PCA-on-Financial-Market-in-2020

July 7, 2021

1 Introduction

A hypothesis behind econometric analysis is that financial market is multivariate. However, this experiment intends to challenge this hypothesis, as COVID-19 influenced almost every part of our lives in 2020. Because one single factor (COVID-19) caused very powerful impact on global economy, price fluctuation of global stock market may largely be explained by COVID-19 spread rate. I do not intend to hypothesize that 100% of variance can be explained by the spread of COVID-19; however, I would like to assume that an analysis could explain approximately 80% of variance. Because I want to understand correlation between market indices and COVID-19 confirmed rate. I would like to combine a number of data and would like to calculate explained rate of variance in each principal component.

2 Methodology

2.1 Explained Variance of First Principal Component

In PCA, $\mathbf{W}, \mathbf{Z}$ can be found using either an eigenvalue decomposition or the SVD. Suppose the matrix $\mathbf{X}$ is centered so that

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i = \vec{0} \in \mathbb{R}^p$$

The covariance matrix is defined to be $\mathbf{C} = \mathbf{X}\mathbf{X}^T$. The eigen-decomposition of $\mathbf{C} = \mathbf{W}\ \mathbf{W}^T$ gives the matrices $\mathbf{W}$ and $\mathbf{Z} = \ \mathbf{W}^T$. Here

$$= \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_p^2 \end{bmatrix}$$

is the diagonal matrix with eigenvalues σ_i^2. The proportion of explained variance by our low-dimensional projection is defined to be

$$\mathrm{PV}(q) := \frac{\sum_{i=1}^{q}\sigma_i^2}{\sum_{i=1}^{p}\sigma_i^2}$$

I suppose that 80% of variance could be explained by the first principal component. In other words, I suppose that 40 global stock indices and COVID-19 confirmed rate of corresponding country may move together by the degree of 80%.

3 Experiment

3.1 Import Libraries and Define Required Functions

[1]:
```
# Load Packages

import yfinance as yf
import csv
import pandas as pd
import numpy as np
from datetime import datetime, date, time, timedelta
from countryinfo import CountryInfo
from sklearn.decomposition import PCA
from datapackage import Package

today = datetime.today()
yesterday = str(today - timedelta(2))[:10]

# apply the z-score method in Pandas using the .mean() and .std() methods
def z_score(df):
    # copy the dataframe
    df_std = df.copy()
    # apply the z-score method
    for column in df_std.columns:
        df_std[column] = (df_std[column] - df_std[column].mean()) /
 ↪df_std[column].std()

    return df_std

# Convert Date
def date_convert(dates):
    dates_return = []

    for date in dates:
        date = date.split("/")
        year = '20' + str(date[2])
        month = str(date[0])
        day = str(date[1])

        if int(month) < 10:
            month = '0' + month

        if int(day) < 10:
```

```
            day = '0' + day

        date = year + "-" + month + "-" + day
        dates_return.append(date)

    return dates_return

start = "2020-01-22"
end = "2021-01-01"

# Matplotlib
import matplotlib.pyplot as plt
pd.set_option('max_rows', 500)
pd.set_option('max_columns', 500)
np.set_printoptions(suppress=True)

%matplotlib inline
plt.rcParams["figure.figsize"] = (16, 12)
plt.style.use('seaborn-pastel')
plt.rcParams['lines.linewidth'] = 1
plt.figure(dpi=300)
plt.rcParams['lines.color'] = 'b'
plt.rcParams['axes.grid'] = True
plt.tight_layout()
```

```
<Figure size 4800x3600 with 0 Axes>
```

3.2 Data Import

3.2.1 Import Stock Indices

This part of code imports **40 Market Indices** that represent major financial market in the globe.

[2]:
```
# Import 40 Market Indices

BUK100P = yf.download("^BUK100P", start, end)['Adj Close'].to_frame()
SPY = yf.download("SPY", start, end)['Adj Close'].to_frame()
Singapore = yf.download("^STI", start, end)['Adj Close'].to_frame()
Dow = yf.download("^DJI", start, end)['Adj Close'].to_frame()
Nasdaq = yf.download("^IXIC", start, end)['Adj Close'].to_frame()
FTSE100 = yf.download("^FTSE", start, end)['Adj Close'].to_frame()
FTSE250 = yf.download("^FTSE", start, end)['Adj Close'].to_frame()
FTSE350 = yf.download("^FTLC", start, end)['Adj Close'].to_frame()
FTAI = yf.download("^FTAI", start, end)['Adj Close'].to_frame()
N225 = yf.download("^N225", start, end)['Adj Close'].to_frame()
N500 = yf.download("^N500", start, end)['Adj Close'].to_frame()
N1000 = yf.download("^N1000", start, end)['Adj Close'].to_frame()
HSI = yf.download("^HSI", start, end)['Adj Close'].to_frame()
```

```
Taiwan = yf.download("^TWII", start, end)['Adj Close'].to_frame()
SSE = yf.download("000001.SS", start, end)['Adj Close'].to_frame()
Shenzhen = yf.download("399001.SZ", start, end)['Adj Close'].to_frame()
DAX = yf.download("^GDAXI", start, end)['Adj Close'].to_frame()
France = yf.download("^FCHI", start, end)['Adj Close'].to_frame()
Indonesia = yf.download("^JKSE", start, end)['Adj Close'].to_frame()
PSEI = yf.download("PSEI.PS", start, end)['Adj Close'].to_frame()
AORD = yf.download("^AORD", start, end)['Adj Close'].to_frame()
AXJO = yf.download("^AXJO", start, end)['Adj Close'].to_frame()
AXKO = yf.download("^AXKO", start, end)['Adj Close'].to_frame()
kospi = yf.download("^KS11", start, end)['Adj Close'].to_frame()
India = yf.download("^BSESN", start, end)['Adj Close'].to_frame()
NZ50 = yf.download("^NZ50", start, end)['Adj Close'].to_frame()
XAX = yf.download("^XAX", start, end)['Adj Close'].to_frame()
RUI = yf.download("^RUI", start, end)['Adj Close'].to_frame()
RUT = yf.download("^RUT", start, end)['Adj Close'].to_frame()
RUA = yf.download("^RUA", start, end)['Adj Close'].to_frame()
GSPTSE = yf.download("^GSPTSE", start, end)['Adj Close'].to_frame()
N100 = yf.download("^N100", start, end)['Adj Close'].to_frame()
N150 = yf.download("^N150", start, end)['Adj Close'].to_frame()
BFX = yf.download("^BFX", start, end)['Adj Close'].to_frame()
IMOEX = yf.download("IMOEX.ME", start, end)['Adj Close'].to_frame()
MERV = yf.download("^MERV", start, end)['Adj Close'].to_frame()
TA125 = yf.download("^TA125.TA", start, end)['Adj Close'].to_frame()
JNOU = yf.download("^JNOU.JO", start, end)['Adj Close'].to_frame()
AEX = yf.download("^AEX", start, end)['Adj Close'].to_frame()
ATOI = yf.download("^ATOI", start, end)['Adj Close'].to_frame()
BVSP = yf.download("^BVSP", start, end)['Adj Close'].to_frame()
MIB = yf.download("FTSEMIB.MI", start, end)['Adj Close'].to_frame()
ATX = yf.download("^ATX", start, end)['Adj Close'].to_frame()
ISEQ = yf.download("^ISEQ", start, end)['Adj Close'].to_frame()
NSEI = yf.download("^NSEI", start, end)['Adj Close'].to_frame()
MXX = yf.download("^MXX", start, end)['Adj Close'].to_frame()
SSMI = yf.download("^SSMI", start, end)['Adj Close'].to_frame()
STOXX50E = yf.download("^STOXX50E", start, end)['Adj Close'].to_frame()
MDAXI = yf.download("^MDAXI", start, end)['Adj Close'].to_frame()
SDAXI = yf.download("^SDAXI", start, end)['Adj Close'].to_frame()
HSCC = yf.download("^HSCC", start, end)['Adj Close'].to_frame()
HSCE = yf.download("^HSCE", start, end)['Adj Close'].to_frame()
KLSE = yf.download("^KLSE", start, end)['Adj Close'].to_frame()

# Transform into Dataframe
df = pd.concat([
    BUK100P, Dow, Nasdaq, FTSE100, FTSE250, FTAI, N225, SSE, Shenzhen,
    DAX, France, Indonesia, PSEI, AXKO, kospi, NZ50, RUI, RUT, RUA,
    GSPTSE, N100, N150, BFX, IMOEX, MERV, TA125, JNOU, SPY, Singapore,
    AEX, ATOI, BVSP, MIB, ATX, ISEQ, MXX, STOXX50E, MDAXI, SDAXI, KLSE],
```

```
    axis=1
)

# Set Columns
# Columns include Country Code, so that I can match country to COVID-19␣
↪confirmed rate.
df.columns=[
    'US-Dow', 'US-Nasdaq', 'GB-FTSE100', 'GB-FTSE250', 'GB-FTAI',
    'GB-BUK100P', 'JP-N225', 'CN-SSE', 'CN-Shenzhen', 'DE-DAX',
    'FR-FCHI', 'ID-JKSE', 'PH-PSEI', 'AU-AXKO', 'KR-KSII', 'NZ-NZ50',
    'US-RUI', 'US-RUT', 'US-RUA', 'CA-GSPTSE', 'FR-N100', 'FR-N150',
    'BE-BFS', 'RU-IMOEX', 'AR-MERV', 'IL-TA125', 'ZA-JNOU', 'US-SPX',
    'SG-STI', 'NL-AEX', 'AU-ATOI', 'BR-BVSP', 'IT-MIB', 'AT-ATX', 'IE-ISEQ',
    'MX-MXX', 'DE-Stoxx50E', 'DE-MDAXI', 'DE-SDAXI', 'MY-KLSE']

# Eliminate Missing Values
daily_return = df.fillna(method='ffill').fillna(method='bfill')

# Normalize Data
daily_return = z_score(daily_return)

# Copy it for the future use
daily_return1 = daily_return
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

```
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
[*********************100%***********************]  1 of 1 completed
```

3.2.2 COVID-19 Confirmed Data

This part of code imports **corresponding COVID-19 Confirmed Rate**.

[3]:
```python
# COVID-19 Dataset

states_url = "https://covidtracking.com/api/states/daily"
us_url = "https://covidtracking.com/api/us/daily"
case_threshold = 100

cases = ["confirmed", "deaths", "recovered"]
sheet = "https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/
 ↪csse_covid_19_data/csse_covid_19_time_series/time_series_covid19_"
suffix = "_global.csv"
df_list = []

url_confirmed = sheet + "confirmed" + suffix
```

```python
df_confirmed = pd.read_csv(url_confirmed, header=0, escapechar="\\")
df_confirmed1 = df_confirmed.drop(columns=["Lat", "Long"])
df_confirmed = df_confirmed.drop(columns=["Lat", "Long"])

df_confirmed = df_confirmed.groupby("Country/Region").agg("sum").T
df_confirmed1 = df_confirmed1.groupby("Province/State").agg("sum").T

# Preprocess Data
dates = df_confirmed.index.tolist()
dates = date_convert(dates)
US = df_confirmed["US"].tolist()
China = df_confirmed["China"].tolist()
Germany = df_confirmed["Germany"].tolist()
Japan = df_confirmed["Japan"].tolist()
UK = df_confirmed["United Kingdom"].tolist()
Korea = df_confirmed["Korea, South"].tolist()
Australia = df_confirmed["Australia"].tolist()
Austria = df_confirmed["Austria"].tolist()
Denmark = df_confirmed["Denmark"].tolist()
Greece = df_confirmed["Greece"].tolist()
Finland = df_confirmed["Finland"].tolist()
Ireland = df_confirmed["Ireland"].tolist()
Italy = df_confirmed["Italy"].tolist()
SouthAfrica = df_confirmed["South Africa"].tolist()
Spain = df_confirmed["Spain"].tolist()
Singapore = df_confirmed["Singapore"].tolist()
Russia = df_confirmed["Russia"].tolist()
NewZealand = df_confirmed["New Zealand"].tolist()
Canada = df_confirmed["Canada"].tolist()
France = df_confirmed["France"].tolist()
Netherlands = df_confirmed["Netherlands"].tolist()
Mexico = df_confirmed["Mexico"].tolist()
Brazil = df_confirmed["Brazil"].tolist()
Philippines = df_confirmed["Philippines"].tolist()
India = df_confirmed["India"].tolist()
Argentina = df_confirmed["Argentina"].tolist()
Indonesia = df_confirmed["Indonesia"].tolist()
Malaysia = df_confirmed["Malaysia"].tolist()
Israel = df_confirmed["Israel"].tolist()
Poland = df_confirmed["Poland"].tolist()
Afghanistan = df_confirmed["Afghanistan"].tolist()

data = [
    US, China, Japan,
    Korea, Australia, Austria,
    Germany, UK, Denmark,
    Greece, Italy, SouthAfrica,
```

```
    Spain, Singapore, Russia,
    NewZealand, Canada, France,
    Netherlands, Mexico, Philippines,
    India, Argentina, Indonesia,
    Malaysia, Israel, Poland,
    Brazil, Spain
]

# Country Codes
country_codes = [
    "US", "CN", "JP",
    "KR", "AU", "AT",
    "DE", "GB", "DK",
    "GR", "IT", "ZA",
    "ES", "SG", "RU",
    "NZ", "CA", "FR",
    "NL", "MX", "PH",
    "IN", "AR", "ID",
    "MY", "IL", "PL",
    "BR", "ES"
]

daily_confirmed = pd.DataFrame(data, index=country_codes, columns=dates).T
daily_confirmed = z_score(daily_confirmed)

for code in country_codes:
    population = CountryInfo(code).population()
    daily_confirmed[code] = daily_confirmed[code].div(population, axis=0)

daily_confirmed.index.name = 'Date'
daily_confirmed1 = daily_confirmed
```

3.3 Merge Dataframe

Because I need to **merge dataframe** to apply Principal Component Analysis as a means of calculating explained variance, I now merge two dataframes: **Daily Return** and **Daily Confirmed**.

[4]:
```
# List of Dates
confirmed = daily_confirmed.index.tolist()
returns = daily_return.index.tolist()

# Build a list to include dates in common
dates_common = []
for date in returns:
    date = (str(date)[:10])
    if date in confirmed:
        dates_common.append(date)
```

```python
# Only leave dates in common from daily_confirmed
for date in daily_confirmed.index:
    if date not in dates_common:
        daily_confirmed = daily_confirmed.drop(date)

# Only leave dates in common from daily_return
daily_return_index = []
for var in daily_return.index.tolist():
    date = (str(var))[:10]
    if date not in dates_common:
        daily_return = daily_return.drop(var)

    else:
        daily_return_index.append(str(date))

daily_return.index = daily_return_index
daily_return.index.name = 'Date'

# Now, merge them in same index
df_merged = pd.concat([daily_return, daily_confirmed], axis=1)

# Normalize
df_merged = z_score(df_merged)
```

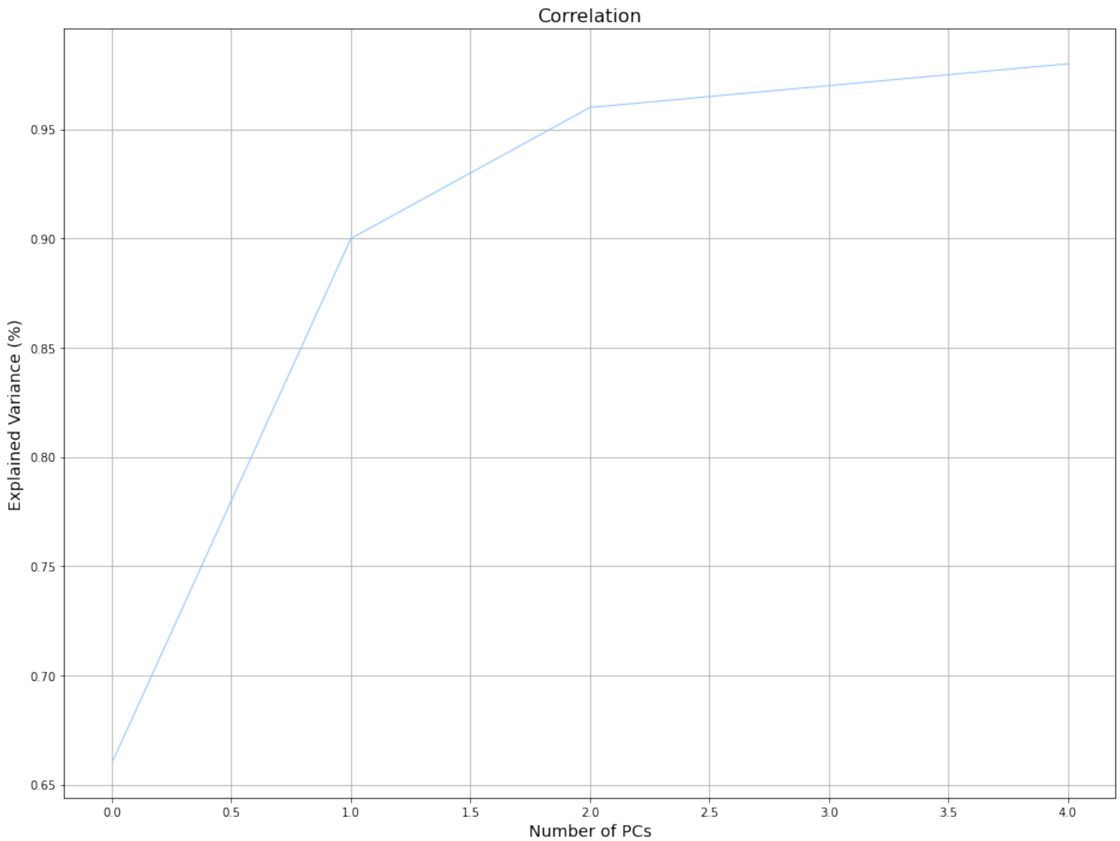
4 Principal Component Analysis

4.1 Extraction of Five Principal Components

I extract first 5 principal components. They represent how a limited number of principal components, which are orthogonal combinations of vectors, can explain total variance.

[5]:
```python
pca = PCA(5).fit(df_merged)
daily_return_factors = pd.Series(index=df_merged.columns, data=pca.
 ↪components_[0])
print("TOP 5 Principal Components:", pca.explained_variance_ratio_.round(2))
variance_stock = pca.explained_variance_ratio_.cumsum().round(2)
plt.plot(variance_stock)
plt.title('Correlation', fontsize = 16)
plt.xlabel('Number of PCs', fontsize = 14)
plt.ylabel('Explained Variance (%)', fontsize = 14)
plt.show()
```

```
TOP 5 Principal Components: [0.66 0.24 0.06 0.01 0.01]
```



First principal component explains 66% of entire variance, which means H_0 is accepted. If we take second principal component into account, it combinedly explains 80% of variance. It appears that COVID-19 is very powerful variable to explain financial market's fluctuation. However, it appears that it still does not explain 80% of entire variance, which means that its impact is still limited to challenge multivariate assumption.

5 Comparison of Eigenvalues

While explained variance methodology proves that COVID-19 Confirmed Rate is very limited variable to explain financial fluctuation alone, I still believe

5.1 Eigenvalues

Principal Component Analysis is essentially about reducing complex dimensionality of data by taking **orthogonal transformation** of vectors. Each loading of principal component, which is `eigenvalue` whose sum of entire eigenvalues is equal to 1, as `unit vector`.

$$u := \min(\frac{1}{n}\sum_{i}^{n}(x_i^T x_i - (u_1^T x_i)^2))$$

Principal Component Analysis aims for minimizing total distance of a unit vector whose perpendicular distance is minimized as a result. And it is the eigenvector of the covariance matrix of X.

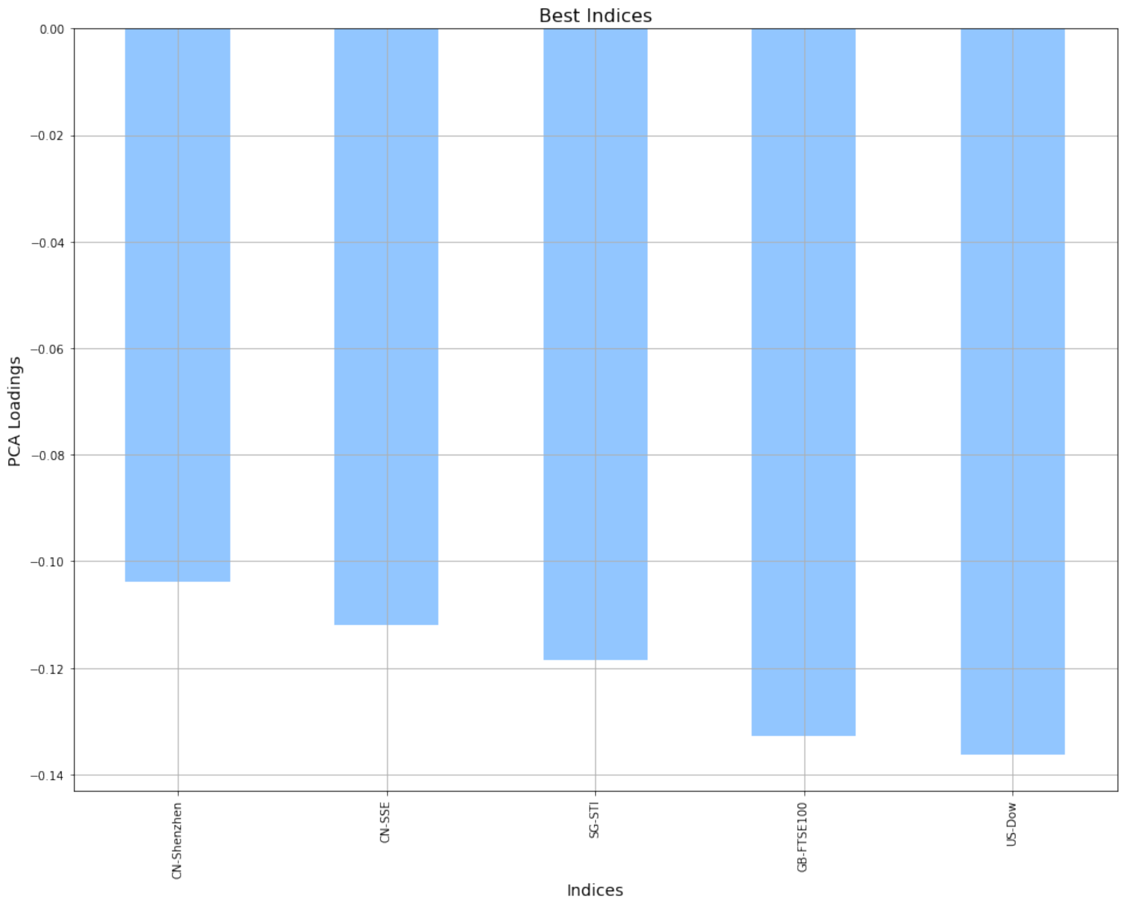
$$Av = \lambda v$$

5.2 PCA Loading: "Degree of Impact"

Loading of first principal component could reveal how variables affected price fluctuation in 2020. Because first principal component, as proven above, explains nearly 70% of entire variance, I would like to assume that proportion of eigenvalues suggests how each indice was influenced.

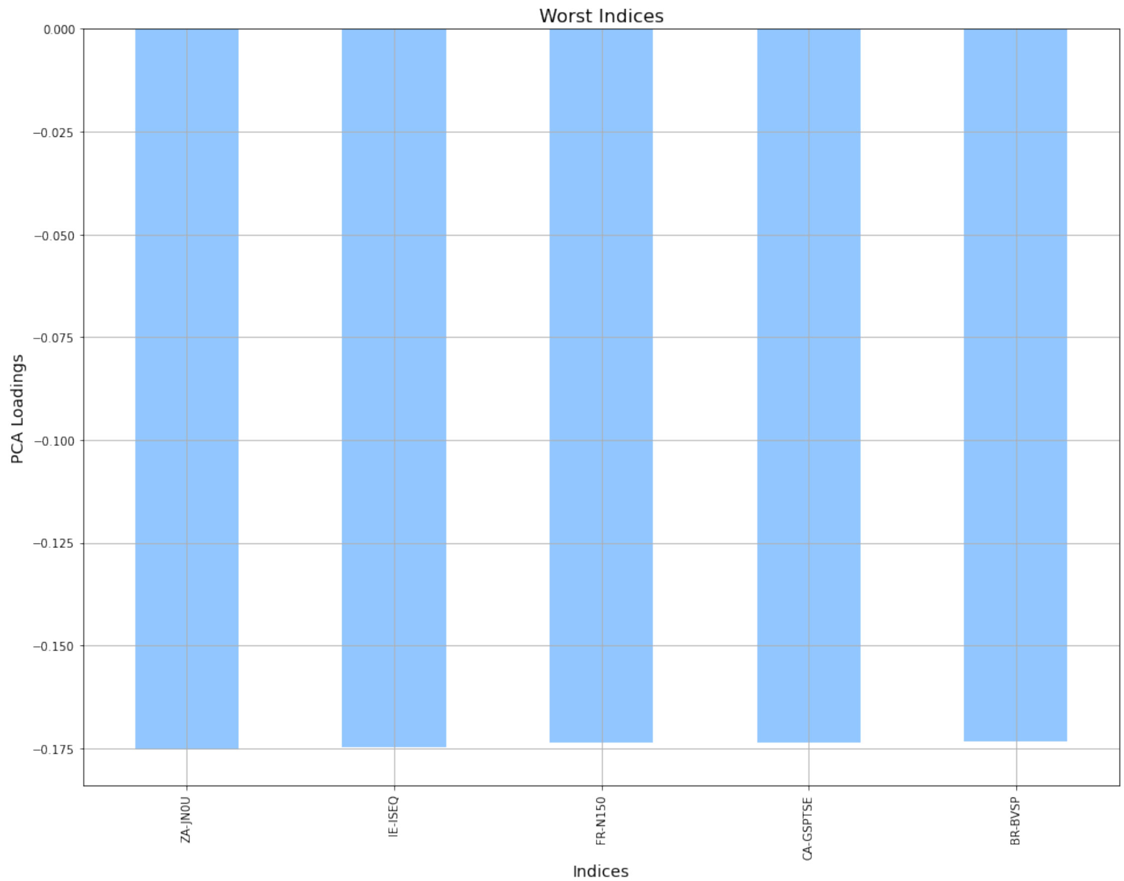
[6]:
```
pca_dr = PCA(1).fit(daily_return1)
daily_return_factors = pd.Series(index=daily_return1.columns, data=pca_dr.
 ↪components_[0])

daily_return_factors.nlargest(5).plot.bar()
plt.title('Best Indices', fontsize=16)
plt.xlabel('Indices', fontsize=14)
plt.ylabel('PCA Loadings', fontsize = 14)
# plt.savefig('Best Indices.png', dpi=300)
```

[6]:
```
Text(0, 0.5, 'PCA Loadings')
```



[7]:
```
daily_return_factors.nsmallest(5).plot.bar()
plt.title('Worst Indices', fontsize=16)
plt.xlabel('Indices', fontsize=14)
plt.ylabel('PCA Loadings', fontsize = 14)
```

[7]: `Text(0, 0.5, 'PCA Loadings')`



6 Conclusion

While uni-variate model could not explain enough variance, it still mathematically proves that COVID-19 has very strong impact on financial price fluctuation. It appears that countries with strong capacity to enforce powerful monetary policy, such as United States and Great Britain, could change the trend.

[9]:
```
print("Most Positively Influenced Indices Are:")
print(daily_return_factors.nlargest(5).index.tolist())
print("")
print("Degree of Impact is", daily_return_factors.nlargest(5).values.tolist())
print("")

print("Most Negatively Influenced Indices Are:")
print(daily_return_factors.nsmallest(5).index.tolist())
print("")
print("Degree of Impact is", daily_return_factors.nsmallest(5).values.tolist())
print("")
```

```
Most Positively Influenced Indices Are:
['CN-Shenzhen', 'CN-SSE', 'SG-STI', 'GB-FTSE100', 'US-Dow']

Degree of Impact is [-0.10386905041725426, -0.11185218241949893,
-0.11849356397466648, -0.13284897201600882, -0.13619868522005837]

Most Negatively Influenced Indices Are:
['ZA-JNOU', 'IE-ISEQ', 'FR-N150', 'CA-GSPTSE', 'BR-BVSP']

Degree of Impact is [-0.1751972802144581, -0.17478189737240282,
-0.1737130864090994, -0.17355907590772135, -0.17322922178860994]
```

Five most **positively** influenced indices belong to China, Singapore, Great Britain, and United States. Five most **negatively** influenced indices belong to South Africa, Spain, France, Canada, and Brazil. Top 5 indices mostly belong to countries who contained COVID-19 well whereas worst 5 ones mostly belong to those who could not control spread of COVID-19 well. However, it appears that **Great Britain** and **United States** are anomalies, as they are most negatively influenced countries with very powerful spread of COVID-19 while they are still listed as best 5 indices. It appears to align with the result of hypothesis testing, as COVID-19 confirmed rate could explain most of indices while failing to explain two countries.