Author: Shantanu Tyagi

Date: 20-03-2021

ID: 201801015

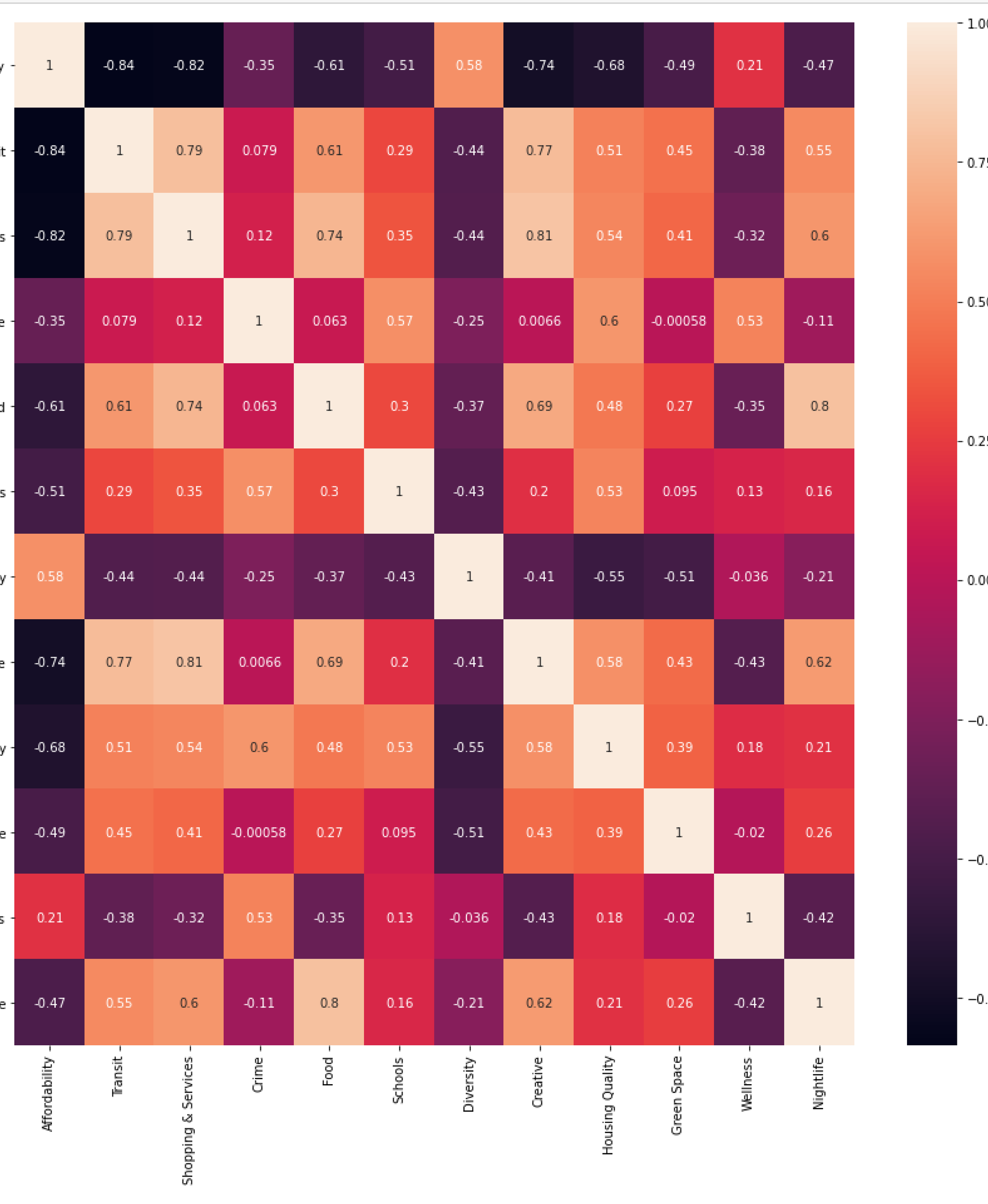
In [106]:
```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import random
import seaborn as sn
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

Pearson correlation matrix for the data:

In [107]:
```
def corrMat(df,n_comp):
    plt.figure(figsize=(15, 15))
    corrMatrix = df.corr()
    sn.heatmap(corrMatrix, annot=True)
    plt.show()
```

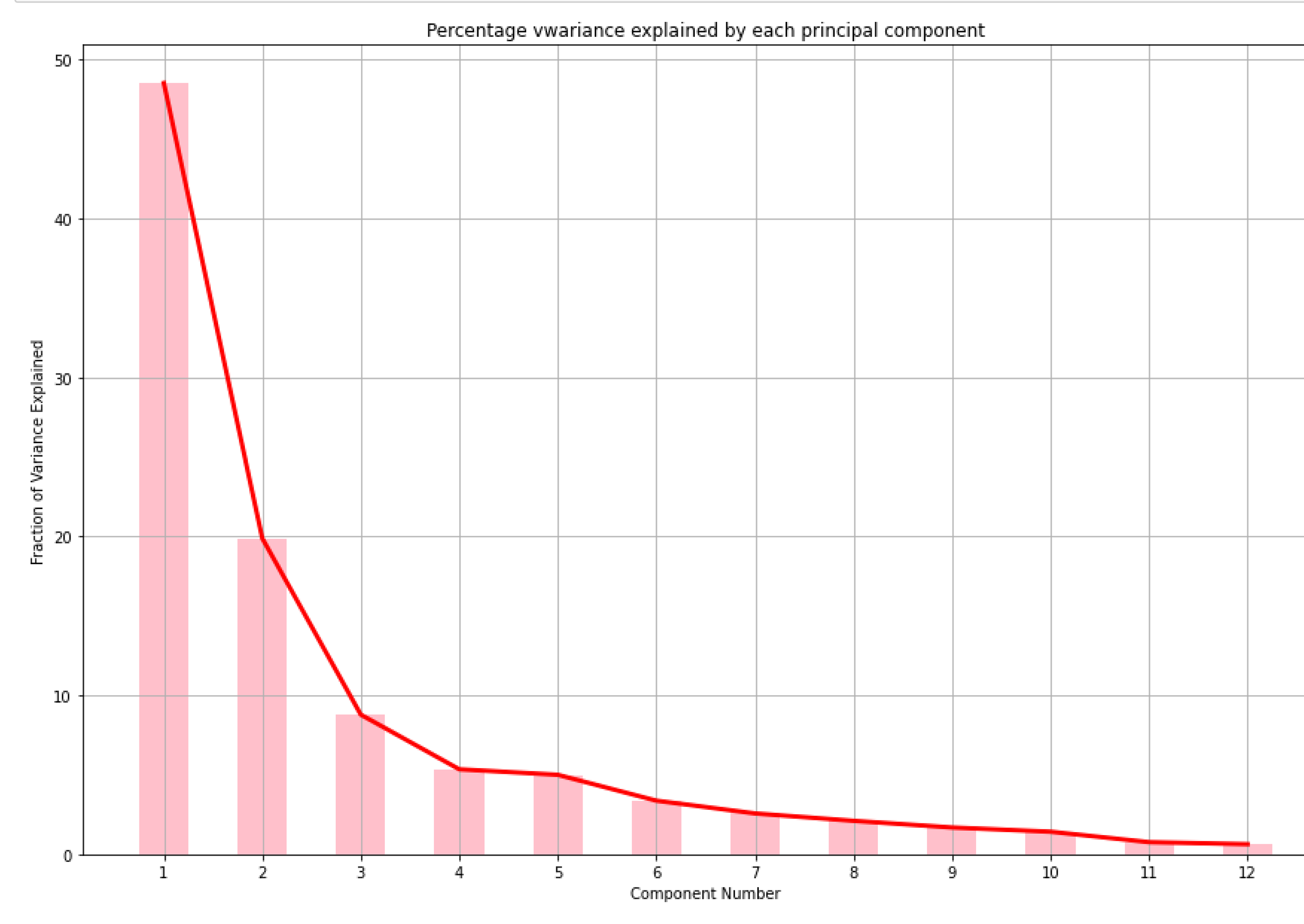
A correlation matrix is a table showing correlation coefficients between variables. Each cell in the table shows the correlation between two variables. A correlation matrix is used to summarize data. Pearson coefficient is a measure of linear correlation between two sets of data.

In [108]:
```
df = pd.read_excel("New_York_Neighborhoods.xlsx", sheet_name="Dataset")
df = df.iloc[:, 1:1+12]
#df = df.reindex(sorted(df.columns), axis=1)
corrMat(df,12)
```

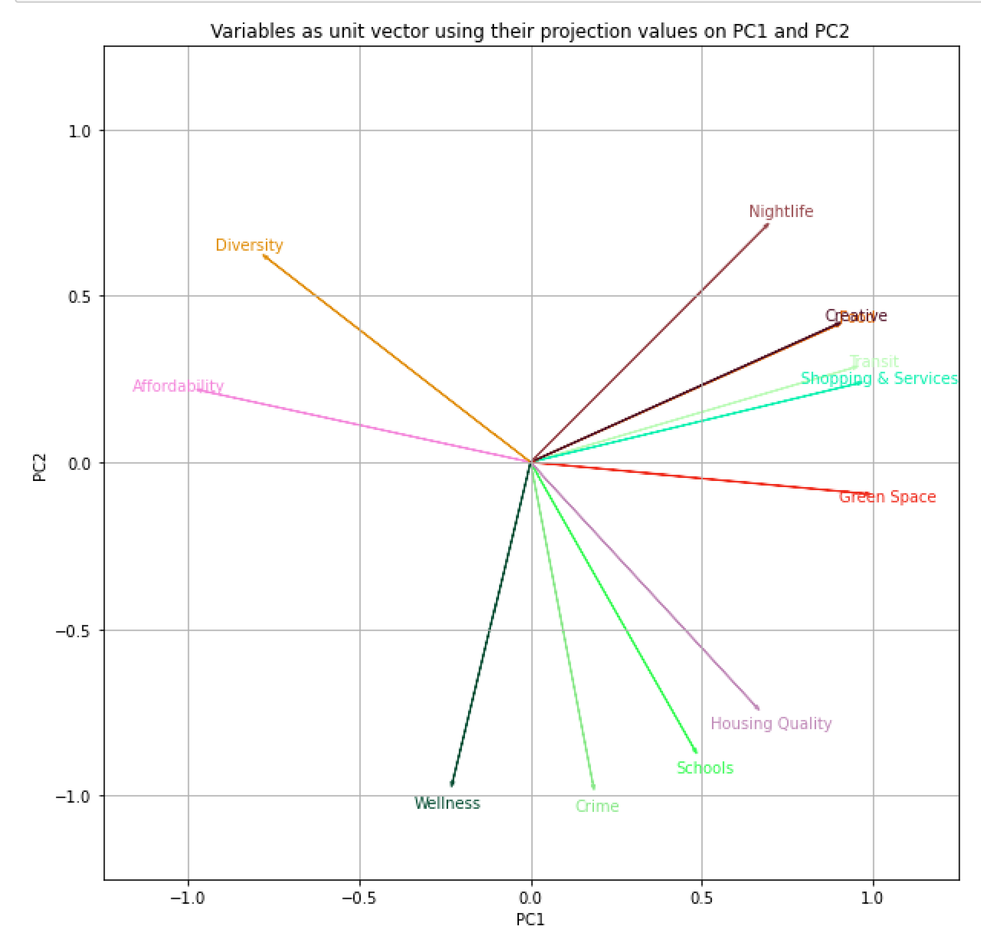


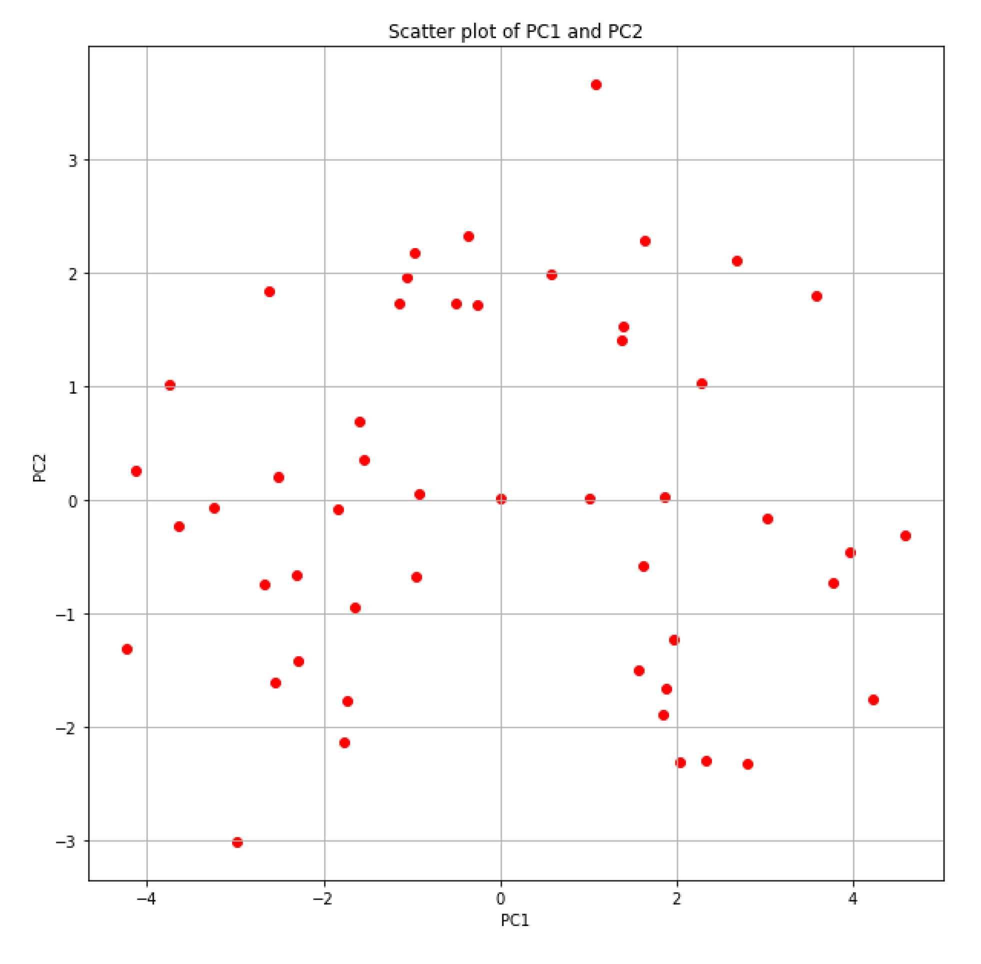
Principal component analysis (PCA) is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance. PCA is used in exploratory data analysis and for making predictive models. It is commonly used for dimensionality reduction by projecting each data point onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. The first principal component can equivalently be defined as a direction that maximizes the variance of the projected data. The i'th principal component can be taken as a direction orthogonal to the first i-1 principal components that maximizes the variance of the projected data. principal components are eigenvectors of the data's covariance matrix. PCA defines a new orthogonal coordinate system that optimally describes variance in a single dataset.

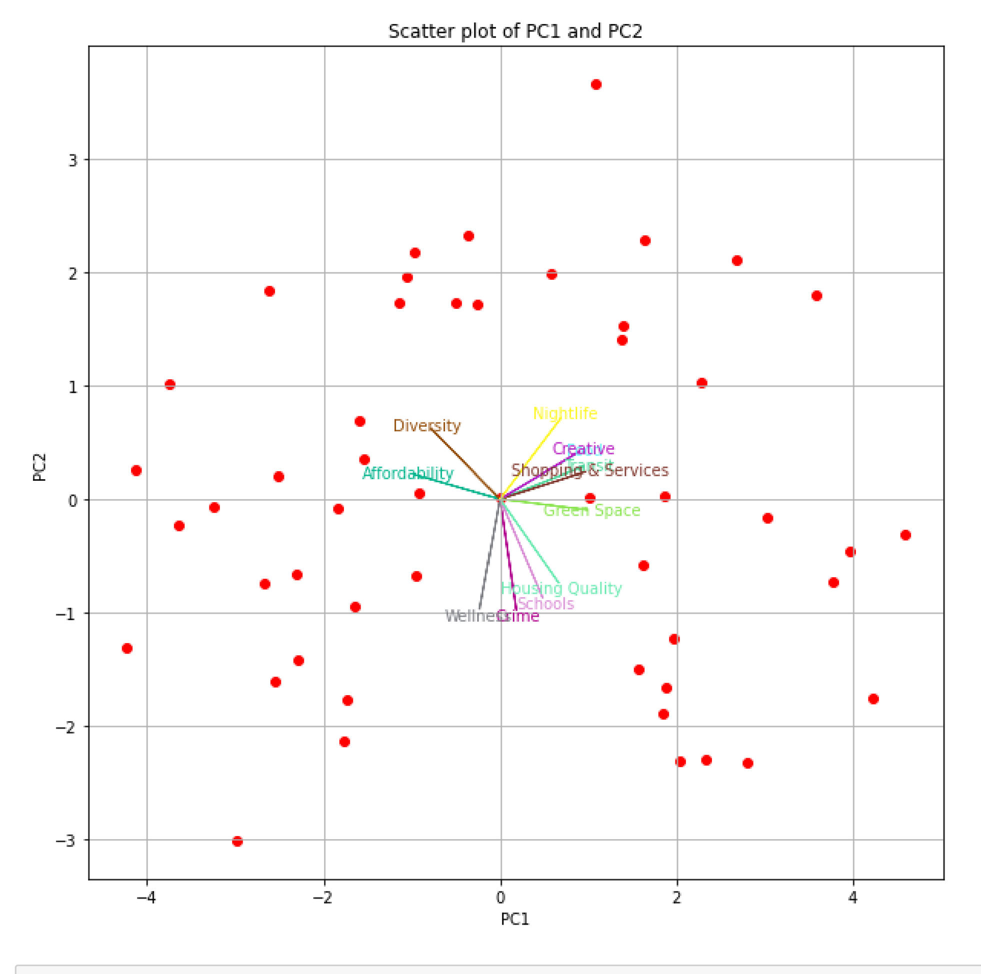
In [109]:
```
scaler = StandardScaler()
scaler.fit(df.values)
dff=scaler.transform(df.values)
pca = PCA(n_components=12)
pca.fit(dff)
pcaComp = pca.fit_transform(dff)
#print(pca.explained_variance_ratio_)
#columns = ['pca_%i' % i for i in range(12)]
#df_pca = pd.DataFrame(pca.transform(df), columns=columns, index=df.index)
plt.figure(figsize=(15, 10))
plt.plot(np.linspace(1,12,12),100*pca.explained_variance_ratio_,linewidth = 3, color ='r')
plt.bar(np.linspace(1,12,12),100*pca.explained_variance_ratio_, color ='pink',width = 0.5)
#plt.scatter(np.linspace(1,12,12),pca.explained_variance_ratio_,'g--')
plt.xticks(np.linspace(1,12,12))
plt.grid()
plt.title('Percentage vwariance explained by each principal component')
plt.xlabel('Component Number')
plt.ylabel('Fraction of Variance Explained')
plt.show()
```



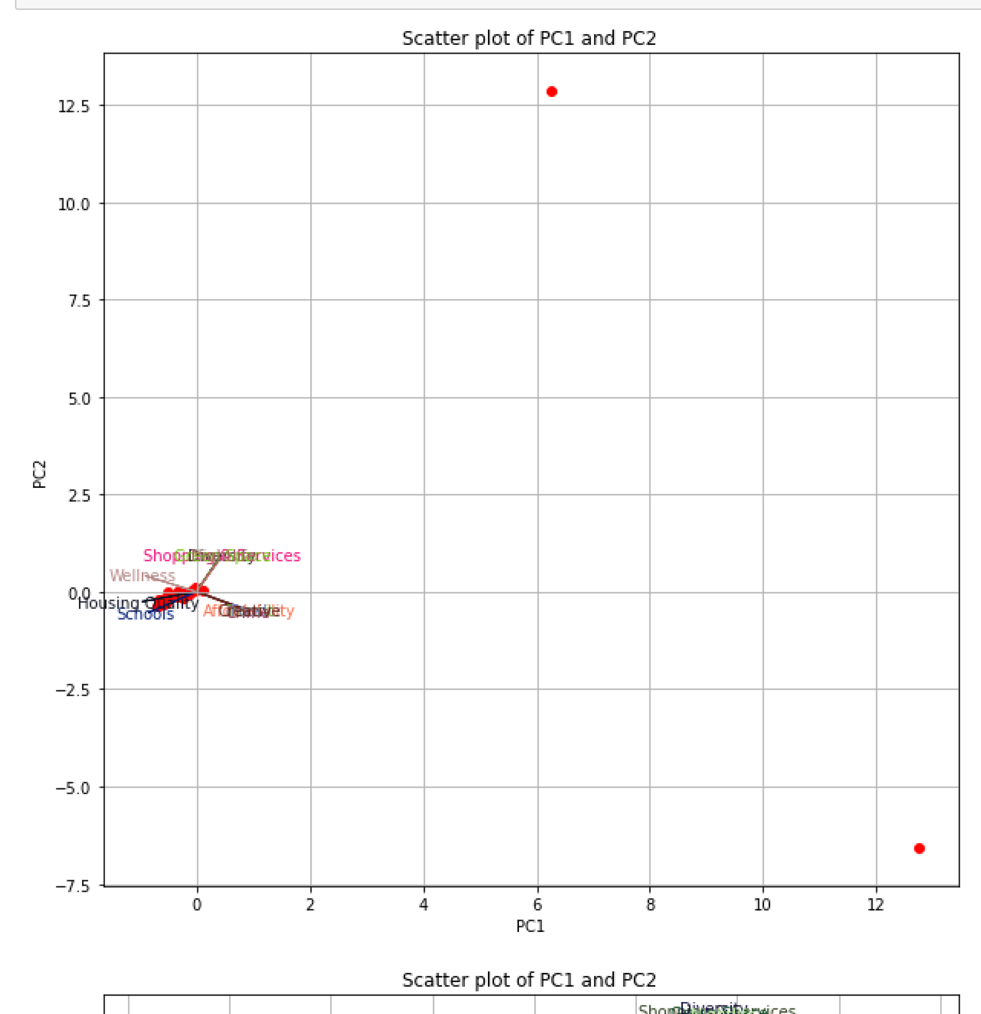
In [110]:
```
def vect(coeff,labels,T):
    n = coeff.shape[0]
    r = random.sample(range(256), n)
    g = random.sample(range(256), n)
    b = random.sample(range(256), n)
    if T == True:
        plt.figure(figsize=(10, 10))
    for i in range(n):
        color = (r[i]/256,b[i]/256,g[i]/256)
        norm = np.sqrt(coeff[i,0]**2 + coeff[i,1]**2)
        plt.arrow(0, 0, coeff[i,0]/norm, coeff[i,1]/norm,head_width=0.01,length_includes_head=True, color = color)
        if labels is None:
            plt.text(coeff[i,0]/norm* 1.05, coeff[i,1]/norm* 1.05, "Var"+str(i+1), color = color, ha = 'center', va
= 'center')
        else:
            plt.text(coeff[i,0]/norm* 1.05, coeff[i,1]/norm* 1.05, labels[i], color = color, ha = 'center', va = 'ce
nter')

    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.title('Variables as unit vector using their projection values on PC1 and PC2')

    if T == True:
        plt.xlim(left=-1.25,right=1.25)
        plt.ylim(top=1.25,bottom=-1.25)
    plt.grid()
    if T == True:
        plt.show()
vect(np.transpose(pca.components_[0:2, :]),df.columns,True)

def scat(coeff,T,scale):
    r = random.sample(range(256), 1)
    g = random.sample(range(256), 1)
    b = random.sample(range(256), 1)
    color = (r[0]/256,b[0]/256,g[0]/256)
    if T == True:
        plt.figure(figsize=(10, 10))
    if scale == False:
        plt.scatter(coeff[:,0],coeff[:,1], color ='r')
    else:
        s1 = max(coeff[:,0]) - min(coeff[:,0])
        s2 = max(coeff[:,1]) - min(coeff[:,1])
        plt.scatter(coeff[:,0]/s1,coeff[:,1]/s2, color ='r')

    plt.title('Scatter plot of PC1 and PC2')

    plt.xlabel("PC{}".format(1))
    plt.ylabel("PC{}".format(2))
    plt.grid()
    if T == True:
        plt.show()
scat(pcaComp,True,False)

def both(scale):
    plt.figure(figsize=(10, 10))
    vect(np.transpose(pca.components_[0:2, :]),df.columns,False)
    scat(pcaComp,False,scale)
    plt.grid()
    plt.show()
both(False)
```

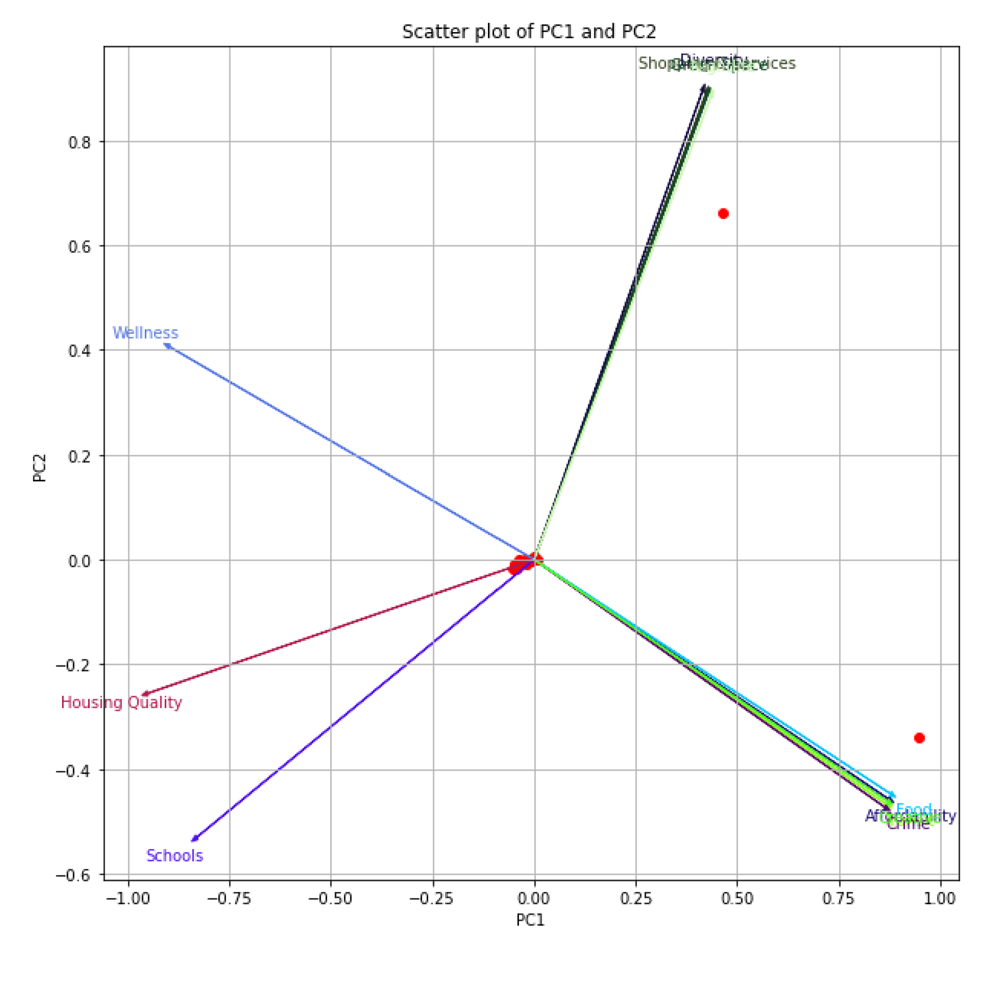






In [111]:
```
df = pd.read_excel("outlier.xlsx", sheet_name="Dataset")
df = df.iloc[:, 1:1+12]
scaler = StandardScaler()
scaler.fit(df.values)
dff=scaler.transform(df.values)
pca = PCA(n_components=12)
pca.fit(dff)
pcaComp = pca.fit_transform(dff)
both(False)
both(True)
```





First plot is an unscaled plot while the second poot is a scaled plot for the same

In [ ]: