In [49]:
```
# 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
```

In [75]:
```
path = '/Users/8umelec/Downloads/davis-data-set/Davis.csv'
path
```

Out[75]: '/Users/8umelec/Downloads/davis-data-set/Davis.csv'

In [82]:
```
davis_df1 = pd.read_csv(path)
```

In [83]:
```
davis_df1.dropna(inplace=True)
```

In [2]:
```
import pandas as pd
davis_df = pd.read_csv('https://vincentarelbundock.github.io/Rdatasets/cs
```

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

In [96]:
```
davis_df
```

Out[96]:

	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
5	6	M	76	170	76.0	165.0
6	7	M	76	167	77.0	165.0

7	8	M	69	186	73.0	180.0
8	9	M	71	178	71.0	175.0
9	10	M	65	171	64.0	170.0
10	11	M	70	175	75.0	174.0
11	12	F	166	57	56.0	163.0
12	13	F	51	161	52.0	158.0
13	14	F	64	168	64.0	165.0
14	15	F	52	163	57.0	160.0
15	16	F	65	166	66.0	165.0
16	17	M	92	187	101.0	185.0
17	18	F	62	168	62.0	165.0
18	19	M	76	197	75.0	200.0
19	20	F	61	175	61.0	171.0
20	21	M	119	180	124.0	178.0
21	22	F	61	170	61.0	170.0
22	23	M	65	175	66.0	173.0
23	24	M	66	173	70.0	170.0
24	25	F	54	171	59.0	168.0
25	26	F	50	166	50.0	165.0
26	27	F	63	169	61.0	168.0
27	28	F	58	166	60.0	160.0
28	29	F	39	157	41.0	153.0
29	30	M	101	183	100.0	180.0
...	...	...	...	...	...	...
164	165	M	56	163	58.0	161.0
165	166	F	59	159	59.0	155.0
166	167	F	63	170	62.0	168.0
167	168	F	66	166	66.0	165.0
168	169	M	96	191	95.0	188.0

169	170	F	53	158	50.0	155.0
170	171	M	76	169	75.0	165.0
172	173	M	61	170	61.0	170.0
174	175	M	62	168	64.0	168.0
175	176	M	71	178	68.0	178.0
177	178	M	66	170	67.0	165.0
178	179	M	81	178	82.0	175.0
179	180	M	68	174	68.0	173.0
180	181	M	80	176	78.0	175.0
183	184	F	63	165	59.0	160.0
184	185	M	70	173	70.0	173.0
185	186	F	56	162	56.0	160.0
186	187	F	60	172	55.0	168.0
187	188	F	58	169	54.0	166.0
188	189	M	76	183	75.0	180.0
189	190	F	50	158	49.0	155.0
190	191	M	88	185	93.0	188.0
191	192	M	89	173	86.0	173.0
192	193	F	59	164	59.0	165.0
193	194	F	51	156	51.0	158.0
194	195	F	62	164	61.0	161.0
195	196	M	74	175	71.0	175.0
196	197	M	83	180	80.0	180.0
198	199	M	90	181	91.0	178.0
199	200	M	79	177	81.0	178.0

181 rows × 6 columns

In [ ]:
```
*Question 1a:** What does the data capture?
```

In [ ]:
```
The data captures 5 attributes sex,weight,height,representative height an
```

In [ ]:
```
**Question 1b:** Who are selected as subjects in the study that collected
```

In [ ]:
```
The subjects are men and woman doing regular exercise.
```

In [97]:
```
davis_df.shape
```

Out[97]: (181, 6)

In [ ]:
```
**Question 1c:** How many data points are in this dataset?
```

In [ ]:
```
There are 181 data points collectively after removing Nan
```

In [ ]:
```
**Question 1d:** How many attributes are in this dataset?
```

In [102]:
```
davis_df.head()
```

Out[102]:

	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 [ ]:
```
There are 6 attributes in the dataset
```

In [ ]:
```
**Question 1e:** What type of attributes are present in the dataset?
```

In [108]:
```
davis_df.ndim
```

Out[108]: 2

In [101]:
```
davis_df.dtypes
```

Out[101]:
```
Unnamed: 0      int64
sex            object
weight          int64
height          int64
repwt         float64
repht         float64
dtype: object
```

In [ ]:
```
There are 4 attributes(weight,height,repwt,repht) and one label(sex)
```

In [127]:
```
davis_df.describe()
```

Out[127]:

	Unnamed: 0	weight	height	repwt	repht
count	181.000000	181.000000	181.000000	181.000000	181.000000
mean	97.480663	66.303867	170.154696	65.679558	168.657459
std	57.857448	15.340992	12.312069	13.834220	9.394668
min	1.000000	39.000000	57.000000	41.000000	148.000000
25%	46.000000	56.000000	164.000000	55.000000	161.000000
50%	96.000000	63.000000	169.000000	63.000000	168.000000
75%	146.000000	75.000000	178.000000	74.000000	175.000000
max	200.000000	166.000000	197.000000	124.000000	200.000000

In [77]:
```
davis_df.drop(columns=davis_df.columns[davis_df.columns.str.contains('unn
```

In [ ]:
```
**Question 2a:** What are range of values the numeric attributes take?
```

In [13]:
```
davis_df.describe(exclude=object)
```

Out[13]:

	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

In [ ]:
```
weight ranges from 39 to 166
height ranges from 57 to 197
repwt ranges from 41 to 124
repht ranges from 148 to 200
```

In [ ]:
```
**Question 2b:** What different values do categorical attributes take?
```

In [18]:
```
davis_df.describe(include=object)
```

Out[18]:

	sex
count	181
unique	2
top	F
freq	99

In [ ]:
```
categorical attributes take 2 values. That is M and F (unique=2; top bein
```

In [ ]:
```
**Question 2c:** What are the mean values for each of the numeric attribu
```

In [21]:
```
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
```

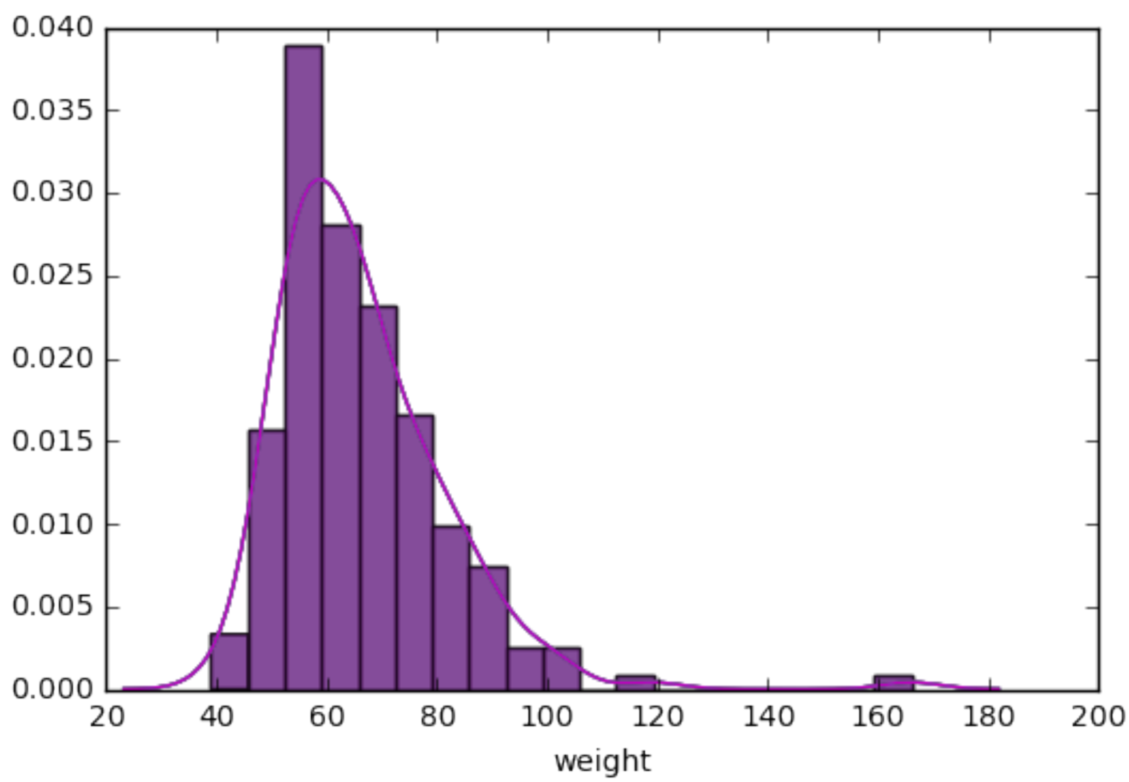
In [ ]:
```
**Question 2d:** What is the variance for each of the numeric attributes?
```

In [23]:
```
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 variance = %.2f' % davis_df[col].var())
```

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

In [ ]:
```
**Question 2e:** Visually examine how the attribute weight is distributed
```

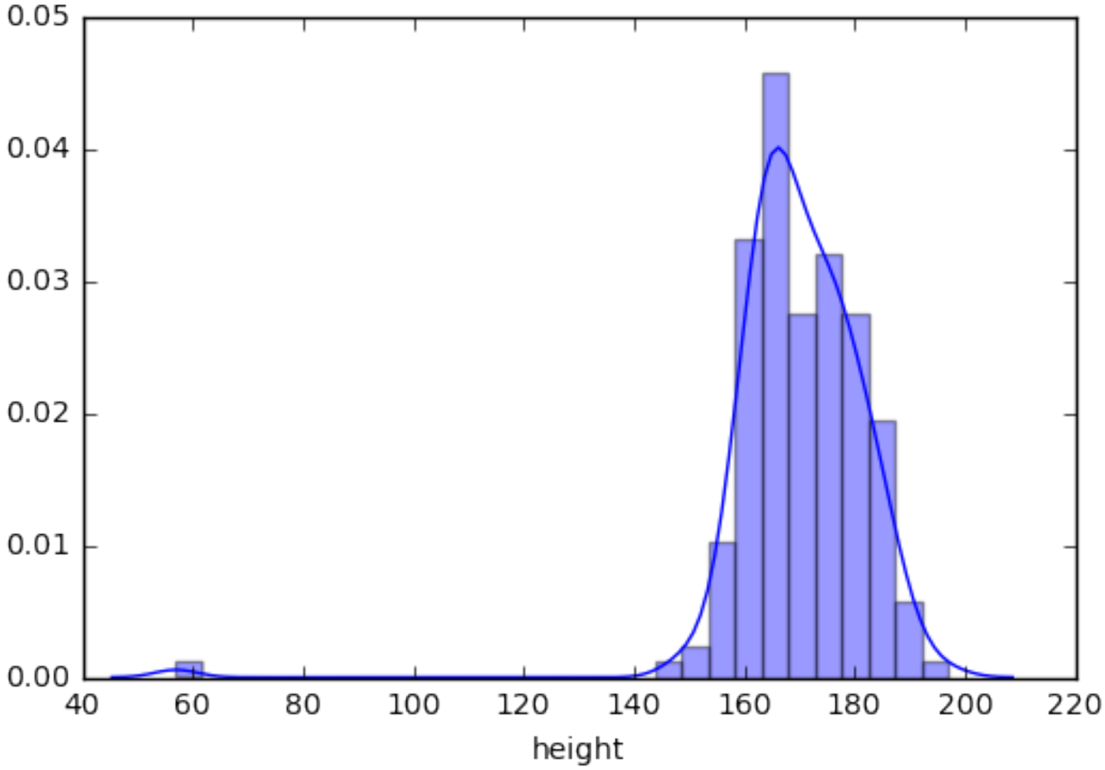
In [8]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.distplot(davis_df['weight'])
plt.show()
```



In [ ]:
```
As we can see, the data(weight attribute has mean around 66) is normally
```

In [ ]: `*Question 2f:** Visually examine how the attribute height is distributed a`

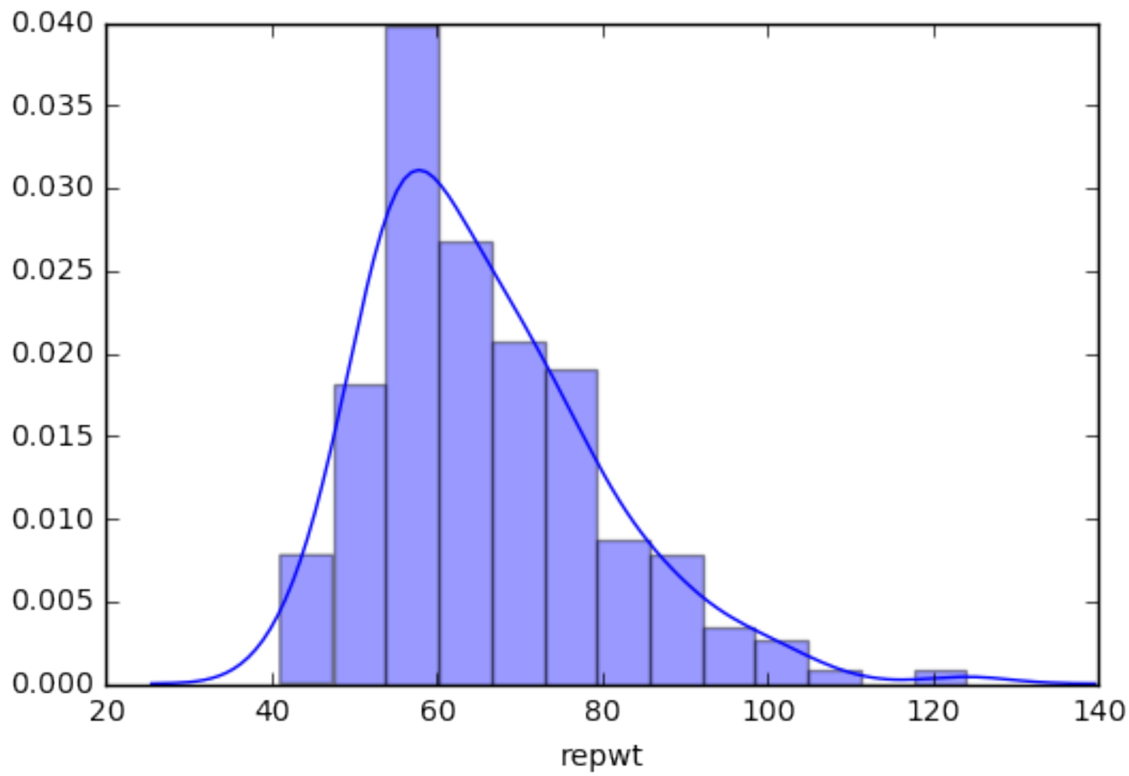
In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.distplot(davis_df['height'])
plt.show()
```



In [ ]: `The attribute height is also normally distributed(highest density around`

In [ ]: `**Question 2g:** Visually examine how the attribute repwt is distributed`

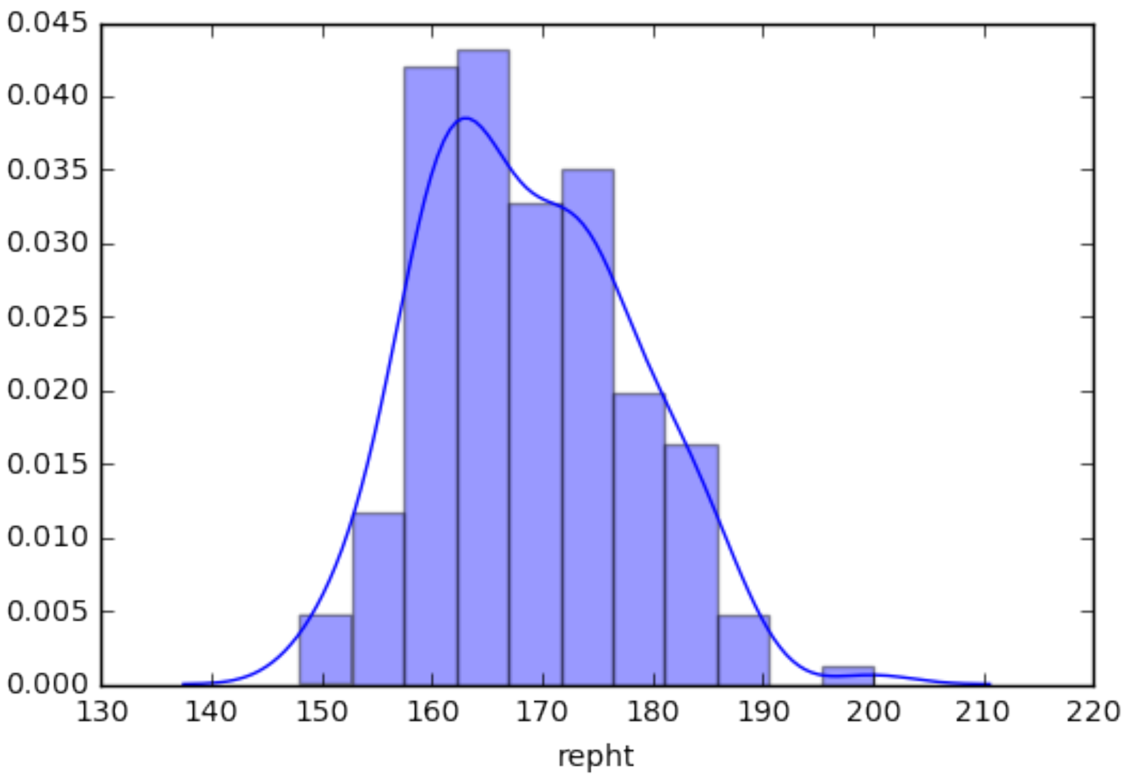
In [10]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.distplot(davis_df['repwt'])
plt.show()
```



In [ ]:
```
Attribute repwt is normally distributed
```

In [ ]:
```
*Question 2h:** Visually examine how the attribute repht is distributed ar
```

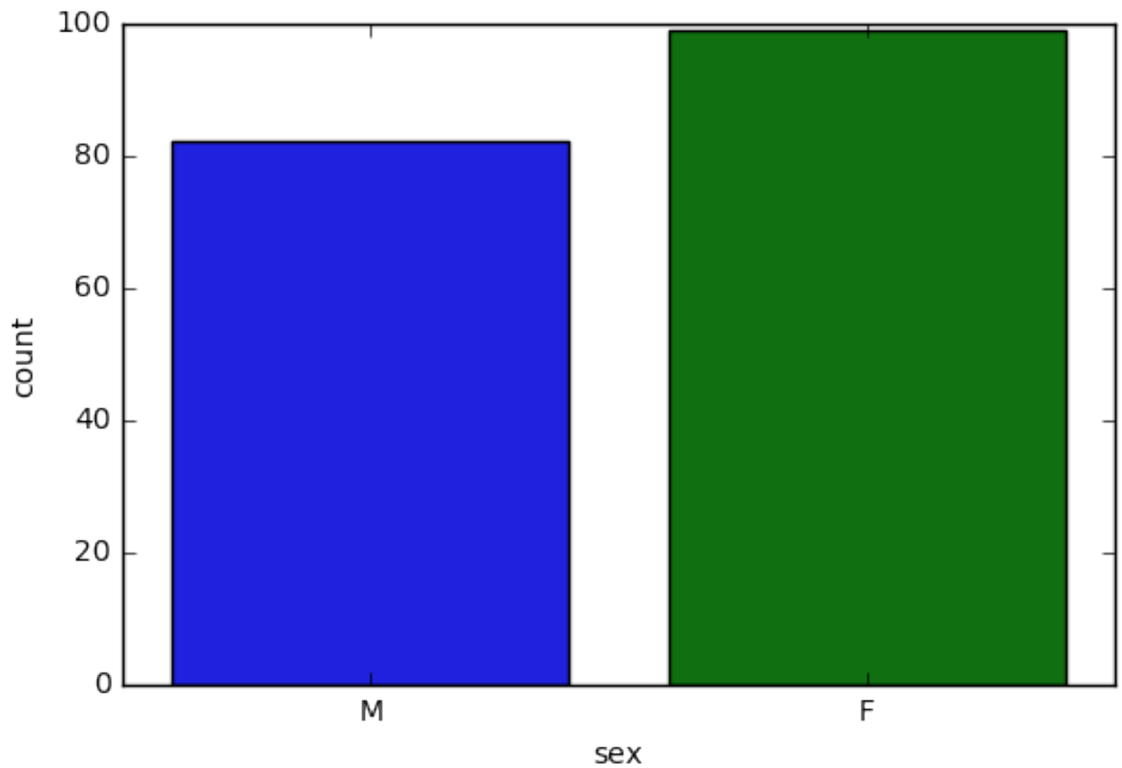
In [11]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.distplot(davis_df['repht'])
plt.show()
```



In [ ]:
```
repht is not distributed normally since bell shape is not properly preser
it looks like bimodal.
```

In [ ]:
```
**Question 2i:** Visually examine how the attribute sex is distributed and
```

In [13]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(davis_df['sex'])
plt.show()
```



In [ ]:
```
Data(sex) is not uniformly distributed. Female count dominates male count
```

In [14]:
```
davis_df_new = davis_df[['repwt','repht']]
```

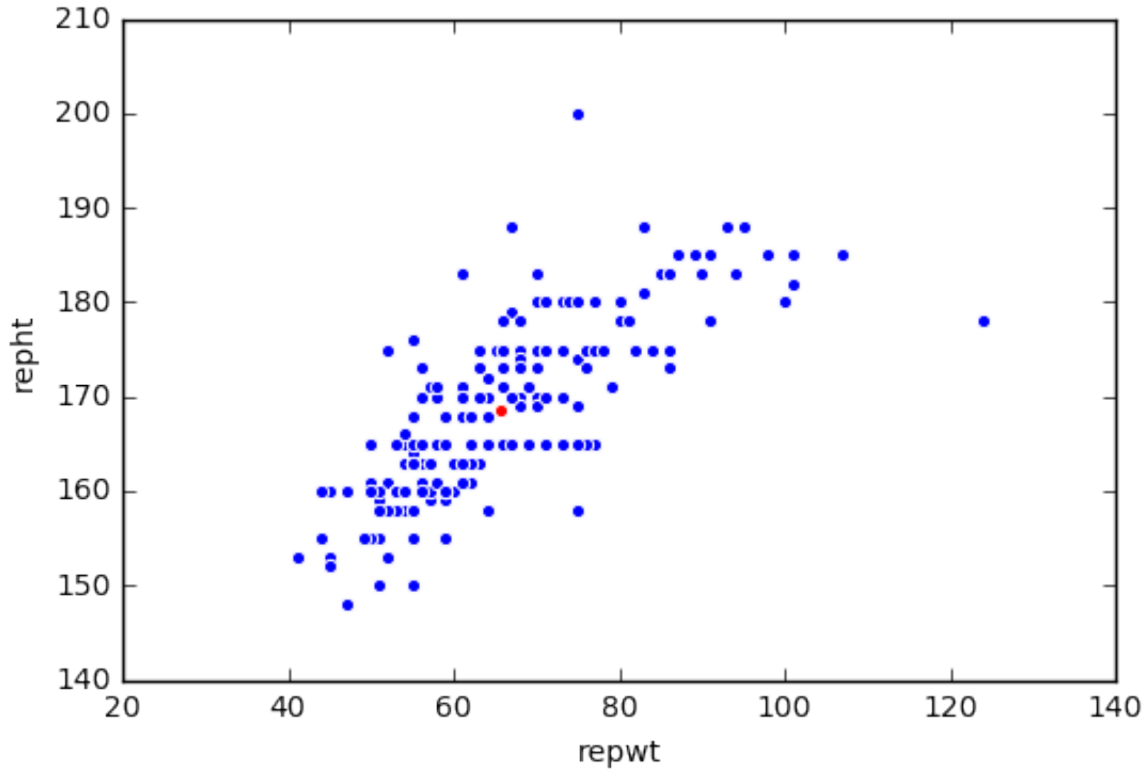
In [17]:
```
davis_df_new.head(5)
```

Out[17]:

	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

In [ ]:
```
**Question 3a:** Show the Geometric view of this new row normalized data
```

In [21]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
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)
plt.show()
```



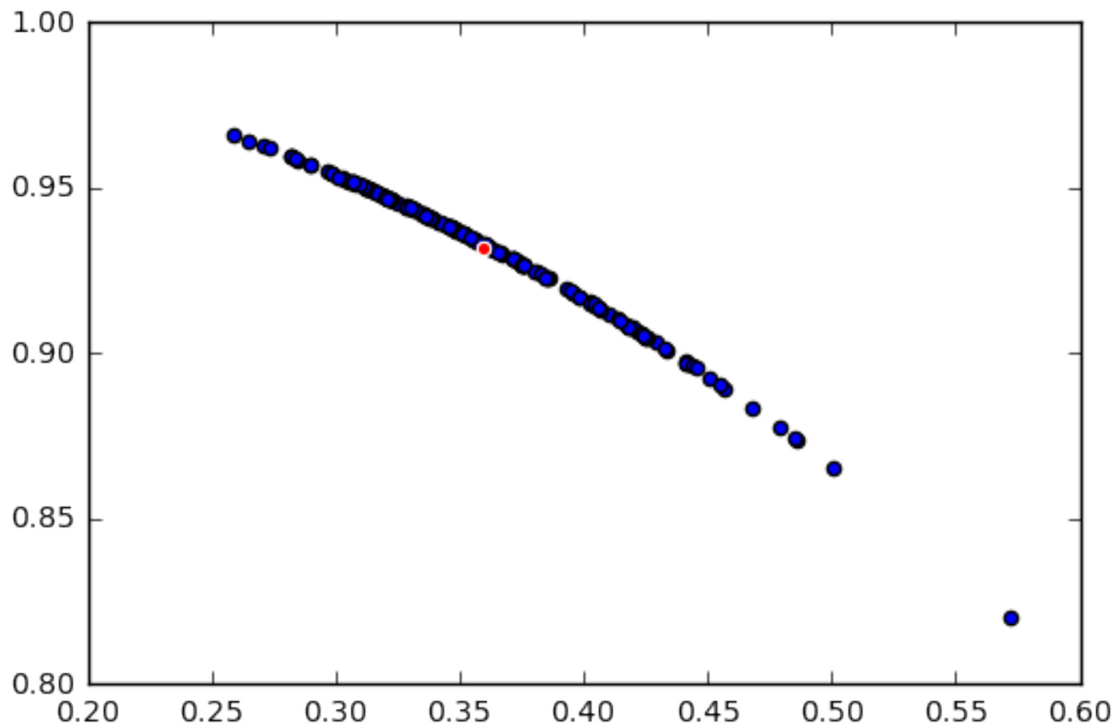
In [27]:
```
  from sklearn.preprocessing import normalize
davis_df_new_row_norm = normalize(davis_df_new, axis=1, norm='l2')
```

In [29]:
```
 davis_df_new_row_norm[1:10,:]
```

Out[29]:
```
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 [ ]:
```
**Question 3b:** Show the Geometric view of this new row normalized data
    Comment on the Geomateric view of the data in comparison to the view
    3a. Provide a reason for the difference in the geometric views in Que
```

In [46]:
```
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
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)
plt.show()
```

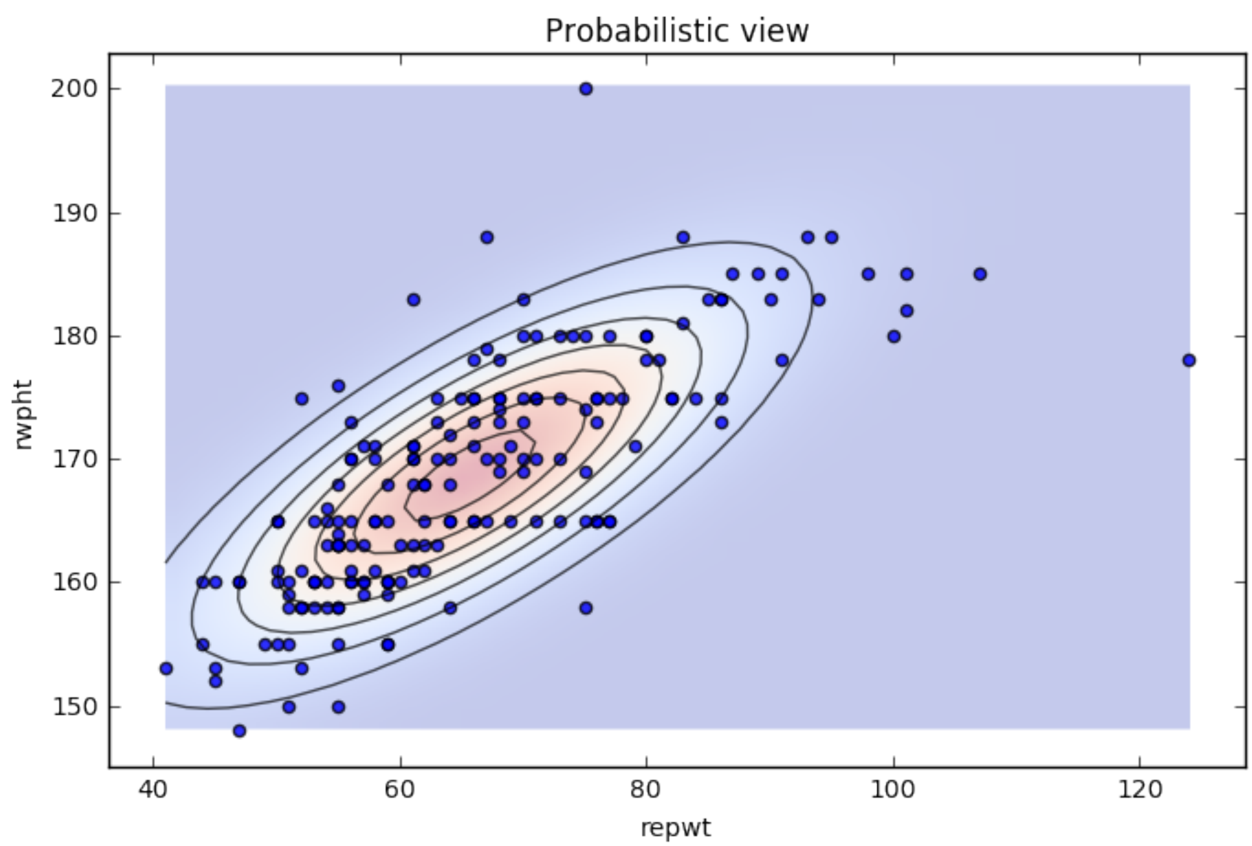


In [ ]:
```
As compared to the previous plot, data looks normalized here ie not scatt
perform statistical analysis. The reason behind this change is we normali
are between 0.20 and 0.60 approximately.
```

In [ ]:
```
**Question 3c:** Show the Probabilistic view of the data davis_df_new.
```

In [57]:
```
from scipy.stats import multivariate_normal
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
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)
fig = plt.figure(figsize=(8,8))
ax = fig.gca()
ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_length,max_length, mi
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('rwpht')
plt.title('Probabilistic view')
plt.show()
```



In [ ]:
```
**Question 3d:** Show the Probabilistic view of the data davis_df_new_col
    covariance structure in the Gaussian distribution with that of Questi
    has affected the shape of the covariance structure.
```

In [63]:
```
from sklearn.preprocessing import normalize
davis_df_new_col_norm = normalize(davis_df_new, axis=0, norm='l2')
```

In [71]:
```
from scipy.stats import multivariate_normal
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
mu = np.mean( davis_df_new_row_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 [75]:
```
from scipy.stats import multivariate_normal
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
fig = plt.figure(figsize=(8,8))
ax = fig.gca()
ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_length,max_length, mi
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')
plt.show()
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
<ipython-input-75-2d727dbb5603> in <module>()
      6 ax = fig.gca()
      7 ax.imshow(np.rot90(Z), cmap='coolwarm', extent=[min_length,max
_length, min_width,max_width], alpha=0.3)
----> 8 cset = ax.contour(x, y, Z, colors='k', alpha=0.7)
      9 plt.scatter( davis_df_new_col_norm[:,0], davis_df_new_col_norm
[:,1])
     10 ax.set_xlabel('repwt')

/Users/8umelec/anaconda/lib/python3.5/site-packages/matplotlib/__init_
```

```
_.py in inner(ax, *args, **kwargs)
   1817                     warnings.warn(msg % (label_namer, func.__n
ame__),
   1818                                   RuntimeWarning, stacklevel=2
)
-> 1819             return func(ax, *args, **kwargs)
   1820         pre_doc = inner.__doc__
   1821         if pre_doc is None:

/Users/8umelec/anaconda/lib/python3.5/site-packages/matplotlib/axes/_a
xes.py in contour(self, *args, **kwargs)
   5617             self.cla()
   5618         kwargs['filled'] = False
-> 5619         return mcontour.QuadContourSet(self, *args, **kwargs)
   5620     contour.__doc__ = mcontour.QuadContourSet.contour_doc
   5621

/Users/8umelec/anaconda/lib/python3.5/site-packages/matplotlib/contour
.py in __init__(self, ax, *args, **kwargs)
   1422         are described in QuadContourSet.contour_doc.
   1423         """
-> 1424         ContourSet.__init__(self, ax, *args, **kwargs)
   1425
   1426     def _process_args(self, *args, **kwargs):

/Users/8umelec/anaconda/lib/python3.5/site-packages/matplotlib/contour
.py in __init__(self, ax, *args, **kwargs)
    862
    863         self._process_args(*args, **kwargs)
--> 864         self._process_levels()
    865
    866         if self.colors is not None:

/Users/8umelec/anaconda/lib/python3.5/site-packages/matplotlib/contour
.py in _process_levels(self)
   1200         # The following attributes are no longer needed, and
   1201         # should be deprecated and removed to reduce confusion
.
-> 1202         self.vmin = np.amin(self.levels)
   1203         self.vmax = np.amax(self.levels)
   1204

/Users/8umelec/anaconda/lib/python3.5/site-packages/numpy/core/fromnum
eric.py in amin(a, axis, out, keepdims)
   2395     else:
   2396         return _methods._amin(a, axis=axis,
-> 2397                               out=out, **kwargs)
   2398
   2399
```

```
/Users/8umelec/anaconda/lib/python3.5/site-packages/numpy/core/_method
s.py in _amin(a, axis, out, keepdims)
     27
     28 def _amin(a, axis=None, out=None, keepdims=False):
---> 29     return umr_minimum(a, axis, None, out, keepdims)
     30
     31 def _sum(a, axis=None, dtype=None, out=None, keepdims=False):

ValueError: zero-size array to reduction operation minimum which has n
o identity
```

In [ ]:
```
**Question 4a:** What is the covariance matrix?
```

In [78]:
```
davis_df.cov()
```

Out[78]:

	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

In [ ]:
```
**Question 4b:** Which pairs of attributes co-vary in the opposite direct
```

In [ ]:
```
As there is no opposite polarity, we can say there there exist no opposit
```

In [ ]:
```
**Question 4c:** Which pairs of attributes are highly correlated?
```

In [79]:
```
davis_df.corr()
```

Out[79]:

	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

In [ ]:
```
weight-repwt(0.835),rewt-repht(0.76186),height-repht(0.739) are highly co
```

In [ ]:
```
**Question 4d:** Which pairs of attributes are uncorrelated?
```

In [ ]:
```
height and weight are least correlated(0.154258)
```

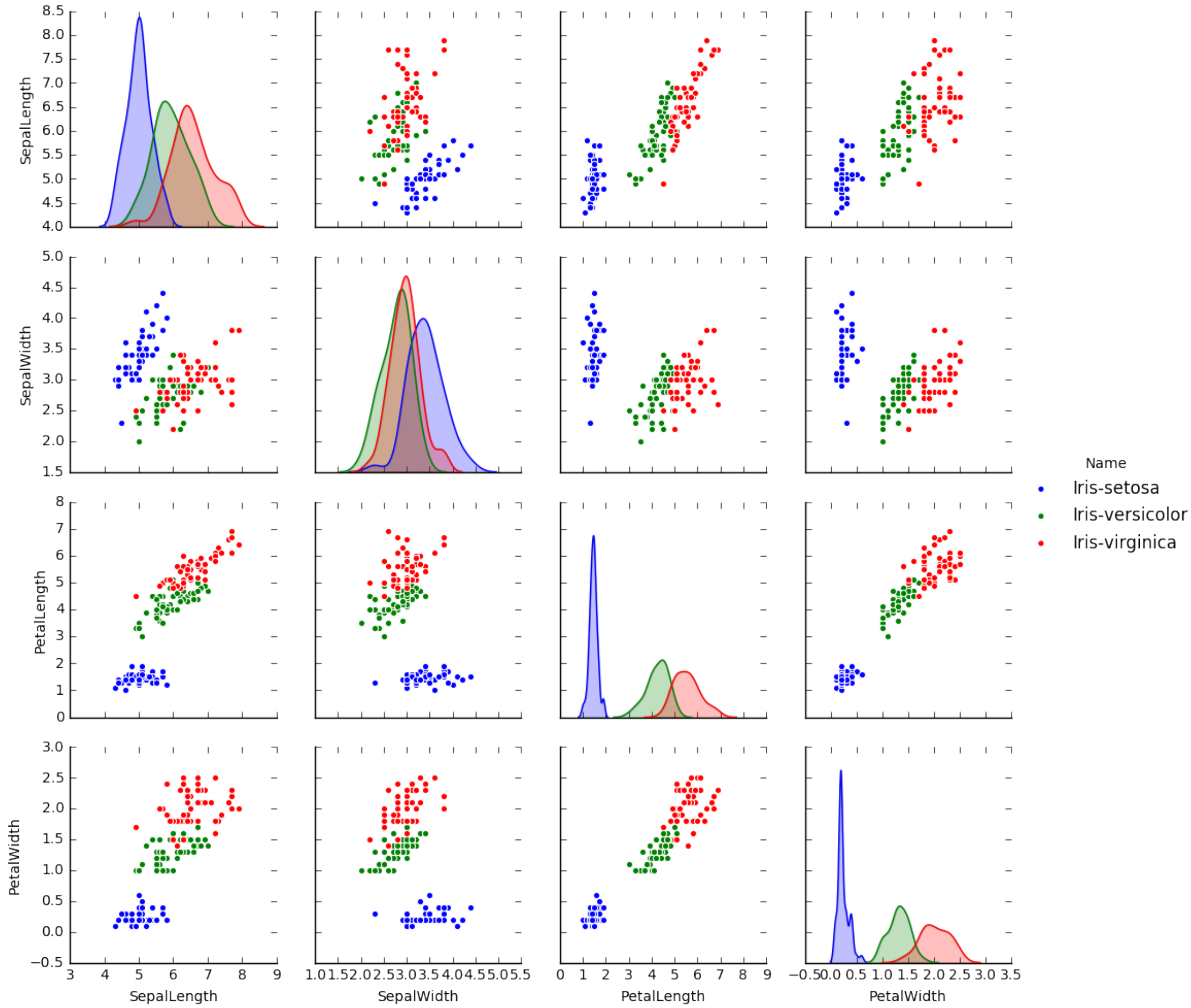
In [ ]:
```
**Question 4e:** What information did you gather from a correlation matri
    covariance matrix?
```

In [ ]:
```
As correlated matrix defines strength of two attributes, it gives strongl
is not normalized. Therefore we cannot find the strngth btw two attribute
```

In [81]:
```
import seaborn as sns
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/
```

In [ ]:
```
**Question 5a:** If you are allowed to select only one attribute, which a
    useful for the clustering task. Provide a reason. Use pairplot to ans
```

In [84]:
```
sns.pairplot(iris_df, hue="Name")
plt.show()
```



In [ ]:
```
Petal length and petal width are two attributes which are feasible to sep
I would select one of those attributes to perform clustering
```

In [ ]:
```
*Question 5b:** If you are allowed to select only two features, which feat
   Provide a reason. Use pairplot to answer this question.
```

In [ ]:
```
I would select petal length and petal width as stated earlier
```

In [ ]:
```
*Question 5c:** In real-world problems ground-truth (types of iris plants)
   how do you perform feature selection in that case?
```

In [ ]:
```
we can compute pairplots of the data given and observe if there are any s
thus we can select out features.
```

In [ ]:
```
*Question 5d:** In real-world problems ground-truth (types of iris plants)
   What limitations does your approach have?
```

In [ ]:
```
As earlier said, we can compute scatter or plairplots to explore the tren
whether we can seperate data or reduce the dimensionality. One disadvanta
if data is not linear, we need to opt different approach such as kernal P
```

In [ ]:
```
**Question 6a:** Perform PCA on Iris dataset and project the data onto th
    Use the attributes 'SepalLength','SepalWidth','PetalLength', and 'Pet
Hint: Use iris_df[['SepalLength','SepalWidth','PetalLength','PetalWidth']
```

In [85]:
```
data = iris_df[['SepalLength','SepalWidth','PetalLength','PetalWidth']]
```

In [87]:
```
data_mean = np.mean(data, axis=0)
```

In [88]:
```
centered_data = data- data_mean
```

In [89]:
```
centered_data_Trans = centered_data.T
```

In [94]:
```
n= data.shape[0]
```

In [95]:
```
covariance = np.dot(centered_data_Trans,centered_data)/(n-1)
```

Out[95]:
```
True
```

In [96]:
```
eigen_val,eigen_vectors = np.linalg.eig(covariance)
```

In [110]:
```
np.cumsum(eigen_val)/np.sum(eigen_val)
```

Out[110]:
```
array([ 0.92461621,  0.97763178,  0.99481691,  1.        ])
```

In [97]:
```
eigen_val,eigen_vectors
```

Out[97]:
```
(array([ 4.22484077,  0.24224357,  0.07852391,  0.02368303]),
 array([[ 0.36158968, -0.65653988, -0.58099728,  0.31725455],
        [-0.08226889, -0.72971237,  0.59641809, -0.32409435],
        [ 0.85657211,  0.1757674 ,  0.07252408, -0.47971899],
        [ 0.35884393,  0.07470647,  0.54906091,  0.75112056]]))
```

In [98]:
```
sum(eigen_val[:2])/sum(eigen_val)*100
```

Out[98]:
```
97.763177502480346
```

In [ ]:
```
Here, our first two components cover 97%. Therefore we select first two v
```

In [99]:
```
principle_components = eigen_vectors[:,:2]
```

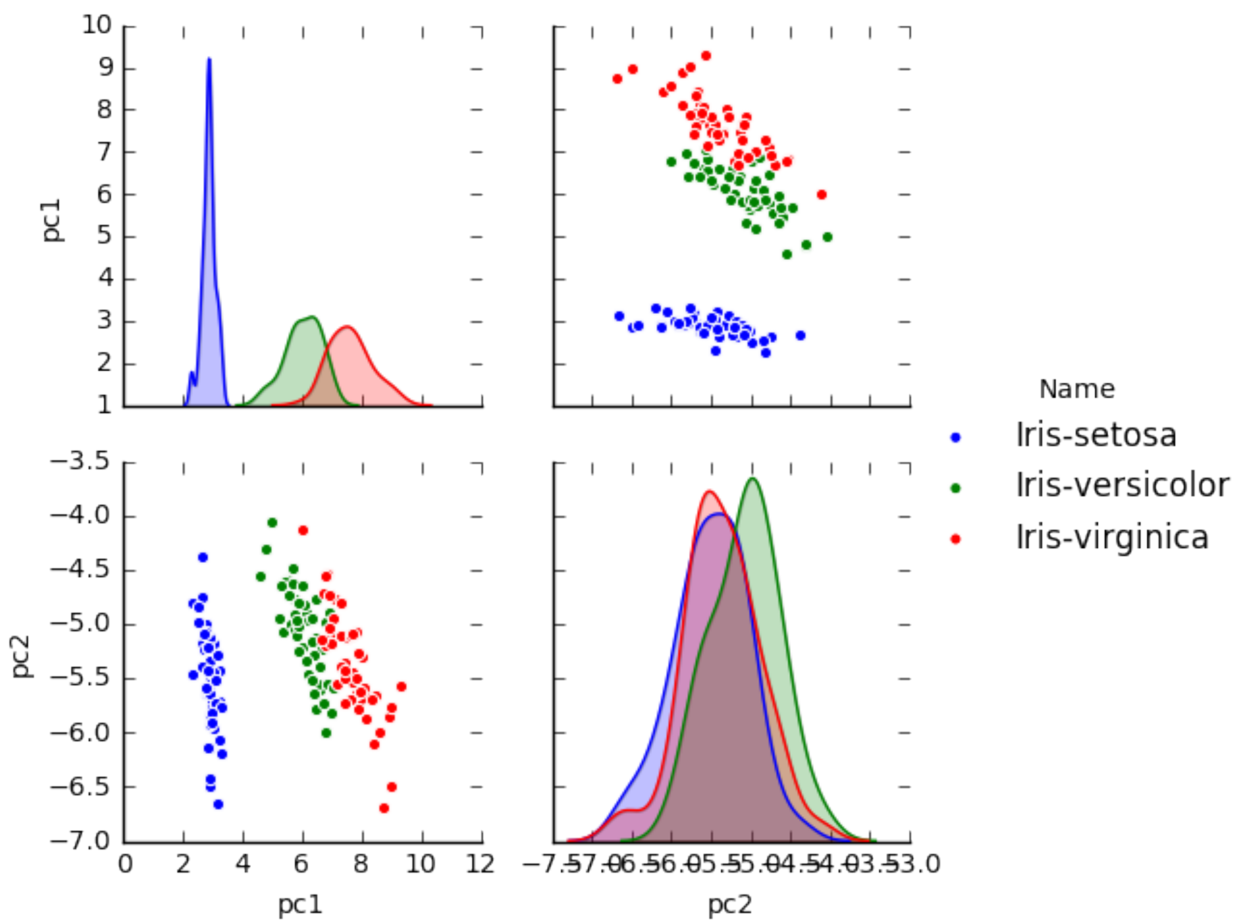
In [104]:
```
Xi_projected =np.dot(data,principle_components ) #projecting data on new
```

In [ ]:
```
**Question 6b:** Generate a pairplot (along with colors for the different
    the two newly generated features using PCA in the above step.
```

In [105]:
```
# using dictionary to store two principle directions as key and value pai
X_columns = {'pc1': Xi_projected[:,0].tolist(),'pc2' : Xi_projected[:,1].
```

In [106]:
```
projected_data_df = pd.DataFrame(X_columns)
```

In [109]:
```
import seaborn as sns
sns.pairplot(projected_data_df, hue='Name')
plt.show()
```



In [186]:
```
from sklearn.datasets import datasets
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=center
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call
last)
<ipython-input-186-6ba90c1fcfae> in <module>()
----> 1 from sklearn.datasets import datasets
      2 n_samples = 1500
      3 random_state = 42
      4 centers = [(-5, -5), (0, 0), (5, 5)]
      5 Blobs_X, Blobs_y = datasets.make_blobs(n_samples=n_samples,cen
ters=centers,random_state=random_state)

ImportError: cannot import name 'datasets'
```

In [ ]:
```
**Question 8a:** Using the code provided in the practice notebook for com
    write your own SVD function (U,S,V = mysvd(A)) to factorize the matri
```

In [129]:
```
def svd(A):
    u, s, v = np.linalg.svd(A)
    return u,s,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]])
```

In [130]:
```
u,s,v = svd(A)
```

In [131]:
```
u,s,v
```

Out[131]:
```
(array([[ -1.39539989e-01,  -1.03845336e-02,  -3.65334804e-03,
          -3.90500889e-03,   9.23979723e-01,  -3.55041310e-01,
          -2.44805513e-02],
        [ -4.18619967e-01,  -3.11536008e-02,  -1.09600441e-02,
          -1.17150267e-02,  -3.76462885e-01,  -8.20460416e-01,
           9.30043765e-02],
        [ -5.58159956e-01,  -4.15381344e-02,  -1.46133922e-02,
          -1.56200355e-02,  -2.62101885e-02,   2.07958652e-01,
          -8.01461624e-01],
        [ -6.97699945e-01,  -5.19226680e-02,  -1.82667402e-02,
          -1.95250444e-02,   6.20499371e-02,   3.96917590e-01,
           5.90262784e-01],
        [ -7.41460406e-02,   5.77757949e-01,   5.52248544e-01,
           5.96422386e-01,  -2.38787725e-16,   1.03373054e-16,
           2.01033999e-17],
        [ -3.49926251e-02,   7.42563145e-01,  -6.23435159e-02,
          -6.65949532e-01,   1.84625888e-16,  -4.31375036e-17,
          -1.49742032e-16],
        [ -1.53637450e-02,   3.30599399e-01,  -8.30935700e-01,
           4.47229086e-01,   6.09828651e-19,  -1.30207836e-16,
           5.51962618e-17]]),
 array([  1.23907772e+01,   9.86730009e+00,   1.35561282e+00,
          5.17051476e-01,   2.41592441e-16,   6.70536613e-18]),
 array([[ -5.74341650e-01,  -5.80325620e-01,  -5.74341650e-01,
          -4.05361801e-02,  -4.05361801e-02,  -1.41120107e-02],
        [ -5.36733664e-02,   4.87942348e-03,  -5.36733664e-02,
           6.77494984e-01,   6.77494984e-01,   2.76071774e-01],
        [ -1.37443928e-01,   2.69935331e-01,  -1.37443928e-01,
           1.73652236e-01,   1.73652236e-01,  -9.10518013e-01],
        [ -3.85175292e-01,   7.68331493e-01,  -3.85175292e-01,
          -9.59284412e-02,  -9.59284412e-02,   3.07477112e-01],
        [ -7.06854911e-01,   6.54438624e-16,   7.06854911e-01,
          -1.88715169e-02,   1.88715169e-02,   5.33376616e-17],
        [ -1.88715169e-02,   7.50729322e-17,   1.88715169e-02,
           7.06854911e-01,  -7.06854911e-01,  -4.22291381e-17]]))
```

In [ ]:
```
**Question 8c:** Perform SVD on iris dataset and visualize the proportion
    each spectral value. List the dimensions that captures less than 10%
```

In [133]:
```
import pandas as pd
iris_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/
```

In [134]:
```
data = iris_df.values[:,0:4]
data = data.astype(float) #converts data format from object to numeric
```

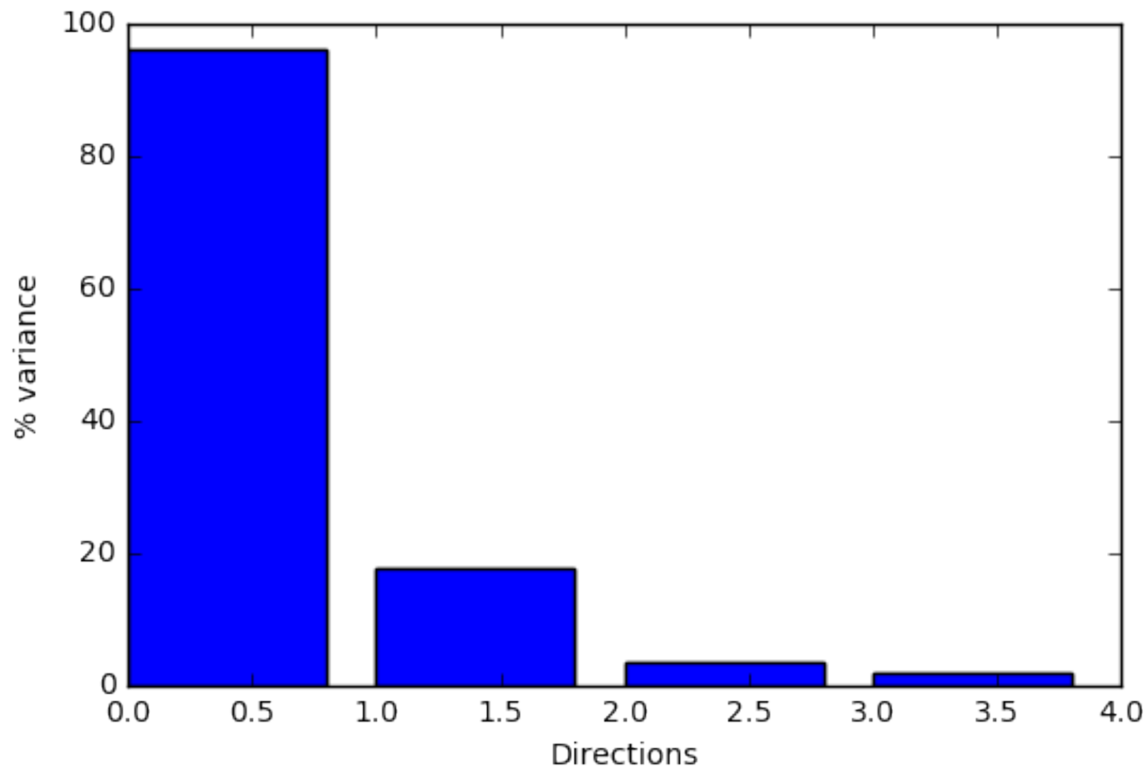
In [139]:
```
u_i,s_i,v_i = svd(data)
```

In [140]:
```
del_values = s_i
```

In [145]:
```
del_values
```

Out[145]:
```
array([ 95.95066751,  17.72295328,   3.46929666,   1.87891236])
```

In [148]:
```
import numpy as np
plt.bar(np.arange(4), del_values)
plt.xlabel('Directions')
plt.ylabel('% variance')
plt.show()
```



In [151]:
```
var_percentage = (s_i/sum(s_i))*100
```

In [152]:
```
var_percentage
```

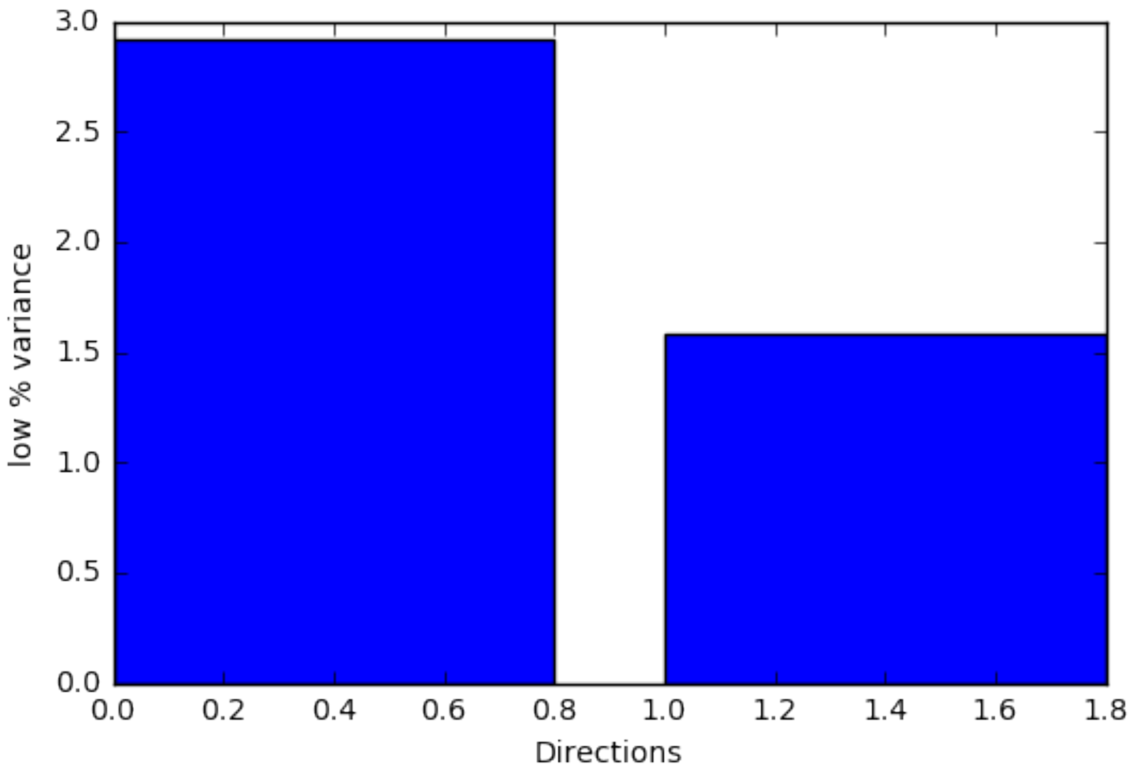
Out[152]:
```
array([ 80.61602452,  14.89050648,   2.91484064,   1.57862836])
```

In [154]:
```
low_percentage=var_percentage[2:]
```

In [155]:
```
low_percentage
```

Out[155]:
```
array([ 2.91484064,  1.57862836])
```

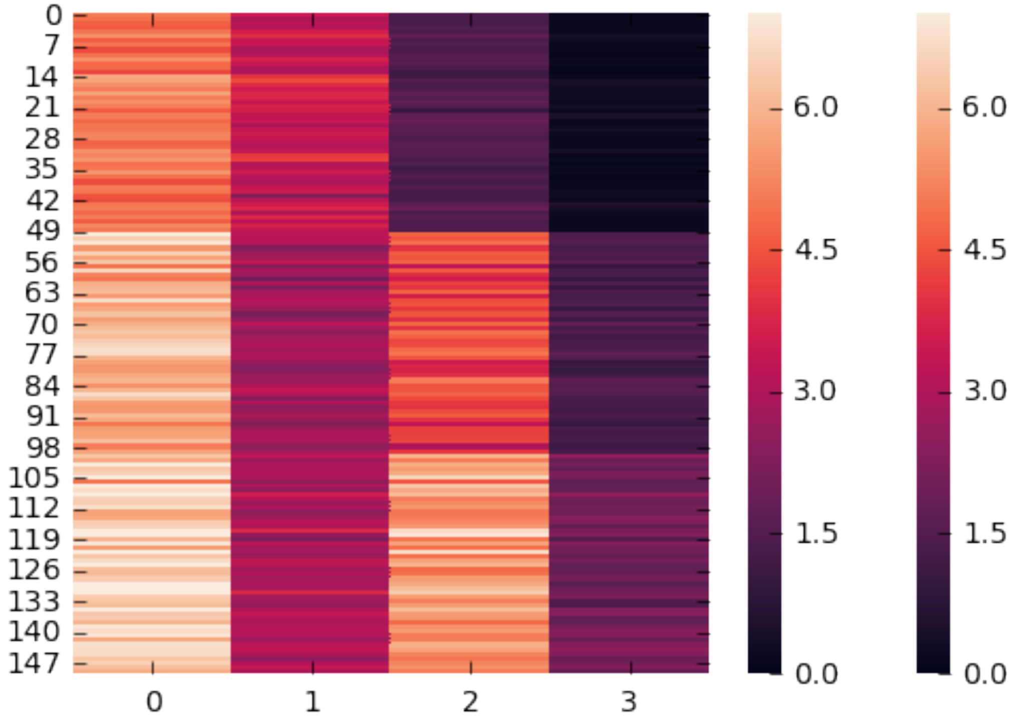
In [158]:
```
import numpy as np
plt.bar(np.arange(2), low_percentage)
plt.xlabel('Directions')
plt.ylabel('low % variance')
plt.show()
```



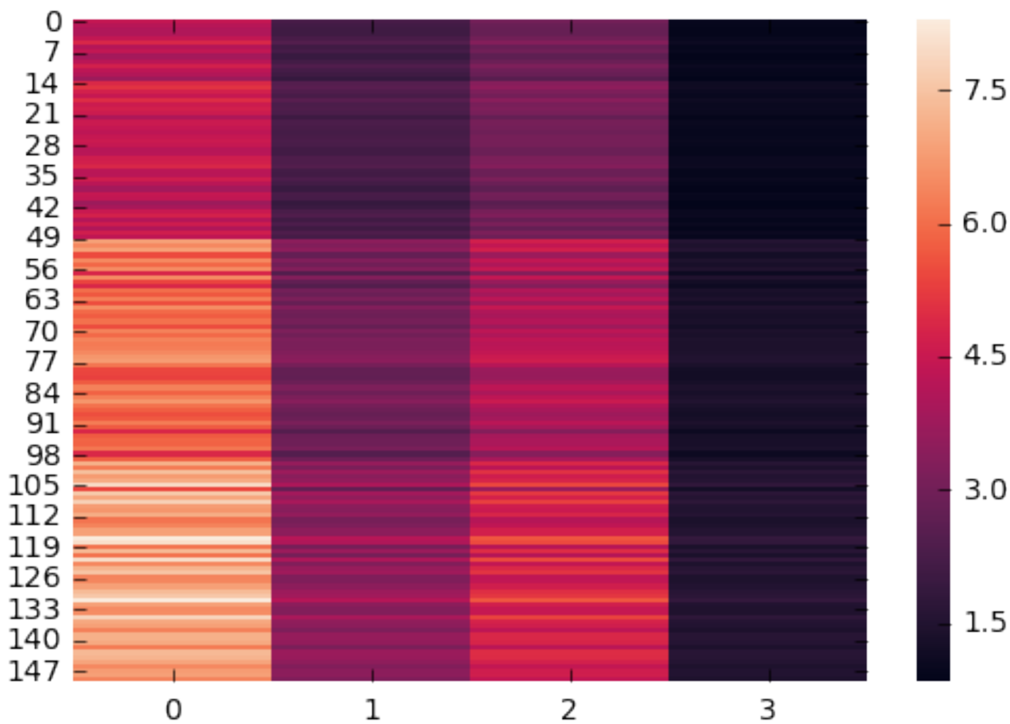
In [ ]:
```
Above two are the directions which capture least variance(less than 10)
```

In [ ]:
```
**Question 8d:** The heatmap of the full data is shown below. Plot all th
    matrices based on SVD.
```

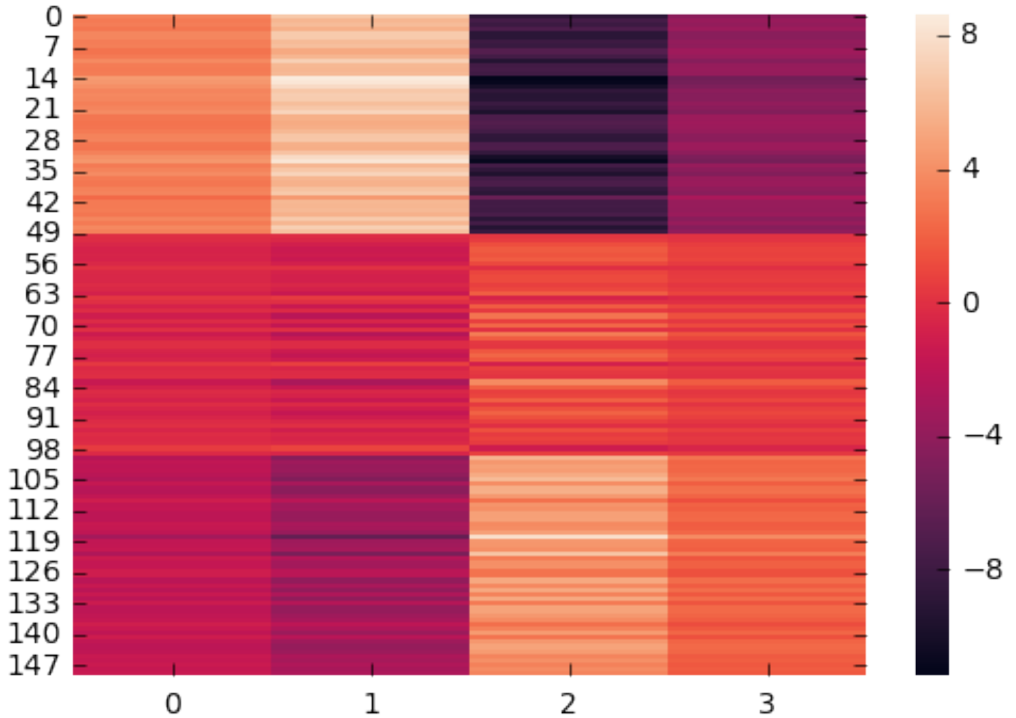
In [160]:
```
sns.heatmap(data,vmin=0,vmax=7)
plt.show()
```



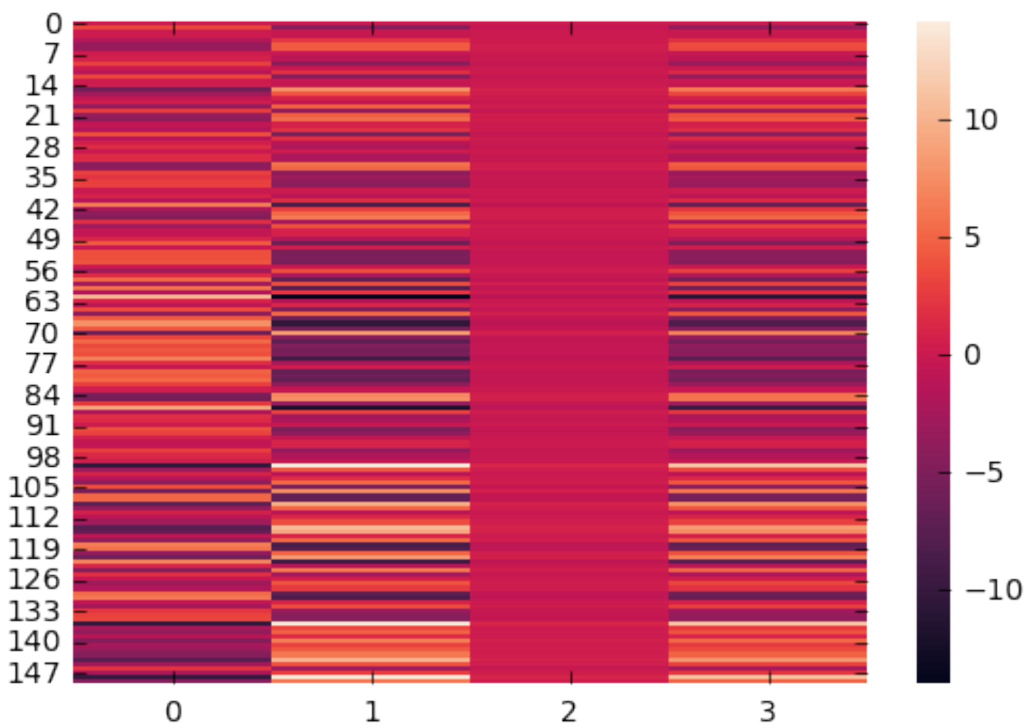
In [161]:
```
sns.heatmap(s_i[0]*np.outer(u_i[:,0],v_i[0,:]))
plt.show()
```



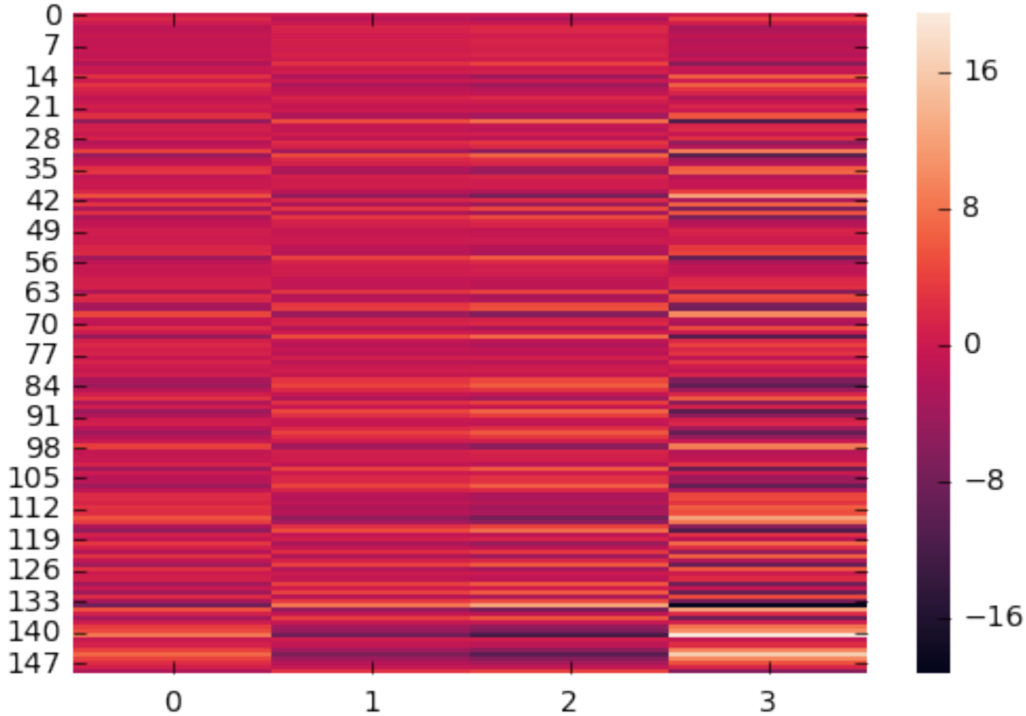
In [162]:
```
sns.heatmap(s_i[0]*np.outer(u_i[:,1],v_i[1,:]))
plt.show()
```



In [163]:
```
sns.heatmap(s_i[0]*np.outer(u_i[:,2],v_i[2,:]))
plt.show()
```



In [164]:
```
sns.heatmap(s_i[0]*np.outer(u_i[:,3],v_i[3,:]))
plt.show()
```



In [168]:
```
from sklearn.datasets import load_digits
digits = load_digits()
```

In [169]:
```
 digits.data.shape
```

Out[169]: (1797, 64)

In [170]:
```
digits.target
```

Out[170]: array([0, 1, 2, ..., 8, 9, 8])

In [171]:
```
Threes = np.where(digits.target==3)
Eights = np.where(digits.target==8)
[np.size(Threes), np.size(Eights)]
```

Out[171]: [183, 174]

In [172]:
```
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 [173]:
```
X
```

Out[173]:
```
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 [175]:
```
X.shape
```

Out[175]:
```
(357, 64)
```

In [177]:
```
y
```

Out[177]:
```
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.
```

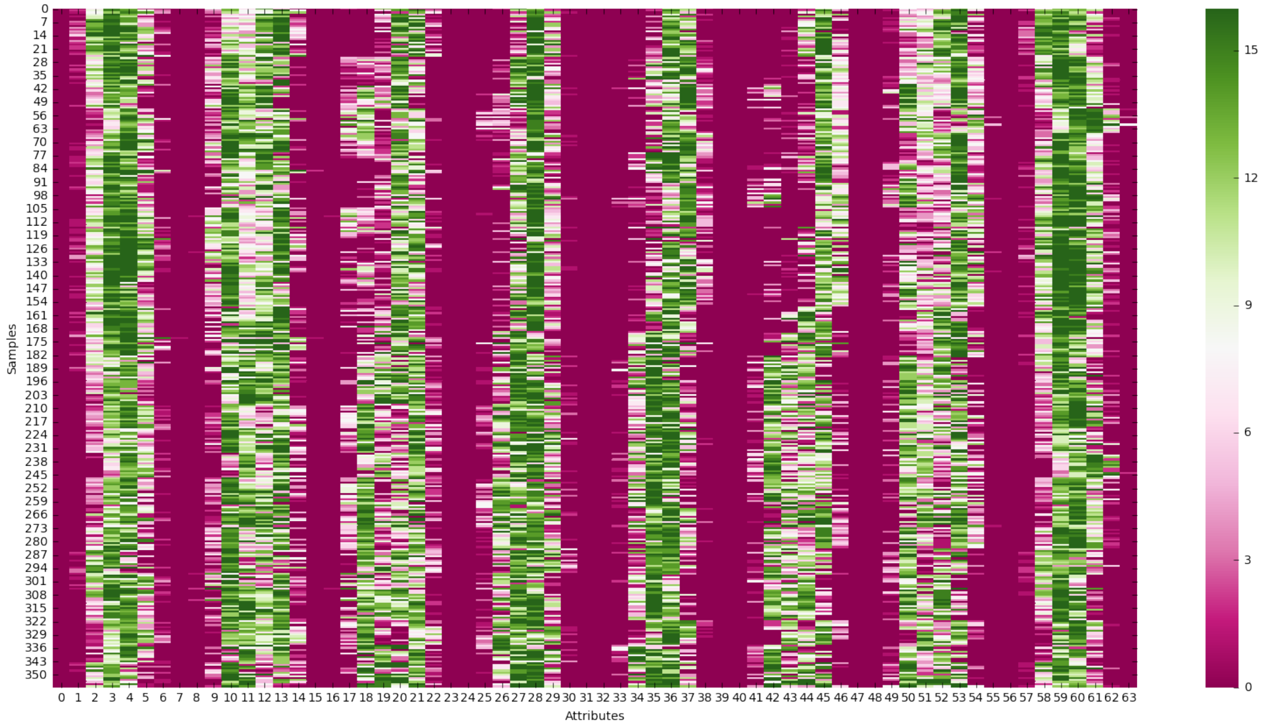
```
,
        8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.
,
        8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.
,
        8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.
,
        8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.,  8.
,
        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 [178]:
```
y.shape
```

Out[178]: (357,)

In [ ]:
```
**Question 9a:** Visually examine the following heatmap of the data X and
    separate the 3s from 8s. Also comment on (approximately) how many mis
    attribute is used for projection in LDA.
```

In [181]:
```
plt.figure(figsize=(20,10))
ax = sns.heatmap(X,cmap='PiYG')
ax.set(xlabel='Attributes', ylabel='Samples')
plt.show()
```



In [ ]:
```
42nd attribute is classifying 3s and 8s. There would be approximately 30
```

In [ ]:
```
**Question 9b:** Perform LDA on this data. Plot the heatmap of the projec
    points will be wrongly predicted based on this projection.
```

In [ ]:
```
The classes are almost seperated with appropriate seperations(as per heat
```