A1163_genome_coverage_GH

August 21, 2022

1 To find how much LncRNAs, novel protein coding regions cover of the genome, on both strands. Can also see how much they overlap themselves.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

[2]:
```
import warnings
warnings.filterwarnings('ignore')
```

We need Chromosome regions

Mobile element regions

All genes - no transposon genes

CDS

Gene regions

genome region spanning LncRNAs

genome region spanning new potential protein coding regions

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

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

1.2 Convert to a bed and sort

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

1.3 Mobile transposable elements

cat Aspergillus_fumigatusa1163.ASM15014v1.53.gff3|grep ena_mobile_element> a1163_mobile.gff3 gff2bed < a1163_mobile.gff3> A1163_mobile.bed

1.4 All genes - no transposons

cat Aspergillus_fumigatusa1163.ASM15014v1.53.gff3|awk '$3~ "gene" {print $0}'>A1163_all_gene.gff3

1.5 need to make this into a bed file

gff2bed < A1163_all_gene.gff3>A1163_all_gene.bed cat A1163_all_gene.bed|grep transposon-related -v >A1163_all_gene_NT.bed sort -V -k1,1 -k2,2 A1163_all_gene_NT.bed>A1163_all_gene_NT.sorted.bed

1.6 CDS in A1163 annotation - no transposon

cat Aspergillus_fumigatusa1163.ASM15014v1.53.gff3|awk '$3~ "CDS{print $0}'> CDS_A1163.gff gff2bed < CDS_A1163.gff> CDS_A1163.bed cat CDS_A1163.bed|grep transposon-related -v >CDS_A1163_NT.bed

1.7 For potential LncRNAs

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 gtf2bed < transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.gtf> transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.bed

1.8 For potential protein coding genes

cat A1163PT_UX_6_8_TPM_CUTOFF_6_8.gtf|awk '$3~ "transcript" {print $0}'> transcript_A1163PT_UX_6_8_TPM_CUTOFF_6_8.gtf conda activate GFF_utils gtf2bed < transcript_A1163PT_UX_6_8_TPM_CUTOFF_6_8.gtf> transcript_A1163PT_UX_6_8_TPM_CUTOFF_6_8.bed

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

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

[3]:
```python
def coverage (coverage_pd,chromosome_number,chr_length,to_measure):
#put data into dataframes

    to_measure_2=to_measure[to_measure["chromo"]==chromosome_number]


    #make a column of range numbers used
    to_measure_2["numbers"] = [list(range(x,y)) for x,y in zip(to_measure_2.
↪loc[:,"start"], to_measure_2.loc[:,"stop"])]

    #make a dictionary of keys
    dict_keys = set(list(range(chr_length)))
    chr_cov=dict.fromkeys(dict_keys,0)
    for line in to_measure_2["numbers"]:
        for unit in range(len(line)):
            chr_cov[line[unit]]+=1
    chr_pd=pd.DataFrame.from_dict(chr_cov, orient='index')
    chr_pd.columns=["coverage"]
    coverage_pd["chromosome length"]+=chr_length
    coverage_pd["coverage_0"]+=sum(chr_pd["coverage"]==0)
    coverage_pd["coverage_m_0"]+=(sum(chr_pd["coverage"]>0))
    coverage_pd["coverage_m_1"]+=(sum(chr_pd["coverage"]>1))
    coverage_pd["coverage_m_2"]+=(sum(chr_pd["coverage"]>2) )
    return coverage_pd
```

[4]:
```
# Gaps in assembly
gap=pd.read_csv("A1163_assembly_gap.sorted.bed", sep="\t", header=None)
gap_sum=sum(gap.loc[:,2]- gap.loc[:,1])

#CDS coverage
#import A1163_New annotation protein coding make sure no transposons
annot_CDS=pd.read_csv("CDS_A1163_NT.bed", sep=("\t"),header=None)
annot_CDS2=annot_CDS.iloc[:,0:3]

#chromosomes
chromosome = pd.read_csv("A1163_new_chr.sorted.bed", sep=("\t"),header=None).
↪iloc[:,:3]
chromosome.columns= ["chromo", "start", "stop"]
chromosome["length"]= chromosome.stop - chromosome.start

#annotated gene coverage -allgenes including ncrna and tRNA
#import A1163_New annotation protein coding make sure no transposons
annot_gene=pd.read_csv("A1163_all_gene_NT.sorted.bed", sep=("\t"),header=None)
annot_gene2=annot_gene.iloc[:,0:3]
annot_CDS2.columns= ["chromo","start","stop"]

# LncRNA
UX_LncRNA=pd.read_csv("transcript_A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.bed",␣
↪sep=("\t"),header=None)
UX_LncRNA2=UX_LncRNA.iloc[:,0:3]
UX_LncRNA2.columns= ["chromo","start","stop"]

# POTPROT - potential new protein coding
```

```
UX_PotP=pd.read_csv("transcript_A1163PT_UX_6_8_TPM_CUTOFF_6_8.bed",␣
 ↪sep=("\t"),header=None)
UX_PotP2=UX_PotP.iloc[:,0:3]
UX_PotP2.columns= ["chromo","start","stop"]

# Transposons - from annotation
MOBS=pd.read_csv("A1163_mobile.bed", sep=("\t"),header=None)
MOBS=UX_PotP.iloc[:,0:3]
MOBS.columns= ["chromo","start","stop"]
```

[5]:
```
# All CDS coverage
coverage_pd = pd.DataFrame(np.zeros((1, 5)).astype(int),columns= ["chromosome␣
 ↪length", "coverage_0", "coverage_m_0","coverage_m_1","coverage_m_2"])
for i in range (0,len(chromosome)):
    coverage(coverage_pd,chromosome.iloc[i,0], chromosome.iloc[i,3],annot_CDS2)
# need to subtract annotn gaps
genome_length = int(coverage_pd["chromosome length"])-gap_sum
# reduce 0 by gap size
coverage_pd.iloc[:,1]=coverage_pd.iloc[:,1]-gap_sum
CDS_coverage=coverage_pd/genome_length *100
```

[6]:
```
# all gene coverage
annot_gene2.columns= ["chromo","start","stop"]
coverage_pd = pd.DataFrame(np.zeros((1, 5)).astype(int),columns= ["chromosome␣
 ↪length", "coverage_0", "coverage_m_0","coverage_m_1","coverage_m_2"])
for i in range (0,len(chromosome)):
    coverage(coverage_pd,chromosome.iloc[i,0], chromosome.iloc[i,3],annot_gene2)
# need to subtract annotn gaps
genome_length = int(coverage_pd["chromosome length"])-gap_sum
# reduce 0 by gap size
coverage_pd.iloc[:,1]=coverage_pd.iloc[:,1]-gap_sum
gene_coverage=coverage_pd/genome_length *100
```

[7]:
```
#Novel LncRNA coverage
coverage_pd = pd.DataFrame(np.zeros((1, 5)).astype(int),columns= ["chromosome␣
 ↪length", "coverage_0", "coverage_m_0","coverage_m_1","coverage_m_2"])
for i in range (0,len(chromosome)):
    coverage(coverage_pd,chromosome.iloc[i,0], chromosome.iloc[i,3],UX_LncRNA2)
# need to subtract annotn gaps
genome_length = int(coverage_pd["chromosome length"])-gap_sum
# reduce 0 by gap size
coverage_pd.iloc[:,1]=coverage_pd.iloc[:,1]-gap_sum
LncRNA=coverage_pd/genome_length *100
```

[8]:
```
# Novel potential protein coverage
coverage_pd = pd.DataFrame(np.zeros((1, 5)).astype(int),columns= ["chromosome␣
 ↪length", "coverage_0", "coverage_m_0","coverage_m_1","coverage_m_2"])
```

```
for i in range (0,len(chromosome)):
    coverage(coverage_pd,chromosome.iloc[i,0], chromosome.iloc[i,3],UX_PotP2)
# need to subtract annotn gaps
genome_length = int(coverage_pd["chromosome length"])-gap_sum
# reduce 0 by gap size
coverage_pd.iloc[:,1]=coverage_pd.iloc[:,1]-gap_sum
POTP=coverage_pd/genome_length *100
```

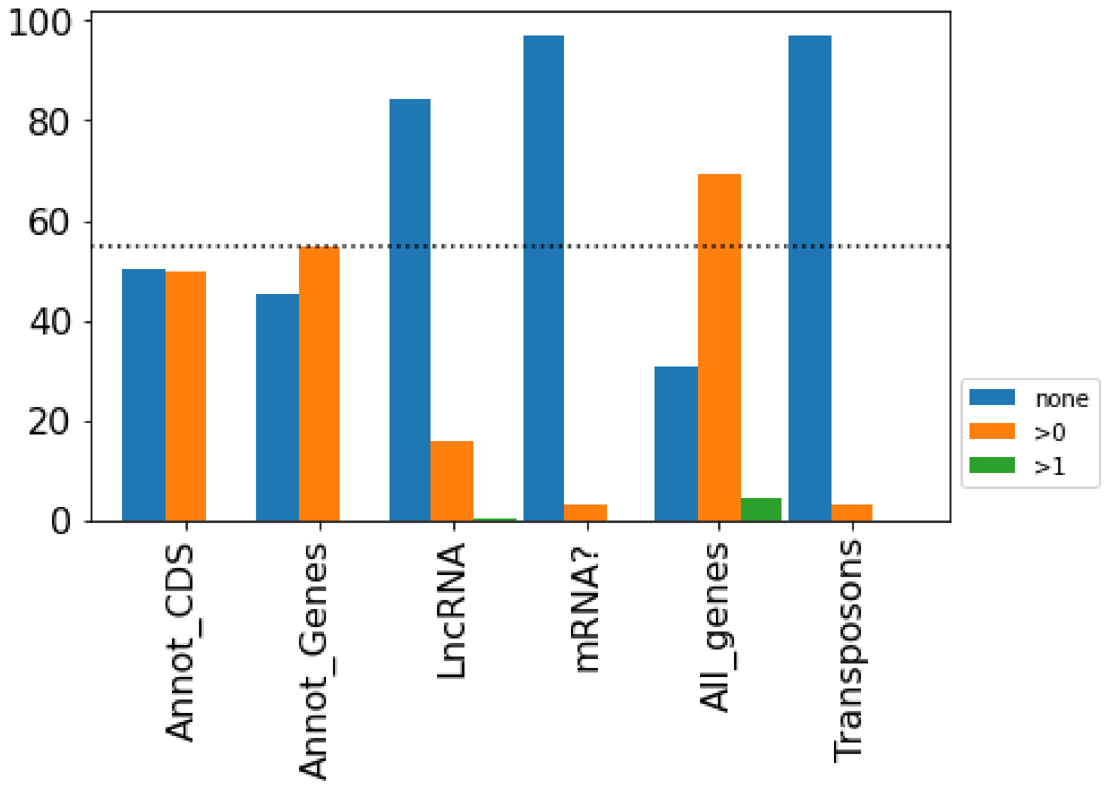
[9]:
```
# Mobile element coverage
coverage_pd = pd.DataFrame(np.zeros((1, 5)).astype(int),columns= ["chromosome␣
↪length", "coverage_0", "coverage_m_0","coverage_m_1","coverage_m_2"])
for i in range (0,len(chromosome)):
    coverage(coverage_pd,chromosome.iloc[i,0], chromosome.iloc[i,3],MOBS)
# need to subtract annotn gaps
genome_length = int(coverage_pd["chromosome length"])-gap_sum
# reduce 0 by gap size
coverage_pd.iloc[:,1]=coverage_pd.iloc[:,1]-gap_sum
MOBS_COVER=coverage_pd/genome_length *100
```

[10]:
```
# get coverage of all gene models
frames = [annot_gene2,UX_LncRNA2, UX_PotP2]
all_genes=pd.concat(frames)

coverage_pd = pd.DataFrame(np.zeros((1, 5)).astype(int),columns= ["chromosome␣
↪length", "coverage_0", "coverage_m_0","coverage_m_1","coverage_m_2"])
for i in range (0,len(chromosome)):
    coverage(coverage_pd,chromosome.iloc[i,0], chromosome.iloc[i,3],all_genes)
# need to subtract annotn gaps
genome_length = int(coverage_pd["chromosome length"])-gap_sum
# reduce 0 by gap size
coverage_pd.iloc[:,1]=coverage_pd.iloc[:,1]-gap_sum
ALL_coverage=coverage_pd/genome_length *100
```

[11]:
```
# Make data frame for plot  - concatenate data frame
A1163_coverage=pd.concat([CDS_coverage,␣
↪gene_coverage,LncRNA,POTP,ALL_coverage,MOBS_COVER])
A1163_coverage.columns=["length", "none", ">0", ">1", ">2"]
A1163_coverage.index=["Annot_CDS", "Annot_Genes", "LncRNA","mRNA?
↪","All_genes","Transposons"]

# PLot figure - no coverage, single coverage, both strands
A1163_coverage.iloc[:,1:4].plot(kind="bar",width=.95,figsize=(7, 5)).
↪legend(bbox_to_anchor=(1,0.3))
plt.xticks(size=16)
plt.yticks( size=16)
plt.axhline(y=54.951592, color='k', linestyle='dotted')
plt.tight_layout()
plt.show()
```


Annot_CDS
Annot_Genes
LncRNA
mRNA?
All_genes
Transposons
none
>0
>1