Hands-on Practice for Module 1: Exploratory Data Analysis

0.Importing important packages

In [1]:
```
# 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.

Question 1a: What does the data capture?

Answer: Data captures 5 properties (Sex, Height, Weight, Reported Height, Reported Weight) of 200 persons who engaged in regular excercise

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

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

Question 1c: How many data points are in this dataset?
Answer: Initially 200. After dropping missing values we've 181 values.

In [2]:
```
davis_df = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/carData/Davis.csv')
```

In [3]:
```
davis_df.dropna(inplace=True);
```

In [4]:
```
davis_df.dtypes
```

Out[4]:
```
Unnamed: 0      int64
sex            object
weight          int64
height          int64
repwt         float64
repht         float64
dtype: object
```

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

Answer: 6 including the first unnamed column (row_no or unique ID)

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

Answer: 1 Nominal categorical attribute, 2 discrete numerical attributes and 2 continuos numerical attribute

2. Generating summary statistics

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

In [5]:
```
davis_df.drop(columns=davis_df.columns[davis_df.columns.str.contains('unnamed'
, case=False)], inplace=True)
davis_df.head()
```

Out[5]:

	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 [6]:
```
davis_df.describe(exclude = object).T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
weight	181.0	66.303867	15.340992	39.0	56.0	63.0	75.0	166.0
height	181.0	170.154696	12.312069	57.0	164.0	169.0	178.0	197.0
repwt	181.0	65.679558	13.834220	41.0	55.0	63.0	74.0	124.0
repht	181.0	168.657459	9.394668	148.0	161.0	168.0	175.0	200.0

Answer: Weigth: 39 - 166; Range of Weight = 166 - 39 = 127 Height: 57 - 197; Range of Height = 197 - 57 = 140 RepWeight: 41.0 - 124.0; Range of RepWeight = 124 - 41 = 83 RepHeight: 148 - 200; Range of RepHeight = 200 - 148 = 52

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

In [7]:
```
davis_df.describe(include = object)
```

Out[7]:

	sex
count	181
unique	2
top	F
freq	99

Answer: 2 unique values Male and Female

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

In [8]:
```
from pandas.api.types import is_numeric_dtype

for col in davis_df.columns:
    if is_numeric_dtype(davis_df[col]):
        print('%s:' % (col))
        print('\t Mean = %.2f' % davis_df[col].mean())
```

```
weight:
         Mean = 66.30
height:
         Mean = 170.15
repwt:
         Mean = 65.68
repht:
         Mean = 168.66
```

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

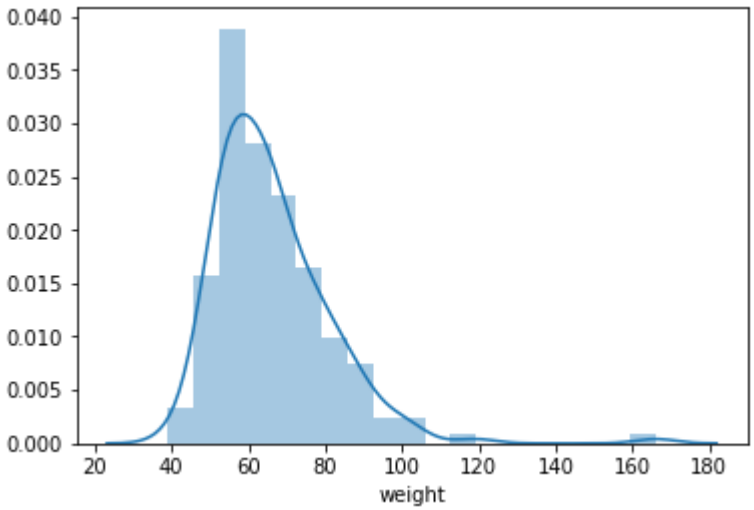
In [9]:
```
from pandas.api.types import is_numeric_dtype

for  col in davis_df:
    if is_numeric_dtype(davis_df[col]):
        print('%s:' % (col))
        print('\t Variance = %.2f' % davis_df[col].var())
```

```
weight:
         Variance = 235.35
height:
         Variance = 151.59
repwt:
         Variance = 191.39
repht:
         Variance = 88.26
```

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

In [10]:
```
sns.distplot(davis_df['weight'])
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27f92e5a160>
```

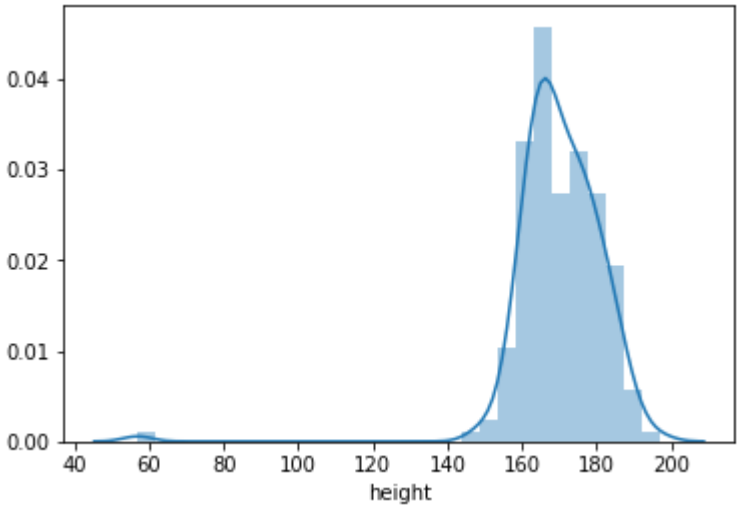


Answer: Yes it's normally distributed

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

In [11]:
```
sns.distplot(davis_df['height'])
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27f9319d240>
```

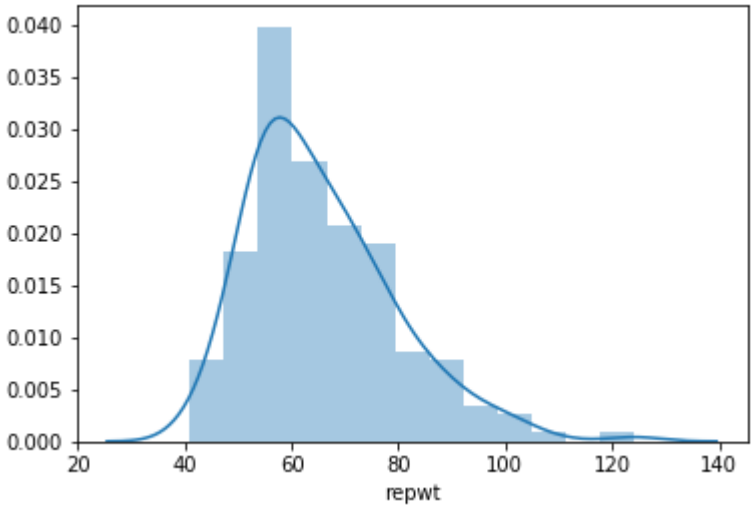


Answer: Yes it's normally distributed.

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

In [12]:
```
sns.distplot(davis_df['repwt'])
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27f931d62e8>
```

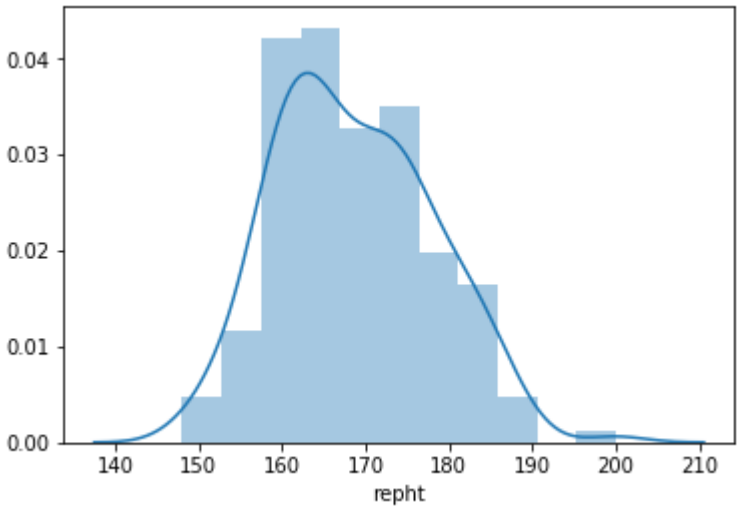


Answer: Yes it's normally distributed

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

In [13]:
```
sns.distplot(davis_df['repht'])
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27f932d6908>
```

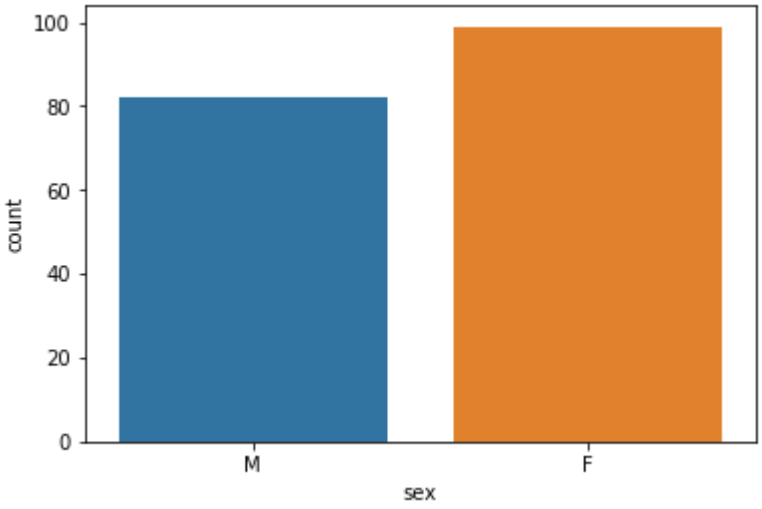


Answer:No its not a normal distribution. It has a protuberance between 170 - 180 apart from one dominant peak

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

In [14]:
```
sns.countplot(davis_df['sex'])
```

Out[14]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27f933539b0>
```



Answer: No it's not uniformly distributed

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 [15]:
```
davis_df_new = davis_df[['repwt','repht']]
```

In [16]:
```
davis_df_new.head()
```

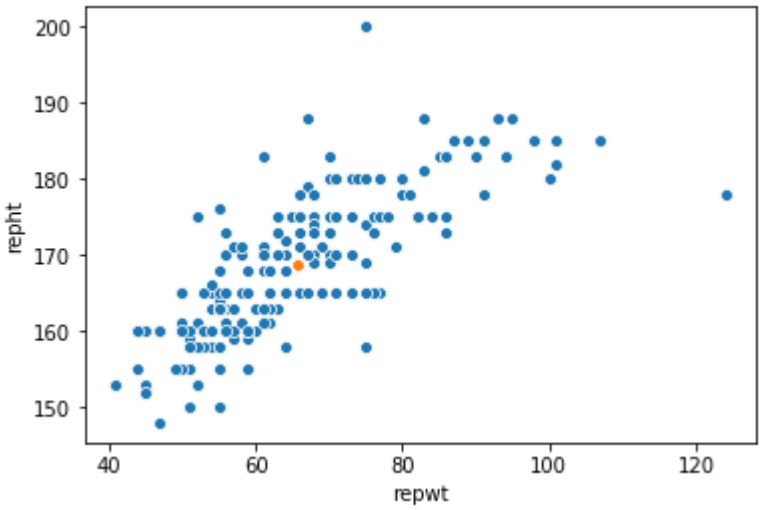
Out[16]:

	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 davis_df_new data on a 2D space along with the mean.

In [17]:
```
fig, ax = plt.subplots()
sns.scatterplot(x='repwt', y='repht', data=davis_df_new, ax=ax)
mu = np.mean(davis_df_new.values, 0)
sns.scatterplot(x=[mu[0], mu[0]], y=[mu[1], mu[1]])
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27f933b7240>
```



In [ ]:
```
```

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 [18]:
```
from sklearn.preprocessing import normalize
davis_df_new_row_norm = normalize(davis_df_new, axis=1, norm='l2')
```

In [19]:
```
davis_df_new_row_norm[1:10,:]
```

Out[19]:
```
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]])
```

In [20]:
```
davis_df_new_row_norm_df = pd.DataFrame(davis_df_new_row_norm, columns=["repwt", "repht"])
type(davis_df_new_row_norm_df)
davis_df_new_row_norm.shape
davis_df_new_row_norm_df.shape
davis_df_new_row_norm_df.head()
```

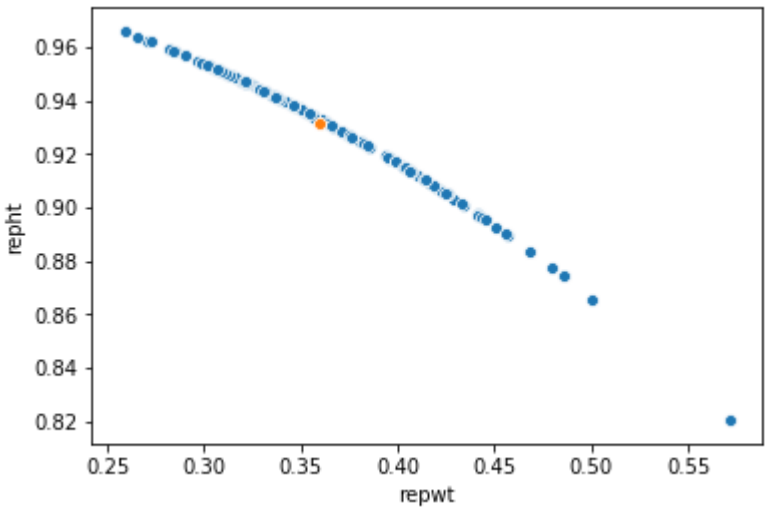
Out[20]:

	repwt	repht
0	0.393303	0.919409
1	0.305428	0.952215
2	0.323405	0.946260
3	0.371391	0.928477
4	0.355745	0.934583

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 [21]:
```
fig, ay = plt.subplots()
sns.scatterplot(x='repwt', y='repht', data=davis_df_new_row_norm_df, ax = ay)
norm_mean = np.mean(davis_df_new_row_norm_df.values, 0)
sns.scatterplot(x=[norm_mean[0], norm_mean[0]], y = [norm_mean[1], norm_mean[1]])
```

Out[21]: <matplotlib.axes._subplots.AxesSubplot at 0x27f93403e80>



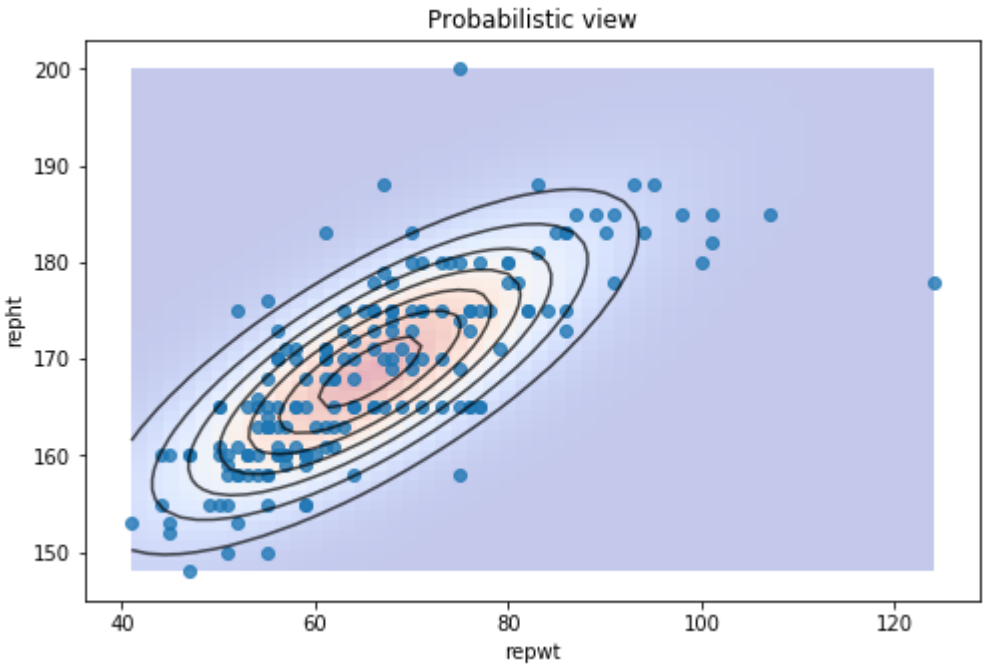
Answer: Geometric view of the normalized data shows that the 2 attributes have a negative linear relationship whereas the Geometric view of the original data suggested a postive linear relationship. Also Normalized data are like a line close to each other whereas the original data is spread out more. Main reason for the difference in graphs was data_norm was normalized based on row thus changing the relationship in the original data

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

In [22]:
```
from scipy.stats import multivariate_normal

mu = np.mean(davis_df_new.values,0)
Sigma = np.cov(davis_df_new.values.transpose())

min_length = np.min(davis_df_new.values[:,0]);
min_width = np.min(davis_df_new.values[:,1]);
max_length = np.max(davis_df_new.values[:,0]);
max_width = np.max(davis_df_new.values[:,1]);
x, y = np.mgrid[min_length:max_length:50j, min_width:max_width:50j]

positions = np.empty(x.shape + (2,))
positions[:, :, 0] = x;
positions[:, :, 1] = y

F = multivariate_normal(mu, Sigma)
Z = F.pdf(positions)
```

In [23]:
```
fig = plt.figure(figsize=(8,8))
ax = fig.gca()
ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_length,max_length, min_wid
th,max_width], 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[23]: Text(0.5, 1.0, '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 [24]:
```
davis_df_new_col_norm = normalize(davis_df_new, axis=0, norm='l2')
```

In [25]:
```
davis_df_new_col_norm[1:10,:]
```

Out[25]:
```
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 [26]:
```
davis_df_new_col_norm_df = pd.DataFrame(davis_df_new_col_norm, columns=["repwt", "repht"])
```

In [27]:
```
type(davis_df_new_col_norm_df)
davis_df_new_col_norm_df.shape
davis_df_new_col_norm_df.head()
```

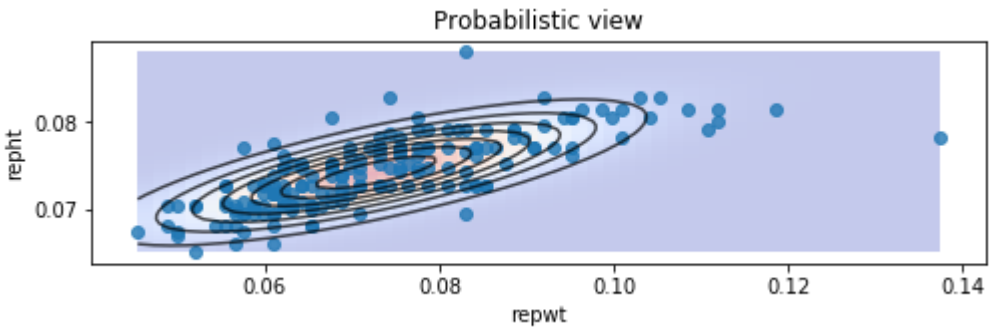
Out[27]:

	repwt	repht
0	0.085280	0.079206
1	0.056484	0.069965
2	0.059807	0.069525
3	0.077527	0.077006
4	0.065344	0.068205

In [28]:
```
from scipy.stats import multivariate_normal

mu = np.mean(davis_df_new_col_norm_df.values,0)
Sigma = np.cov(davis_df_new_col_norm_df.values.transpose())

min_length = np.min(davis_df_new_col_norm_df.values[:,0]);
min_width = np.min(davis_df_new_col_norm_df.values[:,1]);
max_length = np.max(davis_df_new_col_norm_df.values[:,0]);
max_width = np.max(davis_df_new_col_norm_df.values[:,1]);
x, y = np.mgrid[min_length:max_length:50j, min_width:max_width:50j]

positions = np.empty(x.shape + (2,))
positions[:, :, 0] = x;
positions[:, :, 1] = y

F = multivariate_normal(mu, Sigma)
Z = F.pdf(positions)
```

In [29]:
```
fig = plt.figure(figsize=(8,8))
ax = fig.gca()
ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_length,max_length, min_wid
th,max_width], alpha=0.3)
cset = ax.contour(x, y, Z, colors='k', alpha=0.7)
plt.scatter(davis_df_new_col_norm_df.values[:,0],davis_df_new_col_norm_df.valu
es[:,1],alpha=0.8)
ax.set_xlabel('repwt')
ax.set_ylabel('repht')
plt.title('Probabilistic view')
```

Out[29]: Text(0.5, 1.0, 'Probabilistic view')



Answer: The actual shape of the the covariance structure doesn't change. It is represented on a more shrinked y-axis. Column normalisation has not affected the shape of covariance structure

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 [30]:
```
davis_df.cov()
```

Out[30]:

	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: None

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

In [31]:
```
davis_df.corr()
```

Out[31]:

	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: Highly correlated pairs listed in decreasing order of correlation.
weight and repwt - 0.835
repwt and repht - 0.761
height and repht - 0.739

Question 4d: Which pairs of attributes are uncorrelated?

Answer: Lowly correlated pairs listed in increasing order of correlation.
weight and height - 0.154

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

Answer: From correlation matrix We understood to what degrees(i.e magnitude) the attributes are related to each other but from covariance matrix we cannot say anything about the magnitude, It'll only tell if attributes are positively or negatively correlated

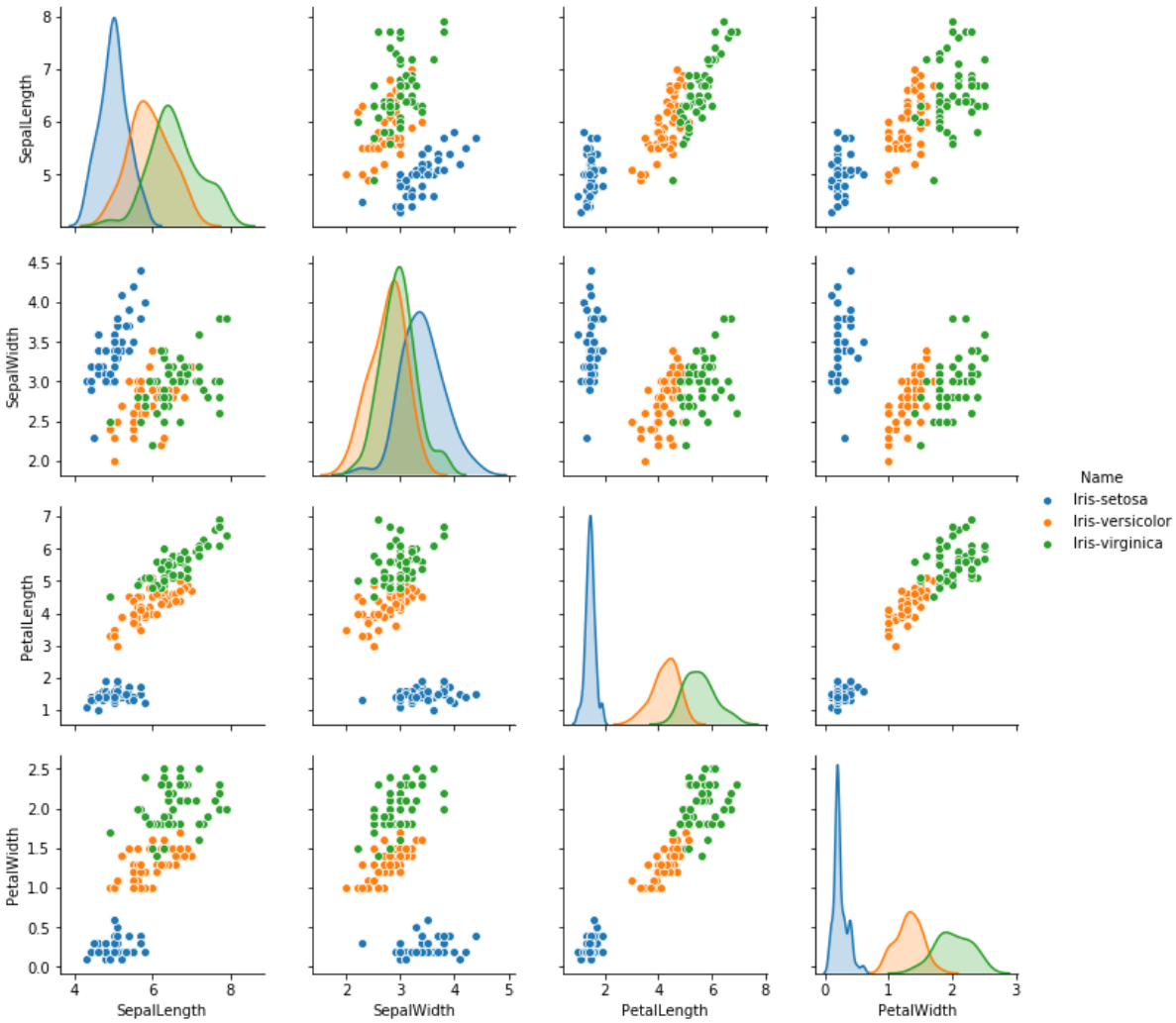
5. Dimensionality Reduction: Feature Selection

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 [32]:
```
import seaborn as sns
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/iris.csv')
sns.pairplot(iris_df, hue='Name')
```

Out[32]: <seaborn.axisgrid.PairGrid at 0x27f939480f0>



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.

Answer: Petal length, as the overlapping area between classes is least in this attribute

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: Petal Length and Petal Width, from the pairplot this has minimum number of overlaps and has the least variance within the cluster

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: We can use the correlation matrix to find out the attributes which are strongly related to the target variable and then classify based on those attributes

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: We can use feature construction to create r new intrinsic dimensions from original d dimensions. we can use this r new dimensions to do analysis and generally r is very small compared to d, but sometimes it's very large and if r is very large we've to deal with it. PCA and SVD are examples of feature construction

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'.

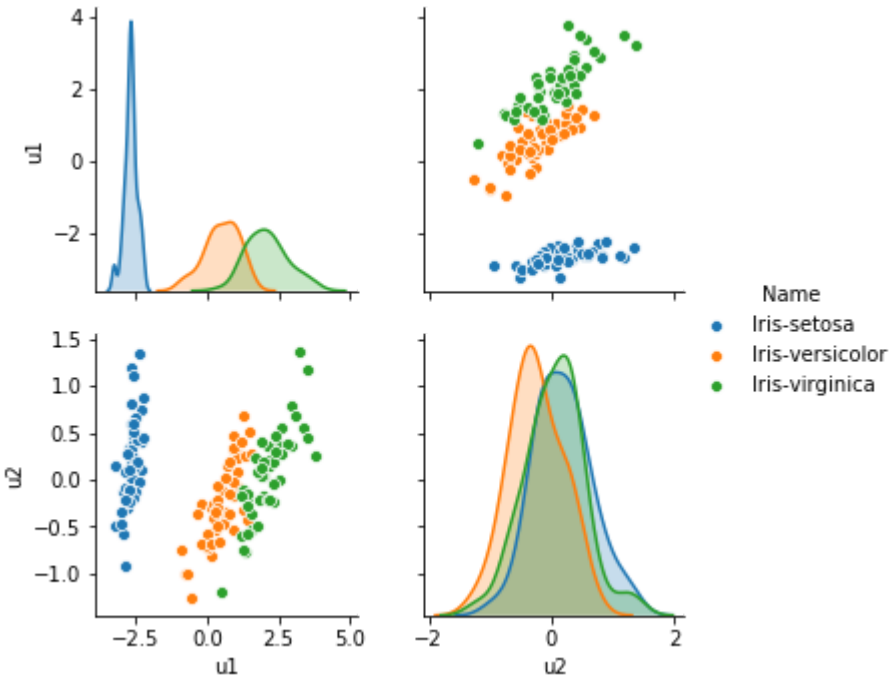
Hint: Use iris_df[['SepalLength','SepalWidth','PetalLength','PetalWidth']] to use the specified attributes.

In [77]:
```
from sklearn.decomposition import PCA
pca = PCA(2)  # project from 64 to 2 dimensions
projected = pca.fit_transform(iris_df[['SepalLength','SepalWidth','PetalLengt
h','PetalWidth']])
projected_df = pd.DataFrame(projected, columns=["u1", "u2"])
projected_df["Name"] = iris_df['Name']
```

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 [78]:
```
sns.pairplot(projected_df, hue='Name')
```

Out[78]: <seaborn.axisgrid.PairGrid at 0x27f99477c50>



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: u1, as there's a evident gap in u1's distribution. No. Assuming the question was about 5a u1 is same as the attribute selected 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: No. Assuming the question is about 5b The Intra-cluster variance here is bit high compared to the 5b

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.

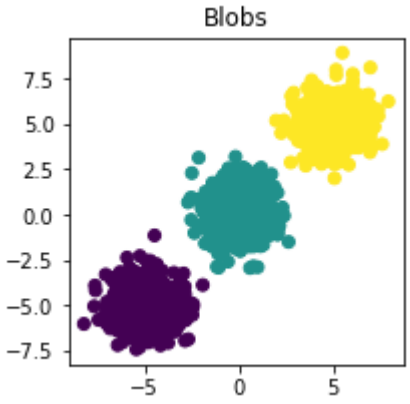
In [35]:
```
n_samples = 1500
random_state = 42
centers = [(-5, -5), (0, 0), (5, 5)]
Blobs_X, Blobs_y = datasets.make_blobs(n_samples=n_samples,centers=centers,ran
dom_state=random_state)
```

In [36]:
```
Blobs_X.shape
Blobs_X
```

Out[36]:
```
array([[ 0.16846098,  1.31759754],
       [-3.53435123, -5.2257763 ],
       [-6.52552517, -5.69190807],
       ...,
       [-1.24386324, -0.6929052 ],
       [-0.20902326, -0.85052045],
       [ 4.8053027 ,  4.93581182]])
```

In [37]:
```
plt.figure(figsize=(3,3))
plt.scatter(Blobs_X[:, 0], Blobs_X[:, 1], c= Blobs_y)
plt.title('Blobs')
```

Out[37]:
```
Text(0.5, 1.0, '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 [38]:
```
Blobs1= pd.DataFrame(Blobs_X)
Blobs1['2'] = np.random.randn(1500)
Blobs1.head()
```

Out[38]:

	0	1	2
0	0.168461	1.317598	-0.787584
1	-3.534351	-5.225776	-0.791907
2	-6.525525	-5.691908	-0.482421
3	-0.120948	0.419532	-0.103534
4	-5.469474	-4.457440	-1.224326

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 [39]:
```
Blobs2= pd.DataFrame(Blobs_X)
Blobs2['2'] = np.random.randn(1500)*10
Blobs2.head()
```

Out[39]:

	0	1	2
0	0.168461	1.317598	-22.384322
1	-3.534351	-5.225776	-20.861058
2	-6.525525	-5.691908	-10.318450
3	-0.120948	0.419532	2.367574
4	-5.469474	-4.457440	-3.551410

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 [40]:
```
Blobs3= pd.DataFrame(Blobs_X)
Blobs3['2'] = np.random.randn(1500)*10
Blobs3['3'] = np.random.randn(1500)*10
Blobs3.head()
Blobs1.head()
```

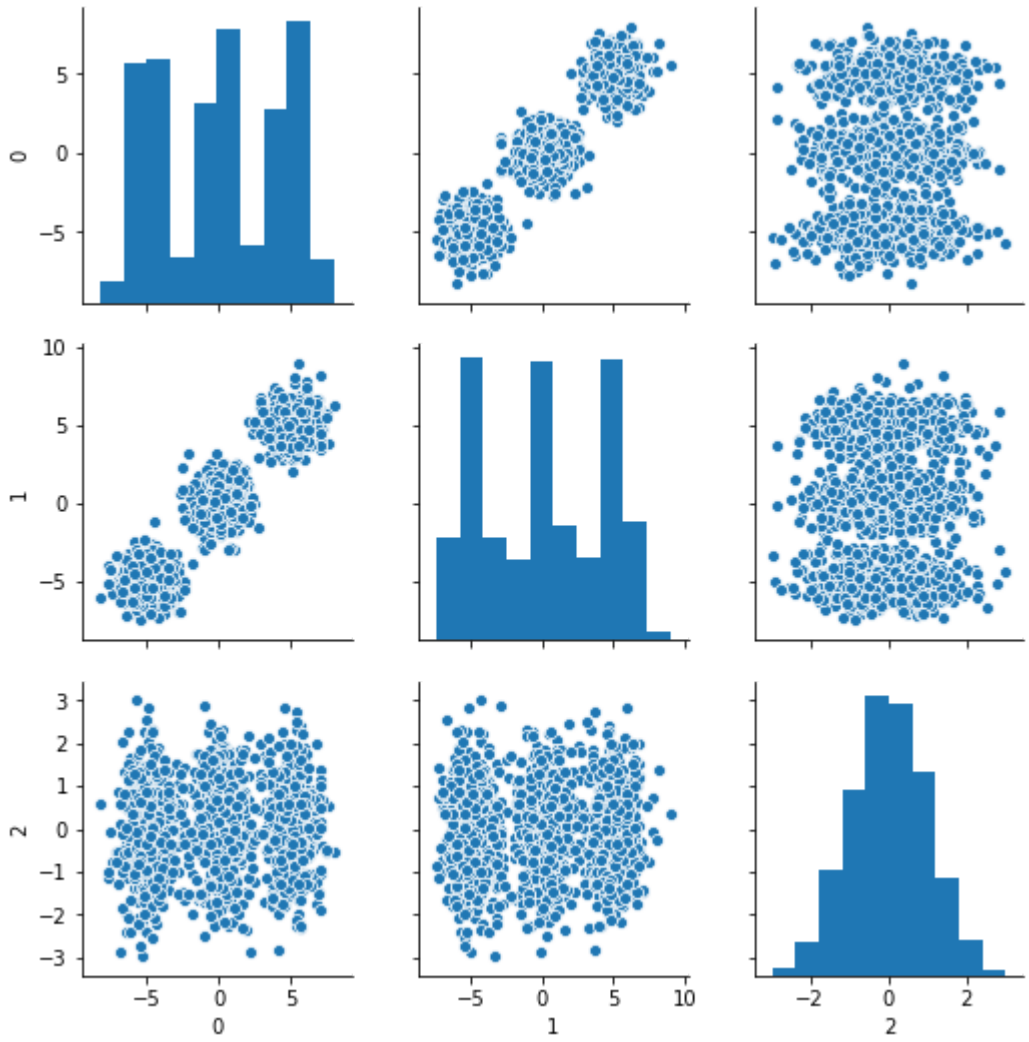
Out[40]:

	0	1	2
0	0.168461	1.317598	-0.787584
1	-3.534351	-5.225776	-0.791907
2	-6.525525	-5.691908	-0.482421
3	-0.120948	0.419532	-0.103534
4	-5.469474	-4.457440	-1.224326

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 [41]:
```
sns.pairplot(Blobs1)
```

Out[41]: <seaborn.axisgrid.PairGrid at 0x27f95d79a58>



Answer: Seeing the graph we can say that attribute 3 is normally distributed and has low variance compared to other 2 attributes

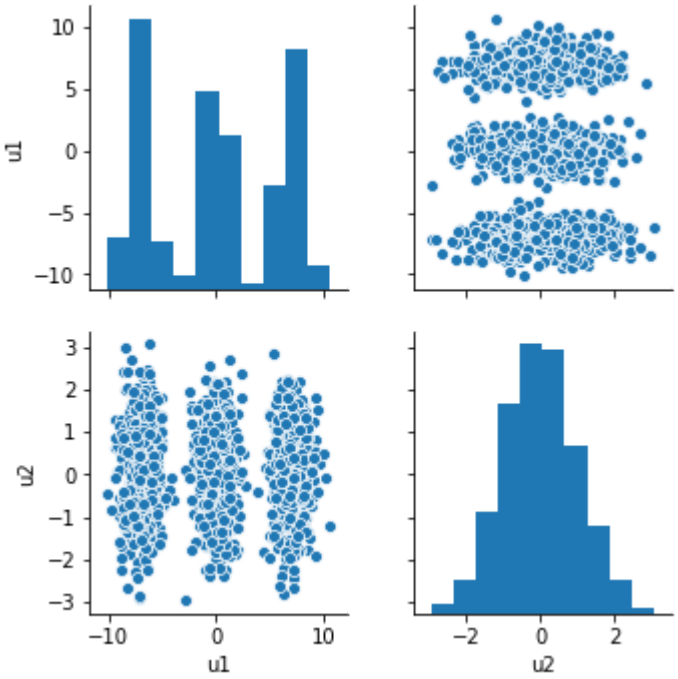
Question 7b: Perform PCA on **Blobs1** data. Project data onto the first two principal components. Generate a pairplot for the newly constructed attributes.

In [42]:
```
pca = PCA(2)  # project from 64 to 2 dimensions
projected_blobs1 = pca.fit_transform(Blobs1)
Blobs1_df = pd.DataFrame(projected_blobs1, columns=["u1", "u2"])
Blobs1_df.head()
Blobs1.var()
```

Out[42]:
```
0    17.570371
1    17.643028
2     0.988038
dtype: float64
```

In [43]:
```
sns.pairplot(Blobs1_df)
```

Out[43]: <seaborn.axisgrid.PairGrid at 0x27f962594e0>



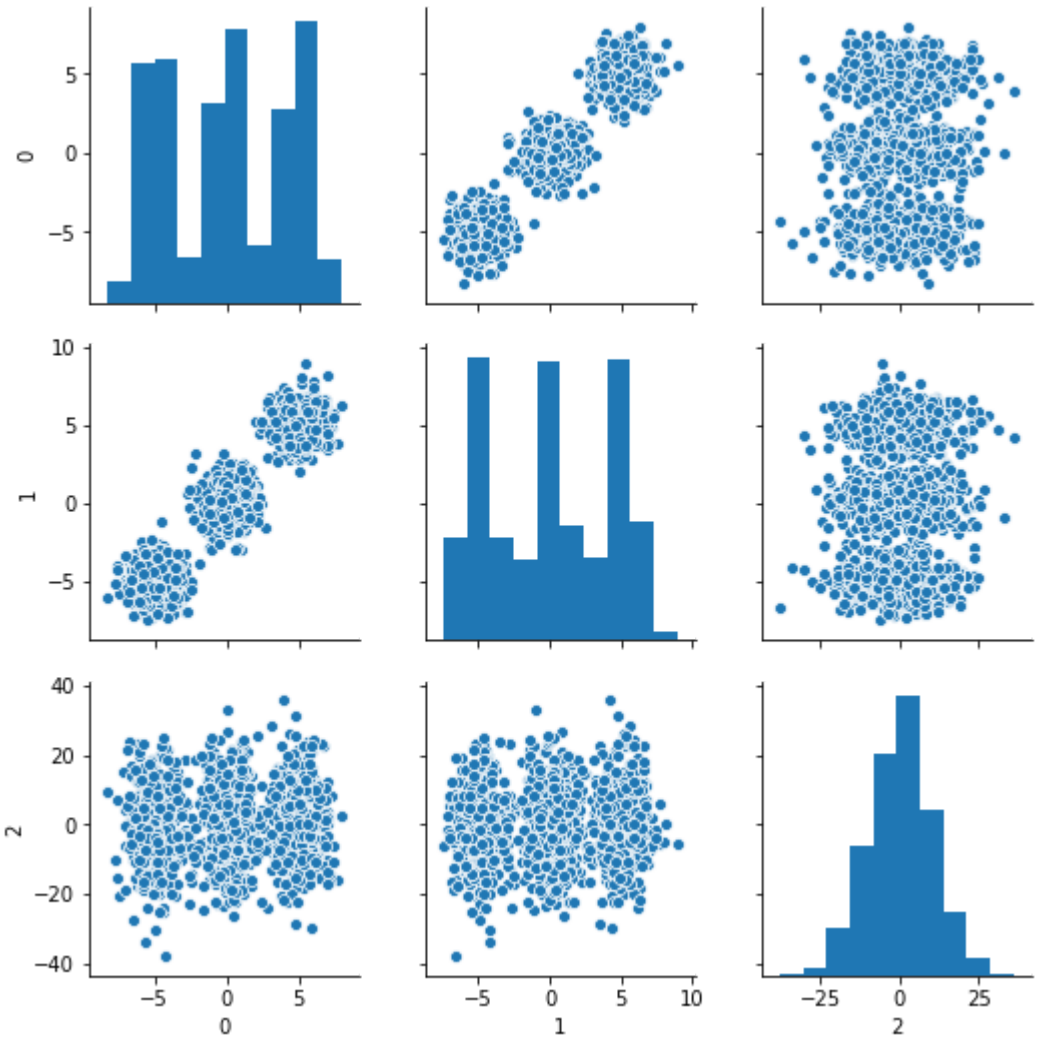
Question 7c: By comparing the distributions for the newly generated attributes in Question 7b with the previous pairplot in Question 7a, determine which attribute is captured by the first principal component and which attribute is captured by the second principal component. Provide a reason for your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

Answer:After comparing the distributions 7b and 7c we can say that u1 captures 2nd attribute '1' which also covers for variance across the 1st attribute 1 as both these data are linearly correlated and u2 captures the 3rd attribute as it's distribution is almost exactly the same of 3rd atribute '2' in blobs1 and it has also next most variance to be captured

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 [44]:
```
sns.pairplot(Blobs2)
```

Out[44]:
```
<seaborn.axisgrid.PairGrid at 0x27f969492b0>
```



Answer: Since the 3rd attribute is normally distributed and from the histogram we can say that variance is very high for the 3rd attribute in comparison to the first two attributes.

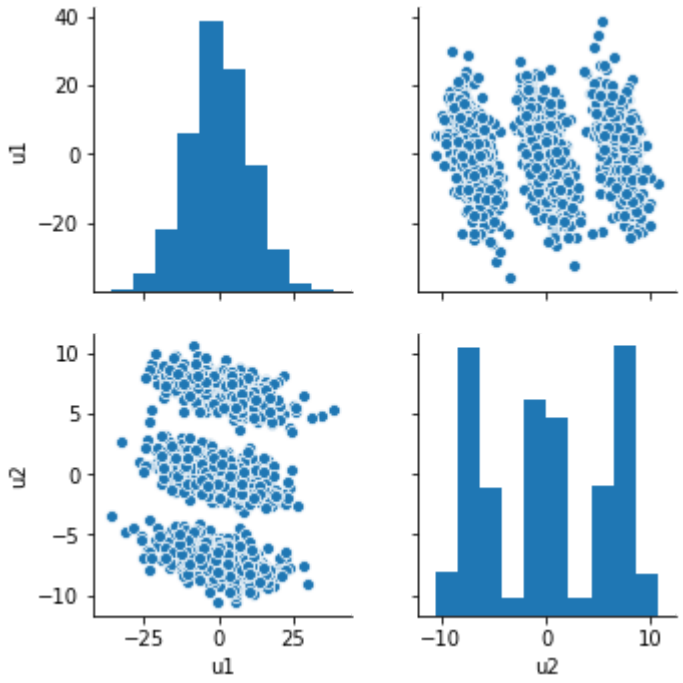
Question 7e: Perform PCA on **Blobs2** data. Project data onto the first two principal components. Generate a pairplot for the newly constructed attributes.

In [45]:
```
pca = PCA(2)  # project from 64 to 2 dimensions
projected_blobs2 = pca.fit_transform(Blobs2)
Blobs2_df = pd.DataFrame(projected_blobs2, columns=["u1", "u2"])
Blobs2_df.head()
Blobs2.var()
```

Out[45]:
```
0     17.570371
1     17.643028
2    102.916900
dtype: float64
```

In [46]:
```
sns.pairplot(Blobs2_df)
```

Out[46]: <seaborn.axisgrid.PairGrid at 0x27f96e39cf8>



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: u1 captures the 3rd attribute '2' as it has the most variance and u2 captures variance across attribute '1', which by the way also accounts for variance across attribute '0' as both these attributes are linearly correlated. We can see that '2' attribute for Blobs2 is captured by u1, whereas '2' attribute is captured by u2 for Blobs1, as we know u1 captures most variance it is likely that '2' had most variance in Blobs2 data and least variance in Blobs1 data. And we can confirm this by looking in to their respective variance

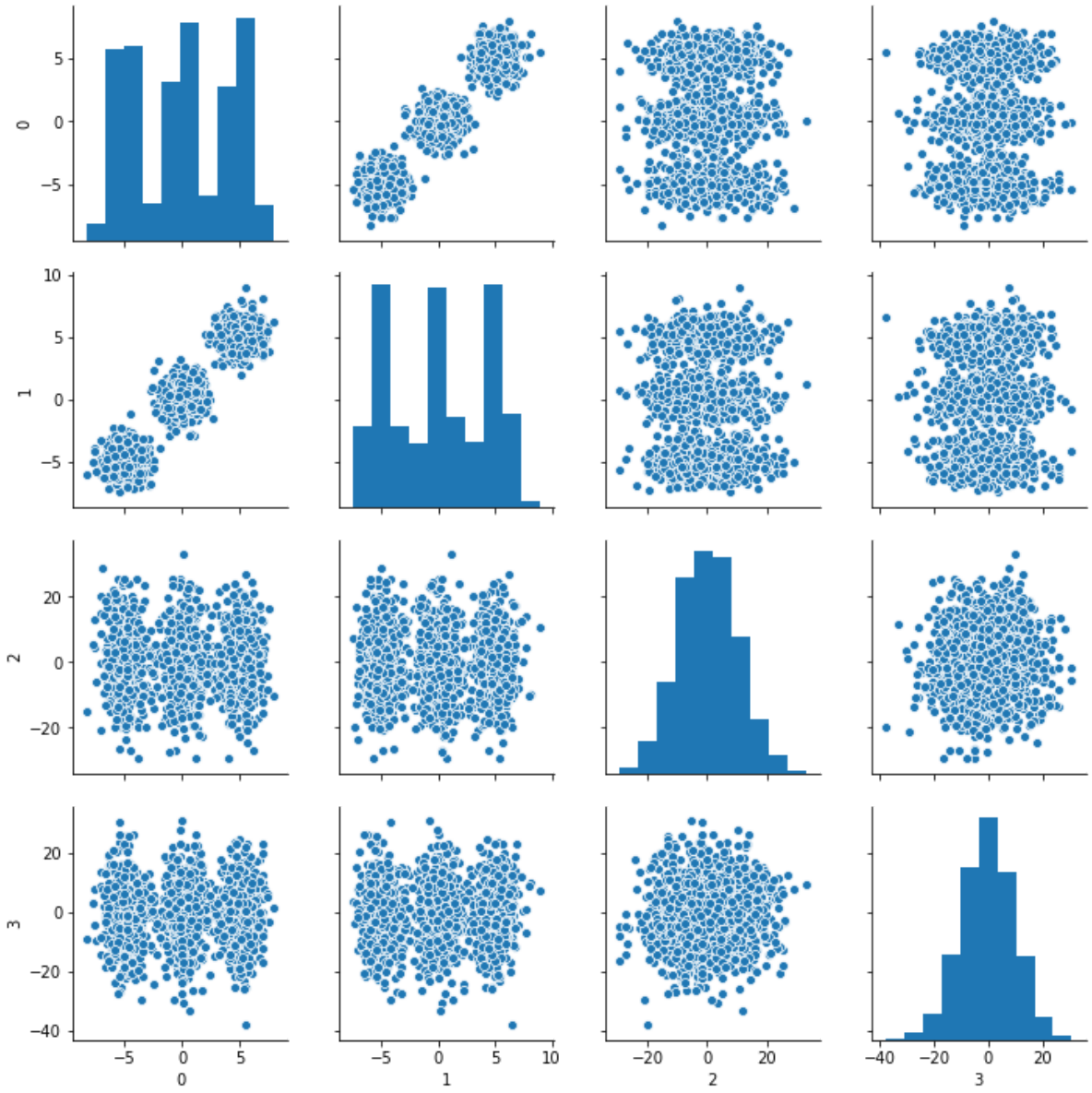
Question 7g: Are the three blobs separately visible after projection based on PCA in Question 7e?

Answer: Yes they're visibile

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 [47]:
```
sns.pairplot(Blobs3)
```

Out[47]: <seaborn.axisgrid.PairGrid at 0x27f97004080>



Answer: The first two attributes are positively linearly correlated, we can say that the second two attributes are not linearly correlated

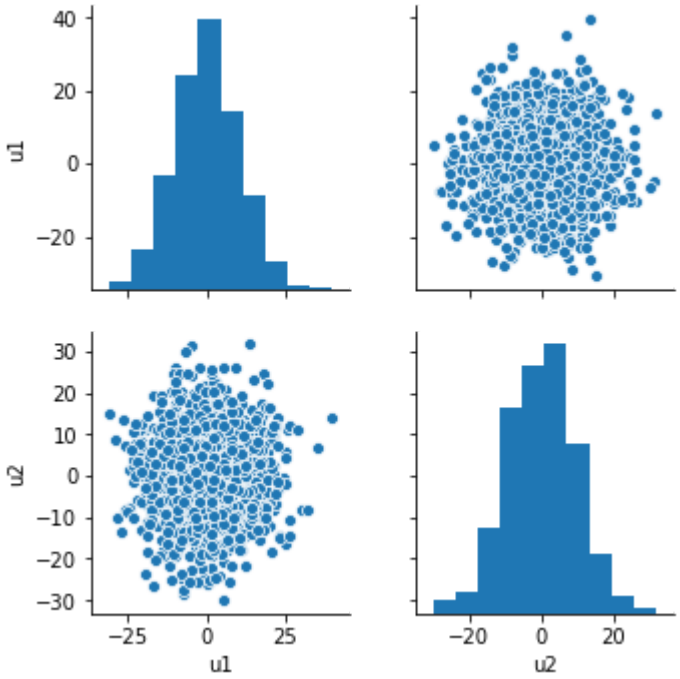
Question 7i: Perform PCA on **Blobs3** data. Project data onto the first two principal components. Generate a pairplot for the newly constructed attributes.

In [48]:
```
pca = PCA(2)  # project from 64 to 2 dimensions
projected_blobs3 = pca.fit_transform(Blobs3)
Blobs3_df = pd.DataFrame(projected_blobs3, columns=["u1", "u2"])
Blobs3_df.head()
Blobs3.var()
```

Out[48]:
```
0    17.570371
1    17.643028
2    97.900050
3    97.019199
dtype: float64
```

In [49]:
```
sns.pairplot(Blobs3_df)
```

Out[49]:
```
<seaborn.axisgrid.PairGrid at 0x27f98839ba8>
```



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: u1 captures the 4th attribute and u2 captures the 3rd attribute. Since the 4th and 3rd attribute have most variances respectively PCA captures these two in the above mentioned order.

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: No the 3 Blobs are not visible, as the new attributes added to the dataset have the most variances thus clouding the original variances from being projected

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

Answer: We've found out from above examples that if we add new data (i.e noise) with large variances , it clouds the original variances thus limiting effects of PCA

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 [106]:
```
def mysvd(A):

    dim_cov = np.cov(A.transpose())
    S, V = np.linalg.eig(dim_cov)


    T, U = np.linalg.eig(np.matmul(A, A.transpose()))

    return U, S, V
```

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 [118]:
```
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 [119]:
```
U,S,V = mysvd(A)
J = np.matmul(np.matmul(U[:, 0:6], np.diag(S)), V.transpose())
J
```

Out[119]:
```
array([[ 1.25323267,  1.15253219,  1.25323267, -1.22694757, -1.22694757,
        -0.51536042],
       [ 3.75969802,  3.45759658,  3.75969802, -3.68084272, -3.68084272,
        -1.54608127],
       [ 5.0129307 ,  4.61012877,  5.0129307 , -4.90779029, -4.90779029,
        -2.0614417 ],
       [ 6.26616337,  5.76266097,  6.26616337, -6.13473786, -6.13473786,
        -2.57680212],
       [ 1.26910173,  1.2655327 ,  1.26910173,  0.23610482,  0.23610482,
        -0.01183   ],
       [ 1.0624283 ,  1.13494275,  1.0624283 ,  0.85076645,  0.85076645,
         0.0078816 ],
       [ 0.41717309,  0.58112037,  0.41717309,  0.39843716,  0.39843716,
        -0.16925207]])
```

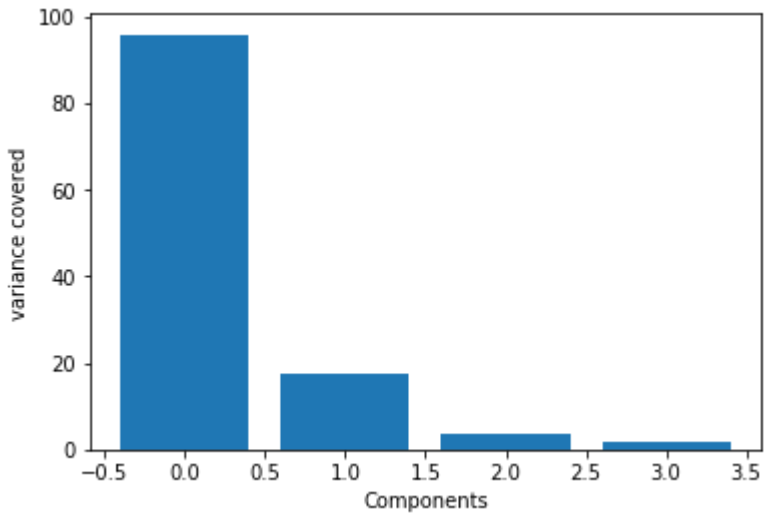
Question 8c: Perform SVD on iris dataset and visualize the proportion of variance captured by each spectral value. List the dimensions that captures less than 10% of the total variance.

In [54]:
```
import pandas as pd
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/iris.csv')
```

In [55]:
```
data = iris_df.values[:,0:4]
data = data.astype(float) #converts data format from object to numeric
```

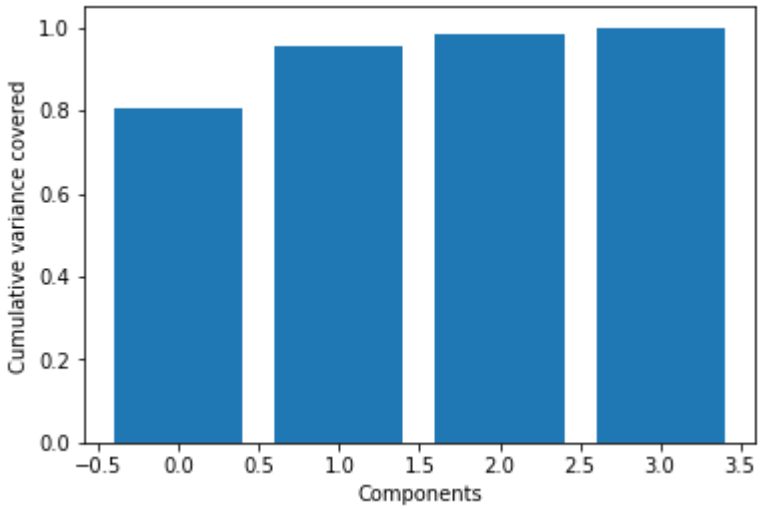
In [56]:
```
U,S,V = svd(data)
plt.bar(np.arange(4),S)
plt.xlabel('Components')
plt.ylabel('variance covered')
```

Out[56]:
```
Text(0, 0.5, 'variance covered')
```



In [57]:
```
plt.bar(np.arange(4),np.cumsum(S)/np.sum(S))
plt.xlabel('Components')
plt.ylabel('Cumulative variance covered')
```

Out[57]:
```
Text(0, 0.5, 'Cumulative variance covered')
```

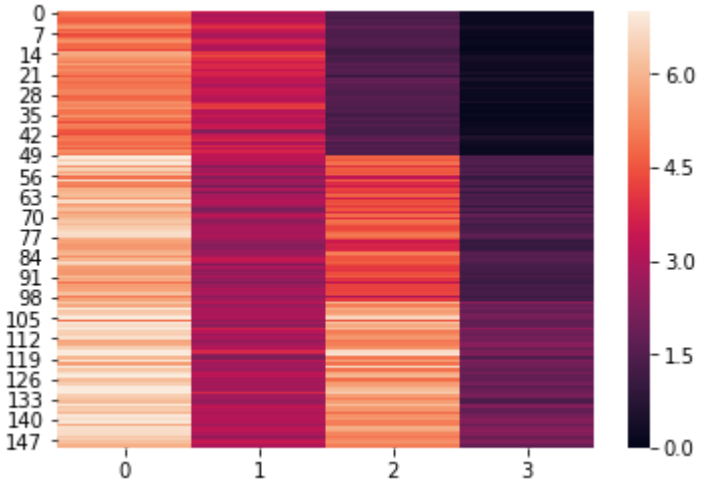


Answer: The last 2 dimmensions capture less than 10% variance

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

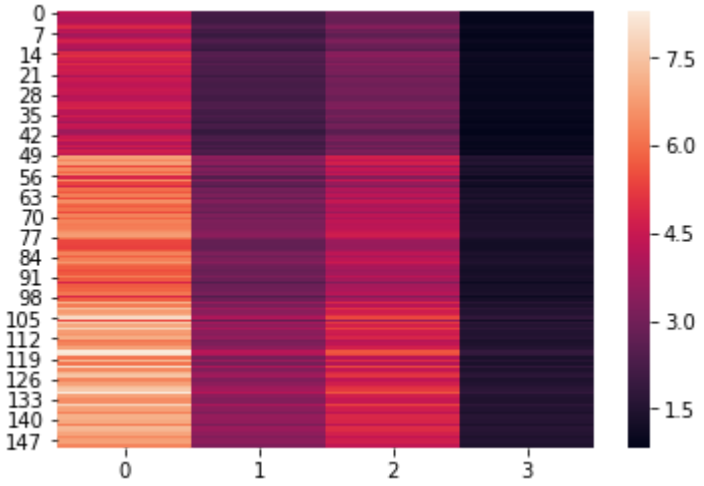
In [58]:
```
sns.heatmap(data,vmin=0, vmax=7)
```

Out[58]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27f98dbd080>
```



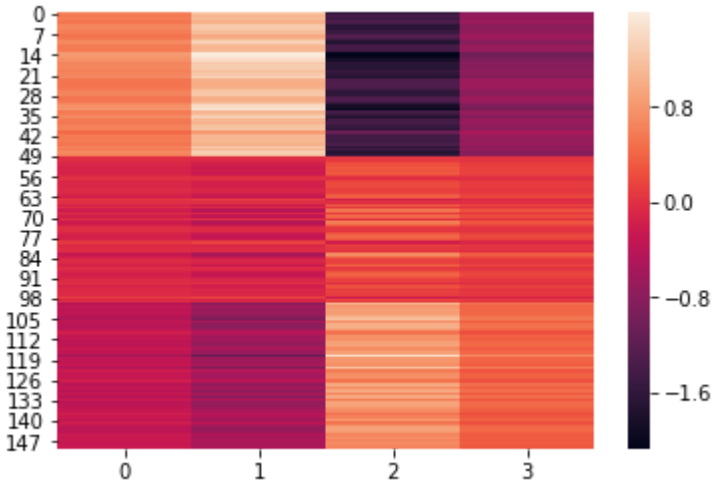
In [59]:
```
sns.heatmap(S[0]*np.outer(U[:,0],V[0,:]))
```

Out[59]: <matplotlib.axes._subplots.AxesSubplot at 0x27f98ea8128>



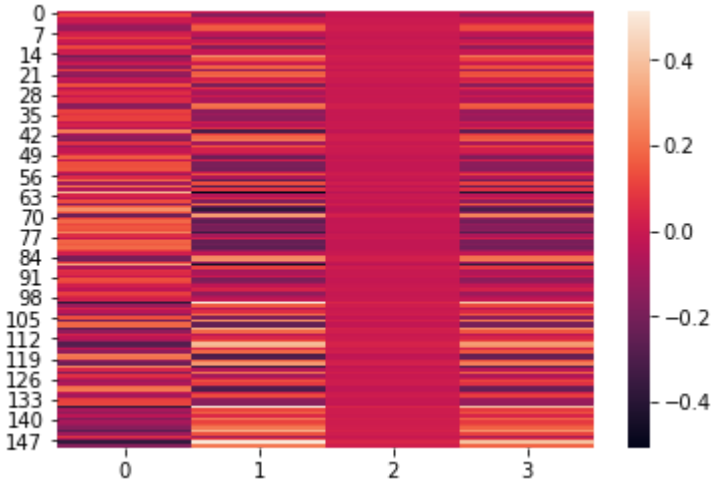
In [60]:
```
sns.heatmap(S[1]*np.outer(U[:,1],V[1,:]))
```

Out[60]: <matplotlib.axes._subplots.AxesSubplot at 0x27f98f6a2b0>



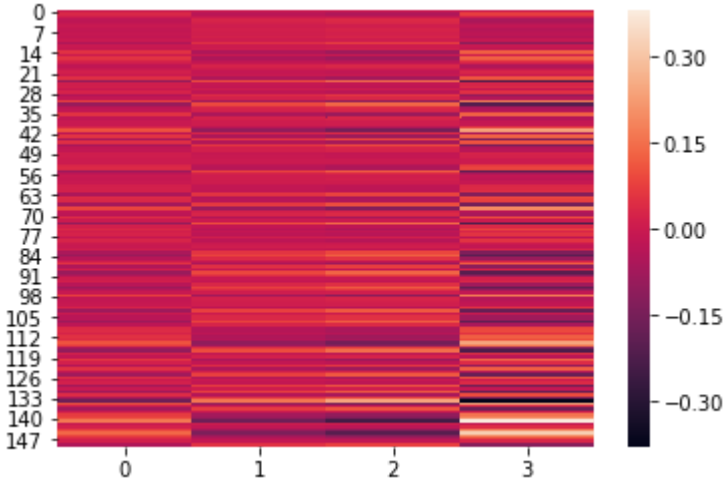
In [61]:
```
sns.heatmap(S[2]*np.outer(U[:,2],V[2,:]))
```

Out[61]: <matplotlib.axes._subplots.AxesSubplot at 0x27f98f2f438>



In [62]:
```
sns.heatmap(S[3]*np.outer(U[:,3],V[3,:]))
```

Out[62]: <matplotlib.axes._subplots.AxesSubplot at 0x27f990e4128>



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: 3rd and the 4th element have least magnitude as we can see from heatmaps above it captures relatively small amount of magnitude compared to first 2

9. Linear Discriminant Analysis

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

In [63]:
```
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 [64]:
```
digits.data.shape
```

Out[64]:
```
(1797, 64)
```

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

In [65]:
```
digits.target
```

Out[65]:
```
array([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 [66]:
```
Threes = np.where(digits.target==3)
Eights = np.where(digits.target==8)
[np.size(Threes), np.size(Eights)]
```

Out[66]:
```
[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 [67]:
```
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 [68]:
```
X
```

Out[68]:
```
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 [69]:
```
X.shape
```

Out[69]:
```
(357, 64)
```

In [70]:
```
y
```

Out[70]:
```
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.])
```

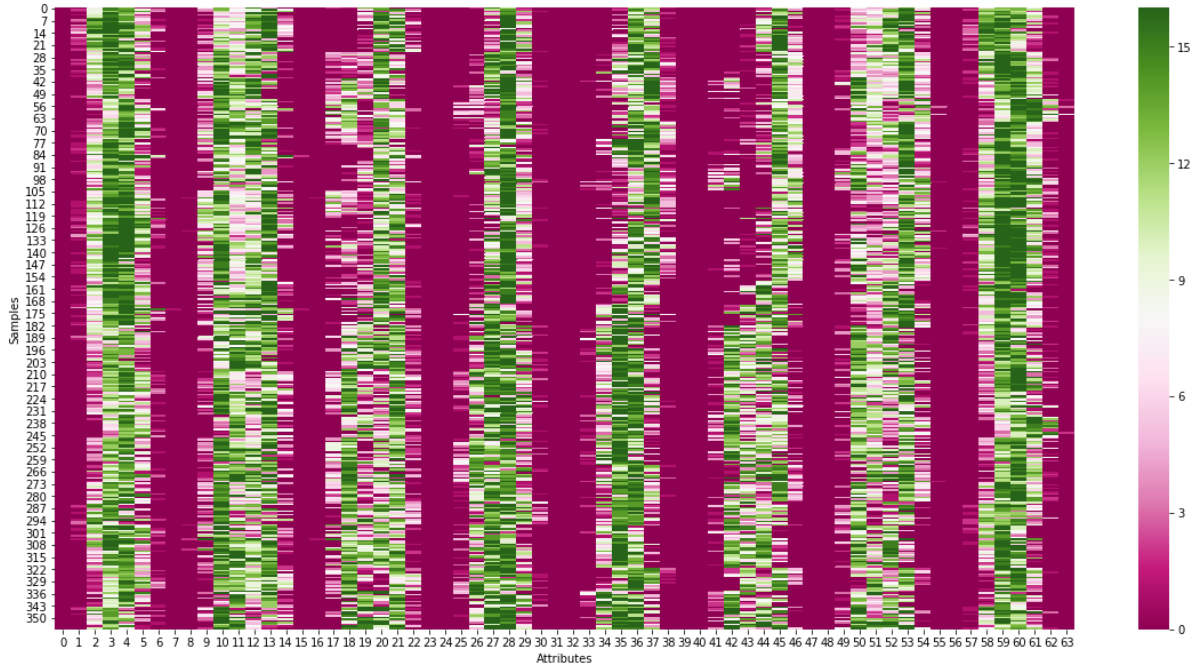
In [71]:
```
y.shape
```

Out[71]:
```
(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 [72]:
```
plt.figure(figsize=(20,10))
ax = sns.heatmap(X,cmap='PiYG')
ax.set(xlabel='Attributes', ylabel='Samples')
```

Out[72]:
```
[Text(159.0, 0.5, 'Samples'), Text(0.5, 69.0, 'Attributes')]
```



Answer: Attribute 42 can be used for seperation and looking from the heatmap it seems there'll be an approximate of 15 - 35 mistakes

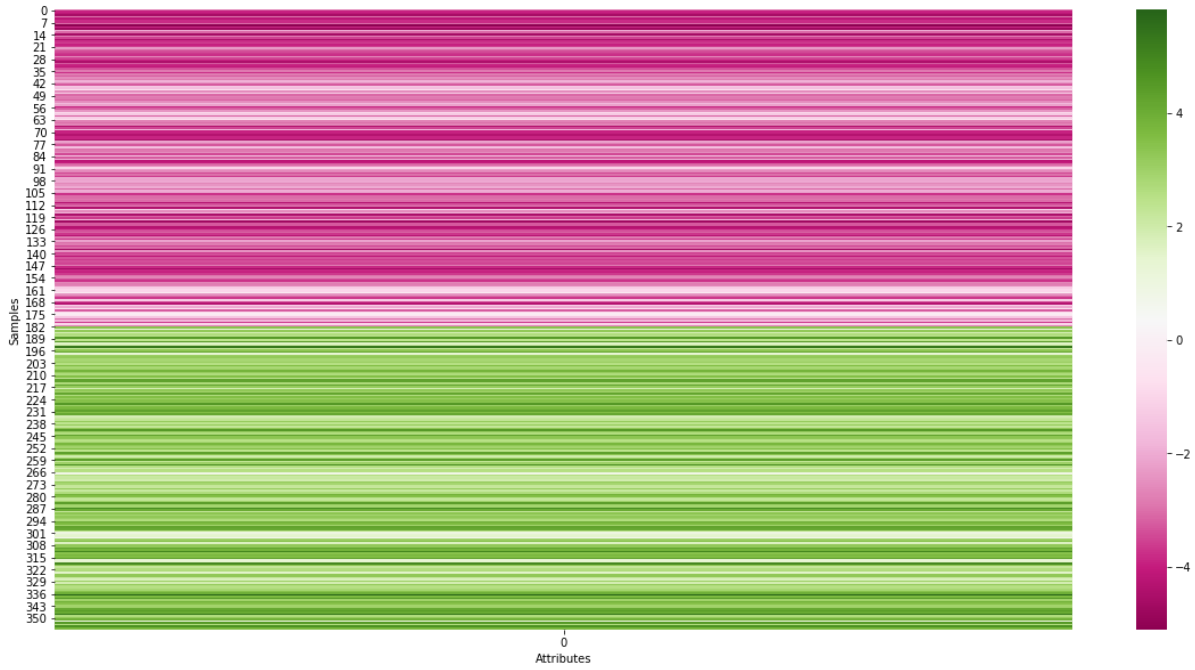
Question 9b: Perform LDA on this data. Plot the heatmap of the projected data and comment how many points will be wrongly predicted based on this projection.

In [73]:
```
lda = LinearDiscriminantAnalysis(n_components=2)
X_r1 = lda.fit(X, y).transform(X)
```

```
C:\Users\nifaullah\Anaconda3\lib\site-packages\sklearn\discriminant_analysis.
py:388: UserWarning: Variables are collinear.
  warnings.warn("Variables are collinear.")
```

In [74]:
```
plt.figure(figsize=(20,10))
ax = sns.heatmap(X_r1,cmap='PiYG')
ax.set(xlabel='Attributes', ylabel='Samples')
```

Out[74]: [Text(159.0, 0.5, 'Samples'), Text(0.5, 69.0, 'Attributes')]



Answer: Based on white lines in between approximately 20 to 40 points will be wrongly predicted based on the projection

In [ ]:
```
```