In [1]:
```
import numpy as np
import numpy.linalg as alg
import pandas as pd
import random
import math
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv('ceph_hgdp_minor_code_XNA.betterAnnotated.csv')
snp = df.drop(columns=['snp', 'chr', 'pos'])
(P, N) = snp.shape
info = pd.read_csv('ceph_hgdp_minor_code_XNA.sampleInformation.csv')
```

MDS

In [3]:
```
k1=500
k2=3000

R1 = np.zeros((k1, P), dtype=float)
R2 = np.zeros((k2, P), dtype=float)

for i in range(k1):
    t = random.sample(range(0, P), k1)
    R1[i, t] = 1/k1
for i in range(k2):
    t = random.sample(range(0, P), k2)
    R2[i, t] = 1/k2
```

In [4]:
```
H = - np.ones((N, N))/N
H += np.eye(N)

X = np.array(snp)

X1 = np.dot(R1, X)
X1_centered = np.dot(X1, H)
K1 = np.dot(X1_centered.T, X1_centered)

X2 = np.dot(R2, X)
X2_centered = np.dot(X2, H)
K2 = np.dot(X2_centered.T, X2_centered)
```

In [5]:
```
eigen_values_k1, eigen_vectors_k1 = alg.eig(K1)
eigen_pairs_k1 = [ (eigen_values_k1[i], eigen_vectors_k1[:, i]) for i in range(len

eigen_values_k2, eigen_vectors_k2 = alg.eig(K2)
eigen_pairs_k2 = [ (eigen_values_k2[i], eigen_vectors_k2[:, i]) for i in range(len
```

In [6]:
```
eigen_pairs_k1.sort(key=lambda eigen_pairs_k1: eigen_pairs_k1[0], reverse=True)
eigen_pairs_k2.sort(key=lambda eigen_pairs_k2: eigen_pairs_k2[0], reverse=True)
```

In [7]:
```
# select top-2 eigen value
lambda1_k1, pca1_k1 = eigen_pairs_k1[0]
pca1_k1 = pca1_k1.astype(np.float64)
cord1_k1 = math.sqrt(lambda1_k1) * pca1_k1
lambda2_k1, pca2_k1 = eigen_pairs_k1[1]
pca2_k1 = pca2_k1.astype(np.float64)
cord2_k1 = math.sqrt(lambda2_k1)* pca2_k1
```

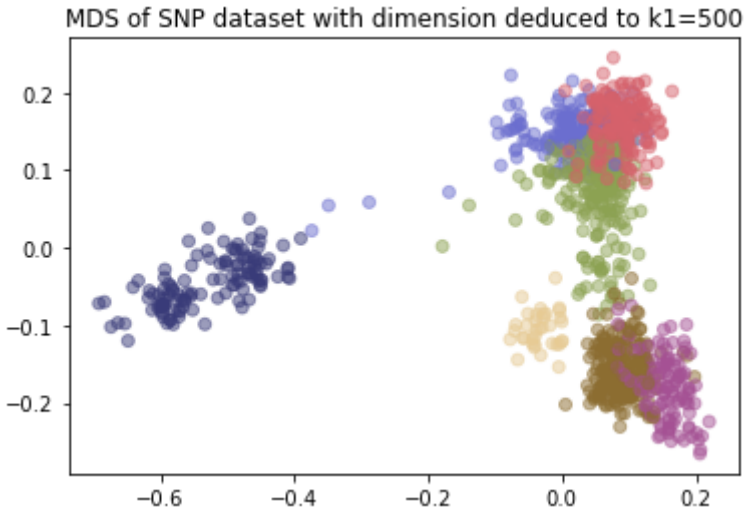
```
lambda1_k2, pca1_k2 = eigen_pairs_k2[0]
pca1_k2 = pca1_k2.astype(np.float64)
cord1_k2 = math.sqrt(lambda1_k2) * pca1_k2
lambda2_k2, pca2_k2 = eigen_pairs_k2[1]
pca2_k2 = pca2_k2.astype(np.float64)
cord2_k2 = math.sqrt(lambda2_k2) * pca2_k2
```

```
C:\Users\kosta\AppData\Local\Temp\ipykernel_23696\3236201392.py:3: ComplexWarning:
Casting complex values to real discards the imaginary part
  pca1_k1 = pca1_k1.astype(np.float64)
C:\Users\kosta\AppData\Local\Temp\ipykernel_23696\3236201392.py:4: ComplexWarning:
Casting complex values to real discards the imaginary part
  cord1_k1 = math.sqrt(lambda1_k1) * pca1_k1
C:\Users\kosta\AppData\Local\Temp\ipykernel_23696\3236201392.py:6: ComplexWarning:
Casting complex values to real discards the imaginary part
  pca2_k1 = pca2_k1.astype(np.float64)
C:\Users\kosta\AppData\Local\Temp\ipykernel_23696\3236201392.py:7: ComplexWarning:
Casting complex values to real discards the imaginary part
  cord2_k1 = math.sqrt(lambda2_k1)* pca2_k1
```

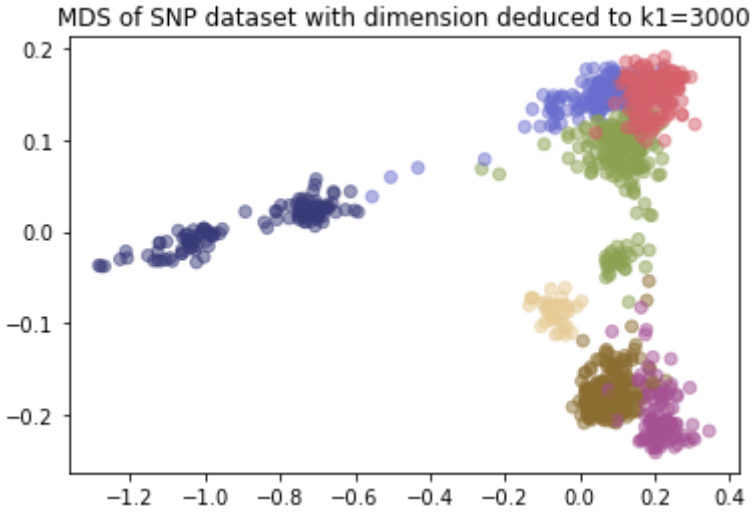
Visualization

In [8]:
```
region = info['region']
keys = list(region.unique())
color_range = list(np.linspace(0, 1, len(keys), endpoint=False))
colors = [plt.cm.tab20b(x) for x in color_range]
color_dict = dict(zip(keys, colors))
color_dict['No data'] = 'lightgray'
```

In [9]:
```
df1 = pd.DataFrame(dict(pca1=cord1_k1, pca2=cord2_k1, region=region))
fig1, ax1 = plt.subplots()
ax1.scatter(df1['pca1'], df1['pca2'], c=df1['region'].map(color_dict), alpha=0.5)
plt.title('MDS of SNP dataset with dimension deduced to k1=500')
plt.show()
```



In [12]:
```
df2 = pd.DataFrame(dict(pca1=-cord1_k2, pca2=cord2_k2, region=region))
fig2, ax2 = plt.subplots()
ax2.scatter(df2['pca1'], df2['pca2'], c=df2['region'].map(color_dict), alpha=0.5)
plt.title('MDS of SNP dataset with dimension deduced to k2=3000')
plt.show()
```


MDS of SNP dataset with dimension deduced to k1=3000
0.2
0.1
0.0
−0.1
−0.2
−1.2
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4