In [185…

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import kstest
import seaborn as sns
```

In [186…

```
combined_path = '/Users/ayeshaferoz/Documents/Re=70k,noise=1e5/adderthefiles.csv'
original_path = '/Users/ayeshaferoz/Documents/Res=35k,noise=0/Files/originalmasses.csv'
```

In [186…

```
# Read the input files into pandas dataframes
combined_df = pd.read_csv(combined_path)
original_df = pd.read_csv(original_path)
```

In [186…

```
#original_df.iloc[original_df.index.get_loc('label1'), original_df.columns.get_loc('column1')]


print (original_df.iloc[:,0])
```

```
0        1
1        1
2        1
3        2
4        2
        ..
1017    13
1018     2
1019    14
1020    14
1021    14
Name: ScanNum, Length: 1022, dtype: int64
```

In [186…

```
# Define the tolerance limit in parts per million
ppmtol = 10

# Create the fdval dataframe by combining the ScanNum and MonoisotopicMass columns from the combined_df
fdval = combined_df[['ScanNum', 'MonoisotopicMass']]
```

In [186…

```
# Convert ScanNum column to int
#fdval['ScanNum'] = fdval['ScanNum'].astype(int)
fdval.loc[:, 'ScanNum'] = fdval['ScanNum']
```

```
/Users/ayeshaferoz/opt/anaconda3/lib/python3.9/site-packages/pandas/core/indexing.py:1951: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self.obj[selected_item_labels] = value
```

In [186…

```
print(fdval.iloc[:, 0])
```

```
0         1.0
1         1.0
2         1.0
3         1.0
4         2.0
        ...
2832    12.0
2833    12.0
2834    12.0
2835    12.0
2836    12.0
Name: ScanNum, Length: 2837, dtype: float64
```

In [186…

```
# Create the tpindex1 logical index by checking if the values in the first column of fdval are present in the first col
#tpindex1 = fdval.iloc[:,0].isin(original_df.iloc[:,0])
tpindex1 = fdval['ScanNum'].isin(original_df['ScanNum'].dropna().replace([np.inf, -np.inf], np.nan).astype(int))

# Create sample data
#original_col = original_df.iloc[:, 0]
#fdval_col = fdval.iloc[:, 0]

# Create a set of unique values in original_col
#original_set = set(original_col)

# Create the 'tpindex1' variable with True and False values
#tpindex1 = [value in fdval_col for value in original_col]
```

```
# Print the result
#print(tpindex1)
```

In [186…

```
#print(tpindex1)
```

In [186…

```
# Define a function to calculate the ppm difference between two values
def ppm_diff(value1, value2):
    return abs(value1 - value2) / value1 * 1e6
```

In [186…

```
print(fdval.iloc[:,1])
```

```
0        6867.795516
1       37174.683432
2       44785.659055
3       44790.591190
4       16083.287644
            ...
2832    19463.486080
2833    22742.753065
2834    25991.998563
2835    28425.436207
2836    41083.837849
Name: MonoisotopicMass, Length: 2837, dtype: float64
```

In [187…

```
print(original_df.iloc[:,1])
```

```
0       56379.465471
1       44789.476232
2       37173.633950
3       55264.825188
4       76615.711702
            ...
1017    35933.542893
1018    32488.889078
1019    36111.675676
1020    27585.192944
1021    41745.089213
Name: MonoisotopicMass, Length: 1022, dtype: float64
```

In [187…

```
# Create the tpindex2 logical index by comparing the values in the second column of fdval and original_df within the ppm
tpindex2 = np.isclose(fdval.iloc[:,1][:, np.newaxis], original_df.iloc[:,1], rtol=ppmtol/1e6, atol=0)
print (tpindex2)
```

```
[[False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False False False]
 ...
 [False False False ... False False False]
 [False False False ... False False False]
 [False False False ... False False False]]
```

```
/var/folders/0h/2m_hz8w9753cyyccd83dq44w0000gn/T/ipykernel_8716/415609627.py:2: FutureWarning: Support for multi-dimensi
onal indexing (e.g. `obj[:, None]`) is deprecated and will be removed in a future version.  Convert to a numpy array bef
ore indexing instead.
  tpindex2 = np.isclose(fdval.iloc[:,1][:, np.newaxis], original_df.iloc[:,1], rtol=ppmtol/1e6, atol=0)
```

In [187…

```
#print(fdval.iloc[:,1])
```

In [187…

```
#print(original_df.iloc[:,1])
```

In [187…

```
# Create the tpindex logical index by combining tpindex1, tpindex2, and the DummyIndex column from combined_df with val
# Create the tpindex logical index by combining tpindex1, tpindex2, and the DummyIndex column from combined_df with val
tpindex = np.logical_and.reduce((
    tpindex1.values.flatten(),
    tpindex2.any(axis=1),
    combined_df['DummyIndex'].values==0
    ))
```

In [187…

```
print(tpindex)
```

```
[False False False ... False False False]
```

In [187…

```
# Create the fpindex logical index by combining the negation of tpindex1, tpindex2, and the DummyIndex column from comb
fpindex = np.logical_and.reduce((
    ~tpindex1.values.flatten(),
    ~tpindex2.any(axis=1),
    combined_df['DummyIndex'].values==0
    ))


fpindex = np.logical_and.reduce((tpindex1.values.flatten(),tpindex2.any(axis=1),combined_df['DummyIndex'].values>0))
```

In [187…

```
print(fpindex)
```

```
[False False False ... False False False]
```

In [187…

```
# Create the Decoyindex logical index by checking whether the value of DummyIndex in combined_df is greater than zero
Decoyindex = combined_df['DummyIndex'] > 0
```

In [187…

```
print(Decoyindex)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
2832     True
2833     True
2834     True
2835     True
2836     True
Name: DummyIndex, Length: 2837, dtype: bool
```

In [188…

```
Decoyindex1 = combined_df['DummyIndex'] ==1
print(Decoyindex1)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
2832     True
2833     True
2834     True
2835    False
```

```
2836    False
Name: DummyIndex, Length: 2837, dtype: bool
```

In [188…

```
Decoyindex2 = combined_df['DummyIndex'] ==2
print(Decoyindex2)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
2832    False
2833    False
2834    False
2835    False
2836    False
Name: DummyIndex, Length: 2837, dtype: bool
```

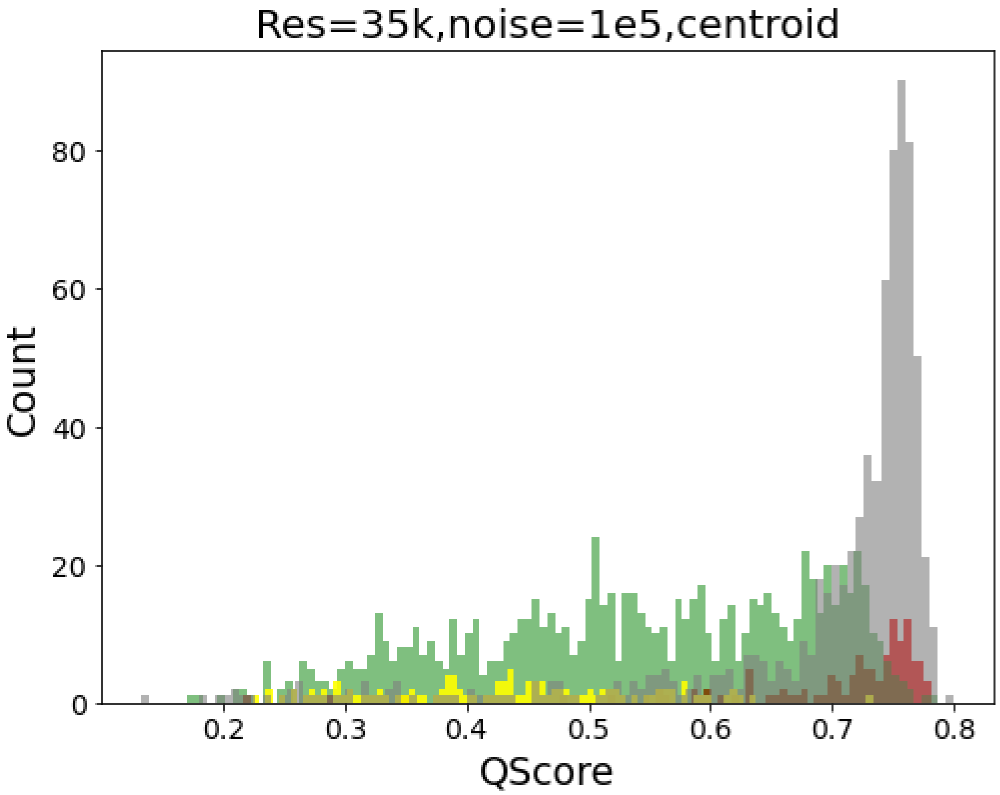
In [188…

```
Decoyindex3 = combined_df['DummyIndex'] ==3
print(Decoyindex3)
```

```
0       False
1       False
2       False
3       False
4       False
        ...
2832    False
2833    False
2834    False
2835     True
2836     True
Name: DummyIndex, Length: 2837, dtype: bool
```

In [188…

```
# Plot the histograms of QScore column in combined_df for fpindex and Decoyindex
plt.figure(figsize=(8,6))
plt.hist(combined_df.loc[fpindex, 'QScore'], bins=100, alpha=0.9, label='False positive masses',color='red')
plt.hist(combined_df.loc[Decoyindex1, 'QScore'], bins=100, alpha=0.5, label='Dummymasses1',color='green')
plt.hist(combined_df.loc[Decoyindex2, 'QScore'], bins=100, alpha=0.9, label='Dummymasses2',color='yellow')
plt.hist(combined_df.loc[Decoyindex3,'QScore'], bins=100,alpha=0.6, label='Dummymasses3',color='grey')
plt.xlabel('QScore', fontsize=19)
plt.ylabel('Count', fontsize=19)
plt.title('Res=35k,noise=1e5,centroid', fontsize=20)
plt.show()
```



In [188…

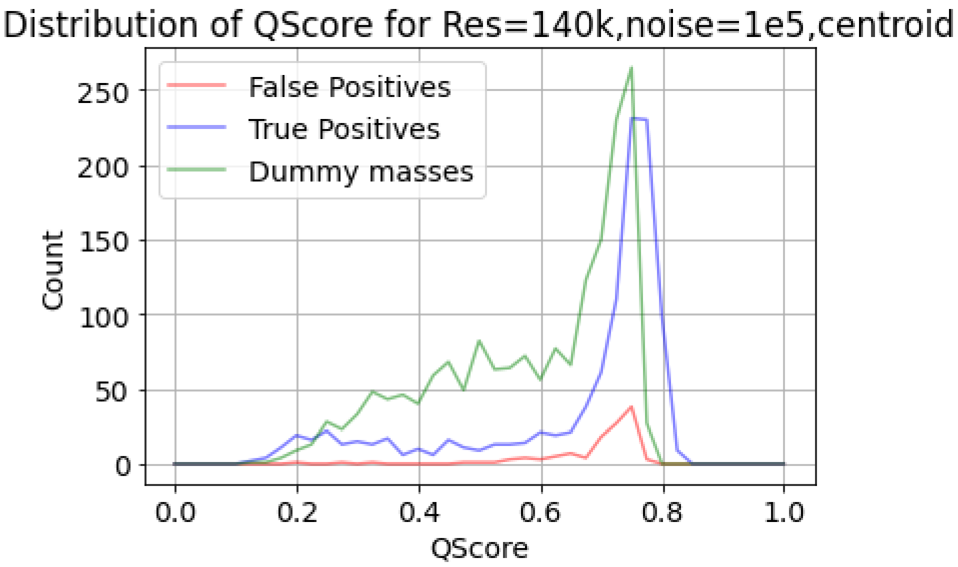
```
pip install nbconvert[webpdf]
```

```
zsh:1: no matches found: nbconvert[webpdf]
Note: you may need to restart the kernel to use updated packages.
```

In [188…

```
# Create two arrays of QScores for false positives and true positives
fp_scores = combined_df['QScore'][fpindex]
tp_scores = combined_df['QScore'][~Decoyindex]
decoy_scores=combined_df['QScore'][Decoyindex]

# Define the bins for the histogram
bins = np.arange(0, 1.05, 0.025)
```

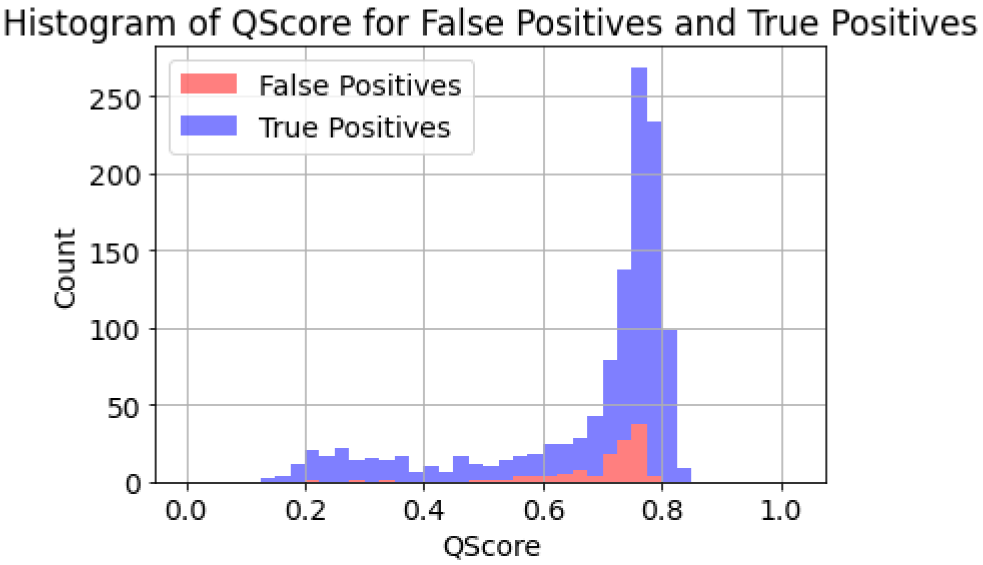
```
# Create a line plot for false positives
plt.plot(bins[:-1], np.histogram(fp_scores, bins=bins)[0], color='red', alpha=0.5)

# Create a line plot for true positives
plt.plot(bins[:-1], np.histogram(tp_scores, bins=bins)[0], color='blue', alpha=0.5)
#Create a line plot from dummy masses
plt.plot(bins[:-1], np.histogram(decoy_scores, bins=bins)[0], color='green', alpha=0.5)

# Add axis labels, title, legend, and grid to the plot
plt.xlabel('QScore')
plt.ylabel('Count')
plt.title('Distribution of QScore for Res=140k,noise=1e5,centroid')
plt.grid(True)
plt.legend(['False Positives', 'True Positives','Dummy masses'], loc='upper left')

# Display the plot
plt.show()
```



In [188…

```
# Compute the histogram values for false positives and true positives
fp_scores, bins = np.histogram(combined_df['QScore'][fpindex], bins=np.arange(0, 1.05, 0.025))
tp_scores, bins = np.histogram(combined_df['QScore'][~Decoyindex], bins=np.arange(0, 1.05, 0.025))

# Define the center positions of the bars
x = (bins[:-1] + bins[1:]) / 2
```

```
# Plot the histogram values as vertical bars
plt.bar(x, fp_scores, width=0.025, color='red', alpha=0.5)
plt.bar(x, tp_scores, width=0.025, color='blue', alpha=0.5, bottom=fp_scores)

# Add axis labels, title, legend, and grid to the plot
plt.xlabel('QScore')
plt.ylabel('Count')
plt.title('Histogram of QScore for False Positives and True Positives')
plt.grid(True)
plt.legend(['False Positives', 'True Positives'], loc='upper left')

# Display the plot
plt.show()
```



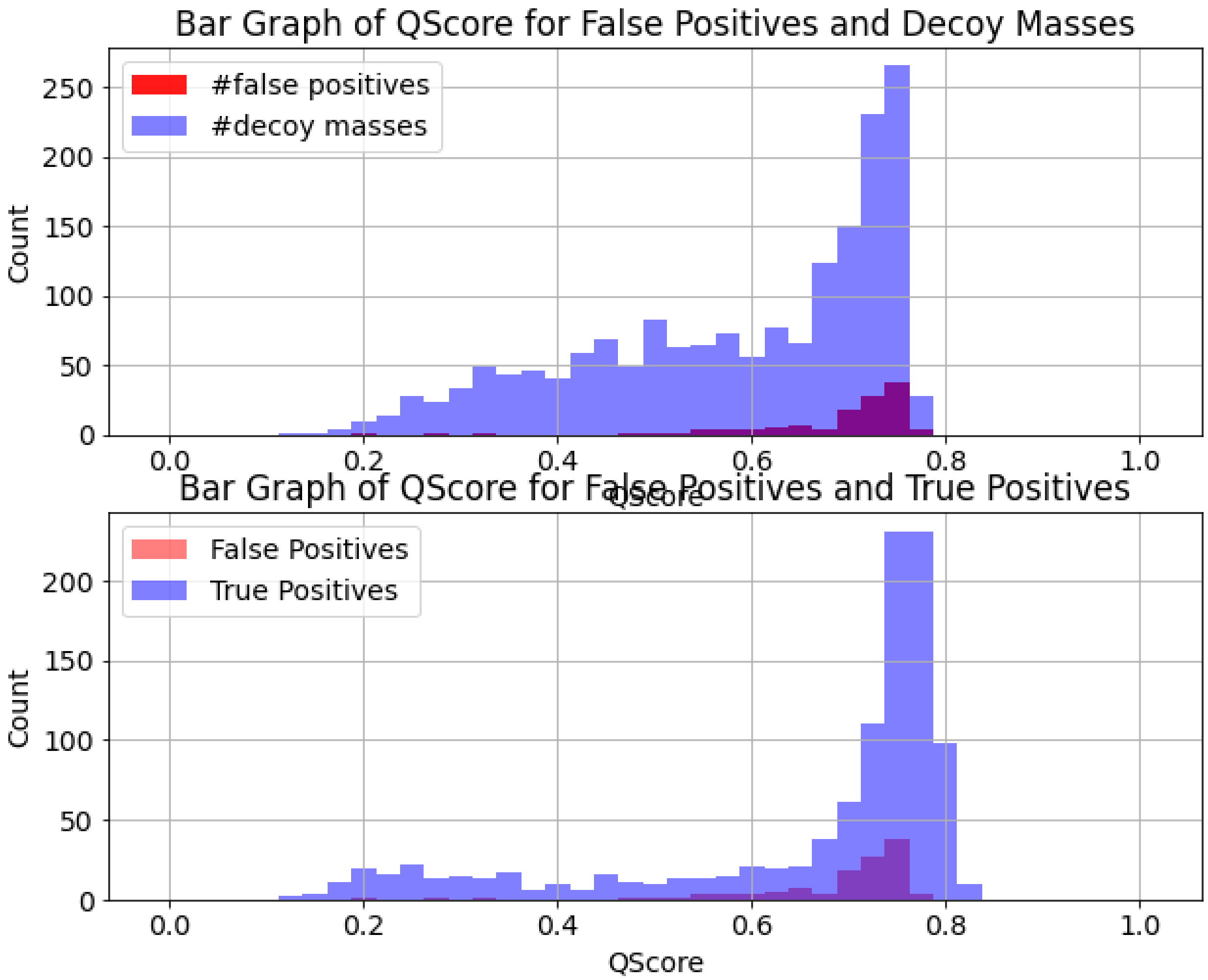
In [188…

```
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(10,8))

bins = np.arange(0, 1.05, 0.025)

fp_counts, _ = np.histogram(combined_df['QScore'][fpindex], bins=bins)
decoy_counts, _ = np.histogram(combined_df['QScore'][Decoyindex], bins=bins)

ax1.bar(bins[:-1], fp_counts, width=0.025, color='red', alpha=0.9)
ax1.bar(bins[:-1], decoy_counts, width=0.025, color='blue', alpha=0.5)
ax1.set_xlabel('QScore')
```

```
ax1.set_ylabel('Count')
ax1.set_title('Bar Graph of QScore for False Positives and Decoy Masses')
ax1.grid(True)
ax1.legend(['#false positives', '#decoy masses'], loc='upper left')

tp_counts, _ = np.histogram(combined_df['QScore'][~Decoyindex], bins=bins)

ax2.bar(bins[:-1], fp_counts, width=0.025, color='red', alpha=0.5)
ax2.bar(bins[:-1], tp_counts, width=0.025, color='blue', alpha=0.5)
ax2.set_xlabel('QScore')
ax2.set_ylabel('Count')
ax2.set_title('Bar Graph of QScore for False Positives and True Positives')
ax2.grid(True)
ax2.legend(['False Positives', 'True Positives'], loc='upper left')

#plt.show()
```

Out[188…

```
<matplotlib.legend.Legend at 0x7fa640e75f70>
```



Plot the histograms of QScore column in combined_df for fpindex and Decoyindex

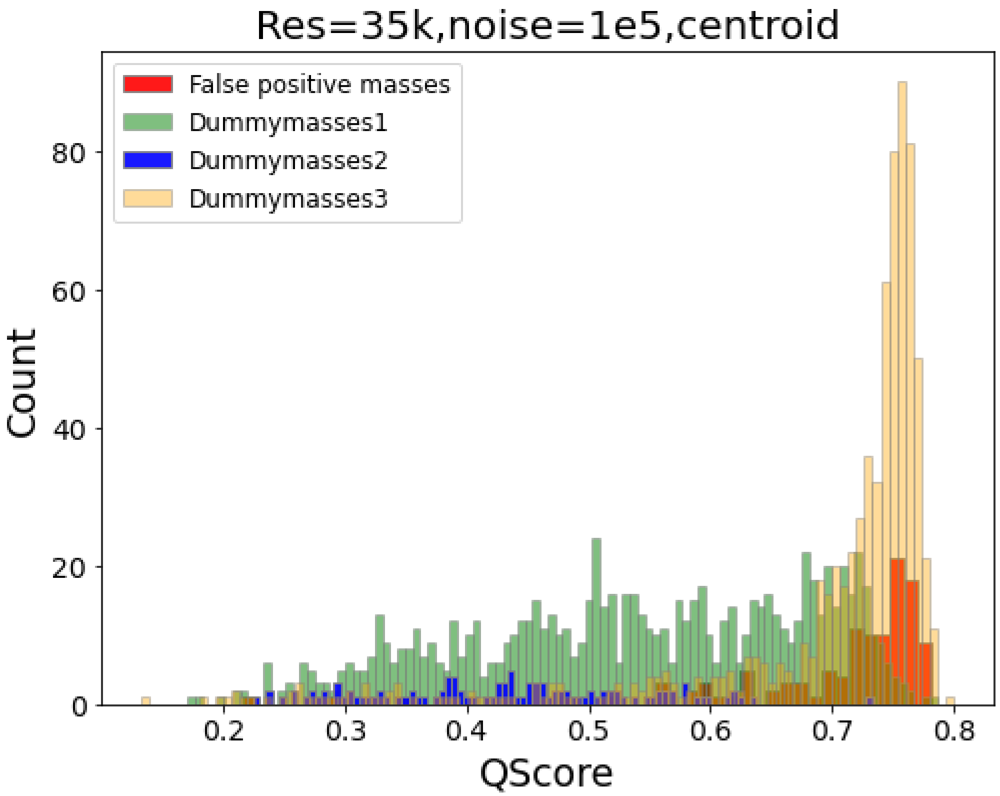
plt.figure(figsize=(8,6)) plt.hist(combined_df.loc[fpindex, 'QScore'], bins=100, alpha=0.7, label='False positives',color='red') plt.hist(combined_df.loc[Decoyindex, 'QScore'], bins=150, alpha=0.5, label='Dummy masses',color='green')

plt.hist(combined_df.loc[tpindex, 'QScore'], bins=150, alpha=0.3, label='True positive',color='blue')

plt.legend(fontsize=12) plt.xlabel('QScore', fontsize=19) plt.ylabel('Count', fontsize=19) plt.show()

In [188…

```
plt.figure(figsize=(8,6))
plt.hist(combined_df.loc[fpindex, 'QScore'], bins=50, alpha=0.9, label='False positive masses', color='red', edgecolor=
plt.hist(combined_df.loc[Decoyindex1, 'QScore'], bins=100, alpha=0.5, label='Dummymasses1', color='green', edgecolor='g
plt.hist(combined_df.loc[Decoyindex2, 'QScore'], bins=100, alpha=0.9, label='Dummymasses2', color='blue', edgecolor='gr
plt.hist(combined_df.loc[Decoyindex3,'QScore'], bins=100, alpha=0.4, label='Dummymasses3', color='orange', edgecolor='g
#plt.hist(combined_df.loc[Decoyindex,'QScore'], bins=100, alpha=0.6, label='Dummymasses', color='green', edgecolor='gre

plt.legend(fontsize=12)
plt.xlabel('QScore', fontsize=19)
plt.ylabel('Count', fontsize=19)
plt.title('Res=35k,noise=1e5,centroid', fontsize=20)
plt.show()
```

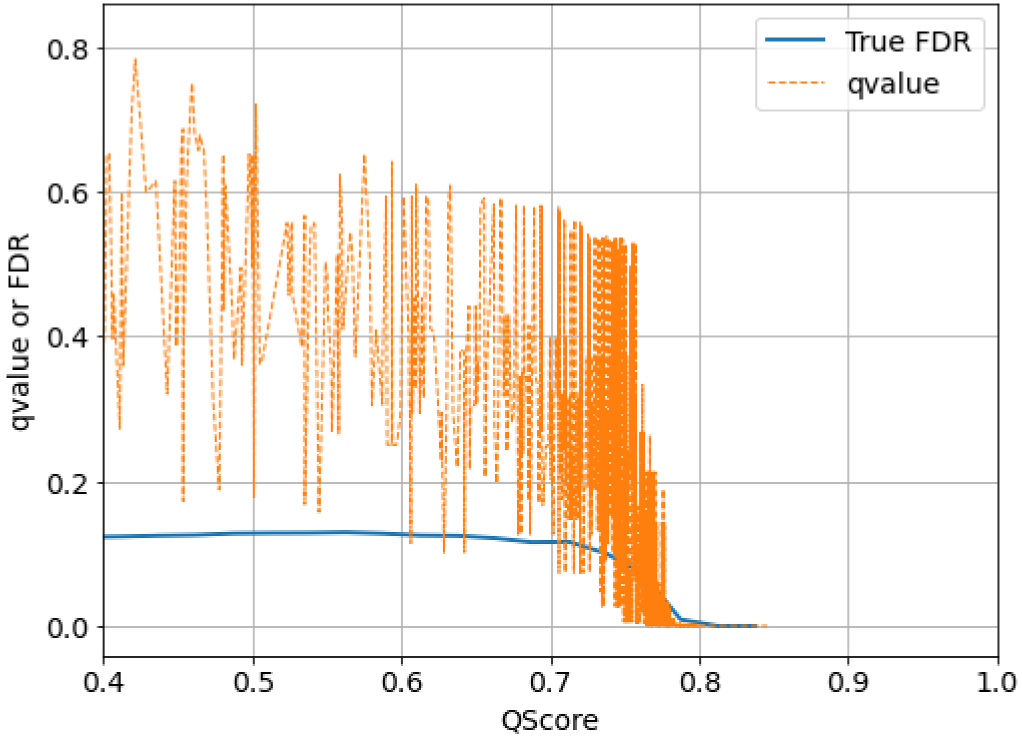


In [188…

```
plt.rc('font', size=14)

fp = np.histogram(combined_df.loc[fpindex, 'QScore'], bins=np.arange(0, 1.025, 0.025))
fpv = fp[0]

dp = np.histogram(combined_df.loc[~Decoyindex,'QScore'], bins=np.arange(0, 1.025, 0.025))
dpv = dp[0]

cdpv = np.zeros_like(dpv)
cfpv = np.zeros_like(fpv)
mask = (cdpv != 0)
ax.plot((dp[1][1:] + dp[1][:-1])/2, np.where(mask, cfpv/cdpv, 0), linewidth=2, color='tab:blue')


for i in range(len(dpv)):
    cdpv[-i-1] = np.sum(dpv[-i-1:])
    cfpv[-i-1] = np.sum(fpv[-i-1:])

sample1 = np.random.choice((dp[1][1:] + dp[1][:-1])/2, size=1000, replace=True, p=dpv/dpv.sum())
sample2 = np.random.choice((dp[1][1:] + dp[1][:-1])/2, size=1000, replace=True, p=fpv/fpv.sum())

h, p = kstest(sample1, sample2)
print(p)

fig, ax = plt.subplots(figsize=(8, 6))

ax.plot((dp[1][1:] + dp[1][:-1])/2, cfpv/cdpv, linewidth=2, color='tab:blue')

tmp = np.column_stack((combined_df.QScore[combined_df.DummyIndex==0], combined_df.Qvalue[combined_df.DummyIndex==0]))
tmp = tmp[tmp[:,0].argsort()]

ax.plot(tmp[:,0], tmp[:,1], '--', linewidth=1, color='tab:orange')

ax.set_xlim([0.4, 1])
ax.set_xlabel('QScore')
ax.set_ylabel('qvalue or FDR')
ax.legend(['True FDR', 'qvalue'])
ax.grid(True)

plt.show()
```

```
6.33938261831875e-34
```

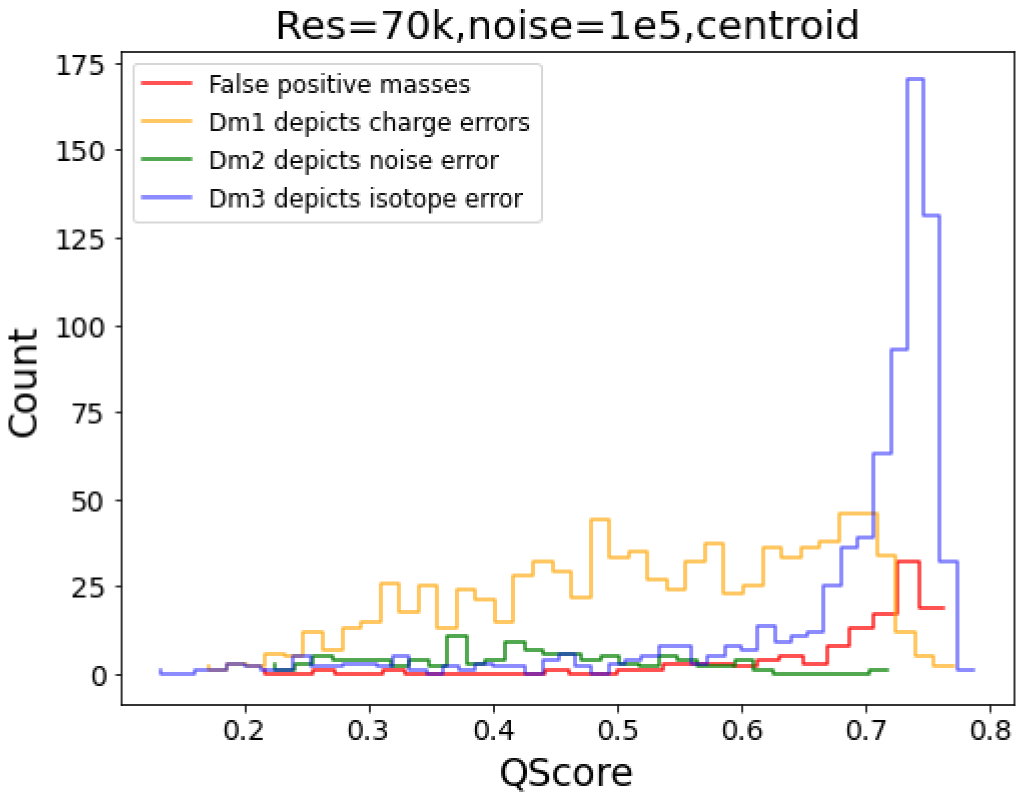
```
/var/folders/0h/2m_hz8w9753cyyccd83dq44w0000gn/T/ipykernel_8716/2923320848.py:12: RuntimeWarning: invalid value encounte
red in true_divide
  ax.plot((dp[1][1:] + dp[1][:-1])/2, np.where(mask, cfpv/cdpv, 0), linewidth=2, color='tab:blue')
/var/folders/0h/2m_hz8w9753cyyccd83dq44w0000gn/T/ipykernel_8716/2923320848.py:27: RuntimeWarning: invalid value encounte
red in true_divide
  ax.plot((dp[1][1:] + dp[1][:-1])/2, cfpv/cdpv, linewidth=2, color='tab:blue')
```



In [189…

```
fig = plt.figure(figsize=(8,6))

# False Positive Masses
fp_scores = combined_df.loc[fpindex, 'QScore']
fp_counts, fp_bins = np.histogram(fp_scores, bins=30)
plt.step(fp_bins[:-1], fp_counts, alpha=0.9, label='False positive masses', color='red', where='pre')

# Dummy Masses
dummy_scores = combined_df.loc[Decoyindex1,'QScore']
dummy_counts, dummy_bins = np.histogram(dummy_scores, bins=40)
plt.step(dummy_bins[:-1], dummy_counts, alpha=0.8, label='Dm1 depicts charge errors', color='orange', where='pre')
dummy_scores = combined_df.loc[Decoyindex2,'QScore']
```

```
dummy_counts, dummy_bins = np.histogram(dummy_scores, bins=33)
plt.step(dummy_bins[:-1], dummy_counts, alpha=0.9, label='Dm2 depicts noise error', color='green', where='pre')
dummy_scores = combined_df.loc[Decoyindex3,'QScore']
dummy_counts, dummy_bins = np.histogram(dummy_scores, bins=50)
plt.step(dummy_bins[:-1], dummy_counts, alpha=0.6, label='Dm3 depicts isotope error', color='blue', where='pre')
plt.legend(fontsize=12)
plt.xlabel('QScore', fontsize=19)
plt.ylabel('Count', fontsize=19)
plt.title('Res=70k,noise=1e5,centroid', fontsize=20)

plt.show()
```



```
In [ ]:
```

```
In [ ]:
```

In [ ]:

In [ ]: