PCA-on-Financial-Market-in-2020

May 31, 2021

1 Motivation

It is explicit, unarguable truth that Coronavirus has been the greatest macroeconomic variable that affected society. While shutdown collapsed supply and demand of labor market, it influenced price fluctuation of financial market. March in 2020 has been an inflection point of asset price; a price dropped in March, and it soon rose in an increasing pace. My interest in this experiment is to verify how big is the impact of Coronavirus on asset price fluctuation. In social science, it is highly extremely unfrequent to have a linear model that a single Δx affects Δy. Economists always build multivariate models to explain econometric situations. But it is my expectation that a single variable may build precise model that expects the price fluctuation. The goal of this experiment is, on the surface, to learn the degree of impact that Coronavirus has made on price fluctuation in asset market. However, in essential, I am intending to make an example that even a single input variable may result in powerful linear model to predict price fluctuation.

2 Literature Review

An idea behind this methodology was given by the paper written by **Kritzman et al. (2010)**: 'Principal Components as a Measure of Systemic Risk' [1]. It introduces the concept of Principal Component Analysis to understand systemic risk of financial market. **Systemic Risk** refers to a ratio of systematic risk to idiosyncratic risk, and principal component analysis extracts principal components, or simplified eigenvectors, as orthogonal transformation of correlated features. That being said, larger **explained variance** of first two principal components implies that market is very strongly correlated; market fragility is warned. If there is a single problem that occurs in the part of economy, market fragility indicated by larger explained variance implies its impact will be much bigger. As a quote, Kritzman et al. (2010) states that:

> "the absorption ratio, which equals the fraction of the total variance of a set of asset returns explained or "absorbed" by a fixed number of eigenvectors. The absorption ratio captures the extent to which markets are unified or tightly coupled. When markets are tightly coupled, they are more fragile in the sense that negative shocks propagate more quickly and broadly than when markets are loosely linked". It is my intention that I take principal components from data 'COVID-19 Confirmed' and 'Daily Return' so that we observe how much variance is explained by first principal component. I will extend the principal component into five, so that I can check the trend of the degree of explained variance in detail.

3 Methodology

While it is my goal to verify the impact of COVID-19 on price fluctuation, I will conduct two experiments. First of all, as an application of the idea of market fragility, I intend to calculate **explained variance** of a first principal component to learn how correlated variables are. Data that I intend to correlate is a combination of **Coronavirus Confirmed Rate** and **Price Fluctuation in 39 Global Market Indices**. If explained variance is high, it means the confirmed rate of COVID-19 and price fluctuation is highly correlated. Second, I will take Data as the value of first principal component (eigenvector), or eigenvalue, as a value to measure how strongly each index was influenced.

4 Null Hypothesis (H_0)

My null hypothesis is that countries whose COVID-19 fluctuation is stronger has stronger market fluctuation, and the direction of it is identical.

Let change in confirmed covid-19 as C and change in Price Fluctuation as P,

$$H_0 : \forall\ C > 0,\ \text{then } P > 0 \text{ and } C < 0,\ \text{then } P < 0$$

5 Principal Component Analysis

5.1 Load Packages and Define Functions

- Load Libraries
- Functions `z_score` and `date_convert` as something *pre-defined*.

In [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>
```

5.2 Data Import

5.2.1 Stock Indices

This part of code imports **39 Market Indices** that represent major financial market in the globe.

```
In [2]: # Import Market Indices
        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([
    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
df.columns=[
    'US-Dow',
    'US-Nasdaq',
    'GB-FTSE100',
    'GB-FTSE250',
    'GB-FTAI',
    '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
```

5.2.2 COVID-19 Confirmed Data

```
In [3]: # 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
        suffix = "_global.csv"
        df_list = []

        url_confirmed = sheet + "confirmed" + suffix

        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
```

5.2.3 Merge Dataframe

We merge two dataframe, which are **Daily Return** and **Daily Confirmed**.

```
In [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)

# 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)
```

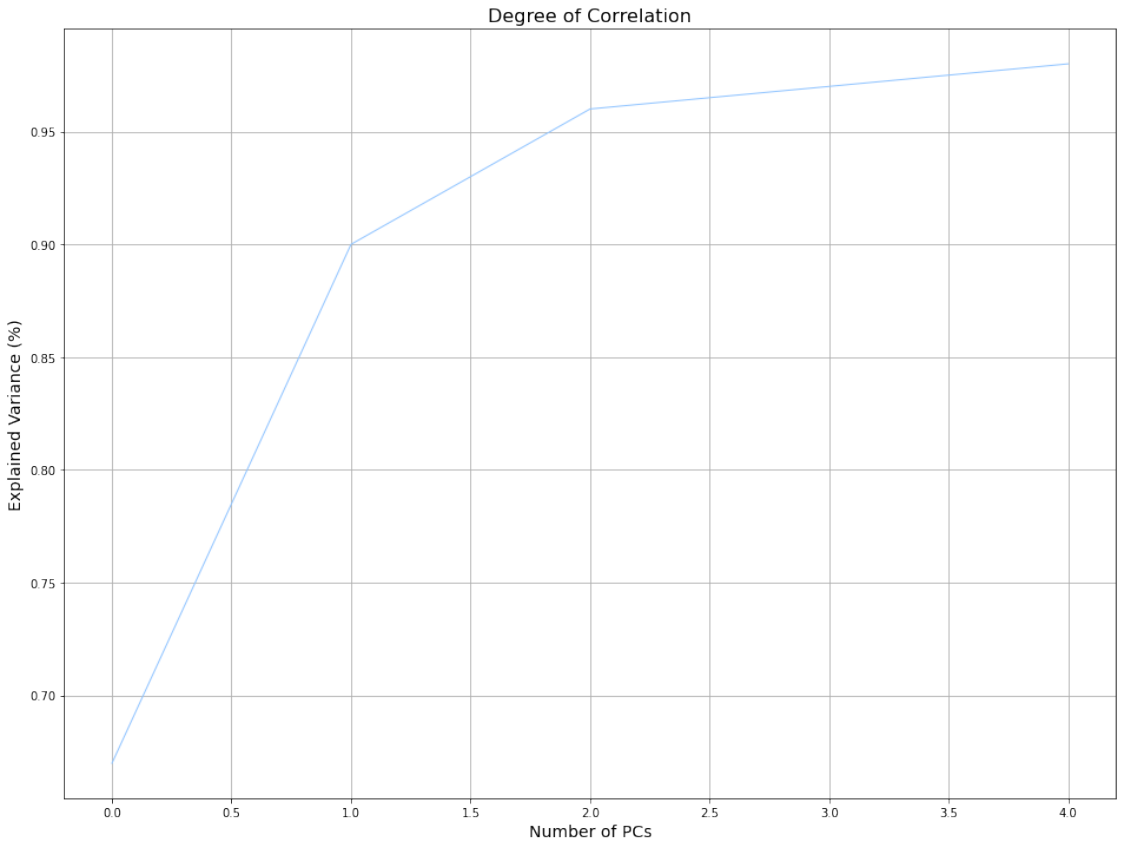
5.3 Extraction of Five Principal Components

```
In [5]: pca = PCA(5).fit(df_merged)
        daily_return_factors = pd.Series(index=df_merged.columns, data=pca.components_[0])
        print("Principal Components", pca.explained_variance_ratio_.round(2))
```

```
Principal Components [0.67 0.23 0.06 0.01 0.01]
```

5.4 Visualization of Explanatory Power for Each Principal Component

```
In [6]: variance_stock = PCA(5).fit(df_merged).explained_variance_ratio_.cumsum().round(2)
        plt.plot(variance_stock)
        plt.title('Degree of Correlation', fontsize = 16)
        plt.xlabel('Number of PCs', fontsize = 14)
        plt.ylabel('Explained Variance (%)', fontsize = 14)
        plt.show()
```



It appears that there is **a significant degree** of corrrelation. A single principal component explains nearly 70% of entire variance, and top three principal components explain higher than 95% of entire variance, which is enough correlated.

6 Analysis on Eigenvectors

While even a single principal component contains a significant degree of explained variance, I want to take deeper-level analysis of how each financial index was influenced. For this effort, I intend to utilize characteristics of eigenvectors behind Principal Component Analysis.

6.1 Meaning of Eigenvectors

As explained, Principal Component Analysis reduces dimensionality of data by correlating similar vectors.

Intuitively speaking, Principal Component Analysis is a technique to reduce dimensionality. Smaller dimension of features can be achieved by a linear combination of columns, which explain the maximum variation explained. This concept is what we used to understand `correlation` and `systemic risk` in the previous section of analysis. Each Principal Component Loading is an example of `unit vector`.

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

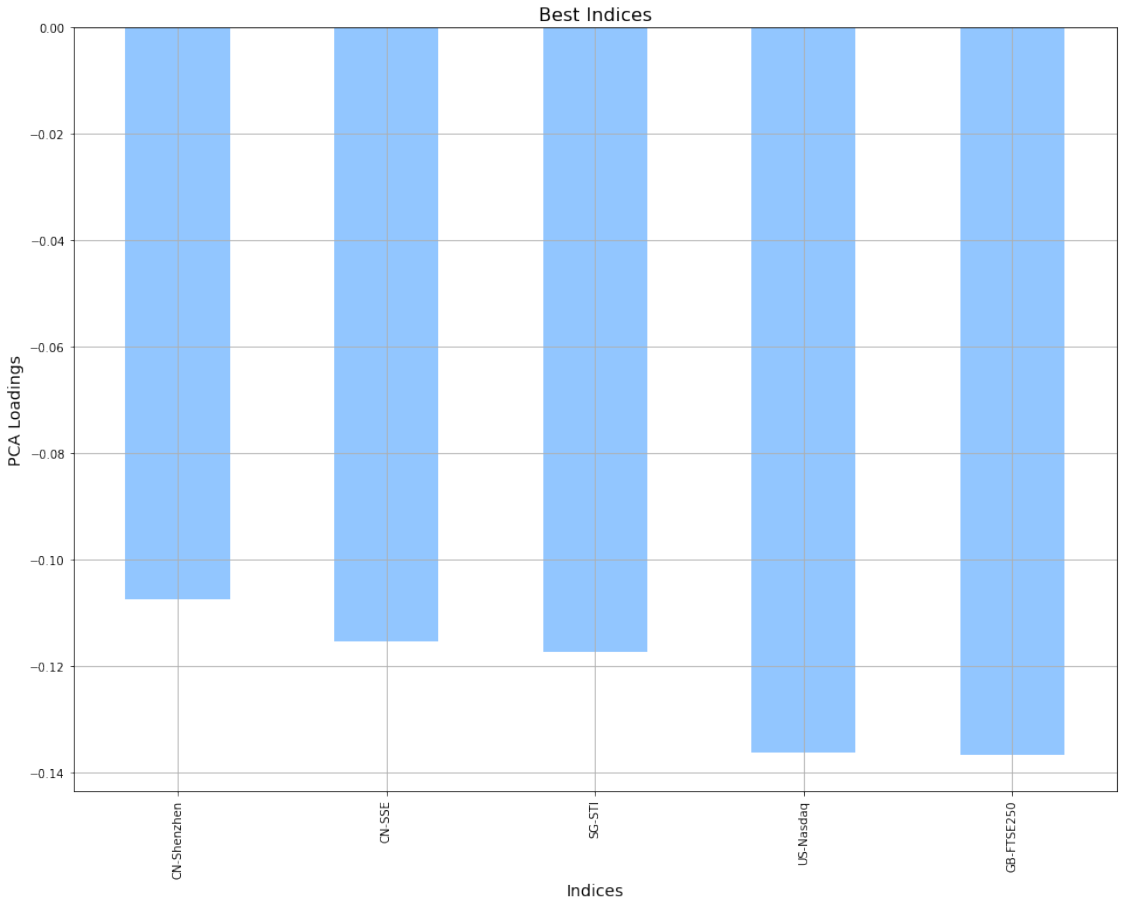
Principal Component Analysis aims for minimizing total distance of a unit vector whose perpendicular distance is minimized as a result. And it is the eigenvector of the covariance matrix of X.

$$Av = \lambda v$$

6.2 PCA Loadings as "Degree of Impact"

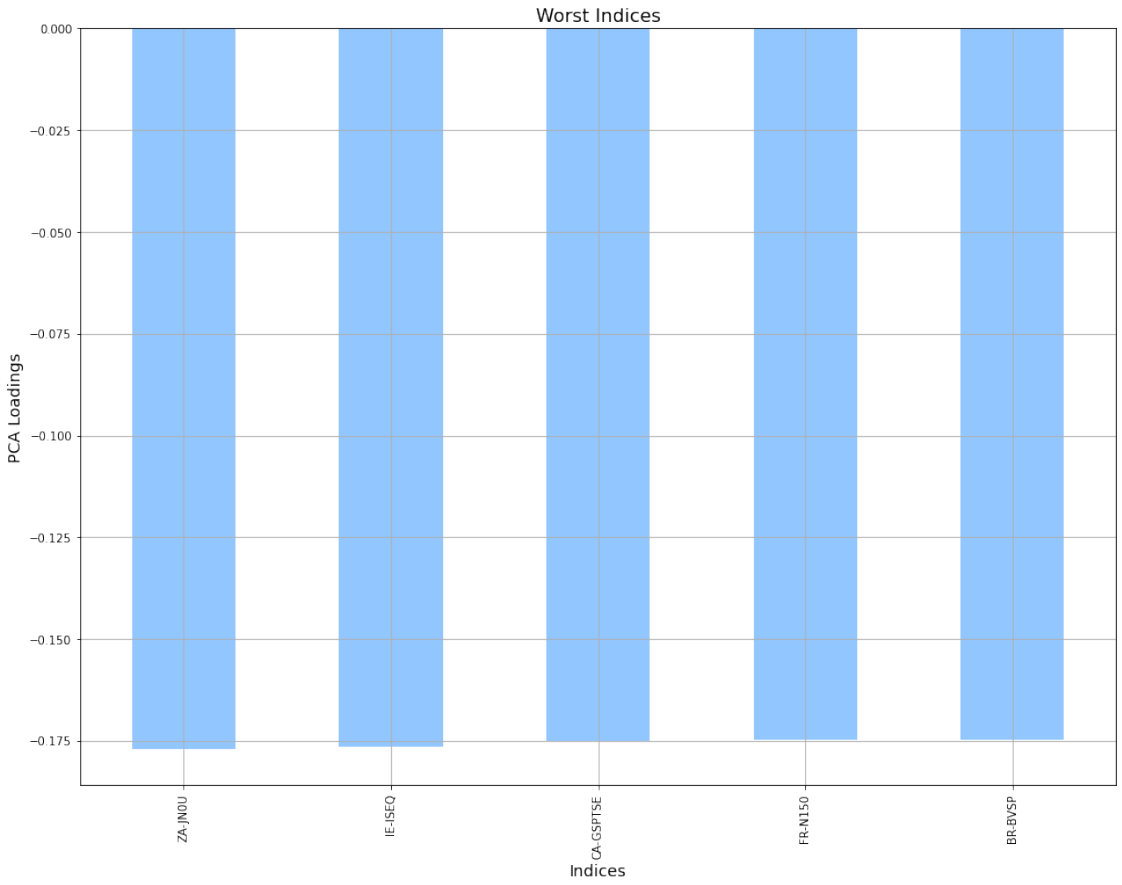
While it is about 'degree of impact', we may verify how large each extent is.

```
In [7]: 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)

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




```
In [8]: daily_return_factors.nsmallest(5).plot.bar()
        plt.title('Worst Indices', fontsize=16)
        plt.xlabel('Indices', fontsize=14)
        plt.ylabel('PCA Loadings', fontsize = 14)
```

```
Out[8]: Text(0, 0.5, 'PCA Loadings')
```



7 Consequence

It appears that Coronavirus Confirmed Rate and Price Fluctuation are highly correlated. First principal component explains nearly 70% of variance, and three principal components explain more than 95% of it. It implies that the spread of Coronavirus is very powerful macroeconomic variable to explain price fluctuation of asset market. It appears that countries who poorly dealt Coronavirus, including South Africa, Canada, France, and Brazil, have lowest eigenvalues while China and Singapore have relatively stronger eigenvalues. However, **`Nasdaq Index` in United States** and **`FTSE Index` in Great Britain**, of the countries who managed COVID-19 the poorest in the world, ironically have strongest eigenvalues. I believe it is another proof that first principal component is not a perfect indicator to explain full variance of market. We need to recognize the truth that there are still other powerful variables, such as Monetary Policy and Fiscal Policy, that impact on market performance. For instance, American market has been stronger than most of other countries partly because of its skillful interference on financial market, whose capability to sedate market panic with injection of dollars as an extended affordability of American to deal with financial loss. Great Britain is another country whose financial institutions are very skillful, and it appears to explain the result correctly. However, the biggest take-away in this experiment is that, we still need multivariate models to explain economic situation precisely. Even in the case of

2020, a period dominated by unprecedentedly dominant, globally-recurring variable of COVID-19 appeared, it cannot explain entire variance of the situation. My H_0 is well rejected with the generalized linear model, and it is very great insight that I could learn from this experiment.

8 Bibliography

[1] Kritzman, Mark and Li, Yuanzhen and Page, Sebastien and Rigobon, Roberto, Principal Components as a Measure of Systemic Risk (June 30, 2010). MIT Sloan Research Paper No. 4785-10, https://doi.org/10.3905/jpm.2011.37.4.112, Available at SSRN: https://ssrn.com/abstract=1633027 or http://dx.doi.org/10.2139/ssrn.1633027