Homework 3

Ishan Jain

We load the `music_scaled.csv` dataset (source (https://archive.ics.uci.edu/ml/datasets/Geographical+Original+of+Music#)). The dataset contains a sample of traditional songs from different cultures. Features F1 to F68 are quantitative summaries of the songs from audio analysis software. These features have been subject to standard scaling. They are stored as predictors in `X`. The latitudes of the countries from which the songs originate are stored as a target variable `y`.

In [1]:

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

import numpy as np

data = pd.read_csv("music_scaled.csv")
X = data.iloc[:,:68]
y = data["Latitude"]
display(X)
```

	F1	F2	F3	F4	F5	F6	F7	F8	
0	-1.094000	-1.280592	2.806926	-0.097576	-0.791472	2.440896	0.003710	-0.864715	0.7
1	-1.285544	-0.940198	-0.721321	-0.172044	-2.127893	2.549762	1.365750	0.489953	1.6
2	0.503962	0.497136	-0.319168	0.330719	-0.398783	-0.749429	-2.380589	0.951098	-0.4
3	-1.119978	0.696697	0.612882	-0.983295	1.333148	1.557607	-0.999593	-1.067051	0.8
4	1.256214	1.066239	0.984965	-0.312513	-2.111077	0.009980	0.509741	0.837961	0.4
...	...	...	...	...	...	...	...	...	
1054	0.089066	0.045324	-0.527161	0.274475	0.113220	-1.011250	-1.071873	-1.533993	-3.4
1055	0.558342	0.274368	-0.356357	-0.410656	0.710800	-0.240603	-0.370162	0.870655	-1.4
1056	0.880180	0.668551	0.115084	-0.373637	0.848817	-0.859218	-0.823580	0.051974	-1.1
1057	-0.990084	-1.094018	3.645894	-0.474362	-1.129270	0.188591	0.092048	0.252823	-0.3
1058	-0.133272	-0.074305	-0.554388	-0.582548	0.307454	-0.958630	-1.523819	-1.589983	-3.6

1059 rows × 68 columns

Question 1 (2 pts)

a) Perform linear regression of **y** on **X** using `sklearn.linear_model.LinearRegression` when the regressors consist of:

- F1 only;
- F1 and F2 only;
- F1, F2, and F3 only;
- F1, F2, F3, ..., and F68.

b) In each of the above 4 cases, print the estimated coefficient for **F1.**

In [2]:

```
model = LinearRegression()
for i in range(X.shape[1]):
    coeffs_i = model.fit(X.iloc[:,:i+1],y)
    print(f'Coeff F1 for 0:{i+1:2} is {coeffs_i.coef_[0]}')
```

```
Coeff F1 for 0: 1 is 0.659733950012265
Coeff F1 for 0: 2 is 1.669923041178091
Coeff F1 for 0: 3 is 2.222157586667258
Coeff F1 for 0: 4 is 2.3244270430823346
Coeff F1 for 0: 5 is 2.268297983012784
Coeff F1 for 0: 6 is 2.8452351924823773
Coeff F1 for 0: 7 is 3.3320870224076193
Coeff F1 for 0: 8 is 3.356678940253303
Coeff F1 for 0: 9 is 3.2698567651136208
Coeff F1 for 0:10 is 3.303231638278588
Coeff F1 for 0:11 is 3.2920770604158536
Coeff F1 for 0:12 is 3.3208903646389163
Coeff F1 for 0:13 is 3.347535129423262
Coeff F1 for 0:14 is 3.3268162146365623
Coeff F1 for 0:15 is 3.285897642646161
Coeff F1 for 0:16 is 2.97362302107284
Coeff F1 for 0:17 is 2.9332092426984095
Coeff F1 for 0:18 is 5.019515373249254
Coeff F1 for 0:19 is 6.364015712678701
Coeff F1 for 0:20 is 4.998354920508962
Coeff F1 for 0:21 is 6.364778172303926
Coeff F1 for 0:22 is 4.133694649943278
Coeff F1 for 0:23 is 4.730007093894282
Coeff F1 for 0:24 is 4.98148878103636
Coeff F1 for 0:25 is 4.930128203655862
Coeff F1 for 0:26 is 5.093094310749812
Coeff F1 for 0:27 is 5.1984797323157785
Coeff F1 for 0:28 is 5.163328587511246
Coeff F1 for 0:29 is 5.157666806840055
Coeff F1 for 0:30 is 5.3129412654417685
Coeff F1 for 0:31 is 5.275584887832919
Coeff F1 for 0:32 is 5.282177798563248
Coeff F1 for 0:33 is 5.30331314747463
Coeff F1 for 0:34 is 4.9388998823916985
Coeff F1 for 0:35 is 4.75844919109582
Coeff F1 for 0:36 is 3.920741356653519
Coeff F1 for 0:37 is 3.979083895754725
Coeff F1 for 0:38 is 3.6551183225110986
Coeff F1 for 0:39 is 2.546008346527516
Coeff F1 for 0:40 is 2.4268589999434877
Coeff F1 for 0:41 is 2.2171089077737784
Coeff F1 for 0:42 is 2.3102492192512774
Coeff F1 for 0:43 is 2.528893660995219
Coeff F1 for 0:44 is 2.5772808457915968
Coeff F1 for 0:45 is 2.5796880236718667
Coeff F1 for 0:46 is 2.7779106469450006
Coeff F1 for 0:47 is 2.778351298560399
Coeff F1 for 0:48 is 2.750938877211275
Coeff F1 for 0:49 is 2.624263278768181
Coeff F1 for 0:50 is 2.613819697931031
Coeff F1 for 0:51 is 2.802340330013286
Coeff F1 for 0:52 is 2.730715055384175
Coeff F1 for 0:53 is 5.26376216977466
Coeff F1 for 0:54 is 6.4243799960593435
Coeff F1 for 0:55 is 8.821811756194647
Coeff F1 for 0:56 is 9.162811807827737
Coeff F1 for 0:57 is 8.98305474203043
Coeff F1 for 0:58 is 8.984069782543166
Coeff F1 for 0:59 is 8.61981963127432
Coeff F1 for 0:60 is 8.961607071015766
Coeff F1 for 0:61 is 8.991982928633284
Coeff F1 for 0:62 is 8.923925522497594
Coeff F1 for 0:63 is 8.864724445412197
Coeff F1 for 0:64 is 8.697396597129543
Coeff F1 for 0:65 is 8.636579698214105
Coeff F1 for 0:66 is 8.896444495901067
Coeff F1 for 0:67 is 8.45382033164647
Coeff F1 for 0:68 is 8.303670971591433
```

Question 2 (3 pts)

a) Perform a manual implementation to compute a full set of principal components of **X** using the following steps:

- Compute the sample covariance matrix
- Perform the eigen decomposition of the covariance matrix
- Project the data onto the principal components (eigenvectors of the covariance matrix)

b) Perform linear regression of **y** on the PCs using `sklearn.linear_model.LinearRegression` when the regressors consist of

- PC1 only;
- PC1 and PC2 only;
- PC1, PC2, and PC3 only;
- and PC1, PC2, PC3, ..., and PC68.

c) In each of the above 4 cases, print the estimated coefficient for **PC1**.

In [3]:

```
X = X.to_numpy().T
X_mean =  X.mean(axis=1)[np.newaxis]
X_norm = (X - X_mean.T)
C = (X_norm @ X_norm.T) / (X.shape[0]-1)

L,E = np.linalg.eigh(C)
L,E = L[::-1],E[:,::-1]
```

In [4]:

```
A = E.T @ X # Project data
```

In [5]:

```
model = LinearRegression()
model.fit(A[:1,:].T,y)
model.coef_
```

Out[5]:

```
array([0.08259023])
```

In [ ]:

```

```

In [6]:

```
model = LinearRegression()
A = A.T
for i in range(X.shape[0]):
    coeffs_i = model.fit(A[:,:i+1],y)
    print(f'Coeff F1 for 0:{i+1:2} is {coeffs_i.coef_[0]}')
```

```
Coeff F1 for 0: 1 is 0.08259022735453127
Coeff F1 for 0: 2 is 0.08259022735453127
Coeff F1 for 0: 3 is 0.08259022735453127
Coeff F1 for 0: 4 is 0.08259022735453127
Coeff F1 for 0: 5 is 0.08259022735453127
Coeff F1 for 0: 6 is 0.08259022735453127
Coeff F1 for 0: 7 is 0.08259022735453127
Coeff F1 for 0: 8 is 0.08259022735453127
Coeff F1 for 0: 9 is 0.08259022735453127
Coeff F1 for 0:10 is 0.08259022735453127
Coeff F1 for 0:11 is 0.08259022735453127
Coeff F1 for 0:12 is 0.08259022735453127
Coeff F1 for 0:13 is 0.08259022735453127
Coeff F1 for 0:14 is 0.08259022735453127
Coeff F1 for 0:15 is 0.08259022735453127
Coeff F1 for 0:16 is 0.08259022735453127
Coeff F1 for 0:17 is 0.08259022735453127
Coeff F1 for 0:18 is 0.08259022735453127
Coeff F1 for 0:19 is 0.08259022735453127
Coeff F1 for 0:20 is 0.08259022735453127
Coeff F1 for 0:21 is 0.08259022735453127
Coeff F1 for 0:22 is 0.08259022735453127
Coeff F1 for 0:23 is 0.08259022735453127
Coeff F1 for 0:24 is 0.08259022735453127
Coeff F1 for 0:25 is 0.08259022735453127
Coeff F1 for 0:26 is 0.08259022735453134
Coeff F1 for 0:27 is 0.08259022735453132
Coeff F1 for 0:28 is 0.08259022735453128
Coeff F1 for 0:29 is 0.08259022735453132
Coeff F1 for 0:30 is 0.0825902273545314
Coeff F1 for 0:31 is 0.0825902273545314
Coeff F1 for 0:32 is 0.08259022735453139
Coeff F1 for 0:33 is 0.08259022735453139
Coeff F1 for 0:34 is 0.0825902273545314
Coeff F1 for 0:35 is 0.08259022735453145
Coeff F1 for 0:36 is 0.08259022735453142
Coeff F1 for 0:37 is 0.08259022735453139
Coeff F1 for 0:38 is 0.08259022735453138
Coeff F1 for 0:39 is 0.08259022735453128
Coeff F1 for 0:40 is 0.0825902273545313
Coeff F1 for 0:41 is 0.08259022735453132
Coeff F1 for 0:42 is 0.08259022735453136
Coeff F1 for 0:43 is 0.08259022735453134
Coeff F1 for 0:44 is 0.08259022735453136
Coeff F1 for 0:45 is 0.08259022735453121
Coeff F1 for 0:46 is 0.08259022735453121
Coeff F1 for 0:47 is 0.08259022735453123
Coeff F1 for 0:48 is 0.08259022735453123
Coeff F1 for 0:49 is 0.08259022735453121
Coeff F1 for 0:50 is 0.0825902273545308
Coeff F1 for 0:51 is 0.08259022735453092
Coeff F1 for 0:52 is 0.08259022735453092
Coeff F1 for 0:53 is 0.08259022735453103
Coeff F1 for 0:54 is 0.08259022735453099
Coeff F1 for 0:55 is 0.08259022735453092
Coeff F1 for 0:56 is 0.08259022735453092
Coeff F1 for 0:57 is 0.08259022735453087
Coeff F1 for 0:58 is 0.08259022735453099
Coeff F1 for 0:59 is 0.08259022735453096
Coeff F1 for 0:60 is 0.08259022735453109
Coeff F1 for 0:61 is 0.08259022735453124
Coeff F1 for 0:62 is 0.08259022735453078
Coeff F1 for 0:63 is 0.08259022735453081
Coeff F1 for 0:64 is 0.08259022735453082
Coeff F1 for 0:65 is 0.08259022735453095
Coeff F1 for 0:66 is 0.08259022735453092
Coeff F1 for 0:67 is 0.08259022735453092
Coeff F1 for 0:68 is 0.08259022735452892
```

Question 3 (3 pts)

a) What do you observe in terms of estimated coefficients of **F1** when the number of regressors are increased compared to the estimated coefficients of **PC1** when the number of regressors are increased?

b) Explain the reason behind this observation.

Answer: We see that the predicted F1 coefficients alter and vary as the number of regressors rises. This demonstrates that as the number of regressors varies, so does the slope in the first dimension. When we projected the data points into the PC space, we saw essentially little change in the predicted coefficients of PC1, however. This is also what I had anticipated at first. The basis vectors are linearly independent when we project data into the PC space, therefore the coefficients should be the same no matter how many PCi axes we include. This is the explanation for this.

Question 4 (2 pts)

a) Print the correlation matrix of all the PC's.

b) Which matrix does it closely resemble to?

c) Which feature of the principal components is depicted through this matrix?

In [7]:

```python
print(E @ E.T)
```

```
[[ 1.00000000e+00 -4.99600361e-16 -1.94289029e-16 ...  4.33680869e-18
  -1.64798730e-17 -8.50014503e-17]
 [-4.99600361e-16  1.00000000e+00  9.54097912e-17 ... -8.67361738e-19
  -2.77555756e-17 -9.54097912e-17]
 [-1.94289029e-16  9.54097912e-17  1.00000000e+00 ... -2.65412692e-16
   9.70903045e-17  1.77863366e-16]
 ...
 [ 4.33680869e-18 -8.67361738e-19 -2.65412692e-16 ...  1.00000000e+00
  -3.00134266e-16  1.39780765e-16]
 [-1.64798730e-17 -2.77555756e-17  9.70903045e-17 ... -3.00134266e-16
   1.00000000e+00 -9.15337684e-17]
 [-8.50014503e-17 -9.54097912e-17  1.77863366e-16 ...  1.39780765e-16
  -9.15337684e-17  1.00000000e+00]]
```

It closely resembles the Identity matrix. Thus, we can say PCs are orthogonal.

Type *Markdown* and LaTeX: α^2