできない
  - 一般的にはどのような変数かわかるような変数名がよい
- 代入演算子 左矢印 <- がよく使用される

Rの基礎

- 数値を保持していた変数が、その後文字列を保持といったように上書きしていく

```
> x <- 4
> x <- "data"
> x <- 100
> x
[1] 100
```

- 大文字、小文字、全角文字、半角文字は異なる文字として扱われる

```
>  A <- 1   #半角大文字
>  A
[1] 1
```

Rの基礎

```
> a <- 2　#半角小文字
> a
[1] 2
> A <- 3　#全角大文字
> A
[1] 3
> a <- 4　#全角小文字
> a
[1] 4
```

Rの基礎

- ( )で囲むことで、表示することも可

```
> ( a <- 4) #()で囲むことでも表示されます
[1] 4
```

- #(ハッシュ記号)
  - コメントアウト
  - ハッシュに続くすべての文字は同一の行にある限りコメントアウトされ、実行されない

データ型

- numeric, logical(TRUE/FALSE)など

	データ型の確認	データ型の変換
実数	is.numeric()	as.numeric()
論理値	is.logical()	as.logical()
整数	is.integer()	as.integer()
複素数	is.complex()	as.complex()
文字列	is.character()	as.character()

データ型

- class関数
  - 変数が保持するデータ型の確認

```
> class(x)
[1] "numeric"
```

- is.numeric関数
  - 変数がnumericがどうかを確認
  - numeric：数値（整数、小数、正数、負数、ゼロ）

```
> is.numeric(x)
[1] TRUE
```

データ型

- 論理値
- logical
  - TRUEもしくはFALSEのどちらかの値をとる
  - TRUEは 1 と同値、FALSEは 0 と同値
- TRUEは1なので、10をかける(乗算)と10になることを確認

```
> TRUE * 10
[1] 10
> FALSE * 10
[1] 0
```

データ型

- is.logical関数を用いて、logicalかどうかの確認

```
> logi <- TRUE
> class(logi)
[1] "logical"
> is.logical(logi)
[1] TRUE
```

データ型

- 2つの数値または文字列を比較してみる

```
> 2 == 4　#2と4は同値であるか
[1] FALSE
> 2 != 4 #2と4は異なるか
[1] TRUE
> 2 < 4 #2は4より小さいか
[1] TRUE
> 2 <= 4 #2は4以下か
[1] TRUE
> 2 > 4 #2は4より大きいか
[1] FALSE
```

データ型

```
> 2 >= 4 #2は4以上か
[1] FALSE
> "data" == "stats" #dataはstatsと同じか
[1] FALSE
```

ベクトル

- ベクトル(vector)
  - 同じ型の要素を集めたもの
  - 異なる型を混在させることはできない
- 例
  - c(1, 2, 3, 4, 5)は数値1,2,3,4,5をこの順番で並べたvector
  - c("room34", "room5", "room1", "room54")はcharacter型 "room34", "room5", "room1", "room54"を要素にもつvector
- cは結合(combine)を表わし、複数の要素を結合してvectorを作成する

```
> (x <- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
 [1]  1  2  3  4  5  6  7  8  9 10
```

ベクトル

- ベクトル演算
- 1から10までのvectorの各要素との計算

```
> x + 2   #加算
 [1]  3  4  5  6  7  8  9 10 11 12
> x - 3   #減算
 [1] -2 -1  0  1  2  3  4  5  6  7
> x * 3   #乗算
 [1]  3  6  9 12 15 18 21 24 27 30
> x / 4   #除算
 [1] 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00 2.25 2.50
```

ベクトル

- c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)は1:10でも表現できる
- 演算子：
  - 連続した数字の列を生成する

```
> 1:10
 [1]  1  2  3  4  5  6  7  8  9 10
```

- 長さの等しいベクトルを2つ準備

```
> x <- 1:10
> y <- -5:4
```

ベクトル

- 2つのベクトル各要素同士の演算をおこなう

```
> x + y    #加算
 [1] -4 -2  0  2  4  6  8 10 12 14
> x - y    #減算
 [1] 6 6 6 6 6 6 6 6 6 6
```

- 長さの異なる2つのvectorに対する演算
  - 短い方のvectorの再利用が行われる
  - 短い方のvectorの要素が長い方のvectorの長さと同じになるまで繰り返される

```
> x + c(1, 2)
 [1]  2  4  4  6  6  8  8 10 10 12
```

ベクトル

- 長い方のvectorが短い方のvectorの倍数になっていない場合は警告が現れる

```
> x + c(1, 2, 5)
 [1]  2  4  8  5  7 11  8 10 14 11
 警告メッセージ:
 x + c(1, 2, 5) で:
   長いオブジェクトの長さが短いオブジェクトの長さの倍数になっていません
```

ベクトル

- vectorの各要素にアクセスするには、[ ]を使用
- 「1」から始まる

```
> x[1]　#1番目(最初)の要素の抜き出し
[1] 1
> x[0]　#0番目には要素なし
integer(0)
> x[1:5]　#1～5番目の連続した要素の抜き出し
[1] 1 2 3 4 5
>　x[c(1,10)]　#1番目と10番目の要素の抜き出し
[1]　1 10
```

ベクトル

- vectorに名前を与えることもできる

```
> c(red = 5, blue = 6, yellow = 7, green = 8)
  red   blue yellow  green
    5      6      7      8
> v <- 5:8
> names(v) <- c("red", "blue", "yellow", "green")
> v
  red   blue yellow  green
    5      6      7      8
```

関数

- 様々な関数(function)が用意されている
- 例
  - mean関数：数値の平均を求める関数
- 平均
  - n個の観測値を便宜的に$x_1, x_2, \cdots, x_n$と表す
  - xの添え字は観測値の番号
  - 観測した1個目,2個目,⋯,x個目の個体からの観測値

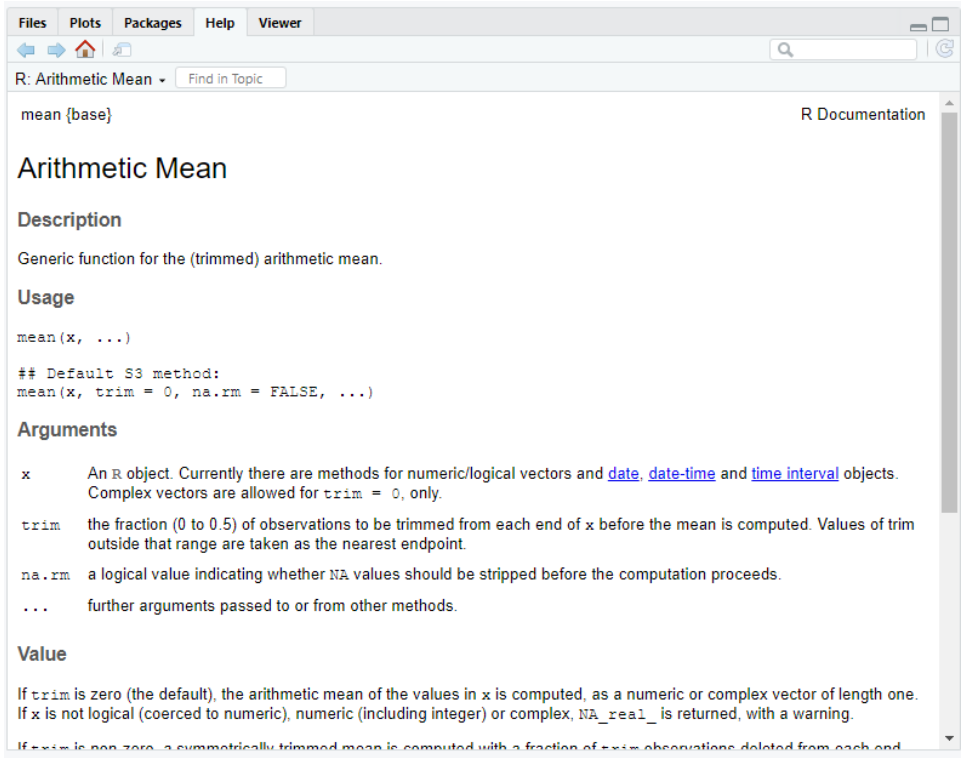
$$\bar{x} = \frac{1}{n}(x_1 + x_2 + \cdots + x_n) = \frac{1}{n}\sum_{i=1}^{n} x_i$$

```
> mean(x)
[1] 5.5
```

関数

- 関数にはドキュメントがあり、参照することができる

```
> help(mean)
> ?mean
starting httpd help server ... done
```



基本統計量

- 不偏分散
- 偏差の2乗の総和をデータ1つあたりにしたもの

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

```
> var(x)
[1] 9.166667
```

- 標準偏差

```
> sqrt(var(x))
[1] 3.02765
> sd(x)
[1] 3.02765
```

基本統計量

- 中央値
  - 観測値を小さい順に並べ、ちょうど真ん中に位置する観測値
  - データの大きさnが奇数の場合：中央に位置する観測値
  - データの大きさnが偶数の場合：真ん中に位置する2つの値の平均が中央値

```
> median(x)
[1] 5.5
```

- 最小値

```
> min(x)
[1] 1
```

- 最大値

```
> max(x)
[1] 10
```

データフレーム

- data.frame：行と列をもつ
  - 行：観測値
  - 列：変数
  - 各列はそれぞれ異なる型のデータを持つことができる
  - 1つの列の要素は同じ型

データフレーム

- data.frame関数を使用して作成

```
> x <- c(1:10)
> y <- c(-4:5)
> z <- c("ice cream", "apple pie", "tiramisu", "chocolate",
"caramel","candy","crepe","cookie","scone","gelato")
> data.frame(x, y, z)
   x  y         z
1   1 -4 ice cream
2   2 -3 apple pie
3   3 -2  tiramisu
4   4 -1 chocolate
5   5  0   caramel
6    6  1     candy
7   7  2     crepe
8   8  3    cookie
9   9  4     scone
10 10  5    gelato
```

データフレーム

- 列名を与える

```
> data.frame(one = x, two = y, dessert = z)
   one two    dessert
1    1  -4 ice cream
2    2  -3 apple pie
3    3  -2  tiramisu
4    4  -1 chocolate
5    5   0   caramel
6    6   1     candy
7    7   2     crepe
8    8   3    cookie
9    9   4     scone
10  10   5    gelato
```

データフレーム

- 行数、列数、または両方を調べる

```
> cafe <- data.frame(one = x, two = y, dessert = z)
> nrow(cafe)
[1] 10
> ncol(cafe)
[1] 3
> dim(cafe)
[1] 10  3
```

データフレーム

- データの一部を表示したい

```
> head(cafe)
  one two   dessert
1   1  -4 ice cream
2   2  -3 apple pie
3   3  -2  tiramisu
4   4  -1 chocolate
5   5   0   caramel
6   6   1     candy
```

データフレーム

```
> head(cafe, 3)
  one two   dessert
1   1  -4 ice cream
2   2  -3 apple pie
3   3  -2  tiramisu
> tail(cafe)
   one two dessert
5    5   0 caramel
6    6   1   candy
7    7   2   crepe
8    8   3  cookie
9    9   4   scone
10  10   5  gelato
```

データフレーム

- 各列が個別のvectorである
- 列の情報を取得する

```
> cafe$dessert
 [1] ice cream apple pie tiramisu  chocolate caramel   candy     crepe
 [8] cookie    scone     gelato
10 Levels: apple pie candy caramel chocolate cookie crepe ... tiramisu
> café[, 3]
 [1] ice cream apple pie tiramisu  chocolate caramel   candy     crepe
 [8] cookie    scone     gelato
10 Levels: apple pie candy caramel chocolate cookie crepe ⋯ tiramisu
```

データフレーム

- 行の情報を取得する

```
> cafe[2] #2行目
 two
1   -4
2   -3
3   -2
4   -1
5    0
6    1
7    2
8    3
9    4
10   5
```

データフレーム

- 各要素の取得

```
> cafe[1, 2] #1行目の3列目
[1] -4
```

- 2から5行目, 1列目と3列目

```
> cafe[2:5, c(1, 3)]
  one   dessert
2   2 apple pie
3   3  tiramisu
4   4 chocolate
5   5   caramel
```

tidy data

人間が理解しやすいデータ形式

studentid	toukei	jyoho	eigo	nihongo
student00	100	40	60	80
student01	20	20	50	10

機械処理しやすいデータ形式

studentid	test	score
student00	toukei	100
student01	toukei	20
student00	jyoho	40
student01	jyoho	20
student00	eigo	60
student01	eigo	50
student00	nihongo	80
student01	nihongo	10

tidy data

- tidy dataの定義
  - 1つ列が1つの変数を表わす
  - 1つの行が1つの観測を表わす
  - 1つのテーブルが1つのデータセットだけを含む

- データフレーム ⇒ tidy data

tidy data

```
> score_data <- data.frame(
  studentid = c("student00", "student01"),
  toukei = c(100,20),
  jyoho = c(40, 20),
  eigo = c(60, 50),
  nihongo = c(80, 10),
  stringsAsFactors = FALSE
 )
> score_data
  studentid toukei jyoho eigo nihongo
1 student00    100    40   60      80
2 student01     20    20   50      10
> library(tidyverse)
> score_tidydata <- gather(score_data,
  key = "test", value = "score",
  toukei, jyoho, eigo, nihongo
 )
```

```
> score_tidydata
studentid    test score
1 student00  toukei   100
2 student01  toukei    20
3 student00   jyoho    40
4 student01   jyoho    20
5 student00    eigo    60
6 student01    eigo    50
7 student00 nihongo    80
8 student01 nihongo    10
```

- tidy data ⇒ データフレーム

```
> spread(score_tidydata, key = test, value = score)
  studentid eigo jyoho nihongo toukei
1 student00   60    40      80    100
2 student01   50    20      10     20
```

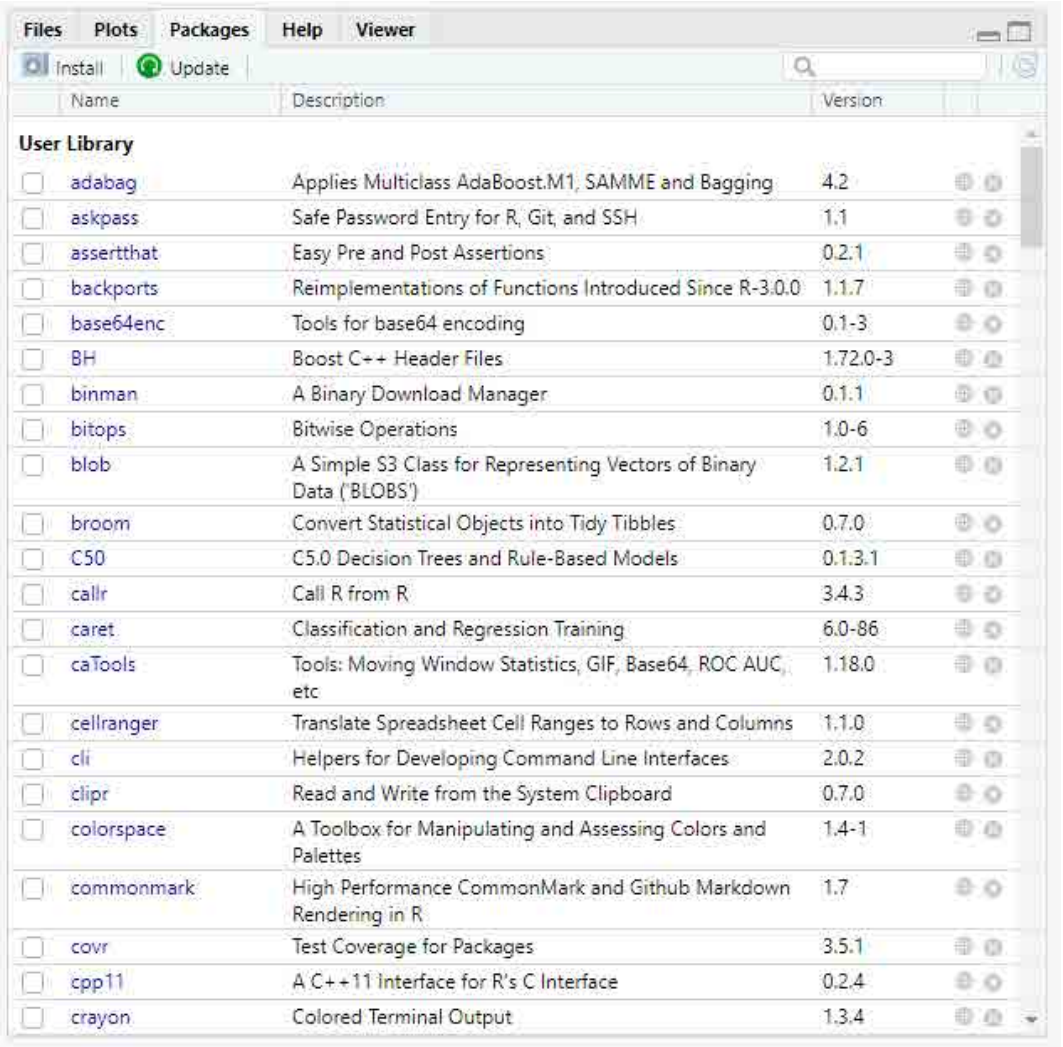
パッケージ

- パッケージの中にはデータを含んでいるものもある
- 現状で登録されているパッケージを調べる

```
> p = installed.packages()
> rownames( p )
  [1] "abind"          "acepack"        "adabag"
  [4] "anytime"        "ape"            "aplpack"
  [7] "arm"            "askpass"        "assertthat"
 [10] "backports"      "base64enc"      "bayesplot"
 [13] "BH"             "binman"         "BiocManager"
```

一部省略

パッケージ



- パッケージのインストール

```
> install.packages("パッケージ名")
```

- パッケージの読込

```
> library(パッケージ名)
> require(パッケージ名)
```

データの読込

- 利用可能なデータのリストを確認

```
> data()
```

```
Data sets in package 'datasets':

AirPassengers       Monthly Airline Passenger Numbers
                    1949-1960
BJsales             Sales Data with Leading Indicator
BJsales.lead (BJsales)
                    Sales Data with Leading Indicator
BOD                 Biochemical Oxygen Demand
CO2                 Carbon Dioxide Uptake in Grass Plants
ChickWeight         Weight versus age of chicks on different
                    diets
DNase               Elisa assay of DNase
EuStockMarkets      Daily Closing Prices of Major European
                    Stock Indices, 1991-1998
Formaldehyde        Determination of Formaldehyde
HairEyeColor        Hair and Eye Color of Statistics
                    Students
Harman23.cor        Harman Example 2.3
Harman74.cor        Harman Example 7.4
Indometh            Pharmacokinetics of Indomethacin
InsectSprays        Effectiveness of Insect Sprays
JohnsonJohnson      Quarterly Earnings per Johnson & Johnson
                    Share
LakeHuron           Level of Lake Huron 1875-1972
LifeCycleSavings    Intercountry Life-Cycle Savings Data
Loblolly            Growth of Loblolly pine trees
Nile                Flow of the River Nile
Orange              Growth of Orange Trees
OrchardSprays       Potency of Orchard Sprays
PlantGrowth         Results from an Experiment on Plant
                    Growth
Puromycin           Reaction Velocity of an Enzymatic
                    Reaction
Seatbelts           Road Casualties in Great Britain 1969-84
```

データの読込

- ggplot2にはdiamondsのデータセットが含まれている
- 約5万4千個のダイヤモンドに関するデータ

```
> library(ggplot2)
> data(diamonds)
> head(diamonds)
# A tibble: 6 x 10
  carat cut       color clarity depth table price     x     y     z
  <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

Prices of over 50,000 round cut diamonds

Description

A dataset containing the prices and other attributes of almost 54,000 diamonds. The variables are as follows:

Usage

```
diamonds
```

Format

A data frame with 53940 rows and 10 variables:

price

: price in US dollars (¥$326–¥$18,823)

carat

: weight of the diamond (0.2–5.01)

cut

: quality of the cut (Fair, Good, Very Good, Premium, Ideal)

color

: diamond colour, from D (best) to J (worst)

clarity

: a measurement of how clear the diamond is (I1 (worst), SI2, SI1, VS2, VS1, VVS2, VVS1, IF (best))

x

: length in mm (0–10.74)

y

: width in mm (0–58.9)

z

: depth in mm (0–31.8)

depth

: total depth percentage = z / mean(x, y) = 2 * z / (x + y) (43–79)

table

: width of top of diamond relative to widest point (43–95)

グラフ

- 1変数のデータのグラフ
- ヒストグラム
- 先ほど使用したdiamondsデータ
- summary関数でデータの内容を確認

```
> summary(diamonds)
     carat               cut        color        clarity          depth      
 Min.   :0.2000   Fair      : 1610   D: 6775   SI1     :13065   Min.   :43.00  
 1st Qu.:0.4000   Good      : 4906   E: 9797   VS2     :12258   1st Qu.:61.00  
 Median :0.7000   Very Good:12082   F: 9542   SI2     : 9194   Median :61.80  
 Mean   :0.7979   Premium   :13791   G:11292   VS1     : 8171   Mean   :61.75  
 3rd Qu.:1.0400   Ideal     :21551   H: 8304   VVS2    : 5066   3rd Qu.:62.50  
 Max.   :5.0100                     I: 5422   VVS1    : 3655   Max.   :79.00  
                                    J: 2808   (Other): 2531                  
```

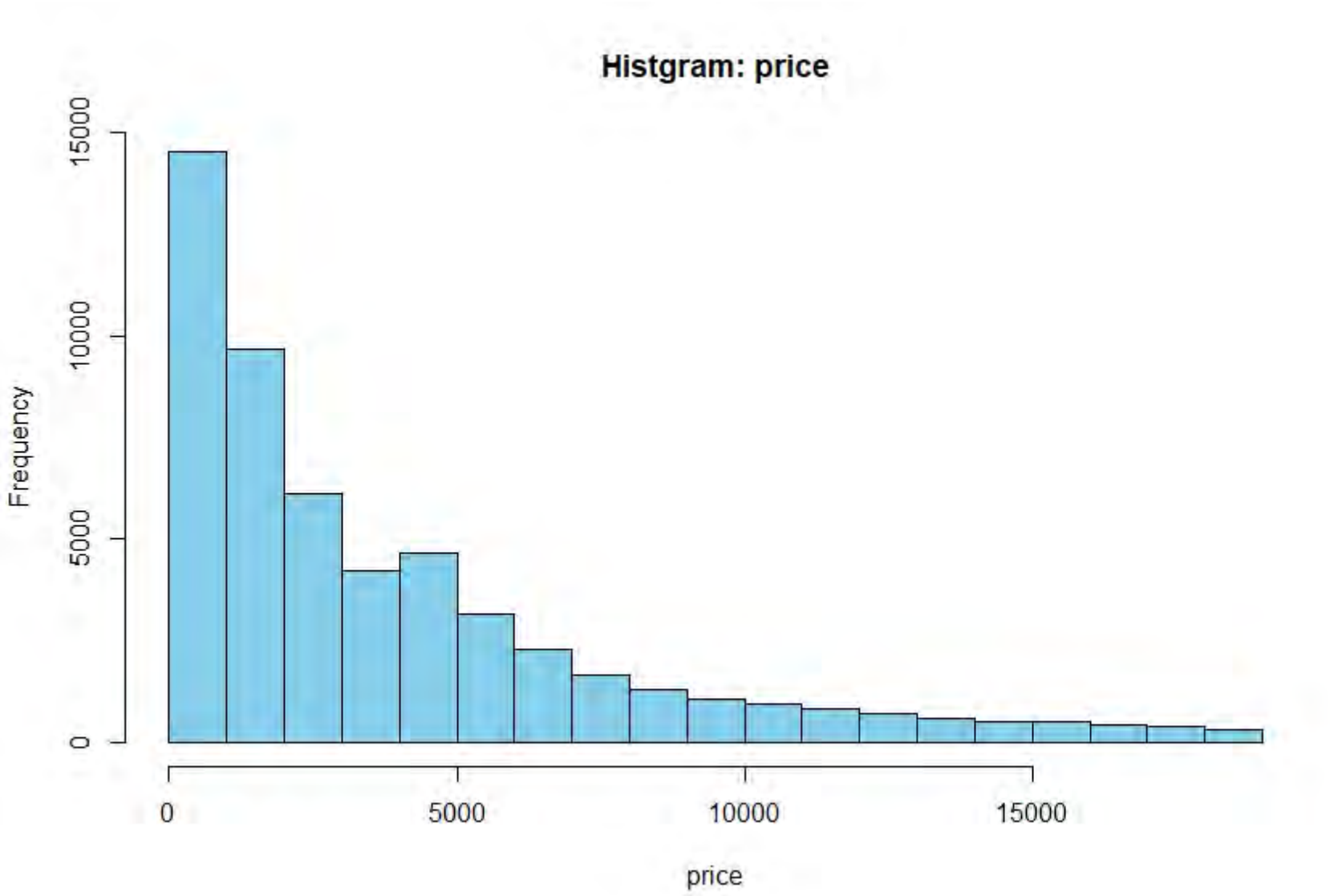
一部省略

グラフ

- ヒストグラム：1変数
  - 度数分布表から作成する柱状のグラフのこと
  - ベル型：左右対称になっているベルのような形の場合
  - 右にすそが長い：ピークが左にあり、大きな値が存在する場合
  - 左にすそが長い：ピークが右にあり、小さな値が存在する場合
  - 一様：ある範囲内でどの値も同程度に出現する場合

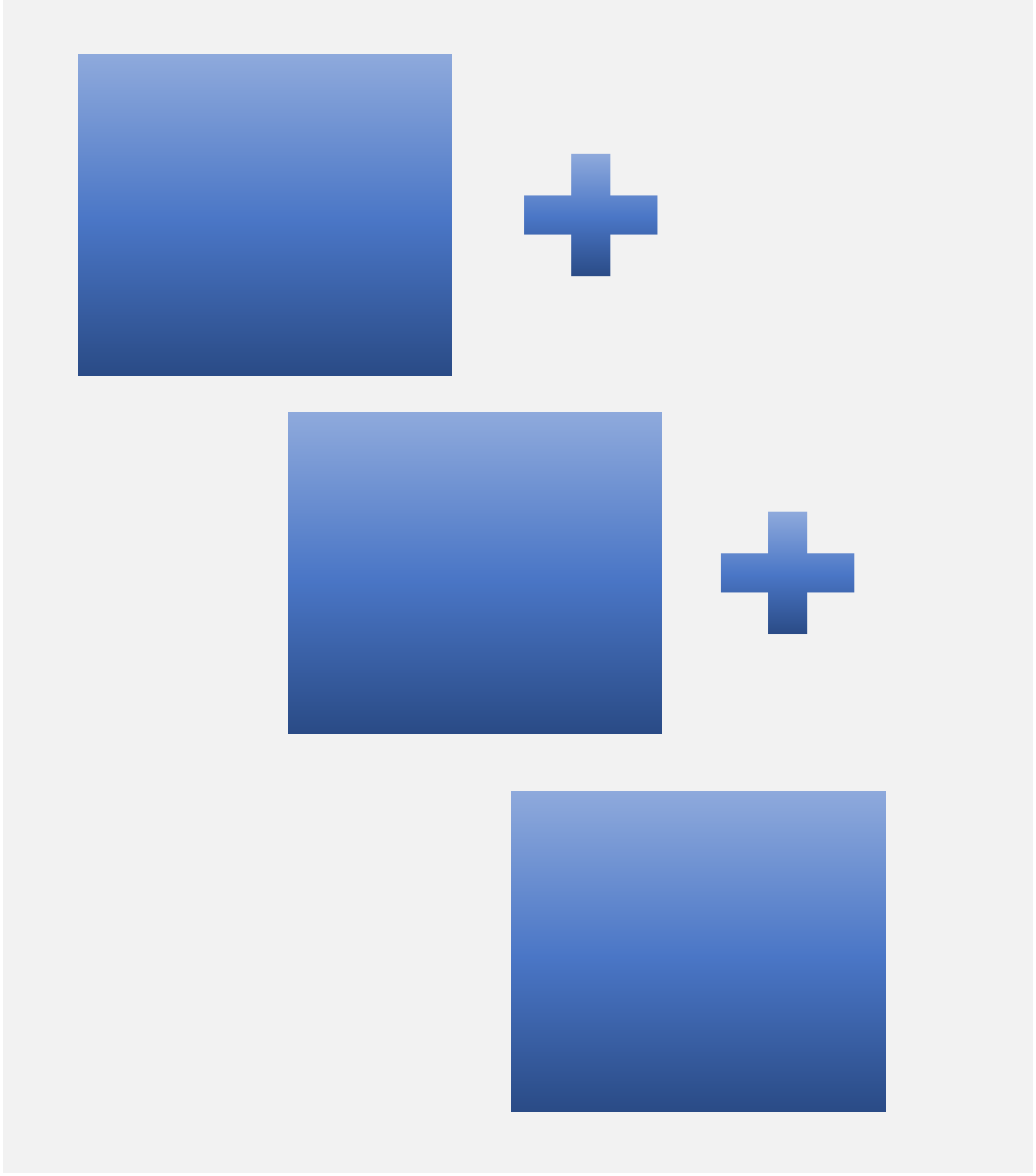
- diamondsのpriceのヒストグラムを描いてみる
  - priceの分布

```
> hist(diamonds$price, main = "Histgram: price", xlab = "price", col = "skyblue", breaks = "Sturges")
```



グラフ

- ggplot2パッケージ
  - データの可視化
  - Grammar of Graphics：複数の工程に可視化の作業を分割
  - 同一のデータセットから一連の工程を部分的に変更可
  - 様々なグラフを書くことができる
  - library(tidyverse)にggplot2は含まれている
  - データセット：データフレームもしくはtibble



- キャンバスの準備

```
> library(ggplot2)　#もしくはlibrary(tidyverse)
> ggplot(data = diamonds)
```

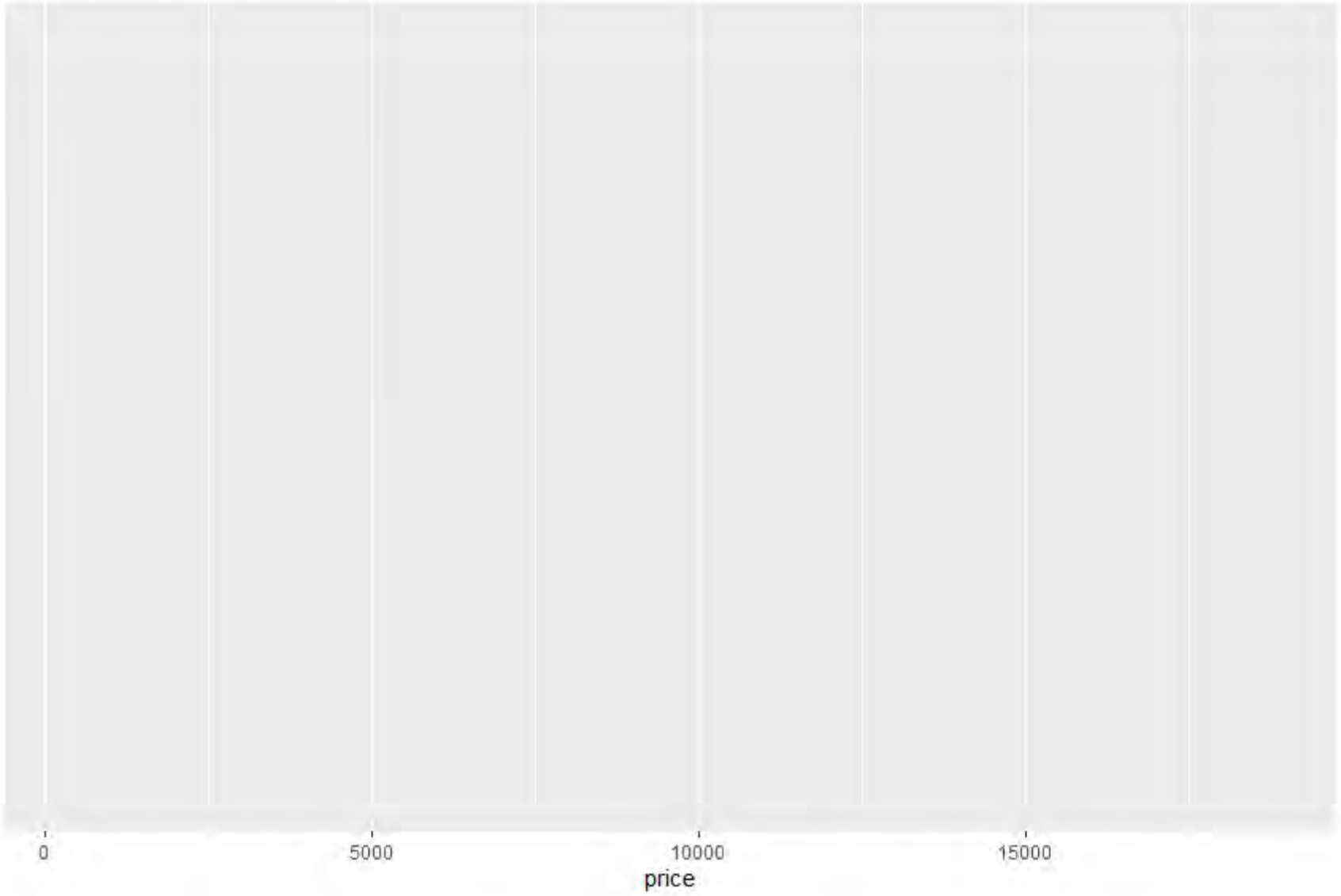
グラフ

- 幾何学的オブジェクト(geometric object)
  - geom_からはじまる
  - 各レイヤを＋でつなぐことで、レイヤを重ねることができる

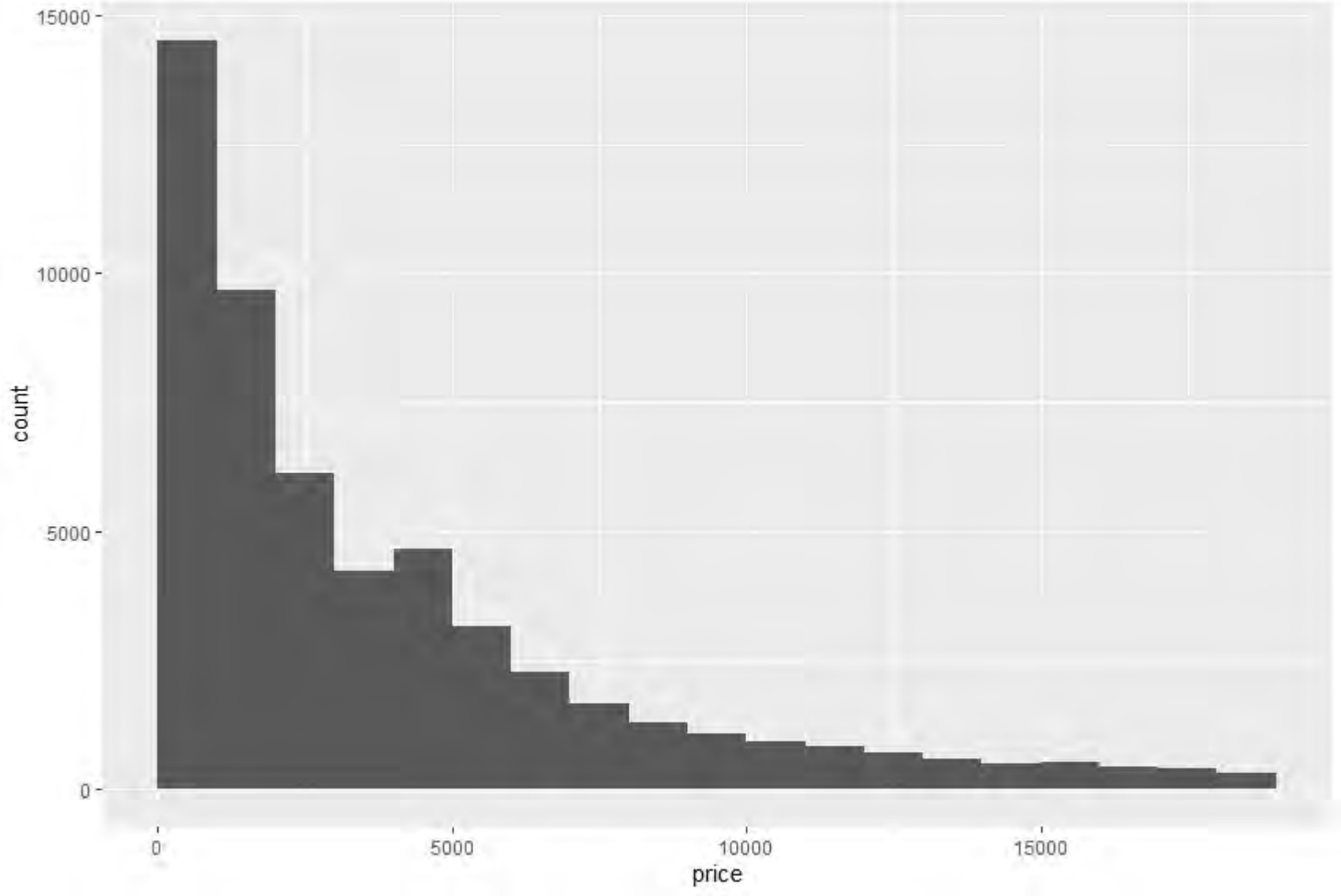
ヒストグラム	geom_histogram()
棒グラフ	geom_bar()
折れ線グラフ	geom_line()
散布図	geom_point()
箱ひげ図	geom_boxpoint()
テキストラベル	geom_text()
誤差棒	geom_errorbar()

- 作成した下地に重ねていく、priceをx軸に

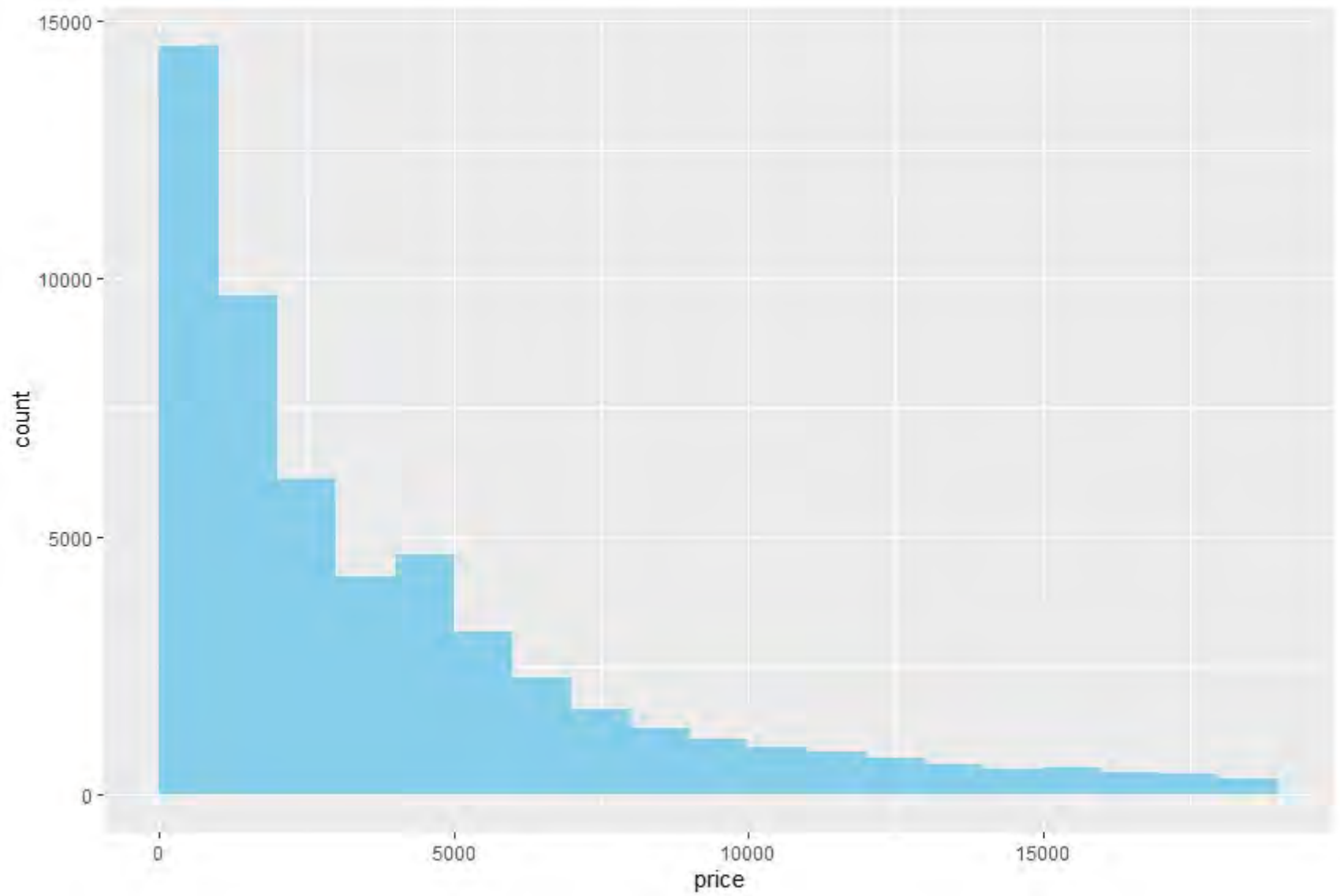
```
> ggplot(data = diamonds) +
  aes(x=price)
```



```
> x <- diamonds$price            # データ
> num <- nclass.Sturges(x)       #階級数
> breaks <- pretty(x, num)       # 階級をだす
> ggplot(data = diamonds) +
 aes(x = price)+
 geom_histogram()
```



```
> ggplot(data = diamonds) +
  aes(x=price) +
  geom_histogram(fill = "skyblue", breaks = breaks)
```

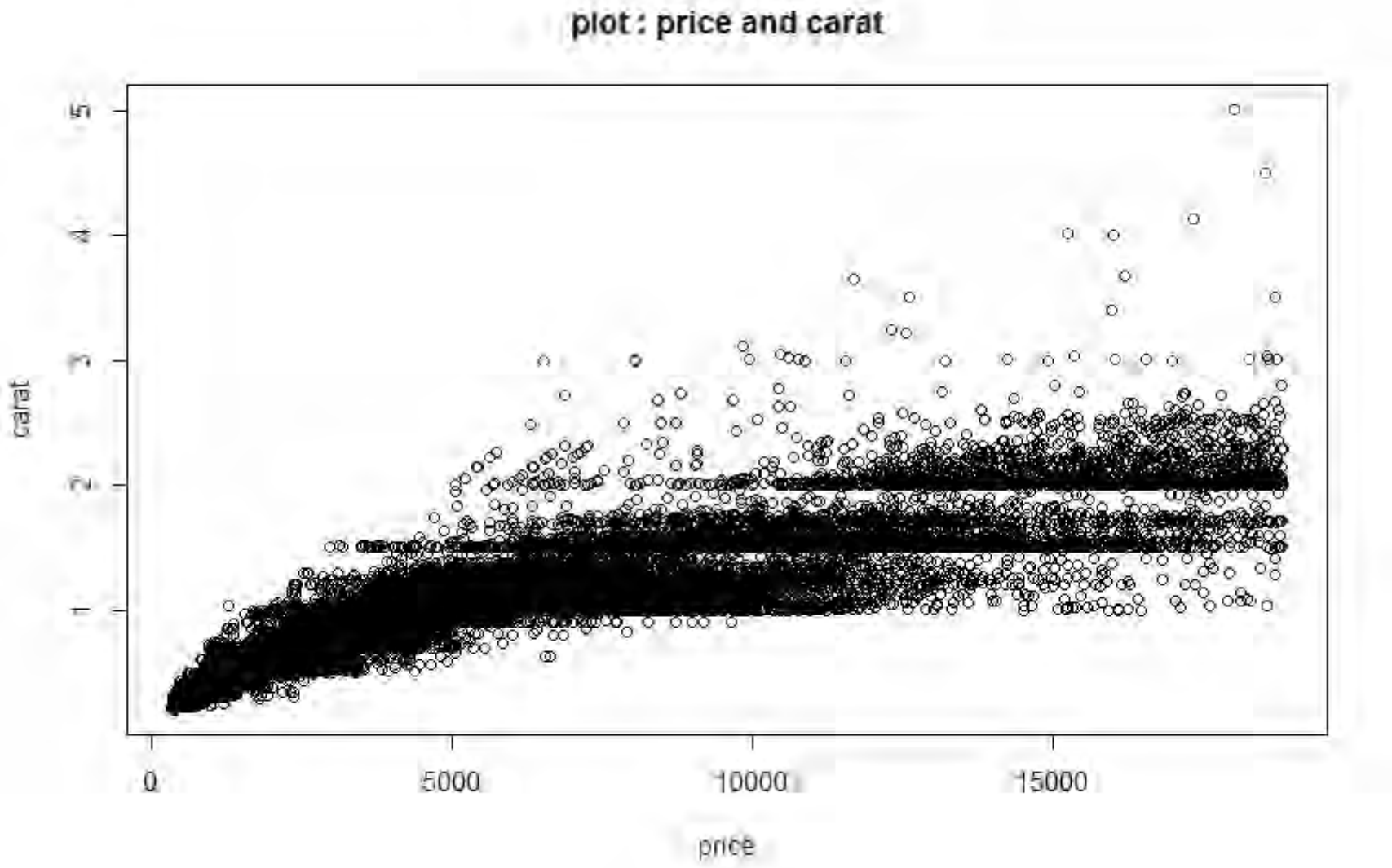


グラフ

- 散布図：2変数の関係をみる
- 散布図（相関図, scatter diagram）
- 縦軸と横軸に2項目の量や大きさを対応させ、データを点でプロットした図
- 2つの変数の値を図にすることで、お互いにどのような関係があるかのかを知ることができる

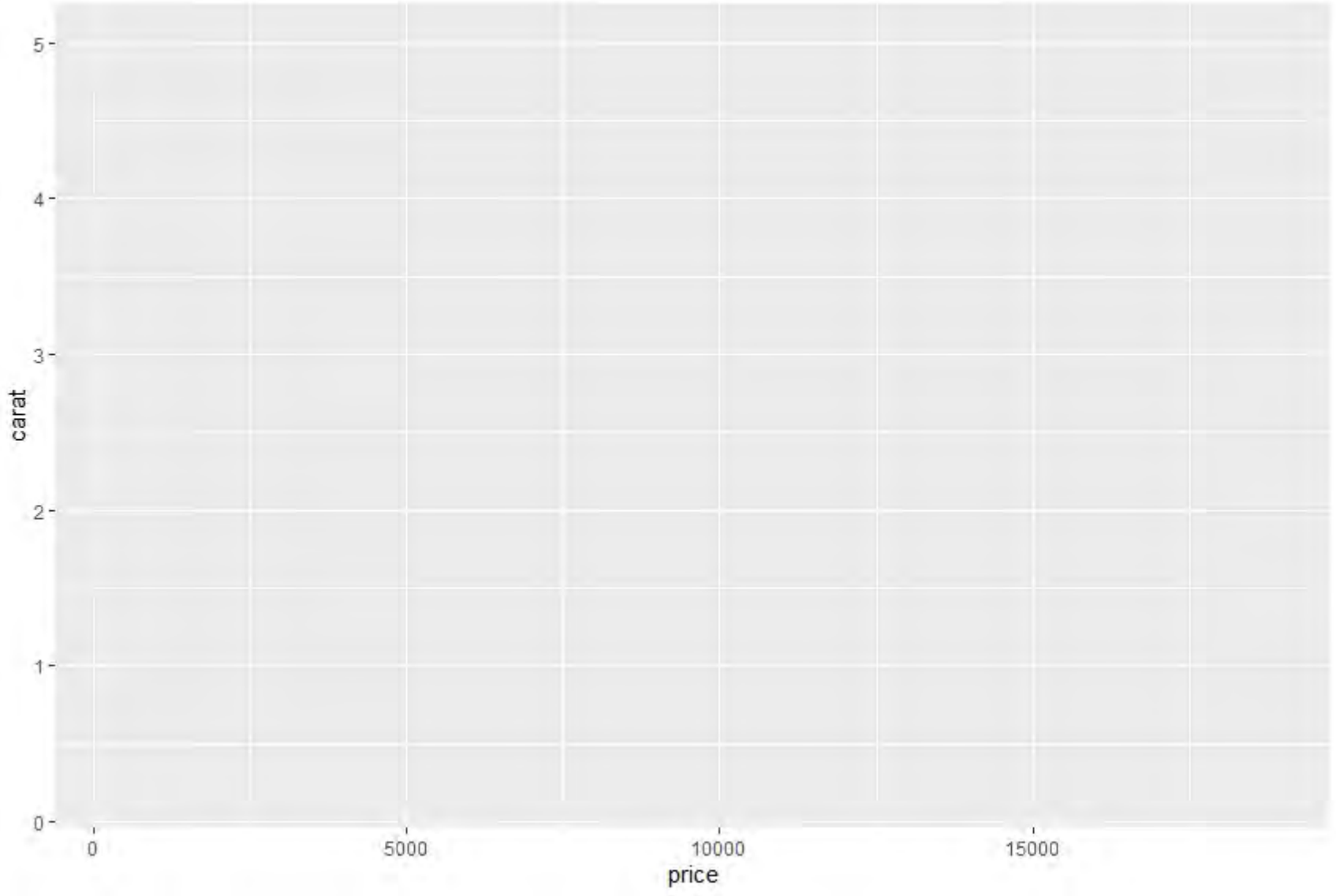
- diamondsのprice(*x*軸)に対するcarat(*y*軸)のプロット図を書いてみる
  - geom_point()
  - priceが高くなると共に、caratは大きくなる

```
> plot(carat~price, data = diamonds, main = "plot : price and carat")
```

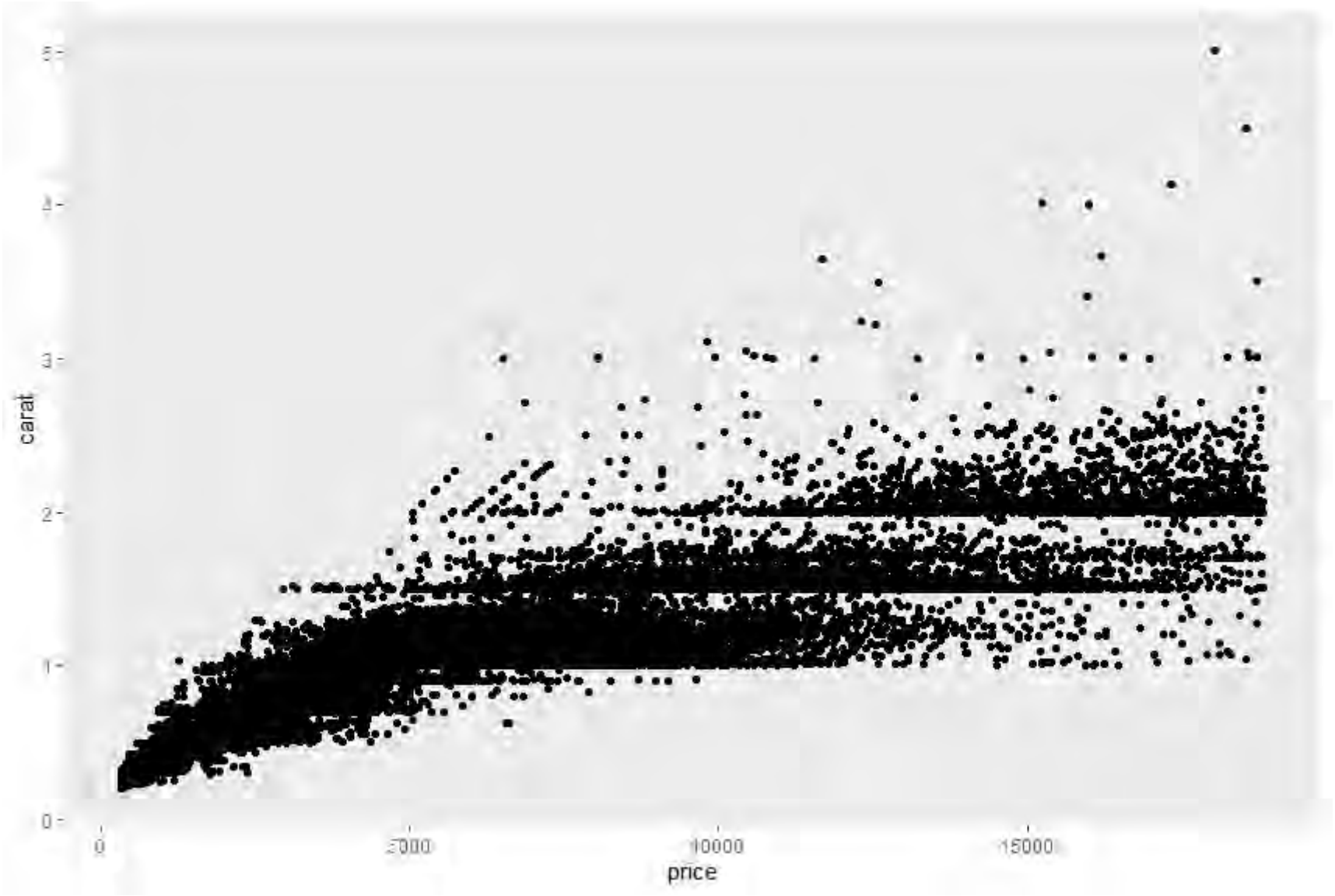


- 作成した下地に重ねていく、price x軸・carat y軸

```
> ggplot(diamonds) +
  aes(x = price, y = carat)
```



```
> ggplot(diamonds) +
  aes(x = price, y = carat) +
  geom_point()
```

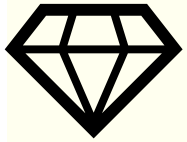


グラフ

- 層を追加していく
- 繰り返しかくのは大変なので、変数にggplotオブジェクトを保存

```
> e <- ggplot(diamonds) +
  aes(x = price, y = carat)
```

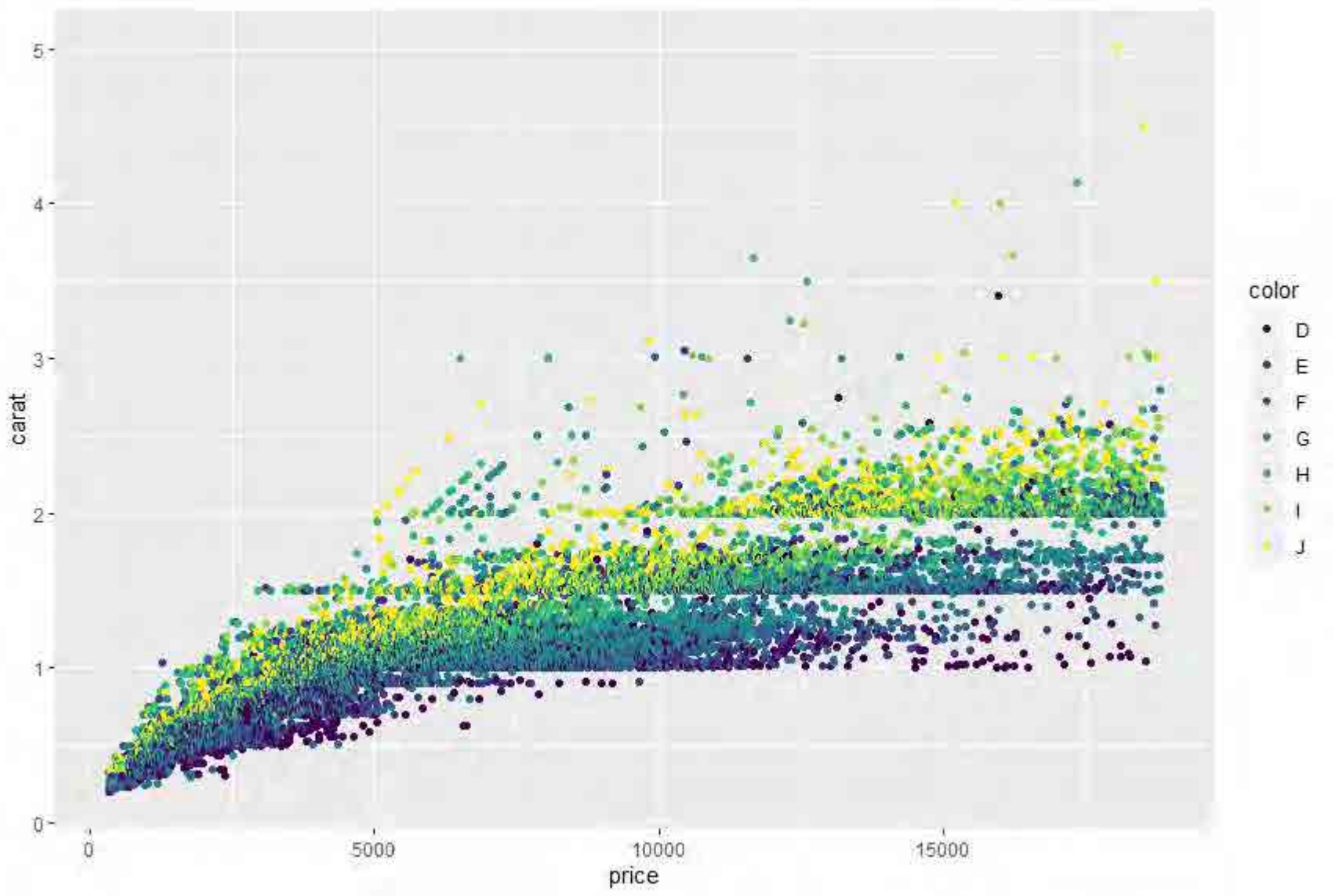
- diamondsデータのcolor別に各色を割り当てて、散布図を作成する



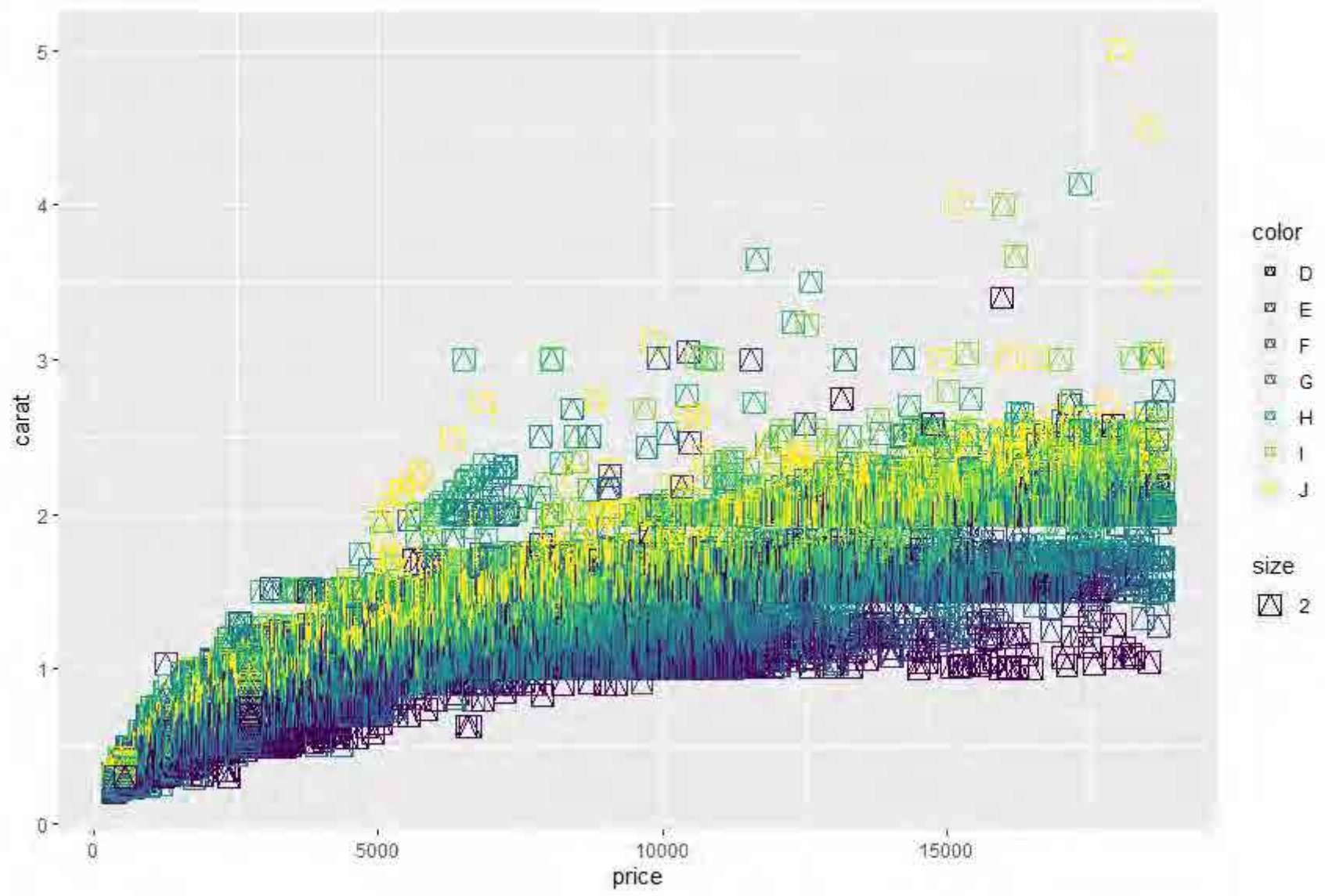
D(無色)　⇔　J(ほぼ無色)

- aes関数のcolor = ではデータに応じて色を変える
- 凡例が自動的に生成される

```
> e +
  geom_point(aes(color = color))
```

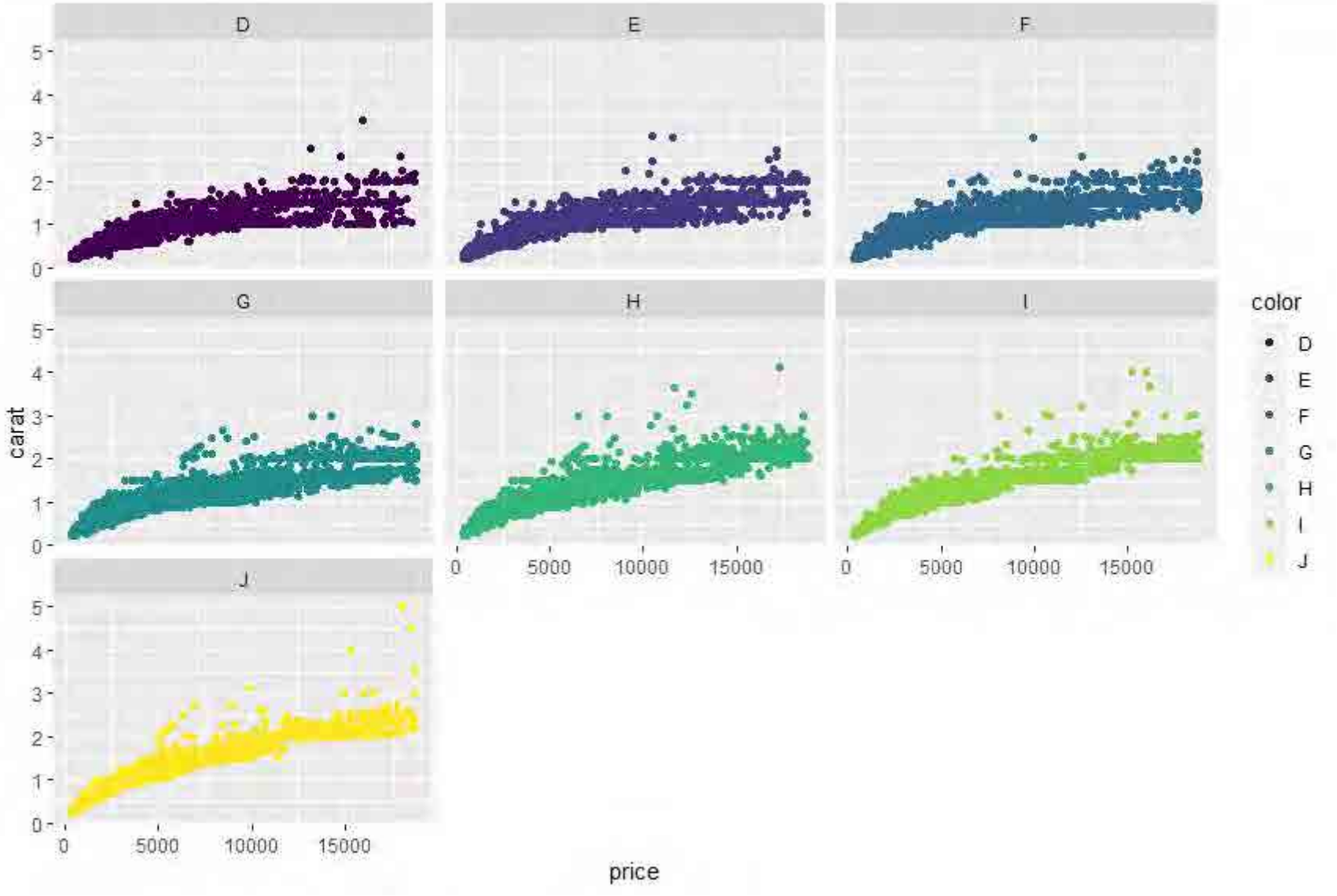


```
> e +
  geom_point(shape = 14 , size = 2, aes(color = color))
```



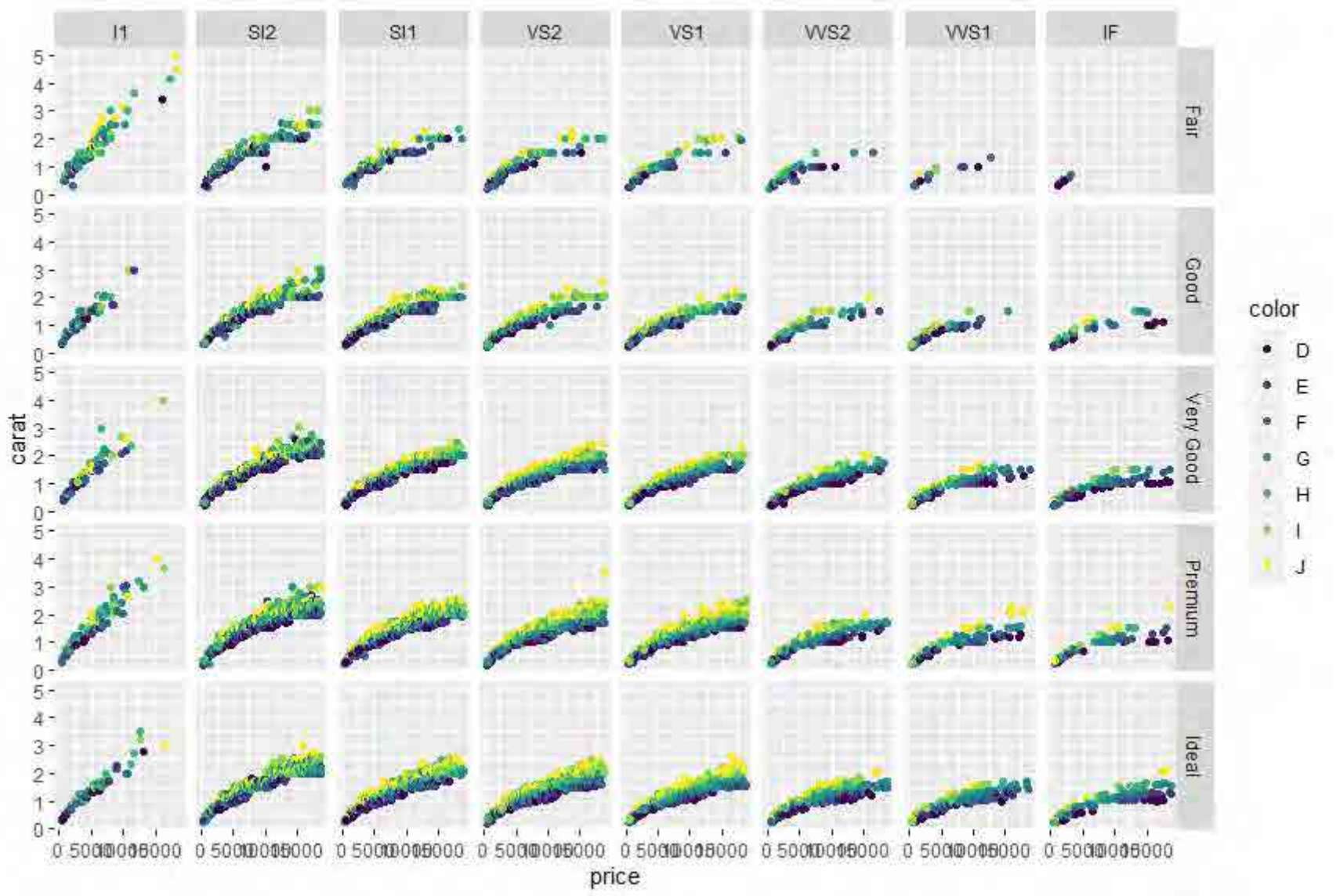
- 水準ごとに異なる描写パネルを用意
- facet_wrapは指定した変数の水準に応じて、データを分割

```
> e +
  geom_point(aes(color = color)) +
  facet_wrap(~color)
```



- facet_grid()では横並びか縦並びか選べる&2変数も可

```
> e +
  geom_point(aes(color = color)) +
  facet_grid(cut~clarity)
```

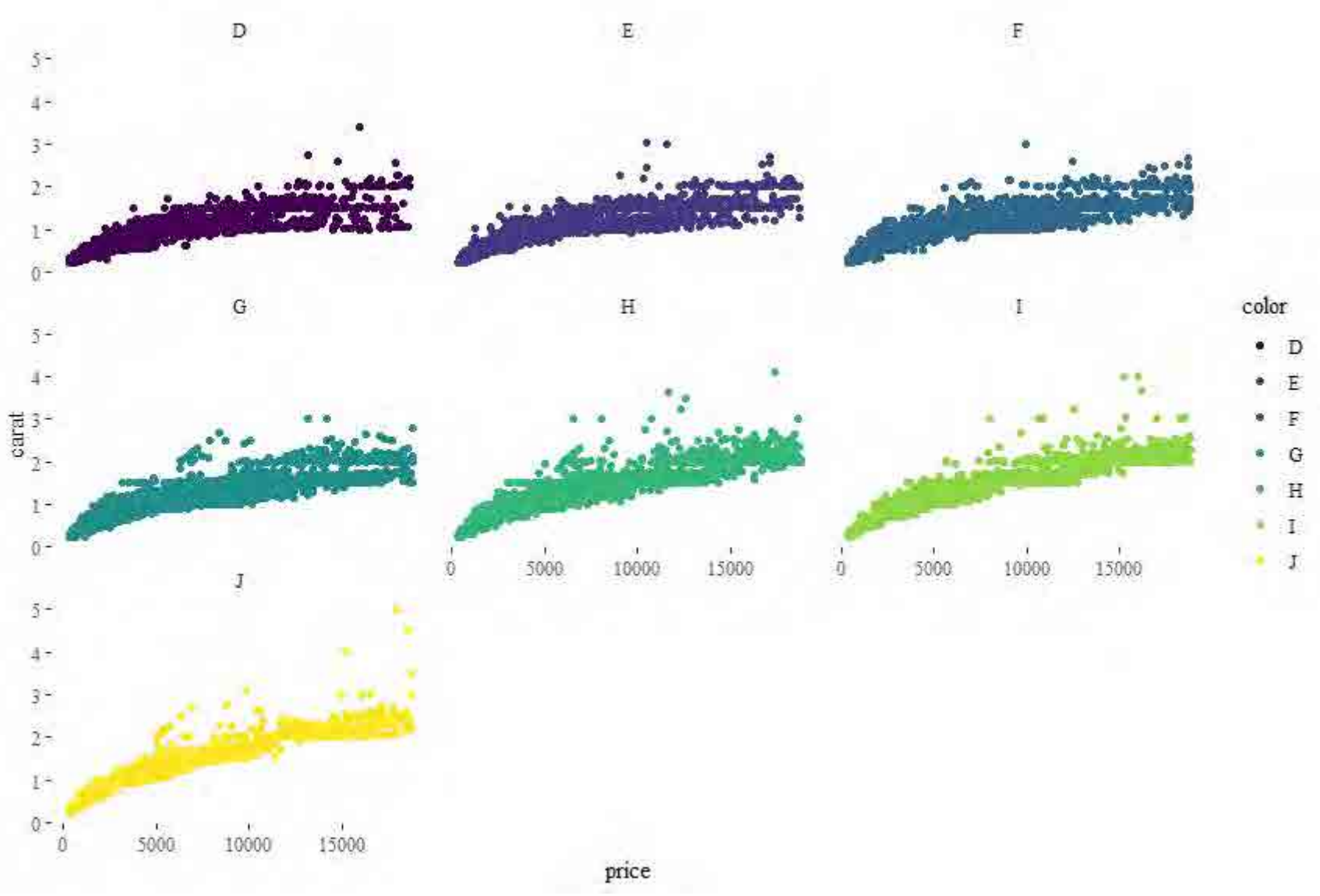


グラフ

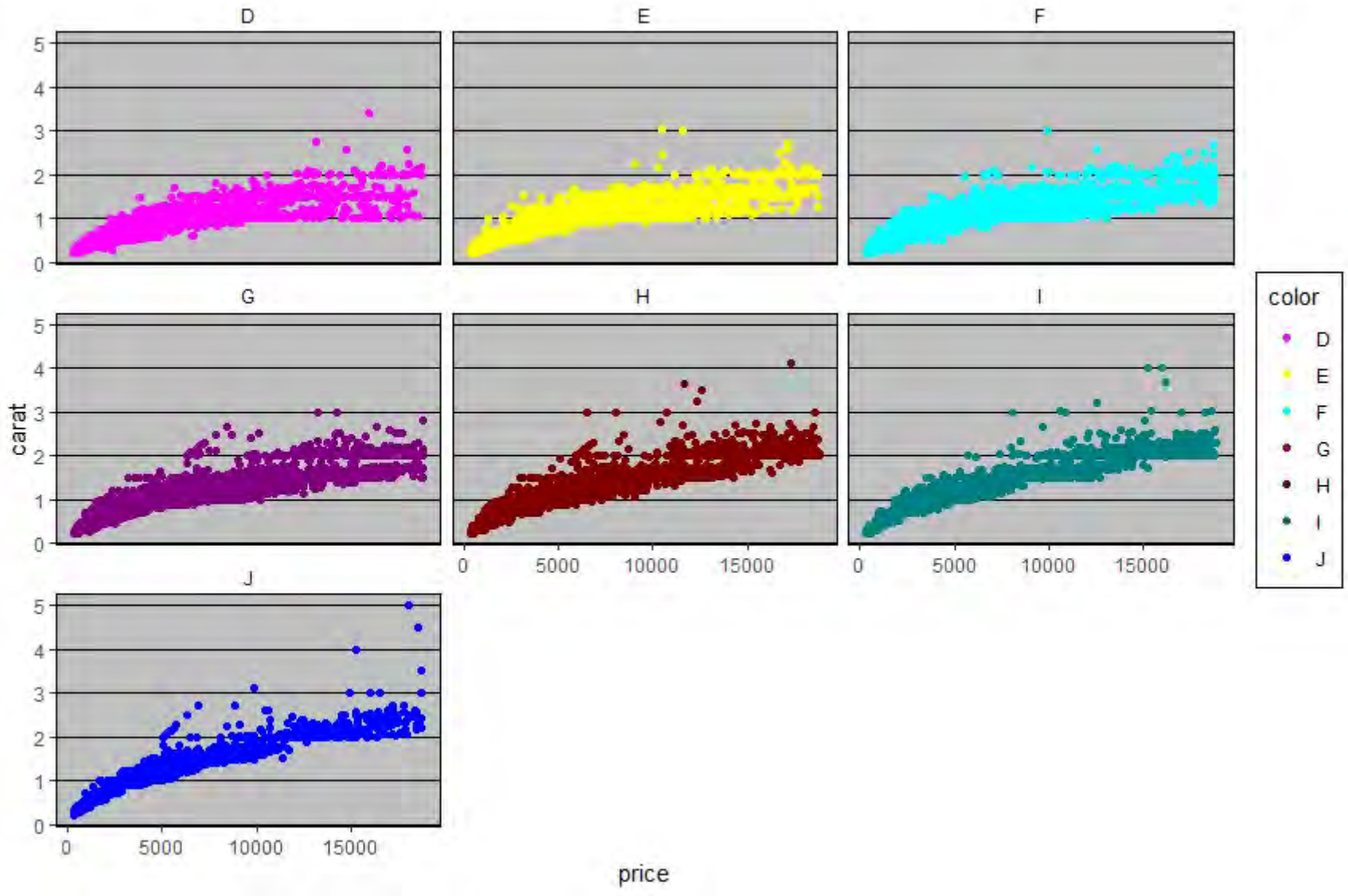
- ggtheme
  - よく使われるグラフのスタイルのテーマを集めたパッケージ

```
> install.packages("ggthemes")
> library(ggthemes)
```

```
> e +
  geom_point(aes(color = color)) +
  facet_wrap(~color) +
  theme_tufte()
```



```
> e +
  geom_point(aes(color = color))+
  facet_wrap(~color) +
  theme_excel() +
  scale_colour_excel()
```



グラフ

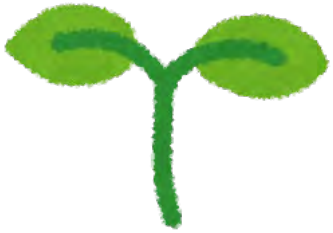
- 保存

```
> hozon <- e +
  geom_point(aes(color = color))+
  facet_wrap(~color) +
  theme_excel() +
  scale_colour_excel()
> ggsave("diamonds.png", hozon, width = 7, height = 4, dpi = 300)
```

特徴抽出

クラスタリング

- クラスター分析
  - データ同士の似ているところ(類似度)を距離に置き換え、これをもとにいくつかのグループにわけることで、データ全体の特徴をとらえる試み
  - 統計的な考え方でデータをグループ分け
- 階層的クラスター分析
  - グループの形成状態を樹形図(デンドログラム)で表現する階層的な分別方法
- 非階層的クラスター分析
  - あらかじめグループ数を決め、その中心となる要素との距離を最小にすることで分類する非階層的な方法



階層的クラスター分析

- データ間の類似度と非類似度をそれぞれ距離に置き換えて、最も似ているデータから順に集めてクラスターを使っていく方法
- 樹形図は逆さにした木の構造に似ていることからツリー構造とよばれることもある
- ラベルが付いている部分は「葉」
- 葉と葉との距離（葉から上に伸びている線が他とつながるまでの高さ）が短いほど似ている

階層的クラスター分析

- クラスターができ、作られていく様子を樹形図（デンドログラム）で示す
- 樹形図
  - いくつかの個体が階層的に集まり、1つのクラスターを形成し、複数のクラスターが最終的には1つのクラスターになる様子が見て取れる
  - 樹形図をある高さ（距離）のところで線を引いて切断することで、クラスターの数が定まり個体の分類が定まる

階層的クラスター分析のプロセス

①距離（または類似度）を求めるデータを行列にする

②①の行列に対して個体間の距離（類似度）を計算した結果を行列にまとめる

③クラスター分析の方法（再近隣法とか最遠隣法など）を選択し、コーフェン行列を求める

④コーフェン行列に基づいて樹形図を作成する

①データ行列⇒②距離行列⇒③コーフェン行列⇒④樹形図を描く

①データ行列

$$\begin{bmatrix} x_{12} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

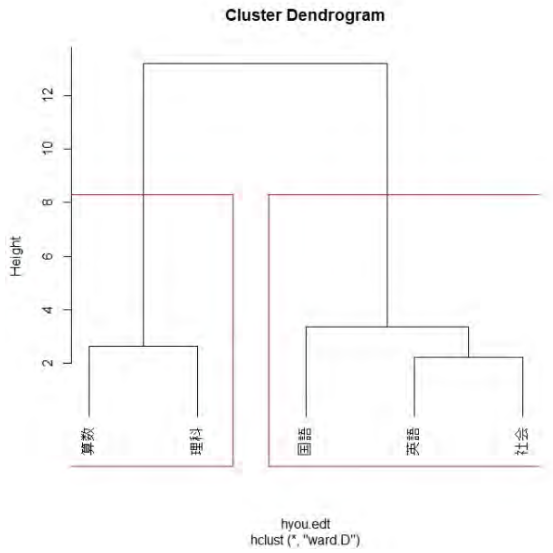
②距離行列

$$\begin{bmatrix} 0 & & & \\ d_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ d_{m1} & d_{m2} & \cdots & 0 \end{bmatrix}$$

③コーフェン行列

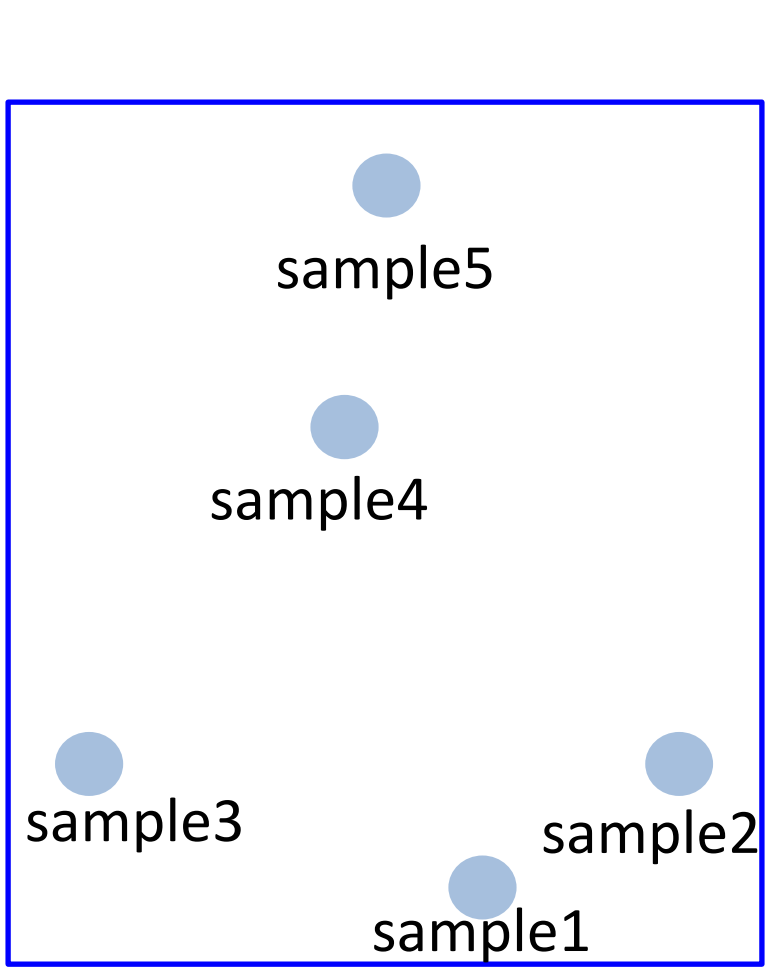
$$\begin{bmatrix} 0 & & & \\ c_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ c_{m1} & c_{m2} & \cdots & 0 \end{bmatrix}$$

④樹形図

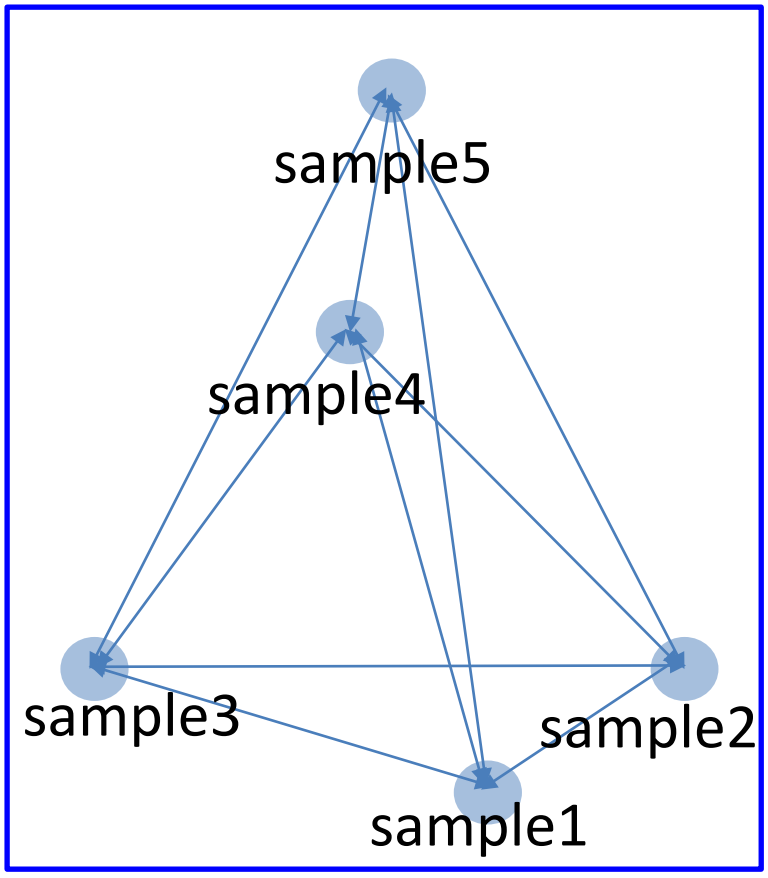


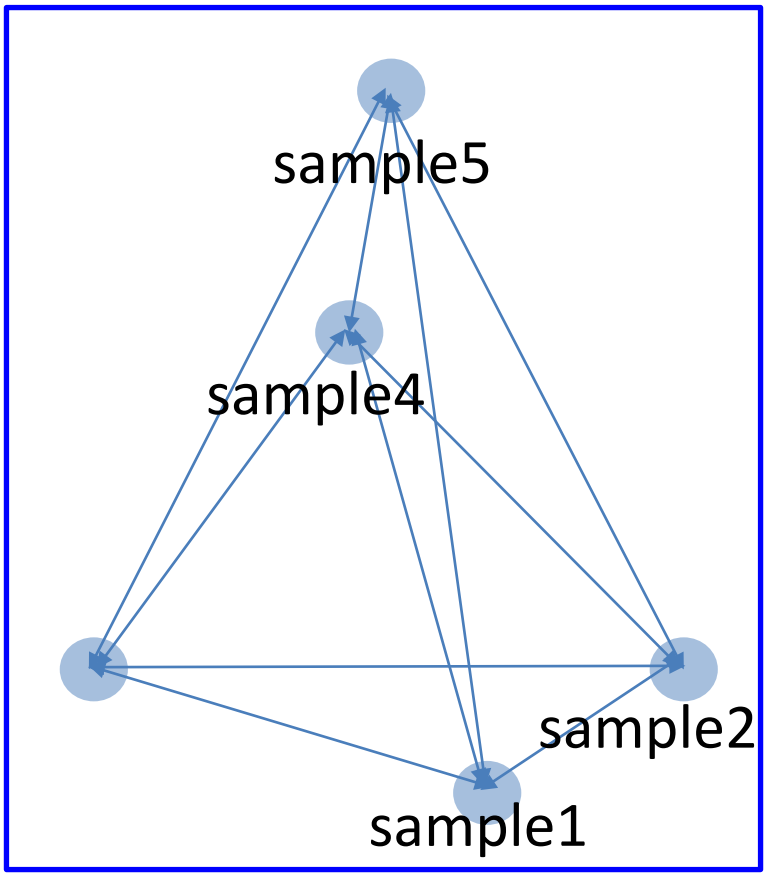
金明哲.(2016).「定性的データ分析」.共立出版株式会社.

階層的クラスター分析のプロセス

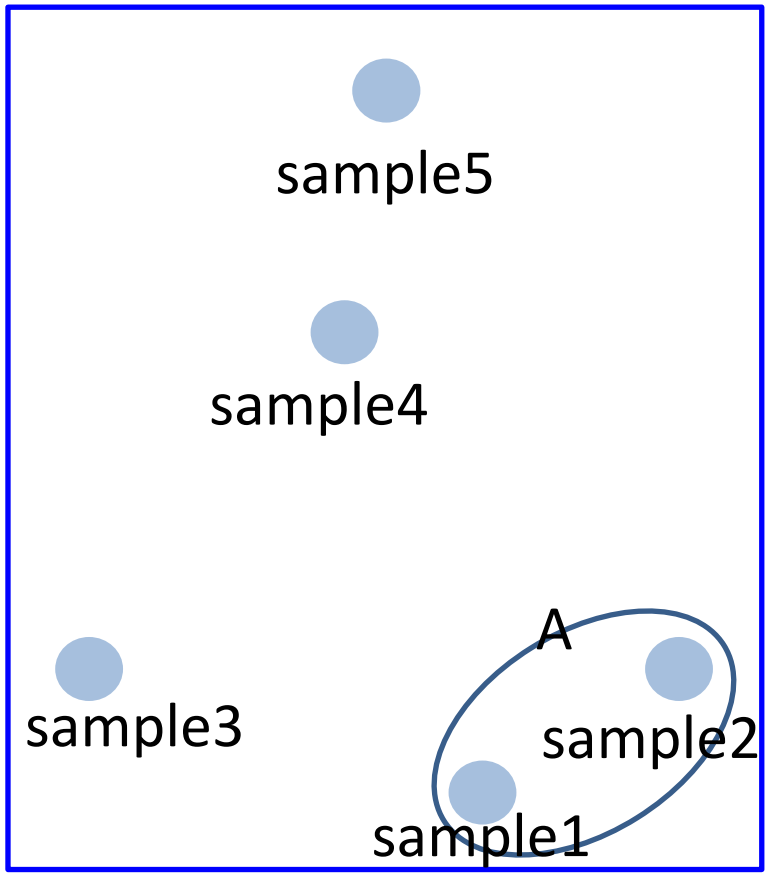


サンプル間の距離を計算



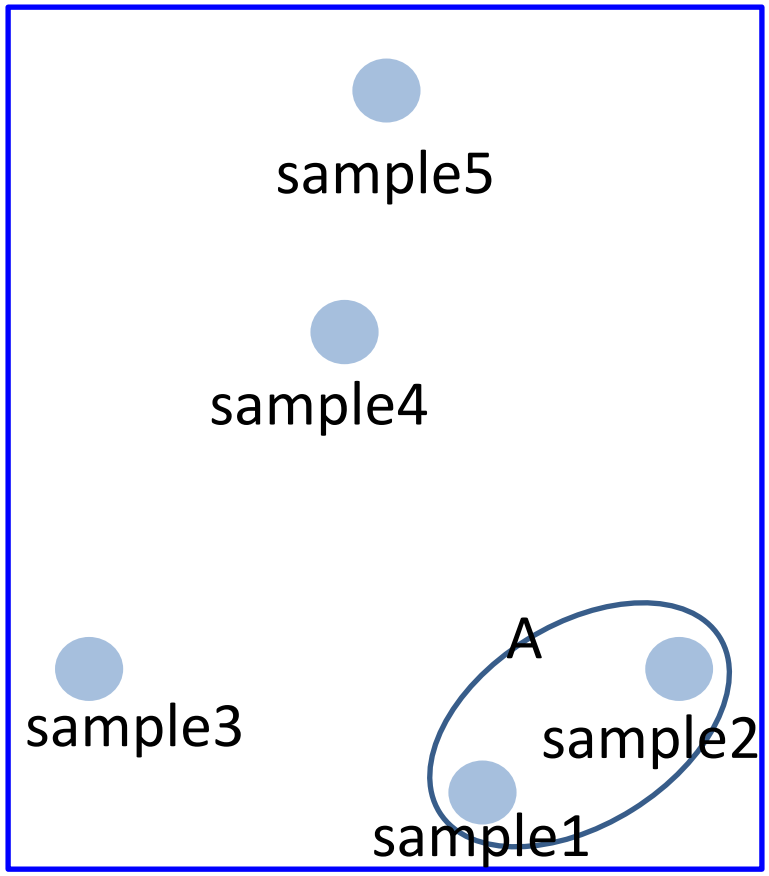


最も近いサンプルを併合



	sample1	sample2	sample3	sample4	sample5
sample1	0				
sample2	d_{21}	0			
sample3	d_{31}	d_{32}	0		
sample4	d_{41}	d_{42}	d_{43}	0	
sample5	d_{51}	d_{52}	d_{53}	d_{54}	0

階層的クラスター分析のプロセス



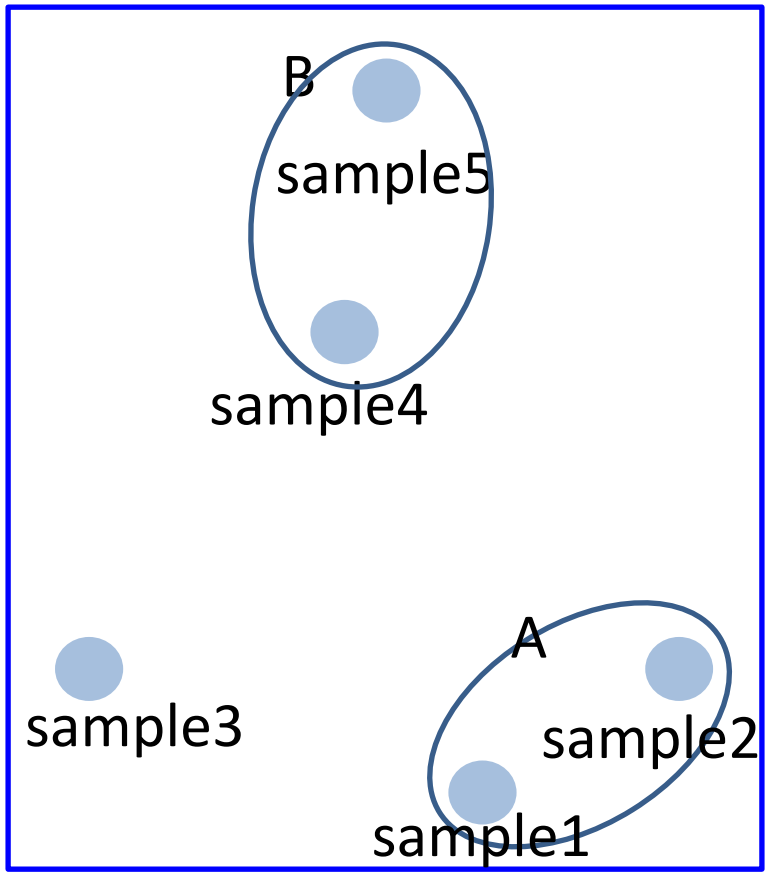
距離を更新

	A	sample3	sample4	sample5
A	0			
sample3	d_{3A}	0		
sample4	d_{4A}	d_{43}	0	
sample5	d_{5A}	d_{53}	d_{54}	0

クラスター数が1になるまで繰り返す

階層的クラスター分析のプロセス

最も近いサンプル(クラスター)を併合



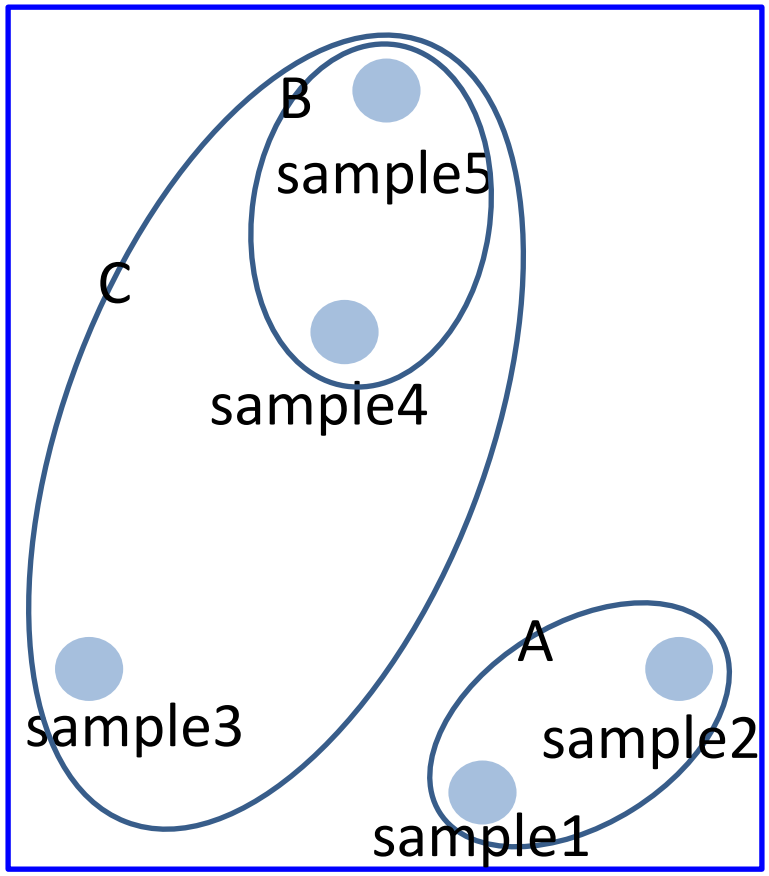
	A	sample3	sample4	sample5
A	0			
sample3	d_{3A}	0		
sample4	d_{4A}	d_{43}	0	
sample5	d_{5A}	d_{53}	d_{54}	0

距離を更新

	A	sample3	B
A	0		
sample3	d_{3A}	0	
B	d_{BA}	d_{B3}	0

階層的クラスター分析のプロセス

最も近いサンプル(クラスター)を併合



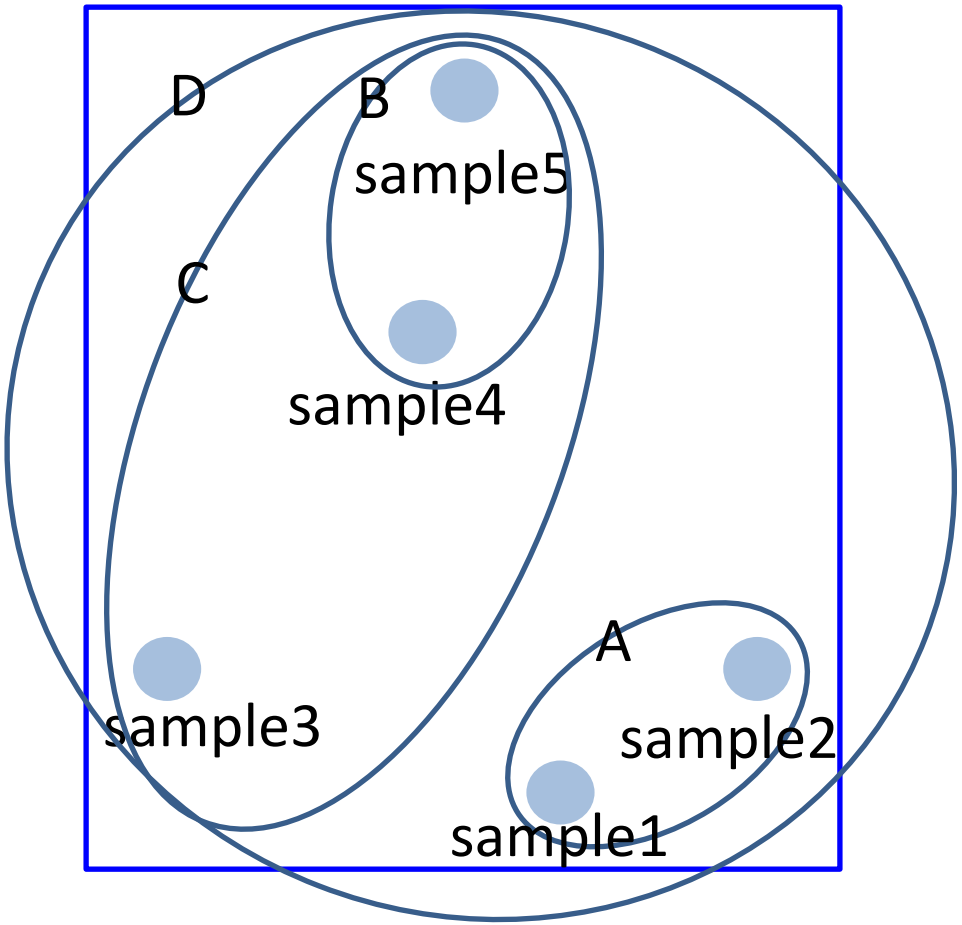
	A	sample3	B
A	0		
sample3	d_{3A}	0	
B	d_{BA}	d_{B3}	0

距離を更新

	A	C
A	0	
C	d_{BC}	0

階層的クラスター分析のプロセス

最も近いサンプル(クラスター)を併合



	A	C
A	0	
C	d_{BC}	0

距離を更新

	D
D	0

距離

- 距離の定義

(1) $d(X,X) = 0$

(2) $d(X,Y) \geq 0$

(3) $d(X,Y) = d(Y,X)$

(4) $d(X,Y) \leq d(Y,Z) + d(Z,Y)$

距離

- n個の分析対象(個体)をm個の項目に分けたデータの回答結果の一般表記は次のように示すことができる

$$X = (x_1, x_2, \cdots, \cdots, x_n)$$
$$Y = (y_1, y_2, \cdots, \cdots, y_n)$$

- ユークリッド距離

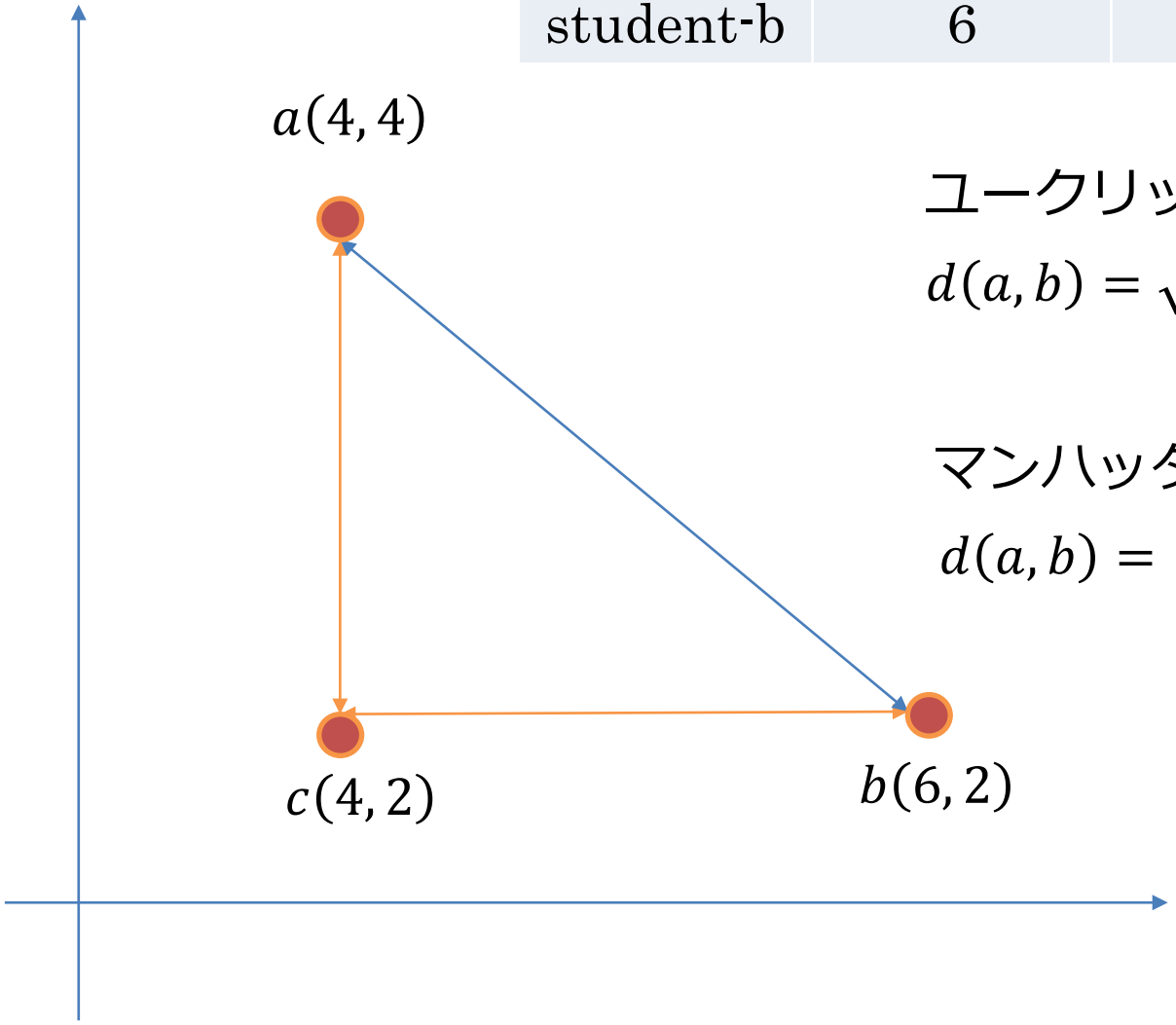
$$d(X,Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_m - y_m)^2} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

- マンハッタン距離

$$d(X,Y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_m - y_m| = \sum_{i=1}^{n} |x_i - y_i|$$

距離

	question1	question2
student-a	4	4
student-b	6	2



ユークリッド距離

$$d(a,b) = \sqrt{(4-6)^2 + (4-2)^2} = 2.828427$$

マンハッタン距離

$$d(a,b) = |4-6| + |4-2| = 4$$

金明哲.(2016).「定性的データ分析」.共立出版株式会社.

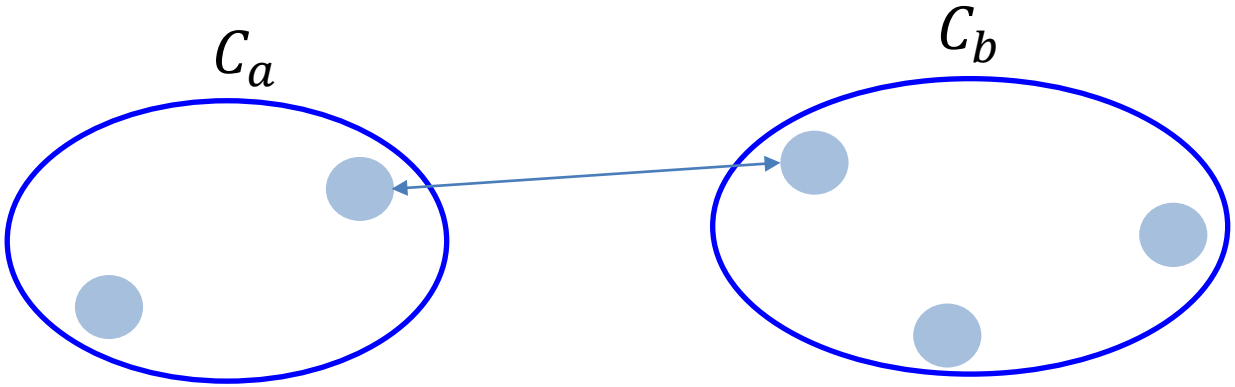
階層的クラスター分析

③距離行列からコーフェン行列を作る

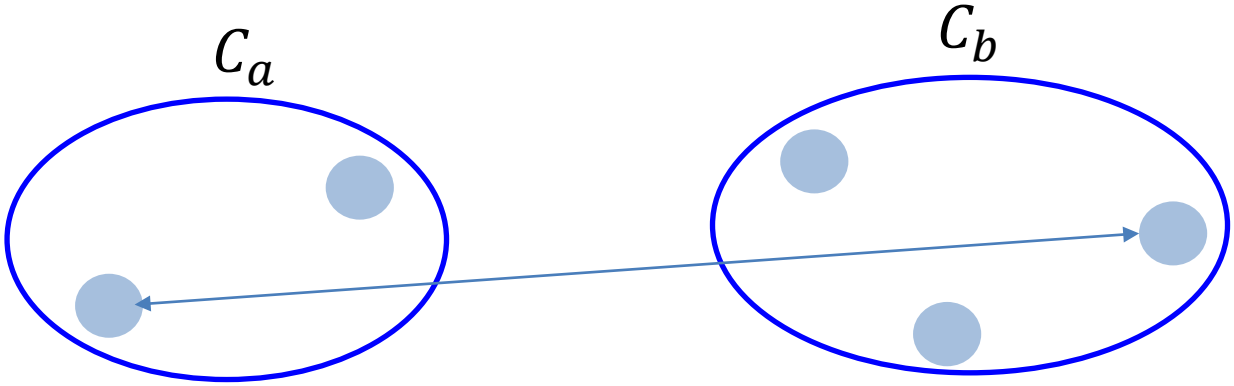
- クラスター間の距離に基づいてクラスタリングを求める
- クラスター間の距離行列をコーフェン(cophene)行列
- 樹形図はコーフェン行列に基づいて作成する
- 第一段階は最も距離が近い2つの個体間の距離をコーフェン行列とし、第一段階が終わった後にどのようなコーフェン行列を求めるかはクラスタリングの方法によって異なる

階層的クラスタリング法

- 最近隣法
  - 2つのクラスターの個体間の距離のなかで最も近い個体間の距離をこの2つのクラスター間の距離とする方法

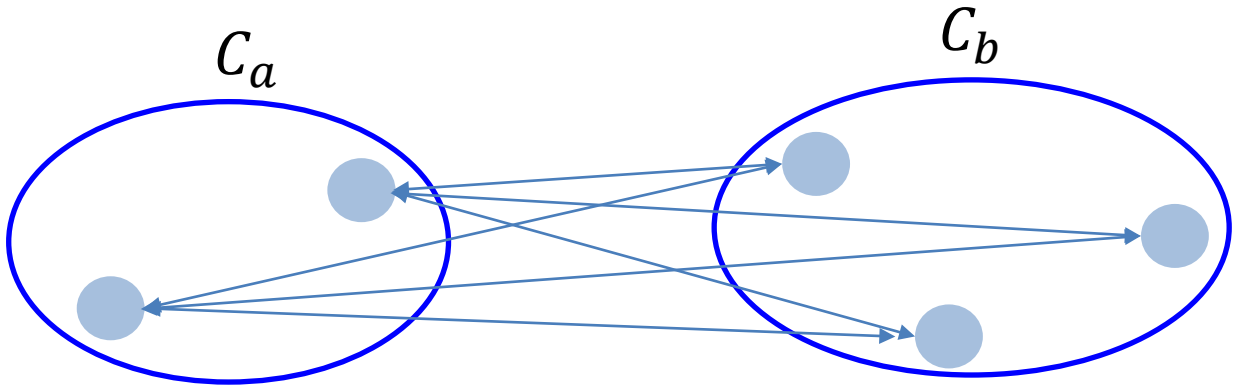


- 最遠隣法
  - 2つのクラスターの個体間の距離のなかで最も遠い個体間の距離をこの2つのクラスター間の距離とする方法

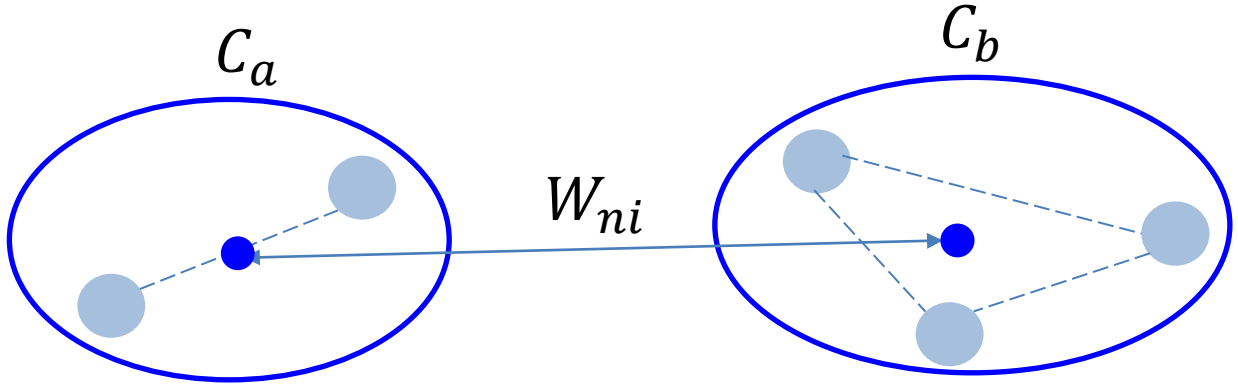


金明哲.(2016).「定性的データ分析」.共立出版株式会社.

- 群平均法
  - 2つのクラスターのそれぞれの個体間の距離の平均値を2つのクラスター間の距離とする

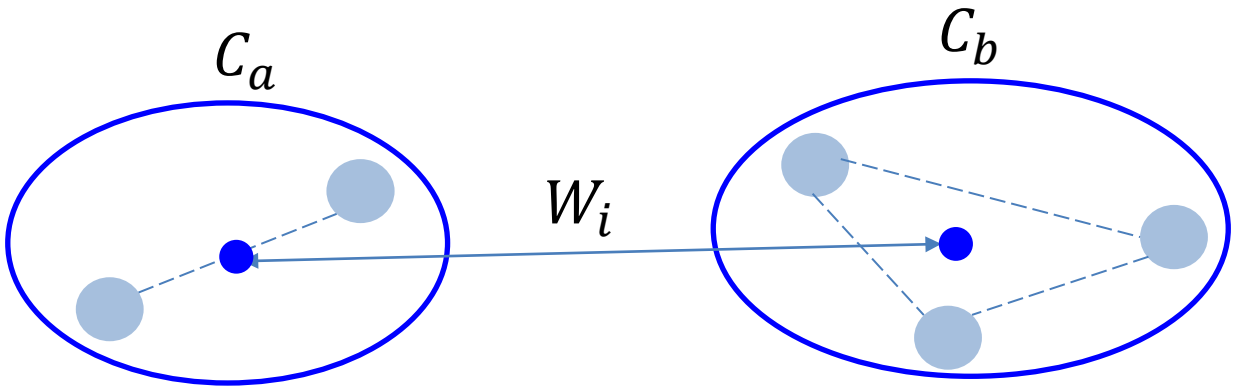


- 重心法
  - クラスターの重心間の距離をクラスター間の距離とする
  - 重心を求める際にはクラスターに含まれる個体数が反映されるように個体数を重みとして用いる



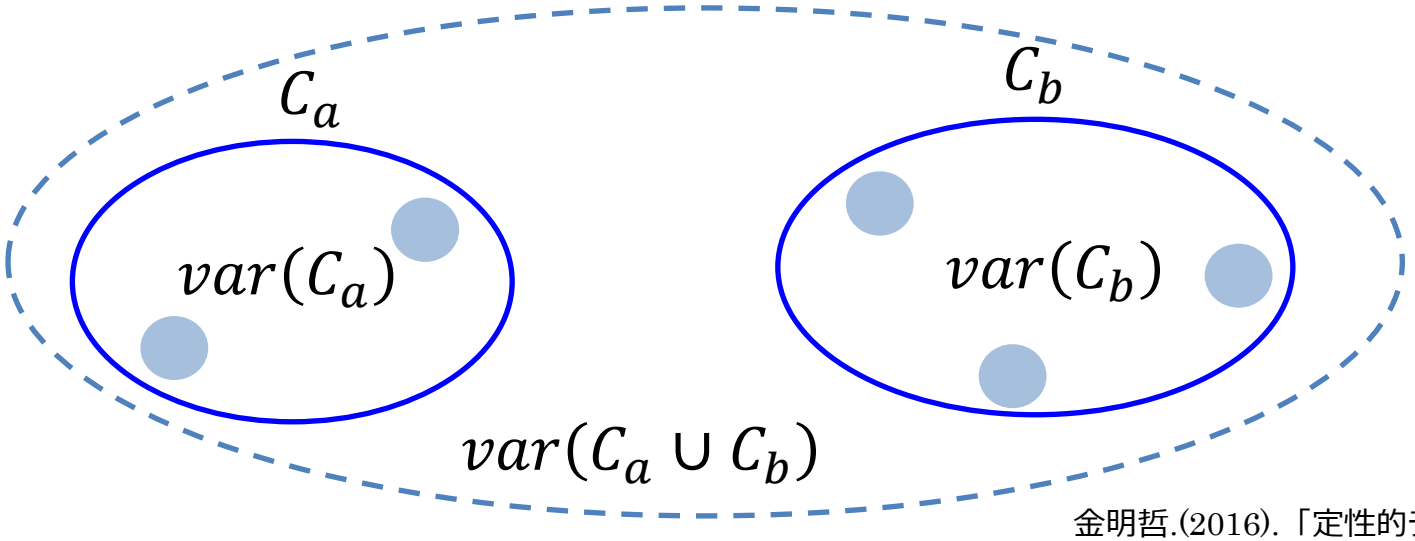
金明哲.(2016).「定性的データ分析」.共立出版株式会社.

- メディアン法
  - 2つのクラスターの重心間の重み付きの距離を求めるとき重みを等しくして求めた距離を2つのクラスター間の距離とする



- ウォード法
  - 郡内の分散と群間の分散の比を最大化する基準でクラスターを形成していく方法である

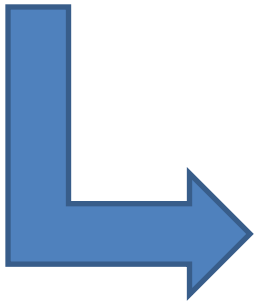
群間の分散 = 全体の分散 – 郡内の分散



金明哲.(2016).「定性的データ分析」.共立出版株式会社.

階層的クラスター分析のプロセス

	bikeA	bikeB
sample1	8	5
sample2	3	5
sample3	9	12
sample4	4	5
sample5	5	10



ユークリッド距離

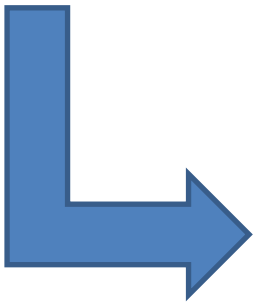
$$d(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

	sample1	sample2	sample3	sample4	sample5
sample1	0				
sample2	5	0			
sample3	7.07	9.22	0		
sample4	4	1	8.6	0	
sample5	5.83	5.39	4.47	5.1	0

階層的クラスター分析のプロセス

最も近いサンプルを併合

	sample1	sample2	sample3	sample4	sample5
sample1	0				
sample2	5	0			
sample3	7.07	9.22	0		
sample4	4	1	8.6	0	
sample5	5.83	5.39	4.47	5.1	0



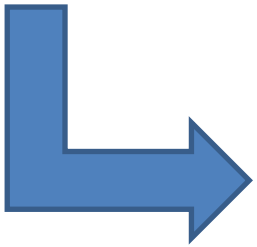
	sample1	sample2,4	sample3	sample5
sample1	0			
sample2,4	5, 4	1		
sample3	7.07	9.22, 8.6	0	
sample5	5.83	5.39, 5.1	4.47	0

- ウォード法

クラスターuとvを合併して新しいクラスターwをつくるとき、クラスターwと別のクラスターtとの間の非類似度はD_{wt}

n_u, n_v, n_tはそれぞれのクラスターu, v, tに含まれるデータ数

	sample1	sample2,4	sample3	sample5
sample1	0			
sample2,4	?	0		
sample3	7.07	?	0	
sample5	5.83	5.39, 5.1	?	0



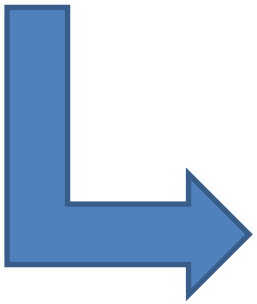
	sample1	sample2,4	sample3	sample5
sample1	0			
sample2,4	5.2	0		
sample3	7.07	10.28	0	
sample5	5.83	6.03	4.47	0

参考：https://bellcurve.jp/statistics/glossary/679.html

階層的クラスター分析のプロセス

最も近いサンプルを併合

	sample1	sample2,4	sample3	sample5
sample1	0			
sample2,4	5.2	0		
sample3	7.07	10.28	0	
sample5	5.83	6.03	4.47	0

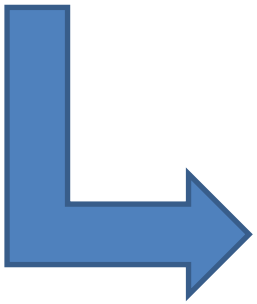


	sample1	sample2,4	sample3,5
sample1	0		
sample2,4	5.2	0	
sample3,5	7.02	9.83	0

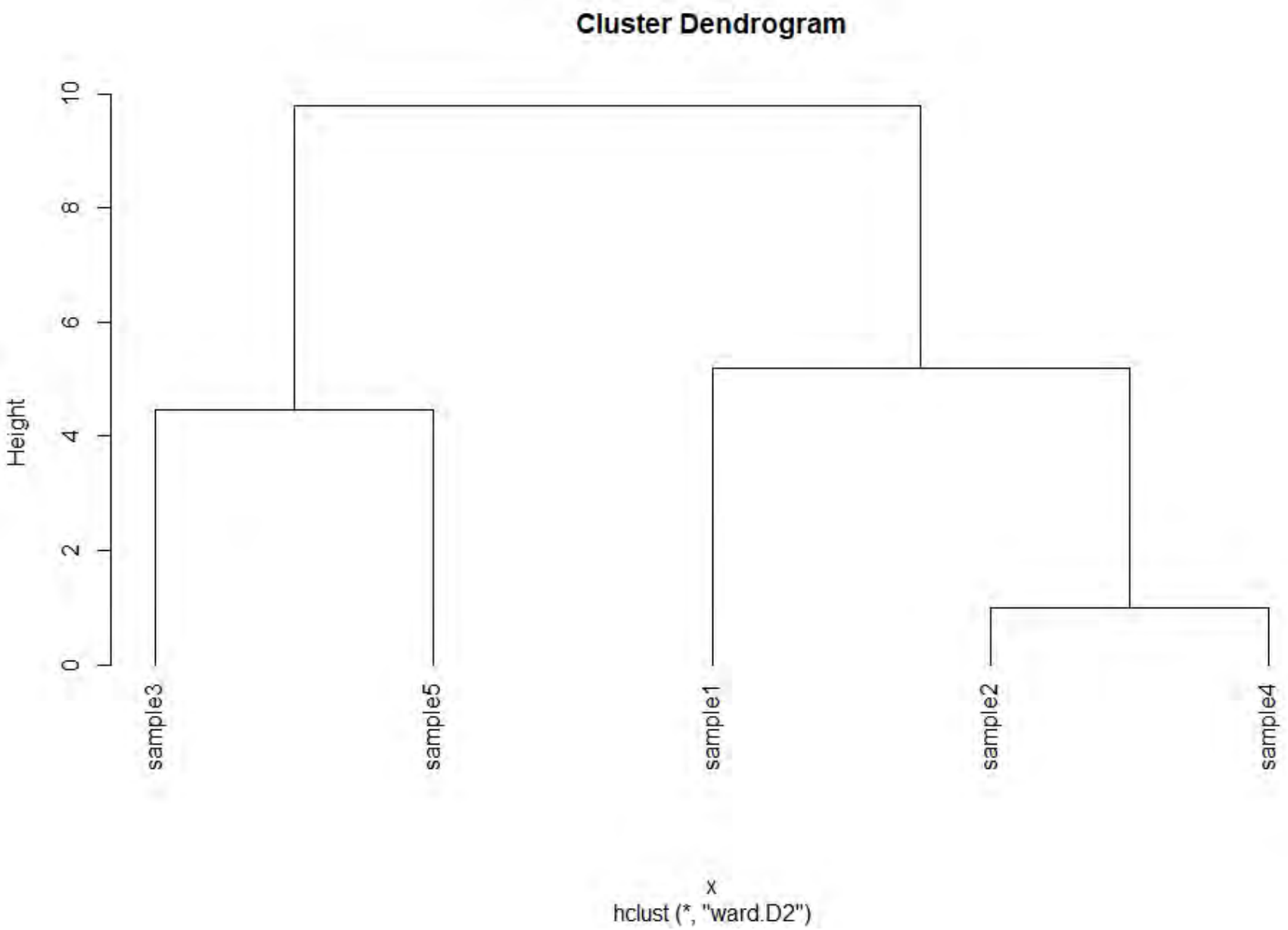
階層的クラスター分析のプロセス

最も近いサンプルを併合

	sample1	sample2,4	sample3,5
sample1	0		
sample2,4	5.2	0	
sample3,5	7.02	9.38	0



	sample1+sample2,4	sample3,5
sample1+sample2,4	0	
sample3,5	9.8	0


Cluster Dendrogram
Height
0
2
4
6
8
10
sample3
sample5
sample1
sample2
sample4
x
hclust (*, "ward.D2")

階層的クラスター分析

①データ行列を作成

```
> seiseki <- matrix(c(35, 40, 50, 81, 91, 80, 85, 90, 57, 70, 50, 45, 55, 41, 60,
45, 55, 60, 78, 85, 80, 75, 85, 55, 65, 87, 92, 95, 90, 85, 67, 46, 50, 89, 90),
7, 5, byrow = TRUE)
> colnames(seiseki) <- c("国語", "英語", "世界史", "数学", "生物")
> rownames(seiseki) <- c("芥川", "直木", "夏目", "太宰", "川端", "志賀",
"村上")
> seiseki
     国語 英語 世界史 数学 生物
芥川   35   40    50   81   91
直木   80   85    90   57   70
夏目   50   45    55   41   60
太宰   45   55    60   78   85
川端   80   75    85   55   65
志賀   87   92    95   90   85
村上   67   46    50   89   90
```

金城俊哉.(2017).「R統計解析パーフェクトマスター」.株式会社秀和システム.

階層的クラスター分析

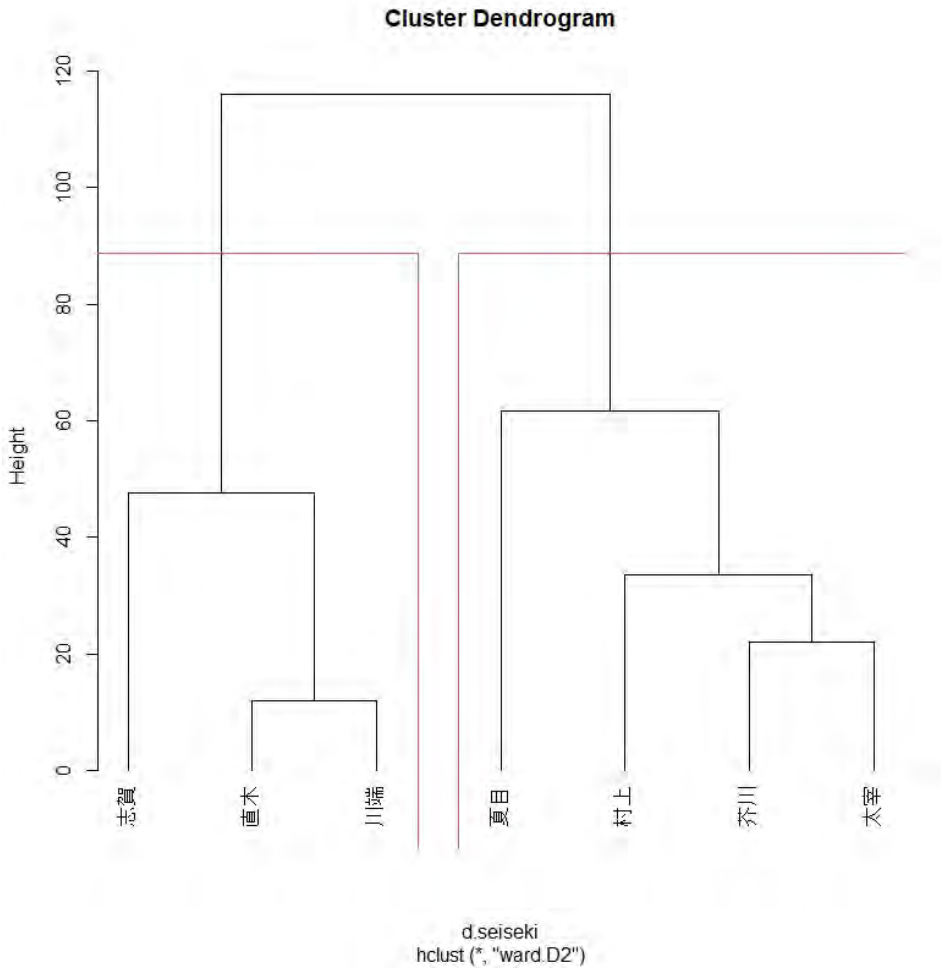
②距離行列の作成

```
> d.seiseki <- round(dist(seiseki, method = "euclidean"))
> d.seiseki
    芥川 直木 夏目 太宰 川端 志賀
直木   82
夏目   53   64
太宰   22   61   46
川端   76   12   54   56
志賀   87   38   91   67   45
村上   34   69   59   28   63   68
```

金城俊哉.(2017).「R統計解析パーフェクトマスター」.株式会社秀和システム.

階層的クラスター分析

```
> c.seiseki <- hclust(d.seiseki, method = "ward.D2")
> plot(c.seiseki, hang = -1)
> rect.hclust(c.seiseki, k = 2)
```



金城俊哉.(2017).「R統計解析パーフェクトマスター」.株式会社秀和システム.

階層的クラスター分析

- グループ分けしたクラスターの番号
- 全体を2つのクラスターを分けるとした場合

```
> cutree(c.seiseki, k = 2)
芥川 直木 夏目 太宰 川端 志賀 村上
  1   2   1   1   2   2   1
```

- 結果の内容リストを確認

```
> summary(c.seiseki)
 Length Class  Mode
merge       12     -none- numeric
height       6     -none- numeric
order        7     -none- numeric
labels       7     -none- character
method       1     -none- character
call         3     -none- call
dist.method  1     -none- character
```

金城俊哉.(2017).「R統計解析パーフェクトマスター」.株式会社秀和システム.

クラスタリングの過程

- マイナス符号をついているのが個体の番号
- ついていないのがクラスターの番号

```
> c.seiseki$merge
[,1] [,2]
[1,]   -2   -5
[2,]   -1   -4
[3,]   -7    2
[4,]   -6    1
[5,]   -3    3
[6,]    4    5
```

金城俊哉.(2017).「R統計解析パーフェクトマスター」.株式会社秀和システム.

クラスタリングの過程

- クラスターの枝の長さ
- mergeと対応している

```
> c.seiseki$height
[1]  12.00000  22.00000  33.64521  47.58851  61.63603 116.03222
```

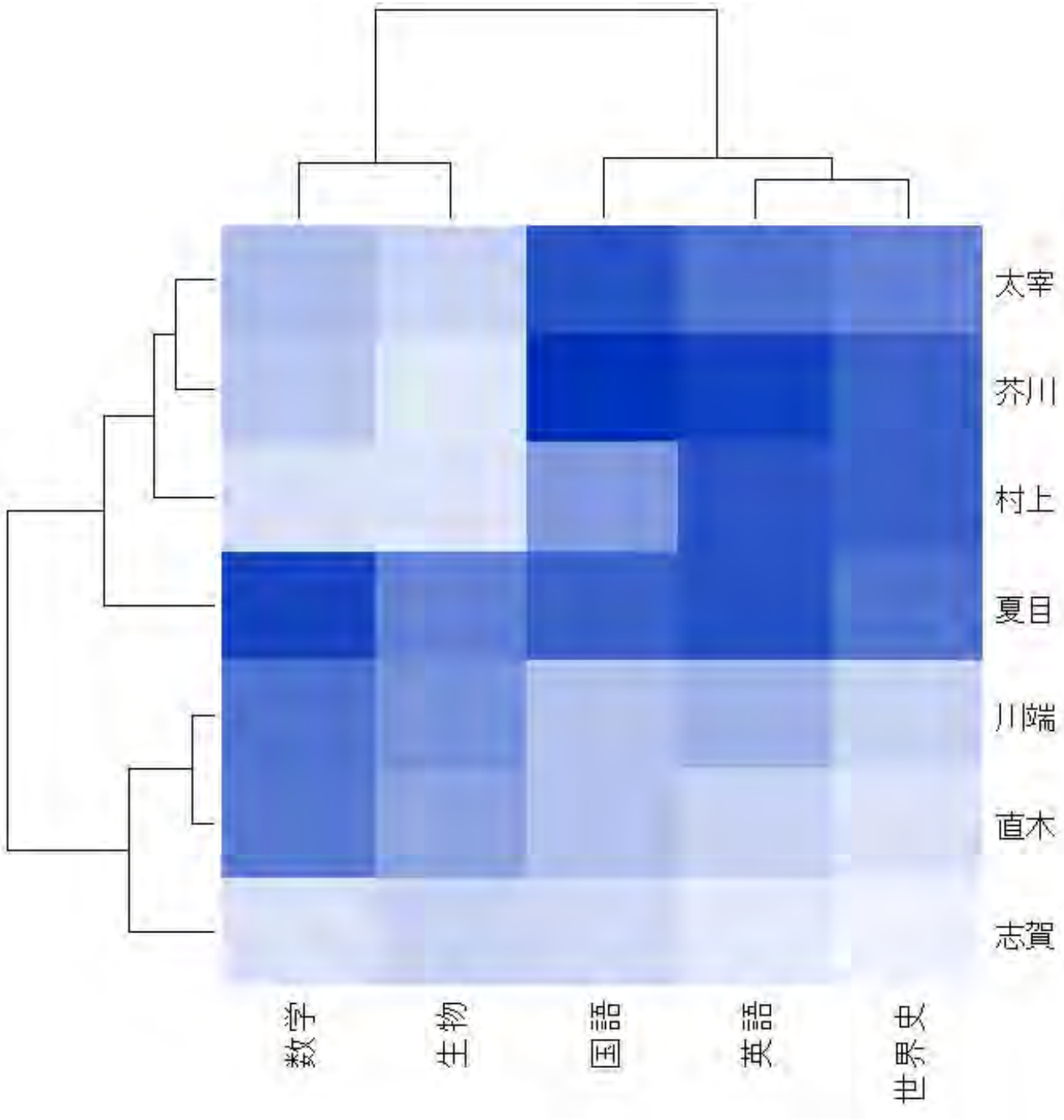
- コーフェン行列

```
> cophenetic(c.seiseki)
        芥川     直木     夏目     太宰     川端     志賀
直木 116.03222
夏目  61.63603 116.03222
太宰  22.00000 116.03222  61.63603
川端 116.03222  12.00000 116.03222 116.03222
志賀 116.03222  47.58851 116.03222 116.03222  47.58851
村上  33.64521 116.03222  61.63603  33.64521 116.03222 116.03222
```

金城俊哉.(2017).「R統計解析パーフェクトマスター」.株式会社秀和システム.

ヒートマップ

- 階層的クラスター分析のヒートマップ
  - 個体の樹形図と変数の樹形図を同時に図示し、色彩を用いて値の大小を表示
  - 横が個体の樹形図
  - 縦が変数の樹形図



- 色が濃いほど値が小さい
- 色が薄いほど値が大きい

```
> palette <-
colorRampPalette(c('#0033BB','#f0f3ff'
))(256)
> c.row <- hclust(dist(seiseki, method =
"euclidean"), method = "ward.D2")
> c.col <- hclust(dist(t(seiseki), method
= "euclidean"), method = "ward.D2")
> heatmap(as.matrix(seiseki),
                Colv = as.dendrogram(c.col),
Rowv = as.dendrogram(c.row),
                scale = "none", col =
palette)
```

非階層的クラスター分析

- 階層的クラスター分析
  - 個体数が多いと計算量が膨大になり、大量のデータ解析にはむいていない
- 大規模のデータセットのクラスター分析には非階層的クラスター
- 非階層的クラスター
  - k平均法 $(k-means)$

非階層的クラスター分析のプロセス

① k個のクラスター中心(seeds)の初期値を適当に与える

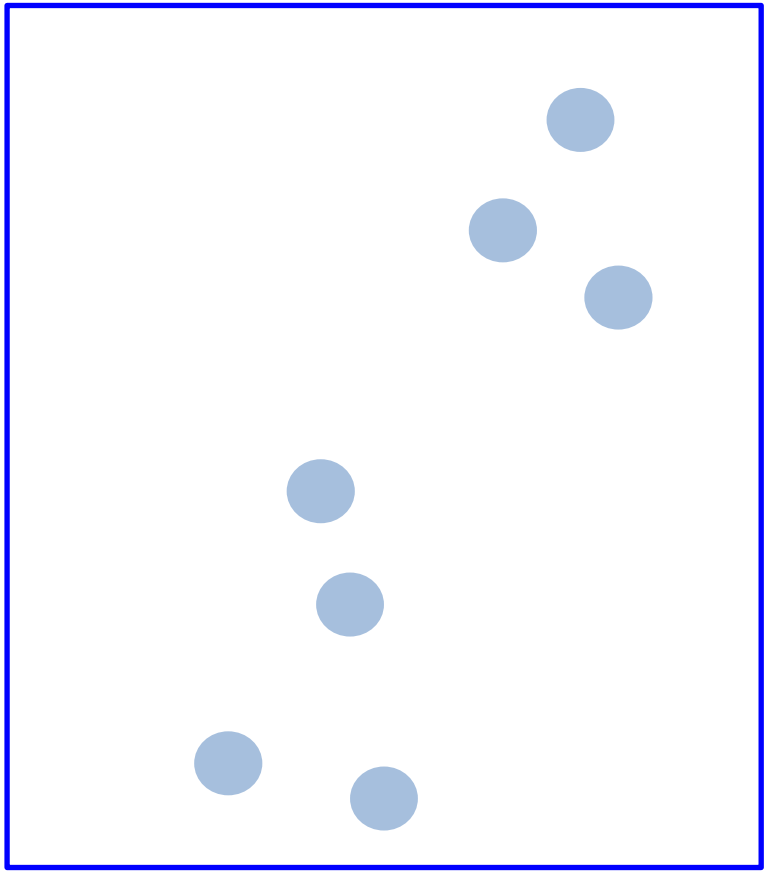
②各データをk個のクラスター中心との距離を求め、最も近いクラスターに分類する

③形成されたクラスターの中心を求める

④クラスターの中心が変化しない時点までステップ2, 3を繰り返す

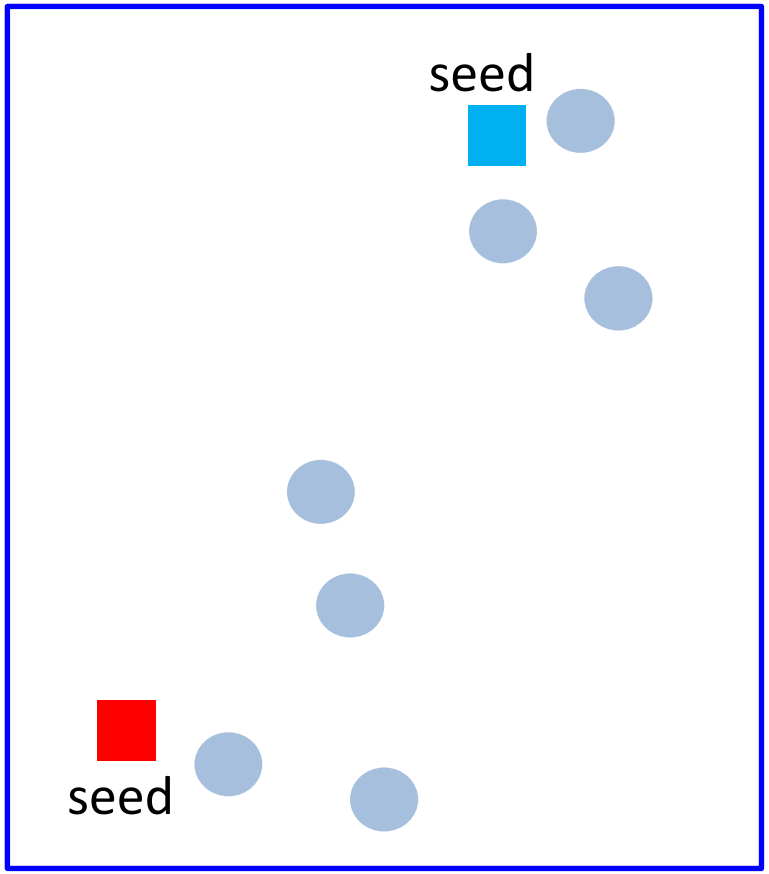
非階層的クラスター分析のプロセス

クラスター数2の場合



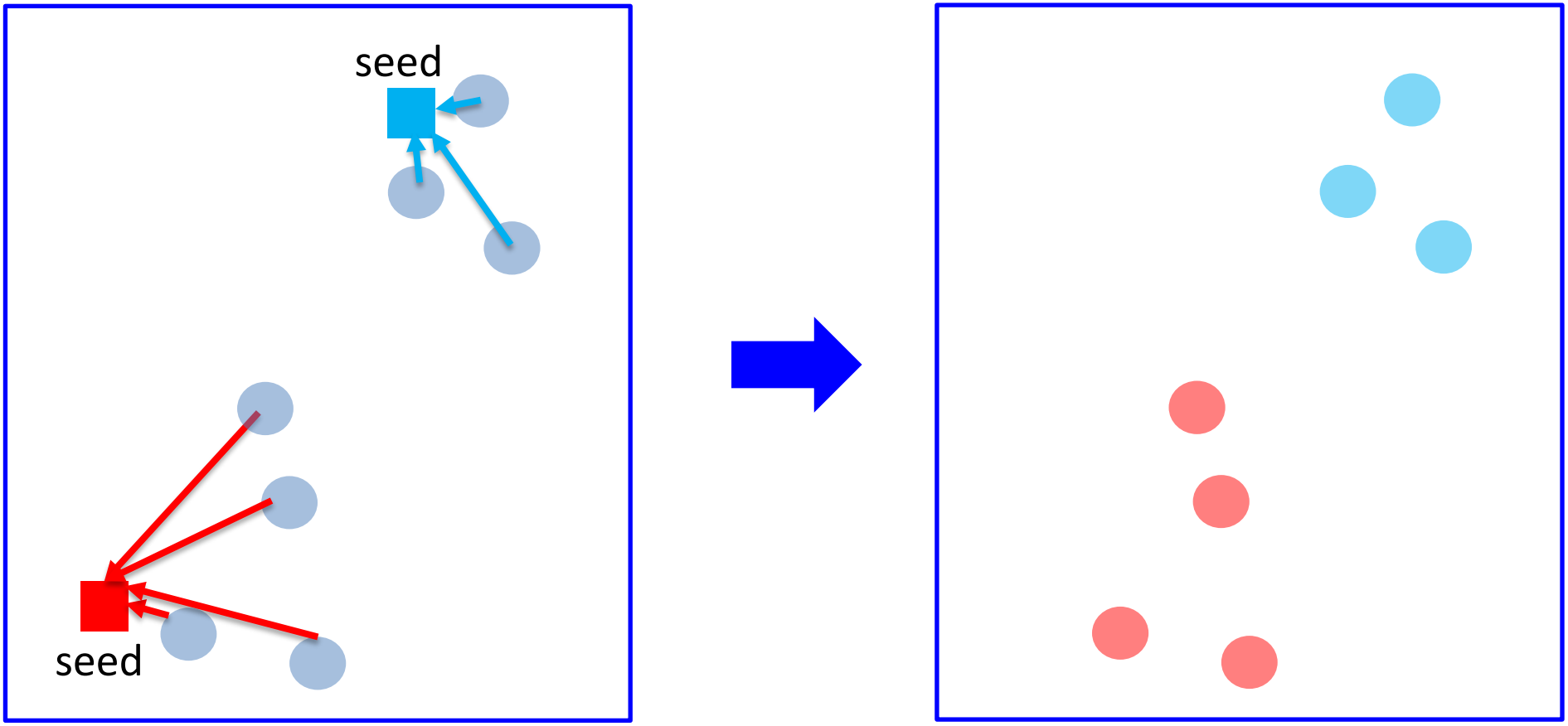
● : サンプル

① k個のクラスター中心(seeds)の初期値を適当に与える



非階層的クラスター分析のプロセス

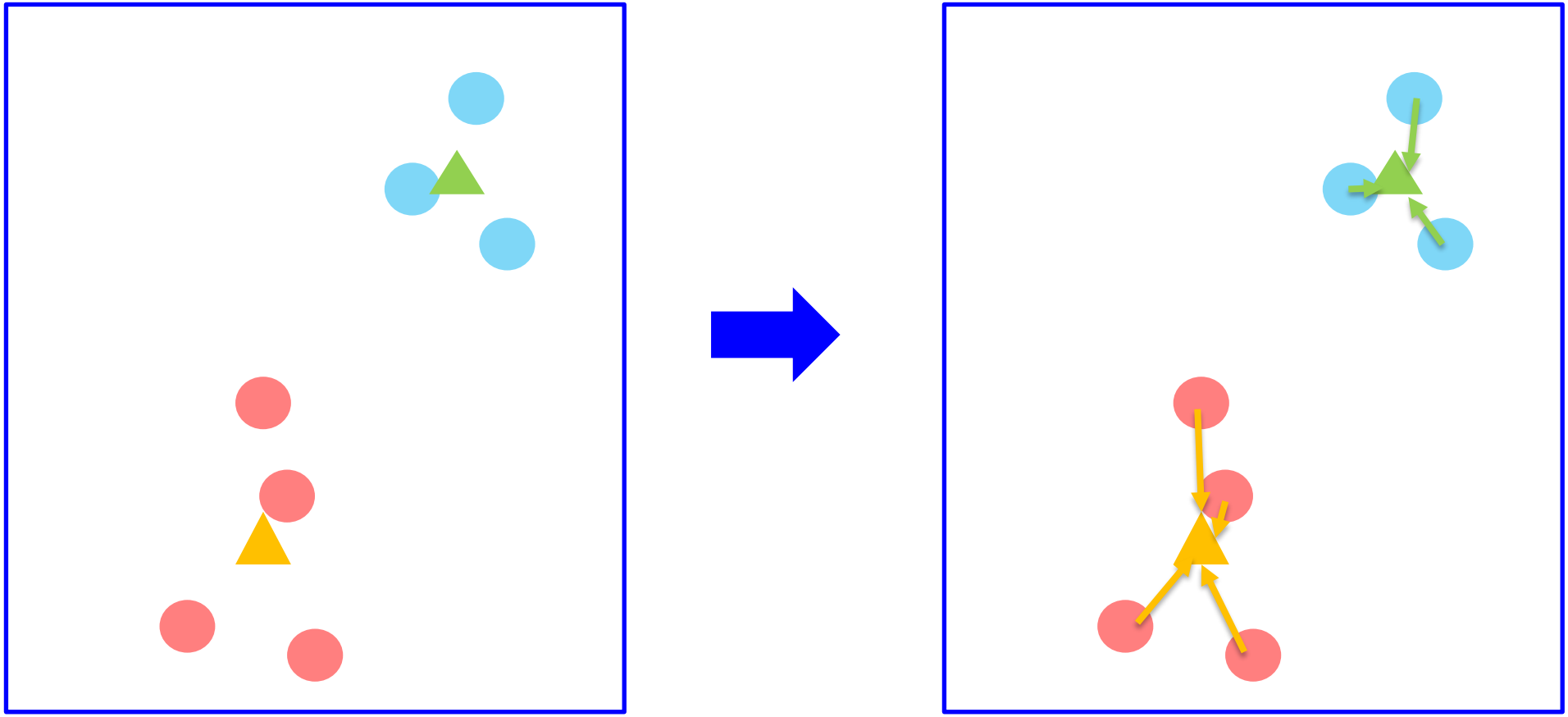
②各データをk個のクラスター中心との距離を求め、最も近いクラスターに分類する



非階層的クラスター分析のプロセス

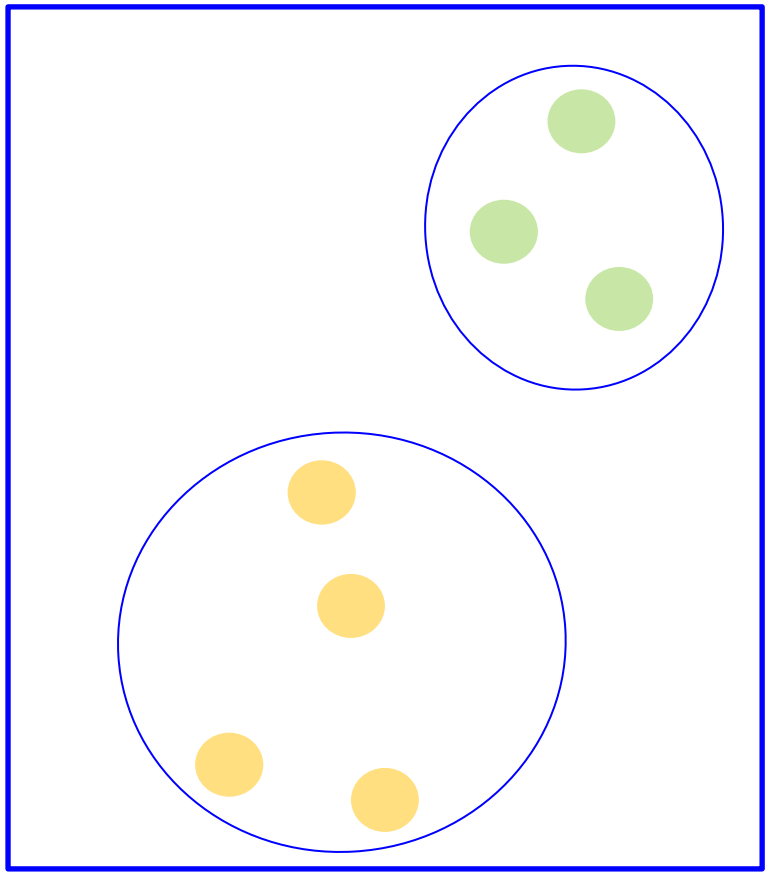
③形成されたクラスターの中心を求める

④クラスターの中心が変化しない時点までステップ2, 3を繰り返す



非階層的クラスター分析のプロセス

収束・分類の完成



$k-means$

- wineデータの上位6行までのデータを確認

```
> head(wine_p)
  Three.cultivars Alcohol Malic.acid  Ash Alcalinity.of.ash Magnesium
1               1   14.23       1.71 2.43              15.6       127
2               1   13.20       1.78 2.14              11.2       100
3               1   13.16       2.36 2.67              18.6       101
4               1   14.37       1.95 2.50              16.8       113
5               1   13.24       2.59 2.87              21.0       118
6               1   14.20       1.76 2.45              15.2       112
  Total.phenols Flavanoids Nonflavanoid.phenols Proanthocyanins
1          2.80       3.06                 0.28            2.29
2          2.65       2.76                 0.26            1.28
3          2.80       3.24                 0.30            2.81
4          3.85       3.49                 0.24            2.18
5          2.80       2.69                 0.39            1.82
6          3.27       3.39                 0.34            1.97
```

一部省略

http://archive.ics.uci.edu/ml/datasets/Wine

$k-means$

- 一列目は3種類の品種に関する情報なので除く

```
> wine_t <- wine_p[,-1]
```

- クラスタ数を指定
  - 3種類の品種なので3を指定してみる

```
> set.seed(12345)
> wine_c3 <- kmeans(x = wine_t, centers = 3, algorithm = "Hartigan-
Wong")
> wine_c3
K-means clustering with 3 clusters of sizes 62, 47, 69

Cluster means:
   Alcohol Malic.acid      Ash Alcalinity.of.ash Magnesium
1 12.92984   2.504032 2.408065         19.89032 103.59677
2 13.80447   1.883404 2.426170         17.02340 105.51064
3 12.51667   2.494203 2.288551         20.82319  92.34783
```

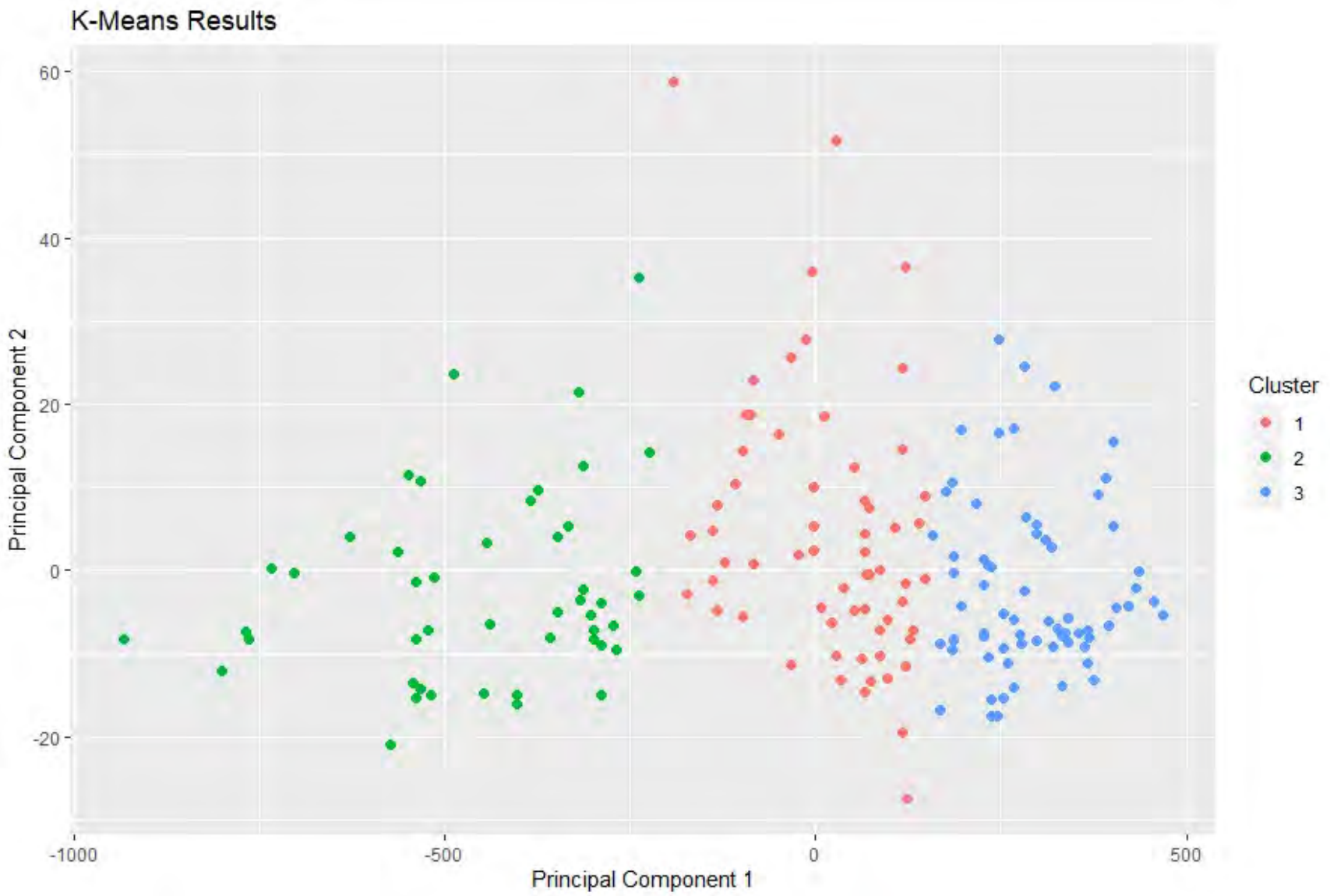
J. A. Hartigan and M. A. Wong. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society*

```
 Total.phenols Flavanoids Nonflavanoid.phenols Proanthocyanins
1      2.111129   1.584032           0.3883871        1.503387
2      2.867234   3.014255           0.2853191        1.910426
3      2.070725   1.758406           0.3901449        1.451884
  Color.intensity       Hue OD280.OD315.of.diluted.wines   Proline
1        5.650323 0.8839677                     2.365484  728.3387
2        5.702553 1.0782979                     3.114043 1195.1489
3        4.086957 0.9411594                     2.490725  458.2319

Clustering vector:
  [1] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 1 1 2 2 1 2 2 2
 [33] 2 2 2 1 1 2 2 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 1 3 1 3
 [65] 3 1 3 3 1 1 1 3 3 2 1 3 3 3 1 3 3 1 1 3 3 3 3 1 1 3 3 3 3 3 1
 [97] 1 3 1 3 1 3 3 3 1 3 3 3 3 1 3 3 1 3 3 3 3 3 3 1 3 3 3 3 3 3 3
[129] 3 3 1 3 3 1 1 1 1 3 3 3 1 1 3 3 1 1 3 1 1 3 3 3 3 1 1 1 3 1 1 1
[161] 3 1 3 1 1 3 1 1 1 1 3 3 1 1 1 1 1 3

Within cluster sum of squares by cluster:
[1]  566572.5 1360950.5  443166.7
 (between_SS / total_SS =  86.5 %)
```

```
Available components:

[1] "cluster"      "centers"      "totss"        "withinss"
[5] "tot.withinss" "betweenss"    "size"         "iter"
[9] "ifault"
```

- $k-means$クラスタリングを可視化することはデータの高次元性のため難しい
- 多次元データを2次元に投影

```
> library(useful)
> plot(wine_c3, data = wine_t)
```



Jared P. Lander著.(2015).「みんなのR」.株式会社マイナビ.

$k-means$

- 複数の初期設定を試行し、最適なものを表示
- nstart = 25 とすると、25個の初期設定を作成

```
> set.seed(12345)
> wine_c3n15 <- kmeans(wine_t, centers = 3, nstart = 25)
> wine_c3$size
[1] 62 47 69
> wine_c3n25$size
[1] 62 47 69
```

- ワインの品種と比較してみる

```
> table(wine_p$Three.cultivars, wine_c3n25$cluster)

   1  2  3
1 13 46  0
2 20  1 50
3 29  0 19
```

Jared P. Lander著.(2015).「みんなのR」.株式会社マイナビ.

$k-means$

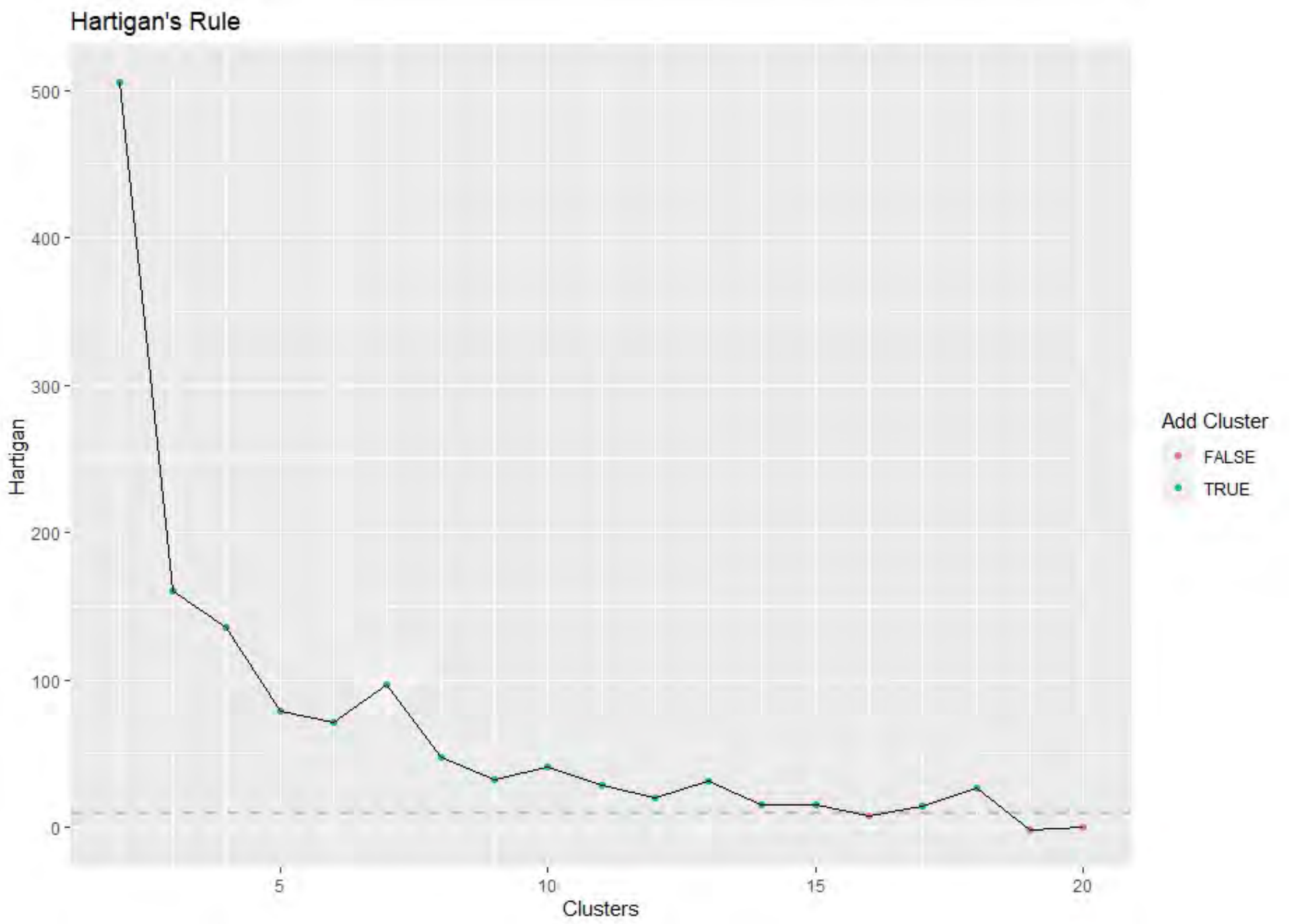
- クラス数の決め方：ハーティガンルール
- k個のクラスタ内平方和と$k+1$個の場合のクラスタ内平方和の比を比較
- この数字が10を超える場合 $k+1$クラスタを使うことに意味がある

```
> wine_no_beset <- FitKMeans(wine_t, max.clusters = 20, nstart = 25,
seed = 12345)
> wine_no_beset
 Clusters    Hartigan AddCluster
1        2 505.4293097       TRUE
2        3 160.4113307       TRUE
3        4 135.7072281       TRUE
4        5  78.4452892       TRUE
5        6  71.4897095       TRUE
6        7  96.6572235       TRUE
7        8  47.5215253       TRUE
8        9  32.5567384       TRUE
```

Jared P. Lander著.(2015).「みんなのR」.株式会社マイナビ.

$k - means$

9	10	40.9079542	TRUE
10	11	28.3126328	TRUE
11	12	20.1309251	TRUE
12	13	31.8416624	TRUE
13	14	15.0897543	TRUE
14	15	15.0566476	TRUE
15	16	8.1708225	FALSE
16	17	14.2827085	TRUE
17	18	27.1061619	TRUE
18	19	-1.8509234	FALSE
19	20	0.2994339	FALSE

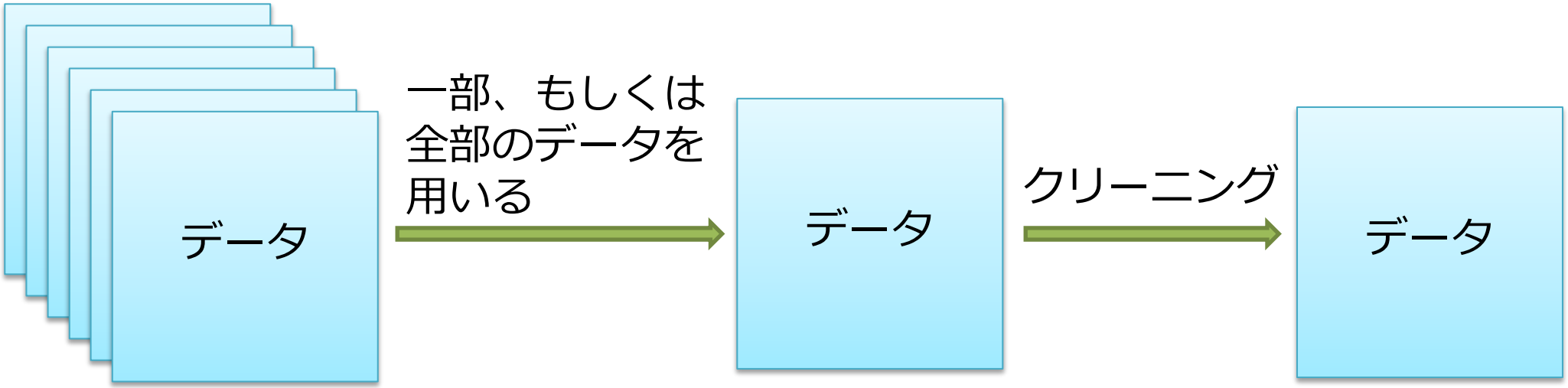


Jared P. Lander著.(2015).「みんなのR」.株式会社マイナビ.

テキストを対象とした研究

- 文章
  - 何らかの文字列が一定の文法規則に基づいている文の集合体
- 文章型テキスト
  - 日記、小説、新聞記事、メール、報告書、Twitterなど
- テキストマイニング
  - 文字や記号列が有機的に結合されている集合体から何らかのパターンを集計し、有用な情報や知識を見つけ出す方法と過程の総称
  - テキストから広義のパターンを機械的に見つけ出す学際的分野
    - データマイニング、統計学、機械学習
    - 計量文献学、計量言語学、情報抽出、自然言語処理

分析のプロセス



分析のプロセス

テキスト処理

辞書

形態素解析

構文解析

意味解析

JUMAN,Cha Sen, Mecab, Cabochaなど

量的データに変換

N-gram

長さの分布

パターン

統計解析・可視化

3人の作家

「一九二〇年の夏のある午後だった。北米オハイオ州の大都市のクリイヴランドの町を、一仕事すました軽い心で私は電車を駆っていた。そのころ私は聖クレア街の一建築事務所に、夏休みを利用して働いていたのだ。午後の小憩に安い昼寄席でも見に行く途中だったと覚えている」

（谷譲次『テキサス無宿』河出書房新社、1969）

「さっきの雷鳴で、雨は、カラっと霽れた。

従来の水たまりに、星がうつっている。いつもなら、爪紅さした品川女郎衆の、素あしなまめかしいよい闇だけれど。

今宵は。

問屋場の油障子に、ぱっとあかるく灯がはえて、右往左往する人かげ」

（林不忘『丹下左膳』河出書房新社、1970）

村上征勝.(1994).「真贋の科学」.朝倉書店.

3人の作家

「眼が外側へ向いているために、人間は自分よりさきにまわりの物ごとに気がつくんだが、これは個体としての赤ん坊もそうだし、進化史的に観た人類の先祖もその認識の過程においてまったくおなじであったろうと思われる。つまり人間の智識は、はじめて網膜にうつった外界の物質から受ける驚異に出発している」

（牧逸馬「第七の天」〔『七時〇三分』〕河出書房新社、1970）

村上征勝.(1994).「真贋の科学」.朝倉書店.

- 谷譲次、林不忘、牧逸馬はペンネームであり、書き手は同じ人物
- 長谷川海太郎（1900～1935）
  - 昭和初期の流行作家
  - 一人で3つのペンネームを用いた
- 谷譲次のペンネーム
  - 6年間の米国留学中の生活を素材に『テキサス無宿』『めりけんじゃっぷ商売往来』などのめりけん物とよばれる小説

村上征勝.(1994).「真贋の科学」.朝倉書店.

- 林不忘のペンネーム
  - 『丹下座膳』『釘抜藤吉捕物覚書』をはじめとする時代物の小説
- 牧逸馬のペンネーム
  - 『第七の天』『七時〇三分』などの探偵小説や『海のない港』『地上の星座』などの風俗小説

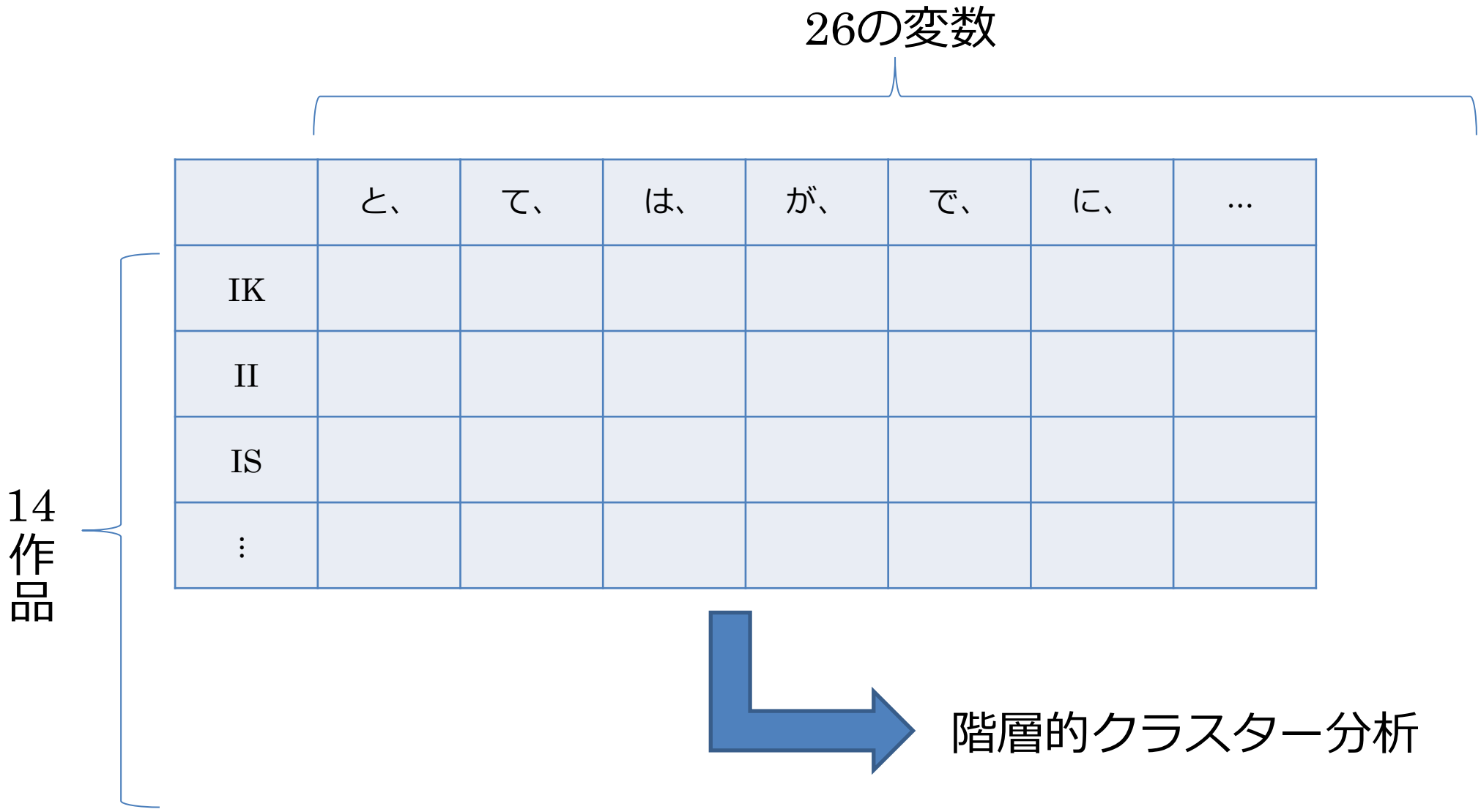
村上征勝.(1994).「真贋の科学」.朝倉書店.

作品の記号	作家	作品名	発表年
IK	井上靖	結婚記念日	1951
II	井上靖	石庭	1950
IS	井上靖	死と恋と波	1950
NS	中島敦	山月記	1942
NM	中島敦	名人伝	1942
ND	中島敦	弟子	1943
NR	中島敦	李陵	1943
HH	林不忘（長谷川海太郎）	早耳三次捕物聞書	1928
HT	林不忘（長谷川海太郎）	丹下左膳	1933
MD	牧逸馬（長谷川海太郎）	第七の天	1928
MS	牧逸馬（長谷川海太郎）	神変美容術師	1933
TH	谷譲次（長谷川海太郎）	白夜幻想曲	1928
TK	谷譲次（長谷川海太郎）	空気になった男	1934
TS	谷譲次（長谷川海太郎）	私刑物語	1934

村上征勝.(1994).「真贋の科学」.朝倉書店.

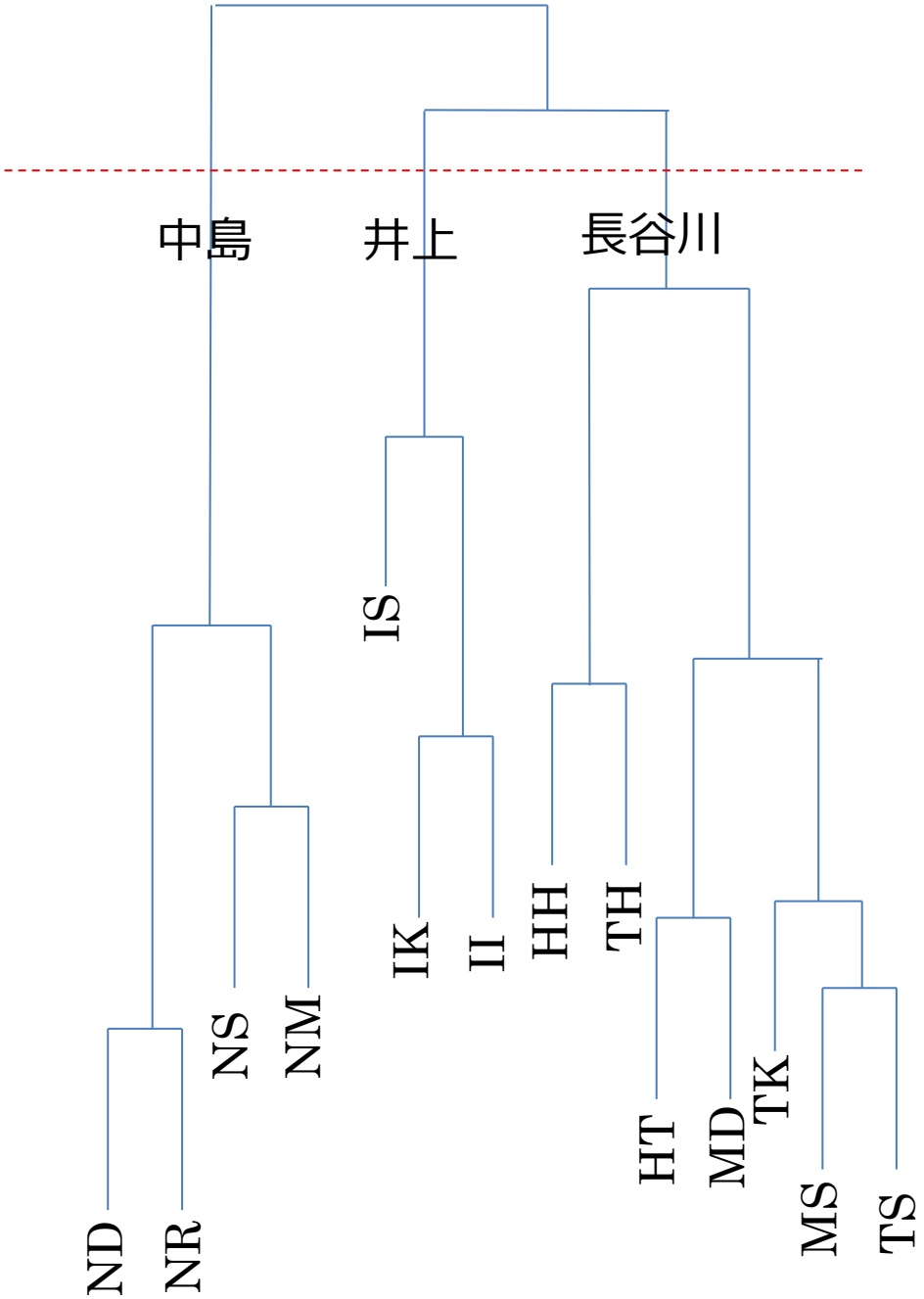
3人の作家

- 読点の前の文字の頻度に関する26の変数

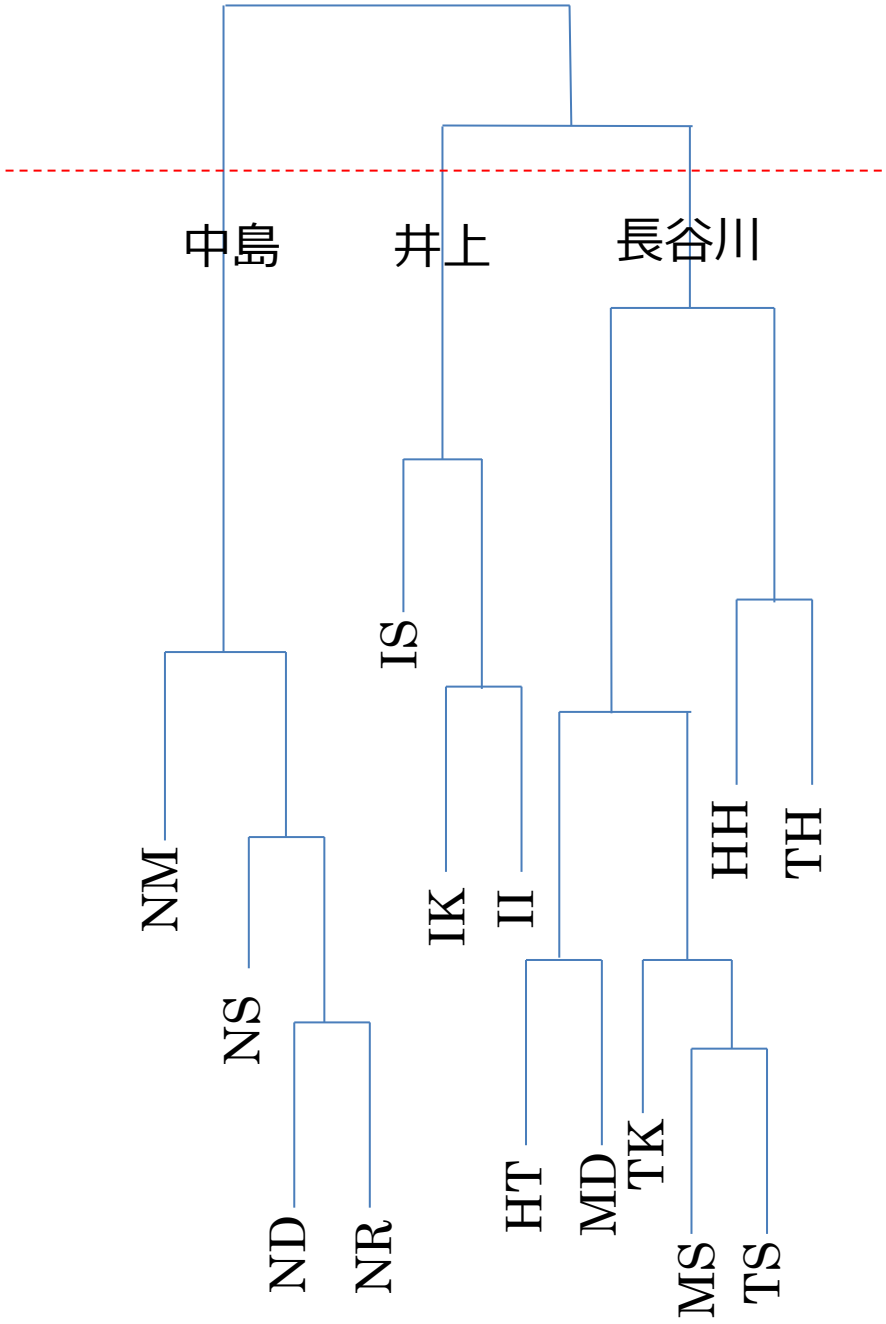


村上征勝.(1994).「真贋の科学」.朝倉書店.

マンハッタン距離, 最長距離法

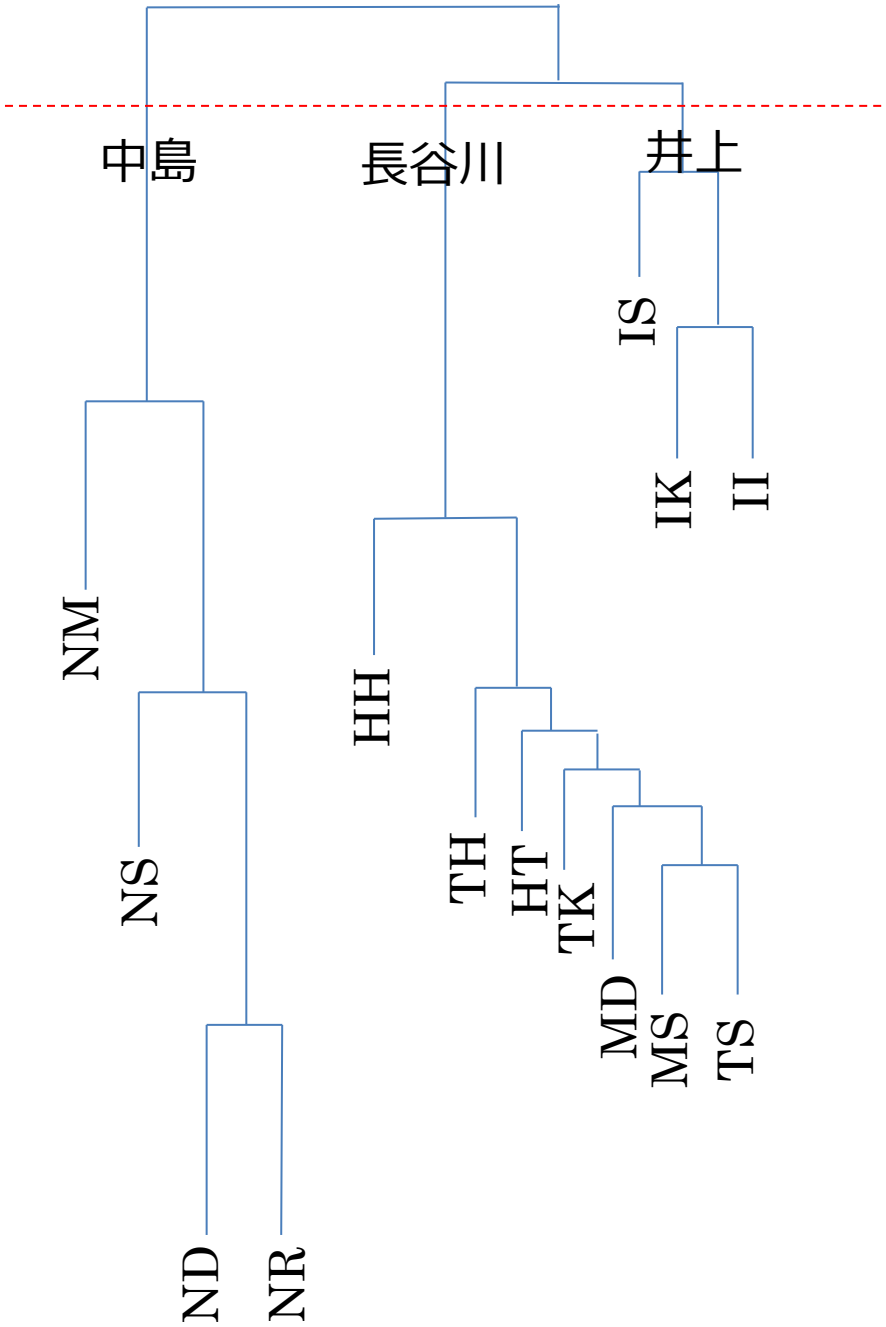


マンハッタン距離, 群平均法



村上征勝.(1994).「真贋の科学」.朝倉書店.

マンハッタン距離, 最短距離法



村上征勝.(1994).「真贋の科学」.朝倉書店.

3人の作家

- 読点の直前の1文字の頻度によって作品が作家別にまとまっている
- 中島、井上、長谷川の作品はそれぞれクラスターを作る
- しかし、林・牧・谷のペンネームごとにはわかれていない
  - 作者ごとに読点の付け方は変わっていない
- これだけを調べれば書き手が特定できるような指紋やDNAのようなものは現状では見つかっていないが、現代日本文においては、読点の付け方に有効な書き手のクセと考えられる

村上征勝.(1994).「真贋の科学」.朝倉書店.

ビジネス活用（意思決定）

人工知能技術のビジネス活用

- 効率性を議論するだけでなく、以下の観点を含めたTrustable AIの実現が必要
  - Valid AI：頑健、検証可能
  - Responsible AI：公平、倫理的
  - Privacy-preserving AI：暗号化、匿名化
  - Explainable AI：人が解釈できる、機械が解釈できる

- 説明可能なAI：検索、学習、計画、推論の性能を高い水準に維持しながら、説明可能なモデルを生成する一連の技術

説明が必要とされる応用領域

- 金融分野
  - ローンの可否判断や保険金額の見積りなど、判断根拠を顧客に説明することが求められる
- ヘルスケア
  - 病気の診断や治療法の選択は、医師が最終決定する
  - その根拠が医師に理解できるものでなければ使われない
- 基幹システム
  - 交通や情報ネットワークは停止させてはいけない
  - しかし、学習データの偏りによる不適切な意思決定が生じる恐れ

- ブラックボックスAIはビジネス応用にとってリスク
  - どのような学習データを用いるか、恣意性が存在
  - 機械学習アルゴリズムは完成形ではなく、限界が存在

総務省AI利活用原則案

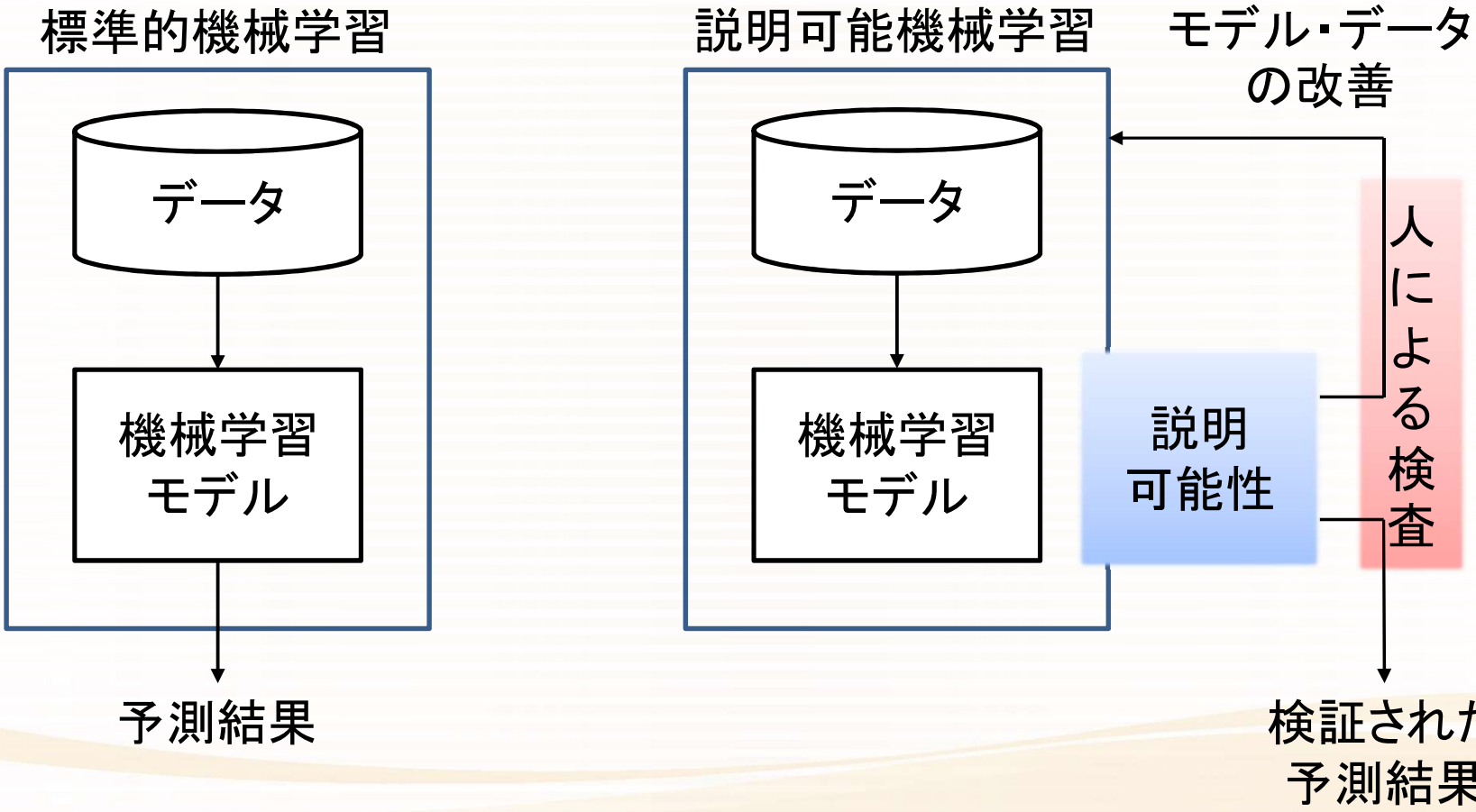
- 透明性の原則：AIサービスプロバイダ及びビジネス利用者は、AIシステム又はAIサービスの入出力等の検証可能性及び判断結果の説明可能性に留意する。
  - ただし、本原則は、アルゴリズム、ソースコード、学習データの開示を想定するものではない。また、本原則の解釈に当たっては、プライバシーや営業秘密への配慮も求められる。
- アカウンタビリティの原則：利用者は、ステークホルダに対しアカウンタビリティを果たすよう努める
  - アカウンタビリティとは、判断の結果についてその判断により影響を受ける者の理解を得るため、判断に関する正当な意味・理由を説明したり、（必要に応じて）賠償・補償したりする等の措置をとること。

AIシステムの説明ができると

- ユーザの受容と信頼を得ることができる
- 法的側面
  - 倫理基準への適合性、公平性を示すことができる
  - 関係者は情報提供を受けることができる
  - 関係者は決定に対し異議を唱えることができる
- 説明的デバッギング
  - 欠陥のある性能指標を知ることができる
  - 不適切な属性を知ることができる
  - 学習データの分布と実環境での分布の違いを知ることができる
- 洞察力を高める
  - 従来入手できなかった情報を得ることができる
  - 因果関係の解明につながる手がかりを入手できる

人と機械の協働

- 人の判断はバイアスともなりえるが、説明は人の介在を可能にする

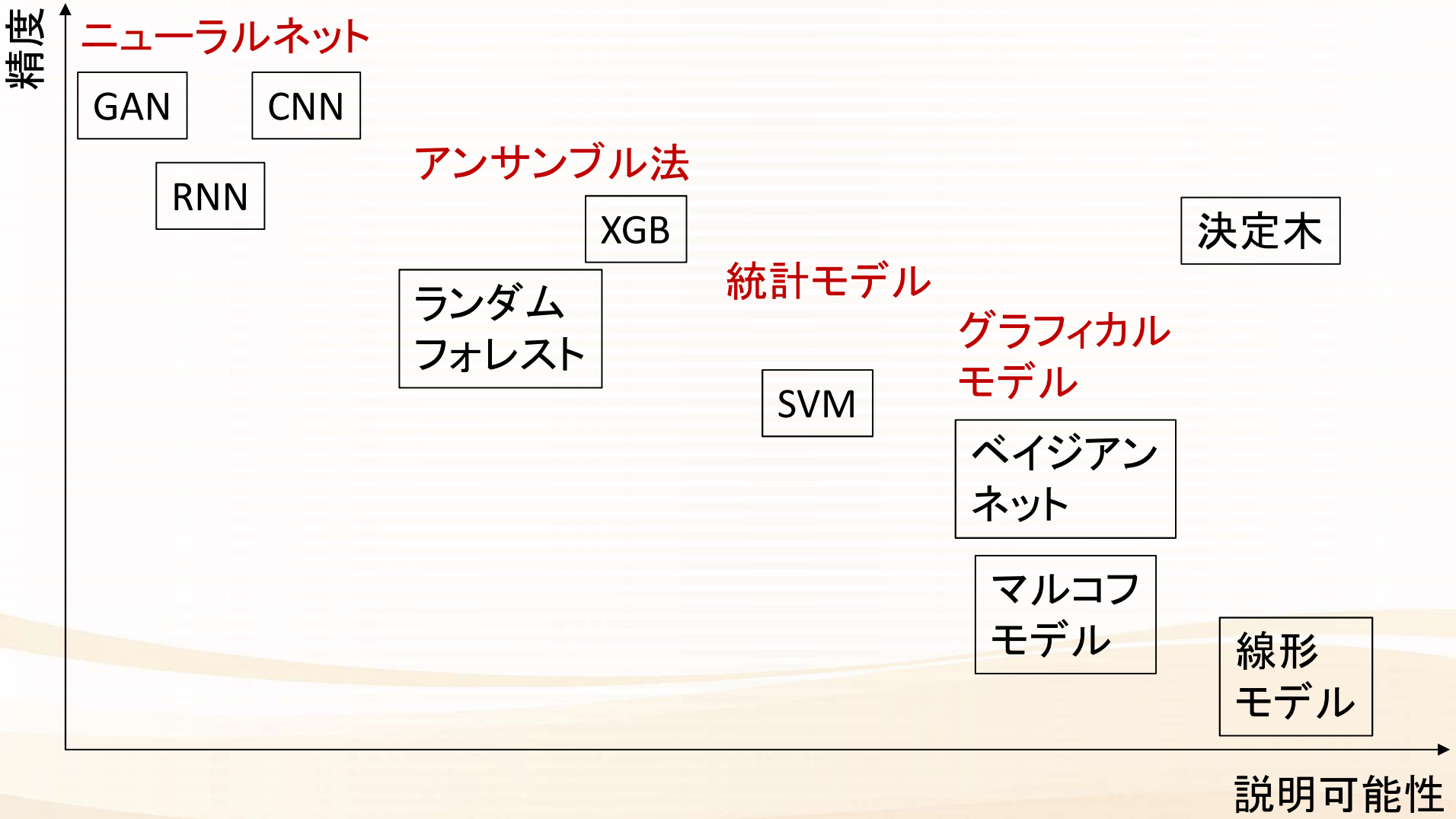


一般化誤差

一般化誤差＋人の経験

予測精度と説明可能性

- 学習モデルの予測精度と説明可能性はトレードオフの関係にある



説明に関する二つのアプローチ

- アプローチ１：事後の説明
  - ブラックボックスモデルがなぜそのように振る舞うかを説明
  - なぜそのように判断したか根拠を提示
  - ブラックボックスAIシステムとは別に、説明用サブシステムを用いて説明を生成
- アプローチ２：透明性のある設計
  - モデルがどのように機能するかを明らかにする
  - 透明性のレベル
    - モデルの透明性：入力に対する結果を人が予測できるか？
    - モデル部品の透明性：パラメータなど個々の部品の意味を理解できるか？
    - 学習アルゴリズムの透明性：アルゴリズムを理解できるか？

説明可能なAI

- 事後の説明
  - 局所的：個々のインスタンスの予測の説明
    - 入力特徴、影響のあるサンプルなどを提示
  - 大域的：モデルの振る舞いに関して大域的な説明を付与
    - 部分従属プロット
- 透明性のある設計
  - 線形回帰、ロジスティック回帰、決定木

局所的説明を与える方法

- 帰属問題
- ある入力に対するモデルの予測を入力の特徴に帰属させる
  - 物体認識の予測結果を、入力画像のピクセル・領域に帰属させる
  - 文章の感情分析結果を、入力文章の単語に帰属させる
  - 資金貸付の可否決定を、申請者の特徴に帰属させる
- 説明とは何か？何が提示できれば説明になるか？には立ち入らない
- 帰属問題の解決法
  - 切除
  - 特徴の勾配
  - スコア逆伝播

切除

- 各特徴を切除して、予測結果が変化した場合、判定根拠をその特徴に帰属させる
- 画像の物体認識問題では、画像の一部を切除して学習済モデルに与え、予測の変化を観察する



- 利点
  - 単純で解釈が直観的
- 欠点
  - 実際に存在しない入力に対して適切な予測が得られるか不明
  - 特徴間に相互作用がある場合に不適切な説明を生む
  - 切除部分の取り方は色々あり、計算量の点で高価

特徴の勾配

- 入力サンプルの特徴量の大きさと、学習モデルにおけるその特徴の勾配の積が大きいものを、判定の根拠とみなす
- ここで、特徴の勾配とは、出力を特徴で偏微分したもの
- 勾配が大きい箇所は、そこが変化すると出力が変化
- それらの箇所の中で、入力サンプルの特徴量が大きいということは、そこが予測を決定していると言える
- 線形モデルでの特徴量と回帰係数の積で考えるのと同じ

$$y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$$

- ただし、特徴量の取りえる範囲に対して、勾配が大きく変化する区間が小さいと、特徴間の重要度の比較が困難になる

スコア逆伝播

- ニューラルネットワークにおいて、予測値を逆方向に再分配
- 多層ネットワークにおいて、i+1層のニューロンのスコアを、i層からi+1層への重みで重みづけしてi層のニューロンのスコアを計算
- これにより、一旦、前向き計算によって予測値を計算し、その後、後ろ向きにたどっていくことで、入力の各特徴の貢献度を計算
- 利点
  - 概念的に単純
  - 実験的に機能することが検証されている
- 欠点
  - ニューロンのしきい関数が非線形であり、分配値をどう計算するかで様々な提案がある。どれを用いればよいか明らかでない。

協力ゲーム理論の応用

- 協力ゲーム理論
- 経済学の一分野であり、プレイヤが合理的であるとの仮定のもと、どのようなグループ分けが実現されるか（提携形成問題）、どのように利得・費用が分配されるか（利得分配問題）を議論

- 費用分担問題
  - 3つの地域が各々新たに一定量の水資源を必要としていて、全体の水資源を共同で開発することになり、3つの地域の間で、その費用をいかに分担するかを決める。交渉が決裂すれば、各地域が独立に水資源の開発を行う。
- アドホックネットワーク
  - ノードがメッセージを順に中継することで、各ノードとサーバが直接通信する場合に比べて伝送容量を増やす。
- 自律移動ロボットのチーム編成

提携形ゲームの定式化

- 複数エージェントの共同作業
- エージェントの集合N={Alice,Bob,Carol}
- 提携の集合{φ,{A},{B},{C},{AB},{AC},{BC},{ABC}}
  - 全員参加の提携{ABC}を全体提携と呼ぶ
- 提携構造
  [{A},{B},{C}], [{AB},{C}], [{AC},{B}], [{BC},{A}],[{ABC}]
- 特性関数
  $v(\varphi)=0$
  $v(A)=10$: Aliceが単独で行動すると10の利得
  $v(B)=20$: Bobが単独で行動すると20の利得
  $v(C)=30$: Carolが単独で行動すると30の利得
  $v(AB)=60$: AliceとBobが協力すると60の利得
  $v(AC)=70$: AliceとCarolが協力すると70の利得
  $v(BC)=80$: BobとCarolが協力すると80の利得
  $v(ABC)=120$: 3人が協力すると120の利得
- どのような提携が成立するか？提携の中で利得はどう分配されるか？

シャプレイ値

- シャプレイ値：限界貢献度（プレイヤの参加による利得の増分）の平均値
- Aliceの貢献度
  - A→B→C: $v(A)-v(\varphi)=10-0=10$
  - A→C→B: $v(A)-v(\varphi)=10-0=10$
  - B→A→C: $v(AB)-v(B)=60-20=40$
  - B→C→A: $v(ABC)-v(BC)=120-80=40$
  - C→A→B: $v(AC)-v(C)=70-30=40$
  - C→B→A: $v(ABC)-v(BC)=120-80=40$
- 上記の6つの場合が等確率で生じるとすると
  - Aliceの期待貢献度　$(10+10+40+40+40+40)/6=30$
- Bob，Carolについても同様に
  - Bobの期待貢献度　$(20+20+50+50+50+50)/6=40$
  - Carolの期待貢献度　$(30+30+60+60+60+60)/6=50$

全部足すと120
＝全体提携の利得

シャプレイ値の性質

- 公準１：全体合理性（効率性，パレート効率性）
  - 配分における利得の総和は提携が得る利益に等しい
- 公準２：nullプレイヤのゼロ評価
  - 提携が得る利益に影響を与えないエージェントの利得は0
- 公準３：対称性、代替性、無名性
  - 同じ限界貢献度を持つエージェントの利得は同じ
- 公準４：加法性
  - ２つの提携形ゲームに対して、２つの特性関数値を加えた新たな提携形ゲームを定義する。このとき、エージェントの利得は元のゲームの解における当該エージェントの利得の和に等しい
- 定理１：ゼロ単調ゲームにおいて、シャプレイ値は上記4つの公準を満たす。また、この4つの公準を満たすゲームの解はシャプレイ値に限る.
  - ゼロ単調ゲーム：Aliceが{Bob, Carol}に加わったとき，$v(\mathrm{ABC})-v(\mathrm{BC})\geq v(\mathrm{A})$が成立

説明へのシャープレイ値の応用

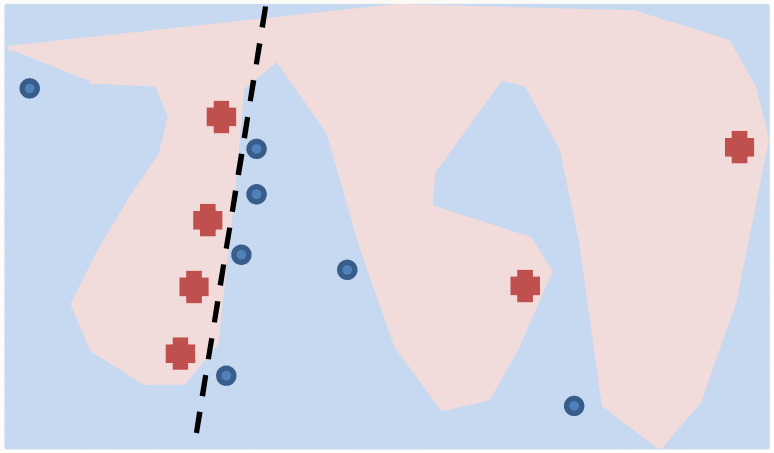
- 入力に対して協力ゲームを定義
- 入力の各特徴がプレイヤ
- モデルの予測が利得
- プレイヤA、B、Cが存在し、住宅物件の家賃を予測するとする
- 例えば、Aは物件の広さ、Bは駅からの距離、Cはペット可がどうか
- 家賃を各特徴量の線形和で近似する手法が考えられる
- しかし、線形回帰では、駅からの距離が近く、かつ、広いという二つの条件が揃った場合、家賃が高くなるといった複雑な状況を表現できない
- そこで、物件の広さ、駅からの距離、ペット可否が提携ゲームのAlice, Bob, Carolに対応すると考えると、二条件揃うことで価値が増大するといった状況を表現できる

説明へのシャプレイ値の応用

- ただし、シャプレイ値の計算には、特性関数のすべての値、つまり、プレイヤのすべての部分集合に対する利得の値がわかっていることが必要
- 例えば、駅からの距離が15分でペット可で、部屋の広さが与えられていない物件は、現実には存在しない
- そこで、その特徴量のみが等しい、または、近い値であるデータをサンプルとして取り出し、そのサンプル全体でのスコアの平均を計算
- 例えば、(60㎡、15分、ペット可)、(70㎡、14.5分、ペット可)、(80㎡、15.5分、ペット可)、…といったデータに対して、平均した家賃を(*、15分、ペット可)の値とする

局所的線形近似

- どの特徴量が判断に重要であったかを提示
- モデルを説明対象データの周辺で線形モデルで近似
- 線形モデルの係数の大小で、各特徴量の大小を決定



- モデル用データ表現 x : one-hot、埋め込み表現など
- 人用データ表現 x' : 単語や画像のパッチなど
- モデル$f(x)$を人用データ表現であるx'を説明変数として線形近似
- 回帰係数の大きさを予測結果への影響の大きさとみなす

大域的説明

- 部分従属プロット
  - モデルの予測値が、1つの属性に対して平均的にどのように依存しているかを示す図。横軸に1つの属性、縦軸にモデル予測値を図示
- 置換
  - 1つの属性に関して、値をランダムに入れ替えて、予測結果を評価。予測誤差が増加する属性は予測の決定に重要と考えられる
- Born Again Trees
  - 複雑なモデルを決定木で近似
  - 学習されたモデルを使って擬似訓練データを作成（元の学習データそのものではない）
  - 生成された擬似訓練データを使って決定木学習

透明性のあるモデル

- 決定木
- 決定集合：意思決定ルールの集合
- If X satisfies (condition A AND condition B) OR (condition C) OR …, then Y=1.
- k近傍法
- 一般化加法モデル

まとめ

- 人工知能技術のビジネス活用において、人間が意思決定権を持つ場合も多い（医療分野など）
- 活用促進には、人を意思決定ループの中に組み入れることが必要
- そのためには、人が理解できる形で、人工知能の判断根拠を示すことが必要
- 説明可能AIとして、活発な研究が行われている
  - 局所的説明：個々のインスタンスに関して、判断根拠を特徴量に帰属させる
  - 大域的説明：モデル全体を可読性の高いモデルで表現
  - 透明性のあるモデル：決定木などモデル自体を理解可能なものに

ビジネス活用（施策実施）

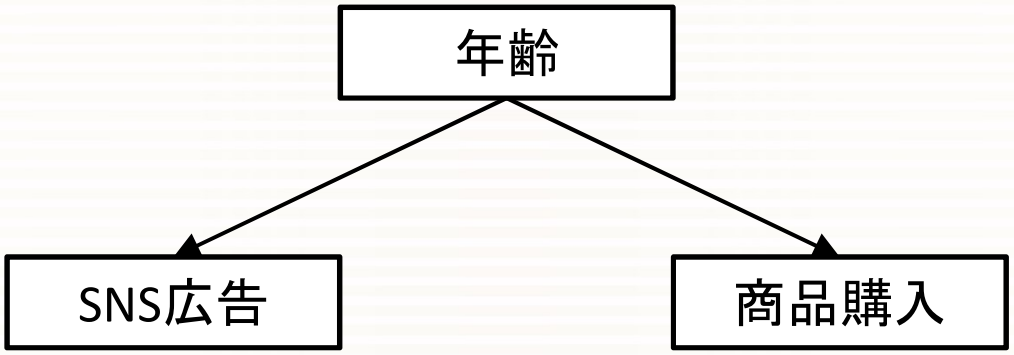
機械学習のビジネス活用におけるリスク

- 機械学習の応用領域の拡大
- ヘルスケアや金融：健康被害や多額の損失など、小さな予測誤りが大きな被害をもたらす可能性が存在
- この種の領域における機械学習への要求
  - 説明可能性
  - 安定性（A病院の治療データを学習して得られたモデルをB病院に適用できるか？）
- 説明の失敗や安定性を欠く要因の一つが相関と因果関係の取り違え
  - 広告掲示数と商品購入率は相関を示す
  - では、商品購入率をさらに増やすには、広告掲示回数を増やせばよいか？
  - 例えば、１１月より１２月に広告掲示回数を増やしていたとすれば、クリスマスシーズンだから商品購入率が増加したのかもしれない

見かけ上の因果関係

- ある商品はEコマースからの購入が多い
- よって、SNS広告を出すと、その商品がよく売れるようになる（本当？）

- 実は、年齢が若いからその商品を購入している、また、年齢が若いからSNSをよく視聴しているということかもしれない



- この場合、見かけ上SNS広告掲示と商品購入に因果関係があると見えるだけで、SNS広告を出しても売り上げは向上しない
- この「年齢」を交絡因子と呼ぶ

因果効果

- 因果効果：原因が結果に与える効果

$$Y^{t=1} - Y^{t=0}$$

- 介入（処置）$t = \{0、1\}$
- $Y^{t=1}$：広告を見たときに商品を購入する確率
- $Y^{t=0}$：広告を見なかったときに商品を購入する確率
- 広告を見せたとすれば、$Y^{t=0}$は反事実的な条件に基づく潜在的結果
- 問題は、現実には$Y^{t=1}$と$Y^{t=0}$のどちらか一方の結果しか観察されないこと
- これは因果推論の根本問題と呼ばれ、個体レベルでの因果効果については識別不可
- 因果推論の根本問題に対する対応策は、平均因果効果を考えること

$$E[Y^t] - E[Y^t]$$

因果推論の前提条件

- データから平均因果効果を推定できるためには、以下の仮定が必要
- 交換可能性

$$E\left[\text{購買あり}^{\text{広告あり}} \middle| \text{広告あり}\right] = E[\text{購買あり}^{\text{広告あり}} | \text{広告なし}]$$

  - 広告配信した集団の中で、広告を配信した場合の購買率（左辺）と広告配信しなかった集団の中で、広告を配信した場合の購買率（右辺）が同じになる
  - 交換可能性を仮定すれば、２群の差は広告配信の有無だけであり、実際に観察される購買率の差は原因「広告のあり／なし」によってもたらされたと推論できる
  - もし「年齢」が交絡因子であるとすると、広告なし群には年齢が高い人が多く含まれることになるため、交換可能性は成立しない

因果推論の前提条件

- 一貫性

$$E\left[\text{購買あり}^{\text{広告あり}}\middle|\text{広告あり}\right] = E[\text{購買あり}|\text{広告あり}]$$

  - 広告配信をした場合に期待される購買率は、実際に広告を配信した場合の購買率に等しい
  - 介入を実際に受けた（受けていない）ときの結果は、介入を受けた（受けていない）場合の潜在的結果に一致する
  - 当たり前のことを言っているように見えるが、商品購入目的が自己利用と転売の2通りあるような場合、一貫性は成立しない
- 正値性
  - いずれの介入（広告あり／なし）に割り当てられる確率も0ではない
  - いずれかの介入に割り当てられる確率が0の群があれば、潜在的結果を定義できなくなる

ランダム化実験

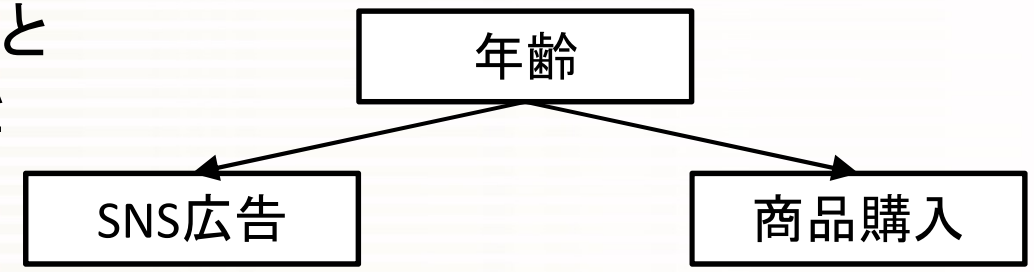
- 先の例で、SNS広告の効果を測定するには？
- ランダム化比較試験（RCT）
  - 介入群と対照群の割り付けを無作為に行い、介入法の違いが結果に与える影響を調べる
  - A/Bテストを行ってコンバージョン率を測定するなど
  - 因果関係研究の標準的手法であるが、高価で時間がかかる
  - 倫理上の問題が生じる場合もある
    - 例えば、タバコと肺がんの因果関係を調べるときに、タバコを吸わせる群と吸わせない群を分けての長期間観察は困難

観察研究

- ランダム化比較試験ができないときどうする？
- 観察研究
  - 介入法の割り付けを無作為に行うことをあきらめる
  - 人為的、能動的な介入を伴わず、ただその場に起きていることや起きたことを観察
  - 単純で効率的
  - ビッグデータの活用の可能性
- SNS広告の例では、
  - すでに蓄積されたデータからSNS広告の配信群と未配信群を特定し、その２群間で商品購入率に差があるか調査
  - ただし、年齢、性別、居住地などによって受け取る広告が異なる場合、それらが交絡因子となる恐れがある

データの偏り

- 交絡バイアス
  - もし、若年層ほどSNSの視聴時間が長いとすると、年齢という点でSNS高視聴群とSNS低視聴群でデータの偏りが存在
  - 本来関係のないSNS広告掲示数と商品購入率に因果関係があると見える



- 選択バイアス
  - 新規サービス受けた人ほど回答しやすい
  - 顧客満足度の高い人がアンケートに回答しやすい
  - 結果、新規サービスは顧客満足度を高めるという関係があるように見える

喫煙習慣と健康

- 「タバコを吸っている人ほど健康である」
- タバコが健康被害をもたらすとしても、
- 調査対象が高齢者集団であった場合、タバコによって健康被害を受けやすい人はすでに亡くなっている可能性が高い
- また、健康状態が良い人ほど長生きする
- つまり、調査対象者は高齢になるまでタバコを吸い続けることができるほど健康であり、健康な喫煙者の割合が多くなることで、タバコを吸うほど健康であるという全く逆の関連が生まれる

因子間の関連の発見

- 有効巡回グラフDAGsを用いる
- 変数の関係性を推測される因果関係の向きに合わせて矢印で繋ぐ
- 二変数以上から影響を受けていない場合は交絡バイアスを引き起こす因子となる可能性がある
- 二変数以上から影響を受けている場合は選択バイアスを引き起こす因子となる
- DAGsの中で、変数間の流れが繋がったときに見せかけの関係性が生じる

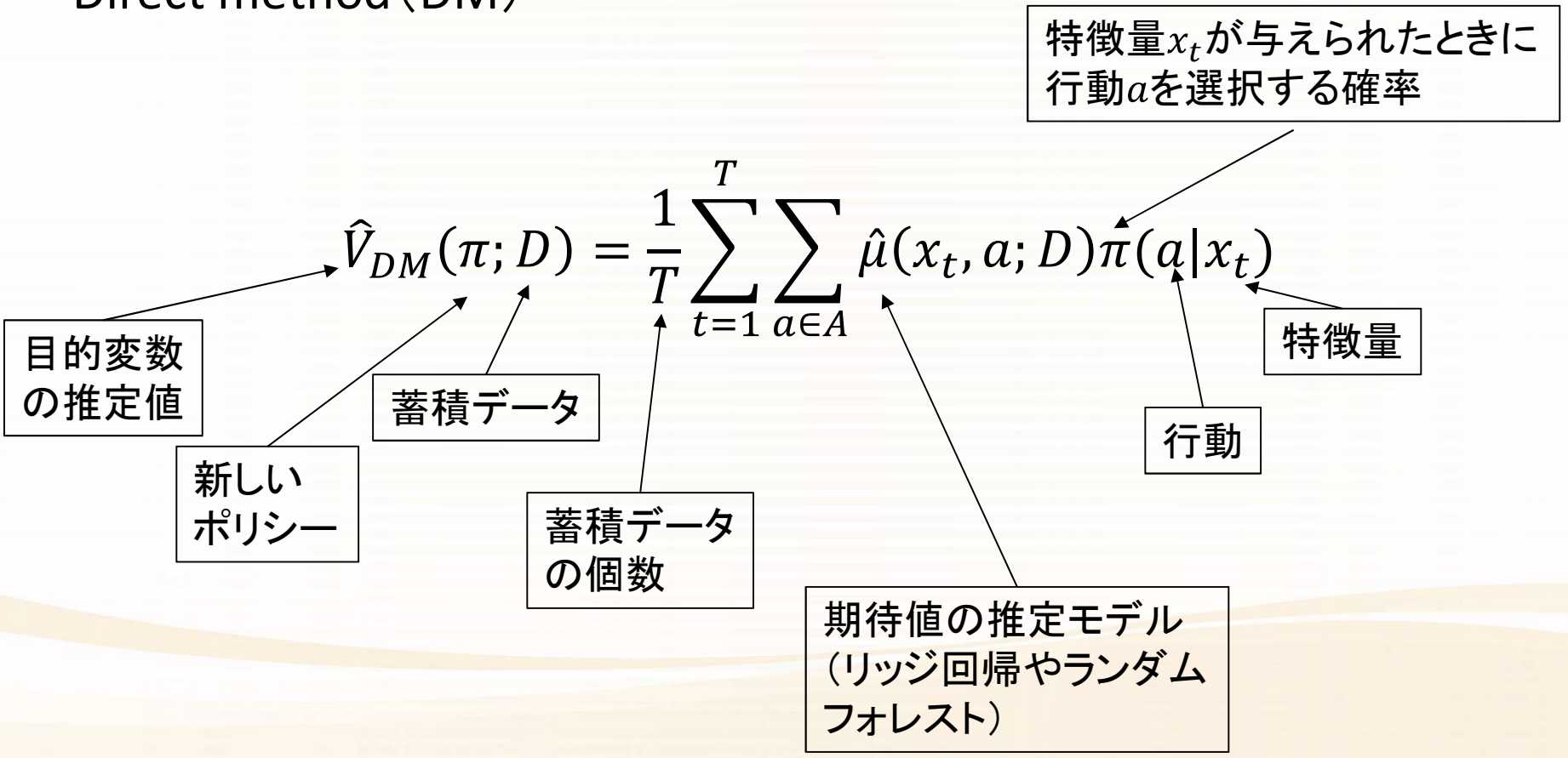
反実仮想機械学習

- 因果効果を予測したり、過去に何らかの基準で収集された雑多なデータを使って仮想的な施策の性能を評価するなどの、反実仮想の推論を含むタスクを解くための技術

- 映画推薦を考える
- 年齢、性別などユーザ属性 X に応じて、映画を推薦
- その映画が視聴されたかどうかのデータが得られる
- ある推薦ポリシーのもとで映画 a を推薦し、そのデータを取得
- そのデータをもとに新たな推薦ポリシーを策定
- このとき、新たな推薦ポリシーの性能は？
- ユーザ属性 X に対して、元の推薦ポリシーと新たな推薦ポリシーが同じ映画 a を推薦するとき、データからユーザの反応を推測できる
- 新たな推薦ポリシーが映画 b を推薦するとき、ユーザの反応を知ることができない

ポリシー外評価

- ポリシー外評価の目的は、ある仮想的な介入方策の性能をそれとは異なる方策が集めたログデータを使って評価すること
- Direct method（DM）



ポリシー外評価

- Inverse probability weighting法（IPW）

$$\hat{V}_{IPW}(\pi; D) = \frac{1}{T}\sum_{t=1}^{T} Y_t \frac{\pi(a_t|x_t)}{\pi_b(a_t|x_t)}$$

新しいポリシー

目的変数の推定値

古いポリシーのもとで観測された目的変数の値

古いポリシー

- Doubly robust法

$$\hat{V}_{DR}(\pi; D) = \hat{V}_{DM(\pi;D)} + \frac{1}{T}\sum_{t=1}^{T} (Y_t - \mu(x_t, a_t)) \frac{\pi(a_t|x_t)}{\pi_b(a_t|x_t)}$$

DMとIPWの組み合わせであり、バイアスが大きくなるDM法の欠点と分散が大きくなるIPW法の欠点を解消

まとめ

- ビジネス活用（施策実施）において、予測をするだけでなく、何らかの行動を取る場合、リスクが存在
- リスクが生じる要因の一つは相関と因果関係の取り違え
- 相関は、因果関係、交絡バイアス、選択バイアスのいずれかであり、後者の二つは見かけの因果関係
- 平均因果効果の測定には、交換可能性、一貫性、正値性が仮定される
- 因果効果測定にはランダム化比較実験があるが、一般には高価
- 観測データを用いる方法は安価であるが、バイアスの存在を考慮する必要がある
- 因果推論に関する機械学習の応用として、反実仮想機械学習がある
- 機械学習の技術を用いて、反実仮想におけるデータを推定