Clustering methods

In [44]:

```
import pandas as pd
import numpy as np
import matplotlib as mpl
from matplotlib import pyplot as plt
import seaborn as sns

from sklearn.preprocessing import scale
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans

from scipy.cluster import hierarchy

%matplotlib inline
plt.style.use('seaborn-white')
```

10.5.1 K-Means Clustering

In [45]:

```
# Generate data
np.random.seed(2)
X = np.random.standard_normal((50,2))
X[:25,0] = X[:25,0]+3
X[:25,1] = X[:25,1]-4
```

K = 2

In [46]:

```
km1 = KMeans(n_clusters=2, n_init=20)
km1.fit(X)
```

Out[46]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means+
+', max_iter=300,
       n_clusters=2, n_init=20, n_jobs=None, precomp
ute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [47]:

```
km1.labels_
```

Out[47]:

```
array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1], dtype=int32)
```

See plot for K=2 below.

K = 3

In [48]:

```
np.random.seed(4)
km2 = KMeans(n_clusters=3, n_init=20)
km2.fit(X)
```

Out[48]:

```
KMeans(algorithm='auto', copy_x=True, init='k-means+
+', max_iter=300,
       n_clusters=3, n_init=20, n_jobs=None, precomp
ute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

In [49]:

```
pd.Series(km2.labels_).value_counts()
```

Out[49]:

```
1    21
0    20
2     9
dtype: int64
```

In [50]:

```
km2.cluster_centers_
```

Out[50]:

```
array([[-0.27876523,  0.51224152],
       [ 2.82805911, -4.11351797],
       [ 0.69945422, -2.14934345]])
```

In [51]:

```
km2.labels_
```

Out[51]:

```
array([1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
2, 2, 1, 1, 1, 1, 2,
       1, 1, 1, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0,
0, 0, 2, 0, 0, 2, 0,
       0, 0, 0, 0, 0, 2], dtype=int32)
```

In [52]:

```
# Sum of distances of samples to their closest cluster center.
km2.inertia_
```

Out[52]:

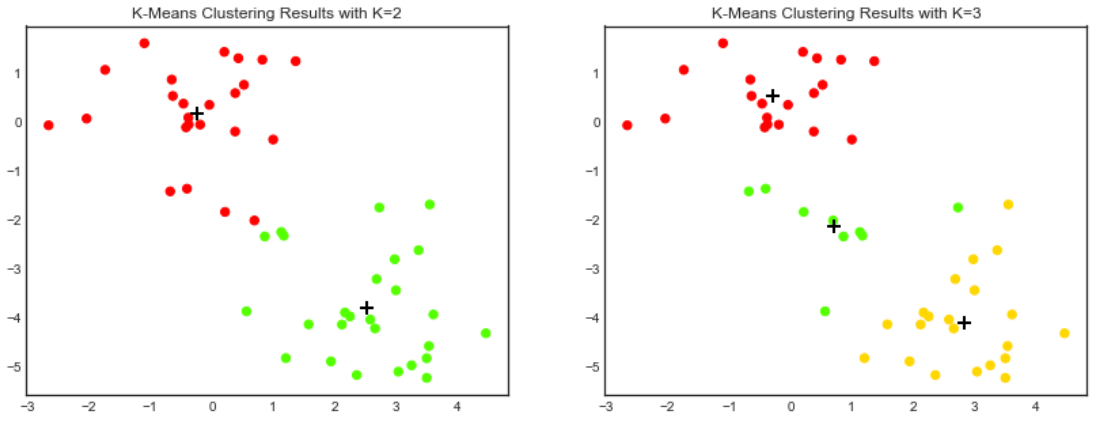
```
68.97379200939726
```

In [53]:

```
fig, (ax1, ax2) = plt.subplots(1,2, figsize=(14,5))

ax1.scatter(X[:,0], X[:,1], s=40, c=km1.labels_, cmap=plt.cm.pris
m)
ax1.set_title('K-Means Clustering Results with K=2')
ax1.scatter(km1.cluster_centers_[:,0], km1.cluster_centers_[:,1],
marker='+', s=100, c='k', linewidth=2)

ax2.scatter(X[:,0], X[:,1], s=40, c=km2.labels_, cmap=plt.cm.pris
m)
ax2.set_title('K-Means Clustering Results with K=3')
ax2.scatter(km2.cluster_centers_[:,0], km2.cluster_centers_[:,1],
marker='+', s=100, c='k', linewidth=2);
```

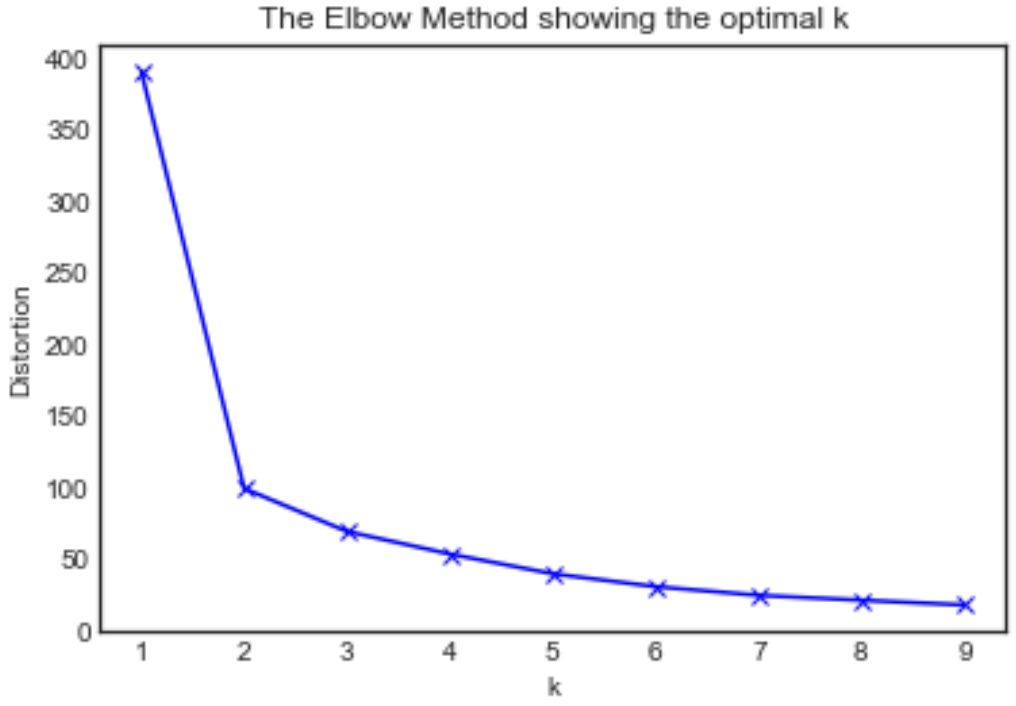


In [54]:

```
wcss = []
# Plot the elbow
K = range(1,11)

K = range(1,10)
for k in K:
    km = KMeans(n_clusters=k).fit(X)
    km.fit(X)
    wcss.append(km.inertia_)

# Plot the elbow
plt.plot(K, wcss, 'bx-')
plt.xlabel('k')
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```



10.5.3 Hierarchical Clustering

scipy

In [55]:

```
fig, (ax1,ax2,ax3) = plt.subplots(3,1, figsize=(15,18))

for linkage, cluster, ax in zip([hierarchy.complete(X), hierarchy
.average(X), hierarchy.single(X)], ['c1','c2','c3'],
                                [ax1,ax2,ax3]):
    cluster = hierarchy.dendrogram(linkage, ax=ax, color_threshol
d=0)

ax1.set_title('Complete Linkage')
ax2.set_title('Average Linkage')
ax3.set_title('Single Linkage');
```