tutorial-6

May 19, 2020

[72]:
```
# Packages
# from matplotlib import (pyplot as plt, lines)
# import seaborn as sns
import numpy as np
import os
import pandas as pd
```

[63]:
```
# Globals
BASE_DIR = "/home/josh/PycharmProjects/eces-450/tutorial/data/algae-genome/"


# Given files
print("Assembly Dir:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR, 'assembly'))␣
 ↪if file.endswith('.fa')))
print("\nRead Dir:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR, 'reads'))))
```

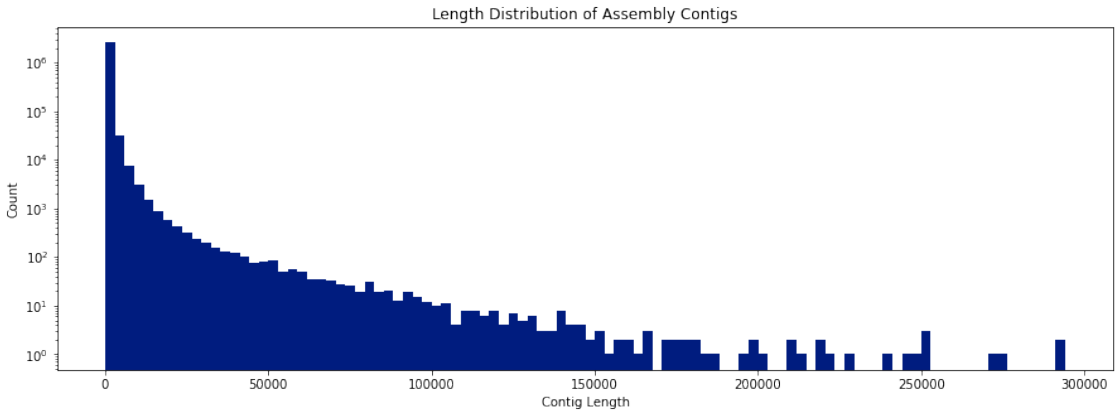
```
Assembly Dir:
simple.contig.fa

Read Dir:
CSJP002C_R1.fastq
CSJP002A_R2.fastq
CSJP002B_R2.fastq
CSJP002A_R1.fastq
CSJP002B_R1.fastq
CSJP002C_R2.fastq
```

[73]:
```
# Read Assembly File
fn = os.path.join(BASE_DIR, 'assembly', 'simple.contig.fa')
with open(fn, "r") as fh:
    lines = fh.readlines()

# Create list: contig_lengths
contig_lengths = []
contigs = []
```

```
i = 0
for line in lines:
    if line[0] == '>':
        contigs.append(line[1:])  # grab record contig id
    elif i < 1000:
        contig_lengths.append(len(line))  # grab record sequence length
    if i<2:
        print(line[0:250], end='')  # print the first record
        i+=1

# Plot the sequence lengths
plt.style.use('seaborn-dark-palette')
fig = plt.figure(figsize=(15, 5))
plt.hist(contig_lengths, bins=100, log=True)
plt.title("Length Distribution of Assembly Contigs")
plt.xlabel("Contig Length")
plt.ylabel("Count")
```

```
>contig-65_0
TGGCAGGCGATGTCGGAGCGCAAACTCCGCCGCCAGCGCGTCATCTTCTACATCTCACGTGGGTTGGAGAACGCCCCGAC
AAAGTTCCAGACCACCCCCGCCCGTCGCGACTACTACAACACGCTGCTCGACCAGCTTCAGACCGAGTTCGAGCACGTTC
ATCGGCTGCTGCTGGAAATCTTCGGTTCAGGCGGCGCGCGCGTGTTGCCGATGACGGATGCCGACCACTTCCGGCACTAC
AAACGGTTTC
```

[73]:
```
Text(0, 0.5, 'Count')
```



[74]:
```
# After running bwa index on the assembly file, several new files are created␣
 ↪which constitute the indexed assembly
print("Assembly Dir:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR,␣
 ↪'assembly'))))
```

```
Assembly Dir:
```

```
simple.contig.fa.amb
simple.contig.fa.ann
simple.contig.fa.pac
simple.contig.fa.sa
simple.contig.fa.bwt
simple.contig.fa
```

bwa mem

The BWA-MEM algorithm performs local alignment. It may produce multiple primary alignments for different part of a query sequence. This is a crucial feature for long sequences. However, some tools such as Picard's markDuplicates does not work with split alignments. One may consider to use option -M to flag shorter split hits as secondary.

[ ]:
```
#!/bin/bash
#### Create a map from the reads to the newly indexed assembly-file
#### This took 10.5 hours to complete on proteus

BASE_DIR="./"
samples=(2A 2B 2C)

for sample in ${samples[@]}
    do
        echo CSJP00${sample}_R1.fastq
        bwa mem ${BASE_DIR}assembly/simple.contig.fa ${BASE_DIR}reads/
 ↪CSJP00${sample}_R1.fastq ${BASE_DIR}reads/CSJP00${sample}_R2.fastq |␣
 ↪samtools view -b -o ${BASE_DIR}mapped/$sample.bam
done
```

[4]:
```
print("\nMapped:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR, 'mapped'))))
```

```
Mapped:
2C.bam
2A.bam
2B.bam
```

[48]:
```
# Sort the bam files for rapid processing, can also be run on proteus

print("\nSorted:")
print('\n'.join(file for file in os.listdir(os.path.join(BASE_DIR,'sorted'))))
```

```
Sorted:
2B.sorted.bam
2C.sorted.bam
2A.sorted.bam
```

[86]:
```
# Read Depth Matrix
fn = os.path.join(BASE_DIR, 'depth', 'depth_matrix.tab')
with open(fn, 'r') as fh:
    df = pd.read_csv(fh, delimiter='\t')
df.head(50)  # Show
```

[86]:

	contigName	contigLen	totalAvgDepth	2A.sorted.bam	2A.sorted.bam-var	\
0	contig-65_0	294096	29.6914	0.030421	0.039280	
1	contig-65_1	293378	34.9726	0.059582	0.078504	
2	contig-65_2	276350	29.1077	0.039613	0.054826	
3	contig-65_3	273109	23.3486	0.017285	0.022174	
4	contig-65_4	251804	17.5704	17.559800	25.608100	
5	contig-65_5	251606	29.6040	0.027420	0.038355	
6	contig-65_6	251169	22.7825	0.019413	0.027571	
7	contig-65_7	247969	30.5614	0.035001	0.049680	
8	contig-65_8	246701	36.4706	0.045220	0.061850	
9	contig-65_9	239899	27.7628	0.036104	0.050767	
10	contig-65_10	227742	29.8739	0.034461	0.048181	
11	contig-65_11	220666	23.1814	0.020311	0.030451	
12	contig-65_12	220439	33.5583	0.052762	0.073962	
13	contig-65_13	217682	15.3589	0.019064	0.024993	
14	contig-65_14	213067	36.2058	0.039062	0.053416	
15	contig-65_15	210155	12.5545	0.030856	0.049785	
16	contig-65_16	209169	29.7595	0.021931	0.031322	
17	contig-65_17	200413	30.7011	0.036851	0.048552	
18	contig-65_18	199480	29.6988	0.029925	0.036689	
19	contig-65_19	198319	23.7893	0.028052	0.044290	
20	contig-65_20	195195	16.9654	0.006224	0.008799	
21	contig-65_21	185758	22.6358	0.006659	0.010197	
22	contig-65_22	182600	23.3662	0.013412	0.020377	
23	contig-65_23	182193	14.7668	0.017331	0.025937	
24	contig-65_24	182140	16.1361	0.015869	0.018644	
25	contig-65_25	177867	15.1804	0.013932	0.023602	
26	contig-65_26	176673	34.4763	0.044357	0.060298	
27	contig-65_27	176345	24.1709	0.023406	0.028539	
28	contig-65_28	174764	15.0156	0.014123	0.021368	
29	contig-65_29	171331	36.4527	0.055450	0.079353	
30	contig-65_30	170660	14.6085	0.012768	0.015454	
31	contig-65_31	166828	35.5028	0.052118	0.065914	
32	contig-65_32	165776	16.4881	16.482600	26.711900	
33	contig-65_33	165370	28.4398	0.028356	0.045658	
34	contig-65_34	164401	45.1723	0.035342	0.044078	
35	contig-65_35	159866	13.8548	0.002962	0.004964	
36	contig-65_36	159152	22.7729	0.029245	0.047278	
37	contig-65_37	157651	23.4432	0.018241	0.021629	
38	contig-65_38	157544	46.7938	0.021449	0.031330	
39	contig-65_39	154927	17.0720	17.056300	22.524200	

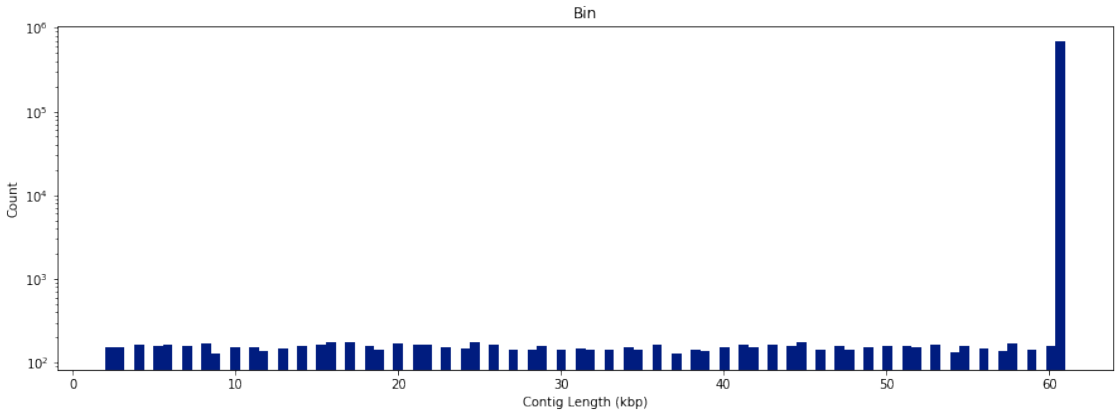
40	contig-65_40	152727	55.4262	0.032751	0.052662
41	contig-65_41	151203	14.4318	0.044296	0.070184
42	contig-65_42	150753	16.1581	0.003433	0.004621
43	contig-65_43	148980	45.6297	0.020386	0.025711
44	contig-65_44	148452	15.1980	0.010694	0.014188
45	contig-65_45	147041	47.2205	47.207500	74.345100
46	contig-65_46	146864	17.8696	17.858800	29.850700
47	contig-65_47	145638	34.9875	0.046409	0.066869
48	contig-65_48	144474	47.8003	0.030439	0.049214
49	contig-65_49	143374	16.7441	0.021428	0.030209

	2B.sorted.bam	2B.sorted.bam-var	2C.sorted.bam	2C.sorted.bam-var
0	10.666600	14.813800	18.994300	32.003500
1	0.780171	1.069020	34.132900	48.283600
2	10.410400	14.596500	18.657700	28.268900
3	13.233400	18.489200	10.098000	13.093500
4	0.006179	0.009984	0.004343	0.007939
5	10.820700	14.780800	18.755900	28.443500
6	12.983900	17.508200	9.779160	12.194100
7	10.750200	15.496000	19.776200	33.387500
8	0.847058	1.113460	35.578300	49.489800
9	10.166100	14.853000	17.560500	26.792200
10	10.905500	14.533400	18.933900	31.096900
11	13.258900	16.679400	9.902200	12.445300
12	0.767959	0.952616	32.737600	48.430000
13	9.684140	13.968400	5.655680	7.540150
14	0.927634	1.269840	35.239100	47.229700
15	3.795130	4.622480	8.728500	11.494100
16	10.832100	13.301600	18.905500	29.232800
17	10.963000	15.624600	19.701200	32.131400
18	10.645100	14.115000	19.023800	29.221800
19	13.478700	19.191000	10.282500	12.554000
20	15.488800	21.553100	1.470410	1.995600
21	19.639300	27.652300	2.989880	4.321350
22	13.125900	19.306800	10.226900	12.948700
23	14.648100	20.483100	0.101399	0.127204
24	4.641220	5.845650	11.479000	14.475100
25	9.764390	14.691900	5.402110	7.341430
26	0.824199	1.079130	33.607700	47.824800
27	13.847500	19.610500	10.299900	13.912000
28	14.886200	20.916200	0.115185	0.135600
29	0.885805	1.094090	35.511400	54.830400
30	9.251570	13.649900	5.344180	6.928070
31	1.305370	1.844200	34.145400	50.379500
32	0.001998	0.003755	0.003429	0.004632
33	10.534000	14.921300	17.877400	25.405200
34	38.542900	57.822200	6.594100	7.907430

35	8.979850	11.836000	4.871990	6.499230
36	13.148100	19.160000	9.595590	13.067100
37	13.427500	19.189200	9.997460	11.668100
38	39.651300	66.943400	7.121060	9.664440
39	0.006306	0.011082	0.009401	0.013839
40	32.707600	50.561300	22.685900	31.687000
41	4.061910	5.286980	10.325600	13.662700
42	14.632700	20.839100	1.521960	2.183480
43	38.731500	57.852000	6.877870	8.425290
44	9.539320	12.088600	5.647930	7.066240
45	0.011362	0.015225	0.001613	0.003215
46	0.006441	0.007745	0.004349	0.006198
47	0.762221	0.939616	34.178900	45.473300
48	39.867000	62.953800	7.902840	9.827320
49	4.796450	5.717060	11.926200	15.315700

[85]:
```
# Visualize Contig lengths in largest bin
fn = os.path.join(BASE_DIR, 'bins',"bin.10.fa")
# fn = os.path.join(BASE_DIR, 'bins',"bin.22.fa")
with open(fn, "r") as fh:
    lines = fh.readlines()

# Create list: contig_lengths
contig_lengths = []
for line in lines:
    if not line[0] == '>':
        contig_lengths.append(len(line))

# Plot the sequence lengths
plt.style.use('seaborn-dark-palette')
fig = plt.figure(figsize=(15, 5))
plt.hist(contig_lengths, bins=100, log=True)
plt.title("Bin")
plt.xlabel("Contig Length (kbp)")
plt.ylabel("Count")
```

[85]:
```
Text(0, 0.5, 'Count')
```



[80]:
```
# Binned data
fn = os.path.join(BASE_DIR, 'bins',"bin.10.fa")
with open(fn, "r") as fh:
    lines = fh.readlines()

# Create list: contigs
contigs = []
for line in lines:
    if line[0] == '>':
        contigs.append(line[1:].strip())
```