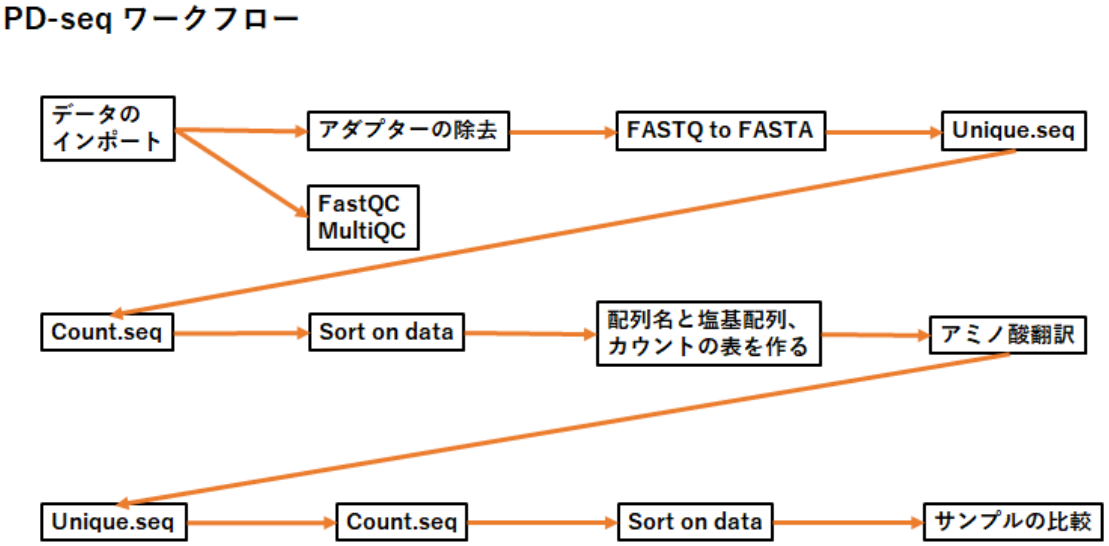
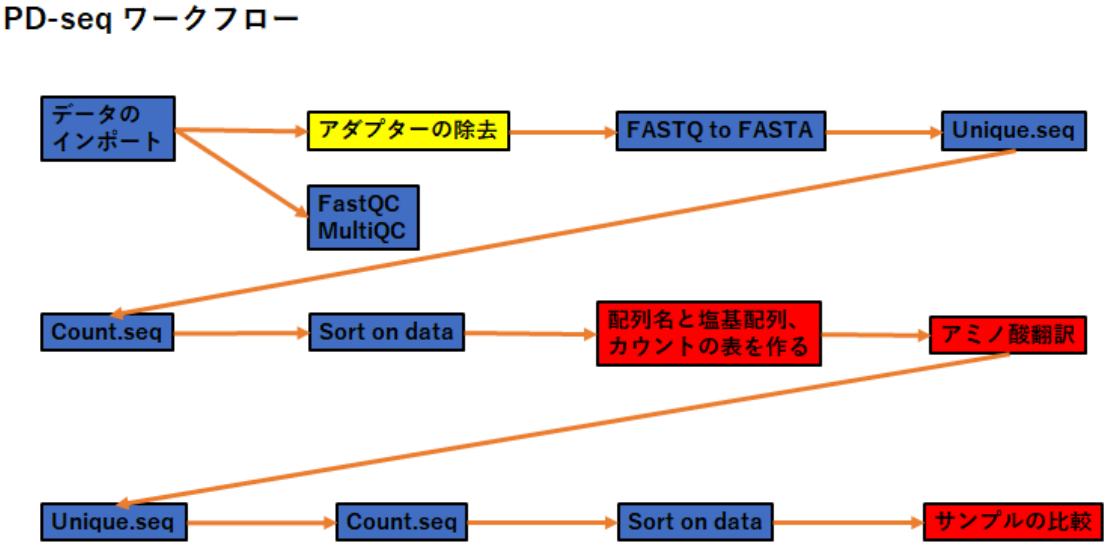
PD-seq ワークフロー関連

最初に考えたワークフロー↓



このワークフローのうち、usegalaxy 上にデフォルトで目的に合うツールがあるか探した結果が下の図。



青色：ツールがある
黄色：しっかりと確かめられていないが恐らくツールがある
赤色：目的に合うツールがない

よって、上図の赤色の部分を行うツールを作成することにした。

1. 配列名と塩基配列、カウント数の表を作成する

まずは、FASTA ファイルを配列名と塩基配列の表形式にした。

fastatotab.py ↓

```
with open('samplefasta.fasta','r') as f, open('output.csv', 'w') as output_file:
        for line in f:
        if line.startswith('>'):
        line = line.replace('\n','\t')
        line.strip('>')
        output_file.write(format(line.strip('>')))
```

samplefasta.fasta(input file) ↓

```
>SN1052:332:C7M91ACXX:2:1101:4699:1999 1:N:0:CGTGAT
TTCGCTACGTATCCTACAGATCAGTAGGACAAATCATATACACAGAATTC
>SN1052:332:C7M91ACXX:2:1101:4970:1997 1:N:0:CGTGAT
TACTCTAAGGAATCTTCTAATGCGTCTCCCCCAACAAAAACGTAGAATTC
>SN1052:332:C7M91ACXX:2:1101:8932:1993 1:N:0:CGTGAT
CTACCTAAACATCCTTACTCGTCAGCTAAATCCGATAAGCCGTAGAATTC
>SN1052:332:C7M91ACXX:2:1101:13606:1989 1:N:0:CGTGAT
CCCACTACCCCCGAACCACCCATCTCATTAGGCTAACCCTTCTGCGAATTC
>SN1052:332:C7M91ACXX:2:1101:16709:1993 1:N:0:CGTGAT
CCCACTACTAAAAACGAAAAACACTAACACAAGCAGAACTTACGGAATTT
>SN1052:332:C7M91ACXX:2:1101:17851:2000 1:N:0:CGTGAT
CTTGCTACAACTAAGCATACAGAGGCGCAGCAAGTAGATTTAAGGGATTG
>SN1052:332:C7M91ACXX:2:1101:1367:2006 1:N:0:CGTGAT
GCAGCTACAAATGAGTAGCACAAGCAACCACAGCAATATTAAAAGCATTC
>SN1052:332:C7M91ACXX:2:1101:2465:2004 1:N:0:CGTGAT
GACTCTACGTCAAAACCCGCTGCGTCGGACTATCACTATGATGAGCATTC
>SN1052:332:C7M91ACXX:2:1101:2292:2219 1:N:0:CGTGAT
GTCACTAATTCCTAAACACAGCCATAAAATGACCCGGCTGCTTAGCATTC
>SN1052:332:C7M91ACXX:2:1101:3162:2245 1:N:0:CGTGAT
ATTACTAAGAACTCAGAGGCAACCAAAACCAACGCCCACCAAAGACTTCG
```

output ↓

1	2
SN1052:332:C7M91ACXX:2:1101:4699:1999 1:N:0:CGTGAT	TTCGCTACGTATCCTACAGATCAGTAGGACAAATCATATACACAGAATTC
SN1052:332:C7M91ACXX:2:1101:4970:1997 1:N:0:CGTGAT	TACTCTAAGGAATCTTCTAATGCGTCTCCCCCAACAAAAACGTAGAATTC
SN1052:332:C7M91ACXX:2:1101:8932:1993 1:N:0:CGTGAT	CTACCTAAACATCCTTACTCGTCAGCTAAATCCGATAAGCCGTAGAATTC
SN1052:332:C7M91ACXX:2:1101:13606:1989 1:N:0:CGTGAT	CCCACTACCCCGAACCACCCATCTCATTAGGCTAACCCTTCTGCGAATTC
SN1052:332:C7M91ACXX:2:1101:16709:1993 1:N:0:CGTGAT	CCCACTACTAAAAACGAAAAACACTAACACAAGCAGAACTTACGGAATTT
SN1052:332:C7M91ACXX:2:1101:17851:2000 1:N:0:CGTGAT	CTTGCTACAACTAAGCATACAGAGGCGCAGCAAGTAGATTTAAGGGATTG
SN1052:332:C7M91ACXX:2:1101:1367:2006 1:N:0:CGTGAT	GCAGCTACAAATGAGTAGCACAAGCAACCACAGCAATATTAAAAGCATTC
SN1052:332:C7M91ACXX:2:1101:2465:2004 1:N:0:CGTGAT	GACTCTACGTCAAAACCCGCTGCGTCGGACTATCACTATGATGAGCATTC
SN1052:332:C7M91ACXX:2:1101:2292:2219 1:N:0:CGTGAT	GTCACTAATTCCTAAACACAGCCATAAAATGACCCGGCTGCTTAGCATTC
SN1052:332:C7M91ACXX:2:1101:3162:2245 1:N:0:CGTGAT	ATTACTAAGAACTCAGAGGCAACCAAAACCAACGCCCACCAAAGACTTCG

ここで、この後合体させる Count.seq のアウトプットを見てみる。

name	total
Representative_Sequence	total
SN1052_332_C7M91ACXX_2_1101_4699_1999	797
SN1052_332_C7M91ACXX_2_1101_4970_1997	278
SN1052_332_C7M91ACXX_2_1101_8932_1993	657
SN1052_332_C7M91ACXX_2_1101_13606_1989	1
SN1052_332_C7M91ACXX_2_1101_16709_1993	1
SN1052_332_C7M91ACXX_2_1101_17851_2000	1
SN1052_332_C7M91ACXX_2_1101_1367_2006	1
SN1052_332_C7M91ACXX_2_1101_2465_2004	3
SN1052_332_C7M91ACXX_2_1101_2292_2219	2
SN1052_332_C7M91ACXX_2_1101_3162_2245	2
SN1052_332_C7M91ACXX_2_1101_3466_2193	1
SN1052_332_C7M91ACXX_2_1101_3707_2061	1
SN1052_332_C7M91ACXX_2_1101_3571_2087	1
SN1052_332_C7M91ACXX_2_1101_3557_2116	7
SN1052_332_C7M91ACXX_2_1101_3822_2192	494
SN1052_332_C7M91ACXX_2_1101_4423_2070	538

作成したアウトプットの配列名部分と比べてみると、

1. 片方では「:」でもう片方は「_」になっている。
2. Count.seq には「1:N:0:～」の部分がない。

といった違いがあったため、そこを修正したスクリプトを作成した。

idnuc.py ↓

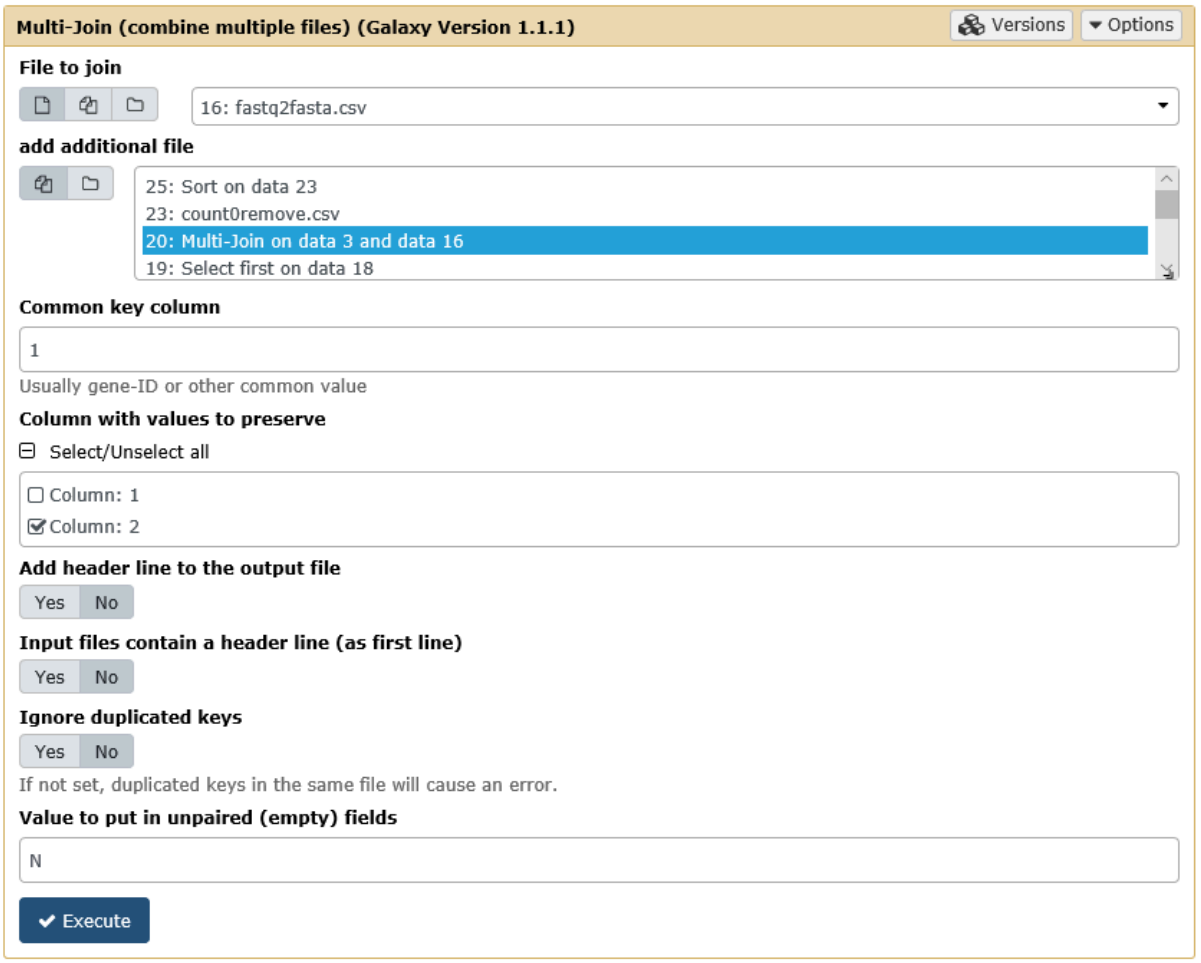
```
with open('sample.fasta','r') as f, open('fastq2fasta.csv', 'w') as output_file:
        for line in f:
        if line.startswith('>'):
        line = line[:-14]
        line = line.replace(':','_')
        line += '\t'
        line.strip('>')
        output_file.write(format(line.strip('>')))
```

inputfile → FASTQ to FASTA 後のアウトプット（最初は unique.seq のアウトプットを使用していたが、count.seq とのファイルの結合時に一意とする配列名が異なることで問題が生じたため、unique 前のFASTA ファイルを使用している。）

output ↓

1	2
SN1052_332_C7M91ACXX_2_1101_4699_1999	TTCGCTACGTATCCTACAGATCAGTAGGACAAATCATATACACAGAATTC
SN1052_332_C7M91ACXX_2_1101_4970_1997	TACTCTAAGGAATCTTCTAATGCGTCTCCCCCAACAAAAACGTAGAATTC
SN1052_332_C7M91ACXX_2_1101_8932_1993	CTACCTAAACATCCTTACTCGTCAGCTAAATCCGATAAGCCGTAGAATTC
SN1052_332_C7M91ACXX_2_1101_13606_1989	CCCACTACCCCGAACCACCCATCTCATTAGGCTAACCCTTCTGCGAATTC
SN1052_332_C7M91ACXX_2_1101_16709_1993	CCCACTACTAAAAACGAAAAACACTAACACAAGCAGAACTTACGGAATTT
SN1052_332_C7M91ACXX_2_1101_17851_2000	CTTGCTACAACTAAGCATACAGAGGCGCAGCAAGTAGATTTAAGGGATTG
SN1052_332_C7M91ACXX_2_1101_1367_2006	GCAGCTACAAATGAGTAGCACAAGCAACCACAGCAATATTAAAAGCATTC
SN1052_332_C7M91ACXX_2_1101_2465_2004	GACTCTACGTCAAAACCCGCTGCGTCGGACTATCACTATGATGAGCATTC
SN1052_332_C7M91ACXX_2_1101_2292_2219	GTCACTAATTCCTAAACACAGCCATAAAATGACCCGGCTGCTTAGCATTC
SN1052_332_C7M91ACXX_2_1101_3162_2245	ATTACTAAGAACTCAGAGGCAACCAAAACCAACGCCCACCAAAGACTTCG
SN1052_332_C7M91ACXX_2_1101_3466_2193	TTTACTAATCACCAGAAAACCAAGGCTAAGGCCAATGCCTATGAGCATTC
SN1052_332_C7M91ACXX_2_1101_3707_2064	ATATCTAATAAGTATAACAAAGAAGAAGCGTATCACAATCCCGAGAATTC

とりあえずうまくいったので、Count.seq のアウトプットとファイルを連結させた。ファイルの連結に関しては usegalaxy の既存のツールである、**Multi-Join (combine multiple files) (Galaxy Version 1.1.1)** を使用し、以下のように設定した。



アウトプットは以下のようになった。

1	2	3
Representative_Sequence	N	total
SN1052_332_C7M91ACXX_2_1101_10000_81728	TTTTCTACTCAGGAATATCATAATGATGCAGCCCACTCATACTAGAATTC	108
SN1052_332_C7M91ACXX_2_1101_10000_9399	GATACTAAGCCATATTAACCCTCATATCATGCGTACCCCGACTAGAATTC	604
SN1052_332_C7M91ACXX_2_1101_10001_16410	CCTTCTACTACCTCTTAGTAATAATATTCTCCCTATTAATAGTAGAATTC	45
SN1052_332_C7M91ACXX_2_1101_10001_18571	CCAACTACCACCCCATATTAAACACAGAATTCGGATCCCCGAGCATCACA	1039
SN1052_332_C7M91ACXX_2_1101_10001_41467	GAGTCTAATCCGGATACCCCGCCCCACGCTGCGAAATACACTAAGAATTC	N
SN1052_332_C7M91ACXX_2_1101_10003_11814	ACGCCTACGTAGCAACCCCCTAATGATGCACCGAAACAAGAAGAGAATTC	347
SN1052_332_C7M91ACXX_2_1101_10005_28686	TCGTCTACACAAGAGAAAGATCAGTCTGAGTACAAGACAGCAAAGAATTC	N
SN1052_332_C7M91ACXX_2_1101_10005_36778	ACGACTACACAAGCATCGTCGACCAAGACGGCACCACCTAAAAAGAATTC	N
SN1052_332_C7M91ACXX_2_1101_10007_18359	CTGTCTACATACTATGATTAGGCAAATTAATAGCAGAATCCAAAGAATTC	528
SN1052_332_C7M91ACXX_2_1101_10008_42849	AAAACTAAAACATCTGACCCCTCAAATAAATATGCCCAGACAGAGAATTC	N
SN1052_332_C7M91ACXX_2_1101_10008_73445	TTCTCTAAACCCTATTAAAAGAACCAATAGTAGTACCCCCCCTAGAATTC	N
SN1052_332_C7M91ACXX_2_1101_10009_32911	CTATCTACAACGACCGACGCTGATTAATCGGCATCCTAACCCCGGAATTC	2
SN1052_332_C7M91ACXX_2_1101_10009_36995	GCAGCTACAAATGAGTAGCACAAGCAACCACAGCAATATTAAAAGAATTC	N
SN1052_332_C7M91ACXX_2_1101_10009_88180	AACTCTACTAAGAAGTACCCAACTAATACCCATGAACATGCATAGAATTC	N
SN1052_332_C7M91ACXX_2_1101_10010_62263	CCCACTACCCACTAGGAACCAGACGATCCTTCCACGCCTCCCTAGAATTC	N

この表から、カウント数が0である行（total が N となっているもの）を取り除く。

count0remove.py ↓

```
with open('nameseqcount.tabular','r') as f, open('count0remove.csv','w') as output_file:
        for line in f:
        if not line.endswith('N\n'):
        output_file.write(line)
```

input file → 上のアウトプット

output ↓

1	2	3
Representative_Sequence	N	total
SN1052_332_C7M91ACXX_2_1101_10000_81728	TTTTCTACTCAGGAATATCATAATGATGCAGCCCACTCATACTAGAATTC	108
SN1052_332_C7M91ACXX_2_1101_10000_9399	GATACTAAGCCATATTAACCCTCATATCATGCGTACCCCGACTAGAATTC	604
SN1052_332_C7M91ACXX_2_1101_10001_16410	CCTTCTACTACCTCTTAGTAATAATATTCTCCCTATTAATAGTAGAATTC	45
SN1052_332_C7M91ACXX_2_1101_10001_18571	CCAACTACCACCCCATATTAAACACAGAATTCGGATCCCCGAGCATCACA	1039
SN1052_332_C7M91ACXX_2_1101_10003_11814	ACGCCTACGTAGCAACCCCCTAATGATGCACCGAAACAAGAAGAGAATTC	347
SN1052_332_C7M91ACXX_2_1101_10007_18359	CTGTCTACATACTATGATTAGGCAAATTAATAGCAGAATCCAAAGAATTC	528
SN1052_332_C7M91ACXX_2_1101_10009_32911	CTATCTACAACGACCGACGCTGATTAATCGGCATCCTAACCCCGGAATTC	2
SN1052_332_C7M91ACXX_2_1101_10011_82376	AAGGCTACCGCACCCACATACACAGCGCCGACCGAGTCAGCGCAGAATTC	179
SN1052_332_C7M91ACXX_2_1101_10013_31633	CCGTCTAACCATCCCTCTTACTAATAATCAGCCAAAAAATCGAAGAATTC	125
SN1052_332_C7M91ACXX_2_1101_10014_30810	CACTCTAATTATAAAAAGCAATAATACAAGCCTAATACTACGAAGAATTC	60
SN1052_332_C7M91ACXX_2_1101_10015_27188	ATAACTAATTATGAGCAAGCTGATAACACGTAGAAGCATTAGCAGAATTC	516
SN1052_332_C7M91ACXX_2_1101_10016_25465	GTTTCTACCTAAACATCGCAAGGATAGAAATCAAAGCATGATAGAATTCG	15
SN1052_332_C7M91ACXX_2_1101_10016_61899	GCCACTACTAAATCAACCCACCACAACTCGACACAAAACTAAAAGAATTC	488
SN1052_332_C7M91ACXX_2_1101_10018_2968	GAATCTAACCCGTCTACACACGCTTCCAATGCAACAGCAAATAAGAATTC	921
SN1052_332_C7M91ACXX_2_1101_10019_12446	TCGTCTAAGCAGGATAACCCCCCCAAACTCCCAATAACAAAAGAATTCGG	369
SN1052_332_C7M91ACXX_2_1101_10025_37188	GTACCTACCAATACGAAGGATCATCCATATAAGACCGAATAGAGAATTCG	34
SN1052_332_C7M91ACXX_2_1101_10027_59564	TAAGCTACAATCAGTCTAAGAACTAGGAACAGTATACCACAGAGAATTCG	11
SN1052_332_C7M91ACXX_2_1101_10028_22179	ACATCTACCGAACCGTAGGCCTCCTATAATTAATATCAAAAATAGAATTC	83
SN1052_332_C7M91ACXX_2_1101_10029_32210	TTATCTAAAGAACATAAATCATCGCACAACCACCGAACAACTAGAATTCG	1793
SN1052_332_C7M91ACXX_2_1101_10029_97571	TCACACTCAATCTTTTATCACGAAGTCATGATTGAATCGCGAGTGGTCGG	2
SN1052_332_C7M91ACXX_2_1101_10034_19135	AATGCTAATAAGTCCCAAACACCATCCAATAATCACGCTGCAAGAATTCG	136
SN1052_332_C7M91ACXX_2_1101_10035_99713	ACCGCTACTAATAACCCCCATTCGAATCCATACCCGCAATATTAGAATTC	71
SN1052_332_C7M91ACXX_2_1101_10037_5006	TGCGCTACCACTCCCACATCAAATTACCAACCTGATTACAACCAGAATTC	152

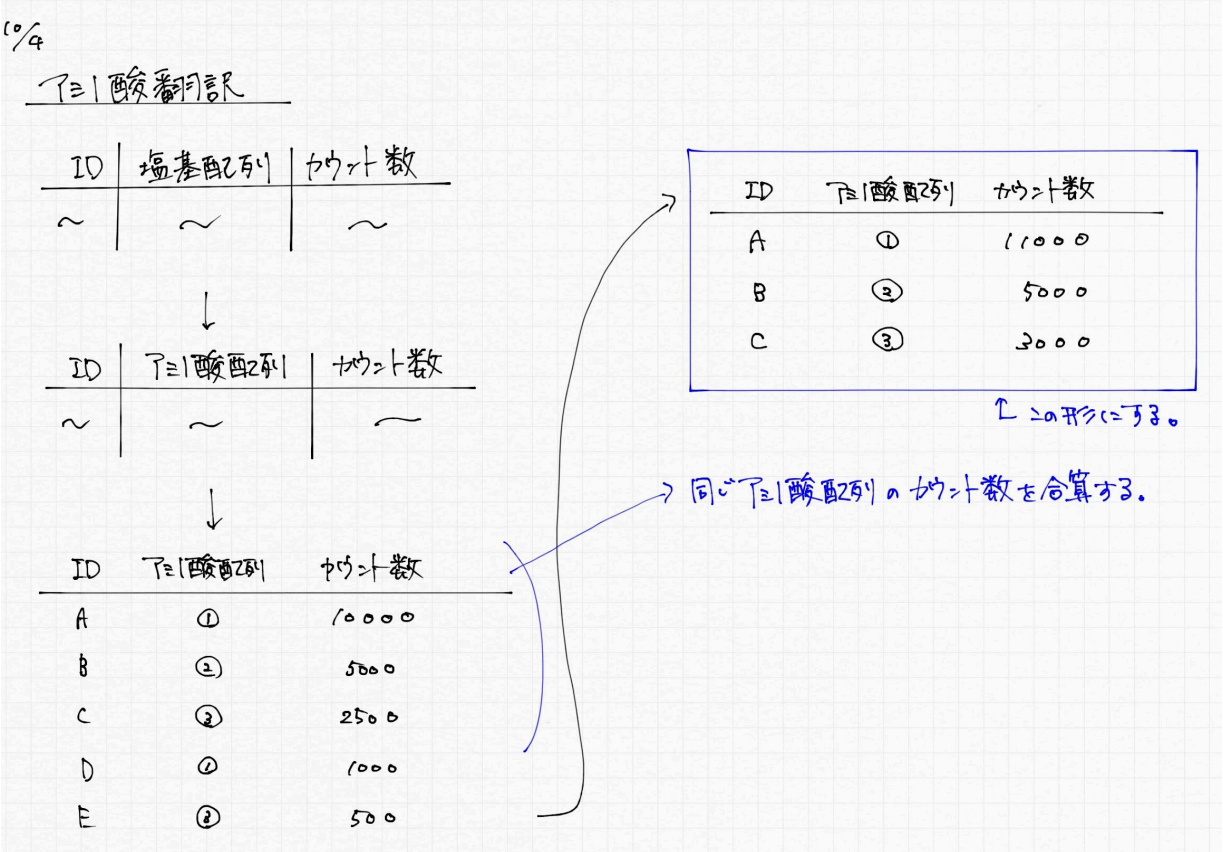
最後に、usegalaxy の既存のツールである、**Sort data in ascending or descending order (Galaxy Version 1.1.0)** を使用してカウント数が多い順にソートして求めている表が完成した。

1	2	3
SN1052_332_C7M91ACXX_2_1101_6006_2092	TTTGCTACTTCCAATGCAGAAGAGAAGTCCGAATACAAATCTTAGAATTC	52487
SN1052_332_C7M91ACXX_2_1101_10697_2034	CTGTCTACTCCCGCATATGCACCGCACCACGACTCATCGTCTGAGAATTC	35904
SN1052_332_C7M91ACXX_2_1101_1927_3172	ATCACTAACAATCCACATTATTCATAATCATCTTAAGAAAATGAATTCGG	21964
SN1052_332_C7M91ACXX_2_1101_9400_2159	AGGACTACTGCACACCACACAACAACTCCCCCAAACGAGGCAGAGAATTC	16875
SN1052_332_C7M91ACXX_2_1101_10204_6047	GTACCTACTGACGCTGCCCCTAAACAACCGCAGCCCTATTATAAGAATTC	15175
SN1052_332_C7M91ACXX_2_1101_11067_2027	GGTGCTAATACTACTTCTGACCCTTCTGCCGAGGCTGATTATCAGAATTC	13511
SN1052_332_C7M91ACXX_2_1101_19728_7521	GGAACTAAGCCGCAGCACAACACAACCGAGCCGCCTTAGTAAAAGAATTC	12736
SN1052_332_C7M91ACXX_2_1101_10121_3973	GCGTCTACGAATTAGGCAGCGAAGCACTACGAACATGACCAATAGAATTC	12555
SN1052_332_C7M91ACXX_2_1101_19352_6204	TACTCTAAGAATTAACACCCAACGTACTAATCAAACCAGTACAAGAATTC	11033
SN1052_332_C7M91ACXX_2_1101_11241_5775	CCCACTACCCCATACCCCGCCCCGGCCCATCATAATACCCAAAAGAATTC	10381
SN1052_332_C7M91ACXX_2_1101_1318_6454	GAAGCTACAAAATCTTCGCATAACCATACTTCGAACTAGCCATAGAATTC	8268
SN1052_332_C7M91ACXX_2_1101_3083_6319	TCACCTACACCCTACCCTGAGGCTACACCCACGACTCCGGAAAAGAATTC	8154
SN1052_332_C7M91ACXX_2_1101_20560_7822	GGGCCTAACTAGACGAATCACCAGAAACAAACTTAGGAAAAATAGAATTC	8067
SN1052_332_C7M91ACXX_2_1101_11209_2978	CTCTCTAAAACGGCGTCCGAAAATGAAAAGAATTCGGATCCCCGAGCATC	7712
SN1052_332_C7M91ACXX_2_1101_21043_6551	AACACTAAGTACTCTAACTACTCTCAAATCAGAAAGCAAAGAGAATTCGG	6983
SN1052_332_C7M91ACXX_2_1101_7856_3195	ACTGCTAATTCCGCGGCTAACCCACCTTAGCACCAATAGACACAGAATTC	6973
SN1052_332_C7M91ACXX_2_1101_12649_7386	CCGTCTACTCCACCTTCGACCCAGAAATCCACCGCCCCCAAAAAGAATTC	6891
SN1052_332_C7M91ACXX_2_1101_8709_2794	CCACCTAATGACCATACCACCACGCCGACGACTAAAGACGCTTAGAATTC	6561
SN1052_332_C7M91ACXX_2_1101_6985_2719	CAAACTACATACGAATAAGCCTCTTCGTAACATTCTCACTAAAAGAATTC	6284
SN1052_332_C7M91ACXX_2_1101_6769_4236	CTTGCTATTCCCCACAGCCAACAGCTTCTACATACCAACACTAGAATTCG	6118
SN1052_332_C7M91ACXX_2_1101_4580_2803	TAGACTAAGAAAACAGAGAAACCAGAGCCCAATGAAACCAAAGAATTCGG	5719
SN1052_332_C7M91ACXX_2_1101_4517_2526	CGCCCTAATAATGCCTAAACGTAGGAGCAAAACGATGCTAACGAGAATTC	5117
SN1052_332_C7M91ACXX_2_1101_5405_2307	GCGCCTAAAGCCACCGCCTAGACGGCTCAAGAAGACCAAAAACAGAATTC	5026
SN1052_332_C7M91ACXX_2_1101_6015_7002	AACACTAAAACGAATTATTACCAATCCTATTACCAAACTAATTAGAATTC	4980
SN1052_332_C7M91ACXX_2_1101_7046_5368	TTAACTACCTCATATACTTAACCCGCCTCTAACAAGTAGGCCAAGAATTC	4719
SN1052_332_C7M91ACXX_2_1101_3664_10597	GACCCTACGGACACAAAACATTACTCAACACATCCCCAATAAAAGAATTC	4677
SN1052_332_C7M91ACXX_2_1101_8661_4005	ATTCCTAACTATGAGCACAACAGAATACACACTATGCTAAACAGAATTCG	4652
SN1052_332_C7M91ACXX_2_1101_8947_4551	GGAACTAAAGAATCGCAGTCCACCTATGCTCAGCATAATCCAGAGAATTC	4571
SN1052_332_C7M91ACXX_2_1101_14763_10819	CCCACTAAAGACCAAGATCAAGCCGCCTATACACAACCTAAAAAGAATTC	4487
SN1052_332_C7M91ACXX_2_1101_9899_6510	ATCCCTAATAACACCTCTTACCCTTAATAGTACCCGACAAACCAGAATTC	4312
SN1052_332_C7M91ACXX_2_1101_19956_9276	CAGACTAAAATGCAGACACCATACTGCCGAATAATATAAAAGAATTCGGA	4241
SN1052_332_C7M91ACXX_2_1101_17869_9736	TAATCTAACGCGTAGAATTACTCTCAACAACAACACTAAGCCTAGAATTC	4079

2. アミノ酸翻訳

インプットファイルはできれば上で作成した表を使用したいので、当初考えていたワークフローを変更する。

unique と count をもう一度やらずに同じアミノ酸になった行のカウント数を合算すればいいことに気づいたのでその方向性で考える。



まずはアミノ酸翻訳を作成する。

https://bi.biopapyrus.jp/python/sample/dna-translation.html
のディクショナリを利用した方法をいじって作成してみる。
次のページがそのスクリプト。

```
DNA2Protein = {
	'TTT' : 'F', 'TCT' : 'S', 'TAT' : 'Y', 'TGT' : 'C',
	'TTC' : 'F', 'TCC' : 'S', 'TAC' : 'Y', 'TGC' : 'C',
	'TTA' : 'L', 'TCA' : 'S', 'TAA' : '*', 'TGA' : '*',
	'TTG' : 'L', 'TCG' : 'S', 'TAG' : '*', 'TGG' : 'W',

	'CTT' : 'L', 'CCT' : 'P', 'CAT' : 'H', 'CGT' : 'R',
	'CTC' : 'L', 'CCC' : 'P', 'CAC' : 'H', 'CGC' : 'R',
	'CTA' : 'L', 'CCA' : 'P', 'CAA' : 'Q', 'CGA' : 'R',
	'CTG' : 'L', 'CCG' : 'P', 'CAG' : 'Q', 'CGG' : 'R',

	'ATT' : 'I', 'ACT' : 'T', 'AAT' : 'N', 'AGT' : 'S',
	'ATC' : 'I', 'ACC' : 'T', 'AAC' : 'N', 'AGC' : 'S',
	'ATA' : 'I', 'ACA' : 'T', 'AAA' : 'K', 'AGA' : 'R',
	'ATG' : 'M', 'ACG' : 'T', 'AAG' : 'K', 'AGG' : 'R',

	'GTT' : 'V', 'GCT' : 'A', 'GAT' : 'D', 'GGT' : 'G',
	'GTC' : 'V', 'GCC' : 'A', 'GAC' : 'D', 'GGC' : 'G',
	'GTA' : 'V', 'GCA' : 'A', 'GAA' : 'E', 'GGA' : 'G',
	'GTG' : 'V', 'GCG' : 'A', 'GAG' : 'E', 'GGG' : 'G'
}


dna = 'GCCTGGGACGAATTTTAA'
p = ''
for i in range(0, int(len(dna) / 3)):
	codon = dna[3 * i : 3 * i + 3]
	if (codon in DNA2Protein):
		p += DNA2Protein[codon]
	else:
		p += 'X'

print(p)
```

「dna = 」の部分に上で作成した表の2列目（塩基配列）を読み込むようにする。
print 部分をwrite に変えて上で作成した表の2列目をアミノ酸に書き換えるようにする。

課題

- 2列目だけを抜き出す方法と置換する方法を調べる。

ディクショナリについて
https://www.pythonweb.jp/tutorial/dictionary/index1.html

（181008）取り敢えず、上の課題について考えてみたが思いつかなかったので、FASTAファイルの塩基配列をアミノ酸翻訳して、アミノ酸配列のFASTAファイルを作るスクリプトを作成した。
以下がそのスクリプトである。

nuc2amino.py ↓

```
DNA2Protein = {
	'TTT' : 'F', 'TCT' : 'S', 'TAT' : 'Y', 'TGT' : 'C',
	'TTC' : 'F', 'TCC' : 'S', 'TAC' : 'Y', 'TGC' : 'C',
	'TTA' : 'L', 'TCA' : 'S', 'TAA' : '*', 'TGA' : '*',
	'TTG' : 'L', 'TCG' : 'S', 'TAG' : '*', 'TGG' : 'W',
	'CTT' : 'L', 'CCT' : 'P', 'CAT' : 'H', 'CGT' : 'R',
	'CTC' : 'L', 'CCC' : 'P', 'CAC' : 'H', 'CGC' : 'R',
	'CTA' : 'L', 'CCA' : 'P', 'CAA' : 'Q', 'CGA' : 'R',
	'CTG' : 'L', 'CCG' : 'P', 'CAG' : 'Q', 'CGG' : 'R',
	'ATT' : 'I', 'ACT' : 'T', 'AAT' : 'N', 'AGT' : 'S',
	'ATC' : 'I', 'ACC' : 'T', 'AAC' : 'N', 'AGC' : 'S',
	'ATA' : 'I', 'ACA' : 'T', 'AAA' : 'K', 'AGA' : 'R',
	'ATG' : 'M', 'ACG' : 'T', 'AAG' : 'K', 'AGG' : 'R',
	'GTT' : 'V', 'GCT' : 'A', 'GAT' : 'D', 'GGT' : 'G',
	'GTC' : 'V', 'GCC' : 'A', 'GAC' : 'D', 'GGC' : 'G',
	'GTA' : 'V', 'GCA' : 'A', 'GAA' : 'E', 'GGA' : 'G',
	'GTG' : 'V', 'GCG' : 'A', 'GAG' : 'E', 'GGG' : 'G'
}

p = ''

with open('samplefasta.fasta','r') as f, open('nuc2amino.fasta', 'w') as output_file:
	for line in f:
		if line.startswith('>'):
			line = line[:-14]
			line = line.replace(':','_')
			line += '\n'
			output_file.write(line)
		else:
			for i in range(0, int(len(line)/3)):
				codon = line[3*i:3*i+3]
				if (codon in DNA2Protein):
					p += DNA2Protein[codon]
				else:
					break
			line.replace(line, p)
			p += '\n'
		output_file.write(p)
		p = ''
```

インプットファイルとアウトプットファイルは以下。

input → samplefasta.fasta (2ページ目に記載)

output ↓

```
>SN1052_332_C7M91ACXX_2_1101_4699_1999
FATYPTDQ*DKSYTQN
>SN1052_332_C7M91ACXX_2_1101_4970_1997
YSKESSNASPPTKT*N
>SN1052_332_C7M91ACXX_2_1101_8932_1993
LPKHPYSSAKSDKP*N
>SN1052_332_C7M91ACXX_2_1101_13606_1989
PTTPNHPSH*ANPSAN
>SN1052_332_C7M91ACXX_2_1101_16709_1993
PTTKNEKH*HKQNLRN
>SN1052_332_C7M91ACXX_2_1101_17851_2000
LATTKHTEAQQVDLRD
>SN1052_332_C7M91ACXX_2_1101_1367_2006
AATNE*HKQPQQY*KH
>SN1052_332_C7M91ACXX_2_1101_2465_2004
DSTSKPAASDYHYDEH
>SN1052_332_C7M91ACXX_2_1101_2292_2219
VTNS*TQP*NDPAA*H
>SN1052_332_C7M91ACXX_2_1101_3162_2245
ITKNSEATKTNAHQRL
```

今までと同様に表形式にして、1 で作った表とマージすることで、2 の目標が達成できないかやってみる。

(181015) 表形式にした。

nuc2aminocsv.py ↓

```
DNA2Protein = {
        'TTT' : 'F', 'TCT' : 'S', 'TAT' : 'Y', 'TGT' : 'C',
        'TTC' : 'F', 'TCC' : 'S', 'TAC' : 'Y', 'TGC' : 'C',
        'TTA' : 'L', 'TCA' : 'S', 'TAA' : '*', 'TGA' : '*',
        'TTG' : 'L', 'TCG' : 'S', 'TAG' : '*', 'TGG' : 'W',
        'CTT' : 'L', 'CCT' : 'P', 'CAT' : 'H', 'CGT' : 'R',
        'CTC' : 'L', 'CCC' : 'P', 'CAC' : 'H', 'CGC' : 'R',
        'CTA' : 'L', 'CCA' : 'P', 'CAA' : 'Q', 'CGA' : 'R',
        'CTG' : 'L', 'CCG' : 'P', 'CAG' : 'Q', 'CGG' : 'R',
        'ATT' : 'I', 'ACT' : 'T', 'AAT' : 'N', 'AGT' : 'S',
        'ATC' : 'I', 'ACC' : 'T', 'AAC' : 'N', 'AGC' : 'S',
        'ATA' : 'I', 'ACA' : 'T', 'AAA' : 'K', 'AGA' : 'R',
        'ATG' : 'M', 'ACG' : 'T', 'AAG' : 'K', 'AGG' : 'R',
        'GTT' : 'V', 'GCT' : 'A', 'GAT' : 'D', 'GGT' : 'G',
        'GTC' : 'V', 'GCC' : 'A', 'GAC' : 'D', 'GGC' : 'G',
        'GTA' : 'V', 'GCA' : 'A', 'GAA' : 'E', 'GGA' : 'G',
        'GTG' : 'V', 'GCG' : 'A', 'GAG' : 'E', 'GGG' : 'G'
}
p = ''
```

```
with open('samplefasta.fasta','r') as f, open('nuc2amino.csv', 'w') as output_file:
	for line in f:
		if line.startswith('>'):
			line = line[:-14]
			line = line.replace(':','_')
			line += '\t'
			line.strip('>')
			output_file.write(format(line.strip('>')))
		else:
			for i in range(0, int(len(line)/3)):
				codon = line[3*i:3*i+3]
				if (codon in DNA2Protein):
					p += DNA2Protein[codon]
				else:
					break
			line.replace(line, p)
			p += '\n'
		output_file.write(p)
		p = ''
```

input : samplefasta.fasta

output ↓

1	2
SN1052_332_C7M91ACXX_2_1101_4699_1999	FATYPTDQ*DKSYTQN
SN1052_332_C7M91ACXX_2_1101_4970_1997	YSKESSNASPPTKT*N
SN1052_332_C7M91ACXX_2_1101_8932_1993	LPKHPYSSAKSDKP*N
SN1052_332_C7M91ACXX_2_1101_13606_1989	PTTPNHPSH*ANPSAN
SN1052_332_C7M91ACXX_2_1101_16709_1993	PTTKNEKH*HKQNLRN
SN1052_332_C7M91ACXX_2_1101_17851_2000	LATTKHTEAQQVDLRD
SN1052_332_C7M91ACXX_2_1101_1367_2006	AATNE*HKQPQQY*KH
SN1052_332_C7M91ACXX_2_1101_2465_2004	DSTSKPAASDYHYDEH
SN1052_332_C7M91ACXX_2_1101_2292_2219	VTNS*TQP*NDPAA*H
SN1052_332_C7M91ACXX_2_1101_3162_2245	ITKNSEATKTNAHQRL

配列ID、塩基配列、アミノ酸配列の表も作成してみた。色々やってみたが2列目がアミノ酸配列で3列目が塩基配列の表になった。

idnucaminocsv.py ↓

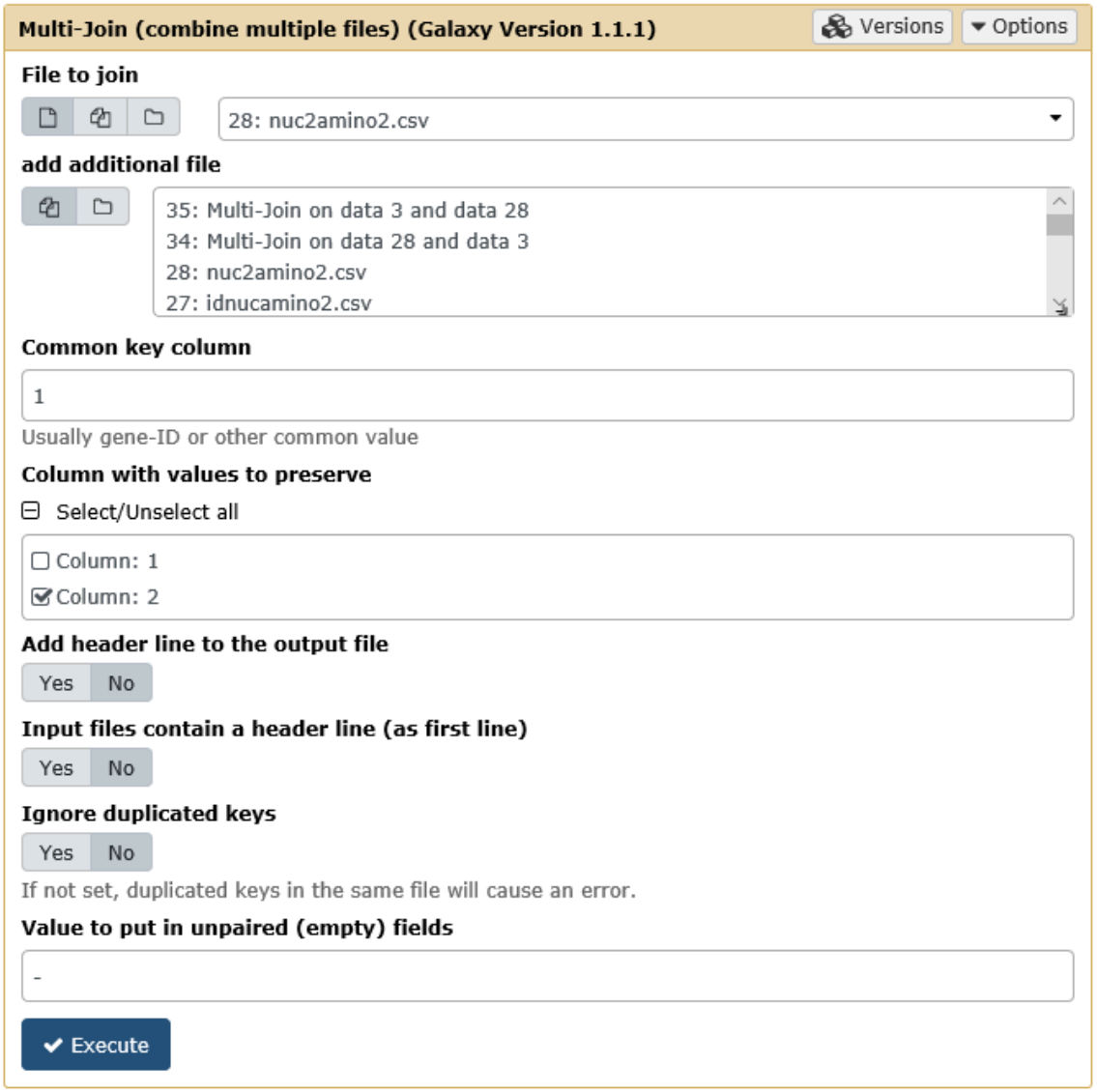
```
DNA2Protein = {
        'TTT' : 'F', 'TCT' : 'S', 'TAT' : 'Y', 'TGT' : 'C',
        'TTC' : 'F', 'TCC' : 'S', 'TAC' : 'Y', 'TGC' : 'C',
        'TTA' : 'L', 'TCA' : 'S', 'TAA' : '*', 'TGA' : '*',
        'TTG' : 'L', 'TCG' : 'S', 'TAG' : '*', 'TGG' : 'W',
        'CTT' : 'L', 'CCT' : 'P', 'CAT' : 'H', 'CGT' : 'R',
        'CTC' : 'L', 'CCC' : 'P', 'CAC' : 'H', 'CGC' : 'R',
        'CTA' : 'L', 'CCA' : 'P', 'CAA' : 'Q', 'CGA' : 'R',
        'CTG' : 'L', 'CCG' : 'P', 'CAG' : 'Q', 'CGG' : 'R',
        'ATT' : 'I', 'ACT' : 'T', 'AAT' : 'N', 'AGT' : 'S',
        'ATC' : 'I', 'ACC' : 'T', 'AAC' : 'N', 'AGC' : 'S',
        'ATA' : 'I', 'ACA' : 'T', 'AAA' : 'K', 'AGA' : 'R',
        'ATG' : 'M', 'ACG' : 'T', 'AAG' : 'K', 'AGG' : 'R',
        'GTT' : 'V', 'GCT' : 'A', 'GAT' : 'D', 'GGT' : 'G',
        'GTC' : 'V', 'GCC' : 'A', 'GAC' : 'D', 'GGC' : 'G',
        'GTA' : 'V', 'GCA' : 'A', 'GAA' : 'E', 'GGA' : 'G',
        'GTG' : 'V', 'GCG' : 'A', 'GAG' : 'E', 'GGG' : 'G'
}

p = ''

with open('samplefasta.fasta','r') as f, open('idnucamino.csv', 'w') as output_file:
        for line in f:
                if line.startswith('>'):
                        line = line[:-14]
                        line = line.replace(':','_')
                        line += '\t'
                        line.strip('>')
                        output_file.write(format(line.strip('>')))
                else:
                        for i in range(0, int(len(line)/3)):
                                codon = line[3*i:3*i+3]
                                if (codon in DNA2Protein):
                                        p += DNA2Protein[codon]
                                else:
                                        break
                        line.replace('\n', '\t')
                        p += '\t'
                        p += line
                        p += '\n'
                        output_file.write(p)
                p = ''
```

input : samplefasta.fasta
output ↓

1	2	3
SN1052_332_C7M91ACXX_2_1101_4699_1999	FATYPTDQ*DKSYTQN	TTCGCTACGTATCCTACAGATCAGTAGGACAAATCATATACACAGAATTC
SN1052_332_C7M91ACXX_2_1101_4970_1997	YSKESSNASPPTKT*N	TACTCTAAGGAATCTTCTAATGCGTCTCCCCCAACAAAAACGTAGAATTC
SN1052_332_C7M91ACXX_2_1101_8932_1993	LPKHPYSSAKSDKP*N	CTACCTAAACATCCTTACTCGTCAGCTAAATCCGATAAGCCGTAGAATTC
SN1052_332_C7M91ACXX_2_1101_13606_1989	PTTPNHPSH*ANPSAN	CCCACTACCCCGAACCACCCATCTCATTAGGCTAACCCTTCTGCGAATTC
SN1052_332_C7M91ACXX_2_1101_16709_1993	PTTKNEKH*HKQNLRN	CCCACTACTAAAAACGAAAAACACTAACACAAGCAGAACTTACGGAATTT
SN1052_332_C7M91ACXX_2_1101_17851_2000	LATTKHTEAQQVDLRD	CTTGCTACAACTAAGCATACAGAGGCGCAGCAAGTAGATTTAAGGGATTG
SN1052_332_C7M91ACXX_2_1101_1367_2006	AATNE*HKQPQQY*KH	GCAGCTACAAATGAGTAGCACAAGCAACCACAGCAATATTAAAAGCATTC
SN1052_332_C7M91ACXX_2_1101_2465_2004	DSTSKPAASDYHYDEH	GACTCTACGTCAAAACCCGCTGCGTCGGACTATCACTATGATGAGCATTC
SN1052_332_C7M91ACXX_2_1101_2292_2219	VTNS*TQP*NDPAA*H	GTCACTAATTCCTAAACACAGCCATAAAATGACCCGGCTGCTTAGCATTC
SN1052_332_C7M91ACXX_2_1101_3162_2245	ITKNSEATKTNAHQRL	ATTACTAAGAACTCAGAGGCAACCAAAACCAACGCCCACCAAAGACTTCG

1の最後の表と、上の表を連結させようと前回と同じように、**Multi-Join (combine multiple files) (Galaxy Version 1.1.1)** を使用したが、うまくいかない。

塩基配列のcountseq のアウトプットと、IDとアミノ酸配列の表で Multi-Join してみる。



1	2	3
Representative_Sequence	-	total
SN1052_332_C7M91ACXX_2_1101_10000_81728	FSTQEYHNDAAHSY*N	108
SN1052_332_C7M91ACXX_2_1101_10000_9399	DTKPY*PSYHAYPD*N	604
SN1052_332_C7M91ACXX_2_1101_10001_16410	PSTTS***YSPY***N	45
SN1052_332_C7M91ACXX_2_1101_10001_18571	PTTTPY*TQNSDPRAS	1039
SN1052_332_C7M91ACXX_2_1101_10001_41467	ESNPDTPPHAAKYTKN	-
SN1052_332_C7M91ACXX_2_1101_10003_11814	TPT*QPPNDAPKQEEN	347
SN1052_332_C7M91ACXX_2_1101_10005_28686	SSTQEKDQSEYKTAKN	-
SN1052_332_C7M91ACXX_2_1101_10005_36778	TTTQASSTKTAPPKKN	-
SN1052_332_C7M91ACXX_2_1101_10007_18359	LSTYYD*AN**QNPKN	528
SN1052_332_C7M91ACXX_2_1101_10008_42849	KTKTSDPSNKYAQTEN	-
SN1052_332_C7M91ACXX_2_1101_10008_73445	FSKPY*KNQ**YPP*N	-
SN1052_332_C7M91ACXX_2_1101_10009_32911	LSTTTDAD*SAS*PRN	2
SN1052_332_C7M91ACXX_2_1101_10009_36995	AATNE*HKQPQQY*KN	-
SN1052_332_C7M91ACXX_2_1101_10009_88180	NSTKKYPTNTHEHA*N	-
SN1052_332_C7M91ACXX_2_1101_10010_62263	PTTH*EPDDPSTPP*N	-
SN1052_332_C7M91ACXX_2_1101_10010_63217	FTTEPNDPSKQK*IEF	-

アミノ酸配列的にカウント0を「-」にした。「-」のある行を消すスクリプトを用意して、ソートをかける（よく考えたらN\nにしていたので-にする必要はなかったが作ったのでこのままいく）。

count-remove.py ↓

```
with open('nameaminocount.tabular','r') as f, open('count-remove.csv','w') as output_file:
    for line in f:
        if not line.endswith('-\n'):
            output_file.write(line)
```

1	2	3
SN1052_332_C7M91ACXX_2_1101_6006_2092	FATSNAEEKSEYKS*N	52487
SN1052_332_C7M91ACXX_2_1101_10697_2034	LSTPAYAPHHDSSSEN	35904
SN1052_332_C7M91ACXX_2_1101_1927_3172	ITNNPHYS*SS*ENEF	21964
SN1052_332_C7M91ACXX_2_1101_9400_2159	RTTAHHTTTPPNEAEN	16875
SN1052_332_C7M91ACXX_2_1101_10204_6047	VPTDAAPKQPQPYYKN	15175
SN1052_332_C7M91ACXX_2_1101_11067_2027	GANTTSDPSAEADYQN	13511
SN1052_332_C7M91ACXX_2_1101_19728_7521	GTKPQHNTTEPP**KN	12736
SN1052_332_C7M91ACXX_2_1101_10121_3973	ASTN*AAKHYEHDQ*N	12555
SN1052_332_C7M91ACXX_2_1101_19352_6204	YSKN*HPTY*SNQYKN	11033
SN1052_332_C7M91ACXX_2_1101_11241_5775	PTTPYPAPAHHNTQKN	10381
SN1052_332_C7M91ACXX_2_1101_1318_6454	EATKSSHNHTSN*P*N	8268
SN1052_332_C7M91ACXX_2_1101_3083_6319	SPTPYPEATPTTPEKN	8154
SN1052_332_C7M91ACXX_2_1101_20560_7822	GPN*TNHQKQT*EK*N	8067
SN1052_332_C7M91ACXX_2_1101_11209_2978	LSKTASENEKNSDPRA	7712
SN1052_332_C7M91ACXX_2_1101_21043_6551	NTKYSNYSQIRKQREF	6983
SN1052_332_C7M91ACXX_2_1101_7856_3195	TANSAANPP*HQ*TQN	6973
SN1052_332_C7M91ACXX_2_1101_12649_7386	PSTPPSTQKSTAPKKN	6891
SN1052_332_C7M91ACXX_2_1101_8709_2794	PPNDHTTTPTTKDA*N	6561
SN1052_332_C7M91ACXX_2_1101_6985_2719	QTTYE*ASS*HSH*KN	6284
SN1052_332_C7M91ACXX_2_1101_6769_4236	LAIPHSQQLLHTNTRI	6118
SN1052_332_C7M91ACXX_2_1101_4580_2803	*TKKTEKPEPNETKEF	5719
SN1052_332_C7M91ACXX_2_1101_4517_2526	RPNNA*T*EQNDANEN	5117
SN1052_332_C7M91ACXX_2_1101_5405_2307	APKATA*TAQEDQKQN	5026
SN1052_332_C7M91ACXX_2_1101_6015_7002	NTKTNYYQSYYQTN*N	4980
SN1052_332_C7M91ACXX_2_1101_7046_5368	LTTSYT*PASNK*AKN	4719

2列目を比較して同じものの3列目（カウント数）を合算する。
方法が思いつかないので、アミノ酸のFASTAファイルを用意してunique → count する。
unique だけでcounttable が出てきたのでcount を使わないでやってみる。

nuc2amino2.fasta → unique.seqs → multi-join → count-remove2.csv → sort

1	2	3
SN1052_332_C7M91ACXX_2_1101_8829_2030	LTNYYPQKPP*HSE*N	69727
SN1052_332_C7M91ACXX_2_1101_4841_2144	MSNKNY*QKLPILKRI	62504
SN1052_332_C7M91ACXX_2_1101_6006_2092	FATSNAEEKSEYKS*N	56231
SN1052_332_C7M91ACXX_2_1101_12814_2216	QSTYSP*PQNKNSSPN	42964
SN1052_332_C7M91ACXX_2_1101_10697_2034	LSTPAYAPHHDSSSEN	39553
SN1052_332_C7M91ACXX_2_1101_14695_2196	EANPPPNYHNPSPLLN	29632
SN1052_332_C7M91ACXX_2_1101_16709_1993	PTTKNEKH*HKQNLRN	26470
SN1052_332_C7M91ACXX_2_1101_2814_4081	VAT*YKPSYHYSP*KH	25182
SN1052_332_C7M91ACXX_2_1101_2972_2567	YSKN*HPTY*SNQYKH	18752
SN1052_332_C7M91ACXX_2_1101_3082_2717	ISTDAAQ*QNHSDDKH	18147
SN1052_332_C7M91ACXX_2_1101_9400_2159	RTTAHHTTTPPNEAEN	17607
SN1052_332_C7M91ACXX_2_1101_17464_2527	HPNQYKTHSQHSLPWN	14301
SN1052_332_C7M91ACXX_2_1101_3557_2116	GANTTSDPSAEADYQH	14097
SN1052_332_C7M91ACXX_2_1101_2245_2880	GTKPQHNTTEPP**KH	13594
SN1052_332_C7M91ACXX_2_1101_13885_2065	ASTN*AAKHYEHDPSN	13291
SN1052_332_C7M91ACXX_2_1101_3180_2896	VPTTDPSSPNEHQDKH	13073
SN1052_332_C7M91ACXX_2_1101_8661_4005	IPNYEHNRIHTMLNRI	12525
SN1052_332_C7M91ACXX_2_1101_2822_3662	EATKSSHNHTSN*P*H	12091
SN1052_332_C7M91ACXX_2_1101_6113_3304	PSNSDN*HDHSTSEKN	11874
SN1052_332_C7M91ACXX_2_1101_4422_3049	LSTPNTKHLLNVV*EF	11859
SN1052_332_C7M91ACXX_2_1101_17896_2317	SPNTQKSYIHRIKKSI	11760
SN1052_332_C7M91ACXX_2_1101_11241_5775	PTTPYPAPAHHNTQKN	10845
SN1052_332_C7M91ACXX_2_1101_11209_2978	LSKTASENEKNSDPRA	10665
SN1052_332_C7M91ACXX_2_1101_2433_4176	FPNPS**PSSNST*KH	9010

前ページの表と比較すると、「配列は異なるが翻訳すると同じアミノ酸配列になる塩基配列」が多いことが分かった。
→ **unique.seqs** がアミノ酸配列に対応していないことでこの結果になっている。つまり上の表は誤り。

→アミノ酸配列用のunique.seqs を作るか、当初考えていた合算を行うしかなくなったようだ。

(181018) 色々やっていたら、アミノ酸配列とその配列がいくつあるかのカウント数の表（多い順にソート済み）のスクリプトが書けた。

input: アミノ酸配列のFASTAファイル

aminouniquesort.py ↓

```
amino = ''
data = []

with open('idamino2.fasta','r') as f, open('aminounique.csv', 'w') as output_file:
    for line in f:
        if line.startswith('>'):
            line = line.replace('\n', '\t')
        if not line.startswith('>'):
            list = [line.strip('\n')]
            data += list
            amino += line
            amino = amino.replace('\n', '\t')
            amino = ''
    else:
        from collections import Counter
        counter = Counter(data)
        for cnt in counter.most_common():
            count = str(cnt)
            count = count.replace(')','\n')
            count = count.strip('(')
            count = count.strip('\'')
            count = count.replace(',','\t')
            count = count.replace('\'\t','\t')
            output_file.write(count)
```

アウトプット↓

1	2
FATSNAEEKSEYKS*N	53464
LSTPAYAPHHDSSSEN	36615
ITNNPHYS*SS*ENEF	24155
RTTAHHTTTPPNEAEN	17294
VPTDAAPKQPQPYYKN	15488
GANTTSDPSAEADYQN	13825
GTKPQHNTTEPP**KN	13021
ASTN*AAKHYEHDQ*N	12751
YSKN*HPTY*SNQYKN	11195
PTTPYPAPAHHNTQKN	10560
LSKTASENEKNSDPRA	10437
EATKSSHNHTSN*P*N	8416
SPTPYPEATPTTPEKN	8332
GPN*TNHQKQT*EK*N	8209
NTKYSNYSQIRKQREF	7622
TANSAANPP*HQ*TQN	7147

15p の表と比べてみると合っているようなので、これを活用してIDも加えた表を作りたい。
あと、そろそろ作ったものをGalaxy にツールとして入れることを考える。

文字列関連
https://qiita.com/tomotaka_ito/items/594ee1396cf982ba9887

https://note.nkmk.me/python-list-str-select-replace/

https://note.nkmk.me/python-grep-like/

https://www.lifewithpython.com/2015/04/python-search-string-pattern.html

引数
https://algorithm.joho.info/programming/python/command-line-arguments-py/

・インプットと比較　濃縮率
・濃縮率出した、コントロールとリダイフェンサンプルで比較 → コントロールで高いやつを除外する

181112
各サンプル（7つ）のアミノ酸配列とカウント数の表を出した。

Multi-Join → Sort on data（2列目で多い順にソート）

1	2	3	4	5	6	7	8
FATSNAEEKSEYKS*N	53465	69472	77418	88585	71736	109413	78252
LSTPAYAPHHDSSSEN	36615	38134	35061	29750	33093	39872	33915
ITNNPHYS*SS*ENEF	24155	24729	23154	18144	19763	25569	21351
RTTAHHTTTPPNEAEN	17294	15290	17098	15878	19718	18838	16470
VPTDAAPKQPQPYYKN	15488	10368	11542	13356	13120	19198	11836
GANTTSDPSAEADYQN	13825	10423	12261	12334	13254	15031	9932
GTKPQHNTTEPP**KN	13021	13867	12689	14936	9853	16734	14145
ASTN*AAKHYEHDQ*N	12752	12349	12068	13075	11057	17579	10553
YSKN*HPTY*SNQYKN	11195	8726	9818	8334	5976	12138	8448
PTTPYPAPAHHNTQKN	10560	11334	9137	10492	9487	10056	7866
LSKTASENEKNSDPRA	10437	10974	12501	9363	9780	8877	9167
EATKSSHNHTSN*P*N	8416	8920	10228	8126	7998	10416	8523
SPTPYPEATPTTPEKN	8332	5520	7275	5877	8196	6805	5629
GPN*TNHQKQT*EK*N	8209	4708	5033	2054	5858	6247	6069
NTKYSNYSQIRKQREF	7622	6767	5778	5576	7477	12527	7109
TANSAANPP*HQ*TQN	7147	4187	6481	8311	4841	3611	5533
PSTPPSTQKSTAPKKN	7048	6413	8467	6835	4262	8975	7145
PPNDHTTTPTTKDA*N	6673	6064	7041	4657	4396	6987	5246
QTTYE*ASS*HSH*KN	6431	6721	6452	4819	3882	6385	6132
*TKKTEKPEPNETKEF	6293	4185	3511	5257	4652	5071	3271
LAIPHSQQLLHTNTRI	6242	5565	3311	5484	4165	8126	4804
	6012	6061	6171	6012	5634	6831	5829
RPNNA*T*EQNDANEN	5195	6204	6979	5321	4274	7042	4305
APKATA*TAQEDQKQN	5111	6208	6787	5213	7714	6286	5146
NTKTNYYQSYYQTN*N	5040	1983	2080	1928	3314	4535	3010
LTTSYT*PASNK*AKN	4823	4637	3310	4717	6403	6280	3706
DPTDTKHYSTHPQ*KN	4758	4018	4346	4924	3204	6151	4206
IPNYEHNRIHTMLNRI	4706	3967	3453	3006	5221	2449	4171
GTKESQSTYAQHNPEN	4659	4570	3981	4320	6359	4334	4772
QTKMQTPYCRII*KNS	4648	3155	4611	3508	2305	4219	3865
PTKDQDQAAYTQPKKN	4550	3516	3743	2913	3708	3275	3352
IPNNTSYP**YPTNQN	4373	3948	4803	3165	4536	6082	5157
PSNQPPSPYQNSDPRA	4208	3285	2357	3602	2817	3782	3477

各サンプルが何なのか分からないので、

KMT-1：インプット
KMT-2~4：薬剤あり
KMT-5~7：コントロール

として続きを行う。

multi-join → sort → compute で濃縮率を出す予定がcompute が上手くいかなかった→multi join の段階でなかったものをNONEではなく0にすると上手くいった。

multijoin → compute → sort(c4)

できた6ファイルを、
1つ目~3つ目、4つ目~6つ目でmultijoin(c4) → compute((c2+c3+c4)/3) → sort(c5) → cut(c1,c5)

できた2ファイルを、
multijoin → filter(c3<1.05) → sort(c2)

1	2	3
ATTEP*DDPP*DEK*N	233.333333333	0.333333333333
WPNQEPD*T*TAGD*N	164.666666667	1.0
ISKTIRRTTIRS*KNS	155.333333333	0.666666666667
MATD*PSQKCE*EQ*N	155.0	0.333333333333
RSKYNHSADQEIKNSD	148.666666667	0.333333333333
AANNDTSDHNTTKQ*N	146.666666667	0.333333333333
TTKNENSPKQEKPAKN	140.666666667	1.0
FAMYPTDQ*DKSYTQN	127.666666667	0.333333333333
FPTSP*HN*TNNTNWN	115.0	0.333333333333
FATDAPY*RSVTRTRE	88.3333333333	0.0
GPTAYYVTPYYAYPKN	77.0	0.333333333333
TPKHDDPLQQRIRIRI	74.6666666667	0.0

正しいデータで正しくやれば2列目が1より大きいものが薬剤に結合するファージなのかなと思われる。

(181130)
塩基配列の FASTA ファイルをインプットとして、ランダム領域のみにしてから（アダプター配列の前後およびアダプター配列を取り除いてから）、アミノ酸翻訳を行い、アウトプットとしてアミノ酸配列のFASTA ファイルを出すスクリプトを作成した。

nuc2aminoadremove.py↓

```
import sys

DNA2Protein = {
    'TTT': 'F', 'TCT': 'S', 'TAT': 'Y', 'TGT': 'C',
    'TTC': 'F', 'TCC': 'S', 'TAC': 'Y', 'TGC': 'C',
    'TTA': 'L', 'TCA': 'S', 'TAA': '*', 'TGA': '*',
    'TTG': 'L', 'TCG': 'S', 'TAG': '*', 'TGG': 'W',
    'CTT': 'L', 'CCT': 'P', 'CAT': 'H', 'CGT': 'R',
    'CTC': 'L', 'CCC': 'P', 'CAC': 'H', 'CGC': 'R',
    'CTA': 'L', 'CCA': 'P', 'CAA': 'Q', 'CGA': 'R',
    'CTG': 'L', 'CCG': 'P', 'CAG': 'Q', 'CGG': 'R',
    'ATT': 'I', 'ACT': 'T', 'AAT': 'N', 'AGT': 'S',
```

```
    'ATC': 'I', 'ACC': 'T', 'AAC': 'N', 'AGC': 'S',
    'ATA': 'I', 'ACA': 'T', 'AAA': 'K', 'AGA': 'R',
    'ATG': 'M', 'ACG': 'T', 'AAG': 'K', 'AGG': 'R',
    'GTT': 'V', 'GCT': 'A', 'GAT': 'D', 'GGT': 'G',
    'GTC': 'V', 'GCC': 'A', 'GAC': 'D', 'GGC': 'G',
    'GTA': 'V', 'GCA': 'A', 'GAA': 'E', 'GGA': 'G',
    'GTG': 'V', 'GCG': 'A', 'GAG': 'E', 'GGG': 'G'
}

p = ''
head = sys.argv[3]
tail = sys.argv[4]

with open(sys.argv[1], 'r') as f, open(sys.argv[2], 'w') as output_file:
    for line in f:
        if line.startswith('>'):
            output_file.write(line)
        else:
            for i in range(0, int(len(line) / 3)):
                codon = line[3 * i:3 * i + 3]
                if (codon in DNA2Protein):
                    p += DNA2Protein[codon]
                else:
                    break
            line.replace(line, p)
            if head in p:
                p = str(p.split(head)[1])
            if tail in p:
                p = str(p.split(tail)[0])
            p += '\n'
        output_file.write(p)
        p = ''
```

実行方法としては、コマンドライン上で、

＄python nuc2aminoadremove.py（インプットファイル名）（アウトプットファイル名）