Hands-on Practice for Module 1: Exploratory Data Analysis

0.Importing important packages

In [5]:
```
# 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

from pandas.api.types import is_numeric_dtype
```

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

Description of the data is provided at 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: The data captures six properties (Unnamed: 0, sex, weight, height, reported weight, reported height) of men and women engaged in regular exercise

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

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

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

In [3]:
```
davis_df = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/csv/carData/Davis.csv')
```

In [4]:
```
davis_df.dropna(inplace=True);
```

In [5]:
```
davis_df.columns
```

Out[5]:
```
Index([u'Unnamed: 0', u'sex', u'weight', u'height', u'repwt', u'repht'], dtype='object')
```

In [6]:
```
davis_df.head()
```

Out[6]:

	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

In [7]:
```
davis_df.shape
```

Out[7]:
```
(181, 6)
```

Answer: 181 data points are there

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

Answer: There are 6 attributes

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

In [8]:
```
davis_df.dtypes
```

Out[8]:
```
Unnamed: 0      int64
sex            object
weight          int64
height          int64
repwt         float64
repht         float64
dtype: object
```

Answer:

The attribute Unnamed: 0 is discrete numeric attribute

The attribute Unnamed: 0 is discrete numeric attribute.
The attributes weight, height, reported weight and reported height are continous numeric attribute
The attribute sex is nominal categorical attribute.

2. Generating summary statistics

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

In [9]:
```
davis_df.drop(columns=davis_df.columns[davis_df.columns.str.contains('unnamed', case=False)], inplace=True)
davis_df.head()
```

Out[9]:

	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 [10]:
```
davis_df.describe(exclude=object)
```

Out[10]:

	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: Weight: 39.0 to 166.0
Height: 57.0 to 197.0
Reported Weight: 41.0 to 124.0
Reported Height: 148.0 to 200.0

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

In [11]:
```
davis_df.describe(include=object)
```

Out[11]:

	sex
count	181
unique	2
top	F
freq	99

In [13]:
```
davis_df.sex.astype('category').cat.categories
```

Out[13]:
```
Index([u'F', u'M'], dtype='object')
```

Answer: It takes two values M and F

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

In [14]:
```
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 [15]:
```
for col in davis_df.columns:
    if is_numeric_dtype(davis_df[col]):
        print('%s:' % (col))
        print('\t Variance = %.2f' % davis_df[col].var())
```

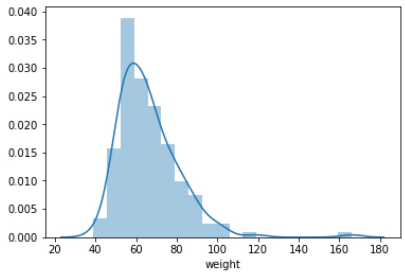
```
weight:
	 Variance = 235.35
height:
	 Variance = 151.59
repwt:
	 Variance = 191.39
repht:
	 Variance = 88.26
```

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

In [16]:
```
sns.distplot(davis_df['weight'])
```

```
/usr/local/python/2.7-conda5.2/lib/python2.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tupl
e sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future th
is will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different r
esult.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aaec9417b90>
```

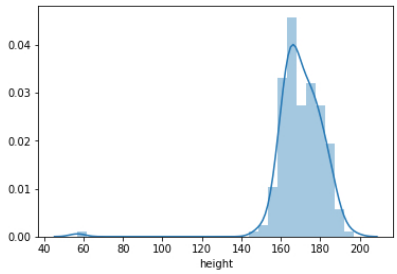


Answer: The weight attribute is normally distributed.

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

In [17]:
```
sns.distplot(davis_df['height'])
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aaec97c8690>
```

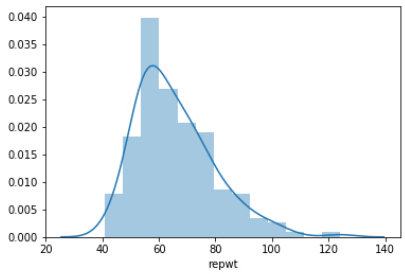


Answer: The height attribute is normally distributed.

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

In [18]:
```
sns.distplot(davis_df['repwt'])
```

Out[18]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aaec9417650>
```

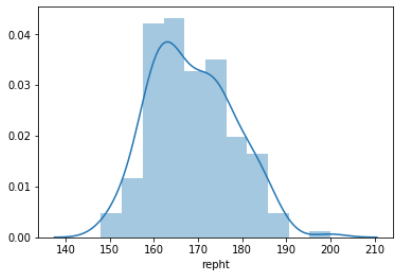


Answer: The repwt attribute is normally distributed.

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

In [19]:
```
sns.distplot(davis_df['repht'])
```

Out[19]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aaec98eb910>
```

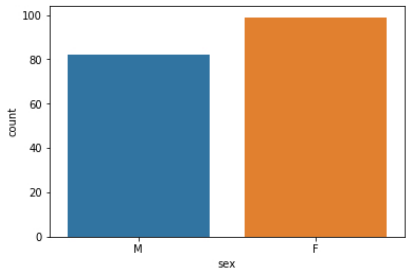


Answer: It is not normally distributed. it appears to be bimodal.

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

In [20]:
```
sns.countplot(davis_df['sex'])
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2aaec998e810>
```



Answer: It is not uniform distributed. The count of 'F' is higher than 'M'.

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 [13]:
```
davis_df_new = davis_df[['repwt','repht']]
```

In [14]:
```
davis_df_new.head()
```

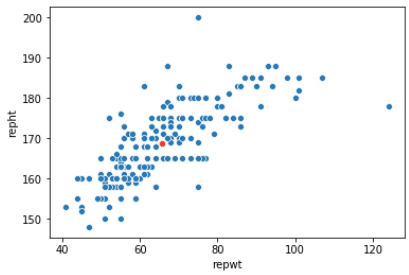
Out[14]:

	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 [15]:
```
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]],color='r',ax=ax)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2ba55fea4dd0>
```



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 [16]:
```
from sklearn.preprocessing import normalize
davis_df_new_row_norm = normalize(davis_df_new, axis=1, norm='l2')
```

In [17]:
```
type(davis_df_new_row_norm)
```

Out[17]:
```
numpy.ndarray
```

In [18]:
```
davis_df_new_row_norm.shape
```

Out[18]:
```
(181, 2)
```

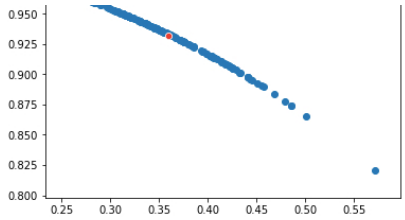
In [19]:
```
davis_df_new_row_norm[1:10,:]
```

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]])
```

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 [20]:
```
fig, ax = plt.subplots()
plt.scatter(x=davis_df_new_row_norm[:,0],y=davis_df_new_row_norm[:, 1])
mu = np.mean(davis_df_new_row_norm,0)
sns.scatterplot(x=[mu[0], mu[0]],y=[mu[1], mu[1]],color='r',ax=ax)
```

Out[20]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2ba5600cddd0>
```



In [21]:
```
mu
```

Out[21]:
```
array([0.35932096, 0.93158092])
```

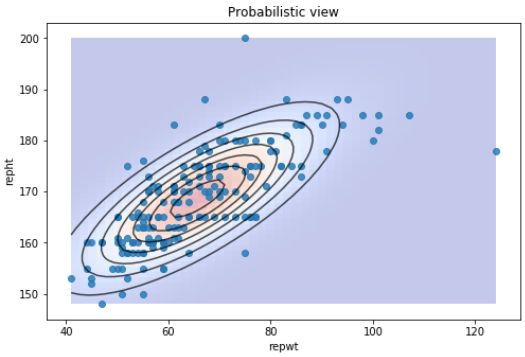
Answer: The ranges of reported weight and reported height are very different. To resolve this we did normalization of the rows of data. The data points in this view form a curved line which is different from the graph in 3a where the instances are scattered. As the instances are normalized in 3b the data points are not scattered and are close to each other indicating that there are not much differences between the instances and they are close to mean. Also from this recent plot, we can infer about covariance and correlation.

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_width,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,'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 [32]:
```
davis_df_new_col_norm = normalize(davis_df_new, axis=0, norm='l2')
```

In [33]:
```
davis_df_new_col_norm[1:10,:]
```

Out[33]:
```
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]])
```

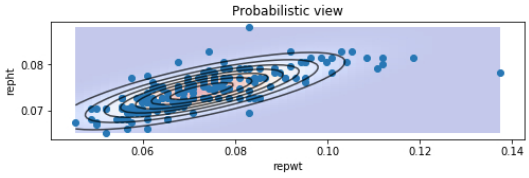
Question 3d: Show the Probabilistic view of the data davis_df_new_col_norm. Compare the shape of the covariance structure in the Gaussian distribution with that of Question 3c and comment if column normalization has affected the shape of the covariance structure.

In [34]:
```
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_length = np.min(davis_df_new_col_norm[:,0]);
min_width = np.min(davis_df_new_col_norm[:,1]);
max_length = np.max(davis_df_new_col_norm[:,0]);
max_width = np.max(davis_df_new_col_norm[:,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 [35]:
```
fig = plt.figure(figsize=(8,8))
ax = fig.gca()
ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_length,max_length, min_width,max_width], 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])
ax.set_xlabel('repwt')
ax.set_ylabel('repht')
plt.title('Probabilistic view')
```

Out[35]:
```
Text(0.5,1,'Probabilistic view')
```



Answer: The column normalization has brought the points more closer to each other. Yes it has affected. This is a normalized covariance structure representing correlation between the two attributes.

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 [24]:
```
davis_df.head()
```

Out[24]:

	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

In [25]:
```
davis_df.cov()
```

Out[25]:

	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 of the attributes covary in opposite direction

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

In [45]:
```
davis_df.corr()
```

Out[45]:

	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: Correlated pairs in decreasing order:
weight, repwt 0.835
repwt, repht 0.7618
height, repht 0.739
repht, weight 0.6314
repwt, height 0.603
height, weight 0.154

Question 4d: Which pairs of attributes are uncorrelated?

Answer: weight and height are least correlated with a value of 0.154

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

Answer: Correlation matrix is normalized due to which we get a proper understanding of strength between the attributes and we can compare this strength between the pairs. But with covariance matrix we cannot compare the strength between the pairs as it is not normalized.
We cannot infer strength between attributes in covariance matrix but we can infer the same in correlation matrix. We can only infer the trend between two attributes in covariance matrix in linear way.

5. Dimensionality Reduction: Feature Selection

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

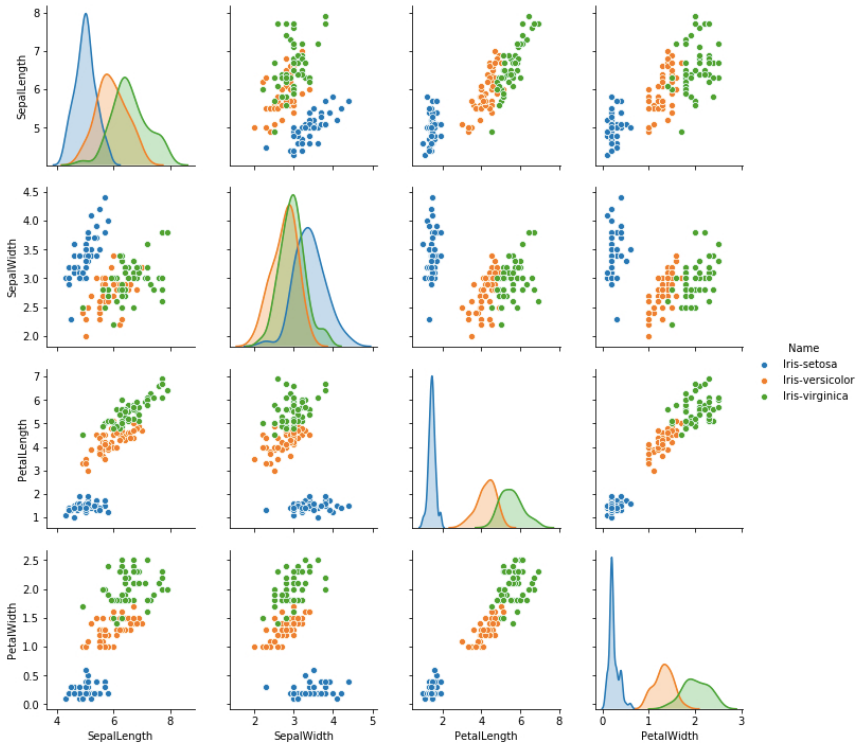
Assumption: Assume that your goal is to cluster the data to identify the species 'Name'. Clustering algorithm takes as input data points and attributes. It groups points that are similar to each other into a separate cluster. It puts points that are dissimilar in different cluster. Note that the 'Name'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

In [27]:
```
import seaborn as sns
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/iris.csv')
```

In [28]:
```
sns.pairplot(iris_df, hue='Name')
```

```
/usr/local/python/2.7-conda5.2/lib/python2.7/site-packages/scipy/stats/stats.py:1713: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```

Out[28]: <seaborn.axisgrid.PairGrid at 0x2ba55f821490>



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: The petal length can be used for clustering task as there is separation seen between the classes in the above pair plot. Petal Width can also be considered for the clustering task. Other attributes are not helping to that extent to seperate the classes.

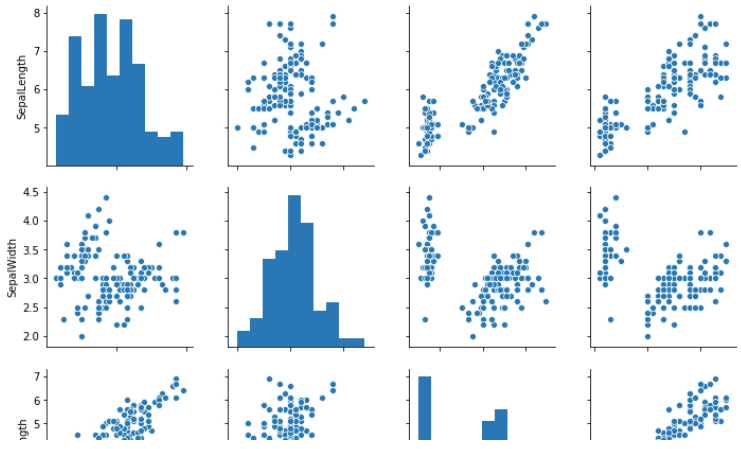
Question 5b: If you are allowed to select only two features, which feature would be highly useful for the clustering task. Provide a reason. Use pairplot to answer this question.

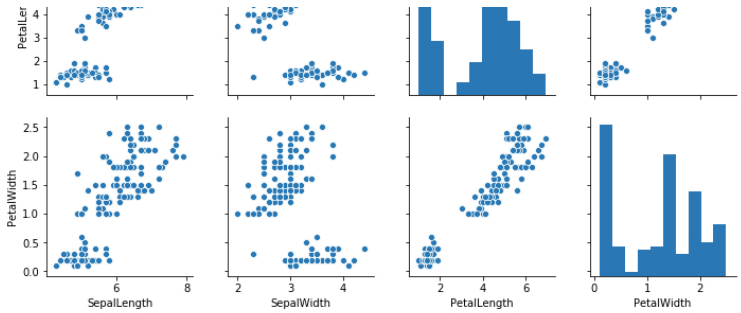
Answer: From the above pairplots, I think the best two attributes to make the class seperable will be PetalLength and PetalWidth

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?

In [29]:
```
sns.pairplot(iris_df)
```

Out[29]: <seaborn.axisgrid.PairGrid at 0x2ba5629f2450>





Answer: We can do pair plots and observe if the points are seperable forming seperate clusters. If it is, then we will choose the corresponding feature pairs.

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 PCA to do the dimensionaly reduction. The limitations of this approach is that there should be linear trend between the features.

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 [30]:
```
data = iris_df[['SepalLength','SepalWidth','PetalLength','PetalWidth']]
```

In [31]:
```
data.dtypes
```

Out[31]:
```
SepalLength    float64
SepalWidth     float64
PetalLength    float64
PetalWidth     float64
dtype: object
```

In [32]:
```
data_mean = np.mean(data, axis=0)
```

In [33]:
```
data_c = data - data_mean
```

In [34]:
```
data_c_t = data_c.T
```

In [35]:
```
n = data.shape[0]
```

In [36]:
```
covr = np.dot(data_c_t, data_c)/(n-1)
```

In [38]:
```
np.allclose(covr, data.cov())
```

Out[38]:
```
True
```

Get the eigen values and eigen vector

In [41]:
```
eigenvalues, eigenvectors = np.linalg.eig(covr)
```

In [42]:
```
#pick first two principal components
sum(eigenvalues[:2])/sum(eigenvalues)
```

Out[42]:
```
0.9776317750248035
```

it is capturing 97.7% variance

In [46]:
```
principal_components = eigenvectors[:, :2]
```

In [47]:
```
projected_data = np.dot(data, principal_components)
```

In [48]:
```
projected_data.shape
```

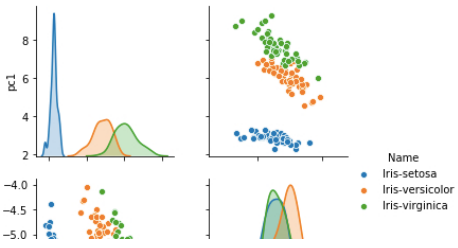
Out[48]:
```
(150, 2)
```

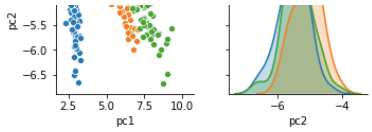
Question 6b: Generate a pairplot (along with colors for the different types of iris plants) between the two newly generated features using PCA in the above step.

In [56]:
```
projected_columns = {'pc1': projected_data[:, 0].tolist(), 'pc2': projected_data[:, 1].tolist(), 'Name': iris_df['Name'
projected_data_df = pd.DataFrame(projected_columns)
```

In [58]:
```
sns.pairplot(projected_data_df, hue='Name')
```

Out[58]:
```
<seaborn.axisgrid.PairGrid at 0x2ba569264490>
```





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 5a?

Answer: The pc1 can be used to do clustering as we can see from the above plots that the classes are seperable using pc1 compared to pc2.
The new feature is no better than the feature in 5a, infact they both are able to form seperate clusters.Both have same eifficiency in seperating the points to two clusters

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 5b?

Answer: The new features are no better than the features selected in 5b, as they both are able to cluster the points into two seperable cluster. Both have same eifficiency in seperating the points to two clusters

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.

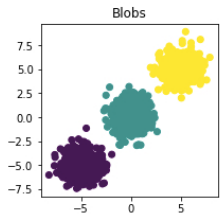
In [61]:
```
n_samples = 1500
random_state = 42
centers = [(-5, -5), (0, 0), (5, 5)]
Blobs_X, Blobs_y = datasets.make_blobs(n_samples=n_samples,centers=centers,random_state=random_state)
```

In [62]:
```
Blobs_X.shape
```

Out[62]:
```
(1500, 2)
```

In [63]:
```
plt.figure(figsize=(3,3))
plt.scatter(Blobs_X[:, 0], Blobs_X[:, 1], c= Blobs_y)
plt.title('Blobs')
```

Out[63]:
```
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 [64]:
```
Blobs1= pd.DataFrame(Blobs_X)
Blobs1['2'] = np.random.randn(1500)
Blobs1.head()
```

Out[64]:

	0	1	2
0	0.168461	1.317598	-1.117178
1	-3.534351	-5.225776	0.789353
2	-6.525525	-5.691908	-0.782990
3	-0.120948	0.419532	1.929008
4	-5.469474	-4.457440	0.510327

In [78]:
```
Blobs1.columns
```

Out[78]:
```
Index([0, 1, u'2'], dtype='object')
```

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.

In [65]:
```
Blobs2= pd.DataFrame(Blobs_X)
Blobs2['2'] = np.random.randn(1500)*10
Blobs2.head()
```

Out[65]:

	0	1	2
0	0.168461	1.317598	-2.213254
1	-3.534351	-5.225776	5.400809
2	-6.525525	-5.691908	25.025208
3	-0.120948	0.419532	-3.326950
4	-5.469474	-4.457440	-2.025654

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 [66]:
```
Blobs3= pd.DataFrame(Blobs_X)
Blobs3['2'] = np.random.randn(1500)*10
```

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

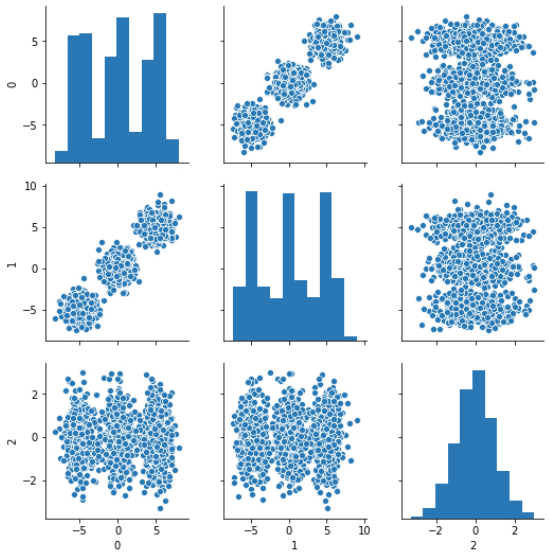
Out[66]:

	0	1	2	3
0	0.168461	1.317598	-2.461219	11.500462
1	-3.534351	-5.225776	-2.559341	12.001297
2	-6.525525	-5.691908	-12.321272	-6.584002
3	-0.120948	0.419532	8.703272	-1.396558
4	-5.469474	-4.457440	2.950061	-2.561777

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 [67]:
```
sns.pairplot(Blobs1)
```

Out[67]: <seaborn.axisgrid.PairGrid at 0x2ba5696cf110>



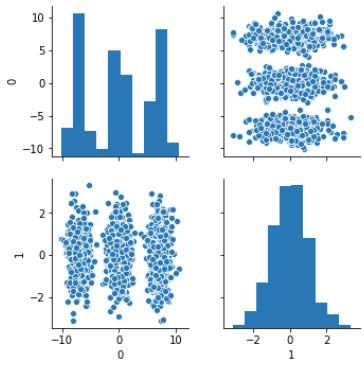
Answer: The variance of third attribute is lesser than that of 0 and 1 attribute. The third attribute is normally distributed
Also, we can observe from the above plots, there is linear trend between 0 and 1 attribute. There is no linear trend observed between 0 and 2 and 1 and 2 attribute.

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

In [107]:
```
projected_blobs1_df = pd.DataFrame(PCA(2).fit_transform(Blobs1))
```

In [108]:
```
sns.pairplot(projected_blobs1_df)
```

Out[108]: <seaborn.axisgrid.PairGrid at 0x2ba56ac73f90>



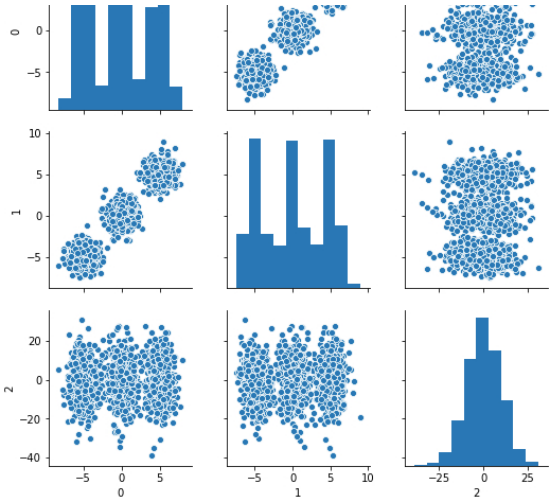
Question 7c: By comparing the distributions for the newly generated attributes in Question 7b with the previous pairplot in Question 7a, determine which attribute is captured by the first principal component and which attribute is captured by the second principal component. Provide a reason for your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

Answer: The first principal component has captured attributes 0 and 1. The second principal component has captured attribute 2. We can see from the above pair plots that the pc1 has same distribution to 0 and 1 attribute i.e. trimodal distribution. The pc2 has the same distribution to attribute 3 i.e. normal distribution.

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 [75]:
```
sns.pairplot(Blobs2)
```

Out[75]: <seaborn.axisgrid.PairGrid at 0x2aaed2f7ebd0>



Answer: The variance of the third attribute is higher than that of 0 and 1 attribute. The third attribute is normal distributed.

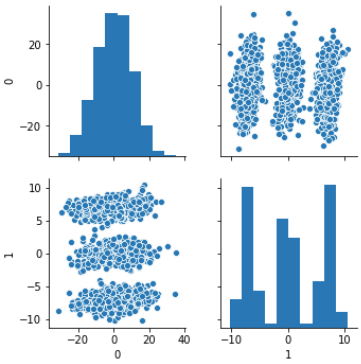
Question 7e: Perform PCA on **Blobs2** data. Project data onto the first two principal components. Generate a pairplot for the newly constructed attributes.

In [105]:
```
projected_blobs2 = PCA(2).fit_transform(Blobs2); projected_blobs3.shape
```

Out[105]:
```
(1500, 2)
```

In [106]:
```
blobs2_df = pd.DataFrame(projected_blobs2)
sns.pairplot(blobs2_df)
```

Out[106]:
```
<seaborn.axisgrid.PairGrid at 0x2ba56ae4c590>
```



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 2 attribute is captured by 1st princinpal component. 0 and 1 attribute is captured by 2nd principal component. The attribute 2 has the most variance, therefore it is captured by 1st principal component.

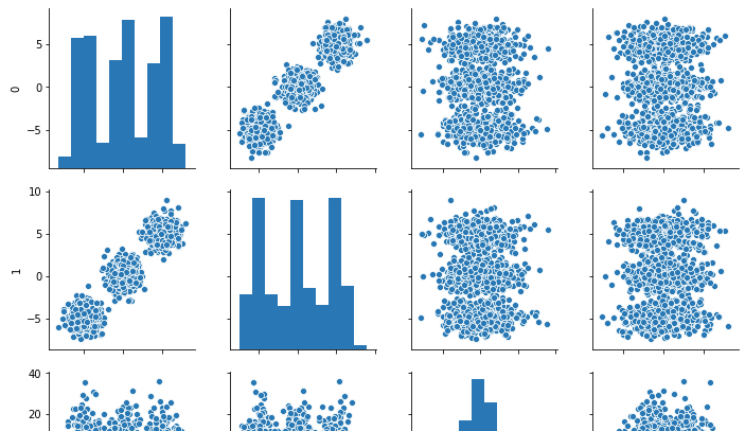
Question 7g: Are the three blobs separately visible after projection based on PCA in Question 7e?

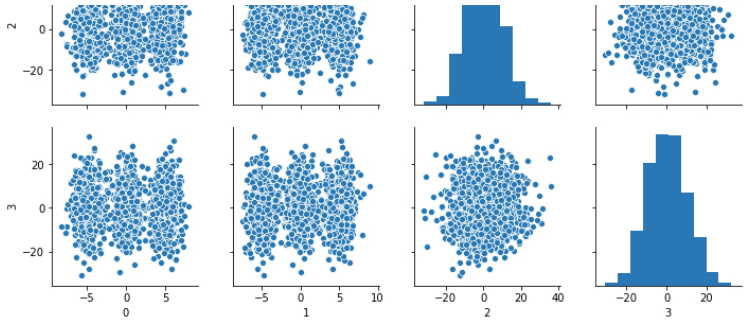
Answer: Yes

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 [80]:
```
sns.pairplot(Blobs3)
```

Out[80]:
```
<seaborn.axisgrid.PairGrid at 0x2aaed3379250>
```





Answer: The attributes 0 and 1 have a linear trend between them. They have positive correlation. The attribute 2 and 3 dont have the linear trend between them. They have no correlation.

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

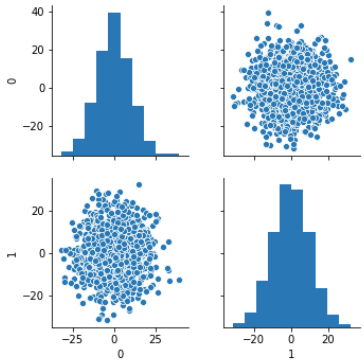
In [99]:
```
projected_blobs3 = PCA(2).fit_transform(Blobs3); projected_blobs3.shape
```

Out[99]:
```
(1500, 2)
```

In [ ]:
```
```

In [100]:
```
projected_blobs3_df = pd.DataFrame(projected_blobs3)
sns.pairplot(projected_blobs3_df)
```

Out[100]:
```
<seaborn.axisgrid.PairGrid at 0x2ba569d3db90>
```



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 attributes 2 and 3 is captured by the first and second principal components due to higher variance in attributes 2 and 3 compared to attribute 0 and 1.

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, they are not seperable. There is no linear trend between attribute 2 and 3 and they are not contributing in making the blobs seperable. And the PCA is picking the attribute 2 and 3 due to their higher variance.

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

Answer: The PCA picks the attributes which have higher variance without considering whether the attributes are contributing in making the blobs seperable.

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 [97]:
```
def mysvd(A):
    sigma_points = np.dot(A, A.T)
    A_c = A-np.mean(A, axis= 0)
    sigma_attributes = np.dot(A_c.T, A_c)/(A.shape[0] -1 )

    ## Getting U and delta values
    delta_values, U = np.linalg.eigh(sigma_points)
    delta_values_sorted_id = delta_values.argsort()[::-1]
    delta_values = delta_values[delta_values_sorted_id]
    U = U[:, delta_values_sorted_id]
    rows_1_1 = np.nonzero(delta_values)
    non_zero_1_1 = delta_values[rows_1_1].copy()
    temp = np.diag(np.sqrt(non_zero_1_1))
    S_1 = np.zeros_like(A).astype(np.float64)
    S_1[:temp.shape[0],:temp.shape[1]] = temp[:A.shape[0], :A.shape[1]]

    ## Getting V
    eigenvalues, V = np.linalg.eigh(sigma_attributes)
    eigenvalues_id = eigenvalues.argsort()[::-1]
    eigenvalues = eigenvalues[eigenvalues_id]
    V = V[:,eigenvalues_id]

    return U, S_1, 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 [98]:
```
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 [99]:
```
U, S, V = mysvd(A)
```

```
/usr/local/python/2.7-conda5.2/lib/python2.7/site-packages/ipykernel/__main__.py:13: RuntimeWarning: invalid value en
countered in sqrt
```

In [105]:
```
np.dot(np.dot(U, S), V.T)
```

Out[105]:
```
array([[ 0.8226186 ,  0.7619401 ,  0.82261861, -0.69710954, -0.69710978,
        -0.30679111],
       [ 2.46785578,  2.28582031,  2.46785586, -2.09132899, -2.09132896,
        -0.92037334],
       [ 3.29047443,  3.04776041,  3.29047442, -2.78843864, -2.78843863,
        -1.22716445],
       [ 4.11309305,  3.80970051,  4.11309301, -3.4855483 , -3.48554828,
        -1.53395556],
       [-1.83987561, -1.47981184, -1.83987561, -3.44237454, -3.44237454,
        -1.15757828],
       [-2.21651292, -3.06779271, -2.21651292, -4.14054844, -4.14054844,
        -0.68884398],
       [-0.89170765, -1.36962352, -0.89170765, -1.98191773, -1.98191773,
         0.82331682]])
```

In [108]:
```
np.allclose(A, np.dot(np.dot(U, S), V.T), rtol=10)
```

Out[108]:
```
True
```

The reconstructed matrix loses a lot of information

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 [148]:
```
import pandas as pd
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/master/iris.csv')
```

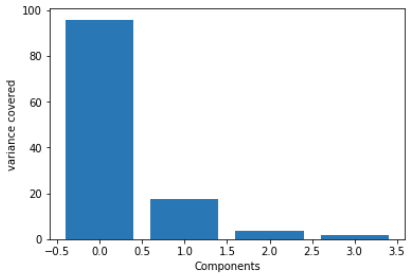
In [149]:
```
data = iris_df.values[:,0:4]
data = data.astype(float) #converts data format from object to numeric
```

In [150]:
```
U_iris, S_iris, V_iris = svd(data, full_matrices=False)
```

In [154]:
```
spectral_values = S_iris
```

In [155]:
```
plt.bar(np.arange(4),spectral_values)
plt.xlabel('Components')
plt.ylabel('variance covered')
```

Out[155]:
```
Text(0,0.5,'variance covered')
```



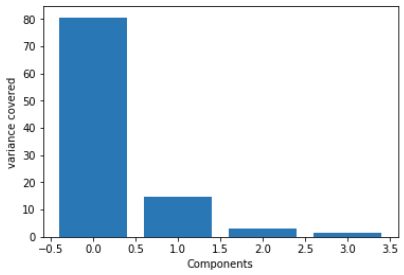
In [159]:
```
spectral_values_percentage = (S_iris/sum(S_iris))*100
```

In [160]:
```
spectral_values_percentage
```

Out[160]:
```
array([80.61602452, 14.89050648,  2.91484064,  1.57862836])
```

In [161]:
```
plt.bar(np.arange(4),spectral_values_percentage)
plt.xlabel('Components')
plt.ylabel('variance covered')
```

Out[161]:
```
Text(0,0.5,'variance covered')
```

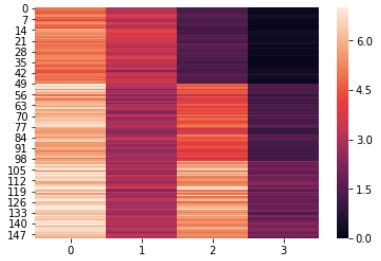


Answer: From the above plot, we can say the last two dimensions are capturing less than 10% variance

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

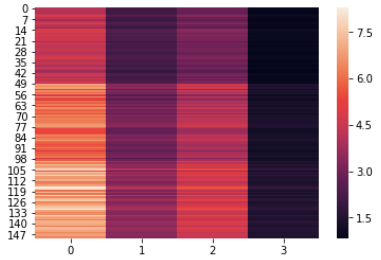
In [162]:
```
sns.heatmap(data,vmin=0, vmax=7)
```

Out[162]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6eaa32fe10>
```



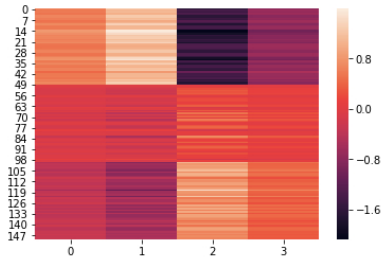
In [180]:
```
sns.heatmap(S_iris[0]*np.outer(U_iris[:,0],V_iris[0,:]))
```

Out[180]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6eaab59a90>
```



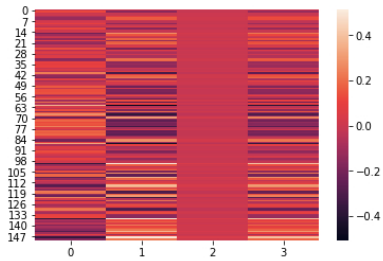
In [181]:
```
sns.heatmap(S_iris[1]*np.outer(U_iris[:,1],V_iris[1,:]))
```

Out[181]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6eaad29990>
```



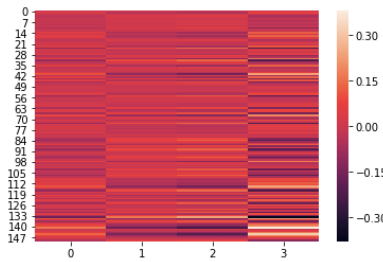
In [182]:
```
sns.heatmap(S_iris[2]*np.outer(U_iris[:,2],V_iris[2,:]))
```

Out[182]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6eaadf7390>
```



In [183]:
```
sns.heatmap(S_iris[3]*np.outer(U_iris[:,3],V_iris[3,:]))
```

Out[183]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x2b6eaa9075d0>
```



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: The last two matrices have relatively smaller magnitude. The variance covered by the last two spectral values are less which is observed in the heatmap of the last two matrices.

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]:
```
digits.data.shape
```

Out[185]:
```
(1797, 64)
```

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

In [186]:
```
digits.target
```

Out[186]:
```
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 [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.])
```

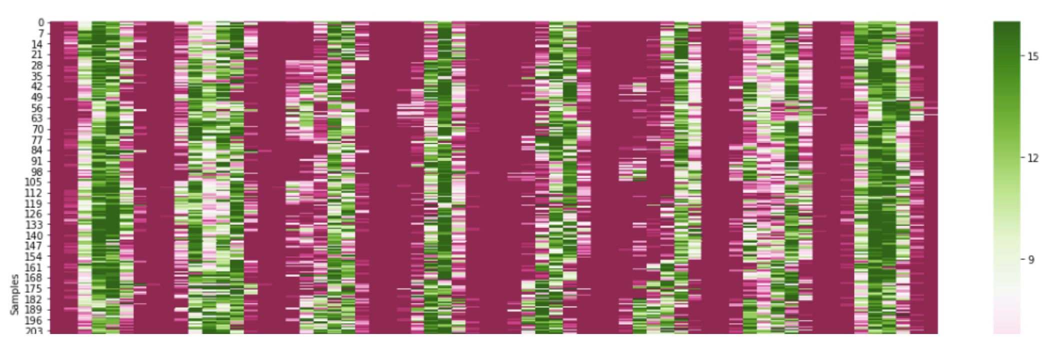
In [192]:
```
y.shape
```

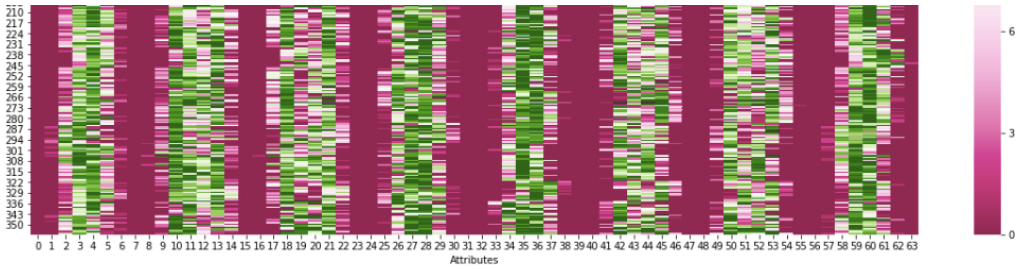
Out[192]:
```
(357,)
```

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 [108]:
```
plt.figure(figsize=(20,10))
ax = sns.heatmap(X,cmap='PiYG')
ax.set(xlabel='Attributes', ylabel='Samples')
```

Out[108]:
```
[Text(159,0.5,'Samples'), Text(0.5,69,'Attributes')]
```





Answer: The 42nd attribute is able to classify the 3s and 8s. Approximately 28 mistakes would be committed if this attribute is used for projection.

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.

Answer: None of the datasets will be wrongly predicted as there is clear sepeartion between the classes

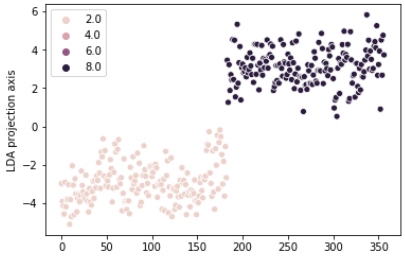
In [193]:
```
lda = LinearDiscriminantAnalysis(n_components=1)
X_r1 = lda.fit(X, y).transform(X)
```

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

In [194]:
```
X_r1.shape
```

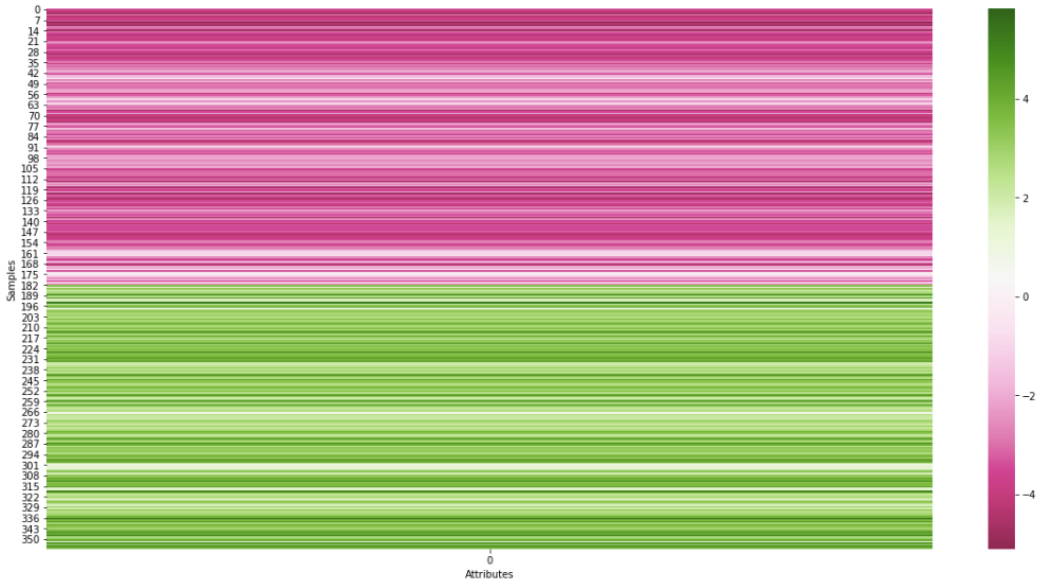
Out[194]:
```
(357, 1)
```

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)
```



In [196]:
```
plt.figure(figsize=(20,10))
ax = sns.heatmap(X_r1,cmap='PiYG')
ax.set(xlabel='Attributes', ylabel='Samples')
```

Out[196]:
```
[Text(159,0.5,'Samples'), Text(0.5,69,'Attributes')]
```



In [ ]: