Homework 4

Hanze Dong

Problem 1

Read Data

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [2]:
```
df=pd.read_csv("ceph_hgdp_minor_code_XNA.betterAnnotated.csv")
df_info=pd.read_csv("ceph_hgdp_minor_code_XNA.sampleInformation.csv")

df.head()
```

Out[2]:

	snp	chr	pos	HGDP00448	HGDP00479	HGDP00985	HGDP01094	HGDP00982	HGDP00911	HGDP01202	...	HGDP01342	HGDP00824	HGDP0
0	rs10000929	4	131516474	1	0	0	1	1	0	1	...	1	0	
1	rs10002472	4	159087423	2	1	2	2	0	2	2	...	2	2	
2	rs10005550	4	128697858	2	2	2	2	1	0	1	...	2	2	
3	rs10007576	4	59063992	2	0	2	1	2	2	2	...	2	1	
4	rs10007998	4	35988597	0	0	0	0	0	0	0	...	2	1	

5 rows × 1046 columns

In [3]:
```
df_info.head()
```

Out[3]:

	ID	Gender	Population	Geographic.origin	Geographic.area	region	distance	latitude	longitude
0	HGDP00448	M	Biaka Pygmies	Central African Republic	Central Africa	Africa	2384.859098	4.0	17.0
1	HGDP00479	M	Biaka Pygmies	Central African Republic	Central Africa	Africa	2384.859098	4.0	17.0
2	HGDP00985	M	Biaka Pygmies	Central African Republic	Central Africa	Africa	2384.859098	4.0	17.0
3	HGDP01094	M	Biaka Pygmies	Central African Republic	Central Africa	Africa	2384.859098	4.0	17.0
4	HGDP00982	M	Mbuti Pygmies	Democratic Republic of Congo	Central Africa	Africa	1335.495772	1.0	29.0

In [4]:
```
X=df.to_numpy()
```

In [5]:
```
X.shape
```

Out[5]:
```
(488919, 1046)
```

In [9]:
```
X=X[:,3:].astype(float).T
```

In [33]:
```
X.shape
```

Out[33]:
```
(1043, 488919)
```

In [14]:
```
y=df_info["Geographic.area"]
```

MDS

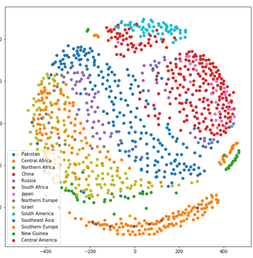
In [15]:
```
from sklearn.manifold import MDS
```

In [16]:
```
embedding = MDS(n_components=2)
```

In [18]:
```
X_transformed = embedding.fit_transform(X)
```

In [27]:
```
plt.figure(figsize=(10,10))
for label in set(list(y)):
    idx= y==label
    plt.scatter(X_transformed[idx,0],X_transformed[idx,1],label=label)
plt.legend()
```

Out[27]:
```
<matplotlib.legend.Legend at 0x7f1b93802d90>
```

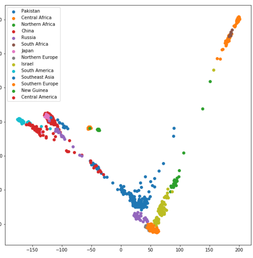


PCA

In [28]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
X_transformed_PCA = pca.fit_transform(X)
```

In [29]:
```
plt.figure(figsize=(10,10))
for label in set(list(y)):
    idx= y==label
    plt.scatter(X_transformed_PCA[idx,0],X_transformed_PCA[idx,1],label=label)
plt.legend()
```

Out[29]:
```
<matplotlib.legend.Legend at 0x7f1b937727d0>
```



In [43]:
```
from numpy.random import default_rng
rng = default_rng(1)
idx = rng.choice(488919, 1000)
X_random = X[:,idx]
```

In [44]:
```
X_random.shape
```

Out[44]:
```
(1043, 1000)
```

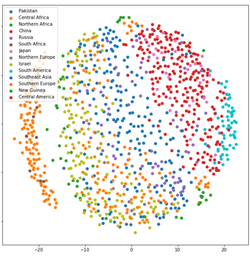
In [45]:
```
X_transformed_random = embedding.fit_transform(X_random)
```

In [46]:
```
X_PCA_random = pca.fit_transform(X_random)
```

Random projection MDS

In [47]:
```
plt.figure(figsize=(10,10))
for label in set(list(y)):
    idx= y==label
    plt.scatter(X_transformed_random[idx,0],X_transformed_random[idx,1],label=label)
plt.legend()
```

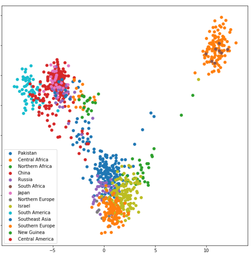
Out[47]:
```
<matplotlib.legend.Legend at 0x7f1b93313c90>
```



Random Projection PCA

In [48]:
```
plt.figure(figsize=(10,10))
for label in set(list(y)):
    idx= y==label
    plt.scatter(X_PCA_random[idx,0],X_PCA_random[idx,1],label=label)
plt.legend()
```

Out[48]:
```
<matplotlib.legend.Legend at 0x7f1b8e723350>
```

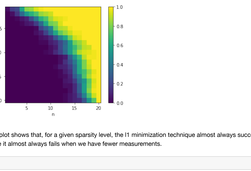


Problem 2

In [96]:
```
d = 20

P_mat = np.zeros([d,d])
for k in range(1,d+1):
    for m in range(1,d+1):
        sucess = 0
        for i in range(50):
            np.random.seed(i)
            x0 = np.zeros(d)
            idx = np.random.choice(d, k)
            x0[idx] = np.random.randint(0,2,[k])*2-1

            A = np.random.randn(m,d)
            b = A@x0

            # Construct the problem.
            x = cp.Variable(d)
            objective = cp.Minimize( cp.sum(cp.abs(x))  )
            constraints = [ A@x==b ]
            prob = cp.Problem(objective, constraints)
            result = prob.solve()
            if np.sum((x.value-x0)**2)**0.5<1e-3:
                sucess+=1
        p = sucess/50
        P_mat[k-1,m-1] = p
```

In [115]:
```
import matplotlib.pyplot as plt
plt.imshow(P_mat)
plt.xlabel('m')
plt.ylabel('k')
plt.xticks([4,9,14,19],[5,10,15,20])
plt.yticks([4,9,14,19],[5,10,15,20])
plt.colorbar()
```

Out[115]:
```
<matplotlib.colorbar.Colorbar at 0x7f1b9074edd0>
```



The plot shows that, for a given sparsity level, the l1 minimization technique almost always succeeds when we have an adequate number of measurements, while it almost always fails when we have fewer measurements.

In [ ]:
```

```