レポートの体裁例と作成にあたっての注意

ページ番号

タイトル

科目名〇〇〇〇

レポート〇〇

学籍番号　〇〇〇〇

必ず記名する

氏　　名　〇〇〇〇

問題 1.1　（1）データを Python を用いて計算した。

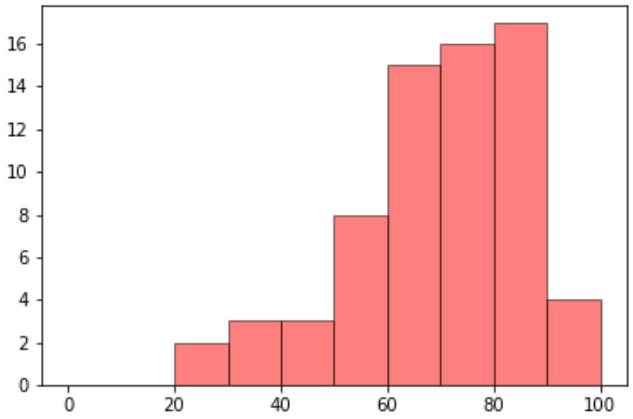
答えだけではなく、計算の根拠などを示す。

大きさ	68
最大値	99.00
最小値	20.00
範囲	79.00
中央値	72.50
平均値	68.56
標準偏差	16.75

Python や Excel を用いて計算すれば、計算結果をコピペして整形するだけで解答になる。コードや計算シートを添付する必要はない。

（2）手計算によって、各階級の度数を数え上げた。

階級	階級値	度数
0 以上 10 未満	5	0
10 以上 20 未満	15	0
20 以上 30 未満	25	2
30 以上 40 未満	35	3
40 以上 50 未満	45	3
50 以上 60 未満	55	8
60 以上 70 未満	65	15
70 以上 80 未満	75	16
80 以上 90 未満	85	17
90 以上 100 以下	95	4



Python や Excel を用いて描画すれば、画像ファイルをコピペするだけでよい。

（3）度数分布表を用いて計算する。

x	f	xf	x^2f
5	0	0	0
15	0	0	0
25	2	50	1250
35	3	105	3675
45	3	135	6075
55	8	440	24200
65	15	975	63375
75	16	1200	90000
85	17	1445	122825
95	4	380	36100
合計	68	4730	347500

平均値：

$$\mu = \frac{1}{n}\sum x_i f_i = \frac{4730}{68} = 69.56$$

答えだけではなく、計算の根拠などを示す。

分散：まず、観測値の 2 乗の平均値

$$\frac{1}{n}\sum x_i^2 f_i = \frac{347500}{68} = 5110.2941$$

を求めて、分散公式を適用すると，

$$\sigma^2 = \frac{1}{n}\sum x_i^2 f_i - \mu^2 = 5110.2941 - 69.5588^2 = 271.86$$

標準偏差：

$$\sigma = \sqrt{271.86} = 16.49$$

コメント：生データと度数データから計算した平均値と標準偏差は一致しない：

	生データ	度数データ
平均値	68.56	69.56
標準偏差	16.75	16.49

発展的な考察：生データから計算された平均値と度数データから計算した平均値は一般に一致しないので、そのずれがどのくらいあるかは重要である。実際、次の計算。。。。。によって、ずれは。。。。。以下にあることが分かる。

気づいたことや発展的な考察などを自分の（自分にしか書けない）言葉で記述する。ただし、見てすぐわかることだけを書いても評価されない（たとえば、本例のコメントだけでは不満足）。また、自分で関連課題を見つけて考察を追記すると評価が高まる。

【レポート作成・提出にあたって】

（1）レポートの原稿をどのような形式で準備するかは自由。WORD 等を利用して準備してもよいし、手書きでも構わない。また，それらの混合でもよい。

（2）提出にあたって、**レポートは 1 個の PDF ファイル**にまとめること。したがって，手書き原稿や切り貼りした原稿は、適切な解像度でスキャンして最終的に PDF に変換する。

（3）WORD, EXCEL, PowerPoint のファイルや JPEG などの画像ファイルは採点対象としないので提出しないこと。

（4）提出レポートの 1 ページ目には記名（学籍番号と氏名）し、レポートが複数ページからなる場合はページ番号を振る。

（5）提出する PDF ファイルは、**「学籍番号_氏名.pdf」の形式のファイル名**を付ける。各自の整理のため付加情報が必要であれば、「学籍番号_氏名_〇〇.pdf」のようにせよ。

（6）提出先：Google Classroom にアップロード（投稿）する。

【当たり前の注意】

（1）講義資料や教科書以外に参考にした書籍や資料があれば明記する。出典を明らかにしない引用は著作権侵害であり不正行為にあたる。もちろん、様々な関連資料をあたって勉強することは大いに推奨される。

（2）他人のレポートの全部または一部のコピー（盗用、剽窃）は不正行為であり、猶予せず厳しく対処する。

【問題1.1 をPython を用いて解答するためのヒント】

各ステップで何ができているかを確認しながら進めている（その方が参考になるから） したがって、解答だけのためには無駄なステップが多い

【数値計算に関する注意】

(1) 計算結果はかなり長い小数表示（14桁くらい）で示されるが、計算機内部では２進数で計算しているので、桁の先の方は正しくない。

(2) 計算の途中では構わないが、最終的に示すべき統計量（平均値や標準偏差など）では、もとのデータの精度を考えて合理的な表示桁数にすること。

(3) 計算途中でも無駄に長い小数表示を初めから避けたければ、冒頭で桁数を宣言する。コマンドは「%precision 4」（小数4桁まで表示）

(4) Python における概数表示は、「厳密な四捨五入」ではなく「偶数丸め」が適用される。

In [1]:

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from pandas.plotting import scatter_matrix
# %precision 4
# 無駄に詳しい小数表示（14桁）を初めから避けて、小数第4位までの表示にするための宣言
# このような概数表示では、「厳密な四捨五入」ではなく、「偶数丸め」が実行される
```

【Google Colaboratory】

ローカルファイルにアクセスできない（ちょっと不便）ので、まず、Google Drive をマウントする。

In [2]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
mount("/content/drive", force_remount=True).
```

In [3]:

```
Data=pd.read_csv('/content/drive/My Drive/Classroom/[CB33301] 数理統計学 2022/ProbData01_1.csv') # 読
```

In [4]:

```
Data.head()  # 先頭の5行だけ表示
```

Out[4]:

	82
0	83
1	74
2	54
3	88
4	65

In [5]:

```
# 元データの csv ファイルは1行目からデータが入っていることがわかる。
# カラム名を付けて、読み込み直す
Data=pd.read_csv('/content/drive/My Drive/Classroom/[CB33301] 数理統計学 2022/ProbData01_1.csv',
                 names=['Score'])
Data.head()
```

Out[5]:

	Score
0	82
1	83
2	74
3	54
4	88

【Jupyter Notebook】

自分のPCで閉じた環境で作業できる。 ローカルファイルへのアクセスは直接できる。

In [6]:

```
# Data=pd.read_csv("D:2022_MathStatProblems/Data/ProbData01_1.csv") # 読み出したいファイルのパス
```

In [7]:

```
# 元データの csv ファイルは1行目からデータが入っていることがわかる。
# カラム名を付けて、読み込み直す
# Data=pd.read_csv("D:2022_MathStatProblems/Data/ProbData01_1.csv",
#                  names=['Score'])
# Data.head()
```

【基本的な統計量を求めよう】

In [8]:

```
# あとで使いやすいように変数名を付けておく
size=len(Data['Score'])         # サイズ
mean=np.mean(Data['Score'])     # 平均値
var=np.var(Data['Score'])       # 分散　デフォルトで標本分散が出る
std=np.std(Data['Score'])       # 標準偏差
minimum=np.min(Data['Score'])   # 最小値 min はコマンド名なので避けた
maximum=np.max(Data['Score'])   # 最大値 max はコマンド名なので避けた
med=np.median(Data['Score'])    # 中央値
```

In [9]:

```
# 基本的な統計量 DataFrame で出力
StatSummary=pd.DataFrame([
              ['サイズ', size],
              ['平均値', mean],
              ['分散', var],
              ['標準偏差', std],
              ['最小値', minimum],
              ['最大値', maximum],
              ['中央値', med],
              ['範囲', maximum-minimum],
             ])
StatSummary
```

Out[9]:

	0	1
0	サイズ	68.000000
1	平均値	68.558824
2	分散	280.628893
3	標準偏差	16.751982
4	最小値	20.000000
5	最大値	99.000000
6	中央値	72.500000
7	範囲	79.000000

In [10]:

```
# 気に入らなければ整形しよう （１）カラム名の変更
StatSummary=StatSummary.rename(columns={0:'統計量'})
StatSummary=StatSummary.rename(columns={1:'Score'})
StatSummary
```

Out[10]:

	統計量	Score
0	サイズ	68.000000
1	平均値	68.558824
2	分散	280.628893
3	標準偏差	16.751982
4	最小値	20.000000
5	最大値	99.000000
6	中央値	72.500000
7	範囲	79.000000

In [11]:

```
# 気に入らなければ整形しよう （２）無駄に長い小数表示を変更
pd.set_option('precision', 2)      # pandas において表示桁数の制御：小数第2位まで表示する
StatSummary
```

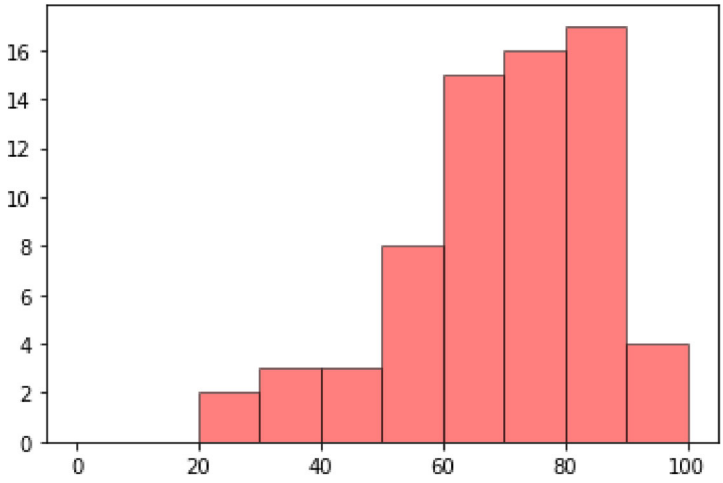
Out[11]:

	統計量	Score
0	サイズ	68.00
1	平均値	68.56
2	分散	280.63
3	標準偏差	16.75
4	最小値	20.00
5	最大値	99.00
6	中央値	72.50
7	範囲	79.00

【ヒストグラム】

In [12]:

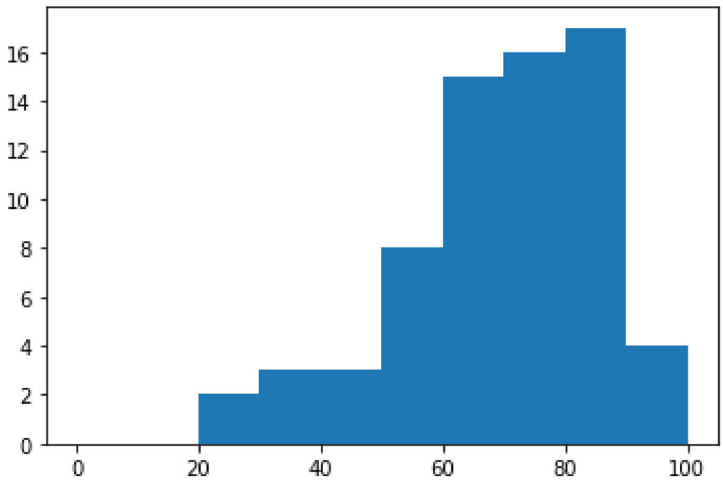
```
# fig = plt.figure(figsize=(6,4)) # 図のサイズを指定。デフォルトは (6,4)
plt.hist(Data['Score'],
         range=(0,100),         # 幅を指定
         bins=10,               # 階級数を指定
         color='red',           # 色を指定
         alpha=0.5,             # 色の濃さ (0～1)
         ec='k')                # 棒に枠線つける (k=black)
# 画像ファイルの保存 (png, jpeg, eps などが可能)
# なお、画像は画面から直接コピーして、Word などに貼り付けできる
plt.savefig('ProbData01_1_Hist.png')  # 画像ファイル(png)として保存
```



【度数分布表】

In [13]:

```
# 度数データをヒストグラムから抽出
Y,X,_= plt.hist(Data['Score'], range=(0,100), bins=10)
```



In [14]:

```
# ヒストグラムから抽出した度数データを見ておく
# 階級の刻み
X
```

Out[14]:

```
array([  0.,   10.,   20.,   30.,   40.,   50.,   60.,   70.,   80.,   90., 100.])
```

In [15]:

```
# 度数
Y
```

Out[15]:

```
array([ 0.,  0.,  2.,  3.,  3.,  8., 15., 16., 17.,  4.])
```

In [16]:

```
# 階級
Jclass=[f'{X[i]}以上{X[i+1]}未満' for i in range(0,10)]  # f-string によって変数を文字列に埋め込み
Jclass
```

Out[16]:

```
['0.0以上10.0未満',
 '10.0以上20.0未満',
 '20.0以上30.0未満',
 '30.0以上40.0未満',
 '40.0以上50.0未満',
 '50.0以上60.0未満',
 '60.0以上70.0未満',
 '70.0以上80.0未満',
 '80.0以上90.0未満',
 '90.0以上100.0未満']
```

In [17]:

```
# 浮動小数点表示が気に入らないので整数に直してもう一度
Jclass=[f'{int(X[i])}以上{int(X[i+1])}未満' for i in range(0,10)]  # f-string によって変数を文字列
Jclass
```

Out[17]:

```
['0以上10未満',
 '10以上20未満',
 '20以上30未満',
 '30以上40未満',
 '40以上50未満',
 '50以上60未満',
 '60以上70未満',
 '70以上80未満',
 '80以上90未満',
 '90以上100未満']
```

In [18]:

```
# 階級値
Jclassmark=[X[i]+5 for i in range(0,10)]
Jclassmark
```

Out[18]:

```
[5.0, 15.0, 25.0, 35.0, 45.0, 55.0, 65.0, 75.0, 85.0, 95.0]
```

In [19]:

```
# こちらも浮動小数点表示を整数に直す
Jclassmark=[int(X[i])+5 for i in range(0,10)]
Jclassmark
```

Out[19]:

```
[5, 15, 25, 35, 45, 55, 65, 75, 85, 95]
```

In [20]:

```
# 度数分布表を DataFrame で表示
FrequencyTable=pd.DataFrame({'階級値': Jclassmark, '度数':Y},
                            index=pd.Index(Jclass, name='階級'))
FrequencyTable
```

Out[20]:

階級	階級値	度数
0以上10未満	5	0.0
10以上20未満	15	0.0
20以上30未満	25	2.0
30以上40未満	35	3.0
40以上50未満	45	3.0
50以上60未満	55	8.0
60以上70未満	65	15.0
70以上80未満	75	16.0
80以上90未満	85	17.0
90以上100未満	95	4.0

In [21]:

```
# 度数が浮動小数点数なので整数値に直す
Y=[int(Y[i]) for i in range(0,10)]
Y
```

Out[21]:

```
[0, 0, 2, 3, 3, 8, 15, 16, 17, 4]
```

In [22]:

```
# 度数分布表を再度作る
FrequencyTable=pd.DataFrame({'階級値': Jclassmark, '度数':Y},
                            index=pd.Index(Jclass, name='階級'))
FrequencyTable
```

Out[22]:

階級	階級値	度数
0以上10未満	5	0
10以上20未満	15	0
20以上30未満	25	2
30以上40未満	35	3
40以上50未満	45	3
50以上60未満	55	8
60以上70未満	65	15
70以上80未満	75	16
80以上90未満	85	17
90以上100未満	95	4

In [23]:

```
# Pandas 度数分布表の最後の階級は ○○以上○○以下をカウントしている。
# 度数分布表の表示は正しくないので修正。
FTable=FrequencyTable.rename(index={'90以上100未満':'90以上100以下'})
FTable
```

Out[23]:

階級	階級値	度数
0以上10未満	5	0
10以上20未満	15	0
20以上30未満	25	2
30以上40未満	35	3
40以上50未満	45	3
50以上60未満	55	8
60以上70未満	65	15
70以上80未満	75	16
80以上90未満	85	17
90以上100以下	95	4

【度数分布表をもとに統計量を計算する】

In [24]:

```
# 度数分布表をもとに統計量を計算
FTable=FTable.rename(columns={'階級値':'x'})  # 変量を x, f のように簡単な英文字に変更
FTable=FTable.rename(columns={'度数':'f'})
FTable
```

Out[24]:

階級	x	f
0以上10未満	5	0
10以上20未満	15	0
20以上30未満	25	2
30以上40未満	35	3
40以上50未満	45	3
50以上60未満	55	8
60以上70未満	65	15
70以上80未満	75	16
80以上90未満	85	17
90以上100以下	95	4

In [25]:

```
FTable['xf']=FTable['x']*FTable['f']
FTable['x^2f']=FTable['x']**2*FTable['f']
FTable
```

Out[25]:

階級	x	f	xf	x^2f
0以上10未満	5	0	0	0
10以上20未満	15	0	0	0
20以上30未満	25	2	50	1250
30以上40未満	35	3	105	3675
40以上50未満	45	3	135	6075
50以上60未満	55	8	440	24200
60以上70未満	65	15	975	63375
70以上80未満	75	16	1200	90000
80以上90未満	85	17	1445	122825
90以上100以下	95	4	380	36100

In [26]:

```
# 和の計算
f_sum=sum(FTable['f'])          # 度数の和＝データの大きさ
xf_sum=sum(FTable['xf'])
xxf_sum=sum(FTable['x^2f'])
print(f_sum,xf_sum,xxf_sum)
```

```
68 4730 347500
```

In [27]:

```
# 度数データから平均値の計算
FT_mean=xf_sum/f_sum
FT_mean
```

Out[27]:

```
69.55882352941177
```

In [28]:

```
# 生データと度数データによる平均値の比較
mean,FT_mean
```

Out[28]:

```
(68.55882352941177, 69.55882352941177)
```

In [29]:

```
# 度数データから分散の計算
FT_square_mean=sum(FTable['x^2f'])/f_sum
FT_var=FT_square_mean-FT_mean**2
FT_square_mean, FT_mean**2, FT_var
```

Out[29]:

```
(5110.294117647059, 4838.429930795848, 271.86418685121043)
```

In [30]:

```
# 生データと度数データによる分散の比較
var, FT_var
```

Out[30]:

```
(280.628892733564, 271.86418685121043)
```

In [31]:

```
# 度数データから標準偏差の計算
FT_std=np.sqrt(FT_var)    # 平方根の計算
```

In [32]:

```
# 生データと度数データによる標準偏差の比較
std, FT_std
```

Out[32]:

```
(16.75198175540924, 16.48830454750307)
```

In [33]:

```
# 比較のまとめ
print('生データの平均値:', mean, '度数データの平均値:', FT_mean)
print('生データの分散:', var, '度数データの分散:', FT_var)
print('生データの標準偏差:', std, '度数データの標準偏差:', FT_std)
```

```
生データの平均値: 68.55882352941177 度数データの平均値: 69.55882352941177
生データの分散: 280.628892733564 度数データの分散: 271.86418685121043
生データの標準偏差: 16.75198175540924 度数データの標準偏差: 16.48830454750307
```

In [34]:

```
# 解答として、無駄に長い小数表示は不可
# 四捨五入で小数第2位まで求める
print('生データの平均値:', np.round(mean,2), '度数データの平均値:', np.round(FT_mean,2))
print('生データの分散:', np.round(var,2), '度数データの分散:', np.round(FT_var,2))
print('生データの標準偏差:', np.round(std,2), '度数データの標準偏差:', np.round(FT_std,2))
```

```
生データの平均値: 68.56 度数データの平均値: 69.56
生データの分散: 280.63 度数データの分散: 271.86
生データの標準偏差: 16.75 度数データの標準偏差: 16.49
```

In [34]: