A1163_UNION_TPM_GH

August 21, 2022

[1]:
```
#To filter preps on TPM > 0.5
#Libraries needed
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import csv
import scipy.stats
from scipy.stats import mannwhitneyu
```

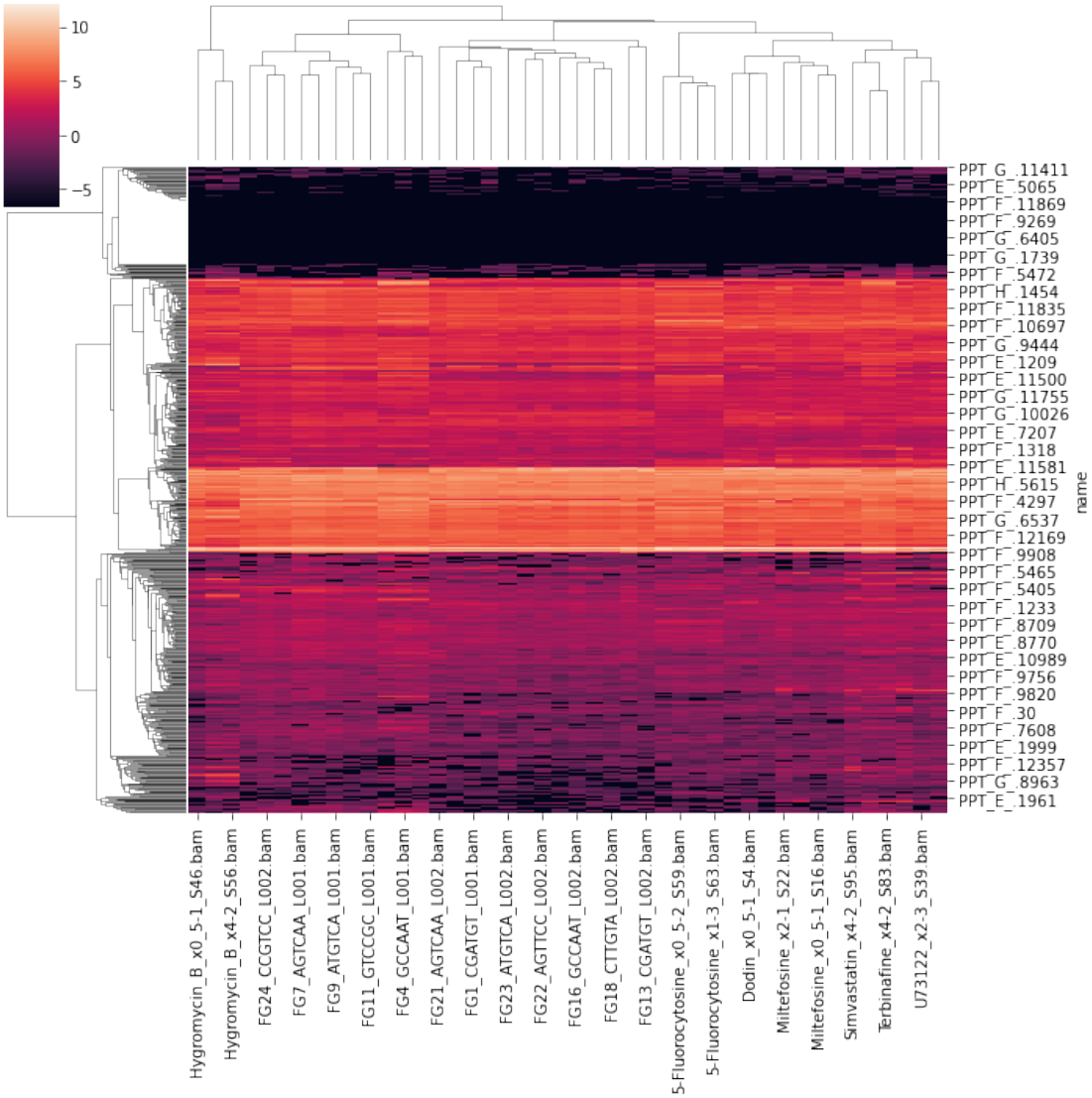
[2]:
```
#Need this info for getting data later
#LncRNAs pre cut-off
LNcRNA_X=pd.read_csv("A1163_ML_X_LncRNA_T_6_8.gtf", sep="\t",comment="#",␣
 ↪header=None )
LNcRNA_X=LNcRNA_X.iloc[:,8].str.split('"', expand=True)[3].tolist()
LNcRNA_U=pd.read_csv("A1163_ML_U_LncRNA_T_6_8.gtf", sep="\t",comment="#",␣
 ↪header=None )
LNcRNA_U=LNcRNA_U.iloc[:,8].str.split('"', expand=True)[3].tolist()
# Novel potential protein coding Pre cut-off
POTP_X=pd.read_csv("A1163PT_X_6_8.gtf", sep="\t",comment="#", header=None )
POTP_X=POTP_X.iloc[:,8].str.split('"', expand=True)[3].tolist()
POTP_U=pd.read_csv("A1163PT_U_6_8.gtf", sep="\t",comment="#", header=None )
POTP_U=POTP_U.iloc[:,8].str.split('"', expand=True)[3].tolist()
# Read in TPM They need to have the gene-names attached
df=pd.read_csv("A1163_UNION_6_8_counts.txt", sep="\t", comment="#")
df=df.set_index('Geneid')
# Step 1 divide by length
df.iloc[:,5:]=df.iloc[:,5:].apply(lambda x: x/df.Length)
#Read in the information of gene names
saf=pd.read_csv("superbedmerged_A1163_6_8_T_B.saf", sep="\t", header=None)
saf.columns= ["zero","one","two","three","name"]
saf.zero=saf.zero.astype(str).str.strip()
# Now merge to get gene name in file
df["zero"]=df.index
df.zero=df.zero.astype(str).str.strip()
df=df.merge(saf, how="inner", on ="zero")
df["named"]=df.name
```

```
df=df.set_index('name')
# Now we can trim down the dataframe to just the columns that we still want
df2=df.iloc[:,5:49]
# get the normalising factor
df3=(df2.sum())/1000000
#Now get the TPM
df_TPM=df2/df3
df_TPM["named"]=df_TPM.index
# Now make two more, df5 with 0.01  and then log2 transformed
# Make df5 with 0.01  and then log2 transformed
df5=df_TPM.copy()

#add the small amount here instead
df5.iloc[:,:44]=df5.iloc[:,:44]+0.01
df5.iloc[1:,:44]= np.log2(df5.iloc[1:,:44])
```

[3]:
```
# we can make cluster maps of log2 TPM values for potential protein
pot_protein=df5[df5.named.apply(lambda x: "PPT_"in x)]
sns.clustermap(pot_protein.iloc[:,0:44])
```

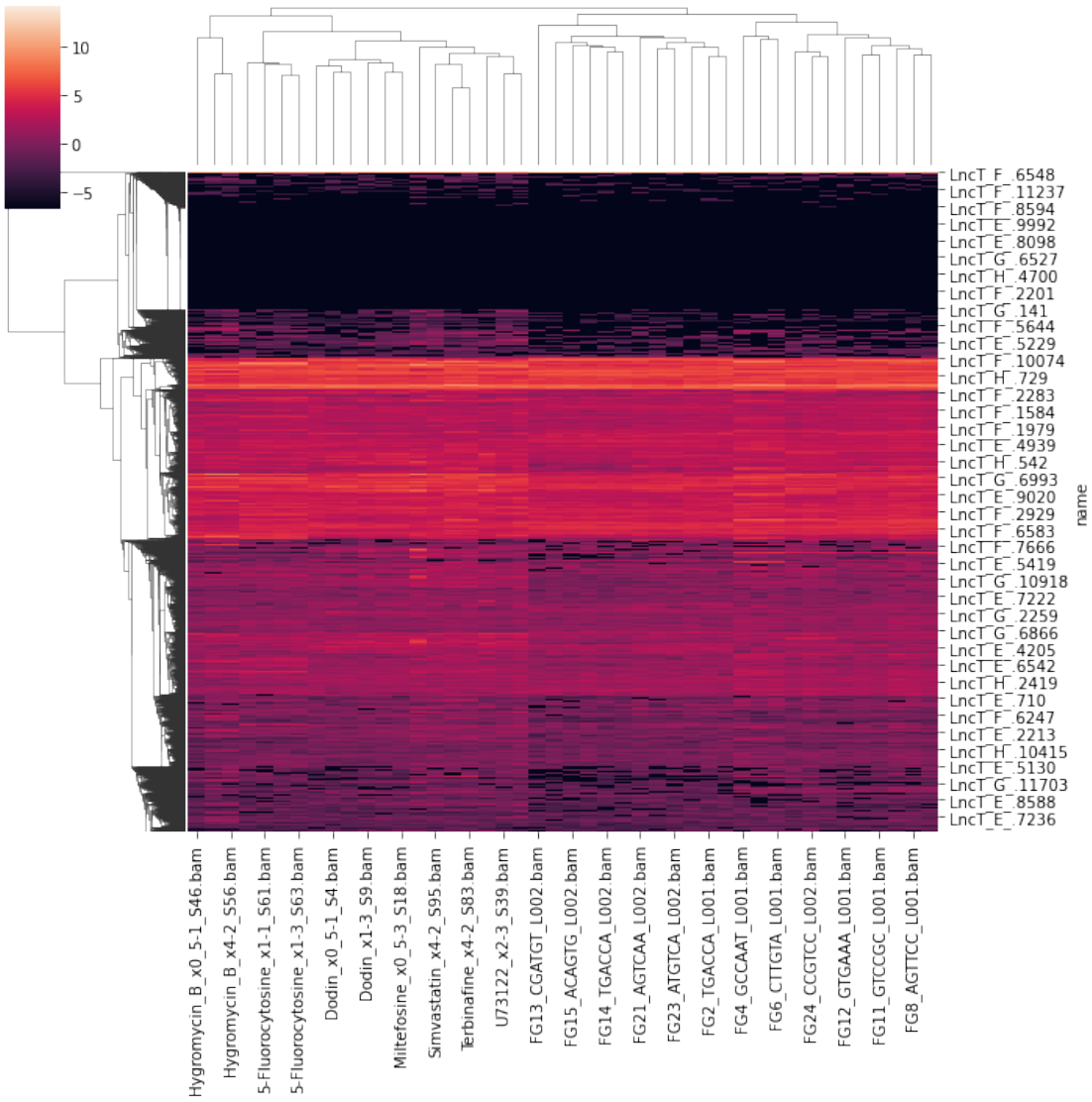
```
/home/marian-linux/anaconda3/lib/python3.9/site-packages/seaborn/matrix.py:654:
UserWarning: Clustering large matrix with scipy. Installing `fastcluster` may
give better performance.
  warnings.warn(msg)
```

[3]: <seaborn.matrix.ClusterGrid at 0x7f3c03756790>



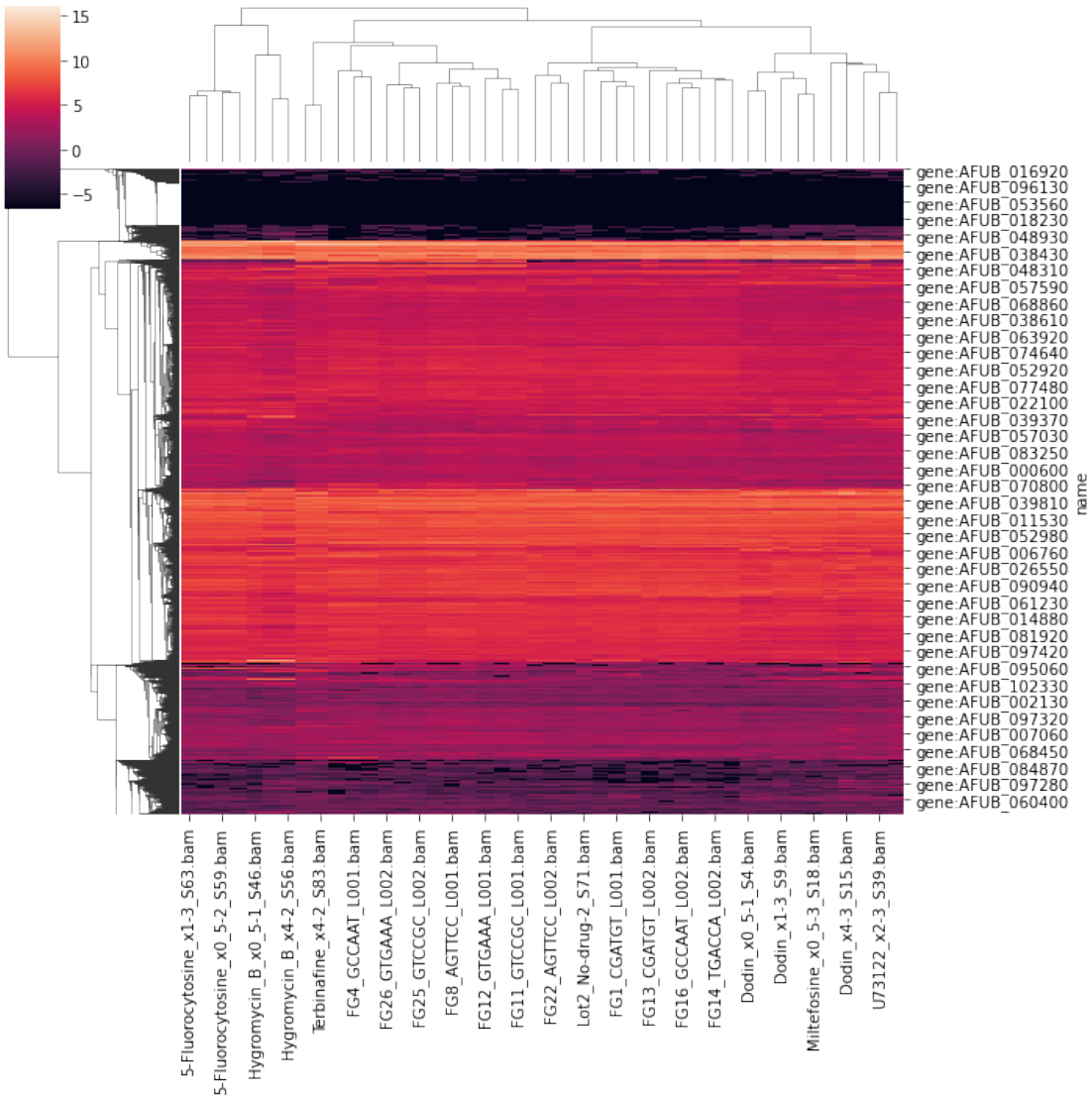
[4]:
```
# we can make cluster maps of log2 TPM values for all LncRNA
LNC_log2=df5[df5.named.apply(lambda x: "Lnc"in x)]
sns.clustermap(LNC_log2.iloc[:,0:44])
```

[4]: <seaborn.matrix.ClusterGrid at 0x7f3bfa945850>



[5]:
```
# Protein-coding in genome annotation
other=df5[df5.named.apply(lambda x: ("Lnc" not in x)& ("PPT_"  not in x)) ]
sns.clustermap(other.iloc[:,0:44])
```

[5]: <seaborn.matrix.ClusterGrid at 0x7f3bf8efbc70>

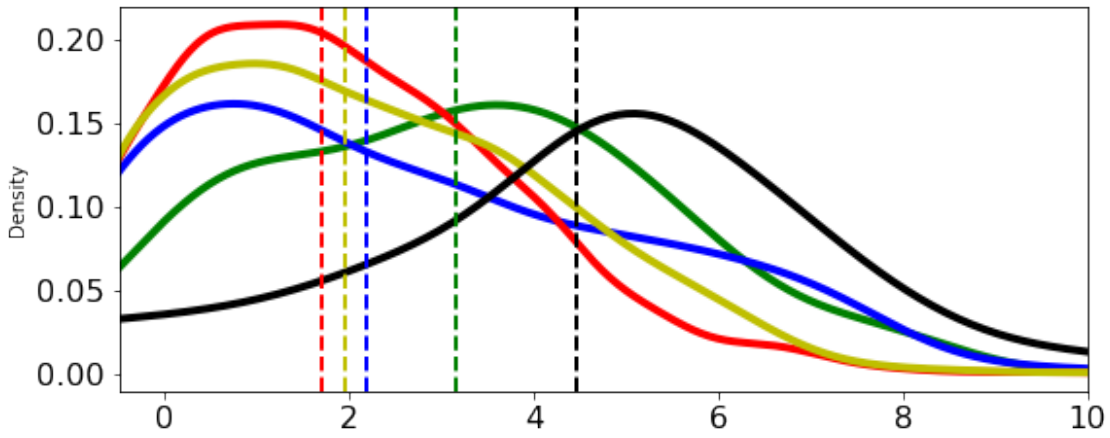


[6]:
```
# Now we need to make data frames for the TPM cut-off
# for LncRNA X type - antisense
Lnc_X_df=df_TPM.copy()[df_TPM.named.apply(lambda x: x in LNcRNA_X)]
Lnc_X_df.loc[:,"median_value"]=Lnc_X_df.iloc[:,:44].median(axis=1)
Lnc_X_df.loc[:,"mean_value"]=Lnc_X_df.iloc[:,:44].mean(axis=1)

# for  LncRNA U type - intergenic
Lnc_U_df=df_TPM.copy()[df_TPM.named.apply(lambda x: x in LNcRNA_U)]
Lnc_U_df.loc[:,"median_value"]=Lnc_U_df.iloc[:,:44].median(axis=1)
Lnc_U_df.loc[:,"mean_value"]=Lnc_U_df.iloc[:,:44].mean(axis=1)

# Novel potential protein coding for U type
Pot_P_U_df=df_TPM.copy()[df_TPM.named.apply(lambda x: x  in POTP_U )]
```

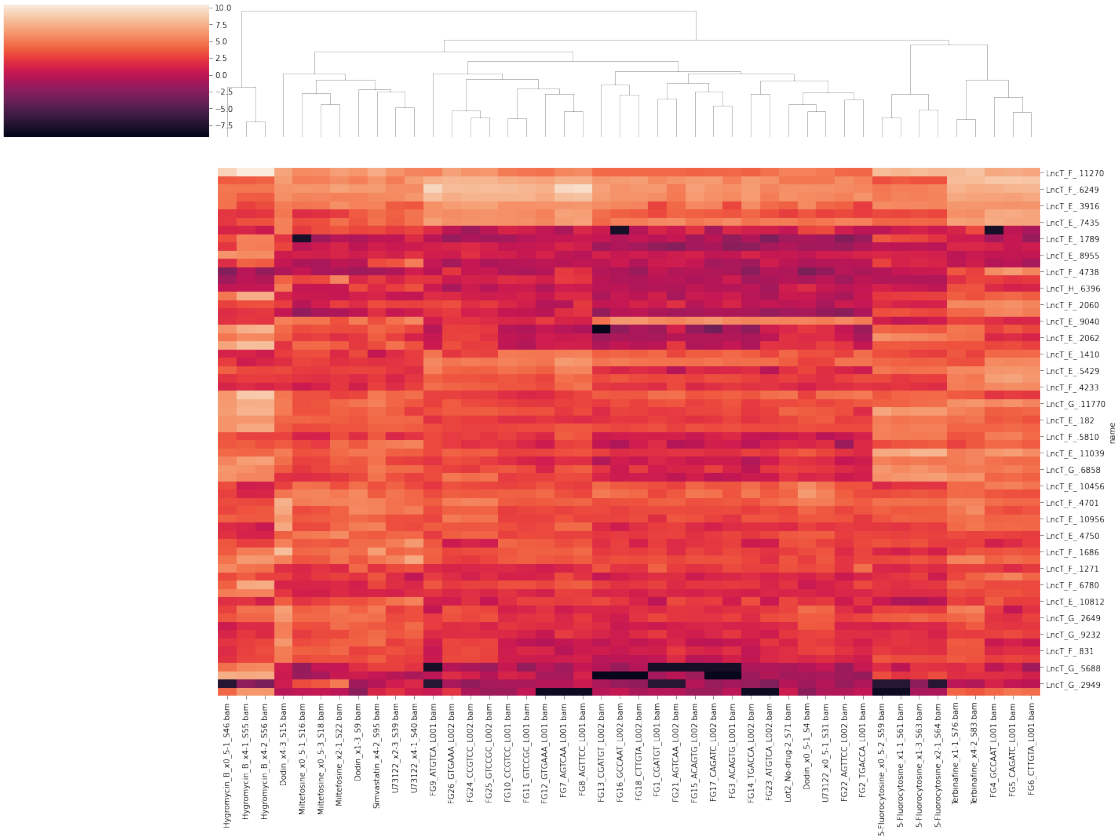
```
Pot_P_U_df.loc[:,"median_value"]=Pot_P_U_df.iloc[:,:44].median(axis=1)
Pot_P_U_df.loc[:,"mean_value"]=Pot_P_U_df.iloc[:,:44].mean(axis=1)

# Novel potential protein coding for X type
Pot_P_X_df=df_TPM.copy()[df_TPM.named.apply(lambda x: x in POTP_X)]
Pot_P_X_df.loc[:,"median_value"]=Pot_P_X_df.iloc[:,:44].median(axis=1)
Pot_P_X_df.loc[:,"mean_value"]=Pot_P_X_df.iloc[:,:44].mean(axis=1)

# for annotation protein coding genes
OTHER_df=df_TPM.copy()[df_TPM.named.apply(lambda x: ("Lnc" not in x) & ("PPT"␣
 ↪not in x))]
OTHER_df.loc[:,"median_value"]=OTHER_df.iloc[:,:44].median(axis=1)
OTHER_df.loc[:,"mean_value"]=OTHER_df.iloc[:,:44].mean(axis=1)

# The cut-off mean value TPM>0.5
Lnc_U_df_cut_off=Lnc_U_df.copy()[Lnc_U_df["mean_value"]>0.5]
Lnc_X_df_cut_off=Lnc_X_df.copy()[Lnc_X_df["mean_value"]>0.5]
Pot_P_X_df_cut_off=Pot_P_X_df.copy()[Pot_P_X_df["mean_value"]>0.5]
Pot_P_U_df_cut_off=Pot_P_U_df.copy()[Pot_P_U_df["mean_value"]>0.5]
```

[7]:
```
# to draw a density plot of mean log2(mean TPM) with medians marked
z=np.log2(OTHER_df.mean_value +0.001)
c=np.log2(Lnc_X_df_cut_off.mean_value+0.001)
d=np.log2(Lnc_U_df_cut_off.mean_value+0.001)
a=np.log2(Pot_P_X_df_cut_off.mean_value+0.001)
b=np.log2(Pot_P_U_df_cut_off.mean_value+0.001)
a2=np.log2(np.median(Pot_P_X_df_cut_off.mean_value)+0.001)
b2=np.log2(np.median(Pot_P_U_df_cut_off.mean_value)+0.001)
c2=np.log2(np.median(Lnc_X_df_cut_off.mean_value)+0.001)
d2=np.log2(np.median(Lnc_U_df_cut_off.mean_value)+0.001)
z2=np.log2(np.median(OTHER_df.mean_value)+0.001)

a.plot(kind='density',color='g',linewidth=4.0,figsize=(8, 3.35))
b.plot(kind='density',color='b',linewidth=4.0,figsize=(8, 3.35))
c.plot(kind='density',color='r',linewidth=4.0,figsize=(8, 3.35))
d.plot(kind='density',color='y',linewidth=4.0,figsize=(8, 3.35))
z.plot(kind='density',color='k',linewidth=4.0,figsize=(8, 3.35))

plt.axvline(x=a2, linestyle='--',color='g',linewidth=2.0)
plt.axvline(x=b2, linestyle='--',color='b',linewidth=2.0)
plt.axvline(x=c2, linestyle='--',color='r',linewidth=2.0)
plt.axvline(x=d2, linestyle='--',color='y',linewidth=2.0)
plt.axvline(x=z2, linestyle='--',color='k',linewidth=2.0)
plt.xticks(size=16)
plt.yticks( size=16)
plt.tight_layout()
plt.xlim([-0.5, 10])
```

```
plt.show()
```



[8]:
```
# Hierarchical clustering of log2 transformed normalized TPM values
#(fold change relative to mean expression level for each gene)
# to get plots with high variation between samples and high max TPM value
Lnc_X_df_cut_offa=Lnc_X_df_cut_off.copy()
Lnc_X_df_cut_offa["max_value"]=Lnc_X_df_cut_offa.iloc[:,:44].max(axis=1)
Lnc_X_df_cut_offa["min_value"]=Lnc_X_df_cut_offa.iloc[:,:44].min(axis=1)
Lnc_X_df_cut_offa["max_min"]=(Lnc_X_df_cut_offa.max_value - Lnc_X_df_cut_offa.
 ↪min_value)/Lnc_X_df_cut_offa.max_value
Lnc_X_df_cut_offa=Lnc_X_df_cut_offa[(Lnc_X_df_cut_offa.max_min>.95)]
Lnc_X_df_cut_offa=Lnc_X_df_cut_offa[(Lnc_X_df_cut_offa.max_value>30.5)]
Lnc_X_df_cut_offa.iloc[:,0:44]=Lnc_X_df_cut_offa.iloc[:,0:44].apply(lambda x:
 ↪x+0.01/(Lnc_X_df_cut_offa.mean_value))
Lnc_X_df_cut_offa= np.log2(Lnc_X_df_cut_offa.iloc[:,0:44]+0.001)
cg=sns.clustermap(Lnc_X_df_cut_offa,figsize=(20,15))
cg.ax_row_dendrogram.set_visible(False)
plt.tight_layout()
```



0.1 MAKE ALL NEW GTFS of both LncRNAs and Potential novel proteins above the expression threshold

[9]:
```
LncU_keepers=Lnc_U_df_cut_off.index
LncX_keepers=Lnc_X_df_cut_off.index
Pot_P_U_keepers=Pot_P_U_df_cut_off.index
Pot_P_X_keepers=Pot_P_X_df_cut_off.index

dfLX=pd.read_csv("A1163_ML_X_LncRNA_T_6_8.gtf", sep="\t",comment="#",␣
 ↪header=None )
dfLX["names"] = dfLX.iloc[:,8].str.split('"', expand=True)[3]
dfLX=dfLX[dfLX.names.apply(lambda x: x in LncX_keepers)]
dfLX.iloc[:,:9].to_csv("A1163_ML_X_LncRNA_T_TPM_CUTOFF_6_8.gtf", header=False,␣
 ↪index= None, sep="\t",quoting=csv.QUOTE_NONE)

dfLU=pd.read_csv("A1163_ML_U_LncRNA_T_6_8.gtf", sep="\t",comment="#",␣
 ↪header=None )
dfLU["names"] = dfLU.iloc[:,8].str.split('"', expand=True)[3]
dfLU=dfLU[dfLU.names.apply(lambda x: x in LncU_keepers)]
```

```
dfLU.iloc[:,:9].to_csv("A1163_ML_U_LncRNA_T_TPM_CUTOFF_6_8.gtf", header=False,␣
 ↪index= None, sep="\t",quoting=csv.QUOTE_NONE)

dfPot_P_U=pd.read_csv("A1163PT_U_6_8.gtf", sep="\t",comment="#", header=None )
dfPot_P_U["names"] = dfPot_P_U.iloc[:,8].str.split('"', expand=True)[3]
dfPot_P_U=dfPot_P_U[dfPot_P_U.names.apply(lambda x: x in Pot_P_U_keepers)]
dfPot_P_U.iloc[:,:9].to_csv("A1163PT_U_6_8_TPM_CUTOFF_6_8.gtf", header=False,␣
 ↪index= None, sep="\t",quoting=csv.QUOTE_NONE)

dfPot_P_X=pd.read_csv("A1163PT_X_6_8.gtf", sep="\t",comment="#", header=None )
dfPot_P_X["names"] = dfPot_P_X.iloc[:,8].str.split('"', expand=True)[3]
dfPot_P_X=dfPot_P_X[dfPot_P_X.names.apply(lambda x: x in Pot_P_X_keepers)]
dfPot_P_X.iloc[:,:9].to_csv("A1163PT_X_6_8_TPM_CUTOFF_6_8.gtf", header=False,␣
 ↪index= None, sep="\t",quoting=csv.QUOTE_NONE)

dfLUX=pd.read_csv("A1163_ML_UX_LncRNA_T_6_8.gtf", sep="\t",comment="#",␣
 ↪header=None )
dfLUX["names"] = dfLUX.iloc[:,8].str.split('"', expand=True)[3]
dfLUX=dfLUX[dfLUX.names.apply(lambda x: (x in LncX_keepers)|(x in␣
 ↪LncU_keepers))]
dfLUX.iloc[:,:9].to_csv("A1163_ML_UX_LncRNA_T_TPM_CUTOFF_6_8.gtf",␣
 ↪header=False, index= None, sep="\t",quoting=csv.QUOTE_NONE)

dfPot_P_U=pd.read_csv("A1163PT_U_6_8.gtf", sep="\t",comment="#", header=None )
dfPot_P_U["names"] = dfPot_P_U.iloc[:,8].str.split('"', expand=True)[3]
dfPot_P_U=dfPot_P_U[dfPot_P_U.names.apply(lambda x: x in Pot_P_U_keepers)]
dfPot_P_U.iloc[:,:9].to_csv("A1163PT_U_6_8_TPM_CUTOFF_6_8.gtf", header=False,␣
 ↪index= None, sep="\t",quoting=csv.QUOTE_NONE)
```

[10]:
```
# To compare mean TPM levels after cut-off between antisense and intergenic␣
 ↪LncRNAs for example

LncX_mean=Lnc_X_df_cut_off.mean_value.tolist()
LncU_mean=Lnc_U_df_cut_off.mean_value.tolist()
POTPU_mean=Pot_P_U_df_cut_off.mean_value.tolist()
POTPX_mean=Pot_P_X_df_cut_off.mean_value.tolist()

mannwhitneyu(LncX_mean,LncU_mean)
```

[10]:
```
MannwhitneyuResult(statistic=1184579.0, pvalue=0.0004854718664515808)
```

[11]:
```
# To compare median TPM levels after cut-off between antisense and intergenic␣
 ↪LncRNAs for example
LncX_median=Lnc_X_df_cut_off.median_value.tolist()
LncU_median=Lnc_U_df_cut_off.median_value.tolist()
POTPU_median=Pot_P_U_df_cut_off.median_value.tolist()
```

```
POTPX_median=Pot_P_X_df_cut_off.median_value.tolist()

mannwhitneyu(LncX_median,LncU_median)
```

[11]: MannwhitneyuResult(statistic=1195771.0, pvalue=0.001983343908479737)