k_means_example1

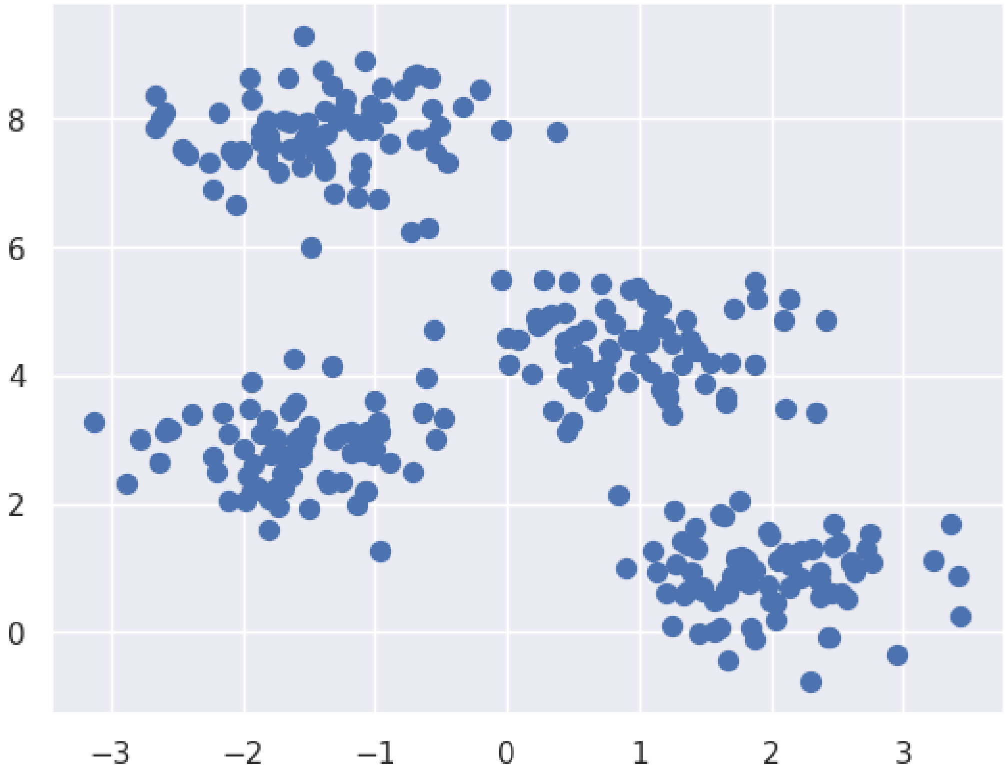
December 19, 2022

[ ]:

```
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()  # for plot styling
import numpy as np
```

[ ]:

```
from sklearn.datasets import make_blobs
X, y_true = make_blobs(n_samples=300, centers=4,
                       cluster_std=0.60, random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50);
```



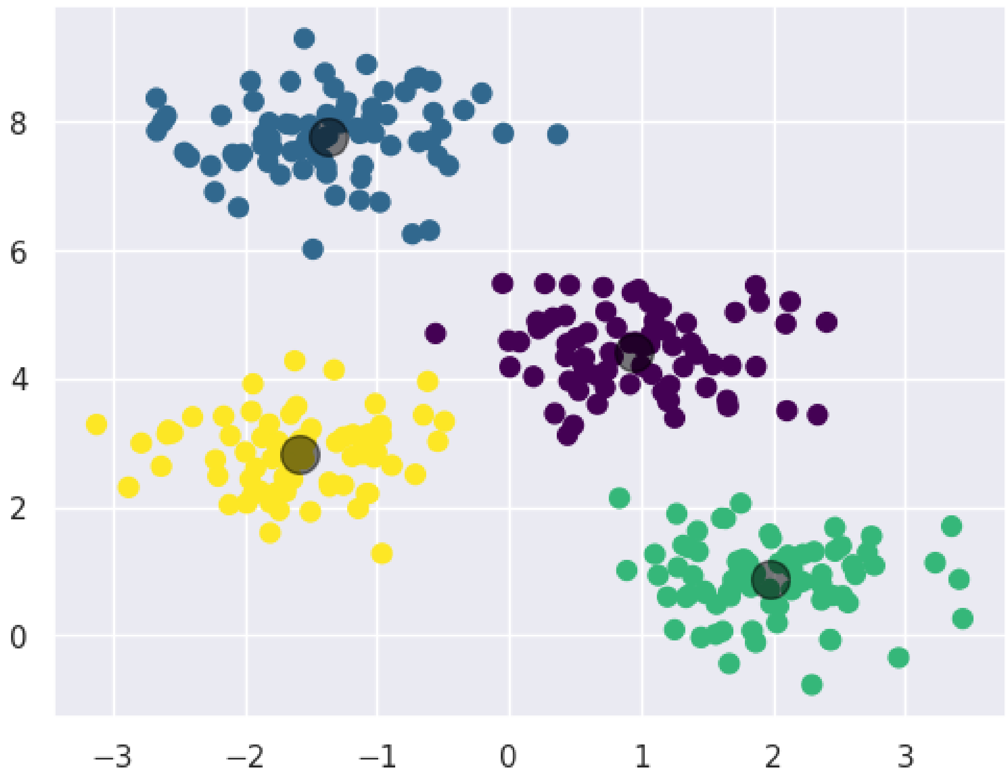
[ ]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4)
```

```
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
```

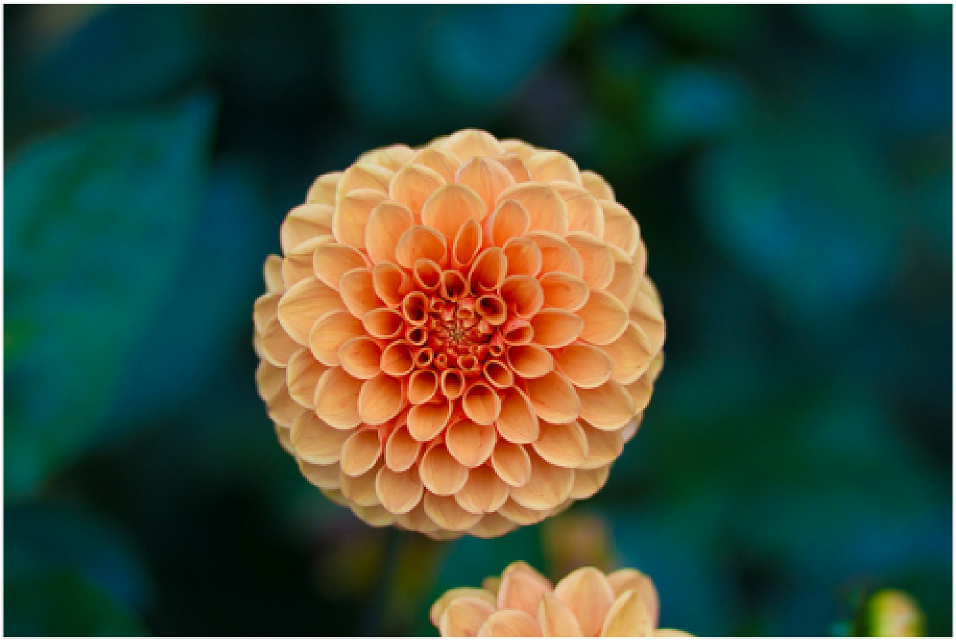
[ ]:

```
from sklearn.metrics import pairwise_distances_argmin

def find_clusters(X, n_clusters, rseed=2):
    # 1. Randomly choose clusters
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # 2a. Assign labels based on closest center
        labels = pairwise_distances_argmin(X, centers)

        # 2b. Find new centers from means of points
        new_centers = np.array([X[labels == i].mean(0)
                                for i in range(n_clusters)])

        # 2c. Check for convergence
        if np.all(centers == new_centers):
            break
        centers = new_centers

    return centers, labels

centers, labels = find_clusters(X, 4)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```



[ ]:

```
from sklearn.datasets import load_sample_image
china = load_sample_image("flower.jpg")
ax = plt.axes(xticks=[], yticks=[])
ax.imshow(china);
```



```
[ ]: china.shape
```

```
[ ]: (427, 640, 3)
```

```
[ ]: data = china / 255.0 # use 0...1 scale
     data = data.reshape(427 * 640, 3)
     data.shape
```

```
[ ]: (273280, 3)
```

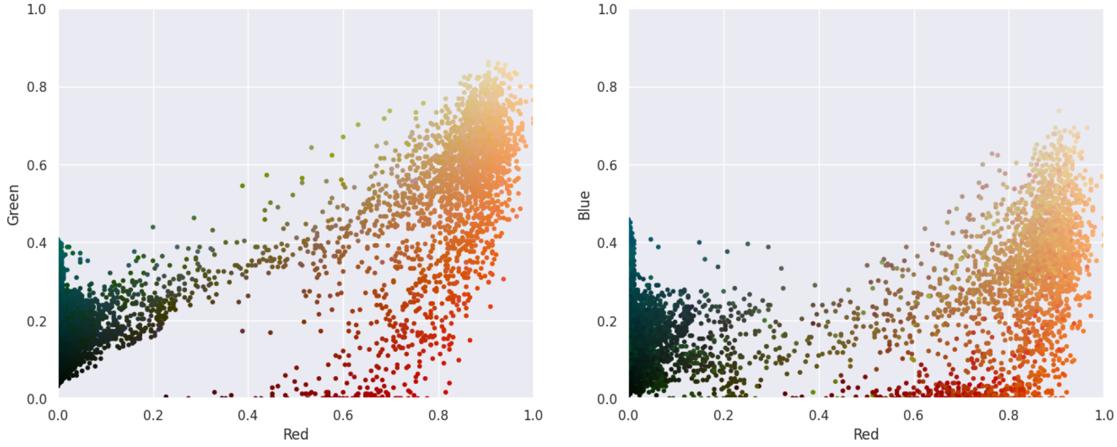
```
[ ]: def plot_pixels(data, title, colors=None, N=10000):
         if colors is None:
             colors = data

         # choose a random subset
         rng = np.random.RandomState(0)
         i = rng.permutation(data.shape[0])[:N]
         colors = colors[i]
         R, G, B = data[i].T

         fig, ax = plt.subplots(1, 2, figsize=(16, 6))
         ax[0].scatter(R, G, color=colors, marker='.')
         ax[0].set(xlabel='Red', ylabel='Green', xlim=(0, 1), ylim=(0, 1))

         ax[1].scatter(R, B, color=colors, marker='.')
```

```
    ax[1].set(xlabel='Red', ylabel='Blue', xlim=(0, 1), ylim=(0, 1))

    fig.suptitle(title, size=20);
```

[ ]:

```
plot_pixels(data, title='Input color space: 16 million possible colors')
```

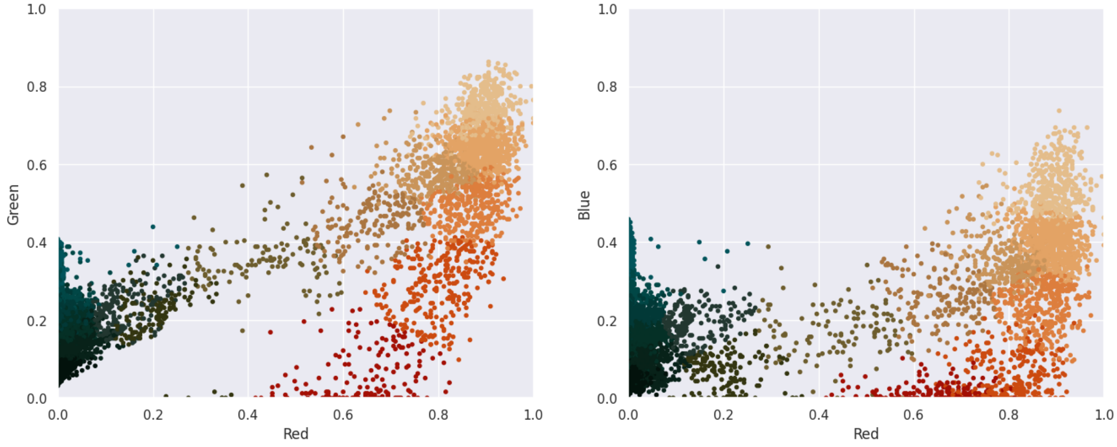
Input color space: 16 million possible colors



[ ]:

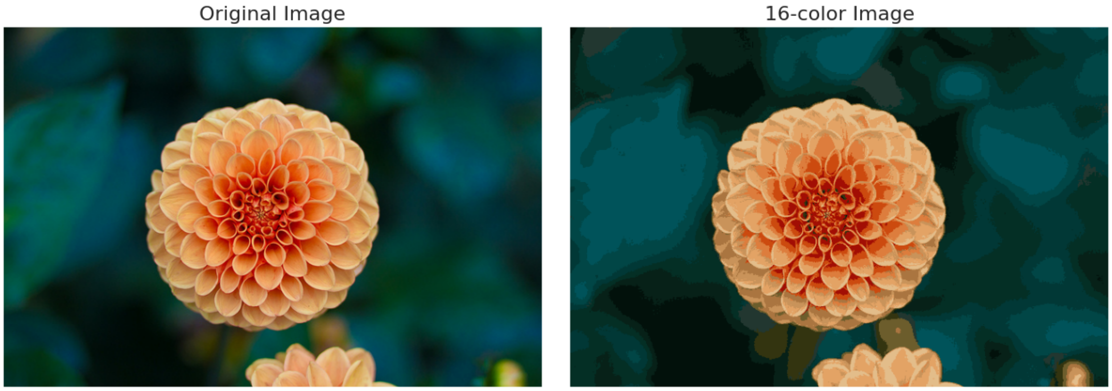
```
import warnings; warnings.simplefilter('ignore')  # Fix NumPy issues.

from sklearn.cluster import MiniBatchKMeans
kmeans = MiniBatchKMeans(16)
kmeans.fit(data)
new_colors = kmeans.cluster_centers_[kmeans.predict(data)]

plot_pixels(data, colors=new_colors,
            title="Reduced color space: 16 colors")
```

Reduced color space: 16 colors



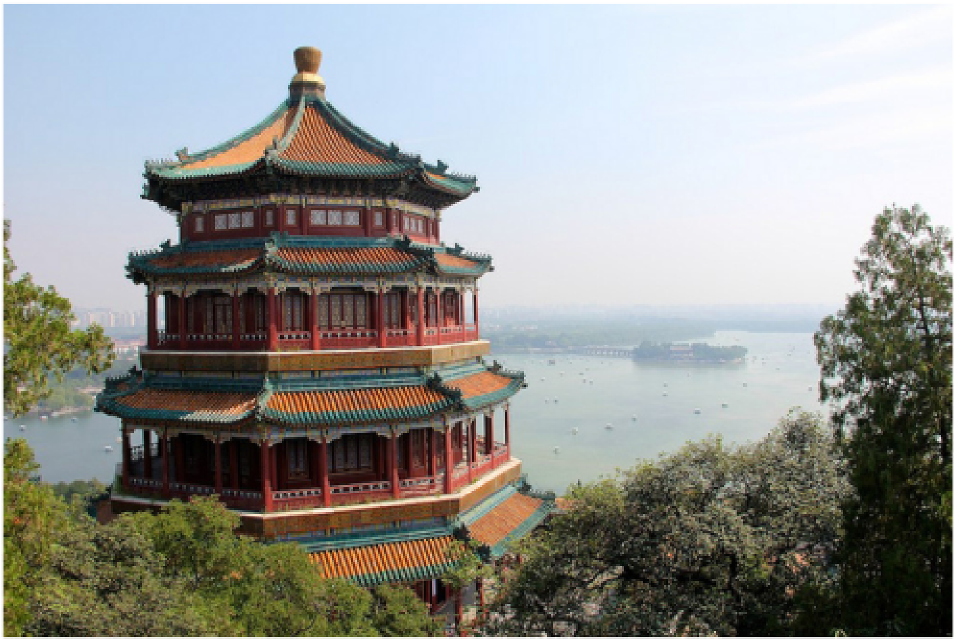
[ ]:

```
china_recolored = new_colors.reshape(china.shape)

fig, ax = plt.subplots(1, 2, figsize=(16, 6),
                       subplot_kw=dict(xticks=[], yticks=[]))
fig.subplots_adjust(wspace=0.05)
ax[0].imshow(china)
ax[0].set_title('Original Image', size=16)
ax[1].imshow(china_recolored)
ax[1].set_title('16-color Image', size=16);
```



[ ]:

```
from sklearn.datasets import load_sample_image
china = load_sample_image("china.jpg")
ax = plt.axes(xticks=[], yticks=[])
ax.imshow(china);
```



[ ]:
```
china.shape
```

[ ]:
```
(427, 640, 3)
```

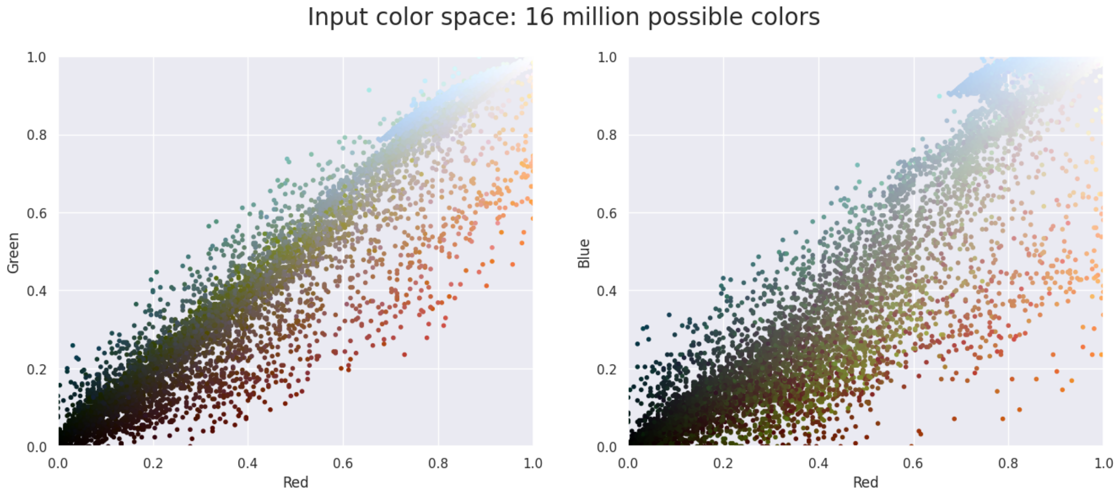
[ ]:
```
data = china / 255.0 # use 0...1 scale
data = data.reshape(427 * 640, 3)
data.shape
```

[ ]:
```
(273280, 3)
```

[ ]:
```
def plot_pixels(data, title, colors=None, N=10000):
    if colors is None:
        colors = data

    # choose a random subset
    rng = np.random.RandomState(0)
    i = rng.permutation(data.shape[0])[:N]
    colors = colors[i]
    R, G, B = data[i].T

    fig, ax = plt.subplots(1, 2, figsize=(16, 6))
    ax[0].scatter(R, G, color=colors, marker='.')
    ax[0].set(xlabel='Red', ylabel='Green', xlim=(0, 1), ylim=(0, 1))

    ax[1].scatter(R, B, color=colors, marker='.')
    ax[1].set(xlabel='Red', ylabel='Blue', xlim=(0, 1), ylim=(0, 1))

    fig.suptitle(title, size=20);
```

[ ]:

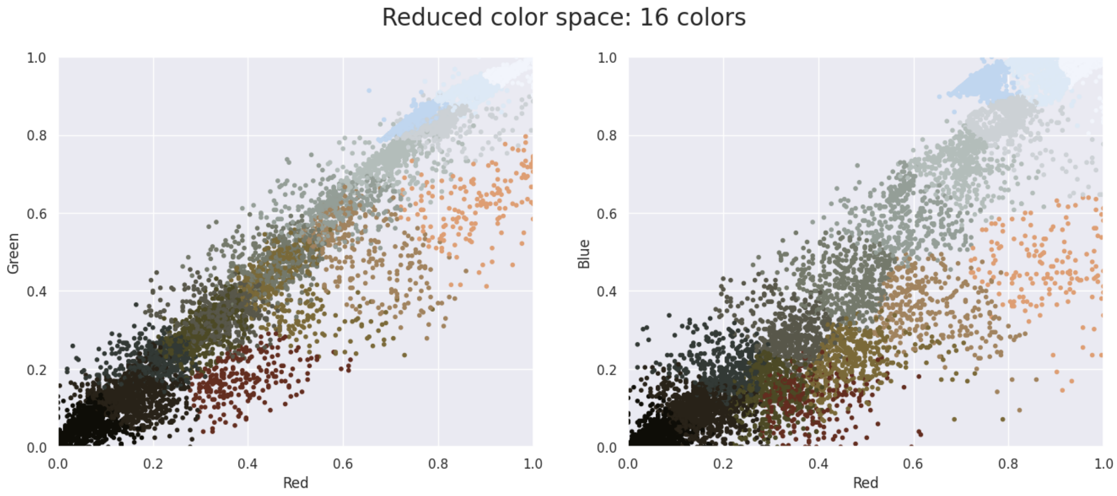
```
plot_pixels(data, title='Input color space: 16 million possible colors')
```



Input color space: 16 million possible colors

[ ]:

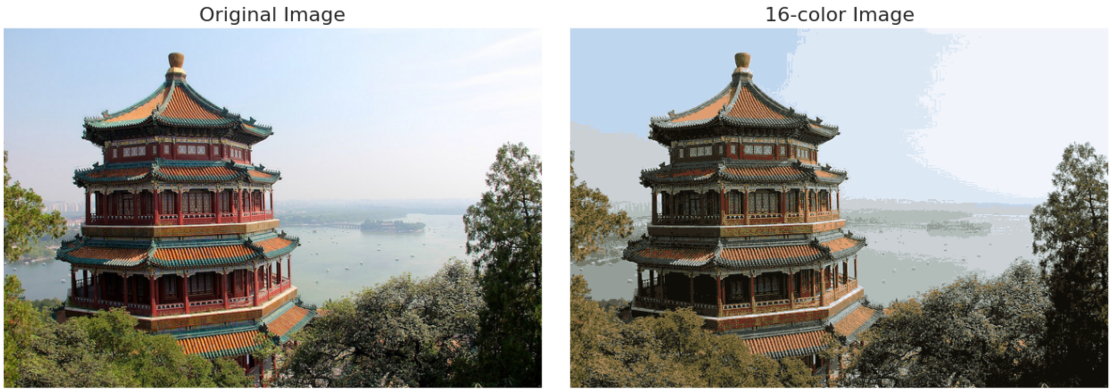
```
import warnings; warnings.simplefilter('ignore')  # Fix NumPy issues.

from sklearn.cluster import MiniBatchKMeans
kmeans = MiniBatchKMeans(16)
kmeans.fit(data)
new_colors = kmeans.cluster_centers_[kmeans.predict(data)]

plot_pixels(data, colors=new_colors,
            title="Reduced color space: 16 colors")
```



Reduced color space: 16 colors

[ ]:

```
china_recolored = new_colors.reshape(china.shape)

fig, ax = plt.subplots(1, 2, figsize=(16, 6),
                       subplot_kw=dict(xticks=[], yticks=[]))
fig.subplots_adjust(wspace=0.05)
ax[0].imshow(china)
ax[0].set_title('Original Image', size=16)
ax[1].imshow(china_recolored)
ax[1].set_title('16-color Image', size=16);
```



[ ]: