A1163_LNCRNA_cluster_genome_50000_GITHUB

August 21, 2022

1 To see if LncRNAs are randomly distributed across genome using the genome cut into 50kb chunks.

2 IN BASH

2.1 To make a chromosome bed grab the chromosome descriptions from the GFF file

cat Aspergillus_fumigatusa1163.ASM15014v1.53.gff3| awk '{if ($3=="supercontig") print $0}'>A1163_chromosome.gff

2.2 Convert to a bed and sort

conda activate GFF_utils gff2bed < A1163_chromosome.gff> A1163_new_chr.bed sort -V -k1,1 -k2,2 A1163_new_chr.bed>A1163_new_chr.sorted.bed

2.3 Then chop into 50kb bits using bedops and sort

bedops –chop 50000 A1163_new_chr.sorted.bed >A1163_50000bit.bed

2.4 now get the region LncRNA transcripts cover on the genome

cat A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.gtf|awk '$3~ "transcript" {print $0}'> transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.gtf

2.5 Make into a bed file

gtf2bed < transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.gtf> transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.bed sort -V -k1,1 -k2,2 transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.bed >transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.sorted.bed conda deactivate ## now intersect the chopped 50kb regions with the positions of LncRNAs using bedtools conda activate aligners bedtools intersect -a transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.sorted.bed -b A1163_50000bit.bed -wa -wb > LncRNA_A116350k.csv conda deactivate

2.6 Do not want to include regions with assembly gaps

2.7 grab assembly gaps from the annotation, make a bed and sort

cat Aspergillus_fumigatusa1163.ASM15014v1.53.gff3|grep ena_assembly_gap >assembly_gap.gff conda activate GFF_utils gff2bed < assembly_gap.gff> assembly_gap.bed conda deactivate sort -V -k1,1 -k2,2 assembly_gap.bed>A1163_assembly_gap.sorted.bed

2.8 Intersect with the 50kb chunks

conda activate aligners bedtools intersect -a A1163_assembly_gap.sorted.bed -b A1163_50000bit.bed -wa -wb > A1163discard_assembly.csv conda deactivate

3 Give each of the chunks a new unique name by opening up the csv files in LibreOffice/excel - "chunk names"

A1163discard_assembly.csv LncRNA_A116350k.csv and making a copy of A1163_50000bit.bed named A1163_50000bit.csv ie see line below use the =CONCAT(K1:M1) to string together the chromosome start and stop of each chunk. DS499594 0 50000 DS499594050000

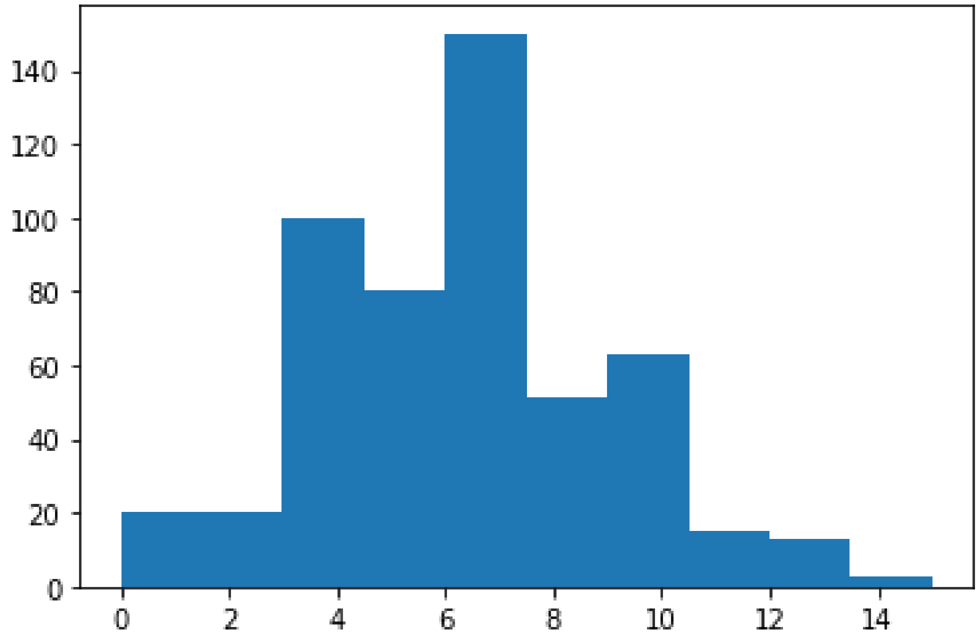
[1]:
```
#libraries needed
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats
from scipy.stats import normaltest
```

[2]:
```
# done at 50000 get rid of slices that are too small or have assembly gaps
# Column 13 was the column my chunk names were in
df_discard=pd.read_csv("A1163discard_assembly.csv",header=None)
to_drop=list(set(df_discard.iloc[:,13].tolist()))
len(to_drop)

#drop regions with assembly gaps
df=pd.read_csv("LncRNA_A116350k.csv",header=None)
df=df[df.iloc[:,13].apply(lambda x:x not in to_drop)]
df.rename(columns={df.columns[13]: 'rename'},inplace=True)
df["length"]=df.iloc[:,12]-df.iloc[:,11]

#drop those under 50000
df=df[df.length==50000]

# 50kb chunks in AF293 genome drop regions with assembly gaps and are smaller␣
 ↪in length than 50k
df2=pd.read_csv("A1163_50000bit.csv", header=None)
df2=df2[df2.iloc[:,3].apply(lambda x:x not in to_drop)]
df2["length"] = df2.iloc[:,2]-df2.iloc[:,1]
df2=df2[df2.length==50000]
```

```
# count number of LncRNAs in each chunk
df3=df.groupby(by="rename").agg('count')

# is the number of usable chunks in the genome
a=df2.shape[0]

# number of chunks LncRNAs are in
b=df3.shape[0]


# get the counts of LncRNAs per 50kb chunk
# chunks with ones in
Lncr=df3.length.tolist()

#chunks with none
list_of_zeros = [0] * (a-b)

#combime the two
LNCR=list_of_zeros+Lncr
```

[3]:
```
plt.rcParams["figure.figsize"] = (3,2.5)
plt.hist(LNCR)
```

[3]:
```
(array([ 20.,  20., 100.,  80., 150.,  51.,  63.,  15.,  13.,   3.]),
 array([ 0. ,  1.5,  3. ,  4.5,  6. ,  7.5,  9. , 10.5, 12. , 13.5, 15. ]),
 <BarContainer object of 10 artists>)
```



```
[4]: # Check for normal distribution
     scipy.stats.shapiro(LNCR)
```

```
[4]: ShapiroResult(statistic=0.981400191783905, pvalue=3.7701638575526886e-06)
```

```
[5]: scipy.stats.normaltest(LNCR)
```

```
[5]: NormaltestResult(statistic=8.647468324359421, pvalue=0.013250312231583843)
```