Analysis of stock prices using PCA / Notebook 1

In this take-home final you are to analyze the daily changes in stock prices using PCA and to mesure the dimension of stock sequences.

We Start by downloading data and pre-processing it to make it ready for analysis using Spark.

In [1]:

```
import sys,os
import numpy as np
from numpy.linalg import norm
import matplotlib.pyplot as plt
%matplotlib inline

from time import time
import math
import pandas as pd
from glob import glob
import pickle
```

Stock Info

The file `data/TickerInfo.tsv` contains information relating companies to sectors.

In [2]:

```
TickerInfo=pd.read_csv('data/tickerInfo.tsv',sep='\t')
print(TickerInfo.shape)
TickerInfo.head()
```

```
(505, 5)
```

Out[2]:

	Unnamed: 0	Ticker	Name	Sector	SECTOR_ID
0	0	MMM	3M|3M Company	Industrials	INDS
1	1	ABT	Abbott Laboratories	Health Care	HC
2	2	ABBV	AbbVie Inc.	Health Care	HC
3	3	ACN	Accenture plc	Information Technology	IT
4	4	ATVI	Activision Blizzard	Information Technology	IT

In [3]:
```
TickerInfo
```

Out[3]:

	Unnamed: 0	Ticker	Name	Sector	SECTOR_ID
0	0	MMM	3M|3M Company	Industrials	INDS
1	1	ABT	Abbott Laboratories	Health Care	HC
2	2	ABBV	AbbVie Inc.	Health Care	HC
3	3	ACN	Accenture plc	Information Technology	IT
4	4	ATVI	Activision Blizzard	Information Technology	IT
...	...	...	...	...	...
500	500	YHOO	Yahoo Inc.	Information Technology	IT
501	501	YUM	Yum! Brands Inc	Consumer Discretionary	CD
502	502	ZBH	Zimmer Holdings|Zimmer Biomet Holdings	Health Care	HC
503	503	ZION	Zions Bancorp	Financials	FIN
504	504	ZTS	Zoetis	Health Care	HC

505 rows × 5 columns

In [4]:
```
Sectors={'Consumer Discretionary':'CD',
 'Consumer Staples':'CS',
 'Energy':'EN',
 'Financials':'FIN',
 'Health Care':'HC',
 'Industrials':'INDS',
 'Information Technology':'IT',
 'Materials':'MAT',
 'Real Estate':'RE',
 'Telecommunication Services':'TS',
 'Utilities':'UTIL'}
```

Stock and sector information

`TickerInfo` is a pandas table containing, for each Ticker, the company name, the sector, and a sector ID. There are 11 sectors. Some, such as `Consumer Discretionary` and `Information Technology` include many stocks while others, such as `Telecommunication Services` include very few.

In [5]:

```
from collections import Counter
L=list(Counter(TickerInfo['Sector']).items())
print('Sector ID\t\tSector Name\tNo. of Stocks')
print('=========\t\t===========\t=============')
sum=0
for l in L:
    sum+=l[1]
    print('%s\t%30s\t%d\t%d'%(Sectors[l[0]],l[0],l[1],sum))
```

```
Sector ID               Sector Name     No. of Stocks
=========               ===========     =============
INDS                       Industrials  66      66
HC                         Health Care  60      126
IT              Information Technology  70      196
CD              Consumer Discretionary  84      280
UTIL                         Utilities  28      308
FIN                         Financials  66      374
MAT                          Materials  25      399
RE                         Real Estate  31      430
CS                    Consumer Staples  37      467
EN                              Energy  34      501
TS          Telecommunication Services  4       505
```

Download Data

The data is a directory with .csv files, one for each stock. This directory has been tarred and uploaded to S3, at: https://mas-dse-open.s3.amazonaws.com/Stocks/spdata_csv.tgz (https://mas-dse-open.s3.amazonaws.com/Stocks/spdata_csv.tgz)

Download and untar the file to create a subdirectory of the current directory called `spdata_csv`

```
In [6]: !wget https://mas-dse-open.s3.amazonaws.com/Stocks/spdata_csv.tgz
```

```
--2020-06-11 09:30:53--  https://mas-dse-open.s3.amazonaws.com/Stocks/spdata_csv.tgz (https://mas-dse-open.s3.amazonaws.com/Stocks/spdata_cs
v.tgz)
Resolving mas-dse-open.s3.amazonaws.com (mas-dse-open.s3.amazonaws.com)... 52.218.228.27
Connecting to mas-dse-open.s3.amazonaws.com (mas-dse-open.s3.amazonaws.com)|52.218.228.27|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 108227395 (103M) [application/x-tar]
Saving to: ‘spdata_csv.tgz.2’

spdata_csv.tgz.2    100%[===================>] 103.21M  2.30MB/s    in 61s

2020-06-11 09:31:55 (1.70 MB/s) - ‘spdata_csv.tgz.2’ saved [108227395/108227395]
```

```
In [7]: !tar xzf spdata_csv.tgz
```

```
In [8]: files=!ls -1 spdata_csv/train/
        files[:5]
```

```
Out[8]: ['A.csv', 'AAPL.csv', 'ABC.csv', 'ABT.csv', 'ACN.csv']
```

```
In [9]: !ls -l spdata_csv/train/ | wc
        !ls -l spdata_csv/test/ | wc
```

```
     393    3530   21586
      90     803    4809
```

```
In [10]: #change this path to where you stored the raw data   #del Cell
         Data_dir='./spdata_csv/'  #del
         curr_dir=os.getcwd()      #del
         curr_dir                  #del
```

```
Out[10]: '/Users/gio/Documents/DSE/2019-rgm001/DSE230/final'
```

Read Data and create a single table

Your task in this notebook is to read the sock-information `.csv` files, extract from them the column `Adj. Open` and combine them into a single `.csv` file containing all of the information that is relevant for later analysis.

Below we suggest a particular sequence of steps, you can either follow these steps, or do this in your own way. The end result should be a file called `SP500.csv` which stores the information described below.

Step 1: files into pandas dataframes

In this step you read all of the relevant information into a large dictionary we call `Tables` .

The key to this dictionary is the stocks "ticker" which corresponds to the file name excluding the `.csv` extension.

You should read in all of the files in the directory `spdata_csv` other than:

- Files for tickers that are not in the list `Tickers` .
- Files for tickers that were listed in the email you got for this final. As part of the email you got about this final, there is a list of tickers that you should omit from your analysis.

In [11]:

```
#del the rest of this cell
%cd spdata_csv
```

```
/Users/gio/Documents/DSE/2019-rgm001/DSE230/final/spdata_csv
```

In [12]:

```
Tables={}
for filename in glob('*/*.csv'):
    print('\r',filename, end=' ')
    code = filename[:-4]
    tbl=pd.read_csv(filename,index_col='Date',parse_dates=True)
    if(np.shape(tbl)[1]==12):
        Tables[code]=tbl.sort_index()
        Tables[code]
    else:
        print(filename,np.shape(tbl))

%cd ..
```

```
 train/PPL.csv   /Users/gio/Documents/DSE/2019-rgm001/DSE230/final
```

In [13]:

```
# Example of an entry in `Tables`
print(len(Tables))
Tables['train/IBM'].head()
```

```
481
```

Out[13]:

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
1962-01-02	578.5	578.5	572.0	572.00	5162.666667	0.0	1.0	5.180989	5.180989	5.122776	5.122776	387200.0
1962-01-03	572.0	577.0	572.0	577.00	3840.000000	0.0	1.0	5.122776	5.167555	5.122776	5.167555	288000.0
1962-01-04	577.0	577.0	571.0	571.25	3413.333333	0.0	1.0	5.167555	5.167555	5.113820	5.116059	256000.0
1962-01-05	570.5	570.5	559.0	560.00	4842.666667	0.0	1.0	5.109342	5.109342	5.006349	5.015305	363200.0
1962-01-08	559.5	559.5	545.0	549.50	7253.333333	0.0	1.0	5.010827	5.010827	4.880966	4.921268	544000.0

Step 2: Computing diffs and combining into a single table

The next step is to extract from each table the relevant prices, compute an additional quantity we call `diff` and create a single combined pandas dataframe.

The price we ue is the **Adjusted Open Price** which is the price when the stock exchange opens in the morning. We use the **adjusted** price which eliminates technical adjustments such as stock splits.

It is more meaningful to predict *changes* in prices than prices themselves. We therefor compute for each stock a `Diffs` sequence in which $d(t) = \log \frac{p(t+1)}{p(t)}$ where $p(t)$ is the price at time t and $d(t)$ is the price diff or the price ratio.

Obviously, if we have a price sequence of length T then the length of the diff sequence will be $T-1$. To make the price sequence and the diff sequence have the same length we eliminate the last day price for each sequence.

Your task in this step is to join the stock tables by date, compute the diff seqeunce, and create one large Pandas DataFrame where the row index is the date, and there are two columns for each ticker. For example for the ticker `IBM` , there would be two columns `IBM_P` and `IBM_D` . The first corresponds to the prices of the IBM stock $p(t)$ and the second to the price difference $d(t)$

In [14]:

```
Diffs=pd.DataFrame()
# Remove the following lines to process all stocks. Plot the PCA and the time sequences of these stocks
# to see why they are outliers and need to be removed.
Indices=set(Tables.keys())      #remove outlier stock - most likely due to file errors
print(len(Indices))
i=1
for code in Indices:
    print('\r',i,code, end='  ')
    i+=1
    tbl=Tables[code]
    S=tbl['Adj. Open']
    prices=np.array(S)  # The length of "prices" will remain the original length.
    diff=np.log(prices[1:]/prices[:-1])
    I=S.index[:-1]
    #print np.shae(Diffs),np.shape(Sdiff),np.shape(diff),len(I)
    Sdiff=pd.DataFrame({code+'_D':diff,code+'_P':prices[:-1]},index=I)
    Diffs=Diffs.join(Sdiff,how='outer')
#rm above
Diffs.head()
```

```
481
 481 train/VRSN
```

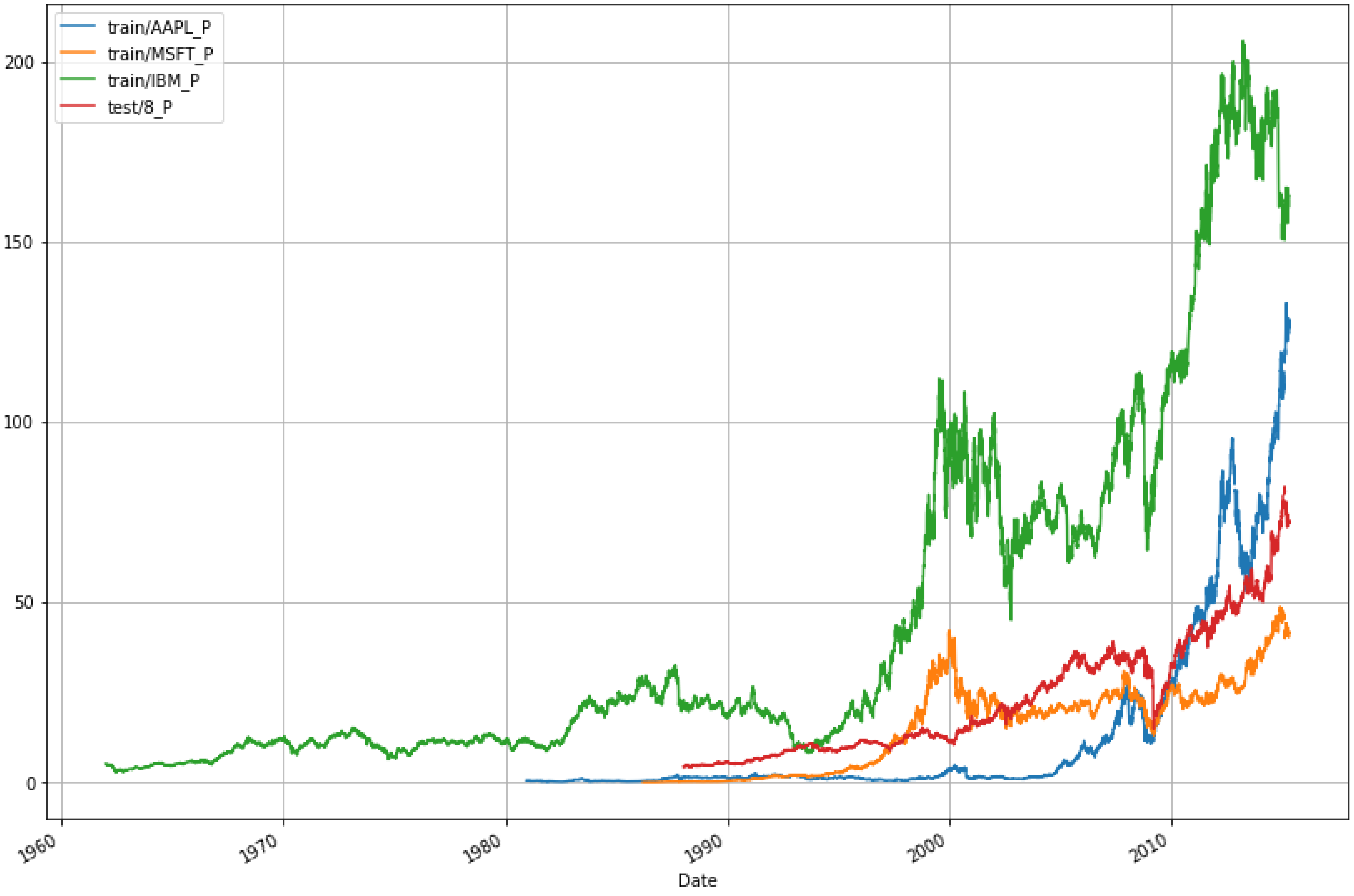
Out[14]:

	train/BIIB_D	train/BIIB_P	train/MO_D	train/MO_P	train/FAST_D	train/FAST_P	train/NUE_D	train/NUE_P	test/43_D	test/43_P	...	train/TXT_D	train/TXT_P	test/48_D	test/48_P	test/
Date																
1962-01-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1962-01-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1962-01-04	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1962-01-05	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	
1962-01-08	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	

5 rows × 962 columns

In [15]:

```
# plot some stocks
Diffs[['train/AAPL_P','train/MSFT_P','train/IBM_P','test/8_P']].plot(figsize=(14,10));
plt.grid()
```

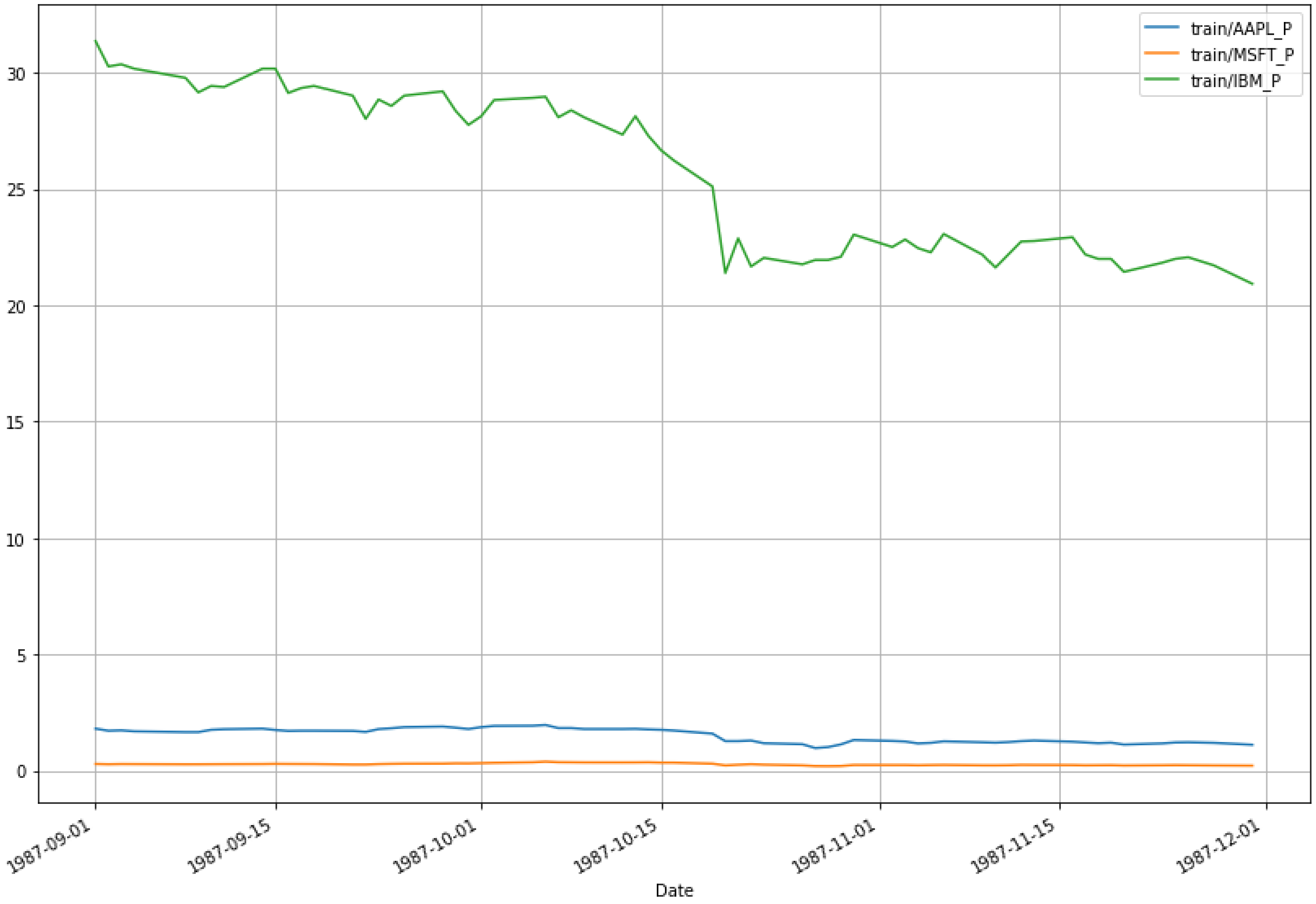


Black Monday

One of the biggest crashes in the US stock market happened on **Black Monday:** Oct 19 1987

We will look at the stocks around that date

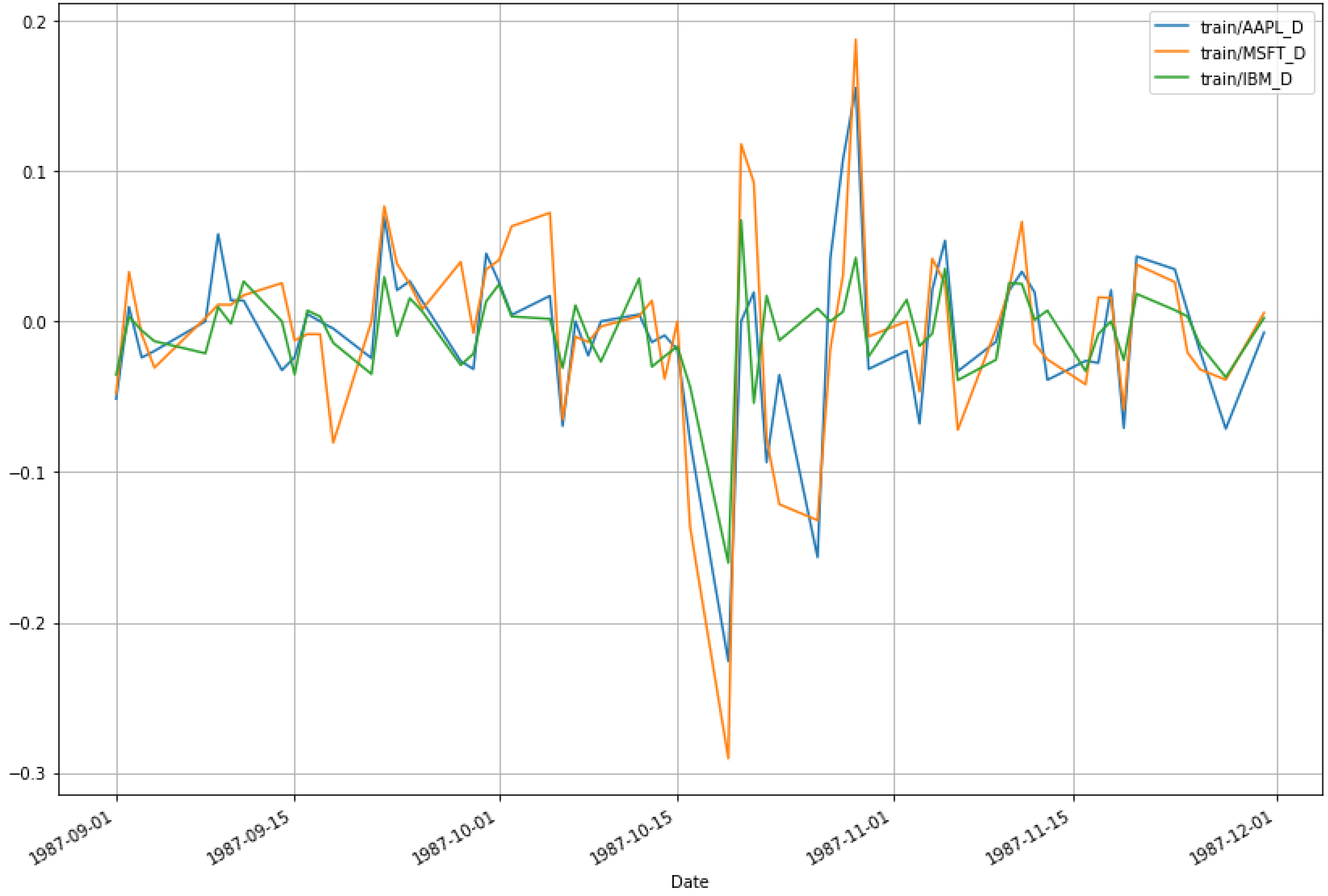
```
In [16]: #Focus on "Black Monday:" the stock crash of Oct 19 1987

         import datetime
         format = "%b-%d-%Y"

         _from = datetime.datetime.strptime('Sep-1-1987', format)
         _to = datetime.datetime.strptime('Nov-30-1987', format)

         Diffs.loc[_from:_to,['train/AAPL_P','train/MSFT_P','train/IBM_P']].plot(figsize=(14,10));
         plt.grid()
```



Why does it seems that the price of IBM fell much more than those of Apple and microsoft?

Because IBM's price started so much higher. As explained above it is more informative to consider $\log(p_{t+1}/p_t)$

In [17]:
```
Diffs.loc[_from:_to,['train/AAPL_D','train/MSFT_D','train/IBM_D']].plot(figsize=(14,10));
plt.grid()
```



Extract tickers

Complete the following function to extract the tickers (company names) from the column names of the dataframe `df` . The tickers can be obtained by removing the `train/` and `test/` prefix and the `_D` or `_P` suffix from the `df` column names

Input: `df` dataframe read in 1.2

Returns: `tickers` - list of tickers

Example Output:

['RF', 'TIF', 'HAL', 'KSS', 'INTU', 'LH',]

```
In [18]: def partition_columns(df):
             """Partition columns of df into train set and test set
             Each of them sorted lexicographically."""

             ### Your code here

             # get train list
         #     train_col = sorted(list(set([i.split('/')[1].split('_')[0] for i in df.columns if 'train' in i])))
             train_col = sorted([i for i in df.columns if 'train' in i])


             # get test list
         #     test_col = sorted(list(set([i.split('/')[1].split('_')[0] for i in df.columns if 'test' in i])))
             test_col = sorted([i for i in df.columns if 'test' in i])

             return  train_col+test_col

         columns = partition_columns(Diffs)
```

```
In [19]: Diffs=Diffs[columns]
```

```
In [20]: columns[:6],columns[-5:]
```

```
Out[20]: (['train/AAPL_D',
           'train/AAPL_P',
           'train/ABC_D',
           'train/ABC_P',
           'train/ABT_D',
           'train/ABT_P'],
          ['test/88_P', 'test/8_D', 'test/8_P', 'test/9_D', 'test/9_P'])
```

In [21]:

```
Diffs.columns[:5],Diffs.columns[-5:]
```

Out[21]:

```
(Index(['train/AAPL_D', 'train/AAPL_P', 'train/ABC_D', 'train/ABC_P',
        'train/ABT_D'],
       dtype='object'),
 Index(['test/88_P', 'test/8_D', 'test/8_P', 'test/9_D', 'test/9_P'], dtype='object'))
```

In [27]:

```
## visible tests.
assert len(columns) == 962, 'Incorrect number of columns'
assert len([a for a in Diffs.columns if '_D' in a]) == 481, 'Incorrect number of diff columns'
assert Diffs.shape == (13422, 962), 'Incorrect data shape'
```

In [23]:

```
#(5 points)
# HIDDEN TESTS
```

In [24]:

```
#%cd ..
Diffs.to_csv('data/SP500.csv')
```

Note

In order to make sure errors in constructing data do not get propagated in other notebooks of the final, you may run the below cell which will download the instructors version of "SP500.csv". For next notebooks, you may use either your own version or the one provided by us. Ideally both should have the same contents

In [25]:

```
%mkdir -p data/
%cd data
!rm -f data.tgz && rm -rf data ## Instructor's version of the output from this notebook
!wget https://mas-dse-open.s3.amazonaws.com/Stocks/data.tgz
!tar -xf data.tgz ## Extracting data
%cd ../
## Going back to `Final` directory to keep it as our working directory
%ls -al data/

## Now the `data` folder should have another `data` folder which contains the instructors version of SP500.csv
```

```
/Users/gio/Documents/DSE/2019-rgm001/DSE230/final/data
--2020-06-11 09:34:13--  https://mas-dse-open.s3.amazonaws.com/Stocks/data.tgz (https://mas-dse-open.s3.amazonaws.com/Stocks/data.tgz)
Resolving mas-dse-open.s3.amazonaws.com (mas-dse-open.s3.amazonaws.com)... 52.218.197.155
Connecting to mas-dse-open.s3.amazonaws.com (mas-dse-open.s3.amazonaws.com)|52.218.197.155|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 52006823 (50M) [application/x-tar]
Saving to: ‘data.tgz’

data.tgz            100%[===================>]  49.60M  1.97MB/s    in 24s

2020-06-11 09:34:38 (2.06 MB/s) - ‘data.tgz’ saved [52006823/52006823]

/Users/gio/Documents/DSE/2019-rgm001/DSE230/final
total 372360
drwxr-xr-x@ 10 gio  staff        320 Jun 11 09:34 ./
drwxr-xr-x@ 19 gio  staff        608 Jun 11 09:33 ../
-rw-r--r--@  1 gio  staff       8196 Jun 10 14:33 .DS_Store
drwxr-xr-x@  2 gio  staff         64 Jun  2 15:26 .ipynb_checkpoints/
-rw-r--r--   1 gio  staff    1864236 Jun  8  2018 PCA.pickle
-rw-r--r--@  1 gio  staff    1864236 Jun  2 14:51 PCA_true.pickle
-rw-r--r--   1 gio  staff  120321934 Jun 11 09:34 SP500.csv
drwxr-xr-x   5 gio  staff        160 Jun  8  2018 data/
-rw-r--r--   1 gio  staff   52006823 Jun  8  2018 data.tgz
-rw-r--r--   1 gio  staff      22748 May 31  2018 tickerInfo.tsv
```

In [26]:

```
## How different is your version than the instructors? (Not for grading only for self evaluation)
!diff data/SP500.csv data/data/SP500.csv
```

```
390c390
< 1963-07-17,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.0,0.3374439728167600
3,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.0,2.468898504589,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,-0.00
73183808076856555,5.7011783037075,,,,,,,,,,,,,,,-0.019418085857101627,0.0829185820784373
7,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.003159560290361214,0.4870718602603799
6,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.011299555253934125,0.12010554312327001,,,,,,,,,,,,,,,-0.005182850645799869,3.898116411186700
6,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.013280407667899069,0.726733598475150
1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
0.0,1.8440057871327,,,,,,,,,,
---
> 1963-07-17,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.0,0.3374439728167600
3,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.0,2.468898504589,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,-0.00
73183808076856555,5.7011783037075,,,,,,,,,,,,,,,-0.019418085857101627,0.0829185820784373
7,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.0031595602903612134,0.4870718602603799
6,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.011299555253934125,0.12010554312327001,,,,,,,,,,,,,,,-0.005182850645799869,3.898116411186700
6,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,0.013280407667899069,0.726733598475150
1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
0.0,1.8440057871327,,,,,,,,,,
863c863
```

In [ ]: