Libraries

```
import numpy as np
import h5py
import matplotlib.pyplot as plt
from scipy.ndimage.filters import convolve
from sklearn.cluster import KMeans
```

```
C:\ProgramData\Anaconda3\lib\site-packages\h5py\__init__.py:36:
FutureWarning: Conversion of the second argument of issubdtype from
`float` to `np.floating` is deprecated. In future, it will be treated
as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

Gausian Filtere and 2D Convolution

```
def gauss2D(shape=(3,3),sigma=0.5):
    m,n = [(ss-1.)/2. for ss in shape]
    y,x = np.ogrid[-m:m+1,-n:n+1]
    h = np.exp( -(x*x + y*y) / (2.*sigma*sigma) )
    h[ h < np.finfo(h.dtype).eps*h.max() ] = 0
    sumh = h.sum()
    if sumh != 0:
        h /= sumh
    return h


def conv2D(x,y,mode='same'):
    if not(mode == 'same'):
        raise Exception("Mode not supported")
    if (len(x.shape) < len(y.shape)):
        dim = x.shape
        for i in range(len(x.shape),len(y.shape)):
            dim = (1,) + dim
        x = x.reshape(dim)
    elif (len(y.shape) < len(x.shape)):
        dim = y.shape
        for i in range(len(y.shape),len(x.shape)):
            dim = (1,) + dim
        y = y.reshape(dim)
    origin = ()
    for i in range(len(x.shape)):
        if ( (x.shape[i] - y.shape[i]) % 2 == 0 and
             x.shape[i] > 1 and
             y.shape[i] > 1):
            origin = origin + (-1,)
        else:
            origin = origin + (0,)
    z = convolve(x,y, mode='constant', origin=origin)
    return z
```

Modified Fuzzy C-Means

```
def biasField(w, y, u, c, k, q):
    s = y.shape
    sum_num = np.zeros((s))
    sum_den = np.zeros((s))
    for i in range(k):
        sum_num += (u[:, :, i]**q)*c[i]
        sum_den += (u[:, :, i]**q)*(c[i]**2)
    num = conv2D(y*sum_num, w, 'same')
    den = conv2D(sum_den, w, 'same')
    return num/den


def cmeans(u, y, w, b, q, k):
    sum_num = conv2D(b, w, 'same')
    sum_den = conv2D(b**2, w, 'same')
    cmeans = np.zeros((k,1))
    for i in range(k):
```

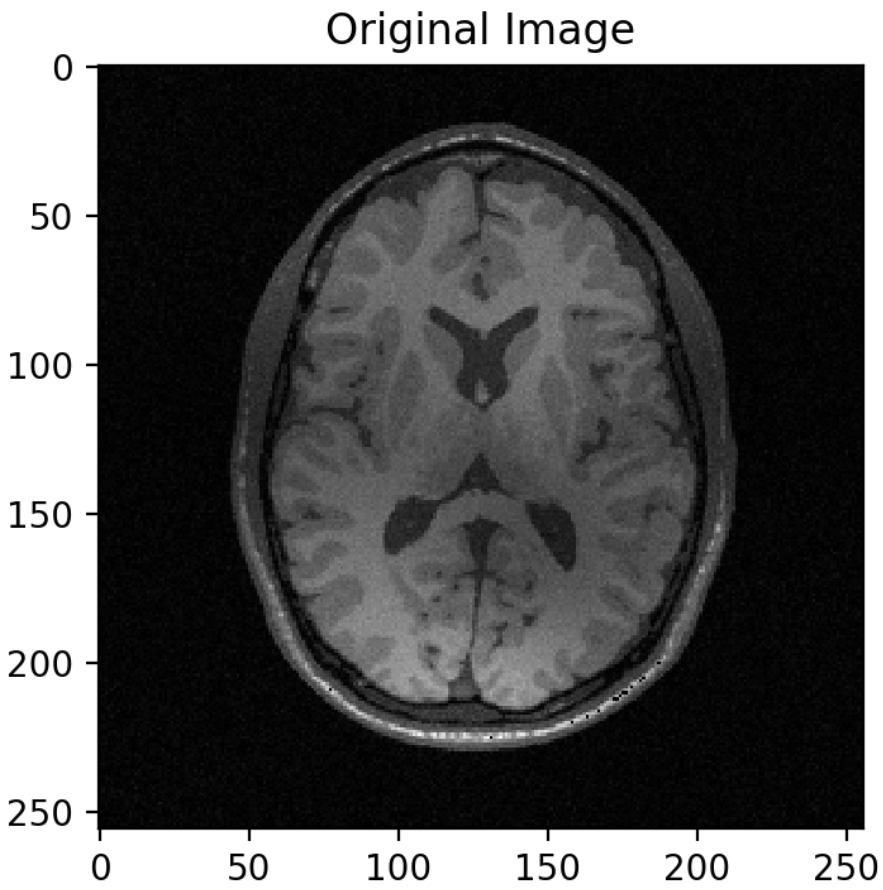
```
        cmeans[i] = np.sum((u[:, :, i]**q)*y*sum_num)/np.sum((u[:, :, i]**q)*sum_den)
    return cmeans


def membership(w, y, c, b, imageMask, k, q):
    s = y.shape
    u = np.zeros((s[0], s[1], k))
    d = np.zeros((s[0], s[1], k))
    sum_mask = np.sum(w)
    t1 = conv2D(b,w,'same')
    t2 = conv2D(b**2,w,'same')
    for i in range(k):
        d[:, :, i] = (y**2)*sum_mask - 2*c[i]*y*t1 + (c[i]**2)*t2
    d[d<0] = 0
    u = np.divide(1,d)**(1/(q-1))
    sum_u = np.nansum(u, 2)
    for i in range(k):
        u_new = u[:, :, i]
        u_new = u_new/sum_u
        u_new[np.where(imageMask == 0)] = 0
        u[:, :, i] = u_new
    return np.nan_to_num(u)


def objFun(y, w, c, b, u, q, k):
    s = y.shape
    d = np.zeros((s[0], s[1], k))
    sum_mask = np.sum(w)
    a1 = conv2D(b, w, 'same')
    a2 = conv2D(b**2, w, 'same')
    a3 = np.zeros((s))
    for i in range(k):
        d[:, :, i] = (y**2)*sum_mask - 2*c[i]*y*a1 + (c[i]**2)*a2
    for i in range(k):
        a3 += (u[:, :, i]**q)*d[:, :, i]
    return np.sum(conv2D(w, a3, 'same'))
```

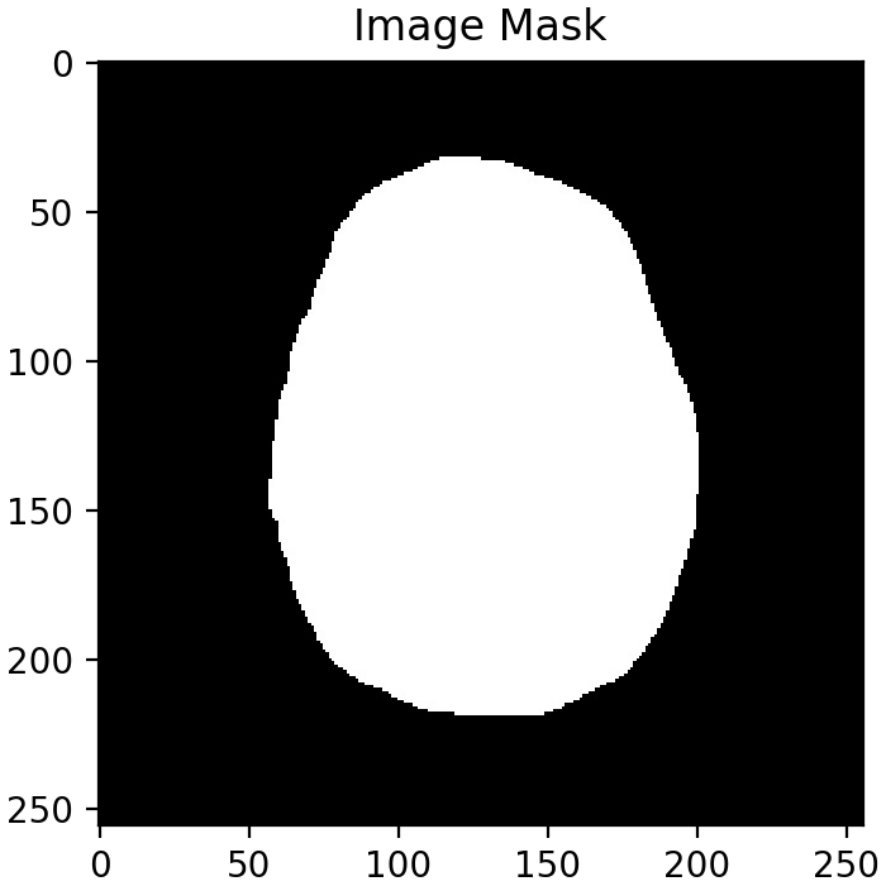
Class Memberships are initialized by kmeans

```
f = h5py.File('../data/assignmentSegmentBrain.mat','r')
#list(f.keys())
imageData = f.get('imageData')
imageMask = f.get('imageMask')
imageData = np.array(imageData)
imageData = imageData.T
imageMask = np.array(imageMask)
imageMask = imageMask.T
s = imageData.shape

#Plot Of Original Image and ImageMaks
plt.imshow(imageData, cmap=plt.cm.gray)
plt.title('Original Image')
plt.figure()

plt.imshow(imageMask, cmap=plt.cm.gray)
plt.title('Image Mask')
plt.figure()
```

```
<Figure size 1200x800 with 0 Axes>
```

```
<Figure size 1200x800 with 0 Axes>
```





Ititialized with simple k-means

```
k = 3
img = imageData[imageMask > 0]
#plt.imshow(img)
img = np.reshape(img, (21282,1))
kmeans = KMeans(n_clusters=k).fit(img)
labels = kmeans.cluster_centers_
```

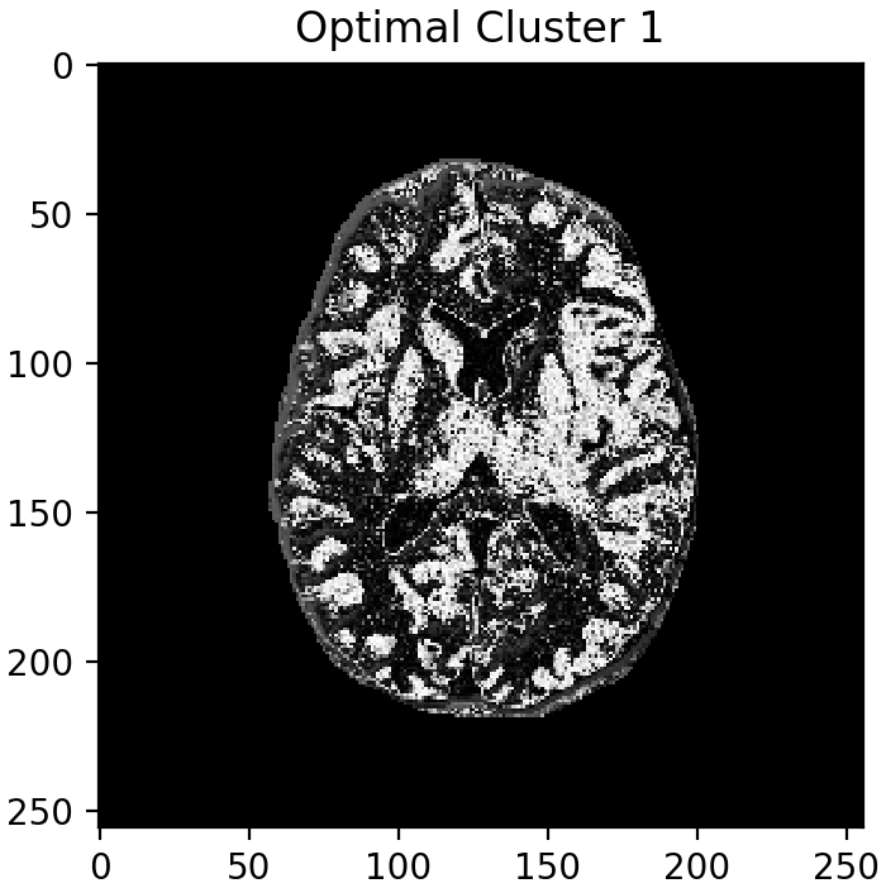
```
u0 = np.zeros((s[0], s[1], k))
for i in range(s[0]):
    for j in range(s[1]):
        if(imageMask[i, j] > 0):
            t = np.ones((k,1))*imageData[i, j]
            index = np.argmin(abs(t - labels))
            u0[i, j, index] = 1

#Parameters Used
q = 2.2
b0 = np.ones((s))*imageMask
window = 5;
w = gauss2D((10, 10))
maxIter = 30
J = np.zeros((maxIter, 1))


#Modified Fuzzy C-Means
y = imageData*imageMask
u = u0
c = labels
b = b0
for i in range(maxIter):
    u = membership(w, y, c, b, imageMask, k, q)
    c = cmeans(u, imageData, w, b, q, k)
    b = biasField(w, imageData, u, c, k, q)
    b[np.where(imageMask == 0)] = 0
    J[i] = objFun(imageData, w, c, b, u, q, k)
    print(J[i])

print('Initial Estimate of cluster centers is : ', kmeans.cluster_centers_)
print('Optimal Estimate of cluster centers is : ', c)
```

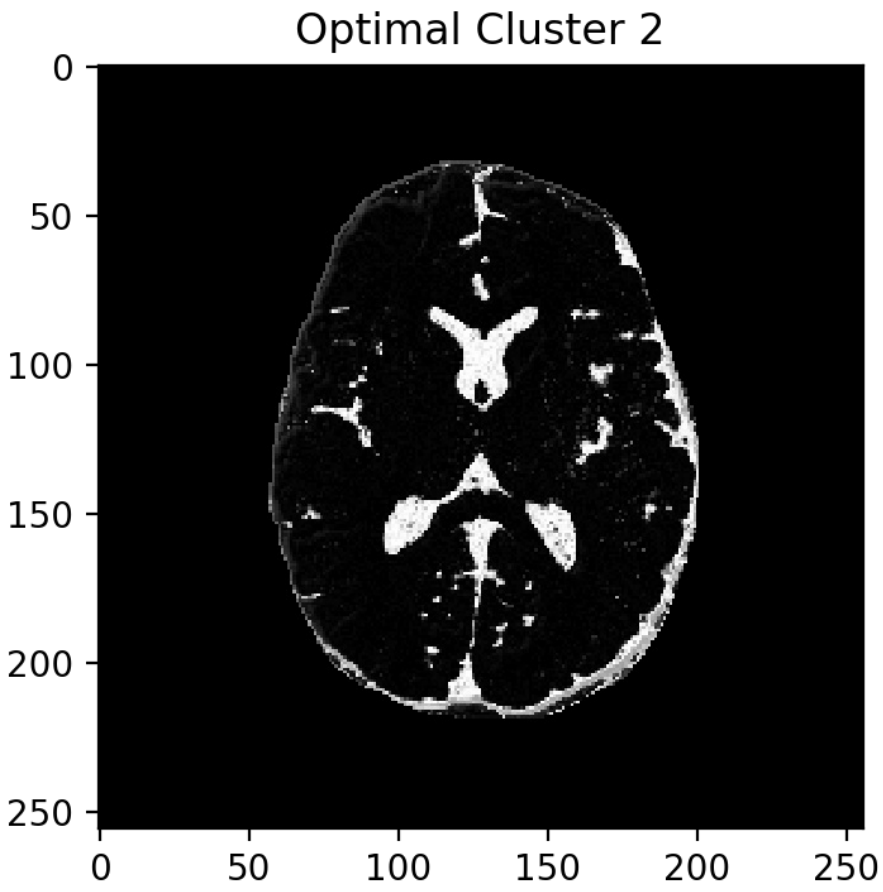
```
C:\ProgramData\Anaconda3\Scripts\pypublish:32: RuntimeWarning: divide
by zero encountered in true_divide
C:\ProgramData\Anaconda3\Scripts\pypublish:36: RuntimeWarning: invalid
value encountered in true_divide
C:\ProgramData\Anaconda3\Scripts\pypublish:10: RuntimeWarning: invalid
value encountered in true_divide
[0.12946719]
[0.08569208]
[0.07949667]
[0.07621052]
[0.07370332]
[0.07151615]
[0.06957778]
[0.06785974]
[0.06632137]
[0.06499771]
[0.0640387]
[0.06325562]
[0.06261766]
[0.06212086]
[0.06173402]
[0.06142854]
[0.06119006]
[0.06100791]
[0.06088098]
[0.06080026]
[0.06074379]
[0.06070257]
[0.06067099]
[0.06064827]
[0.06063536]
[0.06063063]
[0.06063123]
[0.06063427]
[0.06063762]
[0.06064012]
Initial Estimate of cluster centers is :  [[0.4534555 ]
 [0.22457567]
 [0.6345866 ]]
Optimal Estimate of cluster centers is :  [[0.52677315]
 [0.28344989]
 [0.63334299]]
```

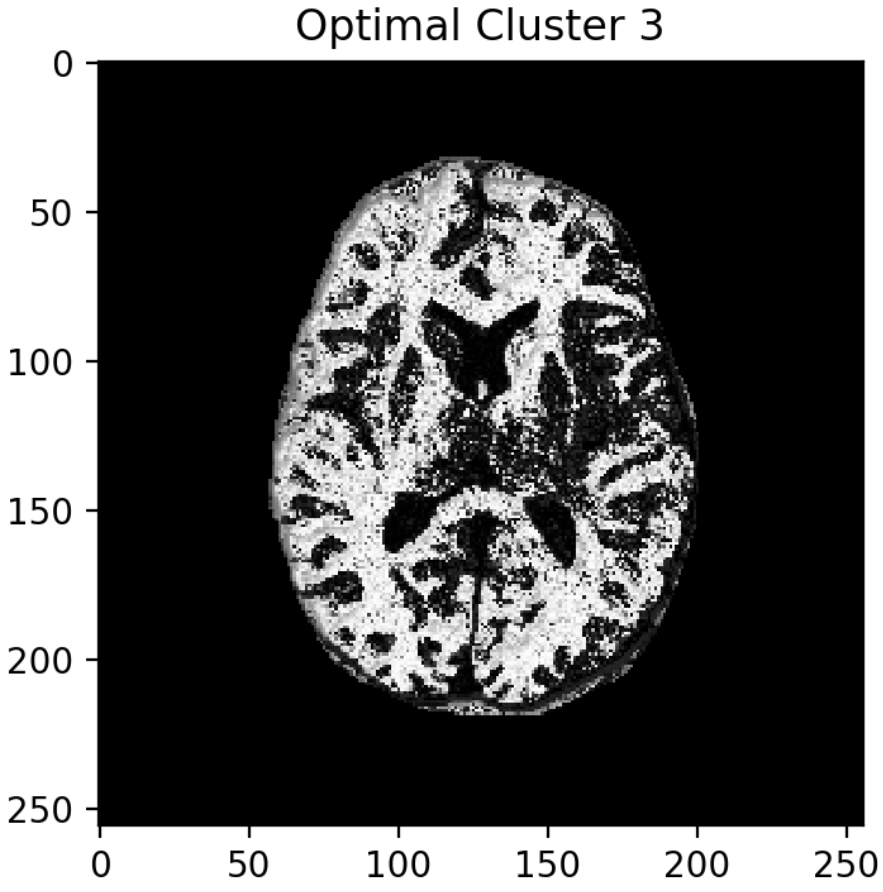
Plot of segmented brain sections

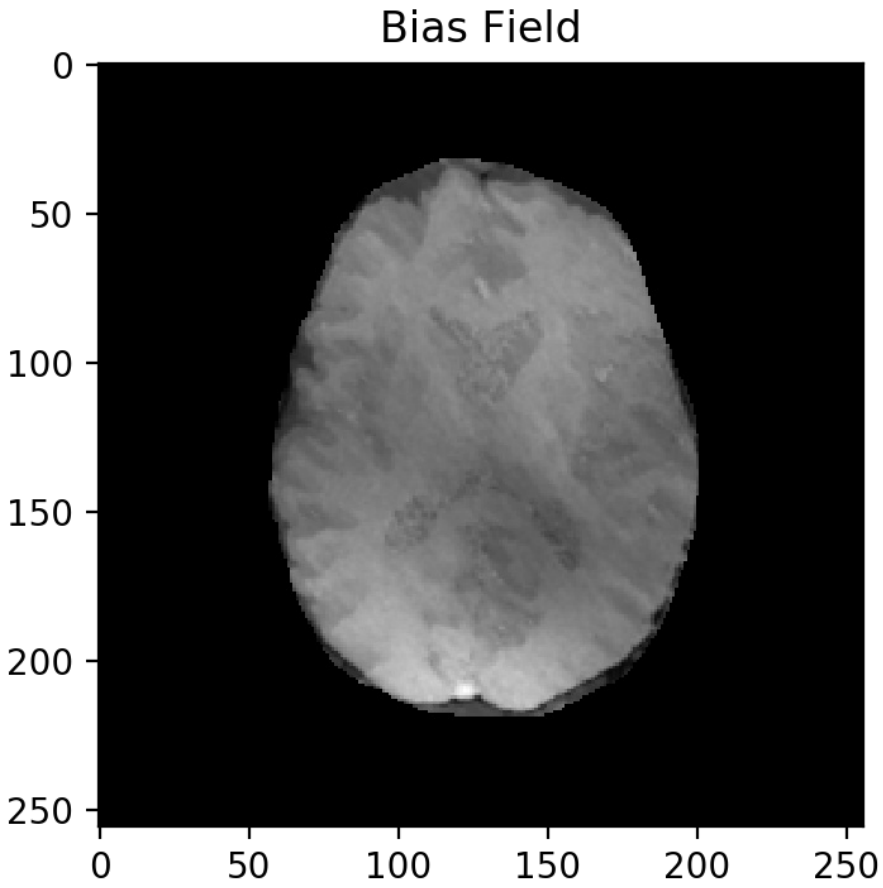
```
plt.imshow(u[:, :, 0], cmap=plt.cm.gray)
plt.title('Optimal Cluster 1')
plt.figure()

plt.imshow(u[:, :, 1], cmap=plt.cm.gray)
plt.title('Optimal Cluster 2')
plt.figure()

plt.imshow(u[:, :, 2], cmap=plt.cm.gray)
plt.title('Optimal Cluster 3')

plt.figure()
plt.imshow(b, cmap=plt.cm.gray)
plt.title('Bias Field')
```

```
Text(0.5,1,'Bias Field')
```




Optimal Cluster 2
0
50
100
150
200
250
0
50
100
150
200
250

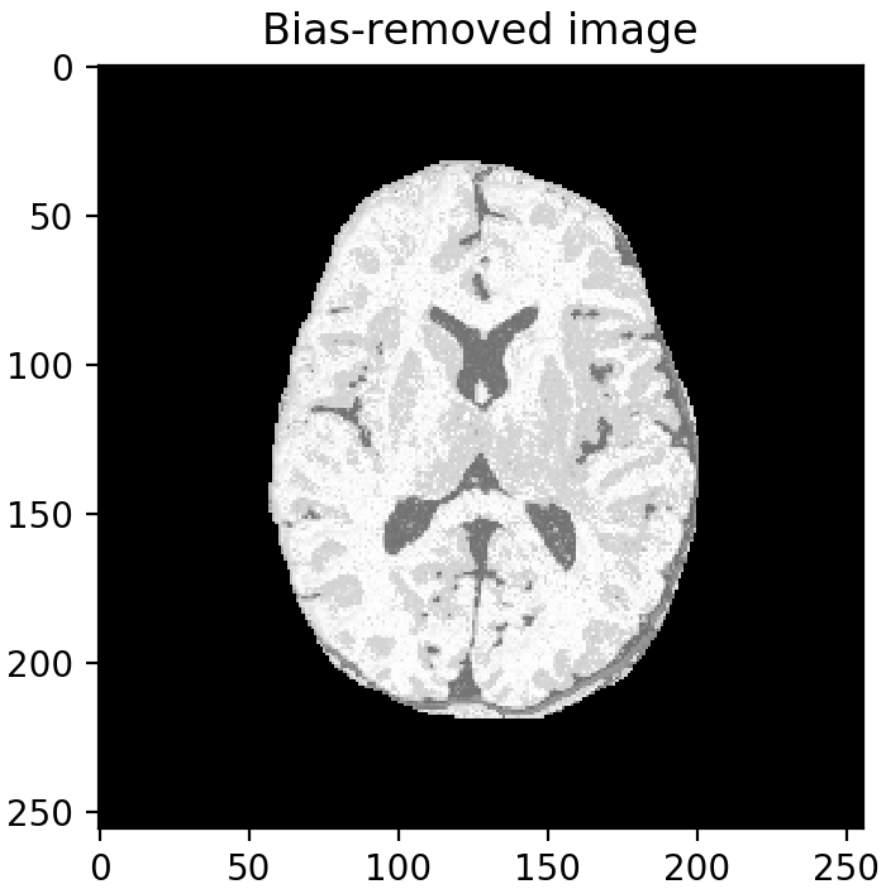

Optimal Cluster 3
0
50
100
150
200
250
0
50
100
150
200
250



Bias Removed Image

```
nb = np.zeros((s));
for i in range(k):
    nb +=  u[:,:,i]*c[i]
nb = nb*imageMask

plt.figure()
plt.imshow(nb, cmap='gray')
plt.title('Bias-removed image')
```

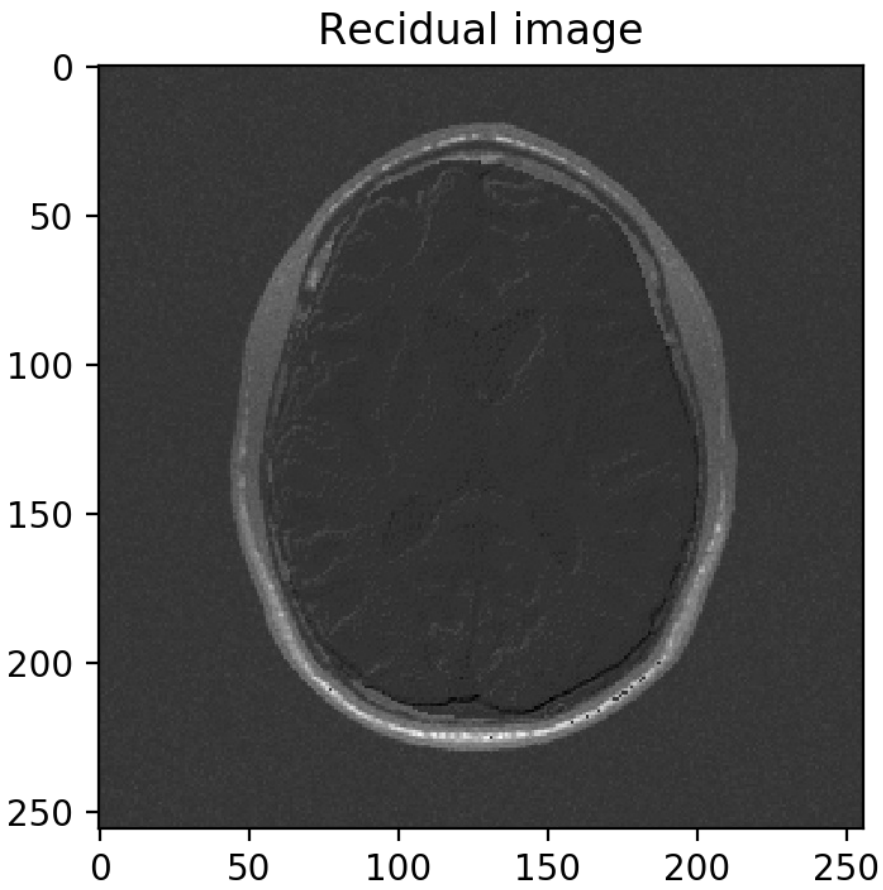
```
Text(0.5,1,'Bias-removed image')
```



Recidual Image

```
eecidual = imageData - nb*b
plt.figure()
plt.imshow(eecidual, cmap=plt.cm.gray)
plt.title('Recidual image')
```

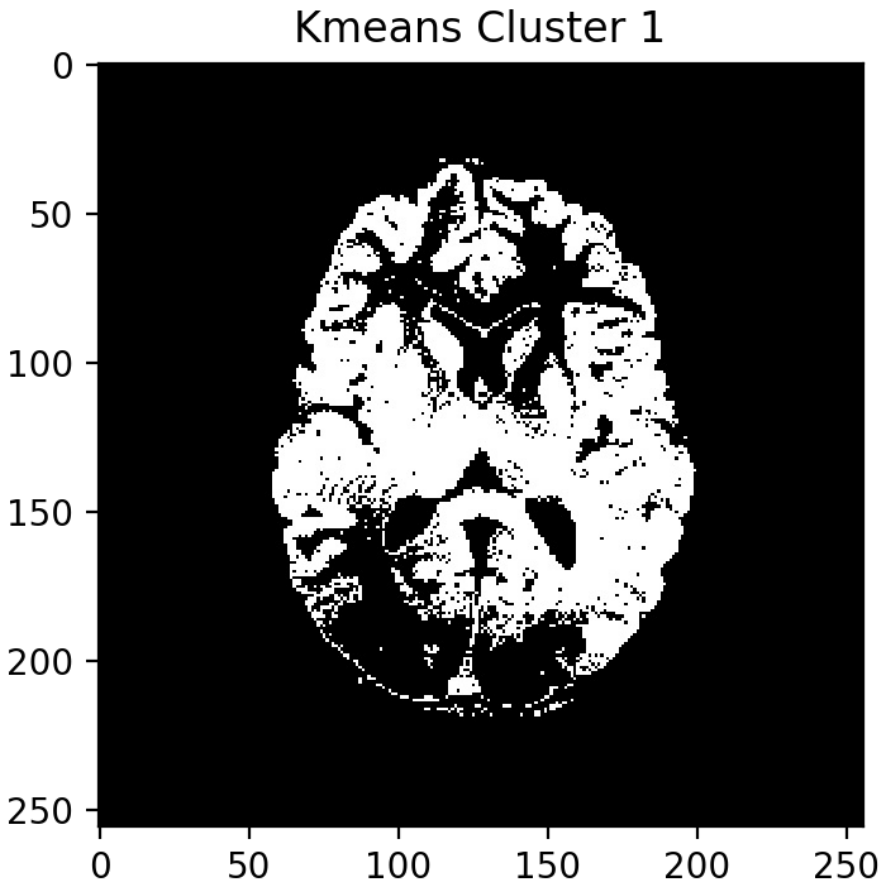
```
Text(0.5,1,'Recidual image')
```

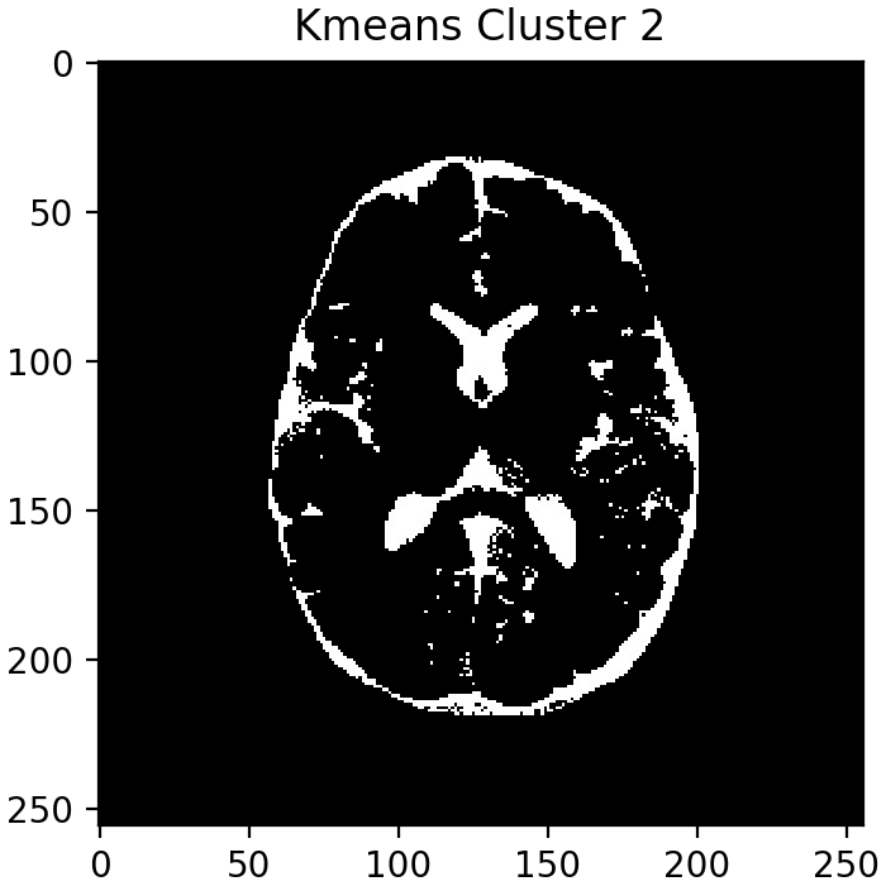


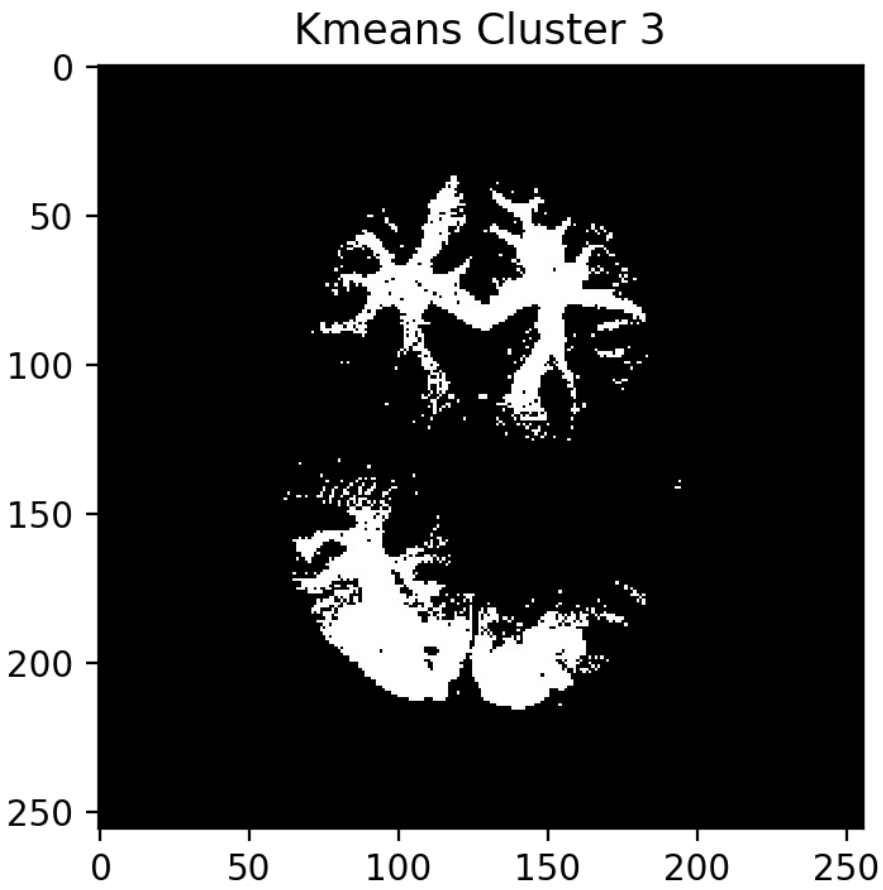
neighborhodd mask

```
plt.figure()
plt.imshow(w, extent=[0, 1, 0, 1])
plt.title('Neighborhodd mask')



 #%%
#Segmentation based on kmeans
plt.imshow(u0[:, :, 0], cmap=plt.cm.gray)
plt.title('Kmeans Cluster 1')
plt.figure()

plt.imshow(u0[:, :, 1], cmap=plt.cm.gray)
plt.title('Kmeans Cluster 2')
plt.figure()

plt.imshow(u0[:, :, 2], cmap=plt.cm.gray)
plt.title('Kmeans Cluster 3')
```

```
Text(0.5,1,'Kmeans Cluster 3')
```


Kmeans Cluster 1
0
50
100
150
200
250
0
50
100
150
200
250


Kmeans Cluster 2
0
50
100
150
200
250
0
50
100
150
200
250



Published from q1.py using Pweave 0.30.3 on 21-03-2019.