Exercise 3

1. Clustering The data for this portion of the assignment will be the Iris flower dataset. The dataset consists of 50 samples from each of three species of Iris, with four features measured from each sample.scikit-learn provides a function to load the dataset (no download required).

The Idea: Choosing k for k-means

Your objective here will be to assess the performance of k-means clustering on the Iris dataset. Recall that the number of clusters, k, is an input parameter to the k-means algorithm. A variety of measurements are available for estimating the optimal value of k. For this assignment, you will look at the sum of squared deviation (SSQ) and the gap statistic. Both of these criteria make use of the intuition that k-means tries to minimize variance (the distance or deviation of each point from each of the k clusters) by iteratively assigning points to their nearest clusters.

Choosing k with SSQ

The SSQ criterion is a direct application of the intuition that k-means tries to minimize variance. Recall that the SSQ criterion sweeps over a range of possible k values, with each value of k associated with a degree of deviation (the distance of each point from each of the k clusters). These deviations can be squared and summed to arrive at the “sum of squared deviation” (SSQ) for each value of k. Larger values of k are expected to continue to reduce the SSQ (because there are more clusters for points to be near, reducing their deviation). However, one could expect a leveling-off in the SSQ once the value of k exceeds the true number of clusters, as this would result in true clusters (that is, clusters actually present in the data) being separated. If, then, one plots the SSQ over a range of k values, this leveling-off point may produce a noticeable “elbow” in the plot. By this criterion, the estimated optimal value of k is that which occurs at this elbow point. While simple, the difficulty with this criterion is that often the elbow point is not distinctive or well-defined.

What to Provide

Your output should contain the following:

The SSQs computed fork values between 1 and 10 (inclusive). There should be one plot corresponding to the SSQs. The gap statistics computed fork values between 1 and 10 (inclusive). There should be two plots corresponding to the gap statistics. Given this output, respond to the following questions:

Where did you estimate the elbow point to be (between what values of k)? What value of k was typically estimated as optimal by the gap statistic? To adequately answer this question, consider generating both measures several times, as there may be some amount of variation in the value of k that they each estimate as optimal. How close are the estimates generated by the elbow point and gap statistic to the number of species of Iris represented in the dataset? Assuming we are trying to generate one cluster for each Iris species represented in the dataset, does one measure seem to be a consistently better criterion for choosing the value of k than the other? Why or why not?

In [16]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns
from sklearn.datasets import load_iris
print("Import Complete")
```

```
Import Complete
```

1. SSQ

The SSQs computed fork values between 1 and 10 inclusive. There should be one plot corresponding to the SSQs.

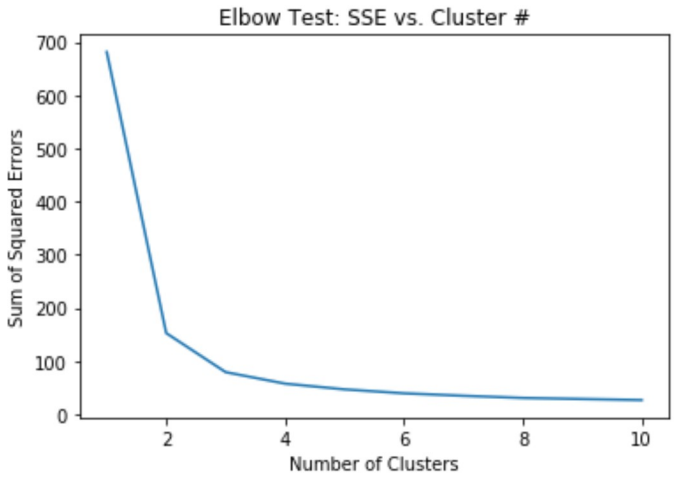
In [7]:
```
# function to compute SSE by testing different values of k clusters.

def compute_SSE(data, n):
    sse = {}
    for k in range(1,n+1):
        km = KMeans(n_clusters=k, max_iter=1000).fit(data)
        sse[k] = km.inertia_
    return(sse)
```

In [9]:
```
def format_SSE(sse_values):
    print('Clusters and their SSE value')
    print('-'*30)
    for k,v in sse_values.items():
        print('Clusters:{} \tSSE: {}'.format(k,v))
```

In [20]:
```
# function to compute elbow graph

def calculate_elbow(sse):
    plt.figure()
    plt.plot(list(sse.keys()), list(sse.values()))
    plt.xlabel('Number of Clusters')
    plt.ylabel('Sum of Squared Errors')
    plt.title('Elbow Test: SSE vs. Cluster #')
```

In [21]:
```
# Main Function

# load dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns = iris['feature_names'])
k = 10

sse_value = compute_SSE(df,k)
format_SSE(sse_value)
calculate_elbow(sse_value)
```

```
Clusters and their SSE value
------------------------------
Clusters:1      SSE: 681.3706
Clusters:2      SSE: 152.34795176035792
Clusters:3      SSE: 78.85144142614601
Clusters:4      SSE: 57.228473214285714
Clusters:5      SSE: 46.44618205128205
Clusters:6      SSE: 39.03998724608725
Clusters:7      SSE: 34.40900974025974
Clusters:8      SSE: 30.33777154862682
Clusters:9      SSE: 28.307124515835046
Clusters:10     SSE: 26.32504303213127
```



GAP Statistic

In [22]:
```
# gap.py
# (c) 2013 Mikael Vejdemo-Johansson
# BSD License
#
# SciPy function to compute the gap statistic for evaluating k-means clustering.
# Gap statistic defined in
# Tibshirani, Walther, Hastie:
#  Estimating the number of clusters in a data set via the gap statistic
#  J. R. Statist. Soc. B (2001) 63, Part 2, pp 411-423
```

In [30]:
```
iris = load_iris()
df = pd.DataFrame(iris.data)
data = df[[0,1,2,3]].values
```

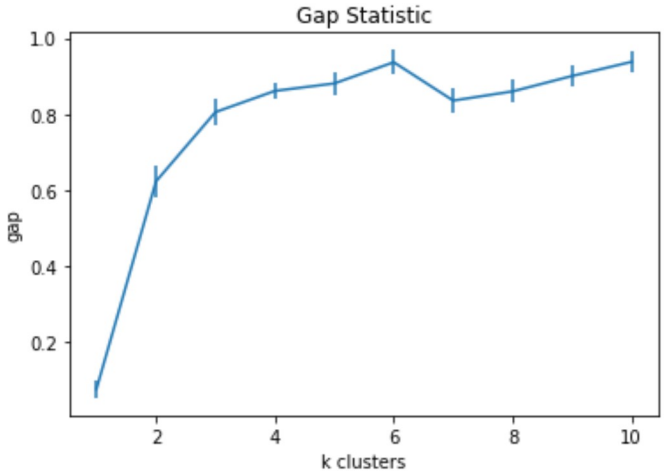
In [42]:
```
import scipy
import scipy.cluster.vq
import scipy.spatial.distance
dst = scipy.spatial.distance.euclidean
```

In [46]:
```
def gap(data, refs=None, nrefs=30, ks=range(1,11)):

    ### Compute the Gap statistic for an nxm dataset in data. Either give a precomp
uted set of reference distributions in refs as an (n,m,k) scipy array,
    ### or state the number k of reference distributions in nrefs for automatic gen
eration with a uniformed distribution within the bounding box of data.
    ### Give the list of k-values for which you want to compute the statistic in k
s.

    shape = data.shape
    if refs==None:
        tops = data.max(axis=0)
        bots = data.min(axis=0)
        dists = scipy.matrix(scipy.diag(tops-bots))

        rands = scipy.random.random_sample(size=(shape[0],shape[1],nrefs))
        for i in range(nrefs):
            rands[:,:,i] = rands[:,:,i]*dists+bots
    else:
        rands = refs

    gaps = scipy.zeros((len(ks),))
    errors = scipy.zeros((len(ks),))
    labels = dict((el,[]) for el in ks)
    for (i,k) in enumerate(ks):
        (kmc,kml) = scipy.cluster.vq.kmeans2(data, k)
        disp = sum([dst(data[m,:],kmc[kml[m],:]) for m in range(shape[0])])
        labels[k] = kml

        refdisps = scipy.zeros((rands.shape[2],))
        for j in range(rands.shape[2]):
            (kmc,kml) = scipy.cluster.vq.kmeans2(rands[:,:,j], k)
            refdisps[j] = sum([dst(rands[m,:,j],kmc[kml[m],:]) for m in range(shape
[0])])
            # calculate gaps
        gaps[i] = scipy.mean(scipy.log(refdisps))-scipy.log(disp)
        # calculate error
        errors[i] = scipy.sqrt(sum(((scipy.log(refdisp)-scipy.mean(scipy.log(refdis
ps)))**2) \
                                   for refdisp in refdisps)/float(nrefs)) * scipy.sqrt
(1+1/nrefs)
    return gaps, labels, errors
```

In [47]:
```
## function to plot gap statistic

def plot_gap_stat(gaps, errors):
    xval = range(1,len(gaps)+1)
    yval = gaps
    plt.errorbar(xval,yval, xerr=None, yerr=errors)
    plt.xlabel('k clusters')
    plt.ylabel('gap')
    plt.title('Gap Statistic')
    plt.show()
```

In [48]:
```
gaps, labels, errors = gap(data, refs=None, nrefs=30, ks=range(1,11))

plot_gap_stat(gaps, errors)
```

```
C:\Users\jfrui\Anaconda3\lib\site-packages\scipy\cluster\vq.py:579: UserWarning:
One of the clusters is empty. Re-run kmeans with a different initialization.
  warnings.warn("One of the clusters is empty. "
```

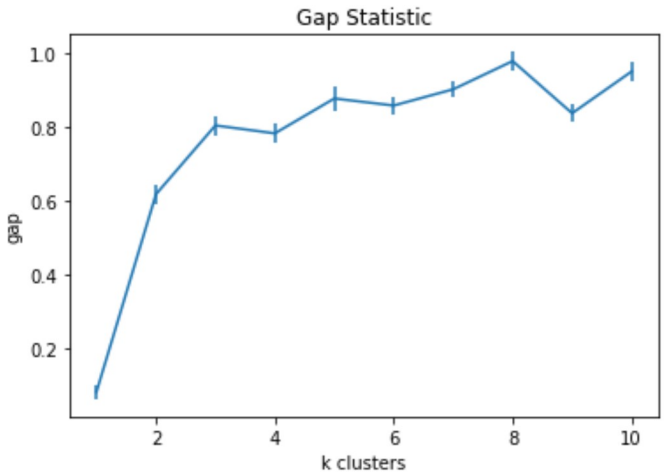


1. Where did you estimate the elbow point to be (between what values of k)? What value of k was typically estimated as optimal by the gap statistic? To adequately answer this question, consider generating both measures several times, as there may be some amount of variation in the value of k that they each estimate as optimal.

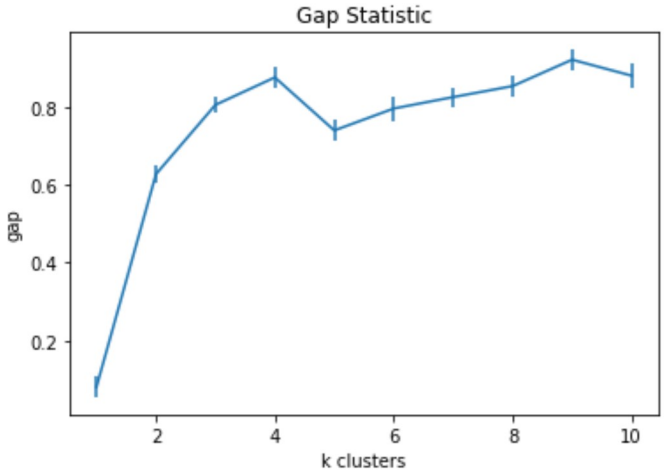
I estimated the elbow point to be 3 based on where the elbow turns. The gap statistics supports this estimation because its graph does not dip below 3 clusters.

In [51]:
```
## Gap Statistic Run Several Times
```

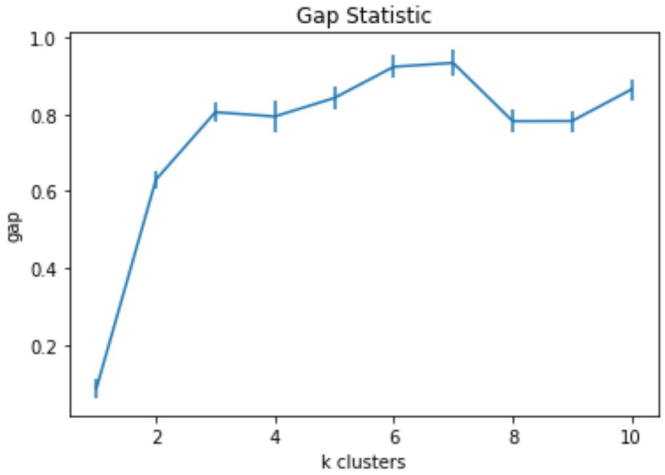
In [49]:
```
gaps, labels, errors = gap(data, refs=None, nrefs=30, ks=range(1,11))

plot_gap_stat(gaps, errors)
```



In [50]:
```
gaps, labels, errors = gap(data, refs=None, nrefs=30, ks=range(1,11))

plot_gap_stat(gaps, errors)
```

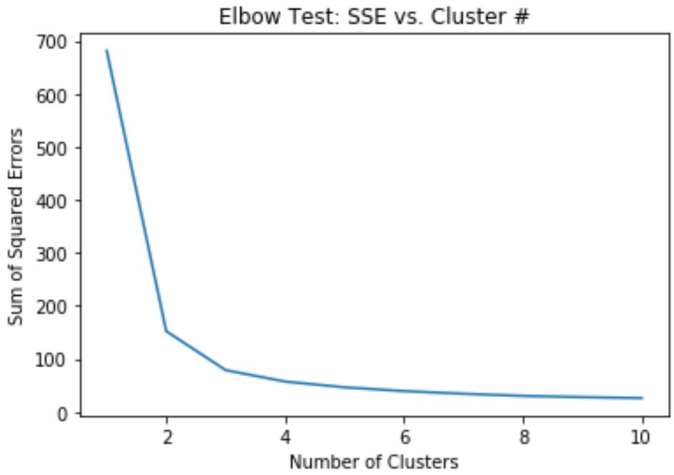


In [52]:
```
gaps, labels, errors = gap(data, refs=None, nrefs=30, ks=range(1,11))

plot_gap_stat(gaps, errors)
```



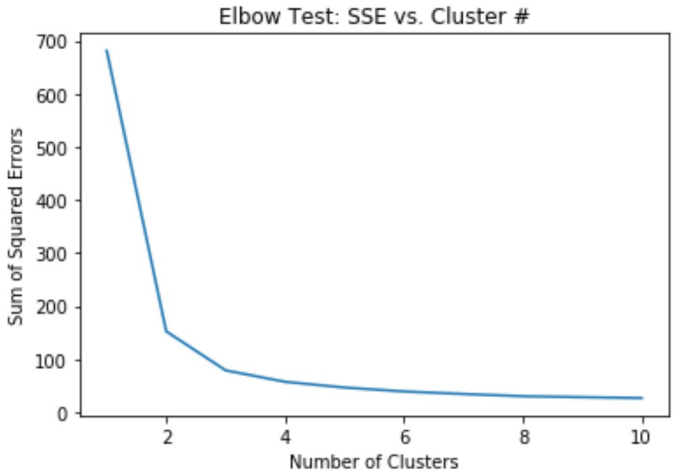
In [53]:
```
## Elbow point run several times
sse_value = compute_SSE(df,k)
format_SSE(sse_value)
calculate_elbow(sse_value)
```

```
Clusters and their SSE value
-----------------------------
Clusters:1      SSE: 681.3706
Clusters:2      SSE: 152.34795176035792
Clusters:3      SSE: 78.85144142614601
Clusters:4      SSE: 57.25600931571815
Clusters:5      SSE: 46.44618205128205
Clusters:6      SSE: 39.29957955377955
Clusters:7      SSE: 34.29822966507177
Clusters:8      SSE: 30.232138210946264
Clusters:9      SSE: 27.8453000683527
Clusters:10     SSE: 26.01457853421089
```



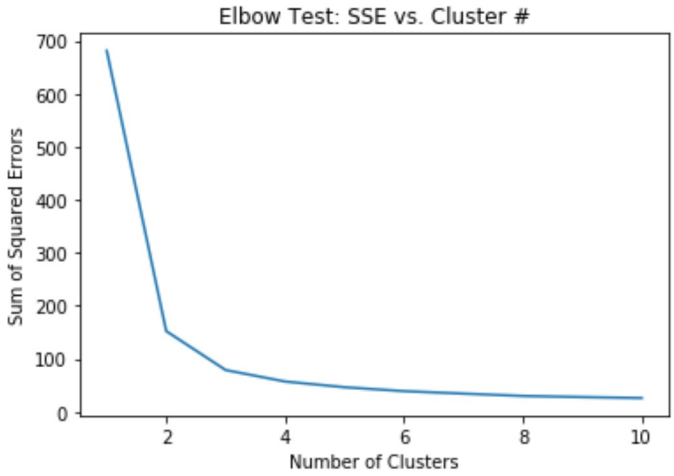
In [54]:
```
sse_value = compute_SSE(df,k)
format_SSE(sse_value)
calculate_elbow(sse_value)
```

```
Clusters and their SSE value
-----------------------------
Clusters:1      SSE: 681.3706
Clusters:2      SSE: 152.34795176035792
Clusters:3      SSE: 78.85144142614601
Clusters:4      SSE: 57.25600931571815
Clusters:5      SSE: 46.44618205128205
Clusters:6      SSE: 39.03998724608725
Clusters:7      SSE: 34.29822966507177
Clusters:8      SSE: 30.093465689916496
Clusters:9      SSE: 28.249994356520673
Clusters:10     SSE: 26.612124515835042
```



In [55]:
```
sse_value = compute_SSE(df,k)
format_SSE(sse_value)
calculate_elbow(sse_value)
```

```
Clusters and their SSE value
-----------------------------
Clusters:1      SSE: 681.3706
Clusters:2      SSE: 152.34795176035792
Clusters:3      SSE: 78.85144142614601
Clusters:4      SSE: 57.228473214285714
Clusters:5      SSE: 46.44618205128205
Clusters:6      SSE: 39.03998724608725
Clusters:7      SSE: 34.40900974025974
Clusters:8      SSE: 29.990426406926414
Clusters:9      SSE: 27.861741540115112
Clusters:10     SSE: 26.00045417010638
```



1. How close are the estimates generated by the elbow point and gap statistic to the number of species of Iris represented in the dataset? Assuming we are trying to generate one cluster for each Iris species represented in the dataset, does one measure seem to be a consistently better criterion for choosing the value of k than the other? Why or why not?

The iris dataset contains 3 species of iris. The gap statistic and elbow point both predict 3 clusters in the data. Though there is no difference in the output of the two methods, my preference would be use the elbow method because it is easier to implement. The Gap statistic would serve as a backup for more complex situations.

In [ ]: