Python による業務効率化：１年生クラス編成

岩本　豊 *　　山本 浩二 †

Enhancing Business Processes with Python: First-Year Class Formation

Yutaka IWAMOTO *　　Koji YAMAMOTO †

In this paper, we report on how we improved the organization of first-year classes for the 2023 school year using Python programming. Previously, the formation of first-year classes had been a challenge due to the extensive workload and the inconsistency in results depending on the person in charge. However, our developed programming significantly reduced the workload and enhanced the accuracy of the system.

1.　緒言

新居浜高専は５学科からなる高専であり、新入生の入学定員は 200 名である。低学年（１・２年）に対しては各学科が入り混じる混合学級制を採用しているため、クラスを編成する必要がある。２年生に対しては１年次の成績を参考にクラス編成を行う。一方、１年生に対しては、異なる入試条件（推薦または学力）と留年生の配置に加え、各学科の人数、男女比、芸術科目の選択、そして配慮の必要な学生に対する対応等、多くの条件を加味しながらクラス編成を行う必要がある。従前、この１年生に対するクラス編成は手作業で行われてきたが、すべての要求に対して満足のいく成果を得ようとすると、１日４時間の編成作業を行うとして、１週間から 10 日の仕事量を必要とした。人知れず行われてきたこの編成作業であるが、その作業量の多さもさることながら、担当者（一般教養科または数理科の教務主事補）によって精度にばらつきがでることが懸念されていた。それらを解消するために、R５年度の編成作業においては、プログラミング言語（Python）を用いて業務効率化を試みた。本稿はその報告である。

2.　クラス編成の基本情報

2－1　人数と男女比

Ｒ５年度の１年生（新入生に留年生を加えた数）は、機械工学科（M科）41 名、電気情報工学科（E科）44 名、電子制御工学科（D科）42 名、生物応用化学科（C科）43 名、環境材料工学科（Z科）43 名の合計 213 名（内女子学生 76 名）となった。各クラスの人数を均等にすると、クラス人数は 42 名または 43 名となる。

余談ではあるが、本校の特徴の１つとして女子学生比率が高いことが挙げられる。Ｒ５年度についても、1 年生の女子学生比率は約 35.7 %であり、高専の全国平均である約 23 % [1] を大きく上回った。

2－2　クラス編成上の留意点

１年生のクラス編成においては以下の項目を均等にすることを目指す。

(1) 総人数
(2) 男女比
(3) 学科人数
(4) 入試成績
(5) 推薦・学力比

また、以下の内容についての配慮を行う。

(6) 同姓人数
(7) 兄弟は必ず別クラス
(8) 学生相談室からの配慮依頼
(9) 芸術の選択者数

令和 5 年 8 月 1 日受付 (Received Aug. 1, 2023)
* 新居浜工業高等専門学校数理科（Faculty of Fundamental Science, National Institute of Technology (KOSEN), Niihama College, Niihama, 792-8580, Japan）
† 新居浜工業高等専門学校エンジニアリングデザイン教育センター (Center for Engineering Design Education, National Institute of Technology (KOSEN), Niihama College, Niihama, 792-8580, Japan)

3. データの前処理

この章では教務係から与えられたエクセルファイル『05 新 1 年クラス分け元.xlsx』に対して、手作業で行う前処理について述べる。クラス編成の成否にかかわる大事な作業である。

3 – 1 与えられるデータ

１年生のクラス編成を行うにあたり、教務係からは次のようなデータが与えられる (X には数字が入る)：

受験番号	40-XXXXX
氏名	高　専　太　郎
氏名カナ	コウセン　タロウ
性別	男
中学校名称	○○市立　○○中学校
入学学科名称	M
・・中略・・	
入試成績	1156
・・中略・・	
備考	

なお、学力入試と推薦入試の満点はそれぞれ 1400 点と 180 点であり、その内何点取ったのかが入試成績に記録されている。

3 – 2 芸術の選択

本校の芸術科目は音楽と美術の２つがあり、授業コマ数の関係で、音楽が２クラス分、美術が３クラス分割当られている。慣例では、１組と３組が音楽美術の混在クラスで、２組・４組が美術のみ、５組が音楽のみのクラスとなる。今年度も、各組と選択者数の関係は慣例に従った。選択の決定に関しては、「選択科目の受講申告について」という希望調査を実施し、その結果をもとに選考する。Ｒ５度は 213 名から２クラス分の人数として、$42 \times 2 = 84$ 名を音楽の選択者として決定した。なお、留年生の芸術選択は本人の希望を優先する。

この選考作業は、苦手な科目を選択させず、かつ公平性を保つ必要のある大事な作業であるが、本題から外れるため、本稿ではその詳細には立ち入らない。

3 – 3 音楽と推薦

教務係から与えられたファイルに対し、音楽決定の欄と、推薦の欄を新たに設けたファイル『05 新 1 年クラス分け.xlsx』を作り、3 – 2 節の選考で音楽に決定された者には○、推薦合格者には●を入力する。つまり、ファイルとしては新たに次の項目が加わることになる：

音楽決定	○
推薦	●

3 – 4 留年生の記入

与えられたデータファイルからあらかじめ入試成績の平均点を算出しておく。今回はおよそ 603 点であった。留年生については、氏名、氏名カナ、性別、そして芸術の選択の情報が与えられる。留年生の内、配慮が必要でない学生については『05 新 1 年クラス分け.xlsx』に記入する。その際、音楽決定の部分と、入試成績の部分を新たに書き加える。また、受験番号は 40-9000X のような形で記入し、更に入試成績については、全体の平均に影響を与えないよう、平均点の 603 を記入しておく（空欄にすると以降の処理でエラーが出る）。

このようにしてできたエクセルファイル『05 新 1 年クラス分け.xlsx』を Python に読み込ませる。

3 – 5 要配慮学生

学生に対する配慮としてクラス編成の立場からは (i) 担任を指定する、(ii) 特定の学生と同一クラスにする、(iii) 特定の学生とは同一クラスにしない、の３つについて対応する。Ｒ５年度については、学生相談室長と相談した結果、(i) の対応は行う必要が無いと判断された。(ii) と (iii) の配慮が必要な学生はそれぞれ１名であった。また、それら配慮の必要が学生に対し、同一クラスにすべき学生、しない学生はそれぞれ１名であったため、２人１組のペアを２つ作り、それら２つのペアに対して同一クラスにするか否かを条件として与えることになった。

(ii) と (iii) の該当学生リストはそれぞれ『同組希望.xlsx』、『ペア禁止学生.xlsx』とした。各ファイルには以下のように記述する：

受験番号１	受験番号２
40-AAAAA	40-BBBBB
40-CCCCC	40-DDDDD

この例では２つのペアが記入されており、40-AAAAA と 40-BBBBB が１つのペアで、40-CCCCC と 40-DDDDD がもう１つのペアとなっている。

以上がデータの前処理となる。

4. 数理モデリングと実装

この章より Python3.9 による処理を始める。ライブラリとしては、pandas, PuLP の２つと、可視化用に matplotlib を用いる。ディストリビューションとして Anaconda を用いれば、PuLP 以外はインストール済である。なお、PuLP に関しては次の HP を参考にした：

`http://www.nct9.ne.jp/m_hiroi/light/pulp01.html`

以下のコードは文献 [2] 第 II 部第 3 章を参考にして作成した。そのため、解説の無いものについては [2] を参照してほしい。また、長いコードについては紙面の都合で折り返して記述している。その際には基本的に『\』を挿入して折り返しているが、実際には『\』を外して改行せずに記述していることに注意する。

基本となるデータフレーム名は `s_df` とした。

```
# データ処理のためのモジュール pandas の取り込み
import pandas as pd
# 新入生のデータ取得
s_df = pd.read_excel('05 新 1 年クラス分け.xlsx')
```

4－1　姓名の項目分割と統計量の確認

ここでは姓名の項目分割と統計量の確認をしておく。これらは後に、上記 (9) の配慮を行う際に使う項目となる。

まず姓と名の項目を分割する：

```
# 姓名の分割
s_df[['姓', '名']] = \
  s_df['氏名カナ'].str.split(' ', expand = True)
```

実行するとデータフレーム s_df には、新たに次の 2 項目が加わる：

姓	コウセン
名	タロウ

次に入試成績の統計量を確認しておく：

```
# 入試成績の統計量の確認
s_df['入試成績'].describe()
```

実行すると以下のような出力が得られた：

```
count      2XX.000000
mean       603.215311
std        424.845669
min        131.000000
25%        159.000000
50%        906.000000
75%        991.000000
max       1224.000000
Name: 入試成績, dtype: float64
```

なお、留年生の人数が影響する部分は XX と表記している。

念のため入試成績の点数分布を図示しておく：

```
# 入試成績の点数分布の確認
s_df['入試成績'].hist()
```

実行すると次のようなヒストグラムが表示される：

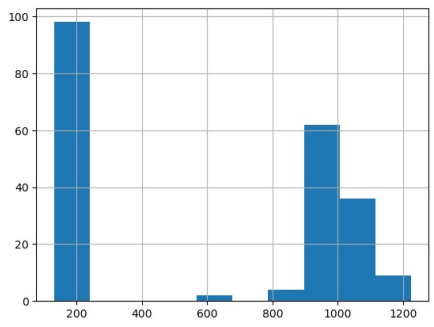

図 1　入試成績の得点分布

このグラフの左の山が推薦入試の学生、600 点あたりが留年生、そして右の山が学力入試による学生の分布である。

4－2　（1）クラス人数の調整

ここでは 2－1 で述べたように、各クラスの人数が 42 名または 43 名になるよう問題を設定する：

```
# Python モジュール PuLP の取り込み
import pulp
# 数理モデルのインスタンス作成
prob = pulp.LpProblem('ClassAssignmentProblem',
  pulp.LpMaximize)
# 学生のリスト
S = s_df['受験番号'].tolist()
# クラスのリスト
C = ['1-1', '1-2', '1-3', '1-4', '1-5']
# 学生とクラス名のペア（全組み合せ）のリスト
SC = [(s,c) for s in S for c in C]
# 学生をどのクラスに割当るかを変数として定義
x = pulp.LpVariable.dicts('x', SC, cat='Binary')
# 各学生は１つのクラスに割当る
for s in S:
    prob += pulp.lpSum([x[s,c] for c in C]) == 1
# 各クラスの学生の人数は 42 人以上、43 人以下とする。
for c in C:
    prob += pulp.lpSum([x[s,c] for s in S]) >= 42
    prob += pulp.lpSum([x[s,c] for s in S]) <= 43
```

上記 x は辞書型変数であり、key は (受験番号, クラス名)、value は 0 か 1 になる。x が 1 のときの受験番号に、そのときのクラス名が割当られることに注意する。

4－3　（2）男女比の調整

女子学生は 76 名のため、各クラス 15 名以上 16 名以下となるよう調整した。

```
# 男子学生のリスト
S_male = [row.受験番号 for row in \
  s_df.itertuples() if row.性別 == '男']
# 女子学生のリスト
S_female = [row.受験番号 for row in \
  s_df.itertuples() if row.性別 == '女']
# (3) 各クラスの女子学生の人数は 15 人以上 15 人以下
for c in C:
    prob += pulp.lpSum([x[s,c] for s in \
      S_female]) >= 15
    prob += pulp.lpSum([x[s,c] for s in \
      S_female]) <= 16
```

4－4 （3）学科人数の調整

M 41 名、E 44 名、D 42 名、C 43 名、Z 43 名であることから、各クラスに同一学科学生は 8 人以上とした。また、E，C，Zについては 42 名以上のため、条件として各クラス 9 名以下の条件も加えている。

```
# 各学科学生のリスト
S_M = [row.受験番号 for row in s_df.itertuples(\
  )  if row.入学学科名称 == 'M']
S_E = [row.受験番号 for row in s_df.itertuples(\
  ) if row.入学学科名称 == 'E']
S_D = [row.受験番号 for row in s_df.itertuples(\
  ) if row.入学学科名称 == 'D']
S_C = [row.受験番号 for row in s_df.itertuples(\
  ) if row.入学学科名称 == 'C']
S_Z = [row.受験番号 for row in s_df.itertuples(\
  ) if row.入学学科名称 == 'Z']
# 各クラスで各学科８人以上とする
for c in C:
    prob += pulp.lpSum([x[s,c] for s in S_M]) \
      >= 8
    prob += pulp.lpSum([x[s,c] for s in S_E]) \
      >= 8
    prob += pulp.lpSum([x[s,c] for s in S_D]) \
      >= 8
    prob += pulp.lpSum([x[s,c] for s in S_C]) \
      >= 8
    prob += pulp.lpSum([x[s,c] for s in S_Z]) \
      >= 8
# E,C,Z は 42 名以上なので９人以下の条件を加える
    prob += pulp.lpSum([x[s,c] for s in S_E]) \
      <= 9
    prob += pulp.lpSum([x[s,c] for s in S_C]) \
      <= 9
    prob += pulp.lpSum([x[s,c] for s in S_Z]) \
      <= 9
```

4－5 （4）入試成績分布の調整

ここではまず各クラスの入試成績の分布が均等になるように調整する。

```
# 入試成績の降順に順位を求め、
# データフレームに「入試成績順位」列を追加
s_df['入試成績順位'] = s_df['入試成績'].rank(\
ascending=False, method='first')
# 順位順にクラス編成し、
# データフレームに「初期割当クラス」列を追加
class_dic = {0:'１-１', 1:'１-２',\
 2:'１-３', 3:'１-４', 4:'１-５'}
s_df['初期割当クラス'] = s_df['入試成績順位'\
].map(lambda x:x % 5).map(class_dic)
```

実行するとデータフレーム s_df には、新たに次の 2 項目が加わる：

```
入試成績順位　74
初期割当クラス 1-4
```

最初に入試順位を 5 で割った剰余を求めて各クラスに振り分けた。

つぎに初期割当クラスの辞書型の init_flag を作成する。これは受験番号とクラス名をすべて組み合わせた集合である。key は (受験番号, クラス名) で、value を 0 で初期化している。なお、value が 0 はクラス未割当てを意味することに注意する。

```
# 初期割当クラスの辞書型の init_flag を作成
init_flag = {(s,c):0 for s in S for c in C}
for row in s_df.itertuples():
    init_flag[row.受験番号, row.初期割当クラス] = 1
```

その後、初期クラス編成と最適化結果のクラス編成をできるだけ一致（最大化）させる。

```
# 目的関数:
prob += pulp.lpSum([x[s,c] * init_flag[s,c]\
 for s,c in SC])
```

最後に平均点による分布の均一化を行う。Ｒ５年度は各クラスの学力試験の平均点を学年平均点±４点の範囲内に設定した。なお、条件を強くしすぎると計算に時間がかかり，かつ学力分布が崩れることに注意が必要である。

```
# 学力を辞書表現に変換
score = {row.受験番号:row.入試成績 \
for row in s_df.itertuples()}
# 平均点の格納
score_mean = s_df['入試成績'].mean()
# 目的関数の設定
for c in C:
    prob += (score_mean -4) * pulp.lpSum([x[s,c]\
     for s in S]) \
     <= pulp.lpSum([x[s,c]*score[s] \
     for s in S])
    prob += pulp.lpSum([x[s,c]*score[s] \
     for s in S])\
     <= (score_mean +4) * pulp.lpSum([x[s,c]\
     for s in S])
```

4－6 （5）推薦・学力入学生の人数調整

推薦入学の学生は 98 名であったため、各クラス 19 名以上に設定した。

```
# 推薦入学者のリスト
S_recommendation = [row.受験番号 for row in \
  s_df.itertuples() if row.入試成績 <= 200]
```

```
# 学力入学者のリスト
S_academic =  [row.受験番号 for row in \
  s_df.itertuples() if row.入試成績 > 200]
# 各クラスに推薦入学者を 19 人以上割当る。
for c in C:
    prob += pulp.lpSum([x[s,c] for s in \
      S_recommendation]) >= 19
```

4－7　（6）同姓人数の調整

同姓名と、その重複数に関するデータ取得する。

```
# 姓と重複数を結び付けて辞書式データに変形
Second_Name = s_df['姓'].value_counts().to_dict()
# 結果を出力して同姓の最大数を確認する
print(Second_Name)
```

結果、Ｒ５年度は同姓が最大８名いることが分かった。したがって、s_df['姓'].value_counts().iat[0] (姓の重複度最大値) は８である。このことから、以下の for 文で i は $0-1=-1$ から $7-1=6$ まで動くことに注意する。

```
# 同姓人数の条件設定
for i in range(s_df['姓'].value_counts(\
).iat[0] - 1):
    N_n = [k for k,v in Second_Name.items(\
    ) if v == i+2]
    Num_N_n = [( row.姓, row.受験番号) for \
    row in s_df.itertuples() if row.姓 in N_n]
    for k in N_n:
        Ni = [s for name, s in Num_N_n \
        if name == k]
        for c in C:
            if i <=3: #3+2=5 の重複同姓人数の条件
                prob +=pulp.lpSum([x[s,c] \
                for s in Ni]) <= 1
            else:
                prob +=pulp.lpSum([x[s,c] \
                for s in Ni]) <= 2
```

たとえば $i=0$ のとき、N_n は ['アベ', 'イノウエ', ...] のような重複数が 2 の姓の集合であり、Num_N_n は [('アベ', '40-aaaaa'), ('アベ', '40-bbbbb'), ('イノウエ', '40-ccccc'), ('イノウエ', '40-dddd'),...] のようなデータになる。更に name == 'アベ' ならば、Ni は姓の同じ受験番号の集合 ['40-aaaaa', '40-bbbbb'] になっていることに注意する。

ここでは同姓が５名以下なら、クラスを別にし、６名以上の場合のみ同一クラスに２名の重複同姓者を許すという（強い）条件にした。年度によってはもう少し条件を緩めないと解が得られない可能性がある。

4－8　（7）・（8）配慮依頼への対応

Ｒ５年度は兄弟らしい学生はみあたらなかった。もし、兄弟が入学した場合には、次のペア禁止学生ファイルにその該当学生の受験番号を記入する。

特定ペアデータ (ペア禁止学生ファイル.xlsx) の確認

```
# Excel ファイルからペア禁止学生データの取得
N_pair_df = pd.read_excel('ペア禁止学生.xlsx')
# 学生の特定ペア禁止リスト
SN = [(row.受験番号 1, row.受験番号 2) \
  for row in N_pair_df.itertuples()]
# 特定ペアの学生は同一クラスに割当ない。
for s1, s2 in SN:
    for c in C:
        prob += x[s1,c] + x[s2,c] <= 1
```

同組に配置する学生については以下のように設定する。

```
# Excel ファイルから同組希望データの取得
s_pairA_df = pd.read_excel('同組希望.xlsx')
# 学生の同組リスト
SSA = [(row.受験番号 1, row.受験番号 2) \
  for row in s_pairA_df.itertuples()]
# (7) 特定ペアの学生は同一クラスに割当ない。
for s1, s2 in SSA:
    for c in C:
        prob += x[s1,c] == x[s2,c]
```

4－9　（9）芸術の選択

音楽に決定された 84 名を、音楽と美術の混在クラスでは 19 人以上 21 人以下、音楽のみのクラスでは 42 人以下、美術のみのクラスでは 0 人として振り分けるよう設定した。

```
# 音楽選択決定者のリスト
S_music = [row.受験番号 for row in \
  s_df.itertuples() if row.音楽決定 == '〇']
# 美術決定者のリスト
S_art = [row.受験番号 for row in \
  s_df.itertuples() if row.音楽決定 != '〇']
# 音楽と美術の混合クラス
C_music_art = ['1-1', '1-3']
# 音楽クラス
C_music = ['1-5']
# 美術クラス
C_art = ['1-2', '1-4']
# 音楽と美術の混合クラス編成
for c in C_music_art:
    prob += pulp.lpSum([x[s,c] for \
      s in S_music]) >= 19
    prob += pulp.lpSum([x[s,c] for \
      s in S_music]) <= 21
# 音楽クラス
```

```
for c in C_music:
    prob += pulp.lpSum([x[s,c] for \
      s in S_music]) <= 42
for s in S_art:
    prob += pulp.lpSum([x[s,c] for \
      c in C_music]) == 0
# 美術クラス
for c in C_art:
    prob += pulp.lpSum([x[s,c] for \
      s in S_music]) == 0
```

4－10　求解

解を求める。解があれば status は Optimal (1) になり、解がない場合は Infeasible (-1) となる。

```
# 求解
status = prob.solve()
print(pulp.LpStatus[status])
```

Ｒ５年度は、ここまでに設定した条件で Optimal となり、無事解を求めることができた。

4－11　最適化結果の出力

クラス編成の最適化を実行した結果をエクセルファイルとして出力する。なお、ここでは最適化という言葉を用いたが、Python によって得られる解は必ずしも最適ではなく、実際には与えられた条件を満たす解による（最適解に向けた）近似解の１つである。そのため、問題の解として得られる結果は毎回異なることに注意する。

まず、各学生が複数のクラスに割当られていないかどうかを確認しておく。

```
# マッチングの確認
for s in S:
    # 割当られたクラスを取得
    assigned_class = [c for c in C \
    if x[s,c].value()==1]
    # 1 つのクラスに割当られているか確認
    if len(assigned_class) != 1:
        print('error:', s, assigned_class)
```

実行して error が出力されていなければ、各学生に対し、ただ１つのクラスが割当られていることになる。

以上を確認した後、今回の解をエクセルファイルに出力させる。以下のコードにおいて、S2C の key は受験番号、value はクラス名の辞書型データとなっていることに注意する。

```
# 出力用のデータフレーム
result_df = s_df.copy()
# 各学生の割当られたクラスの情報を辞書に格納
S2C = {s:c for s in S for c in C \
  if x[s,c].value()==1}
# 出力用データフレームに「割当クラス」列を追加
result_df['割当クラス'] \
  = result_df['受験番号'].map(S2C)
# 検証用のデータフレームを Excel 出力　※成果物
result_df.to_excel('./クラス編成案.xlsx',\
  sheet_name= 'クラス編成')
```

実行すると「クラス編成案.xlsx」というエクセルファイルが作成される。「クラス編成」シートに今回の成果物が出力され、「割当クラス」列に書かれているのが今回のプログラムによって割当られたクラスである。

5.　最適化結果の検証

最適化モデルの検証を行う。ここで用いるデータフレームは前章で用意した result_df である。

5－1　基本条件の確認

ここでは、各基本条件の確認に使うコードを一覧で並べる。検証結果の表示は煩雑になるため省略する。

```
# 各クラスの人数は 42 名以上 43 名以下にできたか。
result_df.groupby('割当クラス'\
  )['受験番号'].count()
# 各クラスの男女比は均等にできたか。
result_df.groupby(['割当クラス', '性別']\
  )['受験番号'].count()
# 各クラスの推薦人数は均等にできたか。
result_df.groupby(['割当クラス', '推薦']\
  )['受験番号'].count()
# 各クラスの芸術選択者数の確認。
result_df.groupby(['割当クラス', '音楽決定']\
  )['受験番号'].count()
# 各クラスの学科人数は均等にできたか。
result_df.groupby(['割当クラス', \
  '入学学科名称'])['受験番号'].count()
# 同組を希望した学生の確認。
for i, (s1, s2) in enumerate(SSA):
    print('case:',i)
    c1 = S2C[s1]
    c2 = S2C[s2]
    print('s1:{}-{}'.format(s1, c1))
    print('s2:{}-{}'.format(s2, c2))
    print('')
# 同姓者の重複を解消できたか。
for i in range(7):
    N_n = [k for k,v in Second_Name.items() \
    if v == i+2]
    Num_N_n = [( row.姓, row.受験番号) for \
    row in s_df.itertuples() if row.姓 in N_n]
    for k in N_n:
        Ni = [s for name, s in Num_N_n \
```

```
        if name == k]
        NiC = [S2C[s] for s in Ni]
        #print('case', i)
        print(NiC)
```

5－2　入試成績平均の確認

各クラスの入試成績は均等にできたかを確認する。
まずは初期割当クラス平均を確認する。

```
result_df.groupby('初期割当クラス'\
  )['入試成績'].mean()
```

```
初期割当クラス
1-1    593.880952
1-2    607.209302
1-3    610.238095
1-4    608.071429
1-5    596.571429
Name: 入試成績, dtype: float64
```

初期割当クラスでは、平均点から±７点ほどの差がでているようである。
次に割当クラスの平均が平均点±４点の範囲内に収まっていることを確認する。

```
result_df.groupby('割当クラス'\
  )['入試成績'].mean()
```

```
割当クラス
1-1    605.883721
1-2    599.500000
1-3    600.428571
1-4    605.595238
1-5    604.595238
Name: 入試成績, dtype: float64
```

今回は平均点±４点としたが、平均点±２点までは有効時間内に解が得られた。にもかかわらず条件を緩めた理由は、次節で確認する成績分布が大きく崩れたためである。

5－3　成績分布の確認

初期割当クラスの成績分布を視覚化する。

```
# データの可視化モジュール matplotlib の取り込み
import matplotlib.pyplot as plt
# 日本語表示用のフォント設定
plt.rcParams['font.family'] = "Meiryo"
#グラフの出力
fig = plt.figure(figsize=(12,20))
for i, c in enumerate(C):
    cls_df = s_df[s_df['初期割当クラス']==c]
    ax = fig.add_subplot(
    4, 2, i+1, xlabel='score',
    ylabel='num', xlim=(0, 1400)#横軸目盛,
    ylim=(0, 16)#縦軸目盛,
    title='Class:{:s}'.format(c))
    ax.hist(cls_df['入試成績'],
    bins=range(0,1500,40))
```

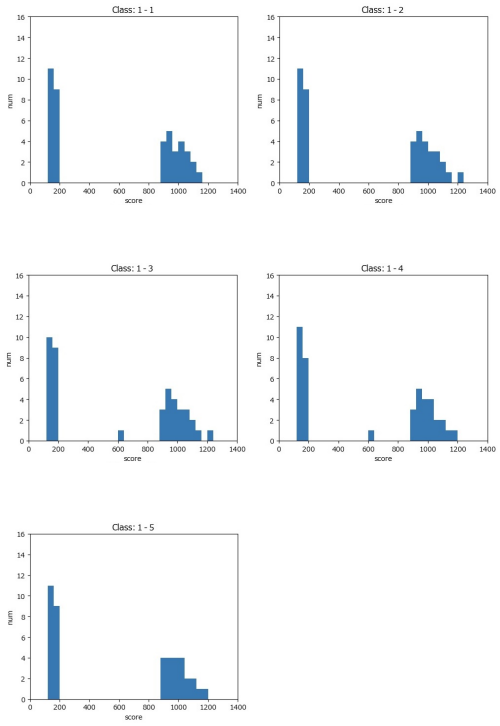

図２　初期配置クラスの成績分布

初期割当クラスは成績順に並べたものであるから、これが成績分布としては最もきれいな形になる。
参考までに、もし４－５（４）入試成績分布の調整の条件を加味せず、クラスの平均点だけで最適化させると次のような分布になる（ここでは紙面の都合上、１－１と１－２だけ掲載する）。

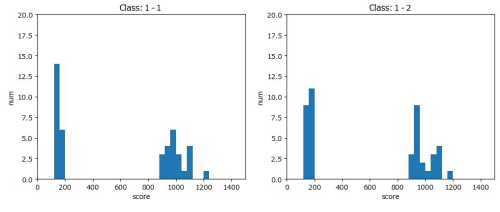

図３　平均点のみで最適化した成績分布

平均点だけの調整では、この右側の山における分布のように中央値付近が大きく凹んだ成績分布になってしまう。これでは、成績によってクラスが２極化してしまう。このような分布を避けるために上記調整を行った次第である。
最後に割当クラスの成績分布を視覚化して確認する。

```
# 割当クラスの入試成績のヒストグラム
fig = plt.figure(figsize=(12,20))
for i, c in enumerate(C):
```

```
cls_df = result_df[\
result_df['割当クラス']==c]
ax = fig.add_subplot(\
4, 2, i+1, xlabel='score',
ylabel='num', xlim=(0, 1400)#横軸目盛,
ylim=(0, 16)#縦軸目盛,
title='Class:{:s}'.format(c))
ax.hist(cls_df['入試成績'],
bins=range(0,1500,40))
```

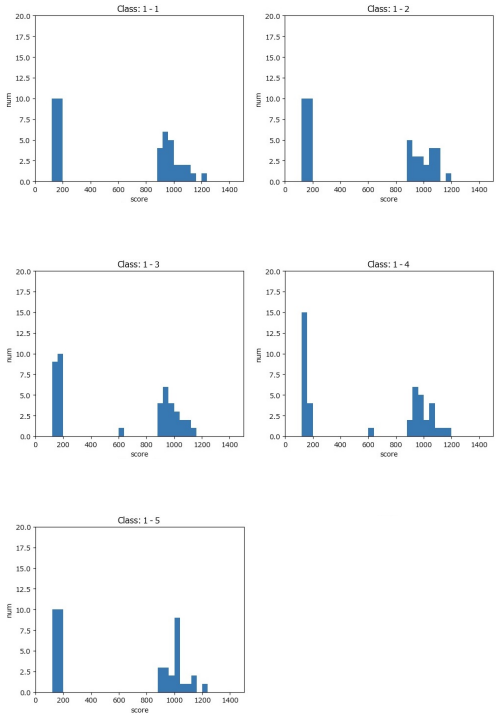

図 4　割当クラスの成績分布

最終的な割当クラス（図 4）は、右側の山における中央値付近が大きく凹むことなく、つまり、クラスの学力が２極化することなく編成することができている。

6.　クラス編成の完成

最後に、配慮の必要な留年生を加えることでクラス編成が完成する。これについては、配慮に必要な条件、芸術の選択状況、そして性別を考慮する。この最後の作業は、条件によってはプログラミングに組み込める可能性があるが、現状では手作業で行う必要がある。

Ｒ５年度のクラス編成は、以上のような手順で編成した結果、次のような人数構成となった。

１－１
男 28、女 15、 計 43（Ｍ 8、 Ｅ 9、 Ｄ 8、 Ｃ 9、 Ｚ 9）
推薦 ● 20、音楽 ○ 21

１－２
男 27、 女 15、 計 42（Ｍ 8、 Ｅ 8、 Ｄ 9、 Ｃ 8、 Ｚ 9）
推薦 ● 20、音楽 ○ 0

１－３
男 27、 女 16、 計 43（Ｍ 8、 Ｅ 9、 Ｄ 8、 Ｃ 9、 Ｚ 9）
推薦 ● 19、音楽 ○ 21

１－４
男 28、女 15、 計 43（Ｍ 9、 Ｅ 9、 Ｄ 9、 Ｃ 8、 Ｚ 8）
推薦 ● 19、音楽 ○ 0

１－５
男 27、 女 15、計 42（Ｍ 8、 Ｅ 9、 Ｄ 8、 Ｃ 9、 Ｚ 8）
推薦 ● 20、音楽 ○ 42

7.　結言

ここまで見てきたように、Ｒ５年度の１年生に対するクラス編成は Python によるプログラミングを活用することで効率化することを目指した。結果として、データの前処理に半日程度要するとしても、その後の編成はプログラムを走らせた後に、配慮の必要な留年生を加えるだけとなり、これまで１週間から 10 日必要だったクラス編成が、１日あるいは２日に短縮された。本業務の大幅な効率化が実現されたことになる。プログラムを利用することで、単調な作業の繰り返しを人の手から解放することができ、留年生に配慮した丁寧な対応を行うための余裕を確保できた。さらに、同姓者の重複数や、成績分布の調整も自動的に行われるため、担当者によるクラス編成の精度のばらつきについても一定以上の均一化が実現できたといえる。本プログラムは今後改良を加え、２年生のクラス編成にも応用されることが期待される。

今なら ChatGPT によってコードを作成させることも可能であるし、誰でも事務的な仕事にプログラミングを活用できるようになると思われる。今回のように Python をはじめとしたプログラミングを活用することで、教員の事務作業の軽減が実現され、ひいては創造的な活動に使える時間の増加による教育の質的向上につながることを期待する。

参考文献

[1] 国立高等専門学校機構ホーム > 男女共同参画推進>各種情報 (https://www.kosen-k.go.jp/gender/information/)
[2] 岩永二郎・石原響太・西村直樹・田中一樹 (2022)『Python ではじめる数理最適化　ケーススタディでモデリングのスキルを身につけよう』オーム社