PCA Index | Replicating Dow Index

[74]:
```
import os
import numpy as np
import pandas as pd
import pandas_datareader.data as web
from sklearn.decomposition import PCA as sklearnPCA


from pylab import plt, mpl
plt.style.use('seaborn-v0_8')
mpl.rcParams['savefig.dpi'] = 300
mpl.rcParams['font.family'] = 'serif'
pd.set_option('mode.chained_assignment', None)
pd.set_option('display.float_format', '{:.4f}'.format)
np.set_printoptions(suppress=True, precision=4)
os.environ['PYTHONHASHSEED'] = '0'
```

Constructing a PCA Index

Principal Component Analysis (PCA) is a statistical procedure that employs an orthogonal transformation to convert a set of observations of potentially correlated variables into a set of values of linearly uncorrelated variables known as principal components. The number of principal components is either equal to or less than the number of original variables. This transformation is defined in a manner that ensures the first principal component exhibits the maximum possible variance, effectively accounting for as much of the variability in the data as possible. Subsequent components, while being orthogonal (i.e., uncorrelated) to the preceding components, strive to possess the highest variance achievable under this constraint.

In this section, we present an illustrative example showcasing the application of Principal Component Analysis (PCA) in a specific context. We gather data for both the German DAX index and all the individual stocks comprising the index. Subsequently, we employ PCA to extract the principal components, which we utilize to construct a composite index referred to as the PCA_index.

Data

The DAX Index and Its 30 Stocks

[75]:
```
import pandas as pd
import yfinance as yf
```

```
# Define the symbols of the Dow Jones Industrial Average
dow_symbols = ['AAPL', 'AXP', 'BA', 'CAT', 'CSCO', 'CVX', 'DIS', 'GS',␣
 ↪'HD','IBM',
               'INTC', 'JNJ', 'JPM', 'KO', 'MCD', 'MMM', 'MRK', 'MSFT',␣
 ↪'NKE','PFE',
               'PG', 'TRV', 'UNH', 'V', 'VZ', 'WBA', 'WMT', 'XOM', '^DJI']

data = pd.DataFrame()
data = yf.download(dow_symbols, start="2010-01-01", end="2023-08-31")["Close"]
data = data.dropna()

# Separate the index data
dow_index = pd.DataFrame(data.pop('^DJI'))
```

```
[*********************100%***********************]  29 of 29 completed
```

The DataFrame object 'data' now contains data for all DAX stock

[76]:
```
data[data.columns[:10]].head()
```

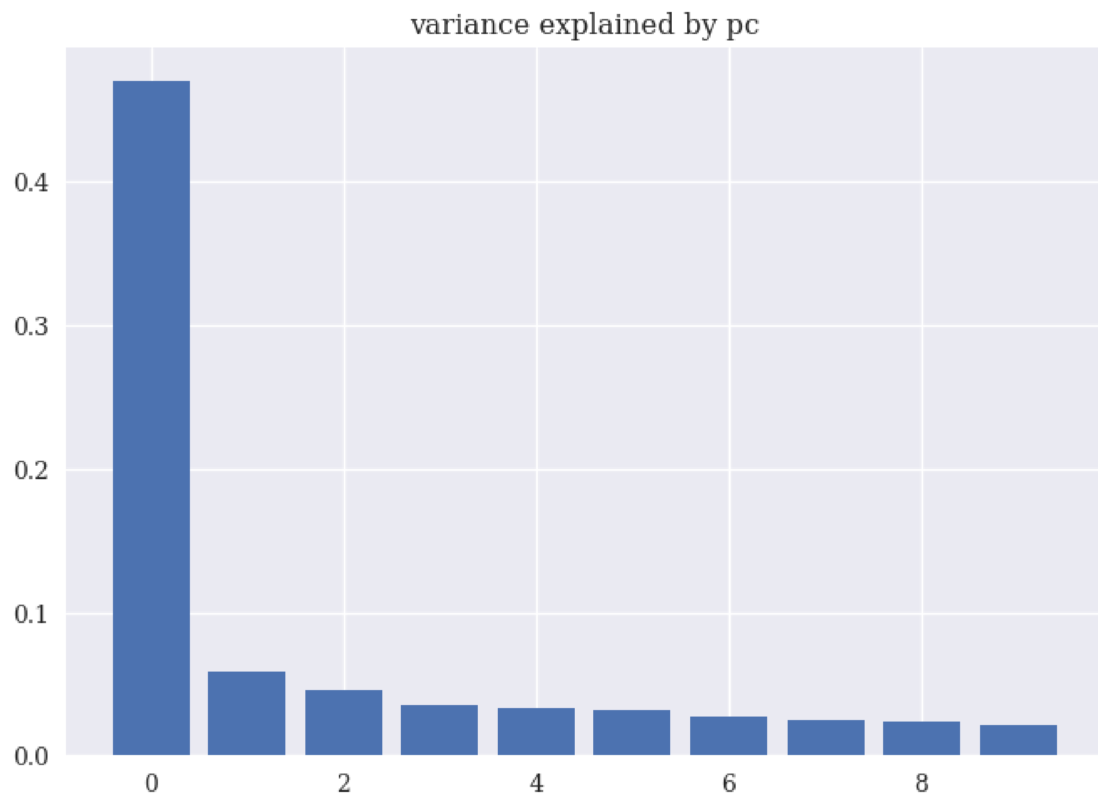
[76]:
```
              AAPL     AXP      BA     CAT    CSCO     CVX     DIS        GS  \
Date
2010-01-04  7.6432 40.9200 56.1800 58.5500 24.6900 79.0600 32.0700 173.0800
2010-01-05  7.6564 40.8300 58.0200 59.2500 24.5800 79.6200 31.9900 176.1400
2010-01-06  7.5346 41.4900 59.7800 59.4300 24.4200 79.6300 31.8200 174.2600
2010-01-07  7.5207 41.9800 62.2000 59.6700 24.5300 79.3300 31.8300 177.6700
2010-01-08  7.5707 41.9500 61.6000 60.3400 24.6600 79.4700 31.8800 174.3100

                 HD      IBM
Date
2010-01-04  28.6700 126.6252
2010-01-05  28.8800 125.0956
2010-01-06  28.7800 124.2830
2010-01-07  29.1200 123.8528
2010-01-08  28.9800 125.0956
```

[77]:
```
# dow_index cummulative returns, normalized
#dow_index_returns = dow_index.pct_change()
dow_index_returns = np.log(dow_index / dow_index.shift(1))
dow_index_returns.dropna(inplace=True)

dow_ret_idx =  dow_index_returns.cumsum() + 1

dow_ret_idx.columns =['dow']
```

[78]:
```
#stocks_ret = data.pct_change()
stocks_ret = np.log(data / data.shift(1))
stocks_ret.dropna(inplace=True)

stocks_ret .head()
```

[78]:
```
              AAPL     AXP      BA     CAT    CSCO     CVX     DIS      GS  \
Date
2010-01-05  0.0017 -0.0022  0.0322 0.0119 -0.0045  0.0071 -0.0025  0.0175
2010-01-06 -0.0160  0.0160  0.0299 0.0030 -0.0065  0.0001 -0.0053 -0.0107
2010-01-07 -0.0019  0.0117  0.0397 0.0040  0.0045 -0.0038  0.0003  0.0194
2010-01-08  0.0066 -0.0007 -0.0097 0.0112  0.0053  0.0018  0.0016 -0.0191
2010-01-11 -0.0089 -0.0115 -0.0119 0.0609 -0.0028  0.0176 -0.0164 -0.0159

                HD     IBM  …     NKE     PFE      PG     TRV     UNH  \
Date                        …
2010-01-05  0.0073 -0.0122  …  0.0040 -0.0144  0.0003 -0.0240 -0.0016
2010-01-06 -0.0035 -0.0065  … -0.0061 -0.0032 -0.0048 -0.0143  0.0098
2010-01-07  0.0117 -0.0035  …  0.0098 -0.0038 -0.0054  0.0143  0.0377
2010-01-08 -0.0048  0.0100  … -0.0020  0.0081 -0.0013 -0.0014 -0.0094
2010-01-11 -0.0287 -0.0105  … -0.0124  0.0080 -0.0040 -0.0004  0.0067

                 V      VZ     WBA     WMT     XOM
Date
2010-01-05 -0.0115  0.0018 -0.0081 -0.0100  0.0039
2010-01-06 -0.0135 -0.0435 -0.0076 -0.0022  0.0086
2010-01-07  0.0093 -0.0060  0.0060  0.0006 -0.0031
2010-01-08  0.0028  0.0006  0.0014 -0.0050 -0.0040
2010-01-11 -0.0029  0.0041  0.0016  0.0164  0.0112

[5 rows x 28 columns]
```

[86]:
```
# USING SKLEARN
sklearn_pca = sklearnPCA(n_components=10) # let's look at the first 20␣
 ↪components
pc = sklearn_pca.fit_transform(stocks_ret)

# plot the variance explained by pcs
plt.bar(range(10),sklearn_pca.explained_variance_ratio_)
plt.title('variance explained by pc')
plt.show()
```

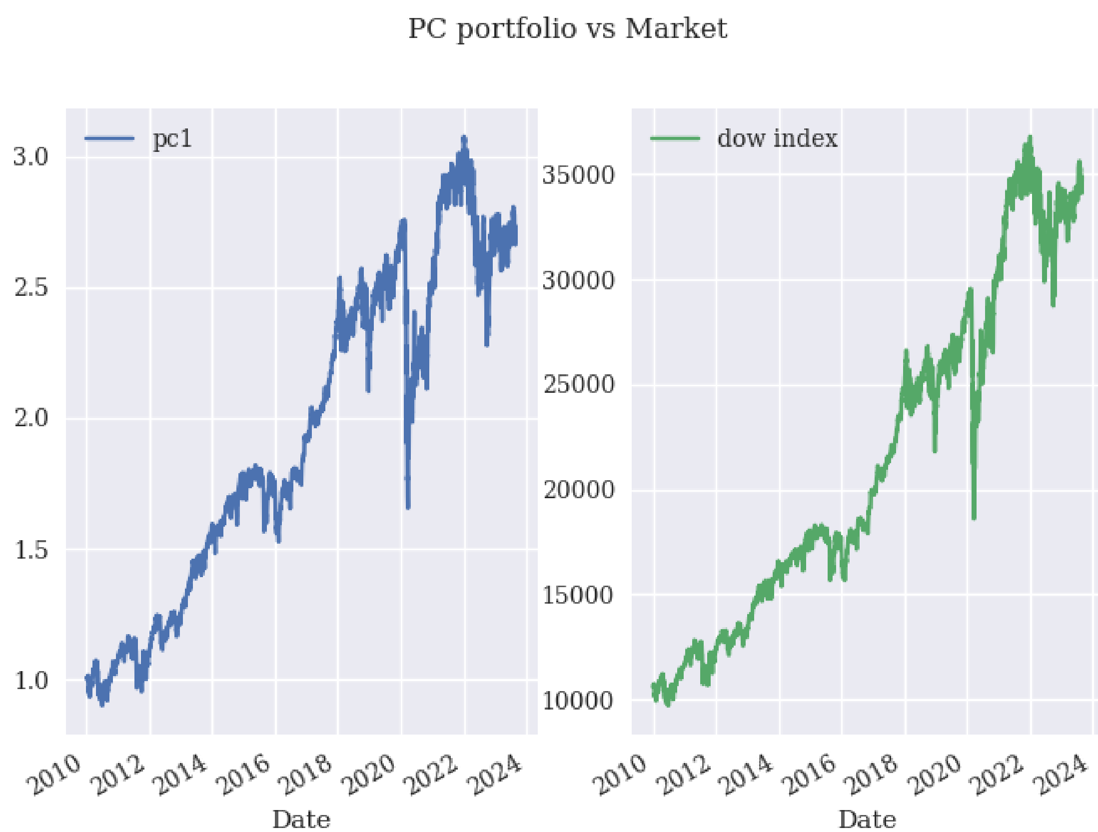


[80]:
```
# check the explained variance reatio
sklearn_pca.explained_variance_ratio_
```

[80]:
```
array([0.4702, 0.0581, 0.0456, 0.0359, 0.0335, 0.0318, 0.0273, 0.0252,
       0.0231, 0.0216, 0.0202, 0.0196, 0.0181, 0.018 , 0.0155, 0.0144,
       0.0144, 0.0136, 0.0129, 0.012 ])
```

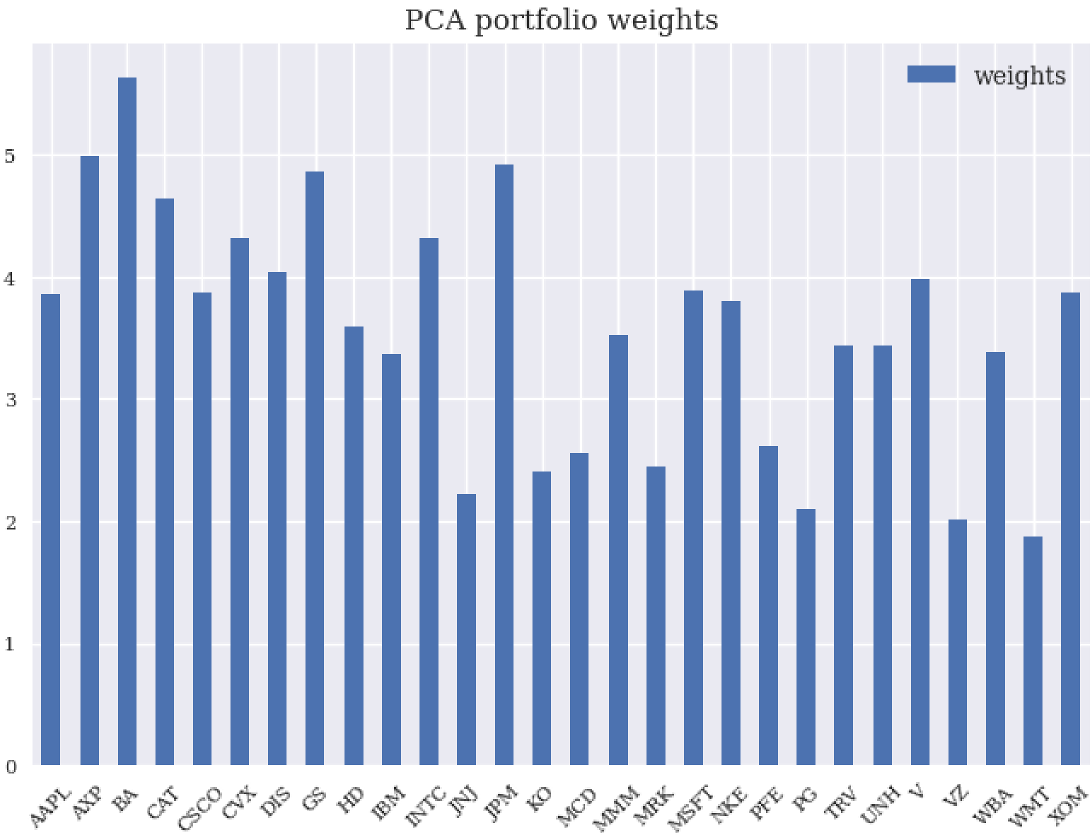
[81]:
```
# get the Principal components
pcs =sklearn_pca.components_

#first component
pc1 = pcs[0,:]

# normalized to 1
pc_w = np.asmatrix(pc1/sum(pc1)).T

# apply our first componenet as weight of the stocks
pc1_ret = stocks_ret.values*pc_w

# plot the total return index of the first PC portfolio
pc_ret = pd.DataFrame(data =pc1_ret, index= stocks_ret.index)
```

```
pc_ret_idx = pc_ret+1
pc_ret_idx= pc_ret_idx.cumprod()
pc_ret_idx.columns =['pc1']

pc_ret_idx['dow index'] = dow_index[1:]
pc_ret_idx.plot(subplots=True,title ='PC portfolio vs Market',layout =[1,2])
plt.show()
```



The weights of the equities in our portfolio are:

[82]:
```
# plot the weights in the PC
weights_df = pd.DataFrame(data = pc_w*100,index = data.columns)
weights_df.columns=['weights']
weights_df.plot.bar(title='PCA portfolio weights',rot =45,fontsize =8)
plt.show()
```


PCA portfolio weights
weights
0
1
2
3
4
5
AAPL
AXP
BA
CAT
CSCO
CVX
DIS
GS
HD
IBM
INTC
JNJ
JPM
KO
MCD
MMM
MRK
MSFT
NKE
PFE
PG
TRV
UNH
V
VZ
WBA
WMT
XOM