# Homework 7: Unsupervised Learning

Tianyue Niu

In [88]:
```
#import necessary packages
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from sklearn.preprocessing import scale
from sklearn.metrics import silhouette_score

#disable future warning
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

### k-means clustering

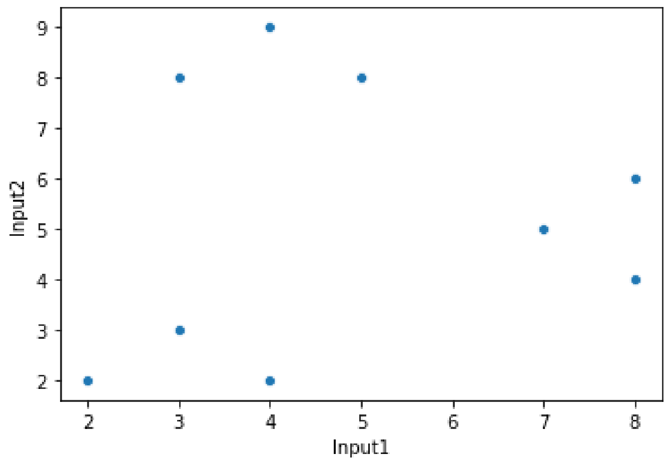
1.*(5 points) Imitate the k-means random initialization part of the algorithm by assigning each observation to a cluster at random.*

In [2]:
```
np.random.seed(2)
input1 = np.array([5,8,7,8,3,4,2,3,4,5])
input2 = np.array([8,6,5,4,3,2,2,8,9,8])
```

In [3]:
```
np.random.seed(2)
labels= np.random.choice(3,10,replace=True)
```

In [4]:
```
df = pd.DataFrame({'Input1':input1, 'Input2':input2, 'Label':labels})
```

In [5]:
```
sb.scatterplot(x='Input1', y='Input2', data=df);
```



2.*(5 points) Compute the cluster centroid and update cluster assignments for each observation iteratively based on spatial similarity.*

In [6]:
```
def get_distance(x1, y1, x2, y2):
    return ((y2-y1)**2+(x2-x1)**2)**1/2
```

In [7]:
```
def fit_k_means(ks, df):
    labels = []
    fitted = df.copy()
    centroids={}

    for k in range(ks):
        centroidx = fitted[fitted['Label']==k]['Input1'].mean()
        centroidy = fitted[fitted['Label']==k]['Input2'].mean()
        centroids[k] = (centroidx, centroidy)

    for i, row in fitted.iterrows():
        min_distance = 10000
        for key, value in centroids.items():
            distance = get_distance(row['Input1'],row['Input2'],
                                    value[0],value[1])
            if distance < min_distance:
                min_distance = distance
                c = key
        fitted.set_value(i, 'Label', c)
        labels.append(c)

    if list(df['Label']==labels):
        return fitted

    else:
        return fit_k_means(ks, df)
```

In [8]:
```
fitted = fit_k_means(3, df)
```

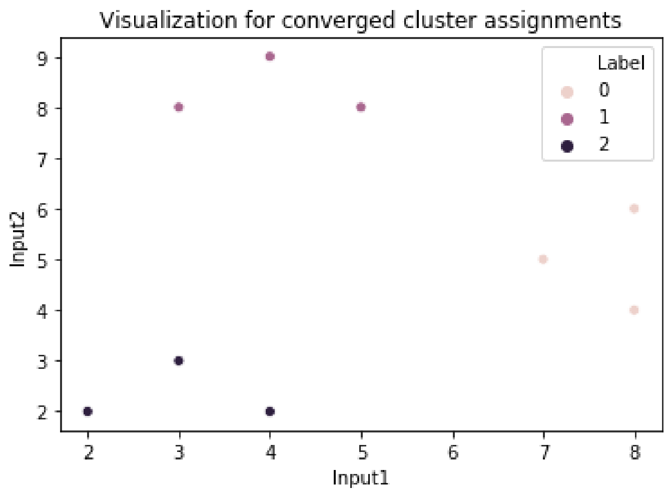
In [9]:
```
fitted
```

Out[9]:

| | Input1 | Input2 | Label |
|---|---|---|---|
| **0** | 5 | 8 | 1 |
| **1** | 8 | 6 | 0 |
| **2** | 7 | 5 | 0 |
| **3** | 8 | 4 | 0 |
| **4** | 3 | 3 | 2 |
| **5** | 4 | 2 | 2 |
| **6** | 2 | 2 | 2 |
| **7** | 3 | 8 | 1 |
| **8** | 4 | 9 | 1 |
| **9** | 5 | 8 | 1 |

3.*(5 points) Present a visual description of the final, converged (stopped) cluster assignments.*

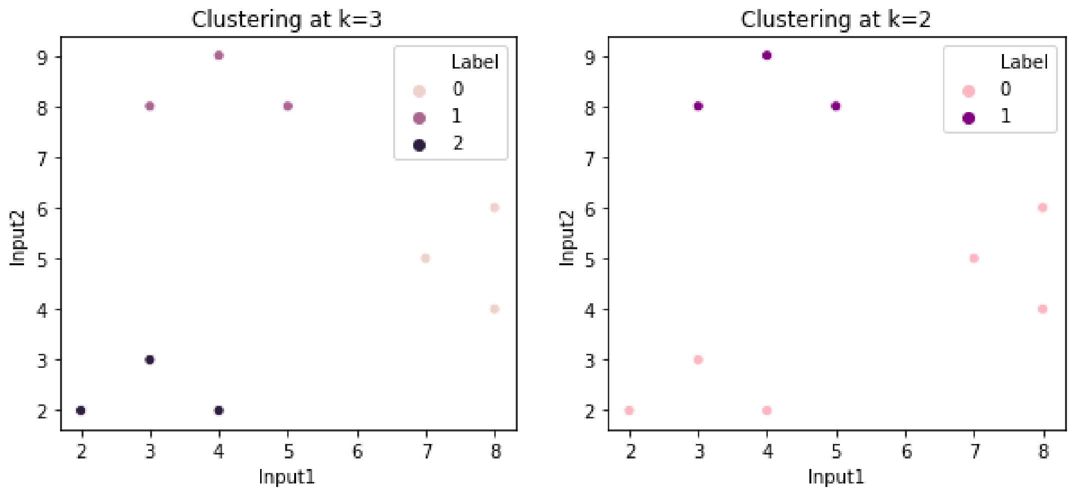
In [10]:
```
sb.scatterplot(x='Input1', y='Input2', data=fitted, hue='Label')
plt.title('Visualization for converged cluster assignments');
```



*4.(5 points) Now, repeat the process, but this time initialize at k = 2 and present a final cluster assignment visually next to the previous search at k = 3.*

In [11]:
```
fitted_2 = fit_k_means(2, df)
```

In [54]:
```
plt.figure(figsize=(10, 4))
ax = plt.subplot(121)
ax.set_title('Clustering at k=3')
sb.scatterplot(x='Input1', y='Input2', data=fitted, hue='Label')
ax = plt.subplot(122)
ax.set_title('Clustering at k=2')
sb.scatterplot(x='Input1', y='Input2', data=fitted_2, hue='Label',palette=['lightp
ink','purple']);
```



*5.(10 points) Did your initial hunch of 3 clusters pan out, or would other values of k, like 2, fit these data better? Why or why not?*

```
Compared to k-means with k=2, k-means with k=3 seems to fit the data better because th
ere exists a natual geographic pattern/spatial distribution among these data points, w
here three distinct cluters naturally exist.
```

**Applications**

Dimension Reduction

6.*(15 points) Perform PCA on the dataset and plot the observations on the first and second principal components. Describe your results, e.g.,*

- What variables appear strongly correlated on the first principal component?
- What about the second principal component?

In [55]:
```
wiki = pd.read_csv('wiki.csv')
```

In [56]:
```
wiki.head(5)
```

Out[56]:

| | age | gender | phd | yearsexp | userwiki | pu1 | pu2 | pu3 | peu1 | peu2 | ... | exp5 | domain_Sciences | domain_H |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 40 | 0 | 1 | 14 | 0 | 4 | 4 | 3 | 5 | 5 | ... | 2 | 1 | |
| **1** | 42 | 0 | 1 | 18 | 0 | 2 | 3 | 3 | 4 | 4 | ... | 4 | 0 | |
| **2** | 37 | 0 | 1 | 13 | 0 | 2 | 2 | 2 | 4 | 4 | ... | 3 | 0 | |
| **3** | 40 | 0 | 0 | 13 | 0 | 3 | 3 | 4 | 3 | 3 | ... | 4 | 0 | |
| **4** | 51 | 0 | 0 | 8 | 1 | 4 | 3 | 5 | 5 | 4 | ... | 4 | 0 | |

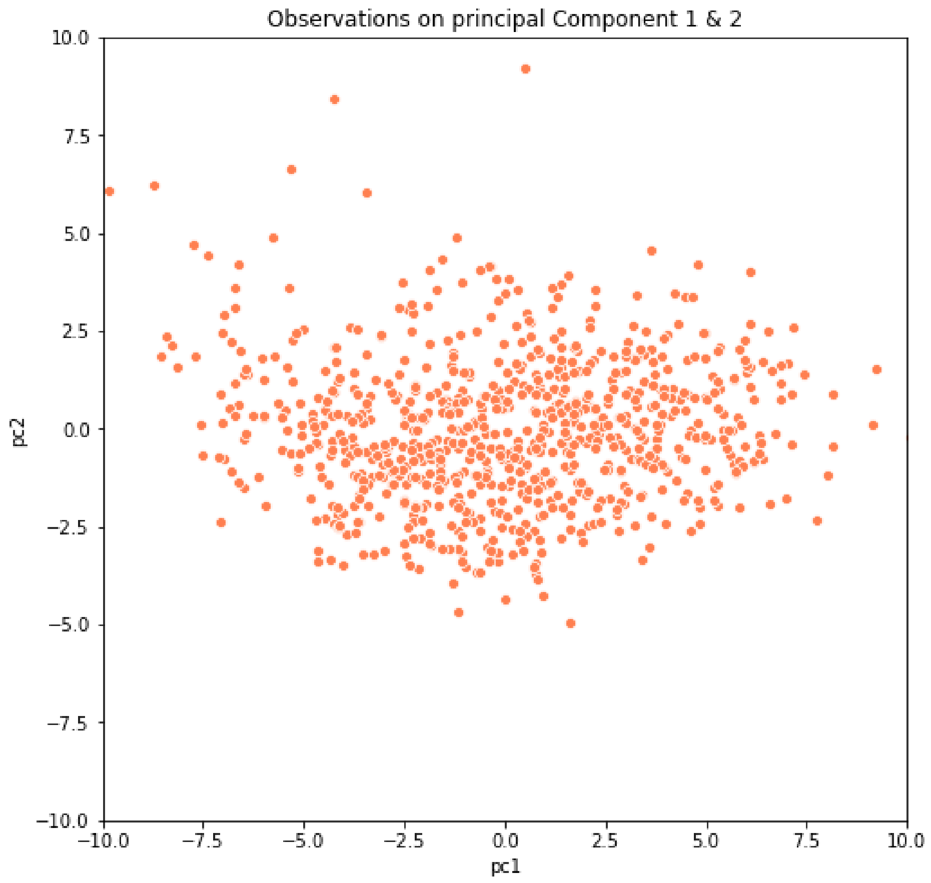
5 rows × 57 columns

In [57]:
```
#drop response variable
X = wiki.drop(columns=['userwiki'])
```

In [58]:
```
#scale X to have mean = 0 and 1 standard unit of varaince
X = scale(X)
#fit PCA
pca = PCA(n_components=2).fit(X)
```

In [59]:
```
pc = pca.transform(X)
pcdf = pd.DataFrame(pc, columns=['pc1','pc2'])
```

In [60]:
```
plt.figure(figsize = (8,8))
f = sb.scatterplot(pcdf['pc1'],pcdf['pc2'], color='coral')
f.set(ylim=(-10,10))
f.set(xlim=(-10,10))
plt.title('Observations on principal Component 1 & 2');
```



In [61]:
```
#get a list of predictors
v = wiki.drop(columns=['userwiki']).columns
```

In [62]:
```
#combine lists to create data frame
df_pca = pd.DataFrame({'vars': v,'pc1': pca.components_[0], 'pc2':  pca.components
_[1]})
#create columns for absolute values
df_pca['pc1_abs'] = df_pca['pc1'].abs()
df_pca['pc2_abs'] = df_pca['pc2'].abs()
```

In [63]:
```
#find variables with highest correlations with pca1
df_pca.sort_values(by=['pc1_abs'], ascending=False).head(5)
```

Out[63]:

| | vars | pc1 | pc2 | pc1_abs | pc2_abs |
|---|---|---|---|---|---|
| **37** | bi2 | 0.231545 | 0.088573 | 0.231545 | 0.088573 |
| **36** | bi1 | 0.226856 | 0.060835 | 0.226856 | 0.060835 |
| **28** | use3 | 0.219028 | 0.160996 | 0.219028 | 0.160996 |
| **29** | use4 | 0.214833 | 0.166242 | 0.214833 | 0.166242 |
| **6** | pu3 | 0.211832 | 0.038815 | 0.211832 | 0.038815 |

```
We see that bi2, bi1, use3, use4, pu3 have the highest correlation with pca1, which co
rrepond to the following variables:
```

- bi1: In the future I will recommend the use of Wikipedia to my colleagues and students
- bi2: In the future I will use Wikipedia in my teaching activity
- use3: I recommend my students to use Wikipedia
- use4: I recommend my colleagues to use Wikipedia
- pu3: The use of Wikipedia makes it easier for students to develop new skills

```
We can interpret pca1 as a general preference (reflected from user behaviors) toward W
ikipedia, which intuitively makes a lot of sense.
```

In [64]:
```
#find variables with highest correlations with pca2
df_pca.sort_values(by=['pc2_abs'], ascending=False).head(5)
```

Out[64]:

| | vars | pc1 | pc2 | pc1_abs | pc2_abs |
|---|---|---|---|---|---|
| **7** | peu1 | 0.062337 | -0.271286 | 0.062337 | 0.271286 |
| **38** | inc1 | 0.105760 | -0.248797 | 0.105760 | 0.248797 |
| **25** | sa3 | 0.121358 | -0.246823 | 0.121358 | 0.246823 |
| **23** | sa1 | 0.122289 | -0.236574 | 0.122289 | 0.236574 |
| **24** | sa2 | 0.118397 | -0.233428 | 0.118397 | 0.233428 |

```
Predictors peu1, inc1, sa3, sa1, and sa2 are most highly correlated with the second PC
A. These correspond to:
```

- peu1: Wikipedia is user-friendly
- inc1: To design educational activities using Wikipedia, it would be helpful: a best practices guide
- sa3: It is important that students become familiar with online collaborative environments
- sa1: It is important to share academic content in open platforms
- sa2: It is important to publish research results in other media than academic journals or books

```
These varaibles could be collectively interpreted as opinions on open platforms and on
line interactions, which might also influence one's decision to use wikipedia in clas
s.
```

7.*(5 points) Calculate the proportion of variance explained (PVE) and the cumulative PVE for all the principal components. Approximately how much of the variance is explained by the first two principal components?*

In [65]:
```
PVE12 = pca.explained_variance_ratio_[0]+pca.explained_variance_ratio_[1]
print(PVE12,"% of varaince is explained by the first two principal components.")
```

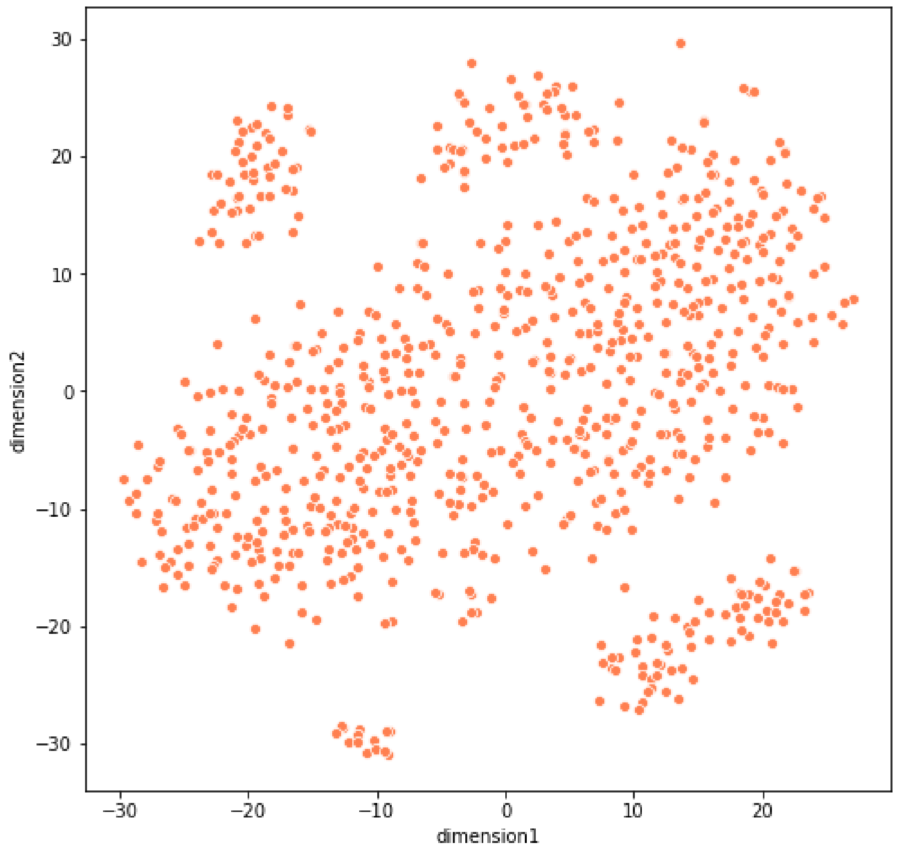
```
0.2947644028293388 % of varaince is explained by the first two principal componen
ts.
```

8.*(10 points) Perform t-SNE on the dataset and plot the observations on the first and second dimensions. Describe your results.*

```
In [66]: X_embedded = TSNE(n_components=2, random_state=2).fit_transform(X)
```

```
In [67]: X_embedded = pd.DataFrame(X_embedded, columns = ['dimension1','dimension2'])
```

```
In [68]: plt.figure(figsize = (8,8))
         f = sb.scatterplot(X_embedded['dimension1'],X_embedded['dimension2'], color='cora
         l');
```



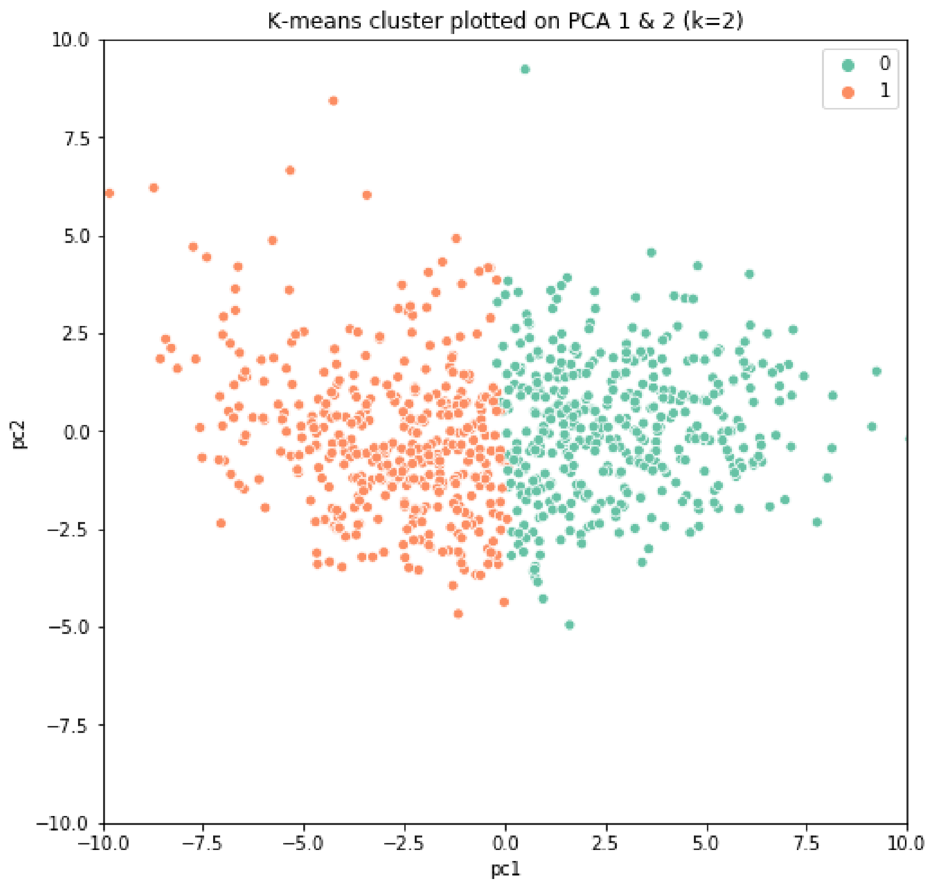
```
From the above graph we see that there is a main cluster in the middle, and four other
smaller clusters around it. This probably means that most of data points are quite sim
ilar to each other, for example, most professors may generally dislike Wikipedia & pre
fer to not use it/recommend it to others. However, different opinions exist, as indica
ted by the surrounding smaller clusters.
```

## Clustering

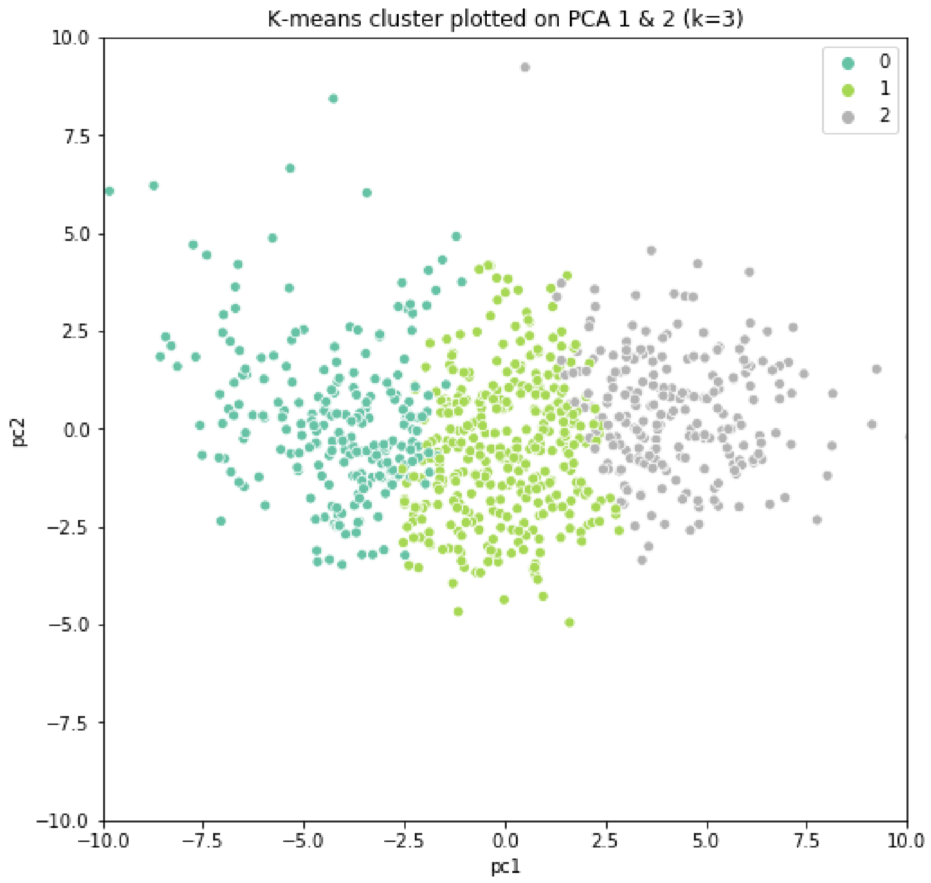
9.*(15 points) Perform k-means clustering with k = 2, 3, 4. Be sure to scale each feature (i.e.,mean zero and standard deviation one). Plot the observations on the first and second principal components from PCA and color-code each observation based on their cluster membership. Discuss your results.*

In [69]:
```
kmeans2 = KMeans(n_clusters=2, random_state=2).fit(X)
kmeans3 = KMeans(n_clusters=3, random_state=2).fit(X)
kmeans4 = KMeans(n_clusters=4, random_state=2).fit(X)
```

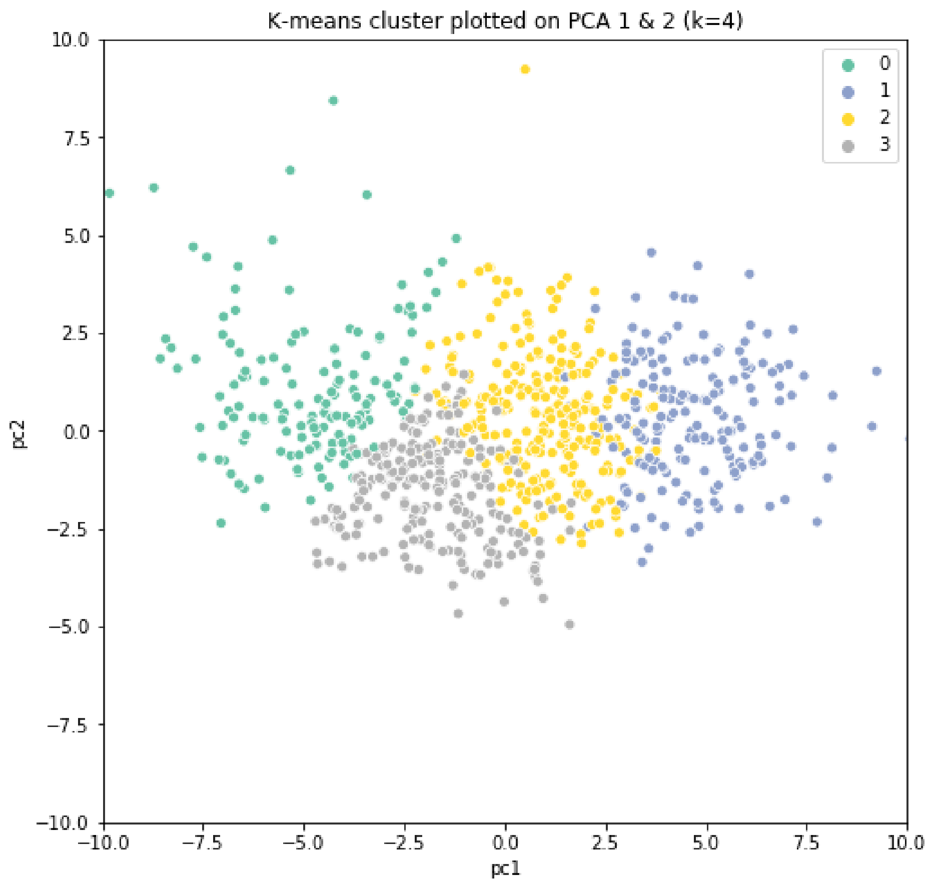
In [83]:
```
plt.figure(figsize = (8,8))
f = sb.scatterplot(pcdf['pc1'],pcdf['pc2'], hue = kmeans2.labels_, palette='Set2')
f.set(ylim=(-10,10))
f.set(xlim=(-10,10))
plt.title('K-means cluster plotted on PCA 1 & 2 (k=2)');
```



In [85]:
```
plt.figure(figsize = (8,8))
f = sb.scatterplot(pcdf['pc1'],pcdf['pc2'], hue = kmeans3.labels_, palette='Set2')
f.set(ylim=(-10,10))
f.set(xlim=(-10,10))
plt.title('K-means cluster plotted on PCA 1 & 2 (k=3)');
```



In [87]:
```
plt.figure(figsize = (8,8))
f = sb.scatterplot(pcdf['pc1'],pcdf['pc2'], hue = kmeans4.labels_, palette='Set2')
f.set(ylim=(-10,10))
f.set(xlim=(-10,10))
plt.title('K-means cluster plotted on PCA 1 & 2 (k=4)');
```

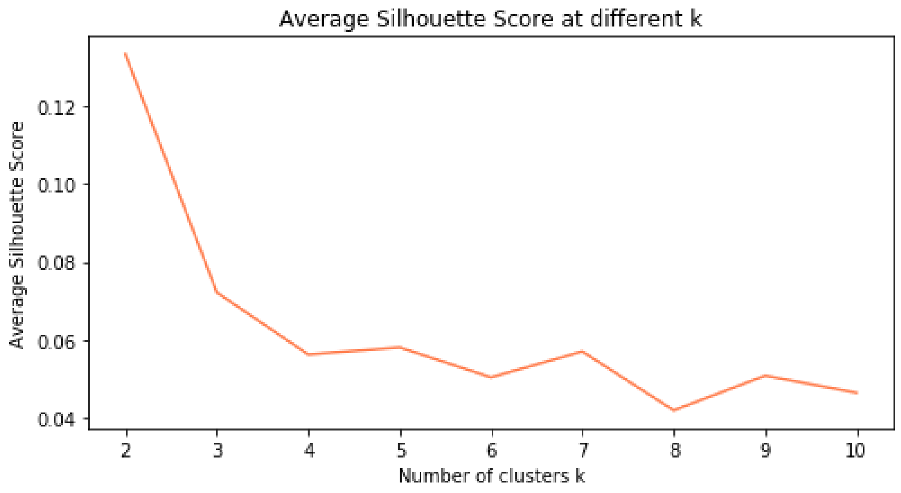


```
At k=2, we see the clusters can be well explained by the first two principal component
s, with no overlap between the two clusters. This probably means that our first and se
cond principal components do capture the most significant differences between clusters
when k =2. As we go on, we see that at k=3, cluster points start to overlap a little b
it, and at k=4, the extent of overlap increases. However, we can still generally separ
ate the clusters from each other. This probably means that, while the first and second
principal components can't explain all the differences between clusters as k increase
s, they can still explain a majority of the variances.
```

10.*(10 points) Use the elbow method, average silhouette, and/or gap statistic to identify the optimal number of clusters based on k-means clustering with scaled features.*

In [109]:
```
#the sihoulette method
sil_scores = []
for k in range(2,11):
    clusterer = KMeans(n_clusters=k, random_state=2)
    cluster_labels = clusterer.fit_predict(X)
    silhouette_avg = silhouette_score(X, cluster_labels)
    sil_scores.append(silhouette_avg)
```

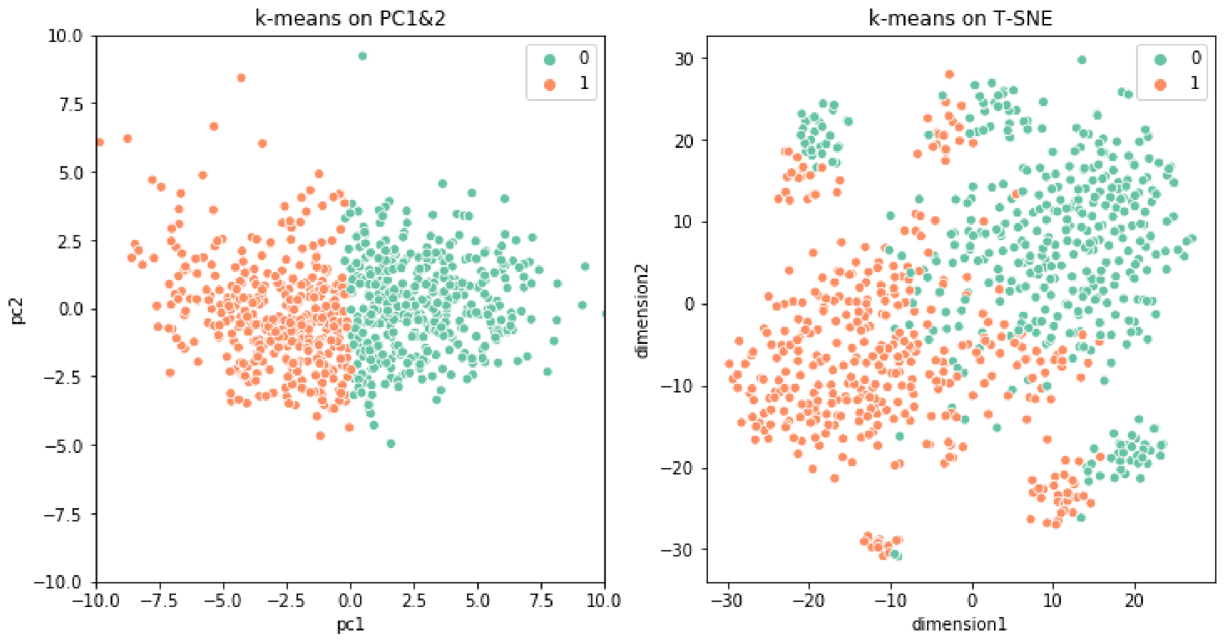
In [110]:
```
plt.figure(figsize=(8,4))
sb.lineplot(np.arange(2,11,1), sil_scores, color='coral')
plt.xlabel('Number of clusters k')
plt.ylabel('Average Silhouette Score')
plt.title('Average Silhouette Score at different k');
```



```
We see that at k=2, we have the highest average silhouette score.
```

11.*(15 points) Visualize the results of the optimal $\hat{k}$-means clustering model. First use the first and second principal components from PCA, and color-code each observation based on their cluster membership. Next use the first and second dimensions from t-SNE, and color-code each observation based on their cluster membership. Describe your results. How do your interpretations differ between PCA and t-SNE?*

In [112]:
```
plt.figure(figsize = (12,6))
ax = plt.subplot(121)
f = sb.scatterplot(pcdf['pc1'],pcdf['pc2'], hue = kmeans2.labels_, palette='Set2')
f.set(ylim=(-10,10))
f.set(xlim=(-10,10))
plt.title('k-means on PC1&2');
ax = plt.subplot(122)
f = sb.scatterplot(X_embedded['dimension1'],X_embedded['dimension2'],
                   hue = kmeans2.labels_, palette='Set2')
plt.title('k-means on T-SNE');
```



```
The above two graphs clearly demonstrate that PCA explains the separation of the 2 clu
sters better than t-SNE. This is probably because PCA's goal is to separate different
 observations and group similar observations into the same cluster, whereas t-SNE tend
to preserve the individual relations between neighboring observations. In this case, w
hether or not to use wikipedia in teaching has a clear binary response that can be wel
l approximated by the first and second principal components, so pca explains the separ
ation of the 2 clusters better.
```