In [66]:
```
!pwd
!ls
```

```
/home/ubuntu/cases/src
check.py              liver.csv              prostate          thyroid
colon                 lung                   README.md         Untitled.ipy
nb
gen_miRNA_matrix.py   lung.csv               request_meta.py   uterus
kidney                parse_file_case_id.py  stomach           uterus.csv
liver                 predict.py             stomach.csv       utils
```

In [82]:
```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

df_stomach = pd.read_csv(os.getcwd() + "/stomach.csv")
df_lung = pd.read_csv(os.getcwd() + "/lung.csv")
df_liver = pd.read_csv(os.getcwd() + "/liver.csv")
df_uterus = pd.read_csv(os.getcwd() + "/uterus.csv")

frames = [df_stomach, df_lung, df_liver, df_uterus]

df = pd.concat(frames, ignore_index=True)
df.head(-1)
```

1858	411	43b9-8ef1-0777acce750c	14692	14689	14699	34798	11301	3090	2513	5727	...
1859	412	134f9e0b-1a7a-4014-a7a7-703ae7b9fe9a	32164	31925	32125	61122	1581	5240	4356	8044	...
1860	413	e201d44f-8b2c-479d-ad0e-98b064b00cef	101933	101202	101786	138591	8834	6025	6740	33400	...
1861	414	c1fbe431-2aa7-4686-b64c-867620992be5	56854	56240	56714	100830	27511	3655	6391	14376	...
1862	415	f924c6c5-2b36-48fb-a6c2-45daa281a618	31952	31892	31746	39045	4043	2339	4580	19559	...
		b88f0785-b453-430a-									

In [83]:
```
df['y'] = df['Primary Site'] + df['label']
y_data = df.pop('y')

df.pop('label')
df.pop('Primary Site')
df.pop('file_id')
df.pop('Unnamed: 0')
```

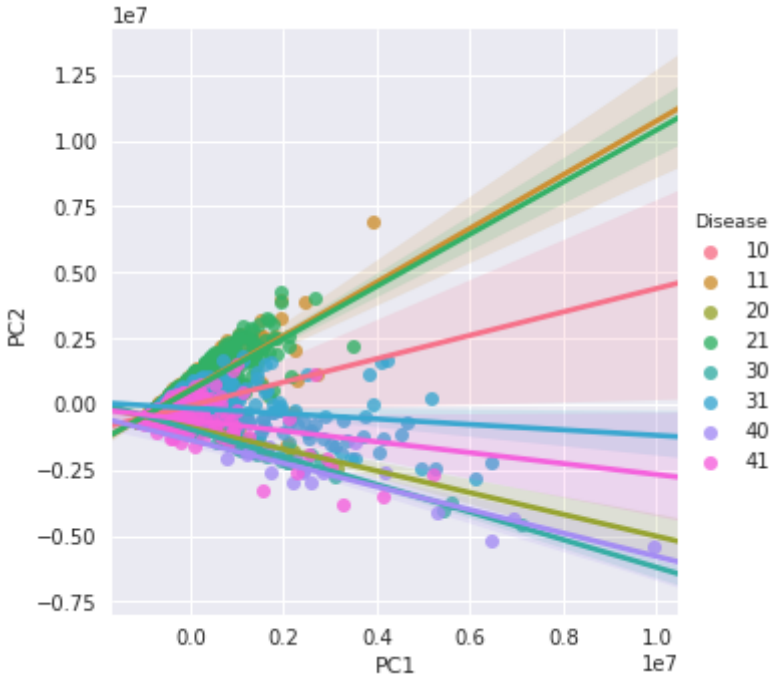
Out[83]:

hsa-let-7a-1	hsa-let-7a-2	hsa-let-7a-3	hsa-let-7b	hsa-let-7c	hsa-let-7d	hsa-let-7e	hsa-let-7f-1	hsa-let-7f-2	hsa-let-7g	...	hsa-mir-941-5	hsa-mir-942	hsa-mir-943	hsa-mir-944	hsa-mir-95	hs m 95

0 rows × 1881 columns

In [89]:
```
from sklearn.decomposition import PCA
pc = pca.fit_transform(df)
pc_df = pd.DataFrame(data = pc , columns = ['PC1', 'PC2'])
pc_df['Disease'] = y_data

sns.set()
ax = sns.lmplot(x='PC1', y='PC2', hue="Disease", data=pc_df)
```



In [87]:
```
sns.set(color_codes=True)
g = sns.clustermap(df)
g = sns.clustermap(df, method="single")
g = sns.clustermap(df, cmap="mako", robust=True) # ignoring outliers
g = sns.clustermap(df, cmap="mako", z_score=0) # normalizing the rows
```

In [*]:
```
columns = df.columns
X_data = df.values

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_data, y_data,
                                                    test_size=0.3,
                                                    random_state=0)
```

In [*]:
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.neighbors import KNeighborsClassifier

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.transform(X_test)

pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_std)
X_test_pca = pca.transform(X_test_std)

knn = KNeighborsClassifier(n_neighbors=5, p=2, metric='minkowski')
knn.fit(X_train_pca, y_train)
y_pred = knn.predict(X_test_pca)
```

In [*]:
```
# decision regions plot code is from the mlxtend library

from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap

def plot_decision_regions(X, y, classifier, resolution=0.02):
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])
    # setting up marker generator and color map

    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())
    # plot the decision surface
    for idx, cl in enumerate(np.unique(y)): plt.scatter(x=X[y == cl, 0],
                            y=X[y == cl, 1],
                            alpha=0.6,
                            c=cmap(idx),
                            edgecolor='black',
                            marker=markers[idx],
                            label=cl)
    # plot class samples
```

In [*]:
```
plt.subplot(1, 2, 1)
plot_decision_regions(X_train_pca, y_train, classifier=knn)
plt.xlabel('PC 1 of training set')
plt.ylabel('PC 2')
plt.subplot(1, 2, 2)
plot_decision_regions(X_test_pca, y_test, classifier=knn)
plt.xlabel('PC 1 of test set')
plt.legend(loc='best')
plt.show()
```

In [ ]:
```
```