A1163_analysis_StringTie_Transcriptomes-GH

August 21, 2022

1 Plotting features of StringTie Transcriptomes

[1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

list_samples = pd.read_csv("csv_bed_list.txt", header=None)
samples=list_samples[0]
```

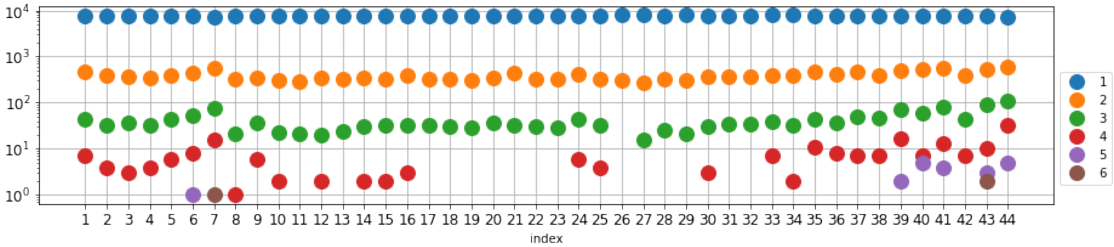
[2]:

```
# get the overaps of the longest transcripts with annotated genes on same␣
 ↪strand generated by get_longest_pt3.sh
# function calculates the number of annotated genes in the longest␣
 ↪transcript,and how many have 1,2,3
def plot_polycistrons3(x):
    df=pd.read_csv(samples[x], sep="\t", header=None)
    df.iloc[:,19]=df.iloc[:,19].str.split(";", expand=True)[0]
    c=list(range(0,df.shape[1]))
    d=([str(x) for x in c])
    df.columns=d
    df=df.drop_duplicates(["3","19"])
    gene_number=df.groupby(by="19").size()
    a,b = np.histogram(gene_number, bins=((np.max(gene_number))-1))
    return list(a)
# goes through each run and applies the function plot_polycistrons3 and stores␣
 ↪the results as a dictionary
list_all={}

for x in range(len(samples)):
    y=samples[x].split("A1163")[1]
    z=y.split("default")[0]
    list_all[z]=plot_polycistrons3(x)
```

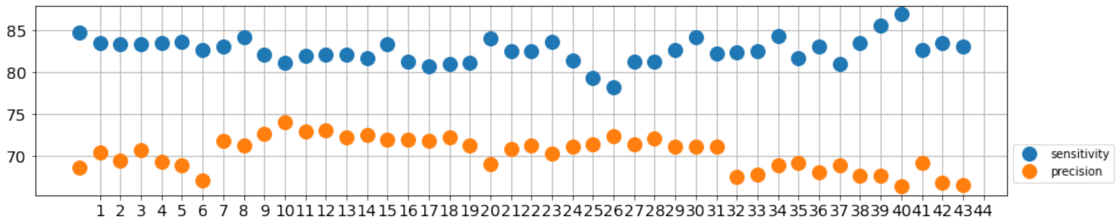
[3]:

```
# Makes a dataframe from the dictionary
# need to know the largest number of genes contained in a polycistron
df= pd.DataFrame.from_dict(list_all, orient='index')
df=df.replace(np.nan, 0)
```

```
df=df.astype('int32')
longest = max(len(item) for item in list_all.values())
df.columns=list(range(1,longest+1))
df["run"]=df.index.values.tolist()
df["index"]=list (range(1,45))
df=df.set_index('index')
# I always make an order column so I can make sure that nothing gets mixed up
df["order"]=df.index.values.tolist()

ax=df.iloc[:,0:6].
  ↪plot(style='o',ms=12,logy=True,fontsize=12,xticks=list(range(1,45)),␣
  ↪grid=True,figsize=(15,3)).legend(bbox_to_anchor=(1,0.7))
```

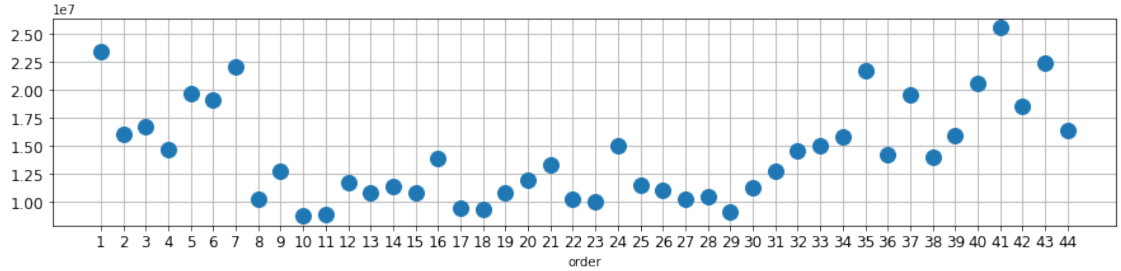


[4]:
```
# sensitivity and precision per base (extra and missed loci are also generated␣
  ↪by GFFcompare)
new_df=pd.read_csv("trial_output.txt", sep=" ", header=None).drop_duplicates()
new_df.columns = ["run","sensitivity","precision"]
new_df2= pd.merge( df,new_df, how='inner', left_on = 'run', right_on = 'run')
new_df2.iloc[:,8:10].plot(style='o',fontsize=14,ms=12,xticks=list␣
  ↪(range(1,45)), grid=True,figsize=(15,3)).legend(bbox_to_anchor=(1,0.3))
plt.show()
```



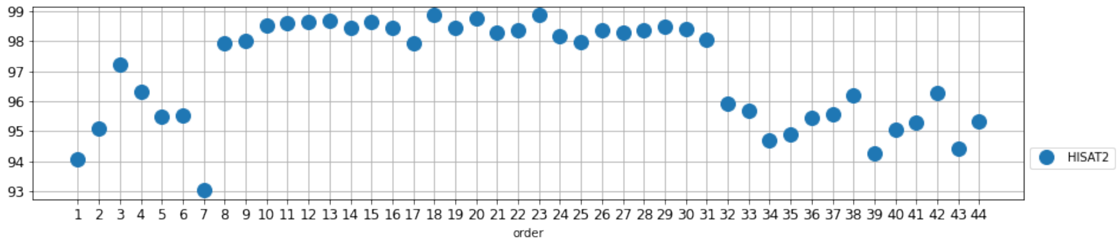
[5]:
```
# Map rates and number mapped are in the error messages when running HISAT2 on␣
  ↪CSF
# Number mapped
run_info=pd.read_csv("hisat2A1163.txt", header=None, sep=("\t"))
run_info.columns=["run", "mapped","HISAT2"]
```

```
new_df3= pd.merge(run_info,new_df2, how='inner', left_on = 'run', right_on =␣
 ↪'run')
new_df3=new_df3.sort_values(by=['order'])
new_df3=new_df3.reset_index(drop=True)
new_df3=new_df3.set_index('order',drop=False)
new_df3["mapped"]=new_df3["mapped"].astype('int64')
new_df3["mapped"].plot(style='o',fontsize=12,ms=12,xticks=list (range(1,45)),␣
 ↪grid=True,figsize=(15,3))
plt.show()
```



[6]:

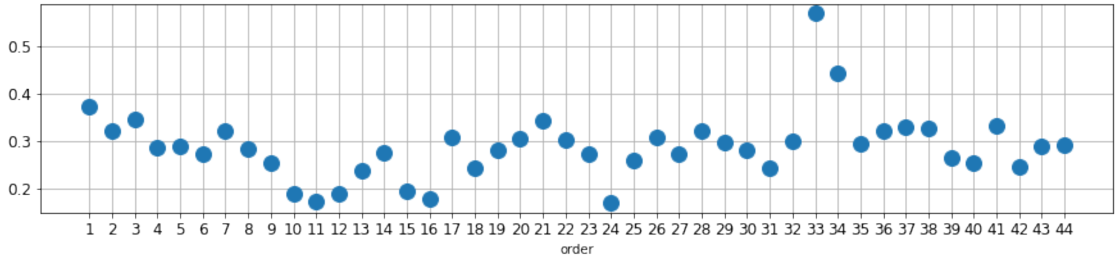
```
#Map rate
new_df3.loc[:,"HISAT2"]=new_df3.loc[:,"HISAT2"].astype('float')
new_df3.loc[:,"HISAT2"].plot(style='o',ms=12,fontsize=12,xticks=list␣
 ↪(range(1,45)), grid=True,figsize=(15,3)).legend(bbox_to_anchor=(1,0.3))
plt.savefig("A1163_HISAT2.png", dpi='figure')
plt.show()
```



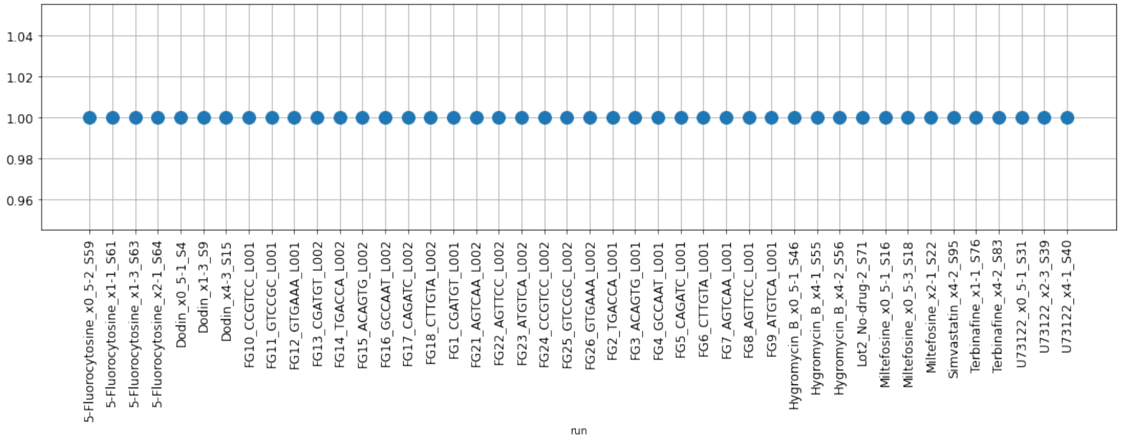
[7]:

```
#get overlap between introns on plus and minus strand of same run
df_intron=pd.read_csv("intron_overlap.csv")

df_intron.columns=["run","%_overlap"]
new_df5= pd.merge(new_df3, df_intron, how='inner', left_on = 'run', right_on =␣
 ↪'run')
new_df5=new_df5.set_index('order',drop=False)
new_df5["%_overlap"].plot(style='o',fontsize=12, ms=12,xticks=list␣
 ↪(range(1,45)), grid=True,figsize=(15,3), legend=False)
```

```
plt.show()
```



[8]:
```
# To get run labels for figure
new_df5=new_df5.set_index('run',drop=False)
new_df5["one"]=1
new_df5["one"].plot(style='o',fontsize=12, ms=12,xticks=list (range(0,44)),␣
 ↪grid=True,figsize=(15,6), legend=False)
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



[ ]: