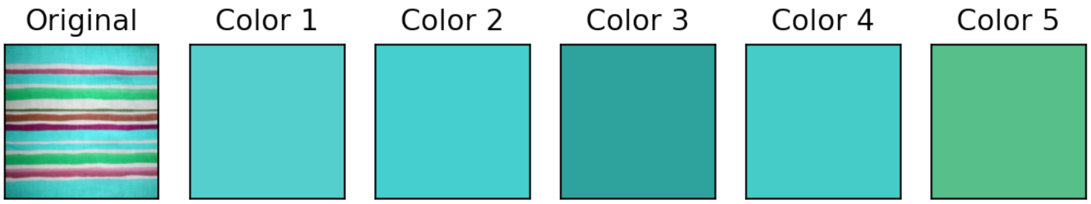
ColorPaletteExtractionAnalysis

June 11, 2021

1 K Means Clustering

[40]:

```
from PIL import Image
import random
from math import sqrt
import numpy as np
from matplotlib import pyplot as plt
import cv2 as cv
from matplotlib.pyplot import figure

class Point:

    def __init__(self, coordinates):
        self.coordinates = coordinates

class Cluster:

    def __init__(self, center, points):
        self.center = center
        self.points = points

def get_points(image_path):
    img = Image.open(image_path)
    img.thumbnail((200, 400))
    img = img.convert("RGB")
    w, h = img.size

    points = []
    for count, color in img.getcolors(w * h):
        for _ in range(count):
            points.append(Point(color))

    return points

def euclidean(p, q):
    n_dim = len(p.coordinates)
    return sqrt(sum([(p.coordinates[i] - q.coordinates[i]) ** 2 for i in␣
↪range(n_dim)]))

class KMeans:

    def __init__(self, n_clusters):
        self.n_clusters = n_clusters

    def calculate_center(self, points):
        n_dim = len(points[0].coordinates)
        vals = [0.0 for i in range(n_dim)]
        for p in points:
            for i in range(n_dim):
                vals[i] += p.coordinates[i]
        coords = [(v / len(points)) for v in vals]
        return Point(coords)

    def assign_points(self, clusters, points):
        plists = [[] for i in range(self.n_clusters)]

        for p in points:
            smallest_distance = float('inf')

        for i in range(self.n_clusters):
            distance = euclidean(p, clusters[i].center)
            if distance < smallest_distance:
                smallest_distance = distance
                idx = i

        plists[idx].append(p)

        return plists

    def fit(self, points):
        clusters = [Cluster(center=p, points=[p]) for p in random.
↪sample(points, self.n_clusters)]

        while True:

            plists = self.assign_points(clusters, points)
            diff = 0
            for i in range(self.n_clusters):
                if not plists[i]:
                    continue
                old = clusters[i]
                center = self.calculate_center(plists[i])
                new = Cluster(center, plists[i])
                clusters[i] = new
                diff = max(diff, euclidean(old.center, new.center))

            if diff < 0.2:
                break

        return clusters

def rgb_to_hex(rgb):
    return '#%s' % ''.join(('%02x' % p for p in rgb))

def get_colors(filename, n_colors=3):
    points = get_points(filename)
    clusters = KMeans(n_clusters=n_colors).fit(points)
    clusters.sort(key=lambda c: len(c.points), reverse = True)
    rgbs = [map(int, c.center.coordinates) for c in clusters]
    return list(map(rgb_to_hex, rgbs))

colors = get_colors('sample/banded_0030.jpg', n_colors=5)
img_original = cv.imread('sample/banded_0030.jpg')
img_original = cv.cvtColor(img_original, cv.COLOR_BGR2RGB)
colors_dec = []
for color in colors:
    color_r = int(color[1:3],16)
    color_g = int(color[3:5],16)
    color_b = int(color[5:7],16)
    colors_dec.append([color_r,color_g,color_b])
color_1 = [[colors_dec[0]]*400]*400
color_2 = [[colors_dec[1]]*400]*400
color_3 = [[colors_dec[2]]*400]*400
color_4 = [[colors_dec[3]]*400]*400
color_5 = [[colors_dec[4]]*400]*400
figure(figsize=(8, 6), dpi=160)
plt.subplot(161),plt.imshow(img_original)
plt.title('Original'), plt.xticks([]), plt.yticks([])
plt.subplot(162),plt.imshow(color_1)
plt.title('Color 1'), plt.xticks([]), plt.yticks([])
plt.subplot(163),plt.imshow(color_2)
plt.title('Color 2'), plt.xticks([]), plt.yticks([])
plt.subplot(164),plt.imshow(color_3)
plt.title('Color 3'), plt.xticks([]), plt.yticks([])
plt.subplot(165),plt.imshow(color_4)
plt.title('Color 4'), plt.xticks([]), plt.yticks([])
plt.subplot(166),plt.imshow(color_5)
plt.title('Color 5'), plt.xticks([]), plt.yticks([])
plt.show()
```


Original
Color 1
Color 2
Color 3
Color 4
Color 5