Lecture 4 Index Replication

Douglas Chung

National Chengchi University
8 October 2021

1 Replicate Dow Jones Industrial Average

In this example, we will learn how to replicate the Dow Jones Industrial Average (DJIA) using Yahoo! Finance data. DJIA is a price-weighted index that captures the performance of 30 large cap companies in the US. To begin, we have to import the yfinance package and the pandas package to Python. The first package is for downloading data from Yahoo! Finance while the second package is useful for manipulating data.

[1]:
```
import yfinance as yf
import pandas as pd
```

1.1 Download DJIA from Yahoo! Finance

Using the yf.download function from the yfinance package, we will download DJIA (ticker: ˆDJI) from Yahoo! Finance. We can change the sampling period by modifying start_date and end_date.

[2]:
```
start_date = "2021-07-01"
end_date = "2021-08-31"
idx = "^DJI"

Index = yf.download(idx, # ticker
                    interval="1d", # daily frequency
                    start=start_date, end=end_date) # sampling period
Index.tail() # preview the last five observations
```

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

[2]:
```
                    Open          High           Low         Close  \
Date
2021-08-24  35382.718750  35445.468750  35356.179688  35366.261719
2021-08-25  35388.769531  35501.140625  35287.628906  35405.500000
2021-08-26  35449.679688  35498.449219  35205.101562  35213.121094
2021-08-27  35231.109375  35479.179688  35231.109375  35455.800781
2021-08-30  35471.800781  35510.710938  35374.460938  35399.839844

               Adj Close     Volume
```

```
Date
2021-08-24  35366.261719  228710000
2021-08-25  35405.500000  237230000
2021-08-26  35213.121094  239740000
2021-08-27  35455.800781  240990000
2021-08-30  35399.839844  245390000
```

In particular, we are only interested in the adjusted close price of the index.

```
[3]: Index = pd.DataFrame(Index['Adj Close'].rename(idx)) # select adjusted close␣
      ↪price
     Index.head() # preview the first five observations
```

```
[3]:                     ^DJI
     Date
     2021-06-30  34502.511719
     2021-07-01  34633.531250
     2021-07-02  34786.351562
     2021-07-06  34577.371094
     2021-07-07  34681.789062
```

1.2 Identify the constituents of DJIA

Next, we have to identify the constituents of DJIA. From Wikipedia (https://en.wikipedia.org/wiki/Dow_Jones_Industrial_Average), we can find the 30 large cap companies that are included in DJIA. The pd.read_html() from the pandas package will help us gather information from the Wikipedia page.

```
[4]: page = pd.read_html('https://en.wikipedia.org/wiki/Dow_Jones_Industrial_Average')
     page[1] # the second html table contains the information we need
```

```
[4]:                      Company Exchange Symbol                        Industry  \
     0                         3M     NYSE    MMM                    Conglomerate
     1           American Express     NYSE    AXP              Financial services
     2                      Amgen   NASDAQ   AMGN         Pharmaceutical industry
     3                 Apple Inc.   NASDAQ   AAPL          Information technology
     4                     Boeing     NYSE     BA           Aerospace and defense
     5           Caterpillar Inc.     NYSE    CAT         Construction and Mining
     6        Chevron Corporation     NYSE    CVX              Petroleum industry
     7              Cisco Systems   NASDAQ   CSCO          Information technology
     8      The Coca-Cola Company     NYSE     KO                   Food industry
     9                   Dow Inc.     NYSE    DOW               Chemical industry
     10             Goldman Sachs     NYSE     GS              Financial services
     11            The Home Depot     NYSE     HD                       Retailing
     12                 Honeywell   NASDAQ    HON                    Conglomerate
     13                       IBM     NYSE    IBM          Information technology
     14                     Intel   NASDAQ   INTC          Information technology
```

```
15         Johnson & Johnson     NYSE    JNJ         Pharmaceutical industry
16            JPMorgan Chase     NYSE    JPM              Financial services
17                McDonald's     NYSE    MCD                   Food industry
18               Merck & Co.     NYSE    MRK         Pharmaceutical industry
19                 Microsoft   NASDAQ   MSFT          Information technology
20                Nike, Inc.     NYSE    NKE                         Apparel
21          Procter & Gamble     NYSE     PG      Fast-moving consumer goods
22                Salesforce     NYSE    CRM          Information technology
23   The Travelers Companies     NYSE    TRV              Financial services
24        UnitedHealth Group     NYSE    UNH             Managed health care
25    Verizon Communications     NYSE     VZ               Telecommunication
26                 Visa Inc.     NYSE      V              Financial services
27  Walgreens Boots Alliance   NASDAQ    WBA                       Retailing
28                   Walmart     NYSE    WMT                       Retailing
29   The Walt Disney Company     NYSE    DIS  Broadcasting and entertainment

    Date added                                    Notes Index weighting
0   1976-08-09   As Minnesota Mining and Manufacturing           3.62%
1   1982-08-30                                      NaN           3.00%
2   2020-08-31                                      NaN           4.18%
3   2015-03-19                                      NaN           2.78%
4   1987-03-12                                      NaN           4.12%
5   1991-05-06                                      NaN           3.96%
6   2008-02-19           Also 1930-07-18 to 1999-11-01           1.82%
7   2009-06-08                                      NaN           1.10%
8   1987-03-12           Also 1932-05-26 to 1935-11-20           1.04%
9   2019-04-02                                      NaN           1.18%
10  2013-09-20                                      NaN           7.60%
11  1999-11-01                                      NaN           6.05%
12  2020-08-31                                      NaN           4.29%
13  1979-06-29           Also 1932-05-26 to 1939-03-04           2.60%
14  1999-11-01                                      NaN           1.00%
15  1997-03-17                                      NaN           3.26%
16  1991-05-06                                      NaN           2.93%
17  1985-10-30                                      NaN           4.44%
18  1979-06-29                                      NaN           1.45%
19  1999-11-01                                      NaN           5.63%
20  2013-09-20                                      NaN           3.16%
21  1932-05-26                                      NaN           2.66%
22  2020-08-31                                      NaN           4.83%
23  2009-06-08                                      NaN           2.99%
24  2012-09-24                                      NaN           7.88%
25  2004-04-08                                      NaN           1.02%
26  2013-09-20                                      NaN           4.36%
27  2018-06-26                                      NaN           0.90%
28  1997-03-17                                      NaN           2.77%
29  1991-05-06                                      NaN           3.32%
```

1.3 Import DJIA components into Python

We have to import the 30 components into a list called “constituents”:

constituents = [“MMM”, “AXP”, “AMGN”, “AAPL”, “BA”, “CAT”, “CVX”, “CSCO”, “KO”, “DOW”, “GS”, “HD”, “HON”, “IBM”, “INTC”, “JNJ”, “JPM”, “MCD”, “MRK”, “MSFT”, “NKE”, “PG”, “CRM”, “TRV”, “UNH”, “VZ”, “V”, “WBA”, “WMT”, “DIS”]

[5]:
```
constituents = page[1]['Symbol'] # we only need tickers
constituents.head()
```

[5]:
```
0     MMM
1     AXP
2    AMGN
3    AAPL
4      BA
Name: Symbol, dtype: object
```

1.4 Download DJIA components from Yahoo! Finance

We will use a for loop to import tickers one by one into yf.download() from constituents. Again, we only need adjusted close price for each component stocks. Therefore, we will select only ‘Adj Close’ from each downloaded data. We will store the data in a dataframe called “df_prc”.

[6]:
```
for i in constituents:

    print(i) # print out the ticker so we know the downloading progress
    prc = yf.download(i, interval="1d", start=start_date, end=end_date)
    prc = pd.DataFrame(prc['Adj Close']) # select adjusted close price only
    prc.columns = [i] # rename the column with the ticker of the stock
    try:
        df_prc = pd.concat([df_prc, prc], axis=1) # if the dataframe already␣
 ↪exists, join the newly downloaded data to the existing table
    except:
        df_prc = prc # create the dataframe for the first ticker

    stk = yf.Ticker(i)

    try:
        stk.info['floatShares']
    except:
        stk.info['floatShares'] = None

    try:
        stk.info['sharesOutstanding']
    except:
        stk.info['sharesOutstanding'] = None
    if stk.info['floatShares']:
        mcap = prc * stk.info['floatShares']
    elif stk.info['sharesOutstanding']:
        mcap = prc * stk.info['sharesOutstanding']
    else:
        mcap = prc * (stk.info['marketCap']/stk.info['previousClose'])

    try:
        df_mcap = pd.concat([df_mcap, mcap], axis=1)
    except:
        df_mcap = mcap
```

```
MMM
[*********************100%***********************]  1 of 1 completed
AXP
[*********************100%***********************]  1 of 1 completed
AMGN
[*********************100%***********************]  1 of 1 completed
AAPL
[*********************100%***********************]  1 of 1 completed
BA
[*********************100%***********************]  1 of 1 completed
CAT
[*********************100%***********************]  1 of 1 completed
CVX
[*********************100%***********************]  1 of 1 completed
CSCO
[*********************100%***********************]  1 of 1 completed
KO
[*********************100%***********************]  1 of 1 completed
DOW
[*********************100%***********************]  1 of 1 completed
GS
[*********************100%***********************]  1 of 1 completed
HD
[*********************100%***********************]  1 of 1 completed
HON
[*********************100%***********************]  1 of 1 completed
IBM
[*********************100%***********************]  1 of 1 completed
INTC
[*********************100%***********************]  1 of 1 completed
JNJ
[*********************100%***********************]  1 of 1 completed
JPM
[*********************100%***********************]  1 of 1 completed
```

```
MCD
[*********************100%***********************]  1 of 1 completed
MRK
[*********************100%***********************]  1 of 1 completed
MSFT
[*********************100%***********************]  1 of 1 completed
NKE
[*********************100%***********************]  1 of 1 completed
PG
[*********************100%***********************]  1 of 1 completed
CRM
[*********************100%***********************]  1 of 1 completed
TRV
[*********************100%***********************]  1 of 1 completed
UNH
[*********************100%***********************]  1 of 1 completed
VZ
[*********************100%***********************]  1 of 1 completed
V
[*********************100%***********************]  1 of 1 completed
WBA
[*********************100%***********************]  1 of 1 completed
WMT
[*********************100%***********************]  1 of 1 completed
DIS
[*********************100%***********************]  1 of 1 completed
```

1.5 Preview the table containing prices of DJIA constituents

[7]:

```
df_prc.head()
```

[7]:

```
                   MMM         AXP        AMGN        AAPL          BA  \
Date
2021-06-30  197.125229  164.799988  241.882187  136.755112  239.559998
2021-07-01  197.581741  166.940002  245.008041  137.064651  239.729996
2021-07-02  198.375687  168.500000  246.794250  139.750626  236.679993
2021-07-06  195.398407  169.559998  241.782944  141.807541  236.139999
2021-07-07  198.345917  170.979996  241.356247  144.353729  231.779999

                   CAT         CVX       CSCO         KO        DOW  ...  \
Date                                                                 ...
2021-06-30  216.468338  103.346657  52.633045  54.110001  62.602482  ...
2021-07-01  215.553238  104.797104  53.069996  53.959999  62.820126  ...
2021-07-02  216.597626  104.658966  53.540001  54.180000  62.830017  ...
2021-07-06  212.380264  102.606636  52.980000  53.880001  61.266937  ...
2021-07-07  213.544022  101.560738  53.259998  54.320000  61.583511  ...
```

```
                   NKE          PG         CRM         TRV         UNH  \
Date
2021-06-30  154.235489  134.086670  244.270004  148.880066  399.039337
2021-07-01  157.739700  134.394745  244.979996  150.988312  403.473785
2021-07-02  159.476837  135.050613  248.199997  150.391632  407.928131
2021-07-06  159.846222  135.130112  250.250000  150.003799  408.825012
2021-07-07  159.896149  136.143753  248.440002  151.346313  410.220093

                   VZ           V        WBA         WMT         DIS
Date
2021-06-30  55.407551  233.501694  52.096992  140.502975  175.770004
2021-07-01  55.664665  234.829865  48.235020  138.809204  177.259995
2021-07-02  55.812996  238.305145  47.700287  139.596298  177.110001
2021-07-06  55.783333  239.273819  47.234871  139.426926  173.690002
2021-07-07  55.901997  239.673279  47.026920  139.197769  172.820007

[5 rows x 30 columns]
```

1.6 Construct a price-weighted index using the DJIA constituents

The price-weighted index at time t is the sum of all component stock prices at time t.

[8]:

```python
PWI = df_prc.sum(axis=1) # sum up prices at the same time period
PWI = pd.DataFrame(PWI.rename('PWI')) # put the result in a dataframe
PWI.tail()
```

[8]:

```
                    PWI
Date
2021-08-24  5363.361877
2021-08-25  5369.298626
2021-08-26  5340.136272
2021-08-27  5377.186588
2021-08-30  5369.418331
```

1.7 Compare the actual index with the replicated index

We want to compare the actual index with the replicated index by plotting the cumulative returns of the two indices using the matplotlib.pyplot package.

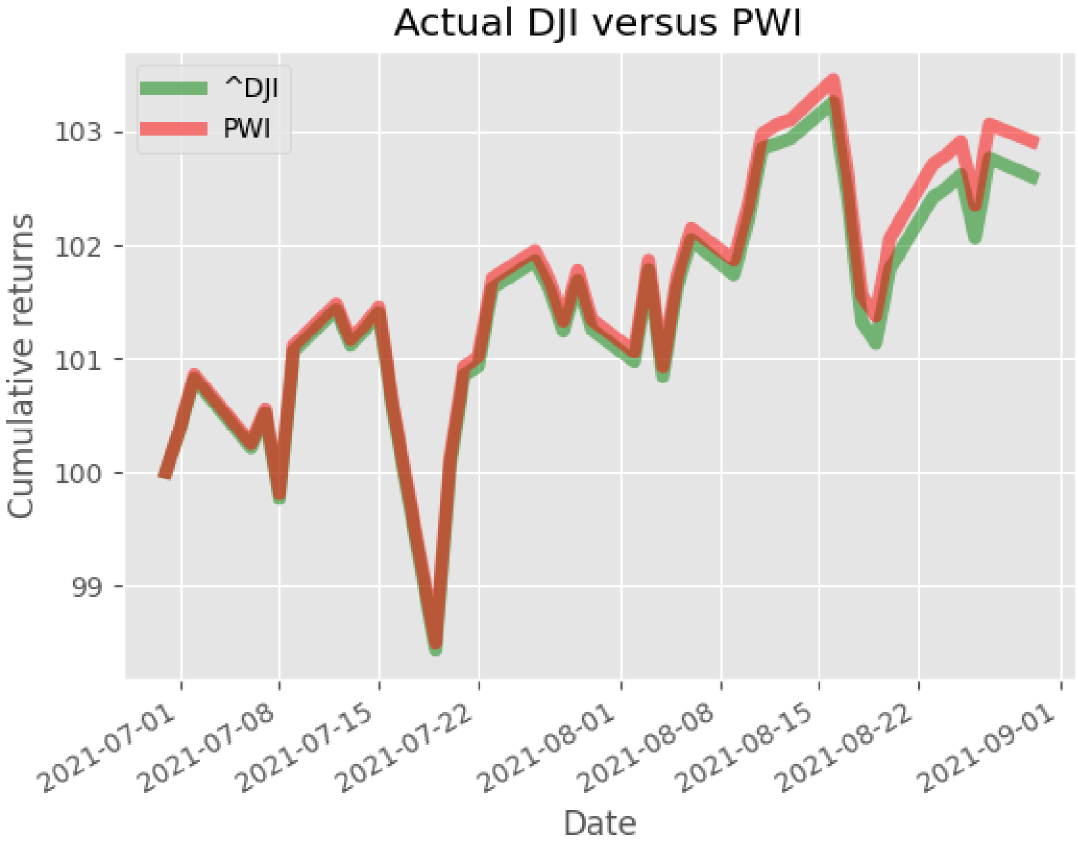
[9]:

```python
import matplotlib.pyplot as plt

TS = Index.join(PWI) # join the actual index with the replicated index
# TS[idx] = TS[idx]/TS[idx][0] # compute cumulative returns of $1 investment in␣
 ↪the actual index
# TS.PW_rep = TS.PWI/TS.PWI[0] # compute cumulative returns of $1 investment in␣
 ↪the replicated index
```

```
TS = TS.divide(TS.iloc[0] / 100)
plt.style.use('ggplot')

fig = TS.plot(color=["green","red"],alpha=0.5,linewidth=5)
plt.title('Actual DJI versus PWI')
plt.legend(loc='best')
plt.ylabel('Cumulative returns')
plt.show()
```

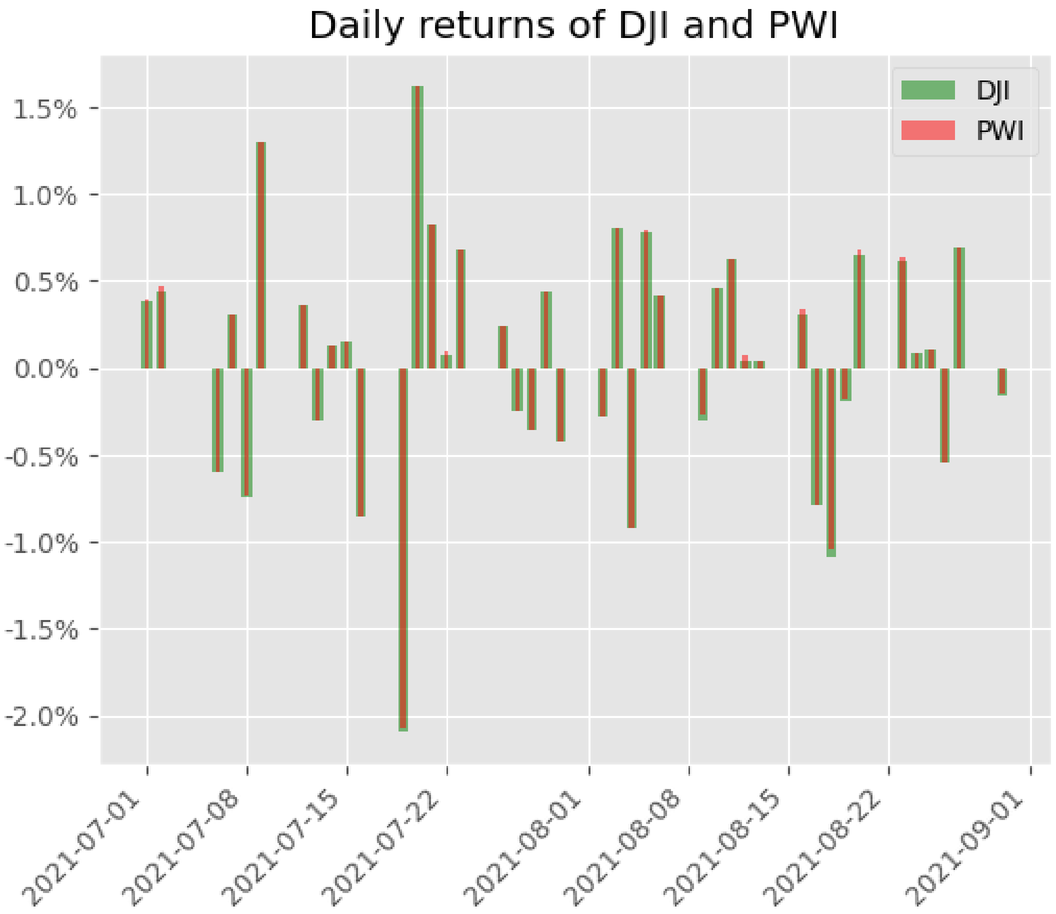


1.8 Compare daily returns

[10]:
```
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

RTS = TS.pct_change()

fig, ax = plt.subplots()
ax.bar(RTS.index,RTS['^DJI'],color='green',alpha=0.5,width=0.75)
ax.bar(RTS.index,RTS['PWI'],color='red',alpha=0.5,width=0.35)
ax.yaxis.set_major_formatter(FuncFormatter(lambda y, _: '{:.1%}'.format(y)))
plt.setp(ax.get_xticklabels(), ha="right", rotation=45)
plt.title('Daily returns of DJI and PWI')
plt.legend(['DJI','PWI'])
plt.show()
```



1.9 Construct a value-weighted index using the DJIA constituents

The value-weighted index at time t is the sum of all component stocks' market capitalizations at time t.

[11]:

```
VWI = df_mcap.sum(axis=1)
VWI = pd.DataFrame(VWI.rename('VWI'))
VWI.tail()
```

[11]:

```
                     VWI
Date
2021-08-24  1.071730e+13
```

```
2021-08-25  1.069581e+13
2021-08-26  1.062984e+13
2021-08-27  1.068485e+13
2021-08-30  1.078056e+13
```

1.10 Construct an equal-weighted index using the DJIA constituents

[12]:

```
EWI = df_prc.pct_change().mean(axis=1)
EWI[0] = 0
EWI = EWI + 1
EWI = EWI.cumprod()
EWI = pd.DataFrame(EWI.rename('EWI'))
EWI.tail()
```

[12]:

```
                 EWI
Date
2021-08-24  1.023343
2021-08-25  1.025043
2021-08-26  1.018529
2021-08-27  1.025725
2021-08-30  1.024868
```

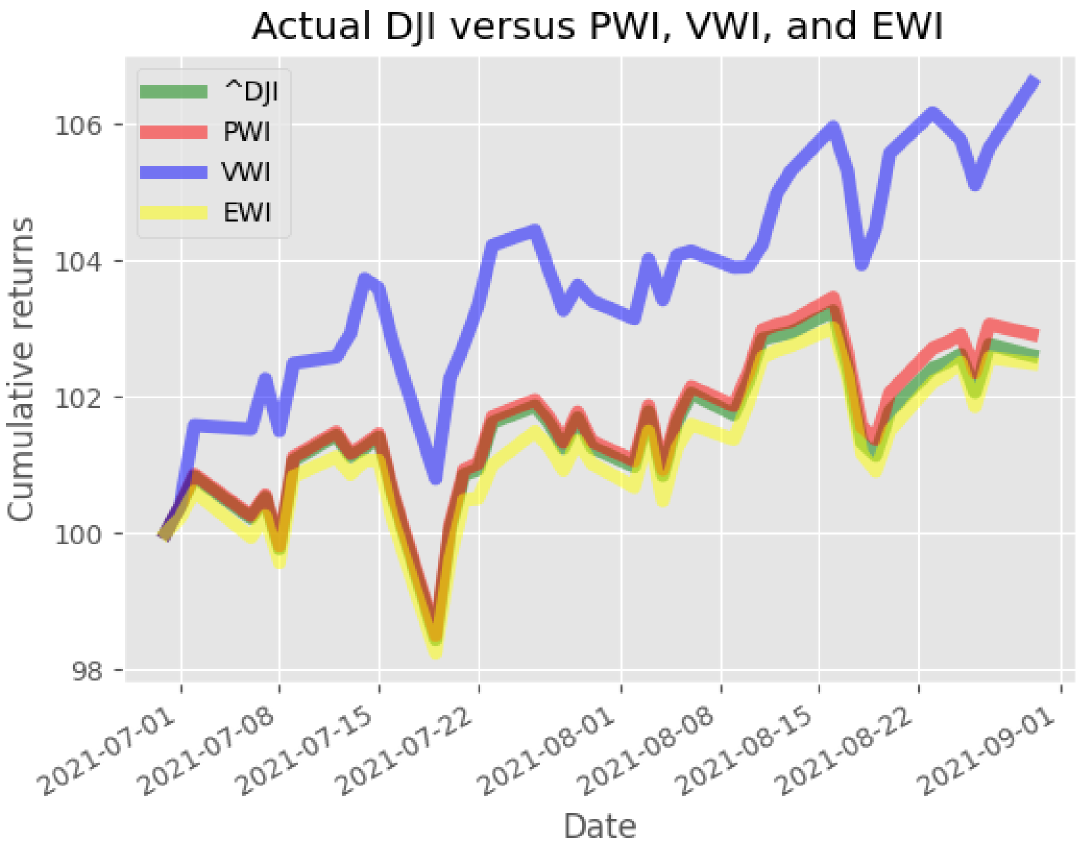
[14]:

```
import matplotlib.pyplot as plt

TS = Index.join([PWI,VWI,EWI])

TS = TS.divide(TS.iloc[0] / 100)
plt.style.use('ggplot')

fig = TS.plot(color=["green","red","blue","yellow"],alpha=0.5,linewidth=5)
plt.title('Actual DJI versus PWI, VWI, and EWI')
plt.legend(loc='best')
plt.ylabel('Cumulative returns')
plt.show()
```


Actual DJI versus PWI, VWI, and EWI
^DJI
PWI
VWI
EWI
Cumulative returns
106
104
102
100
98
2021-07-01
2021-07-08
2021-07-15
2021-07-22
2021-08-01
2021-08-08
2021-08-15
2021-08-22
2021-09-01
Date