

# 新人課題

津嶋 佑旗

2019 年 4 月 17 日

## 1 配列情報処理

### 1.1 ヒト 21 番染色体の先頭コンティグの DNA 配列を秋山研サーバから取得し、A/T/G/C の各塩基数を出力せよ。

以下のように実行する。実行環境は Python2 で、ghostgw 上での動作を確認している。

```
1 python 1_1.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta
```

ソースコード 1 実行方法

プログラム本体は以下の通りである。

```
1 #2019/04/11
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4 #For Python 2.7
5
6 import sys
7
8 #file open
9 args = sys.argv
10 if len(args) != 2:
11     sys.stderr.write('Usage: python 1_1.py /NT_113952.1.fasta\n')
12     sys.exit()
13 path = args[1]
14 try:
15     file = open(path)
16 except:
17     sys.stderr.write('Error in opening file.\n')
18     sys.exit()
19 lines = file.readlines()
20 length = len(lines)
21
22 #counter(num of A/T/G/C)
23 noA = 0
24 noT = 0
25 noG = 0
26 noC = 0
27 #Process with each line
28 #Skip Header
29 for l in range(1, length):
30     noA = noA + lines[l].count('A')
31     noT = noT + lines[l].count('T')
32     noG = noG + lines[l].count('G')
33     noC = noC + lines[l].count('C')
34 print("A:" + str(noA))
35 print("T:" + str(noT))
36 print("G:" + str(noG))
37 print("C:" + str(noC))
38 file.close()
```

ソースコード 2 1\_1.py

実行した結果が以下の通りである。

```
1 A:59200
2 T:56195
3 G:34714
4 C:34246
```

ソースコード 3 実行結果

## 1.2 NT\_113952.1.fasta の逆相補鎖を生成するプログラムを作成せよ.

以下のように実行する. 実行環境は Python2 で, ghostgw 上での動作を確認している.

```
1 python 1_2.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta
```

ソースコード 4 実行方法

プログラム本体は以下の通りである.

```
1 #2019/04/05
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4 #For Python 2.7
5
6 import sys
7 from rev import rev
8 from combine import combine
9
10 #file open
11 args = sys.argv
12 if len(args) != 2:
13     sys.stderr.write('Usage: python 1_2.py ./NT_113952.1.fasta\n')
14     sys.exit()
15 path = args[1]
16 try:
17     file = open(path)
18 except:
19     sys.stderr.write('Error in opening file.\n')
20     sys.exit()
21 lines = file.readlines()
22
23 #Skip Header
24 lines.pop(0)
25 text = combine(lines)
26 #Reverse
27 reverse = rev(text)
28 print(reverse)
29
30 file.close()
```

ソースコード 5 1\_2.py

```
1 #2019/04/05
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4
5 def rev(text):
6     ans = ''
7     for char in text:
8         if char == 'A':
9             ans = 'T' + ans
10        elif char == 'T':
11            ans = 'A' + ans
12        elif char == 'G':
13            ans = 'C' + ans
14        elif char == 'C':
15            ans = 'G' + ans
16    return ans
```

ソースコード 6 rev.py

```
1 #2019/04/05
2 #Yuki Tsushima
```

```

3 #tsushima@bi.c.titech.ac.jp
4
5 def combine(file):
6     ans = ''
7     for line in file:
8         ans = ans + line.strip()
9     return ans

```

ソースコード 7 combine.py

実行結果は result\_1.2.txt として別添する.

### 1.3 ウィンドウ幅 $w$ , ステップ幅 $s$ で, ウィンドウ内の GC 含量を出力するプログラムを作成し, NT\_113952.1.fasta に $w = 1000$ , $s = 300$ で適用した結果を gnuplot 等 Excel 以外のツールでプロットせよ.

以下のように実行する. 実行環境は Python2 で, ghostgw 上での動作を確認している.

```

1 python 1_3.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta 1000 300 > data.txt

```

ソースコード 8 実行方法

プログラム本体は以下の通りである. また, 7 も使用する.

```

1 #2019/04/11
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4 #For Python 2.7
5
6 import sys
7 from combine import combine
8
9 #file open
10 args = sys.argv
11 if len(args) != 4:
12     sys.stderr.write('Usage: python 1_3.py /NT_113952.1.fasta 1000 300\n')
13     sys.exit()
14 path = args[1]
15 try:
16     file = open(path)
17 except:
18     sys.stderr.write('Error in opening file.\n')
19     sys.exit()
20 lines = file.readlines()
21
22 #Skip Header
23 lines.pop(0)
24 text = combine(lines)
25 length = len(text)
26
27 try:
28     w = int(args[2])
29     s = int(args[3])
30 except ValueError as e:
31     sys.stderr.write('Incorrect w or s in command line args\n')
32     sys.exit()
33
34 GC = []
35 bottom = 0
36 while bottom < length:
37     part = text[bottom:bottom+w]
38     if len(part) == 0:
39         break
40     GC.append((part.count('C') + part.count('G'))/float(len(part)))
41     bottom = bottom + s
42 #Export for gnuplot
43 for i in range(len(GC)):
44     print(str(i*300)+'_'+str(GC[i]))
45
46 file.close()

```

ソースコード 9 1.3.py

実行結果を gnuplot で描画した結果以下ようになった。

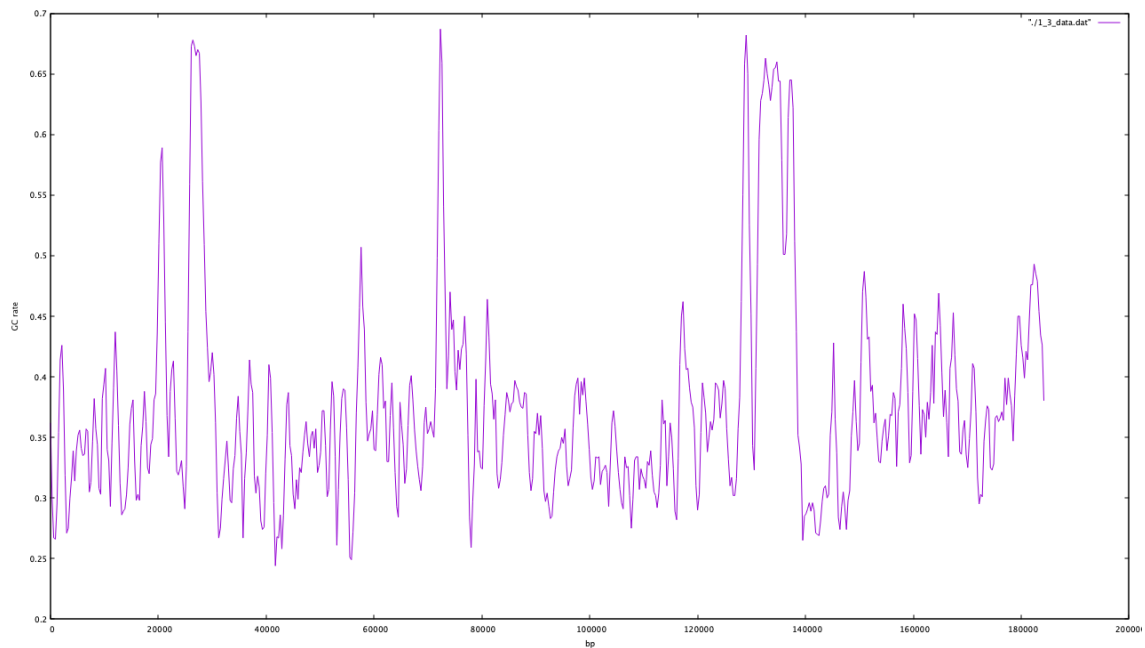


図1 グラフ

#### 1.4 引数で与えられた部分配列を検索し、何文字目に現れたかを表示するプログラムを作成せよ (逆相補鎖上も検索すること). 部分配列を GAATTC および ATG とし、NT\_113952.1.fasta に適用せよ.

以下のように実行する. 実行環境は Python2 で, ghostgw 上での動作を確認している.

```
1 python 1_4.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta GAATTC
2 python 1_4.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta ATG
```

ソースコード 10 実行方法

プログラム本体は以下の通りである. また, 6 と 7 も使用する.

```
1 #2019/04/11
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4 #For Python 2.7
5
6 import sys
7 from combine import combine
8 from rev import rev
9 from findall import findall
10
11 #file open
12 args = sys.argv
13 if len(args) != 3:
14     sys.stderr.write('Usage: python 1_4.py ./NT_113952.1.fasta GAATTC\n')
15     sys.exit()
16 path = args[1]
17 try:
18     file = open(path)
19 except:
20     sys.stderr.write('Error in opening file.\n')
21     sys.exit()
22 lines = file.readlines()
23
```

```

24 lines.pop(0)
25 text = combine(lines)
26 reverse = rev(text)
27 index1 = findall(text, args[2])
28 index2 = findall(reverse, args[2])
29 print(args[2]+'_in_STRING_is_below:')
30 print(index1)
31 print(args[2]+'_in_REVERSE_STRING_is_below:')
32 print(index2)
33 file.close()

```

ソースコード 11 1\_4.py

```

1 #2019/04/08
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4
5 def findall(text, query):
6     index = []
7     data = -1
8     while True:
9         data = text.find(query, data+1)
10        if data == -1:
11            break
12        index.append(data)
13    return index

```

ソースコード 12 findall.py

実行した結果が以下の通りである。ただし、ATG の結果については極端に長いいため result\_1\_4\_ATG.txt として別添する。

```

1 GAATTC in STRING is below:
2 [395, 4319, 5056, 5334, 10567, 14092, 14296, 15029, 19284, 22535, 25293, 25897, 30824, 34194,
   34625, 37333, 56600, 57464, 58202, 60553, 65241, 66725, 79939, 80725, 82012, 86322, 92914,
   94554, 96877, 97628, 99894, 102214, 103133, 120291, 121414, 121612, 124893, 124901, 125389,
   151452, 151874, 156137, 158738, 166310, 175826, 177983, 180004, 182341]
3 GAATTC in REVERSE STRING is below:
4 [2008, 4345, 6366, 8523, 18039, 25611, 28212, 32475, 32897, 58960, 59448, 59456, 62737, 62935,
   64058, 81216, 82135, 84455, 86721, 87472, 89795, 91435, 98027, 102337, 103624, 104410,
   117624, 119108, 123796, 126147, 126885, 127749, 147016, 149724, 150155, 153525, 158452,
   159056, 161814, 165065, 169320, 170053, 170257, 173782, 179015, 179293, 180030, 183954]

```

ソースコード 13 実行結果 (GAATTC)

## 1.5 NT\_113952.1.fasta を 6 つの読み枠でアミノ酸配列に変換せよ。Stop コドンはアンダースコアで表示すること。

以下のように実行する。実行環境は Python2 で、ghostgw 上での動作を確認している。

```
python 1_5.py /mnt/fs/ohue/newcomer/NT_113952.1.fasta
```

ソースコード 14 実行方法

プログラム本体は以下の通りである。また、6 と 7 も使用する。

```

1 #2019/04/11
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4 #For Python 2.7
5
6 import sys
7 from combine import combine
8 from rev import rev
9 from decode import decode
10
11 #file open
12 args = sys.argv
13 if len(args) != 2:
14     sys.stderr.write('Usage: python 1_5.py ./NT_113952.1.fasta\n')

```

```

15     sys.exit()
16 path = args[1]
17 try:
18     file = open(path)
19 except:
20     sys.stderr.write('Error in opening file.\n')
21     sys.exit()
22 lines = file.readlines()
23
24 lines.pop(0)
25 text = combine(lines)
26 reverse = rev(text)
27 print('PEPTIDE_1')
28 print(decode(text,0))
29 print('PEPTIDE_2')
30 print(decode(text,1))
31 print('PEPTIDE_3')
32 print(decode(text,2))
33 print('PEPTIDE_4')
34 print(decode(reverse,0))
35 print('PEPTIDE_5')
36 print(decode(reverse,1))
37 print('PEPTIDE_6')
38 print(decode(reverse,2))
39 file.close()

```

ソースコード 15 1-5.py

```

1  #2019/04/08
2  #Yuki Tsushima
3  #tsushima@bi.c.titech.ac.jp
4
5  import sys
6
7  def decode(text, offset):
8      length = len(text)
9      #if too short
10     if length < offset + 3:
11         return ''
12     peptide = ''
13     CODON = {'TTT':'Phe', 'TTC':'Phe', 'TTA':'Leu', 'TTG':'Leu',\
14             'TCT':'Ser', 'TCC':'Ser', 'TCA':'Ser', 'TCG':'Ser',\
15             'TAT':'Tyr', 'TAC':'Tyr', 'TAA':'_', 'TAG':'_',\
16             'TGT':'Cys', 'TGC':'Cys', 'TGA':'_', 'TGG':'Trp',\
17             'CTT':'Leu', 'CTC':'Leu', 'CTA':'Leu', 'CTG':'Leu',\
18             'CCT':'Pro', 'CCC':'Pro', 'CCA':'Pro', 'CCG':'Pro',\
19             'CAT':'His', 'CAC':'His', 'CAA':'Gln', 'CAG':'Gln',\
20             'CGT':'Arg', 'CGC':'Arg', 'CGA':'Arg', 'CGG':'Arg',\
21             'ATT':'Ile', 'ATC':'Ile', 'ATA':'Ile', 'ATG':'Met',\
22             'ACT':'Thr', 'ACC':'Thr', 'ACA':'Thr', 'ACG':'Thr',\
23             'AAT':'Asn', 'AAC':'Asn', 'AAA':'Lys', 'AAG':'Lys',\
24             'AGT':'Ser', 'AGC':'Ser', 'AGA':'Arg', 'AGG':'Arg',\
25             'GTT':'Val', 'GTC':'Val', 'GTA':'Val', 'GTG':'Val',\
26             'GCT':'Ala', 'GCC':'Ala', 'GCA':'Ala', 'GCG':'Ala',\
27             'GAT':'Asp', 'GAC':'Asp', 'GAA':'Glu', 'GAG':'Glu',\
28             'GGT':'Gly', 'GGC':'Gly', 'GGA':'Gly', 'GGG':'Gly'}
29     start = offset
30     end = start + 3
31     try:
32         #Skip to Met
33         while end <= length:
34             if CODON[text[start:end]] == 'Met':
35                 break
36             start = end
37             end = start + 3
38         #Decode
39         while end <= length:
40             peptide = peptide + '_' + CODON[text[start:end]]
41             if peptide[len(peptide)-1] == '_':
42                 break
43             start = end
44             end = start + 3
45         return peptide.strip()
46     except (KeyError):
47         sys.stderr.write('Unknown CODON Error\n')
48     return ''

```

ソースコード 16 decode.py

実行した結果が以下の通りである。

```
1 PEPTIDE 1
2 Met Ile Val Met Asn Ser Asn Cys Cys Leu Cys Arg Pro Thr Arg Phe Leu Thr Ser Leu Ser Tyr His Phe
3   Leu Leu Ser Tyr Leu Leu Ser Lys Cys Ile Gln Met Lys Gly Cys Gly Glu Cys _
4 PEPTIDE 2
5 Met Pro Arg Glu Ile Ser Arg Ser Ser Val Pro Cys _
6 PEPTIDE 3
7 Met Pro _
8 PEPTIDE 4
9 Met Leu Arg Thr Leu Leu Leu Ile Val _
10 PEPTIDE 5
11 Met Gln Asn Lys Phe Ile Arg His _
12 PEPTIDE 6
13 Met Gly Arg Lys Asp Lys Ala Ala Ile _
```

ソースコード 17 実行結果

## 2 タンパク質構造情報処理

2.1 PyMOL ソフトウェアを利用し、ヒトのヘモグロビンの構造をチェーンごとに異なる色で表示し、4 量体であることを確認せよ。また、A チェインだけを表示し、タンパク質鎖を cartoon 表示して 2 次構造に従って色付けし、結合するヘムを stick 表示して原子ごとに色分けせよ。

チェーンごとに色分けしたヘモグロビンが画像 2 である。



図2 ヘモグロビン

また、A チェインとヘムを表示したものが画像 3 である。



図3 A チェインとヘム

## 2.2 PDB ファイル名とチェーン名を引数にして、その回転半径を計算するプログラムを作成せよ。

以下のプログラムを作成した。実行環境は Anaconda3 の Python 2.7 環境である。

```

1  #2019/04/11
2  #Yuki Tsushima
3  #tsushima@bi.c.titech.ac.jp
4  #For Python 2.7
5
6  import math
7  import sys
8
9  #file open
10 args = sys.argv
11 if len(args) != 3:
12     sys.stderr.write('Usage: python 2_7.py 1BUW.pdb A\n')
13     sys.exit()
14 path = args[1]
15 try:
16     file = open(path)
17 except:
18     sys.stderr.write('Error in opening file.\n')
19     sys.exit()
20 lines = file.readlines()
21
22 WEIGHT = \
23 { 'H':1.008,\
24   'LI':6.941, 'BE':9.012, 'B':10.81, 'C':12.01, 'N':14.01, 'O':16.00, 'F':19.00,\
25   'NA':22.99, 'MG':24.31, 'AL':26.98, 'SI':28.09, 'P':30.97, 'S':32.07, 'CL':35.45,}
26 total_weight = 0
27 x_weight = 0
28 y_weight = 0
29 z_weight = 0
30 atom_data = []
31 for line in lines:
32     if line[0:6] == 'ATOM' and line[21] == args[2]:
33         #Parse Data
34         atom_number = int(line[6:11])
35         atom_name = line[12:16].strip()
36         #identifier = line[16].strip()
37         #res_name = line[17:20].strip()
38         #chain = line[21].strip()
39         #res_number = int(line[22:26])
40         #res_code = line[26].strip()
41         x = float(line[30:38])

```



```

42     y = float(line[38:46])
43     z = float(line[46:54])
44     #occupy = float(line[54:60])
45     #tmp = line[60:66].strip()
46     elem_symbol = line[76:78].strip()
47     #charge = line[78:80].strip()
48
49     #Calculate Center of Gravity
50     total_weight = total_weight + WEIGHT[elem_symbol]
51     x_weight = x_weight + x * WEIGHT[elem_symbol]
52     y_weight = y_weight + y * WEIGHT[elem_symbol]
53     z_weight = z_weight + z * WEIGHT[elem_symbol]
54
55     #Register Atom Data
56     atom_data.append([x, y, z, atom_number, atom_name, elem_symbol])
57 #IF no Data
58 if total_weight == 0:
59     sys.stderr.write('Chain'+args[2]+' does not exist.\n')
60     sys.exit()
61
62 #Calculate Center of Gravity
63 x_balance = x_weight / total_weight
64 y_balance = y_weight / total_weight
65 z_balance = z_weight / total_weight
66
67 #Calculate circle r
68 r = 0
69 for data in atom_data:
70     r = r + math.sqrt((data[0]-x_balance)**2+(data[1]-y_balance)**2+(data[2]-z_balance)**2)
71 r = r / len(atom_data)
72 print('Center: ',str(x_balance)+', ',str(y_balance)+', ',str(z_balance)+')')
73 print('Radius: ',str(r))
74
75 file.close()

```

ソースコード 18 2.7.py

このプログラムを次のように実行する。

```

1 python 2_7.py ./1BUW.pdb A

```

ソースコード 19 実行方法

結果は次の通りである。

```

1 Center: (47.0059686237, 33.4827138428, 35.7021762506)
2 Radius: 14.0781227416

```

ソースコード 20 実行結果

2.3 上記のプログラムの結果を利用し、PyMOL で重心から回転半径の範囲内にある原子を赤で、範囲外にある原子を青で色付けせよ。

## 3 機械学習

3.1 Python の scikit-learn の SVM で ionosphere のデータの 10-fold Cross Validation を実施せよ。予測性能として precision、recall、MCC、ROC 曲線の AUC 値 (AUROC)、F-score を求めよ。カーネルは RBF カーネルとし、パラメータは適宜定めよ。

作成したプログラムを下に示す。Anaconda3 の Python2.7 環境で動作する。

```

1 #2019/04/16
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4 #For Python 2.7
5
6 import sys

```

```

7 from sklearn.svm import SVC
8 from sklearn.metrics import precision_score, recall_score, f1_score, matthews_corrcoef,
   roc_auc_score
9 from load_ionosphere import load_ionosphere
10 from sklearn.model_selection import *
11
12 #TP, TN : True
13 #FN : actually positive, but expected negative
14 #FP : actually negative, but expected positive
15 #Precision : TP / TP + FP
16 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#
   sklearn.metrics.precision_score
17 #Recall : TP / TP + FN
18 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#
   sklearn.metrics.recall_score
19 #F-measure : 2 * Recall * Precision / Recall + Precision
20 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html#sklearn.
   metrics.f1_score
21 #MCC : Matthews Correlation Coefficient
22 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html#
   sklearn.metrics.matthews_corrcoef
23 #AUROC :
24 # https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html#
   sklearn.metrics.roc_auc_score
25
26 svc = SVC(C=1.0, kernel='rbf', gamma=0.2)
27 kf = KFold(n_splits=10, shuffle=True)
28 [target, parameters] = load_ionosphere()
29 counter = 0
30 print('Set\tPreci.\tRecall\tMCC\tAUROC\tF-score')
31 for train_index, test_index in kf.split(parameters, target):
32     counter = counter + 1
33     train_param = []
34     train_target = []
35     test_param = []
36     test_target = []
37     for i in range(0, len(train_index)):
38         train_param.append(parameters[train_index[i]])
39         train_target.append(target[train_index[i]])
40     for i in range(0, len(test_index)):
41         test_param.append(parameters[test_index[i]])
42         test_target.append(target[test_index[i]])
43     sys.stdout.write(str(counter)+'\t')
44     learn = svc.fit(parameters, target)
45     print(str(precision_score(test_target, learn.predict(test_param)))[0:7]+'\\t'\\
46           +str(recall_score(test_target, learn.predict(test_param)))[0:7]+'\\t'\\
47           +str(matthews_corrcoef(test_target, learn.predict(test_param)))[0:7]+'\\t'\\
48           +str(roc_auc_score(test_target, learn.predict(test_param)))[0:7]+'\\t'\\
49           +str(f1_score(test_target, learn.predict(test_param)))[0:7])

```

ソースコード 21 3\_9.py

また、ionosphere.scale の読み込みにあたって、欠番のパラメータを 0 で補完するなどの処理を行うため、読み込み部分は別のプログラムとして分離した。

```

1 #2019/04/16
2 #Yuki Tsushima
3 #tsushima@bi.c.titech.ac.jp
4 #For Python 2.7
5
6 import sys
7
8 def load_ionosphere():
9     #file open
10     try:
11         file = open('./ionosphere.scale')
12     except:
13         sys.stderr.write('Error in opening file.\n')
14         sys.exit()
15
16     lines = file.readlines()
17     target = []
18     parameters = []
19     for line in lines:
20         set = line.split()
21         #target is 1 or 0
22         target.append(int(set.pop(0))+1//2)
23         datas = {}
24         params = []

```

```

25     for data in set:
26         datas[int(data.split(':')[0])] = float(data.split(':')[1])
27     for i in range(0,34):
28         try:
29             params.append(datas[i+1])
30         except KeyError:
31             params.append(0)
32     parameters.append(params)
33 file.close()
34 return [target, parameters]

```

ソースコード 22 load\_ionosphere.py

実行した結果を以下に示す。

```

1 Set Preci. Recall MCC AUROC F-score
2 1 0.92307 1.0 0.87705 0.91666 0.96000
3 2 1.0 0.90909 0.88762 0.95454 0.95238
4 3 1.0 0.96 0.93419 0.98 0.97959
5 4 1.0 1.0 1.0 1.0 1.0
6 5 1.0 0.95238 0.94280 0.97619 0.97560
7 6 0.91666 1.0 0.88070 0.92307 0.95652
8 7 0.95454 1.0 0.94146 0.96428 0.97674
9 8 1.0 1.0 1.0 1.0 1.0
10 9 1.0 1.0 1.0 1.0 1.0
11 10 1.0 1.0 1.0 1.0 1.0

```

ソースコード 23 実行結果

3.2 ionosphere データにおいて、より良い RBF カーネルパラメータ  $\gamma$  とコストパラメータ  $C$  の値を探索せよ。評価方法は 10-fold Cross Validation とし、評価基準は AUROC と F-score の 2 通りを試すこと。