Part-02-Featurization-Mature-miRNA

March 9, 2021

[1]:
```
import pandas as pd
import numpy as np

from tqdm import tqdm
from collections import Counter
```

[2]:
```
df_mature_mirna = pd.read_csv('df_mature_mirna_sequence.csv')
```

[3]:
```
df_mature_mirna.head()
```

[3]:

	Sequences	Species
0	UGAGGUAGUAGGUUGUAUAGUU	Homo sapiens
1	CUAUACAAUCUACUGUCUUUC	Homo sapiens
2	CUGUACAGCCUCCUAGCUUUCC	Homo sapiens
3	UGAGGUAGUAGGUUGUGUGGUU	Homo sapiens
4	CUAUACAACCUACUGCCUUCCC	Homo sapiens

[4]:
```
df_mature_mirna.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4634 entries, 0 to 4633
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   Sequences  4634 non-null   object
 1   Species    4634 non-null   object
dtypes: object(2)
memory usage: 72.5+ KB
```

[5]:
```
all(df_mature_mirna.isnull())
```

[5]: True

- There's no null entries

1 Featurization

[6]:
```
sequences = np.array(df_mature_mirna.Sequences)
```

1.1 Creating Kmers (Mono, Di, Tri and Quad)

[7]:
```
bi_kmer_pair = list()
tri_kmer_pair = list()
quad_kmer_pair = list()
mono_kmer = list()

for i in ['A', 'U', 'G', 'C']:
    mono_kmer.append(str(i))
    for k in ['A', 'U', 'G', 'C']:
        bi_kmer_pair.append(str(i+''+k))
        for j in ['A', 'U', 'G', 'C']:
            tri_kmer_pair.append(str(i+''+k+''+j))
            for q in ['A', 'U', 'G', 'C']:
                quad_kmer_pair.append(str(i+''+k+''+j+''+q))
```

[8]:
```
print(f"Single Nucleotide: {mono_kmer[0:4]}\n")
print(f"Two Pairs Nucleotides: {bi_kmer_pair[0:4]}\n")
print(f"Three Pairs Nucleotides: {tri_kmer_pair[0:4]}\n")
print(f"Four Pairs Nucleotides: {quad_kmer_pair[0:4]}\n")
```

```
Single Nucleotide: ['A', 'U', 'G', 'C']

Two Pairs Nucleotides: ['AA', 'AU', 'AG', 'AC']

Three Pairs Nucleotides: ['AAA', 'AAU', 'AAG', 'AAC']

Four Pairs Nucleotides: ['AAAA', 'AAAU', 'AAAG', 'AAAC']
```

1.2 Extracting Kmer counts

[9]:
```
def build_kmers(sequence, ksize):
    kmers = []
    n_kmers = len(sequence) - ksize + 1

    for i in range(n_kmers):
        kmer = sequence[i:i + ksize]
        kmers.append(kmer)

    return np.array(kmers)

def kmerCounts(sequence, kmers, kmer_length=2):
```

```
    kmer_count = Counter(build_kmers(sequence, kmer_length))

    for pair in kmers:
        kmer_count[pair] = kmer_count[pair]

    # Sorting the order of keys
    kmer_count = dict(sorted(kmer_count.items(),
                             key=lambda item: item[0]))

    return kmer_count

def df_kmers(sequence, kmers, kmer_length=2):

    kmer_counts = list()

    for seq in sequence:
        kmer_count = kmerCounts(seq, kmers, kmer_length)
        kmer_counts.append(list(kmer_count.values()))

    kmer_counts = np.array(kmer_counts)

    kmer_df = pd.DataFrame(kmer_counts, columns=list(kmer_count.keys()))

    return kmer_df
```

[10]:
```
# Mono mer
one_mer_df = df_kmers(sequences, mono_kmer, 1)
one_mer_df.head(3)
```

[10]:

	A	C	G	U
0	5	0	8	9
1	5	6	1	9
2	3	9	3	7

[11]:
```
two_mer_df = df_kmers(sequences, bi_kmer_pair, 2)
two_mer_df.head(3)
```

[11]:

	AA	AC	AG	AU	CA	CC	CG	CU	GA	GC	GG	GU	UA	UC	UG	UU
0	0	0	4	1	0	0	0	0	1	0	2	5	4	0	2	2
1	1	2	0	2	1	0	0	4	0	0	0	1	3	3	1	2
2	0	1	2	0	1	3	0	4	0	2	0	1	2	2	1	2

[12]:
```
three_mer_df = df_kmers(sequences, tri_kmer_pair, 3)
three_mer_df.head(3)
```

[12]:

	AAA	AAC	AAG	AAU	ACA	ACC	ACG	ACU	AGA	AGC	…	UCG	UCU	UGA	UGC	\
0	0	0	0	0	0	0	0	0	0	0	…	0	0	1	0	

```
1    0    0    0    1    1    0    0    1    0    0  …    0    2    0    0
2    0    0    0    0    1    0    0    0    0    2  …    0    0    0    0

   UGG  UGU  UUA  UUC  UUG  UUU
0    0    1    0    0    1    0
1    0    1    0    1    0    1
2    0    1    0    1    0    1

[3 rows x 64 columns]
```

[13]:
```
# quad_kmer_pair
four_mer_df = df_kmers(sequences, quad_kmer_pair, 4)
four_mer_df.head(3)
```

[13]:
```
   AAAA  AAAC  AAAG  AAAU  AACA  AACC  AACG  AACU  AAGA  AAGC  …  UUCG  \
0     0     0     0     0     0     0     0     0     0     0  …     0
1     0     0     0     0     0     0     0     0     0     0  …     0
2     0     0     0     0     0     0     0     0     0     0  …     0

   UUCU  UUGA  UUGC  UUGG  UUGU  UUUA  UUUC  UUUG  UUUU
0     0     0     0     0     1     0     0     0     0
1     0     0     0     0     0     0     1     0     0
2     0     0     0     0     0     0     1     0     0

[3 rows x 256 columns]
```

1.3 zCurve

[14]:
```
def zCurve(sequence):

    a_count = sequence.count('A')
    u_count = sequence.count('U')
    g_count = sequence.count('G')
    c_count = sequence.count('C')

    x_axis = (a_count + g_count) - (c_count + u_count)
    y_axis = (a_count + c_count) - (g_count + u_count)
    z_axis = (a_count + u_count) - (g_count + c_count)

    return (x_axis, y_axis, z_axis)
```

[15]:
```
zcurve_list = []
for sequence in df_mature_mirna.Sequences:
    zcurve_list.append(zCurve(sequence))

df_zcurve = pd.DataFrame(np.array(zcurve_list), columns=['x_axis', 'y_axis',␣
 ↪'z_axis'])
```

[16]:
```
df_zcurve.head()
```

[16]:

	x_axis	y_axis	z_axis
0	4	-12	6
1	-9	1	7
2	-10	2	-2
3	4	-16	2
4	-10	8	0

1.4 GC Content

[17]:
```
def gcContent(sequence):

    a_count = sequence.count('A')
    u_count = sequence.count('U')
    g_count = sequence.count('G')
    c_count = sequence.count('C')

    gc_cont = ((g_count + c_count)/(a_count+u_count+g_count+c_count))  * 100

    return gc_cont
```

[18]:
```
gc_list = []
for sequence in sequences:
    gc_list.append(gcContent(sequence))

gc_content = pd.DataFrame(np.array(gc_list), columns=['gc_content'])
gc_content.head()
```

[18]:

	gc_content
0	36.363636
1	33.333333
2	54.545455
3	45.454545
4	50.000000

1.5 Combing all the features

[19]:
```
df_mature_mirna_features = pd.concat([one_mer_df, two_mer_df,
                                      three_mer_df, four_mer_df,
                                      df_zcurve, gc_content,
                                      df_mature_mirna.Species], axis=1)
```

[20]:
```
df_mature_mirna_features.head()
```

[20]:

	A	C	G	U	AA	AC	AG	AU	CA	CC	…	UUGU	UUUA	UUUC	UUUG	UUUU	\
0	5	0	8	9	0	0	4	1	0	0	…	1	0	0	0	0	

```
1  5   6   1  9   1   2   0   2   1   0  …     0     0     1     0     0
2  3   9   3  7   0   1   2   0   1   3  …     0     0     1     0     0
3  3   0  10  9   0   0   3   0   0   0  …     1     0     0     0     0
4  5  10   1  6   1   3   0   1   1   4  …     0     0     0     0     0

   x_axis  y_axis  z_axis  gc_content       Species
0       4     -12       6   36.363636  Homo sapiens
1      -9       1       7   33.333333  Homo sapiens
2     -10       2      -2   54.545455  Homo sapiens
3       4     -16       2   45.454545  Homo sapiens
4     -10       8       0   50.000000  Homo sapiens

[5 rows x 345 columns]
```

[21]:
```
df_mature_mirna_features.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4634 entries, 0 to 4633
Columns: 345 entries, A to Species
dtypes: float64(1), int32(343), object(1)
memory usage: 6.1+ MB
```

[22]:
```
df_mature_mirna_features.to_csv("df_mature_mirna_sequences_features.csv",␣
↪index=False)
```