In [72]:
```
"""
Landon Buell
Marek Petrik
CS 750.01
13 April 2020
"""

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.decomposition import PCA
```

Problem 1 [33%]

Describe, in words, the results that you would expect if you performed K-means clustering of the eight shoppers in Figure 10.14 in ISL, on the basis of their sock and computer purchases, with $K = 2$. Give three answers, one for each of the variable scalings displayed. Explain.

If we performed K-means clustering on the shpper data from ISL 10.14, based on sock and computer purchases, with K = 2: For the left plot where we compare the quantity of items purchased, I predit that one cluster would cover those who bought n socks and 1 computer, and the other cluster would cover those who bought n socks and 0 computers. For the center plot, comparing the variances, scaling each variable by it's standard deviation would again be grouped by socks with standard deviation n, and those with computer standard deviation 0 or 1.4. Finally, I am not sure about the final plot, but I would predict about the same, clustering based on the varying computer prices against 0 due to how insignificant the sock proces are by comparison.

Problem 2 [33%]

In this problem, you will compute principal components for the Auto dataset. First remove qualitative features, which cannot be handled by PCA. Then:

1. Compute principal components without scaling features. Plot the result (you can use ggfortify::autoplot).

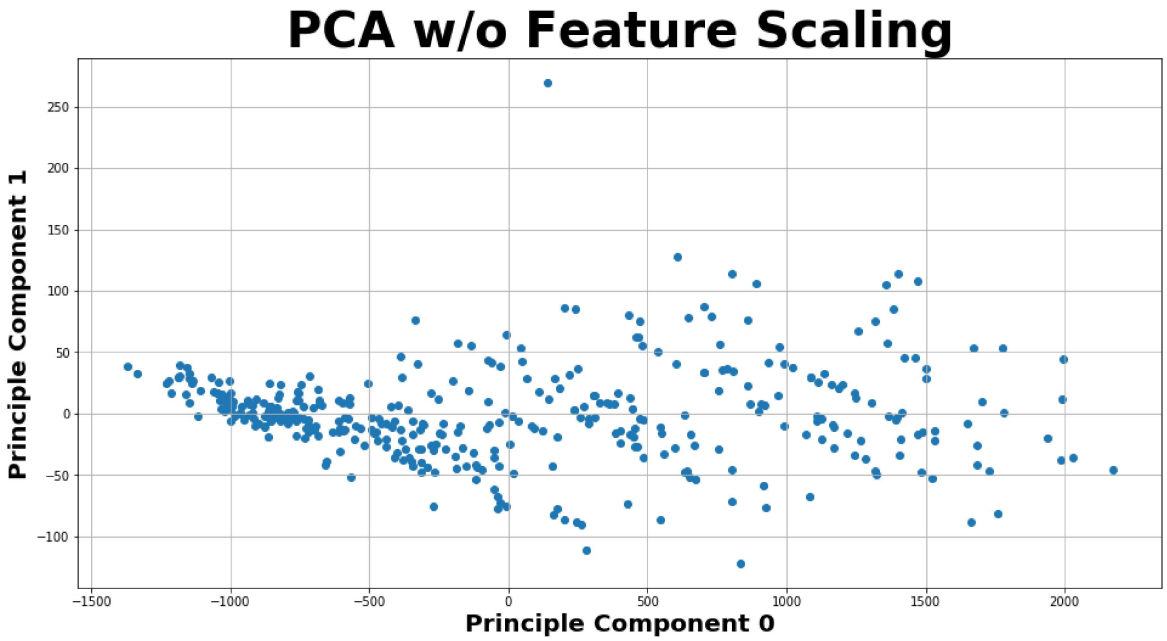
In [73]:
```
# load AUTO data set
AUTO = pd.read_csv("auto.csv").dropna()
AUTO = AUTO.drop('name',axis=1)
AUTO = AUTO[AUTO['horsepower'] != '?']
AUTO
```

Out[73]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin
0	18.0	8	307.0	130	3504	12.0	70	1
1	15.0	8	350.0	165	3693	11.5	70	1
2	18.0	8	318.0	150	3436	11.0	70	1
3	16.0	8	304.0	150	3433	12.0	70	1
4	17.0	8	302.0	140	3449	10.5	70	1
...	...	...	...	...	...	...	...	...
392	27.0	4	140.0	86	2790	15.6	82	1
393	44.0	4	97.0	52	2130	24.6	82	2
394	32.0	4	135.0	84	2295	11.6	82	1
395	28.0	4	120.0	79	2625	18.6	82	1
396	31.0	4	119.0	82	2720	19.4	82	1

392 rows × 8 columns

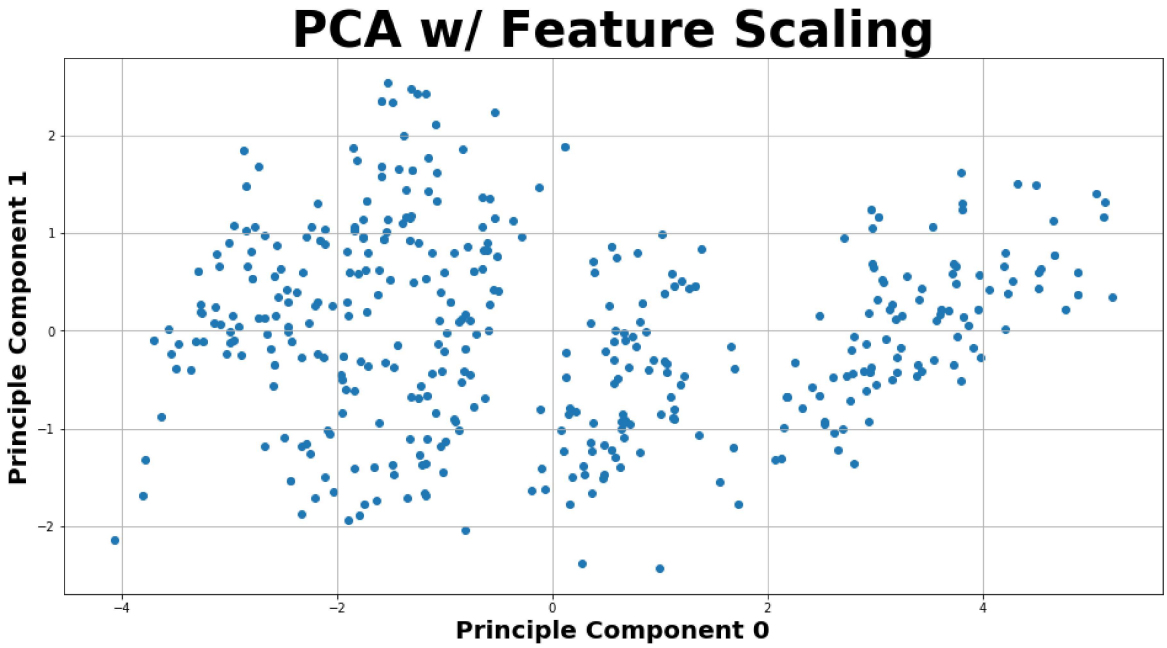
In [74]:
```
# Compute PCA w/o Feature scaling


PCA_obj = PCA(n_components=2)
PCA_obj = PCA_obj.fit(X=AUTO)
X = PCA_obj.transform(AUTO).transpose()

# Visualize
plt.figure(figsize=(16,8))
plt.title("PCA w/o Feature Scaling",size=40,weight='bold')
plt.xlabel("Principle Component 0",size=20,weight='bold')
plt.ylabel("Principle Component 1",size=20,weight='bold')
plt.scatter(X[0],X[1])
plt.grid()
plt.show()
```



2. Compute principal components after scaling features to have unit variance. Plot the result (you can use ggfortify::autoplot).

In [75]:

```
# Scale columsn to unit variance
from sklearn import preprocessing
AUTO_scaled = preprocessing.scale(AUTO)


# Compute PCA w/o Feature scaling
PCA_obj = PCA(n_components=2)
PCA_obj = PCA_obj.fit(X=AUTO_scaled)
X = PCA_obj.transform(AUTO_scaled).transpose()

# Visualize
plt.figure(figsize=(16,8))
plt.title("PCA w/ Feature Scaling",size=40,weight='bold')
plt.xlabel("Principle Component 0",size=20,weight='bold')
plt.ylabel("Principle Component 1",size=20,weight='bold')
plt.scatter(X[0],X[1],)
plt.grid()
plt.show()
```



3. How do the principal components computed in parts 1 and 2 compare?

When not scale, the principle components show data with a much wider range of values throughout. The min and max of each axis cover a large area. When scaled to zero average and unit variance, the resulting principle components show carry a much smaller range of values. This means that the dimensionlity reduction from 7 components to 2 components becoems significantly more concise when scaled appropriately.

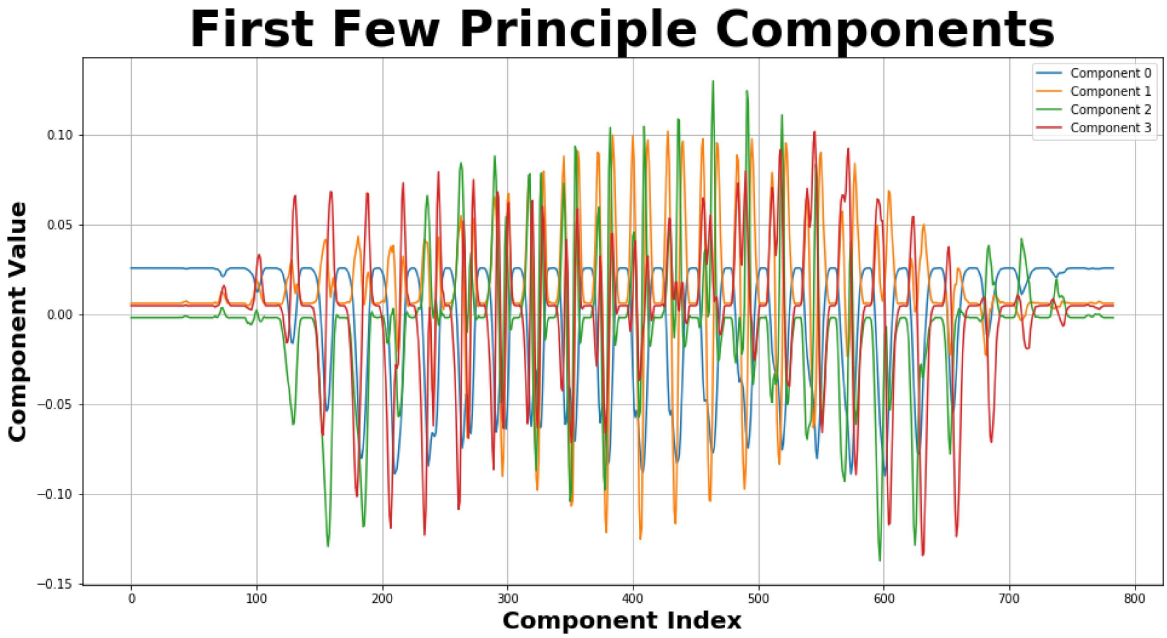
Problem 3 [33%]

1. Apply PCA to a subset of the MNIST dataset. Plot the first few principal vectors. Describe what you observe and interpret the results.

In [76]:
```
# load MNIST data & make training sets
from sklearn.datasets import fetch_openml

X,y = fetch_openml('mnist_784',version=1,return_X_y=True)
X_train,y_train = X[:1000],y[:1000]
```

In [77]:
```
X_train_centered = X_train - np.mean(X_train)
U,S,V_tr = np.linalg.svd(X_train_centered)

C0 = V_tr[0]
C1 = V_tr[1]
C2 = V_tr[2]
C3 = V_tr[3]

idx = np.arange(0,784,1)
```

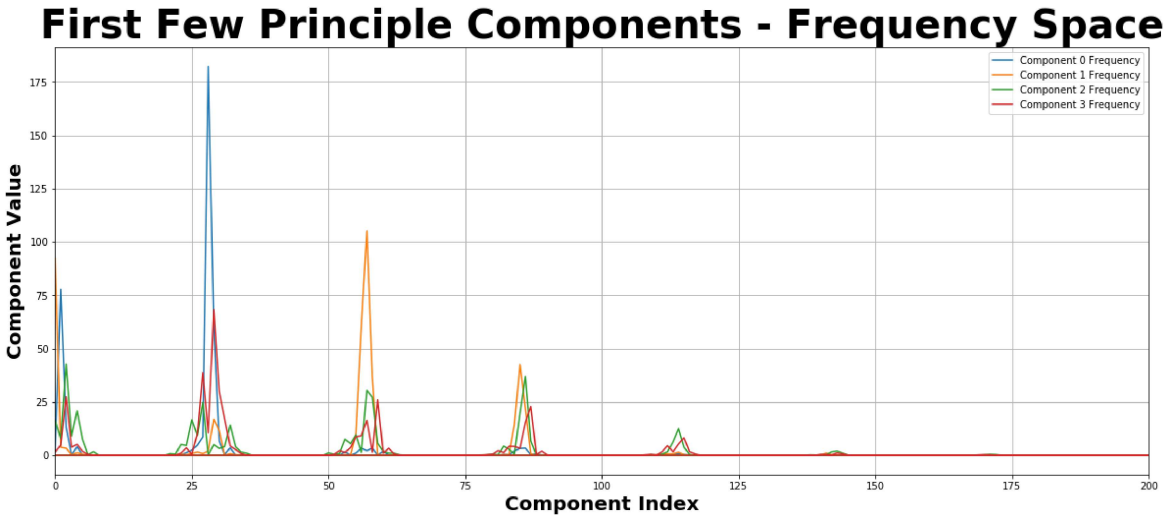
In [78]:
```
plt.figure(figsize=(16,8))
plt.title('First Few Principle Components',size=40,weight='bold')
plt.xlabel('Component Index',size=20,weight='bold')
plt.ylabel('Component Value',size=20,weight='bold')
plt.plot(idx,C0,label='Component 0')
plt.plot(idx,C1,label='Component 1')
plt.plot(idx,C2,label='Component 2')
plt.plot(idx,C3,label='Component 3')
plt.legend()
plt.grid()
plt.show()
```



The plot above is the first four rows of the V^T matrix, which represent the first four principle components of the cenetered design matrix. Each component is 784 indicies long, as expeected to match the 784 feature of the dataset. It seems that the average value of all entries in the array seem to decrease as the component number increases. The ,most striking part is the periodic nature of all of the values in the components.I can apply an FFT to each vector:

In [79]:
```
import scipy.fftpack as fftpack

freq_space = fftpack.fftfreq(n=784,d=1/784) # compute frequnecy space axis
components = np.array([C0,C1,C2,C3])
fft_data = fftpack.fft(components,n=784,axis=-1)
power = np.abs(fft_data)**2
```

In [80]:
```
plt.figure(figsize=(20,8))
plt.title('First Few Principle Components - Frequency Space',size=40,weight='
plt.xlabel('Component Index',size=20,weight='bold')
plt.ylabel('Component Value',size=20,weight='bold')
plt.plot(freq_space,power[0],label='Component 0 Frequency')
plt.plot(freq_space,power[1],label='Component 1 Frequency')
plt.plot(freq_space,power[2],label='Component 2 Frequency')
plt.plot(freq_space,power[3],label='Component 3 Frequency')
plt.xlim(0,+200)
plt.legend()
plt.grid()
plt.show()
```



Bringing the components into frequnecy space indicates that the values in the component indexes have a period of 28 pixels. This makes sense due to the fact that the features are derived from a 28 x 28 images. For each component, the there is a spike every integer multiple of 28 pixels. Iterestingly, the higher the component, the more overtones (the more spikes) that appear higher in the frequency spectrum.

In [ ]: