Hands-on Practice for Module 1: Exploratory Data Analysis

0.Importing important packages

In [200]:

```
# data loading and computing functionality
import pandas as pd
import numpy as np
import scipy as sp

# datasets in sklearn package
from sklearn import datasets
from sklearn.datasets import load_digits

# visualization packages
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm

#PCA, SVD, LDA
from sklearn.decomposition import PCA
from scipy.linalg import svd
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
```

1. Loading data, determining samples, attributes, and types of attributes

Use Davis dataset avaialble at the url https://www.rdocumentation.org/packages/car/versions/2.1-6/topics/Davis (https://www.rdocumentation.org/packages/car/versions/2.1-6/topics/Davis)

Description of the data is provided at https://www.rdocumentation.org/packages/car/versions/2.1-6/topics/Davis (https://www.rdocumentation.org/packages/car/versions/2.1-6/topics/Davis)

Drop rows in the data set with missing values (NA), using dropna(inplace=True) function.

In [201]:

```
davis_df = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/carData/Davis.csv')
```

In [202]:

```
shape_originaldataset=davis_df.shape
df_davis_head=davis_df.head()
shape=davis_df[pd.isnull(davis_df).any(axis=1)].shape

print("shape_originaldataset:" +str(shape_originaldataset))
print("df_davis_head :" +str(df_davis_head))
print("shape :" +str(shape))
```

```
shape_originaldataset:(200, 6)
df_davis_head :   Unnamed: 0 sex  weight  height  repwt  repht
0           1   M      77     182   77.0  180.0
1           2   F      58     161   51.0  159.0
2           3   F      53     161   54.0  158.0
3           4   M      68     177   70.0  175.0
4           5   F      59     157   59.0  155.0
shape :(19, 6)
```

In [203]:

```
davis_df.dropna(inplace=True);
```

Question 1a: What does the data capture?

Answer: Data captures five properties sex,weight,height,reported weight ,reported height of men and women .

Question 1b: Who are selected as subjects in the study that collected the data?

Answer: The selected subjects were men and women engaged in regular exercise.

Question 1c: How many data points are in this dataset?

In [204]:

```
davis_df.shape
```

Out[204]:

```
(181, 6)
```

Answer: 181 Data points are in the dataset.

Question 1d: How many attributes are in this dataset?

Answer: 6 Attributes are there in this dataset.

Question 1e: What type of attributes are present in the dataset?

In [205]:

```
davis_df.dtypes
```

Out[205]:

```
Unnamed: 0      int64
sex            object
weight          int64
height          int64
repwt         float64
repht         float64
dtype: object
```

Answer:The atrribute sex will fall under Categorical -> Nominal attributes and the attributes height,weight ,reported height ,reported weight wil fall under Numerical -> Continuous attributes.

2. Generating summary statistics

Use 'Davis' data. Do not include Unnamed attribute in this analysis.

In [206]:

```
davis_df.drop(columns=davis_df.columns[davis_df.columns.str.contains('unnamed', case=Fa
lse)], inplace=True)
davis_df.head()
```

Out[206]:

	sex	weight	height	repwt	repht
0	M	77	182	77.0	180.0
1	F	58	161	51.0	159.0
2	F	53	161	54.0	158.0
3	M	68	177	70.0	175.0
4	F	59	157	59.0	155.0

Question 2a: What are range of values the numeric attributes take?
[Hint: Use exclude=object option in describe() function to ignore the attribute sex]

In [207]:

```
davis_df.describe(exclude=object)
```

Out[207]:

	weight	height	repwt	repht
count	181.000000	181.000000	181.000000	181.000000
mean	66.303867	170.154696	65.679558	168.657459
std	15.340992	12.312069	13.834220	9.394668
min	39.000000	57.000000	41.000000	148.000000
25%	56.000000	164.000000	55.000000	161.000000
50%	63.000000	169.000000	63.000000	168.000000
75%	75.000000	178.000000	74.000000	175.000000
max	166.000000	197.000000	124.000000	200.000000

Answer: Range of weight[39,166],Range of height[57,197],Range of repwt[41,124],Range of repht[148,200]

Question 2b: What different values do categorical attributes take?
[Hint: Use include=object option in describe() function to ignore the attribute sex]

In [208]:

```
davis_df.describe(include=object)
```

Out[208]:

	sex
count	181
unique	2
top	F
freq	99

In [209]:

```
davis_df.sex.unique()
```

Out[209]:

```
array(['M', 'F'], dtype=object)
```

Answer: The attribute 'SEX' which is categorical takes two values 'M' and 'F'.

Question 2c: What are the mean values for each of the numeric attributes?

In [210]:

```
davis_df.mean()
```

Out[210]:

```
weight     66.303867
height    170.154696
repwt      65.679558
repht     168.657459
dtype: float64
```

Question 2d: What is the variance for each of the numeric attributes?

In [211]:

```
davis_df.var()
```

Out[211]:

```
weight    235.346041
height    151.587047
repwt     191.385635
repht      88.259791
dtype: float64
```

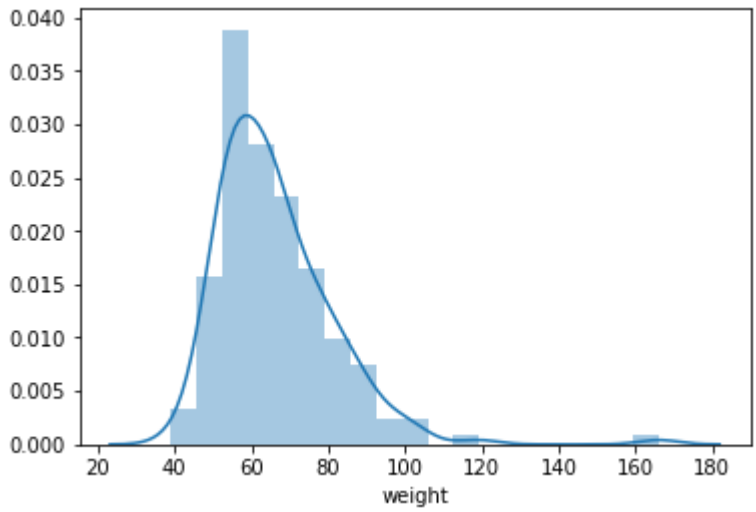
Question 2e: Visually examine how the attribute weight is distributed and comment if the data is Normally distributed?

In [212]:

```
sns.distplot(davis_df['weight'])
```

Out[212]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ad76dc1a490>
```



Answer: The distribution plot for the attribute is not symmetric.

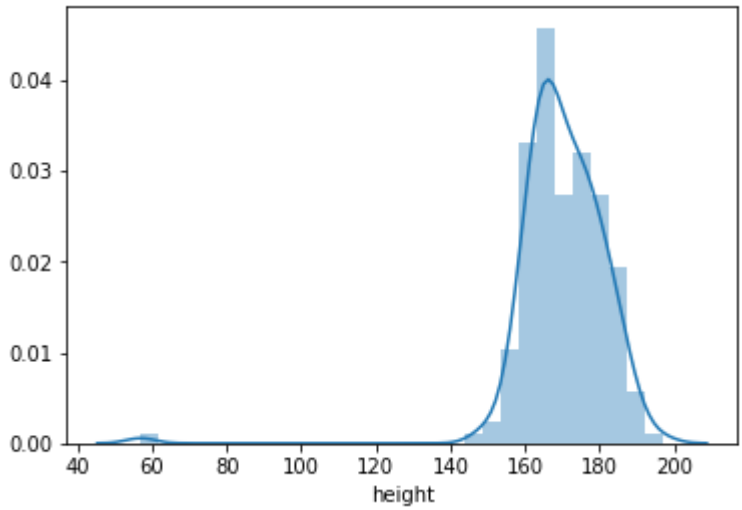
Question 2f: Visually examine how the attribute height is distributed and comment if the data is Normally distributed?

In [213]:

```
sns.distplot(davis_df['height'])
```

Out[213]:

<matplotlib.axes._subplots.AxesSubplot at 0x2ad76dc62f10>



Answer: The distribution plot for the attribute height is not normally distributed as the plot in unsymmetric.

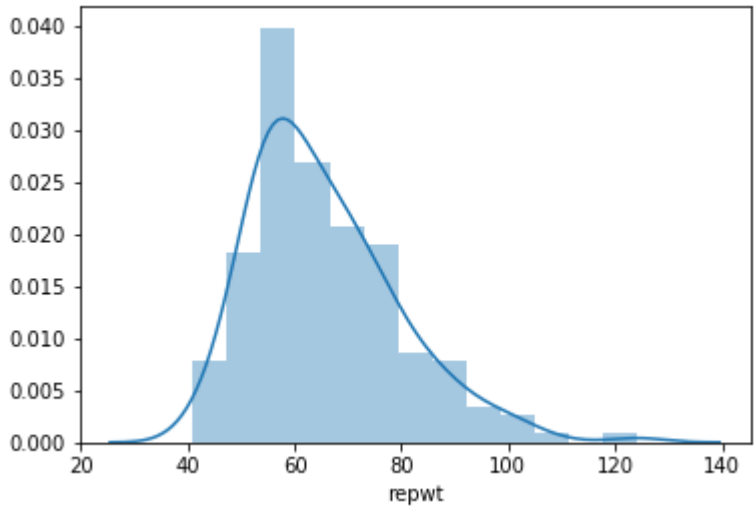
Question 2g: Visually examine how the attribute repwt is distributed and comment if the data is Normally distributed?

In [214]:

```
sns.distplot(davis_df['repwt'])
```

Out[214]:

<matplotlib.axes._subplots.AxesSubplot at 0x2ad76dd7f690>



Answer: The distribution plot for the attribute repwt is not normally distributed as the plot in unsymmetric.

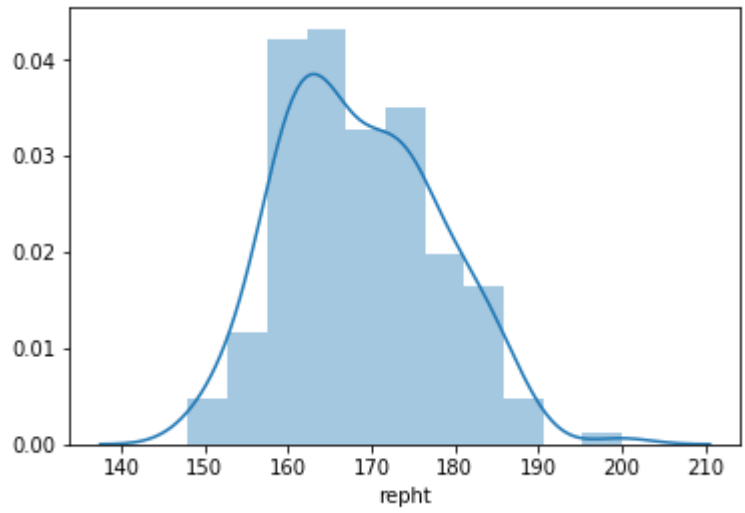
Question 2h: Visually examine how the attribute repht is distributed and comment if the data is Normally distributed?

In [215]:

```
sns.distplot(davis_df['repht'])
```

Out[215]:

<matplotlib.axes._subplots.AxesSubplot at 0x2ad76de1f850>



Answer: The distribution plot for the attribute repht is not normally distributed as the plot in unsymmetric.

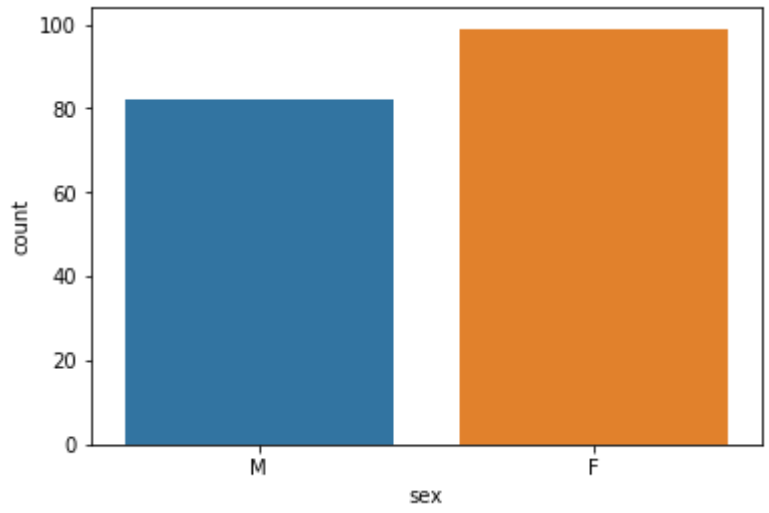
Question 2i: Visually examine how the attribute sex is distributed and comment if the data is uniformly distributed?

In [216]:

```
sns.countplot(davis_df['sex'])
```

Out[216]:

<matplotlib.axes._subplots.AxesSubplot at 0x2ad76de4bc10>



Answer: The data for the attribute 'SEX' is not uniformly distributed as we could see the values were different ,Males were round 80 and Females were around 95.

3. Geometric and Probabilistic view

For this part, we will restrict to repwt and repht attributes in the davis dataset as we can only visualize 2D space.

In [156]:

```
davis_df_new = davis_df[['repwt','repht']]
print(type(davis_df_new))
```

```
<class 'pandas.core.frame.DataFrame'>
```

In [157]:

```
davis_df_new.head()
```

Out[157]:

	repwt	repht
0	77.0	180.0
1	51.0	159.0
2	54.0	158.0
3	70.0	175.0
4	59.0	155.0

Question 3a: Show the Geometric view of this new row normalized data on a 2D space along with the mean.

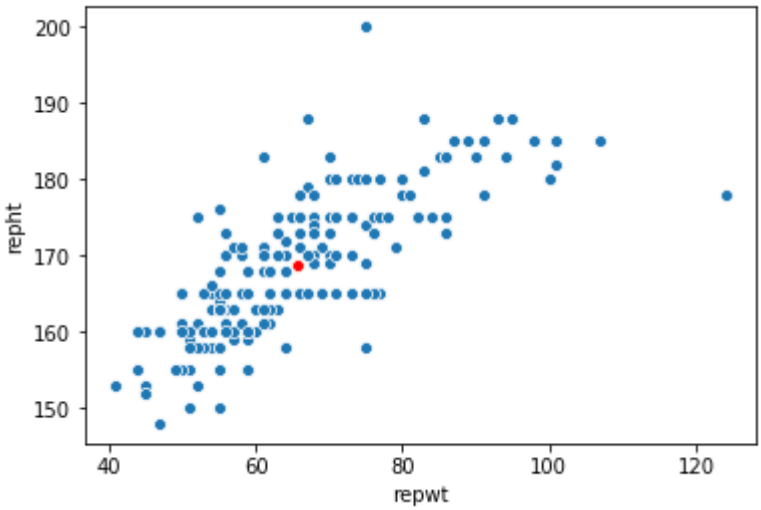
In [75]:

```
fig,fig1=plt.subplots()
sns.scatterplot(x='repwt',y='repht',data=davis_df_new,ax=fig1)
mean=np.mean(davis_df_new.values,0)
print((mean))
sns.scatterplot(x=[mean[0], mean[0]],y=[mean[1], mean[1]],color='r',ax=fig1)
```

```
[ 65.67955801 168.65745856]
```

Out[75]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ad765d3b7d0>
```



We will further normalize the magnitude of each row in the data (davis_df_new) to 1 and use the new dataframe davis_df_new_row_norm.

In [76]:

```
from sklearn.preprocessing import normalize
davis_df_new_row_norm = normalize(davis_df_new, axis=1, norm='l2')
```

In [77]:

```
davis_df_new_row_norm[1:10,:]
#print(type(davis_df_new_row_norm))
```

Out[77]:

```
array([[0.30542755, 0.95221532],
       [0.32340548, 0.94626048],
       [0.37139068, 0.92847669],
       [0.35574458, 0.93458322],
       [0.41835989, 0.90828134],
       [0.42288547, 0.90618314],
       [0.37582461, 0.92669081],
       [0.37595091, 0.92663958],
       [0.35232976, 0.93587592]])
```

Question 3b: Show the Geometric view of this new row normalized data on a 2D space along with the mean. Comment on the Geomateric view of the data in comparison to the view you observed in Question 3a. Provide a reason for the difference in the geometric views in Question 3a and 3b.

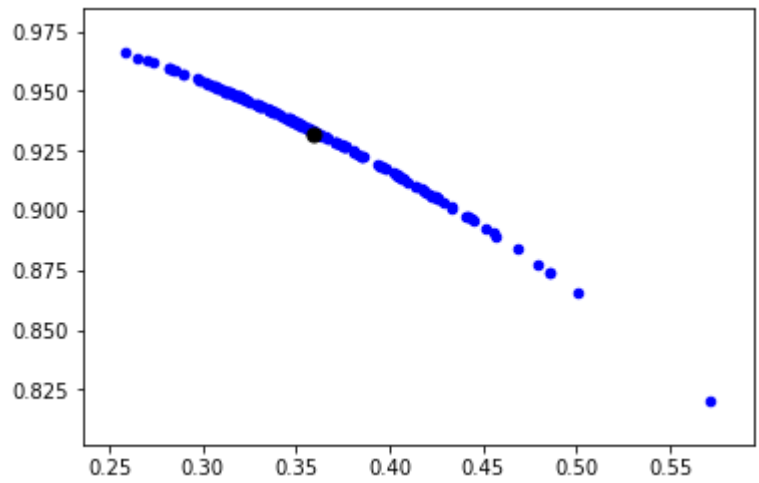
In [78]:

```
mean=np.mean(davis_df_new_row_norm,0)
x=davis_df_new_row_norm[ :,:1]
y=davis_df_new_row_norm[ :,1:2]
print(mean)
plt.scatter(x,y,label='points',color='b',s=20)
plt.scatter(mean[0],mean[1],color='k',label='star',s=50)
#plt.show
#plt.scatterplot(davis_df_new_row_norm)
```

```
[0.35932096 0.93158092]
```

Out[78]:

```
<matplotlib.collections.PathCollection at 0x2ad765de5390>
```



Answer: As we normalized the data the magnitude of the normalized points range from 0 to 1 and the points were very closely aligned when compared with 3a graph ,Mean point is also normalized.

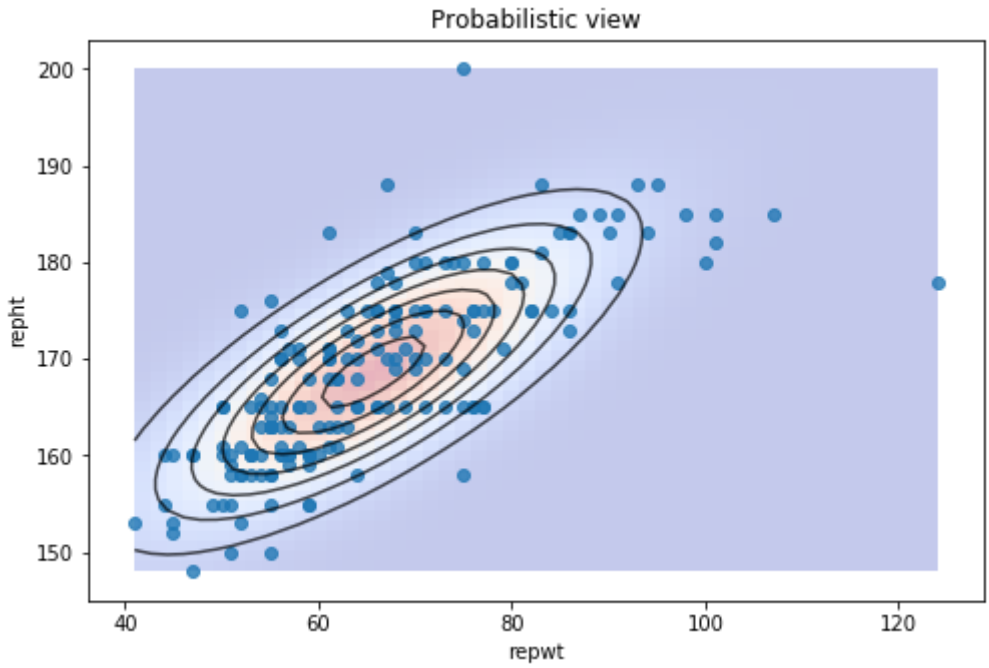
Question 3c: Show the Probabilistic view of the data davis_df_new.

In [79]:

```
from scipy.stats import multivariate_normal
mu=np.mean(davis_df_new.values,0)
sigma=np.cov(davis_df_new.values.transpose())
min_reweight=np.min(davis_df_new.values[ :,0]);
max_reweight=np.max(davis_df_new.values[ :,0]);
min_reheight=np.min(davis_df_new.values[ :,1]);
max_reheight=np.max(davis_df_new.values[ :,1]);
x, y=np.mgrid[min_reweight:max_reweight:50j,min_reheight:max_reheight:50j]
positions=np.empty(x.shape+(2,))
positions[:,:,0]=x;
positions[:,:,1]=y
F = multivariate_normal(mu,sigma)
Z = F.pdf(positions)
fig = plt.figure(figsize=(8,8))
ax = fig.gca()
ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_reweight,max_reweight, min_reheight
,max_reheight], alpha=0.3)
cset = ax.contour(x, y, Z, colors='k', alpha=0.7)
plt.scatter(davis_df_new.values[:,0],davis_df_new.values[:,1],alpha=0.8)
ax.set_xlabel('repwt')
ax.set_ylabel('repht')
plt.title('Probabilistic view')
```

Out[79]:

```
Text(0.5,1,'Probabilistic view')
```



We will normalize the magnitude of each column in the data (davis_df_new) to 1 and use the new dataframe davis_df_new_col_norm.

In [80]:

```
davis_df_new_col_norm = normalize(davis_df_new, axis=0, norm='l2')
```

In [81]:

```
davis_df_new_col_norm[1:10,:]
```

Out[81]:

```
array([[0.05648398, 0.06996539],
       [0.05980657, 0.06952536],
       [0.07752703, 0.07700594],
       [0.06534421, 0.06820526],
       [0.08417221, 0.0726056 ],
       [0.08527974, 0.0726056 ],
       [0.08084962, 0.07920611],
       [0.07863456, 0.07700594],
       [0.07088186, 0.07480577]])
```

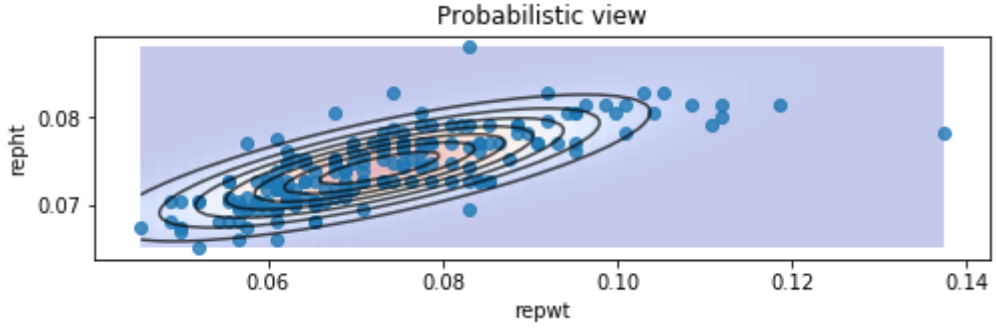
Question 3d: Show the Probabilistic view of the data davis_df_new_col_norm. Compare the shape of the covariance structure in the Gaussian distribution with that of Question 3c and comment if column normalization has affected the shape of the covariance structure.

In [82]:

```
from scipy.stats import multivariate_normal
mu=np.mean(davis_df_new_col_norm,0)
sigma=np.cov(davis_df_new_col_norm.transpose())
min_reweight=np.min(davis_df_new_col_norm[ :,0]);
max_reweight=np.max(davis_df_new_col_norm[ :,0]);
min_reheight=np.min(davis_df_new_col_norm[ :,1]);
max_reheight=np.max(davis_df_new_col_norm[ :,1]);
x, y=np.mgrid[min_reweight:max_reweight:50j,min_reheight:max_reheight:50j]
positions=np.empty(x.shape+(2,))
positions[:,:,0]=x;
positions[:,:,1]=y
F = multivariate_normal(mu,sigma)
Z = F.pdf(positions)
fig = plt.figure(figsize=(8,8))
ax = fig.gca()
ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_reweight,max_reweight, min_reheight
,max_reheight], alpha=0.3)
cset = ax.contour(x, y, Z, colors='k', alpha=0.7)
plt.scatter(davis_df_new_col_norm[:,0],davis_df_new_col_norm[:,1],alpha=0.8)
ax.set_xlabel('repwt')
ax.set_ylabel('repht')
plt.title('Probabilistic view')
```

Out[82]:

```
Text(0.5,1,'Probabilistic view')
```



Answer: The shape of the graph in question 3d is compressed,slightly titlted when compared with the graph 3c and normalization has effected the shape.

4. Understanding the (in)dependencies among attributes using Covariance matrix

Use 'Davis' data. Do not include Unnamed attribute in this analysis.

Question 4a: What is the covariance matrix?

In [158]:

```
data=davis_df.values[:,1:]
print(data[1:10,:])
mean=np.mean(data,axis=0)
print(mean)
davis_df.head()
len(data)
```

```
[[58 161 51.0 159.0]
 [53 161 54.0 158.0]
 [68 177 70.0 175.0]
 [59 157 59.0 155.0]
 [76 170 76.0 165.0]
 [76 167 77.0 165.0]
 [69 186 73.0 180.0]
 [71 178 71.0 175.0]
 [65 171 64.0 170.0]]
[66.30386740331491 170.15469613259668 65.67955801104972 168.6574585635359
3]
```

Out[158]:

181

In [159]:

```
def mycov(data,col_a,col_b):
    mean=np.mean(data,axis=0)
    sum=0;
    for i in range(0,len(data)):
        cola_c=data[i,col_a] -mean[col_a]
        colb_c=data[i,col_b] -mean[col_b]
        value=cola_c*colb_c
        sum=sum+value
        #sum += ((data[i,col_a] - mu[col_a]) * (data[i,col_b] - mu[col_b]))
        return sum/(len(data)-1)
```

In [160]:

```
[ mycov(data,0,0),mycov(data,0,1),mycov(data,0,2),mycov(data,0,3) ]
```

Out[160]:

```
[0.6355958473659407,
 0.7038830045209588,
 0.6726941587049647,
 0.6740073732656372]
```

In [161]:

```
print('Covariance:')
davis_df_drop=davis_df.drop('sex',axis=1)
davis_df_drop.cov()
```

Covariance:

Out[161]:

	weight	height	repwt	repht
weight	235.346041	29.136065	177.292357	91.004665
height	29.136065	151.587047	102.833180	85.497729
repwt	177.292357	102.833180	191.385635	99.017403
repht	91.004665	85.497729	99.017403	88.259791

Question 4b: Which pairs of attributes co-vary in the opposite direction?

Answer: Here the correlation ,covariance between the attributes is positive and none of them are moving in opposite direction with respect to other attribute.

Question 4c: Which pairs of attributes are highly correlated?

In [217]:

```
davis_df_drop.corr()
```

Out[217]:

	weight	height	repwt	repht
weight	1.000000	0.154258	0.835376	0.631435
height	0.154258	1.000000	0.603737	0.739166
repwt	0.835376	0.603737	1.000000	0.761860
repht	0.631435	0.739166	0.761860	1.000000

Answer: Below are the highly correlated pairs: (weight,reported weight) (height,reported height)

Question 4d: Which pairs of attributes are uncorrelated?

Answer: Assuming uncorrelated as weekly correalted ,weight and height and uncorrelated.

Question 4e: What information did you gather from a correlation matrix that is not available in a covariance matrix?

Answer: using the correlation matrix we can see the extent to which the attributes are realated to each other.correlation value 1 or close to 1 indicates that attributes are highly positive correlated and value -1 are close to -1 indicates the attributes are negatively correlated to each other.

5. Dimensionality Reduction: Feature Selection

Data: Iris dataset from the practice notebook. (https://raw.githubusercontent.com/plotly/datasets/master/iris.csv (https://raw.githubusercontent.com/plotly/datasets/master/iris.csv))

Assumption: Assume that your goal is to cluster the data to identify the species 'Name'. Clustering algorithm takes as input data points and attributes. It groups points that are similar to each other into a separate cluster. It puts points that are dissimilar in different cluster. Note that the 'Name' attribute will be hidden from the clustering algorithm.

In [163]:

```
import seaborn as sns
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/iris.cs
v')
```

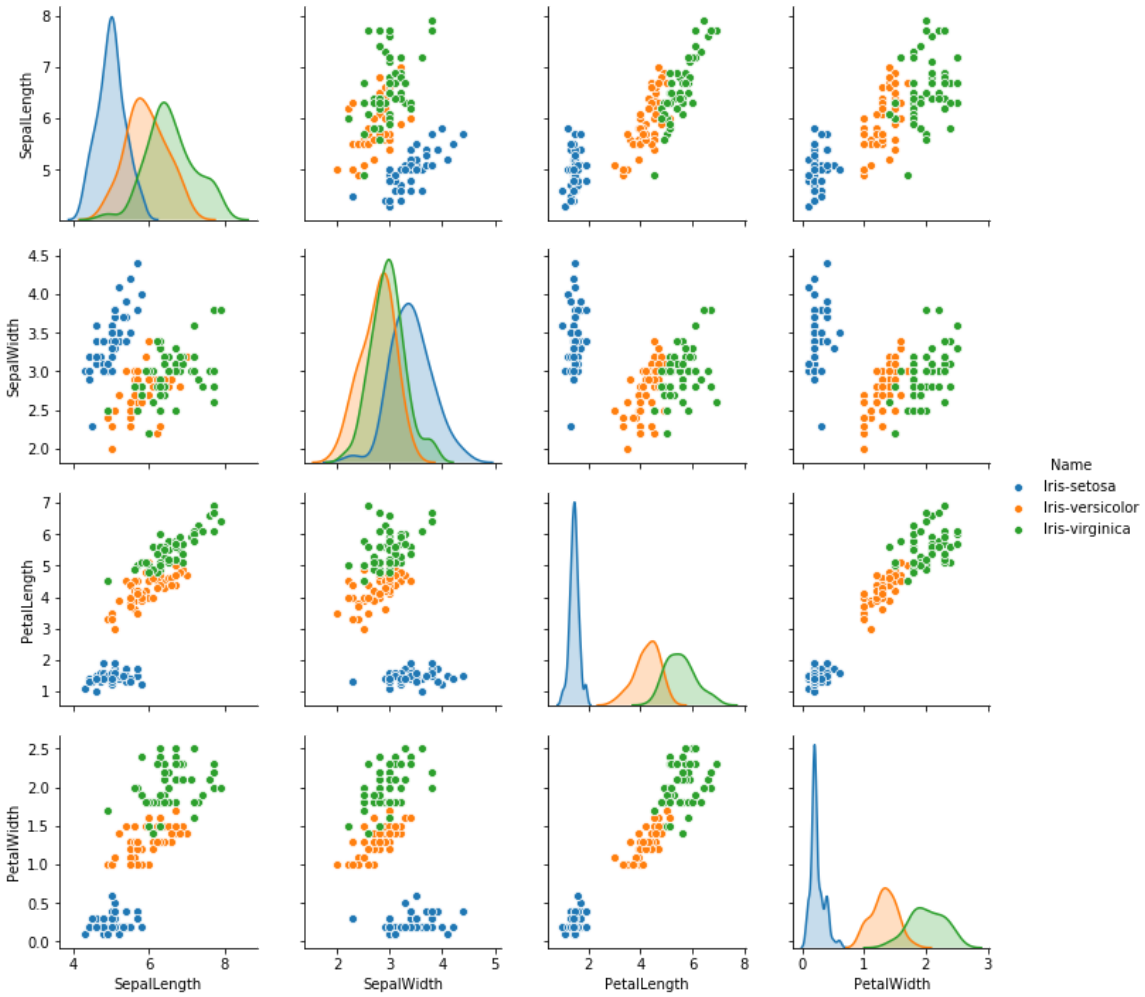
Question 5a: If you are allowed to select only one attribute, which attribute would be highly useful for the clustering task. Provide a reason. Use pairplot to answer this question.

In [164]:

```
sns.pairplot(iris_df,hue='Name')
```

Out[164]:

```
<seaborn.axisgrid.PairGrid at 0x2ad76a45c090>
```



Answer: Petal Length would be the attribute which provides more information by for clustering task as the overlap is less betwwen the flower names.

Question 5b: If you are allowed to select only two features, which feature would be highly useful for the clustering task. Provide a reason. Use pairplot to answer this question.

Answer: PetalLength and PetalWidth would be the pair of attributes which provides useful information for clustering the task.

Question 5c: In real-world problems ground-truth (types of iris plants) will not be available to select the features, how do you perform **feature selection** in that case?

Answer: Feature selection can be identified by clustering and identifying different clusters in the plot from that we can derive some conclusions like how those clusters vary with different attibutes.

Question 5d: In real-world problems ground-truth (types of iris plants) will not be available to select the features, how do you perform **dimensionality reduction** in that case? What limitations does your approach have?

Answer: In real world problems if were unable to perform feature selection , we can perform dimensionality reduction on clusters that exists.The limitation of this approach is if we accidentially miss a single feature which would important in determining the target variable then our model will fail to interpret.

6. Dimensionality Reduction: PCA on Iris Data

Question 6a: Perform PCA on Iris dataset and project the data onto the first two principal components. Use the attributes 'SepalLength','SepalWidth','PetalLength', and 'PetalWidth'.

Hint: Use iris_df[['SepalLength','SepalWidth','PetalLength','PetalWidth']] to use the specified attributes.

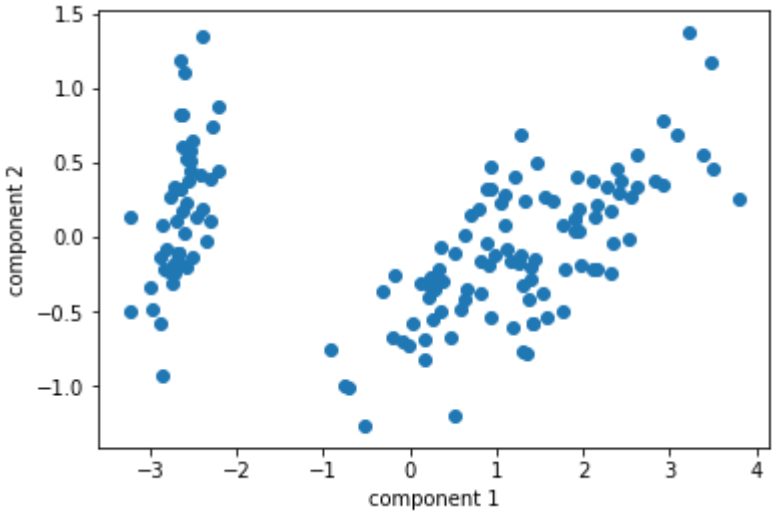
In [167]:

```
pca=PCA(2)
projected = pca.fit_transform(iris_df[['SepalLength','SepalWidth','PetalLength','PetalW
idth']] )
#print(projected)
print(type(projected))
plt.scatter(projected[:, 0], projected[:, 1] )
plt.xlabel('component 1')
plt.ylabel('component 2')
```

```
<type 'numpy.ndarray'>
```

Out[167]:

```
Text(0,0.5,'component 2')
```



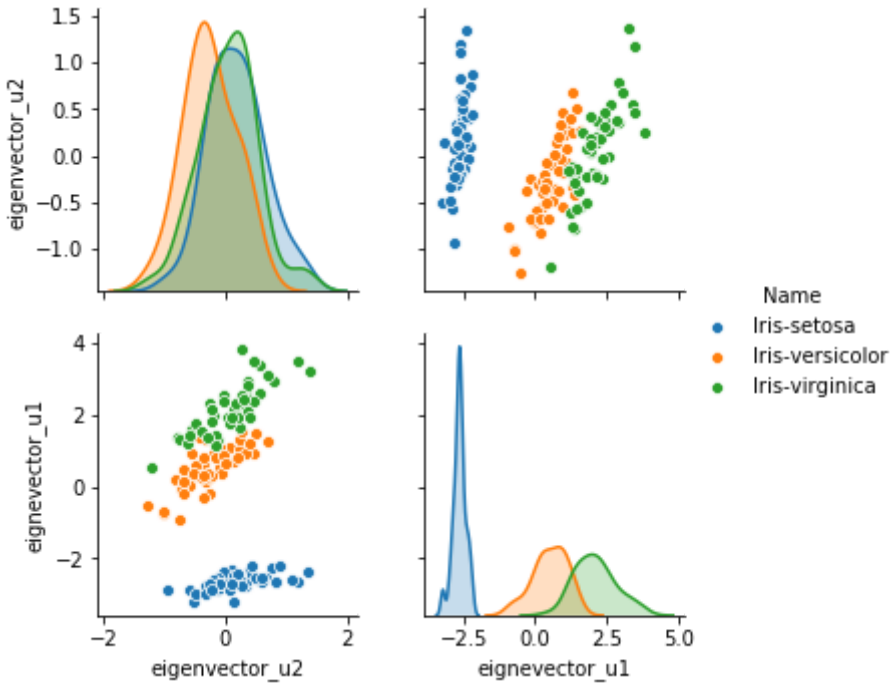
Question 6b: Generate a pairplot (along with colors for the different types of iris plants) between the two newly generated features using PCA in the above step.

In [168]:

```
#As we can't use pairplot which is seaborn fundtion with Nd array we have to convert th
e ndarray to dataframe.
#TypeError: 'data' must be pandas DataFrame object, not: <type 'numpy.ndarray'>
iris_df2=pd.DataFrame({'eignevector_u1':projected[:,0],'eigenvector_u2':projected[:,1
]})
pair_plot_df2=pd.concat([iris_df2,iris_df.iloc[:,4]],axis=1)
sns.pairplot(pair_plot_df2,hue='Name')
```

Out[168]:

```
<seaborn.axisgrid.PairGrid at 0x2ad766df6d90>
```



Question 6c: From the above pairplot, if only one newly generated attribute were to be used for clustering the data which newly generated attribute is best suited. Provide a reason. Is the newly generated attribute better than the feature selected in Question 4a?

Answer: eigenvector_u1 is the best attribute suited for clustering the data and not much better than the attribute mentioned in 5a

Question 6d: From the above pairplot, if two newly generated attributes were to be used for clustering the data, are the two newly generated attributes better than the features selected in Question 4b?

Answer: Yes, the newly generated attributes are better than the features selected in 5b because the overloop between the class is very minimal.

7. Dimensionality Reduction: PCA on synthetic datasets

Consider the following synthetic dataset we refer to as **Blobs**. This dataset has 500 data points centered around (-5, -5), (0,0) and (5,5). This dataset has 1500 data points and 2 attributes.

In [169]:

```
n_samples = 1500
random_state = 42
centers = [(-5, -5), (0, 0), (5, 5)]
Blobs_X, Blobs_y = datasets.make_blobs(n_samples=n_samples,centers=centers,random_state
=random_state)
```

In [170]:

```
Blobs_X.shape
```

Out[170]:

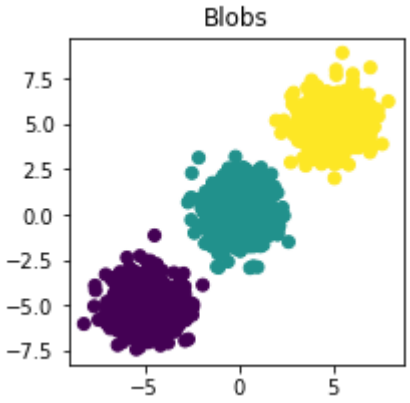
```
(1500, 2)
```

In [171]:

```
plt.figure(figsize=(3,3))
plt.scatter(Blobs_X[:, 0], Blobs_X[:, 1], c= Blobs_y)
plt.title('Blobs')
```

Out[171]:

```
Text(0.5,1,'Blobs')
```



We generated a new dataset **Blobs1** by adding an extra attribute to this 2D Blobs dataset. The values for this new attribute are drawn from a normal distribution with mean 0 and variance 1.

In [172]:

```
Blobs1= pd.DataFrame(Blobs_X)
Blobs1['2'] = np.random.randn(1500)
Blobs1.head()
```

Out[172]:

	0	1	2
0	0.168461	1.317598	0.150744
1	-3.534351	-5.225776	0.343380
2	-6.525525	-5.691908	0.966367
3	-0.120948	0.419532	-0.163757
4	-5.469474	-4.457440	-0.675750

We generated a new dataset **Blobs2** by adding an extra attribute to the 2D Blobs dataset. The values for this new attribute are drawn from a normal distribution with mean 0 and variance 100. Read more about how to do this at https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.random.randn.html (https://docs.scipy.org/doc/numpy-1.15.1/reference/generated/numpy.random.randn.html).

In [173]:

```
Blobs2= pd.DataFrame(Blobs_X)
Blobs2['2'] = np.random.randn(1500)*10
Blobs2.head()
```

Out[173]:

	0	1	2
0	0.168461	1.317598	-13.176320
1	-3.534351	-5.225776	-2.903378
2	-6.525525	-5.691908	0.908439
3	-0.120948	0.419532	-11.743623
4	-5.469474	-4.457440	8.886506

We generated a new dataset **Blobs3** by adding two extra attributes to the 2D Blobs dataset. The values for the two new attributes are drawn from a normal distribution with mean 0 and variance 100.

In [174]:

```
Blobs3= pd.DataFrame(Blobs_X)
Blobs3['2'] = np.random.randn(1500)*10
Blobs3['3'] = np.random.randn(1500)*10
Blobs3.head()
```

Out[174]:

	0	1	2	3
0	0.168461	1.317598	-2.973798	13.320075
1	-3.534351	-5.225776	-24.155910	-12.560644
2	-6.525525	-5.691908	10.752309	10.275300
3	-0.120948	0.419532	-10.056817	0.107939
4	-5.469474	-4.457440	-9.448775	2.044552

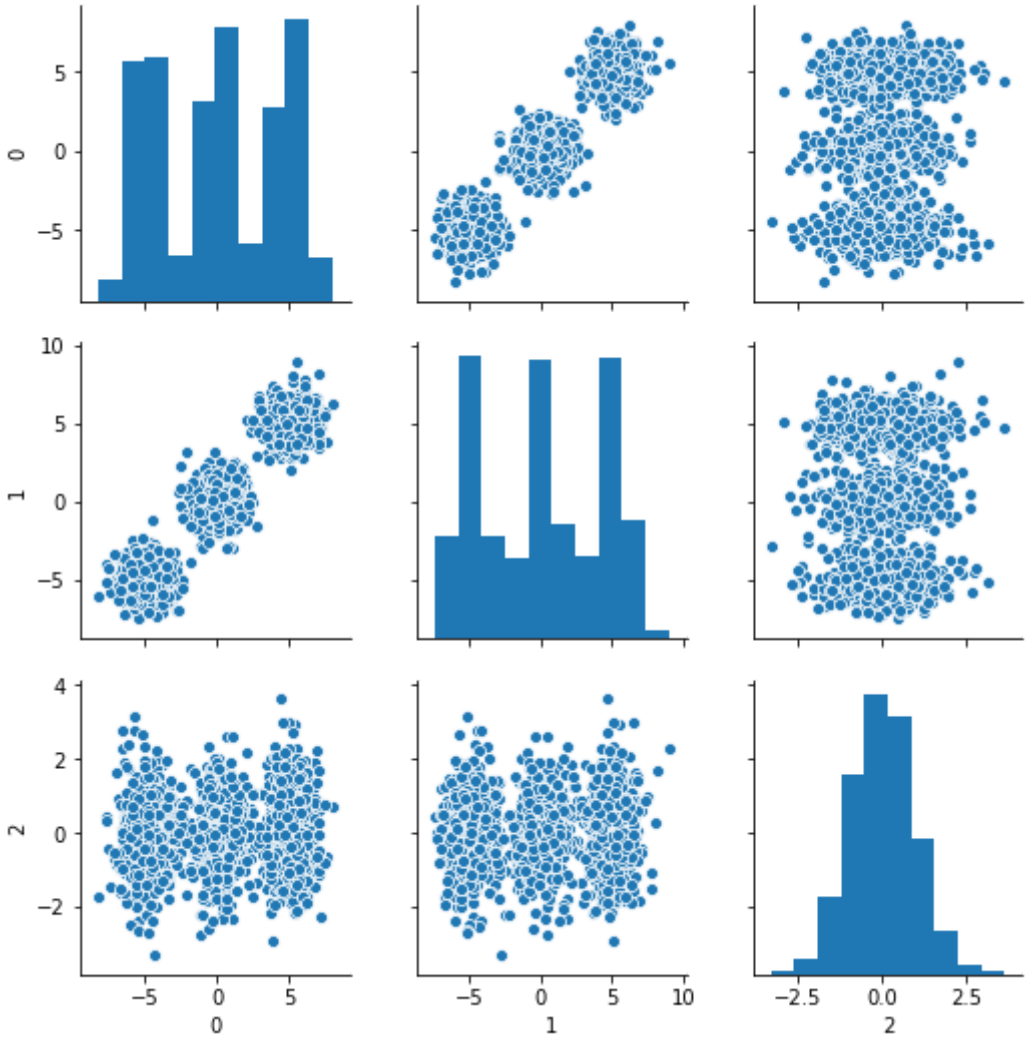
Question 7a: Plot pairplot for **Blobs1** data. By visually examining this plot, comment on the variance of the third attribute in comparison to the first two attributes.

In [175]:

```
sns.pairplot(Blobs1)
Blobs1.cov()
```

Out[175]:

	0	1	2
0	17.570371	16.649141	0.109715
1	16.649141	17.643028	0.133978
2	0.109715	0.133978	0.927399



Answer: The dispersion of the points for the three attribute is symmteric with mean cenetred zero and it is normally distributed and if we draw a line connecting the midpoints at every step ,it forms a bell shaped curve.

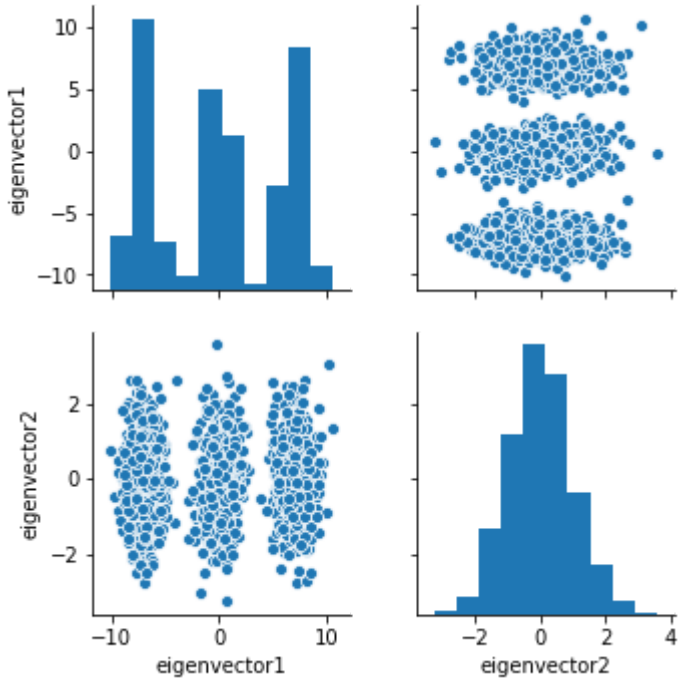
Question 7b: Perform PCA on **Blobs1** data. Project data onto the first two principal components. Generate a pairplot for the newly constructed attributes.

In [176]:

```
pca=PCA(2)
projected = pca.fit_transform(Blobs1)
columns=['eigenvector1','eigenvector2']
projected_pair_plot=pd.DataFrame(data=projected[:,:],columns=columns)
sns.pairplot(projected_pair_plot)
```

Out[176]:

```
<seaborn.axisgrid.PairGrid at 0x2ad76b8ba390>
```



Question 7c: By comparing the distributions for the newly generated attributes in Question 7b with the previous pairplot in Question 7a, determine which attribute is captured by the first principal component and which attribute is captured by the second principal component. Provide a reason for your observations.

Answer: The attribute '1' is captured by the first Prinicpal component and attribute '2' is captured by the second Principal component.

The reason for this even though the attributes '1' and '0' having the variances greater than the attribute '2' the attribute and '0' and '1' are highly correlated and project the variance in same direction so that whole projected variance be along PC1 and as '1' attribute having more than '0' attribute it would play dominant role in PC1 and the attribute '2'will be projected along the other PC.

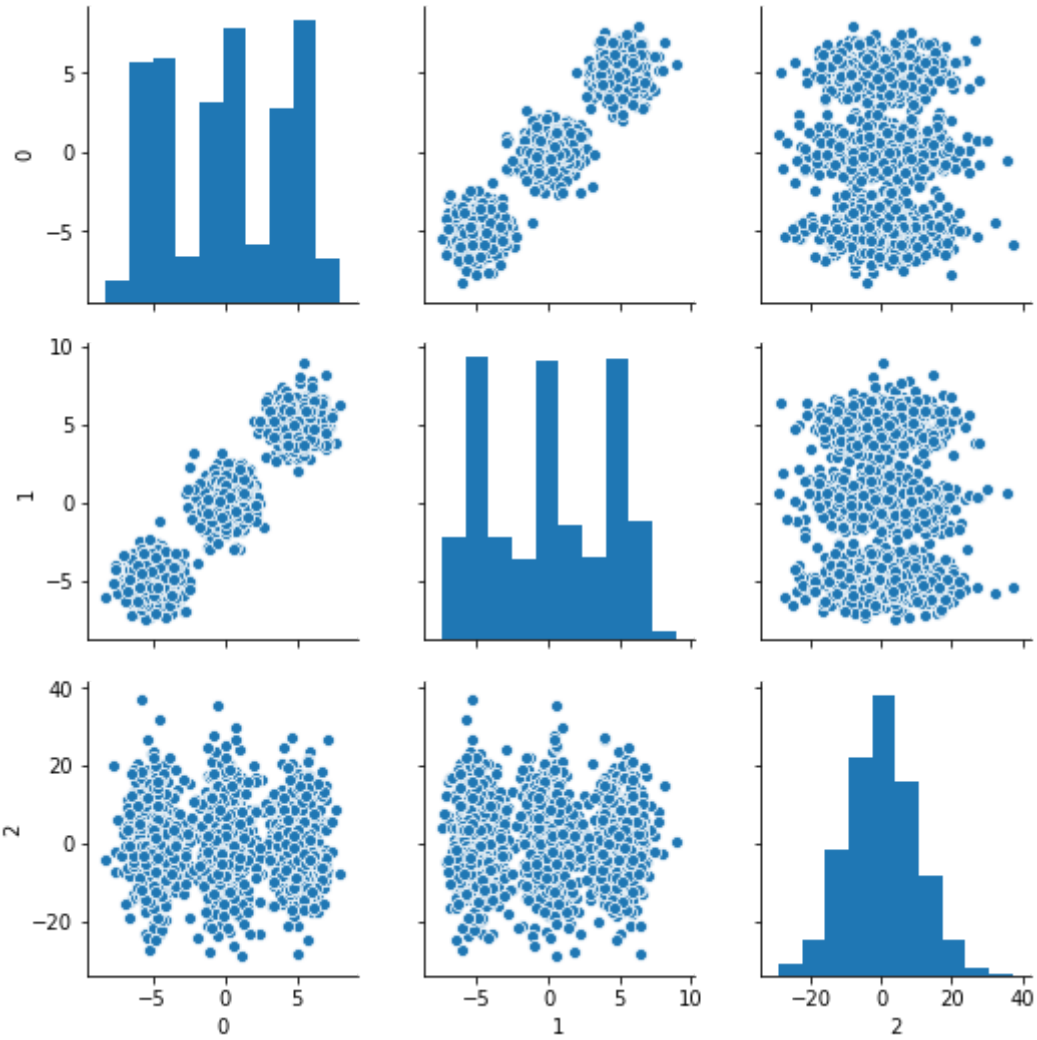
Question 7d: Plot pairplot for **Blobs2** data. By visually examining this plot, comment on the variance of the third attribute in comparison to the first two attributes.

In [177]:

```
sns.pairplot(Blobs2)
```

Out[177]:

<seaborn.axisgrid.PairGrid at 0x2ad76c287e10>



Answer: The '2' attribute is having more variance than '1' and '0' attributes.

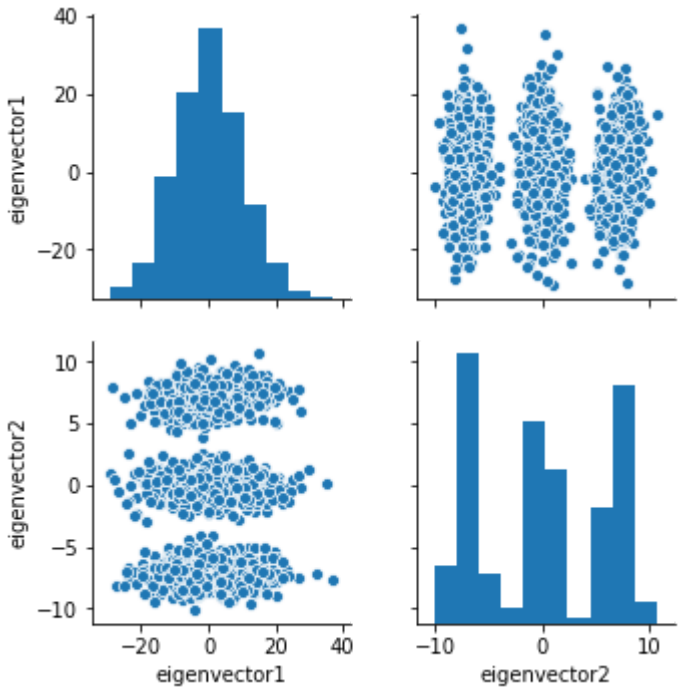
Question 7e: Perform PCA on **Blobs2** data. Project data onto the first two principal components. Generate a pairplot for the newly constructed attributes.

In [178]:

```
pca=PCA(2)
projected_2=pca.fit_transform(Blobs2)
columns=['eigenvector1','eigenvector2']
projected_pair_plot2=pd.DataFrame(data=projected_2[:,:],columns=columns)
sns.pairplot(projected_pair_plot2)
```

Out[178]:

```
<seaborn.axisgrid.PairGrid at 0x2ad76c6cb690>
```



Question 7f: By comparing the distributions for the newly generated attributes in Question 7e with the previous pairplot in Question 7d, determine which attribute is captured by the first principal component and which attribute is captured by the second principal component. Why would have caused this (in comparison to your observation in Question 7c)?

Answer: The attribute '2' is captured by the first Principal component and attribute '1' is captured by second principal component,The reason for interchanging of the PC's when compared with 7c was beacause of Blobs2 '2' attribute is created by randn function with variance 100 times of the Blobs1 '2' attribute and as per definition first PC caputres the maximum projected variance.

Question 7g: Are the three blobs separately visible after projection based on PCA in Question 7e?

Answer: Yes,the Blobs were separately visible after projection them to lower dimension using PCA.

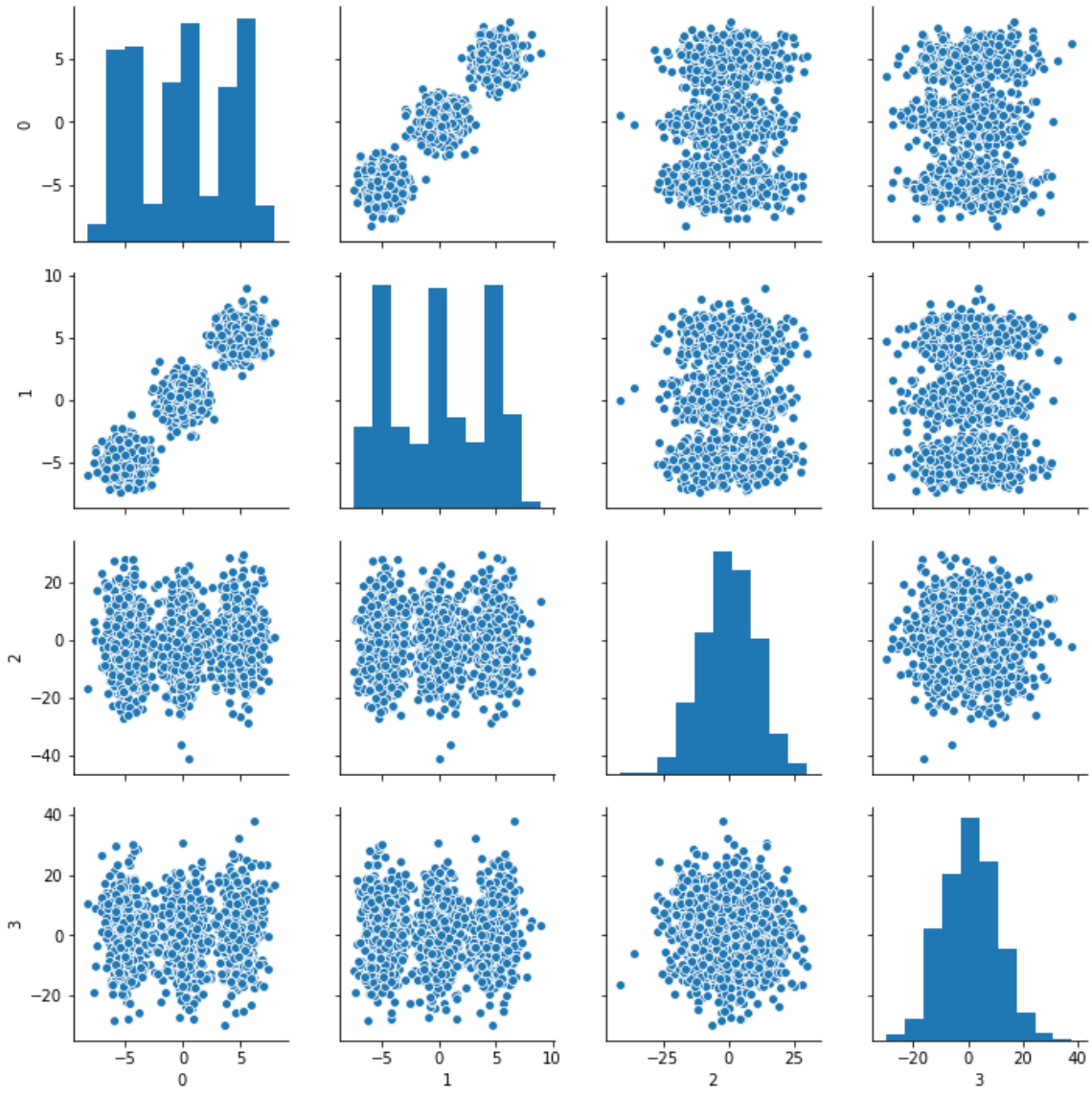
Question 7h: Plot pairplot for **Blobs3** data. By visually examining this plot, comment on the strength of the correlation between the first two attributes. Also, comment on the strength of the correlation between the second two attributes.

In [179]:

```
sns.pairplot(Blobs3)
```

Out[179]:

```
<seaborn.axisgrid.PairGrid at 0x2ad76c6cb9d0>
```



Answer: From the plots the first two attributes(1,2) are highly correlated and other the two attributes(3,4) are weekly correlated.

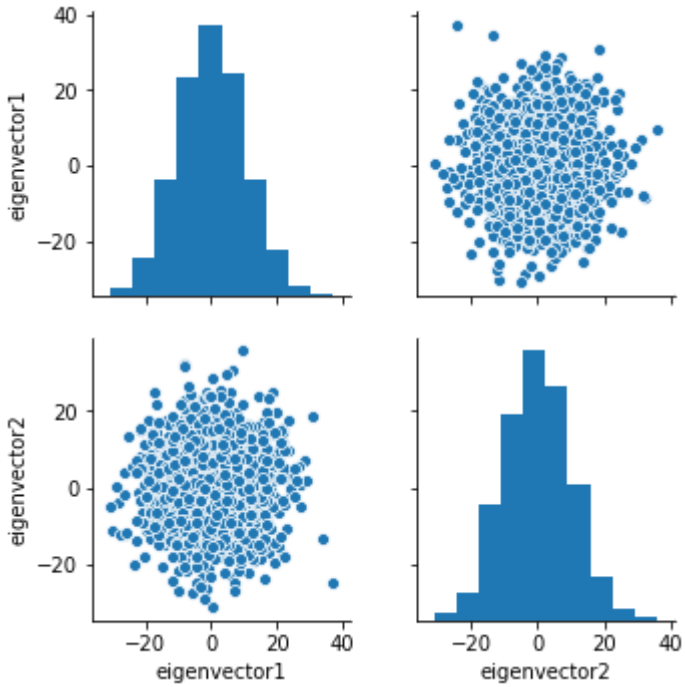
Question 7i: Perform PCA on **Blobs3** data. Project data onto the first two principal components. Generate a pairplot for the newly constructed attributes.

In [180]:

```
pca=PCA(2)
projected_3=pca.fit_transform(Blobs3)
columns=['eigenvector1','eigenvector2']
projected_pair_plot3=pd.DataFrame(data=projected_3[:,:],columns=columns)
sns.pairplot(projected_pair_plot3)
```

Out[180]:

```
<seaborn.axisgrid.PairGrid at 0x2ad76c6cb250>
```



Question 7j: By comparing the distributions for the newly generated attributes in Question 7i with the previous pairplot in Question 7h, determine which attribute is captured by the first principal component and which attribute is captured by the second principal component. Why would have caused this (in comparison to your observation in Question 7f and 7c)?

Answer: The attribute '3' is captured by the first principal component and attribute '2' is captured by the second prinicpal component . 1)The reason is Blobs1 created with an extra attribute '2' with variance one ,after performing PCA attribute '1' is captured by first PC and attribute '2' is captured by second PC and from that we can say that attribute one is having variance greater than one. 2)Blobs2 created with an extra attribute '2' with variance 100 after performing PCA,attribute '2' is captured by first PC and attribute '1' is captured by second PC as attribute '2' is having variance 100 so from this we can say that attribute '1' is having variance greater than one and less than 100. 3)When Blobs3 created with an extra attributes '2','3' with variances 100 and after performing PCA attribute '3' is captured by the first principal component and attribute '2' is captured by the second prinicpal component as PC's capture the maximum projected variance attributes and attribute '1' is having variance less than 100 it was ignored in PC's.

Question 7k: Are the three blobs separately visible after projection based on PCA in Question 7i? What would have caused this, in comparison to your observation in Question 7g?

Answer: Blobs are not seperately visible in the plot 7i because when we are projecting the data from 4d to 2d using PCA which captures the maximum projected variances across the PC's ,it ignored the blobs data contained by '0' and '1' attribute and also this would be of the curse of high dimesnionality when set of points lie close to any of the axes those points would be inivisble in lower dimensional projection.

Question 7l: What limitation of PCA do your observations in Questions 7j, 7f, and 7c highlight?

Answer:1)In 7c the properties of the '0' attribute are ignored because '0' and '1' are highly correlated and as '1' is having more variance than '0' attribute those properties were reflected on PC1 and attribute '2' variance is projected on PC2.

2)It ignores the properties/representation of the attributes with less variances when attributes with larger variances were exists in higher dimensions when projected to lower dimensions .Example:suppose 8 attributes with variances 2,3,4,5,100,200,300,400 in 8 'd'when projected to 4d using PCA it ignores the properties of the attributes 1,2,3,4.

3)when points lie close to the axes in higher dimensions and performed PCA to project points to lower dimensions those would be invisible .

8. Singular Value Decomposition

Question 8a: Using the code provided in the practice notebook for computing PCA, write your own SVD function (U,S,V = mysvd(A)) to factorize the matrix A into U,S, and V.

In [181]:

```
def mysvd(A):
    u=np.matmul(A,np.transpose(A))
    v=np.matmul(np.transpose(A),A)
    lambda_u,eigen_u=np.linalg.eig(u)
    lambda_v,eigen_v=np.linalg.eig(v)
    (l,m)=np.shape(A)
    lambda_u[lambda_u<0]=0
    s=np.sqrt(lambda_u)
    diagonal_matrix=np.empty([l,m])
    for i in range(0,l):
        for j in range(0,m):
            if i==j:
                diagonal_matrix[i,j]=s[i]
            else :
                diagonal_matrix[i,j]=0
    return(eigen_u,diagonal_matrix,eigen_v)

A = np.array([
    [1, 1, 1, 0, 0, 0],
    [3, 3, 3, 0, 0, 0],
    [4, 4, 4, 0, 0, 0],
    [5, 5, 5, 0, 0, 0],
    [0, 1, 0, 4, 4, 1],
    [0, 0, 0, 5, 5, 2],
    [0, 0, 0, 2, 2, 2]])

u,s,v=mysvd(A)
```

Question 8b: Demonstrate that your code is correct by using your function on the following matrix A and showing that the product $USV^T = A$.

In [114]:

```
A = np.array([
    [1, 1, 1, 0, 0, 0],
    [3, 3, 3, 0, 0, 0],
    [4, 4, 4, 0, 0, 0],
    [5, 5, 5, 0, 0, 0],
    [0, 1, 0, 4, 4, 1],
    [0, 0, 0, 5, 5, 2],
    [0, 0, 0, 2, 2, 2]])
```

In [121]:

```
u,s,v=mysvd(A)


B=np.matmul(np.matmul(u,s),np.transpose(v))

K=np.absolute(A)
M=np.absolute(B)

print(K,M)
```

```
(array([[1, 1, 1, 0, 0, 0],
       [3, 3, 3, 0, 0, 0],
       [4, 4, 4, 0, 0, 0],
       [5, 5, 5, 0, 0, 0],
       [0, 1, 0, 4, 4, 1],
       [0, 0, 0, 5, 5, 2],
       [0, 0, 0, 2, 2, 2]]), array([[0.98763913, 1.00367369, 0.98763903,
0.14056221, 0.14056221,
        0.04755794],
       [2.9629172 , 3.01102106, 2.96291728, 0.42168662, 0.42168665,
        0.14267381],
       [3.95055634, 4.01469474, 3.9505563 , 0.56224886, 0.56224882,
        0.19023175],
       [4.9381954 , 5.01836843, 4.9381954 , 0.70281105, 0.70281106,
        0.23778968],
       [0.8177649 , 0.5401995 , 0.8177649 , 3.98468166, 3.98468166,
        0.78442958],
       [0.76330776, 0.02587753, 0.76330776, 4.89878606, 4.89878606,
        2.19950977],
       [0.04053717, 0.57629037, 0.04053717, 2.02893143, 2.02893143,
        1.85242473]]))
```

Question 8c: Perform SVD on iris dataset and visualize the proportion of variance captured by each spectral value. List the dimensions that captures less than 10% of the total variance.

In [122]:

```
import pandas as pd
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/iris.csv')
```

In [123]:

```
data = iris_df.values[:,0:4]
data = data.astype(float) #converts data format from object to numeric
```

In [129]:

```
U, S ,V=svd(data,full_matrices = False)

np.cumsum(S)/np.sum(S)

np.shape(S)


print(S)
```

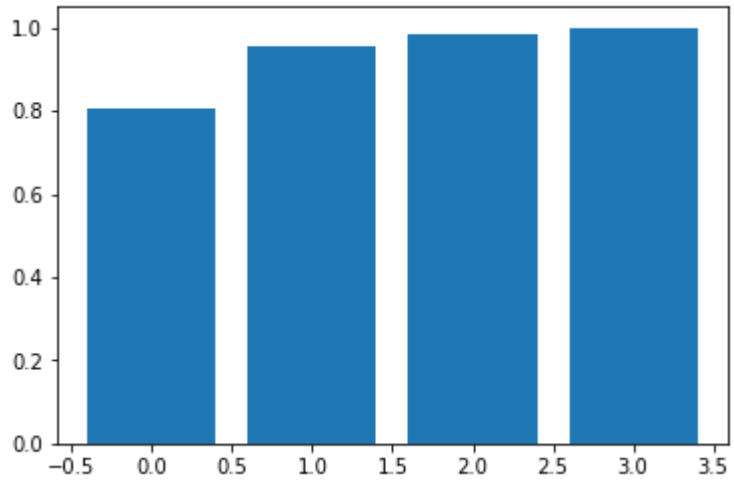
```
[95.95066751 17.72295328  3.46929666  1.87891236]
```

In [131]:

```
plt.bar(np.arange(4),np.cumsum(S)/np.sum(S))
```

Out[131]:

```
<BarContainer object of 4 artists>
```



Answer: Here the spectral values 3,4 cover less than 10% total variance.

Question 8d: The heatmap of the full data is shown below. Plot all the four spectral decomposition matrices based on SVD.

In [ ]:

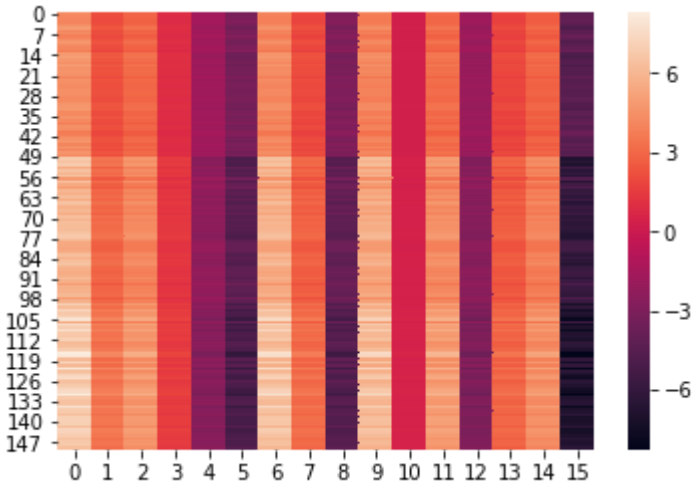
```
sns.heatmap(data,vmin=0, vmax=7)
```

In [134]:

```
sns.heatmap(S[0]*np.outer(U[:,0],V[0:]))
```

Out[134]:

<matplotlib.axes._subplots.AxesSubplot at 0x2ad768b25cd0>

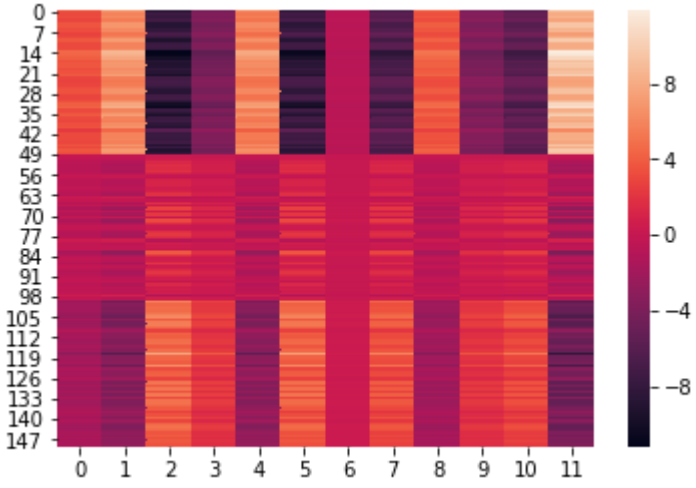


In [135]:

```
sns.heatmap(S[0]*np.outer(U[:,1],V[1:]))
```

Out[135]:

<matplotlib.axes._subplots.AxesSubplot at 0x2ad769e59810>

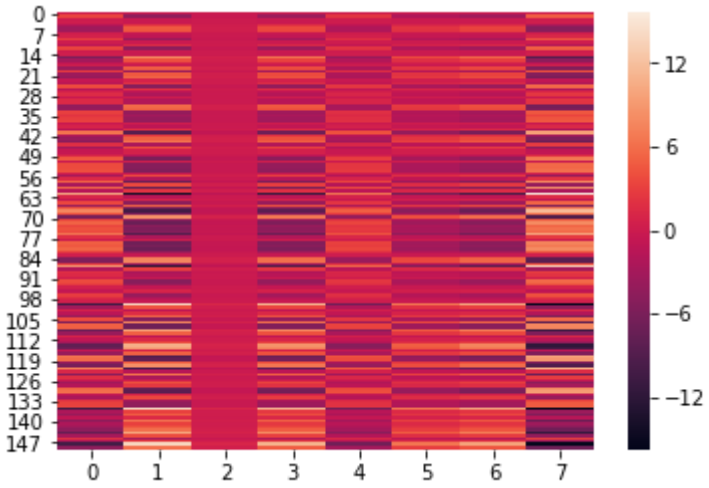


In [136]:

```
sns.heatmap(S[0]*np.outer(U[:,2],V[2:]))
```

Out[136]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ad769f61290>
```

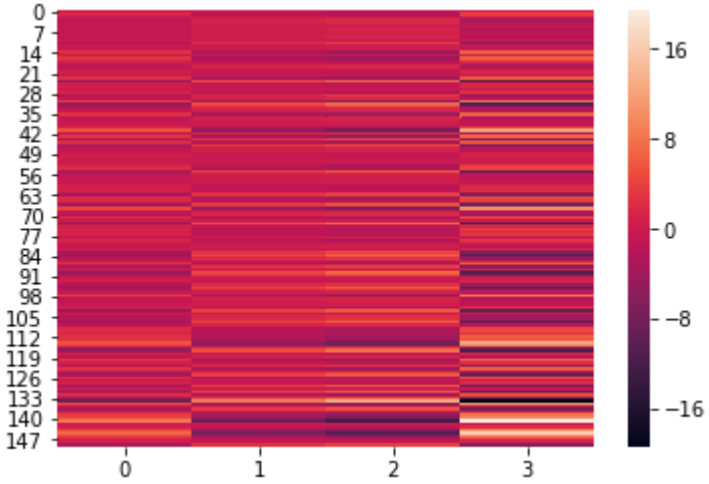


In [137]:

```
sns.heatmap(S[0]*np.outer(U[:,3],V[3:]))
```

Out[137]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x2ad769f614d0>
```



Question 8e: Visually examine the magnitude of values present in each of the four spectral decomposition matrices and comment on which two of the four matrices have elements with relatively small magnitude in them. Provide a reason for this based on your obsevation in Question 8c.

Answer: 2,3 attributes have lower spectral decomposition values and from the heat maps we can say the more lines with the color orange i.e value zero present in 2 and 3 attributes.

9. Linear Discriminant Analysis

We will use digits data for studying the use of LDA.

In [184]:

```
digits = load_digits()
```

The data with 1797 samples and 64 attributes is in the object digits.data. These 64 attributes represent pixels in an 8x8 image.

In [185]:

```
print(digits.data.shape)
```

```
(1797, 64)
```

The 1797 images are digits from 0...9. This information is in the digits.target variable.

In [186]:

```
print(digits.target)
```

```
[0 1 2 ... 8 9 8]
```

For this part, we will only focus on digits 3 and 8. To this end, we generate indices of 183 samples with 3s and indices of 174 samples with 8s.

In [187]:

```
Threes = np.where(digits.target==3)
Eights = np.where(digits.target==8)
[np.size(Threes), np.size(Eights)]
```

Out[187]:

```
[183, 174]
```

We will take samples from these indices and construct a matrix X such that the first 183 samples represent 3s and the remaining ones represent 8s. The variable y captures this information.

In [188]:

```
indices = np.hstack((Threes[0], Eights[0]));
X = digits.data[indices,:]
y = np.hstack((3*np.ones(np.size(Threes)), 8*np.ones(np.size(Eights))))
```

In [189]:

```
X
```

Out[189]:

```
array([[ 0.,  0.,  7., ...,  9.,  0.,  0.],
       [ 0.,  2.,  9., ..., 11.,  0.,  0.],
       [ 0.,  1.,  8., ...,  2.,  0.,  0.],
       ...,
       [ 0.,  0.,  5., ...,  3.,  0.,  0.],
       [ 0.,  0.,  1., ...,  6.,  0.,  0.],
       [ 0.,  0., 10., ..., 12.,  1.,  0.]])
```

In [190]:

```
X.shape
```

Out[190]:

```
(357, 64)
```

In [191]:

```
y
```

Out[191]:

```
array([3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3.,
       3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 3., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
       8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8., 8.,
8.])
```

In [192]:

```
y.shape
```

Out[192]:

```
(357,)
```

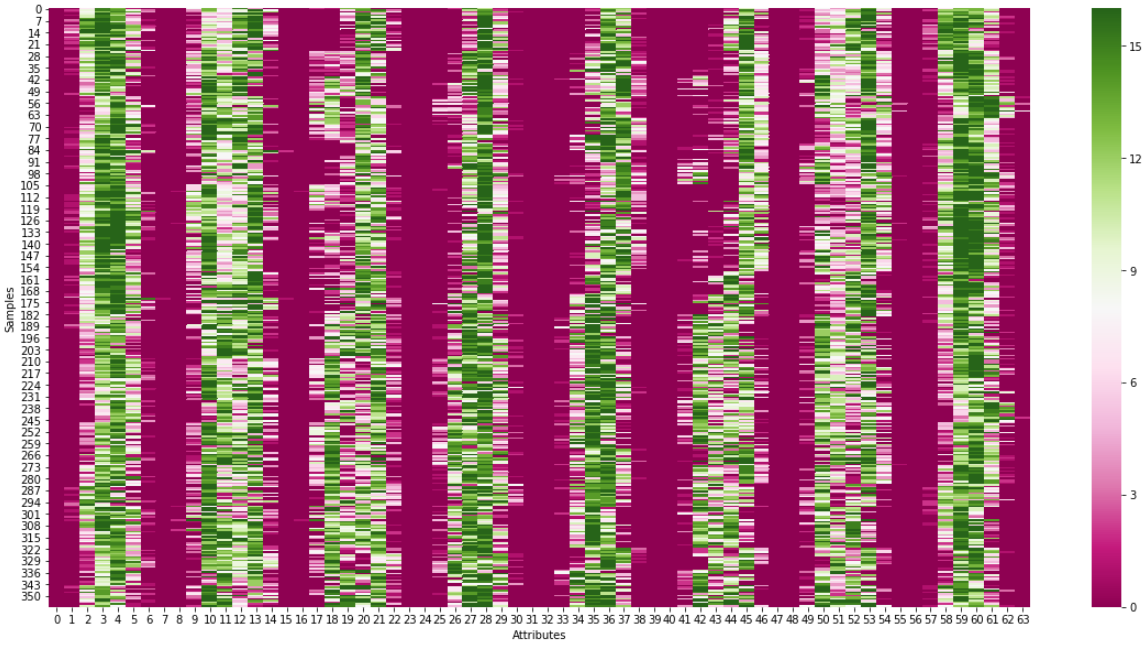
Question 9a: Visually examine the following heatmap of the data X and identify one attribute that can separate the 3s from 8s. Also comment on (approximately) how many mistakes would be committed if this attribute is used for projection in LDA.

In [193]:

```
plt.figure(figsize=(20,10))
ax = sns.heatmap(X,cmap='PiYG')
ax.set(xlabel='Attributes', ylabel='Samples')
```

Out[193]:

```
[Text(159,0.5,'Samples'), Text(0.5,69,'Attributes')]
```



Answer:42 is the attribute used to sepearate 8's and 3's becuase they were more pinks on the sample less than 183 and later with white which represent 8's.

Question 9b: Perform LDA on this data. Plot the heatmap of the projected data and comment how many points will be wrongly predicted based on this projection.

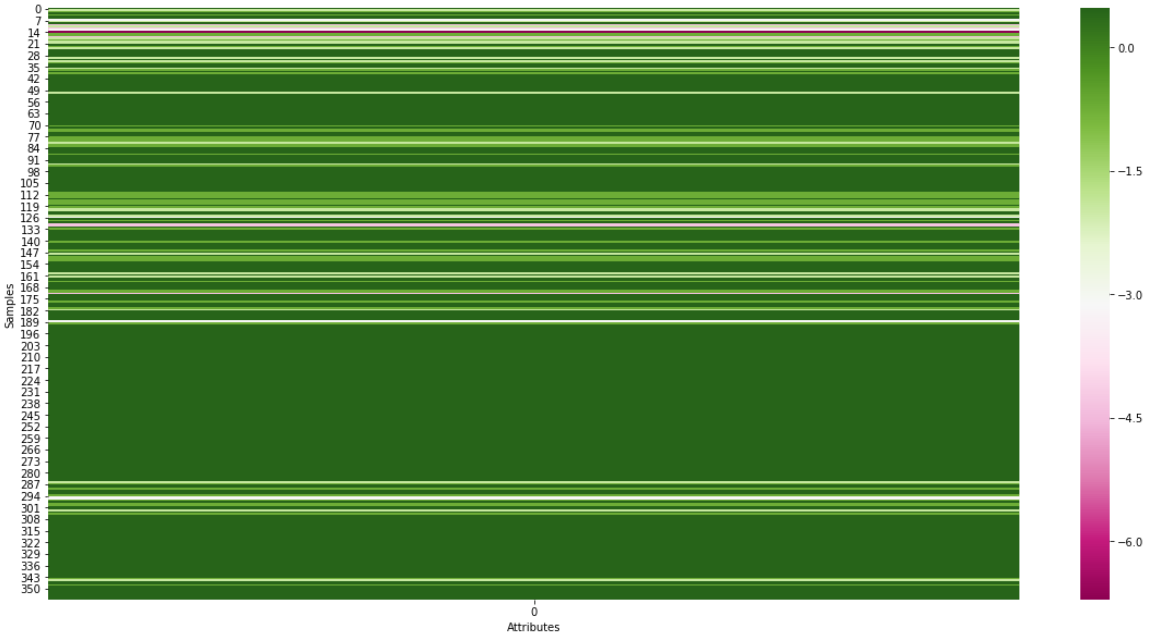
In [198]:

```
lda=LinearDiscriminantAnalysis(n_components=2)
X_r1 = lda.fit(X[:,0:2], y).transform(X[:,0:2])
plt.figure(figsize=(20,10))
ax = sns.heatmap(X_r1,cmap='PiYG')
ax.set(xlabel='Attributes', ylabel='Samples')
```

```
/usr/local/python/2.7-conda5.2/lib/python2.7/site-packages/sklearn/discrim
inant_analysis.py:388: UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
```

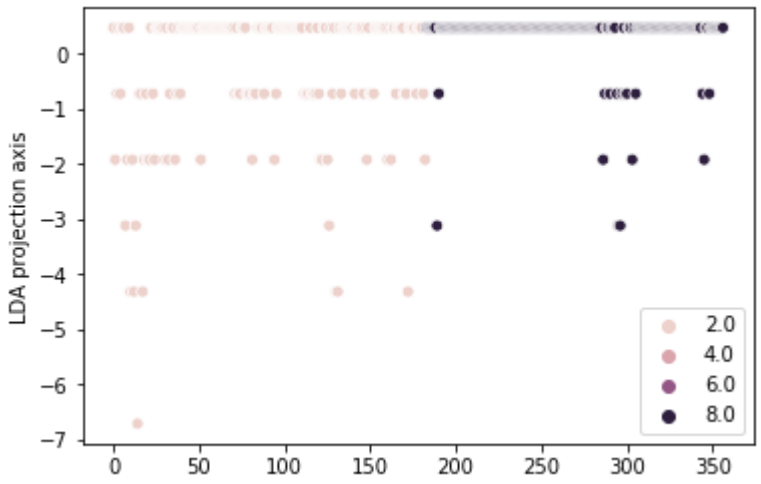
Out[198]:

```
[Text(159,0.5,'Samples'), Text(0.5,69,'Attributes')]
```



In [195]:

```
fig = sns.scatterplot(x=np.arange(np.size(X_r1)),y=X_r1[:,0],hue=y)
plt.ylabel('LDA projection axis')
plt.show(fig)
```



Answer: