CleanRNADataset

December 20, 2018

```
In [37]: samplesToIncludeInEachGroup=30
         numGenes = 60
         transcriptToAnalyze = 'ENSG00000120738.7'
```

```
In [38]: from glob import glob
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt
         import mygene
         from sklearn.ensemble import ExtraTreesClassifier
```

0.0.1 Import the data

The data is being imported and combined into dataframes * Importing the sample data so that the LAML patient IDs can be linked to the expression files * Importing all the FPKM expression data and combining them into a single dataframe * Renaming the filenames with the Patient IDs * Given that this is transcript level RNAseq data, creating a **Gene** column so that the data can be searched by Gene name

```
In [39]: df_samples = pd.read_csv("gdc_sample_sheet.2018-12-06.tsv",delimiter="\t")
         files = glob("Samples\\*\\*.FPKM.txt")
         dfs=[]
         for file in files:
             dfTemp= pd.read_csv(file,delimiter="\t",header=None,names=["Transcript",file.split(
             dfs.append(dfTemp)
         df=pd.concat(dfs,sort=False,axis=1)
```

```
In [40]: rename_dict=pd.Series(df_samples.loc[:,"Case ID"].values,index=df_samples.loc[:,"File N
         df.rename(rename_dict, axis='columns',inplace=True)
```

```
In [ ]: df['Gene']=df.index.str.split('.').str[0]
        cols = df.columns.tolist()
        cols = cols[-1:] + cols[:-1]
        df = df[cols]
        df.head()
```

```
In [42]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 60483 entries, ENSG00000242268.2 to ENSG00000181518.3
Columns: 152 entries, Gene to TCGA-AB-2917
dtypes: float64(151), object(1)
memory usage: 70.6+ MB
```

0.0.2 Analyse Transcript

The patient cohort is seperated into two sub-cohorts: high and low expression based on predefined constants
We analyze the TCGA-LAML RNAseq dataset to find a signature that seperates the two cohorts

```
In [43]: geneToAnalyze = df.loc[transcriptToAnalyze]['Gene']
```

```
In [44]: seriesTrasncriptMeasurements=df.loc[transcriptToAnalyze].T.iloc[1:].squeeze()
         seriesTrasncriptMeasurements=pd.to_numeric(seriesTrasncriptMeasurements)
         #Create the high and low patient lists from the data
         highAnalysisList=seriesTrasncriptMeasurements.sort_values(ascending=False)[:samplesToIn
         lowAnalysisList=seriesTrasncriptMeasurements.sort_values(ascending=False)[-samplesToInc
```

```
In [45]: # High transcript level
         df_high=df.T.loc[highAnalysisList]
         df_high["isHigh"] = 0
         #Low transcript level
         df_low=df.T.loc[lowAnalysisList]
         df_low["isHigh"] = 1
```

```
In [46]: df_high.index
```

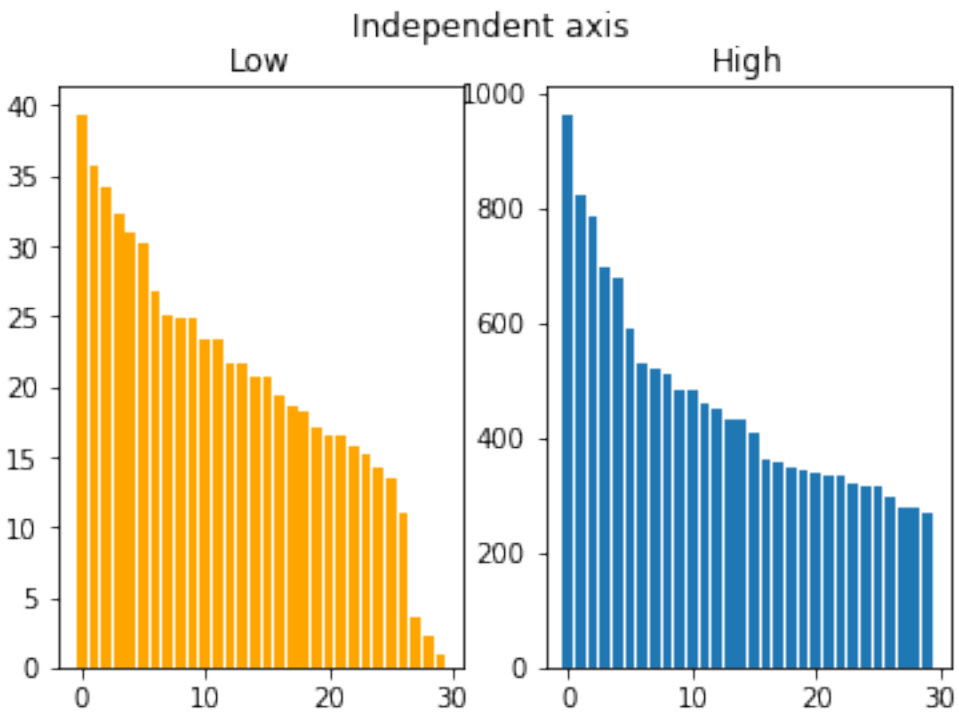
```
Out[46]: Index(['TCGA-AB-2984', 'TCGA-AB-2806', 'TCGA-AB-2975', 'TCGA-AB-3001',
                'TCGA-AB-2990', 'TCGA-AB-2820', 'TCGA-AB-2999', 'TCGA-AB-2894',
                'TCGA-AB-3012', 'TCGA-AB-2851', 'TCGA-AB-2977', 'TCGA-AB-3011',
                'TCGA-AB-2870', 'TCGA-AB-2914', 'TCGA-AB-2874', 'TCGA-AB-2826',
                'TCGA-AB-2910', 'TCGA-AB-2839', 'TCGA-AB-2866', 'TCGA-AB-2949',
                'TCGA-AB-2982', 'TCGA-AB-2928', 'TCGA-AB-2956', 'TCGA-AB-2948',
                'TCGA-AB-2846', 'TCGA-AB-2899', 'TCGA-AB-3009', 'TCGA-AB-2888',
                'TCGA-AB-2942', 'TCGA-AB-3007'],
               dtype='object')
```

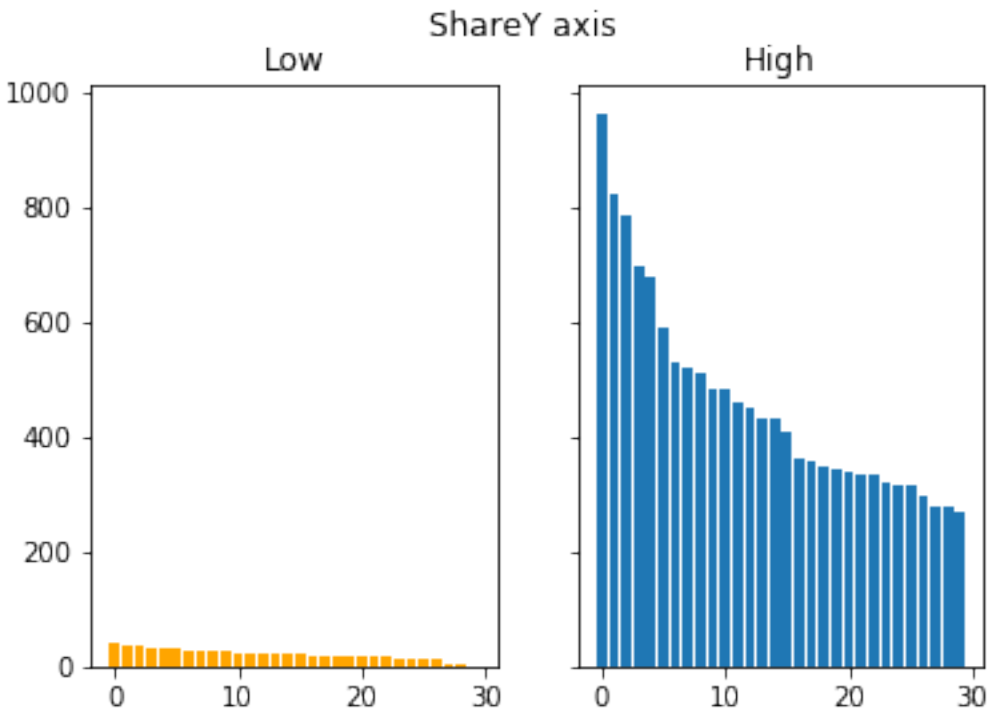
```
In [47]: df_low.index
```

```
Out[47]: Index(['TCGA-AB-2970', 'TCGA-AB-2885', 'TCGA-AB-2844', 'TCGA-AB-2896',
                'TCGA-AB-2871', 'TCGA-AB-2918', 'TCGA-AB-2908', 'TCGA-AB-2946',
                'TCGA-AB-2936', 'TCGA-AB-3008', 'TCGA-AB-2834', 'TCGA-AB-2941',
                'TCGA-AB-2976', 'TCGA-AB-2931', 'TCGA-AB-2937', 'TCGA-AB-2939',
                'TCGA-AB-2988', 'TCGA-AB-2880', 'TCGA-AB-2973', 'TCGA-AB-2900',
                'TCGA-AB-2943', 'TCGA-AB-2932', 'TCGA-AB-2912', 'TCGA-AB-2929',
                'TCGA-AB-2849', 'TCGA-AB-2913', 'TCGA-AB-2920', 'TCGA-AB-2811',
                'TCGA-AB-2966', 'TCGA-AB-2955'],
               dtype='object')
```

We plot the two cohorts to see if there are noticable differences in the expression of the transcript being analyzed

```
In [48]: fig,axes = plt.subplots(1,2)
         axes[0].bar(range(samplesToIncludeInEachGroup),df_low.T.loc[transcriptToAnalyze],color=
         axes[0].set_title("Low")

         axes[1].bar(range(samplesToIncludeInEachGroup),df_high.T.loc[transcriptToAnalyze])
         axes[1].set_title("High")
         fig.suptitle("Independent axis")


         fig,axes = plt.subplots(1,2,sharey=True)
         axes[0].bar(range(samplesToIncludeInEachGroup),df_low.T.loc[transcriptToAnalyze],color=
         axes[0].set_title("Low")

         axes[1].bar(range(samplesToIncludeInEachGroup),df_high.T.loc[transcriptToAnalyze])
         axes[1].set_title("High")

         fig.subplots_adjust(hspace=0.5)
         fig.suptitle("ShareY axis")

         plt.show()
```





The transcript being analyzed is removed from the dataframes that are being concatenated. The groups have been made based on the transcription level of the transcript in question and thus it will always be the best classified. Hence the training only has value where that data in not present

```
In [49]: df_high.drop(transcriptToAnalyze,axis=1,inplace=True)
         df_low.drop(transcriptToAnalyze,axis=1,inplace=True)
```

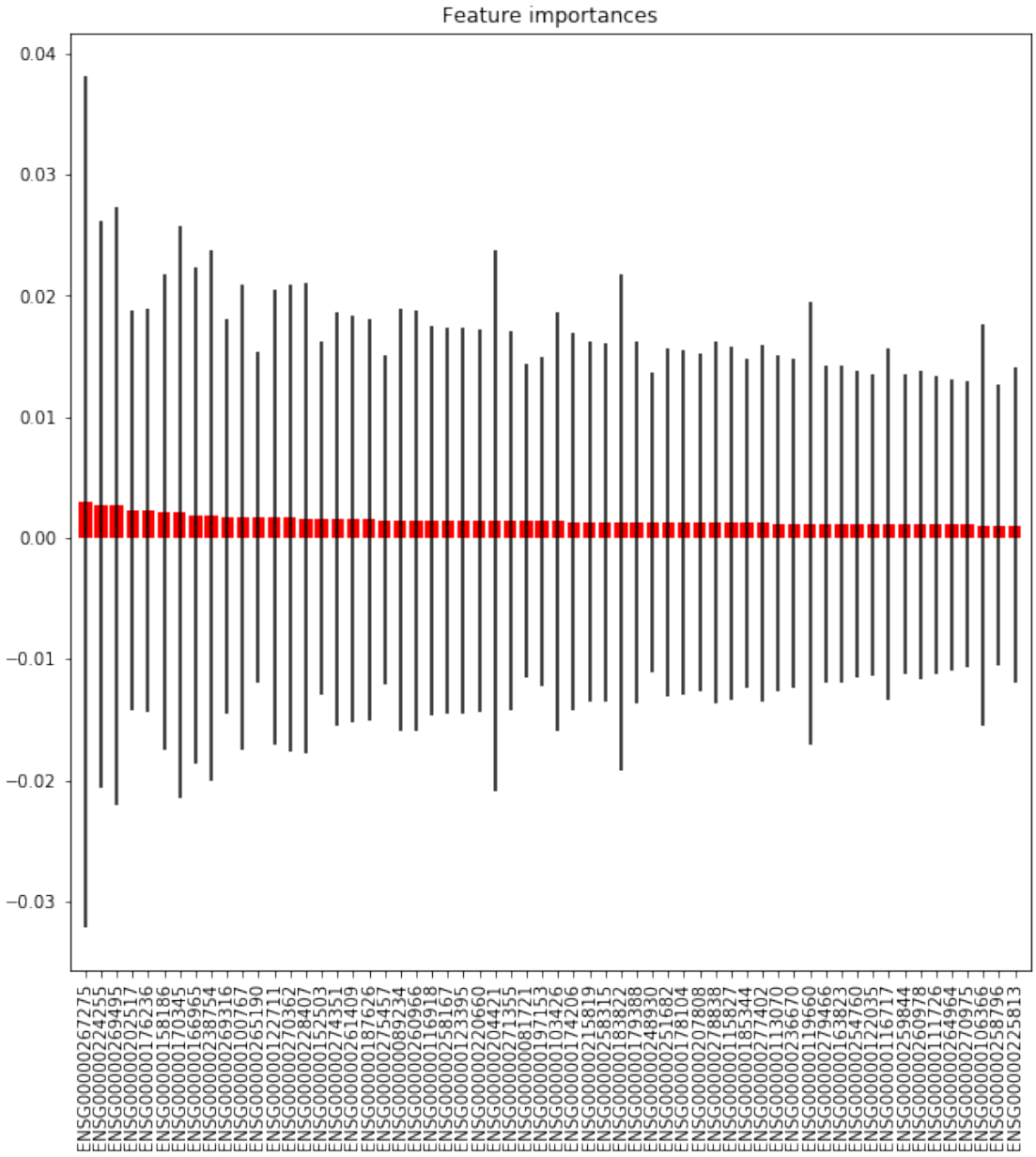
Generate the data and label arrays to enable training

```
In [50]: X = np.concatenate([df_high.values[:,:-1],df_low.values[:,:-1]]).astype('float')
         Y= np.hstack([df_high.values[:,-1],df_low.values[:,-1]]).astype('int')
```

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. (Wikipedia)
A random forest is used as the classifier which is trained to identify if the given patient sample belongs to the high or low category. After training is complete, the importance of each feature is calculated and is used to identify a *signature* for the transcript being analyzed

```
In [51]: # Build a forest and compute the feature importances
         forest = ExtraTreesClassifier(n_estimators=250,
                                       random_state=0)
```

```
forest.fit(X, Y)
importances = forest.feature_importances_
std = np.std([tree.feature_importances_ for tree in forest.estimators_],
             axis=0)
indices = np.argsort(importances)[::-1]

# Plot the feature importances of the forest
fig = plt.figure(figsize=(10,10))
plt.title("Feature importances")
plt.bar(range(numGenes), importances[indices][:numGenes], color="r", yerr=std[indices][
plt.xticks(range(numGenes), df.iloc[indices[:numGenes]]['Gene'],rotation=90)
plt.xlim([-1, numGenes])
plt.show()
fig.savefig('TranscriptImportance')
```



In [52]:
```
# Print the feature ranking
print("Feature ranking:")
for f in range(numGenes):
    print("%d. Transcript %d %s (%f)" % (f + 1, indices[f], df.iloc[indices[f]]["Gene"]
#     print("%s" % df.iloc[indices[f]]["Gene"])
```

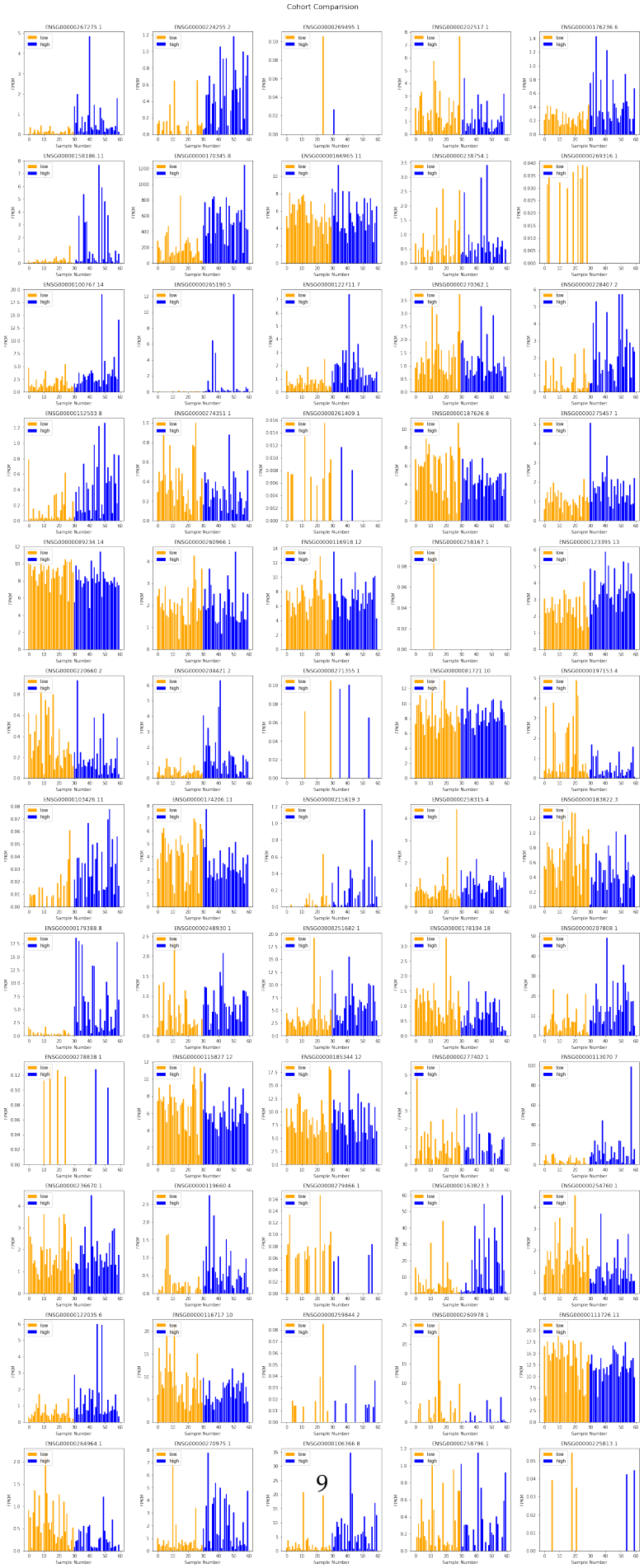
```
Feature ranking:
1. Transcript 28685 ENSG00000267275 (0.003012)
2. Transcript 31200 ENSG00000224255 (0.002759)
3. Transcript 43621 ENSG00000269495 (0.002644)
```

4. Transcript 42022 ENSG00000202517 (0.002294)
5. Transcript 7560 ENSG00000176236 (0.002284)
6. Transcript 27769 ENSG00000158186 (0.002125)
7. Transcript 38529 ENSG00000170345 (0.002123)
8. Transcript 49267 ENSG00000166965 (0.001824)
9. Transcript 51320 ENSG00000238754 (0.001818)
10. Transcript 2523 ENSG00000269316 (0.001751)
11. Transcript 38341 ENSG00000100767 (0.001720)
12. Transcript 16319 ENSG00000265190 (0.001697)
13. Transcript 10461 ENSG00000122711 (0.001686)
14. Transcript 40728 ENSG00000270362 (0.001648)
15. Transcript 17016 ENSG00000228407 (0.001606)
16. Transcript 21744 ENSG00000152503 (0.001566)
17. Transcript 58718 ENSG00000274351 (0.001510)
18. Transcript 60139 ENSG00000261409 (0.001504)
19. Transcript 35417 ENSG00000187626 (0.001489)
20. Transcript 13331 ENSG00000275457 (0.001480)
21. Transcript 36998 ENSG00000089234 (0.001460)
22. Transcript 42216 ENSG00000260966 (0.001443)
23. Transcript 52592 ENSG00000116918 (0.001432)
24. Transcript 48883 ENSG00000258167 (0.001426)
25. Transcript 14817 ENSG00000123395 (0.001419)
26. Transcript 19123 ENSG00000220660 (0.001415)
27. Transcript 7002 ENSG00000204421 (0.001411)
28. Transcript 54531 ENSG00000271355 (0.001402)
29. Transcript 47042 ENSG00000081721 (0.001397)
30. Transcript 27048 ENSG00000197153 (0.001396)
31. Transcript 31230 ENSG00000103426 (0.001357)
32. Transcript 47856 ENSG00000174206 (0.001322)
33. Transcript 28231 ENSG00000215819 (0.001320)
34. Transcript 35838 ENSG00000258315 (0.001298)
35. Transcript 34860 ENSG00000183822 (0.001297)
36. Transcript 12737 ENSG00000179388 (0.001295)
37. Transcript 20944 ENSG00000248930 (0.001292)
38. Transcript 18633 ENSG00000251682 (0.001288)
39. Transcript 16781 ENSG00000178104 (0.001280)
40. Transcript 37898 ENSG00000207808 (0.001253)
41. Transcript 57403 ENSG00000278838 (0.001241)
42. Transcript 35477 ENSG00000115827 (0.001209)
43. Transcript 54646 ENSG00000185344 (0.001208)
44. Transcript 43889 ENSG00000277402 (0.001201)
45. Transcript 26878 ENSG00000113070 (0.001189)
46. Transcript 41437 ENSG00000236670 (0.001183)
47. Transcript 26745 ENSG00000119660 (0.001158)
48. Transcript 9860 ENSG00000279466 (0.001155)
49. Transcript 35001 ENSG00000163823 (0.001154)
50. Transcript 34258 ENSG00000254760 (0.001140)
51. Transcript 21350 ENSG00000122035 (0.001113)

```
52. Transcript 53652 ENSG00000116717 (0.001111)
53. Transcript 46161 ENSG00000259844 (0.001110)
54. Transcript 35577 ENSG00000260978 (0.001089)
55. Transcript 19398 ENSG00000111726 (0.001086)
56. Transcript 48652 ENSG00000264964 (0.001078)
57. Transcript 21647 ENSG00000270975 (0.001061)
58. Transcript 32494 ENSG00000106366 (0.001054)
59. Transcript 43953 ENSG00000258796 (0.001045)
60. Transcript 44482 ENSG00000225813 (0.001036)
```

In [53]:
```
# A fucntion to plot the expression of the important transcripts for the two cohorts

def plotCohorts(transcriptList,dfHigh,dfLow):
    fig, axes = plt.subplots(int((len(transcriptList)+4)/5),5,figsize=(20,4*(int((len(t
    fig.suptitle("Cohort Comparision",y=1.01,fontsize=16)

    for counter,transcript in enumerate(transcriptList):
        ax = axes[int(counter/5)][counter%5]
        ax.bar(range(samplesToIncludeInEachGroup),df_low.T.loc[transcript],color='orang
        ax.bar(range(samplesToIncludeInEachGroup,2*samplesToIncludeInEachGroup),df_high
        ax.set_title(transcript)
        ax.set_xlabel('Sample Number')
        ax.set_ylabel('FPKM')
        ax.legend(loc='upper left')

    for counter in range(len(transcriptList),(int((len(transcriptList)+4)/5)*5)):
        fig.delaxes(axes[int(counter/5)][counter%5])

    plt.tight_layout()
    plt.show()
    fig.savefig('cohortPlots')
```

In [54]:
```
plotCohorts(df.index[indices][:numGenes],df_high,df_low)
```


Cohort Comparision
ENSG00000267275.1
ENSG00000224255.2
ENSG00000269495.1
ENSG00000202517.1
ENSG00000176236.6
ENSG00000158186.11
ENSG00000170345.8
ENSG00000166965.11
ENSG00000238754.1
ENSG00000269316.1
ENSG00000100767.14
ENSG00000265190.5
ENSG00000122711.7
ENSG00000270362.1
ENSG00000228467.2
ENSG00000152503.8
ENSG00000274351.1
ENSG00000261409.1
ENSG00000187626.8
ENSG00000275457.1
ENSG00000089234.14
ENSG00000260966.1
ENSG00000116918.12
ENSG00000258167.1
ENSG00000123395.13
ENSG00000220660.2
ENSG00000284421.2
ENSG00000271355.1
ENSG00000081721.10
ENSG00000197153.4
ENSG00000163426.11
ENSG00000174206.11
ENSG00000215819.3
ENSG00000258315.4
ENSG00000183822.3
ENSG00000179388.8
ENSG00000248930.1
ENSG00000251682.1
ENSG00000178104.18
ENSG00000297808.1
ENSG00000278838.1
ENSG00000115827.12
ENSG00000185344.12
ENSG00000277402.1
ENSG00000113070.7
ENSG00000236670.1
ENSG00000119660.4
ENSG00000279466.1
ENSG00000163823.3
ENSG00000254760.1
ENSG00000122035.6
ENSG00000116717.10
ENSG00000259844.2
ENSG00000260978.1
ENSG00000111726.11
ENSG00000264964.1
ENSG00000270975.1
ENSG00000106366.8
ENSG00000258796.1
ENSG00000225813.1
low
high
FPKM
Sample Number