In [ ]:

```
# TODO: This analysis needs to be documented and the analyses explained.
```

In [1]:

```
#import required packages
import numpy as np
import pandas as pd
import quandl
import datetime
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter, WeekdayLocator,\
    DayLocator, MONDAY
from mpl_finance import candlestick_ohlc

%matplotlib inline
%pylab inline
pylab.rcParams['figure.figsize'] = (15,9)

quandl.ApiConfig.api_key = "5znYSS5KeqDSE_aakDFg"
```

```
Populating the interactive namespace from numpy and matplotlib
```

In [2]:

```
#First Energy (FE)
#Duquesne Light (DQE)
#PECO Energy (PE/PA)
#PP&L (PPL)
#UGI (UGI)

fe = "FE"
pe_pa = "PE/PA"
ppl = "PPL"
ugi = "UGI"
start = datetime.datetime(2016,1,1)
end = datetime.date.today()
```

In [3]:

```
first_energy, peco_energy, ppl_electric, united_gas = (quandl.get("WIKI/"
+ s, start_date=start, end_date=end) for s in [fe, pe_pa, ppl, ugi])
```

In [4]:

```
first_energy.head()
```

Out[4]:

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	C
Date											
2016-01-04	31.52	31.63	31.22	31.50	4524856.0	0.0	1.0	28.513084	28.612590	28.241703	28.49
2016-01-05	31.63	31.64	30.94	31.54	5204237.0	0.0	1.0	28.612590	28.621636	27.988414	28.53
2016-01-06	31.32	32.01	31.18	31.85	3822654.0	0.0	1.0	28.332163	28.956339	28.205519	28.81
2016-01-07	31.52	31.77	30.89	31.00	4218206.0	0.0	1.0	28.513084	28.739234	27.943184	28.04

2016-01-08	31.02	31.77	30.98	31.59	5032004.0	0.0	1.0	28.060782	28.739234	28.024598	28.57

In [5]:

```
peco_energy.head()
```

Out[5]:

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	Adj. Close	Adj. Volume
Date												
2016-01-04	18.41	18.60	18.13	18.47	2920611.0	0.0	1.0	18.41	18.60	18.13	18.47	2920611.0
2016-01-05	18.41	18.97	18.37	18.44	2893197.0	0.0	1.0	18.41	18.97	18.37	18.44	2893197.0
2016-01-06	17.71	18.09	17.47	17.61	2945631.0	0.0	1.0	17.71	18.09	17.47	17.61	2945631.0
2016-01-07	17.14	18.32	17.06	17.78	3124571.0	0.0	1.0	17.14	18.32	17.06	17.78	3124571.0
2016-01-08	18.02	18.55	17.47	17.52	2510695.0	0.0	1.0	18.02	18.55	17.47	17.52	2510695.0

In [6]:

```
ppl_electric.head()
```

Out[6]:

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	
Date											
2016-01-04	33.90	33.93	33.370	33.68	5092491.0	0.0	1.0	31.501150	31.529027	31.008654	31.2
2016-01-05	33.69	34.10	33.230	34.04	5068774.0	0.0	1.0	31.306010	31.686997	30.878561	31.6
2016-01-06	33.77	34.13	33.640	33.93	4119091.0	0.0	1.0	31.380349	31.714874	31.259548	31.5
2016-01-07	33.67	33.83	33.420	33.54	3897710.0	0.0	1.0	31.287425	31.436103	31.055116	31.1
2016-01-08	33.57	33.82	33.325	33.39	5717778.0	0.0	1.0	31.194501	31.426811	30.966838	31.0

In [7]:

```
united_gas.head()
```

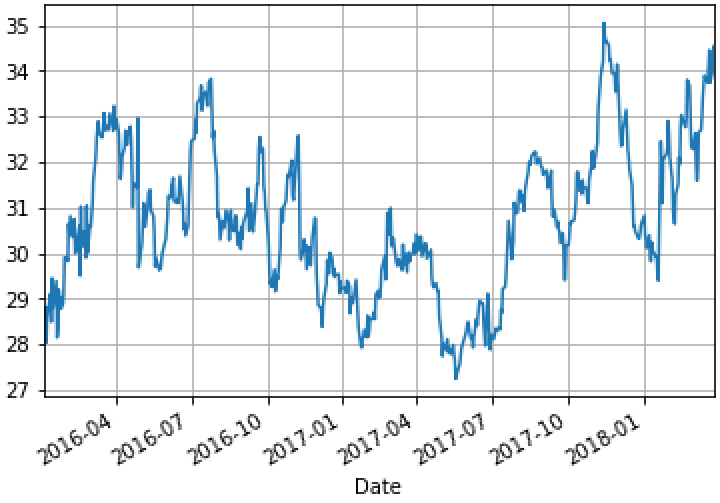
Out[7]:

	Open	High	Low	Close	Volume	Ex-Dividend	Split Ratio	Adj. Open	Adj. High	Adj. Low	C
Date											
2016-											

01-04	33.86	34.37	33.59	33.85	1171111.0	0.0	1.0	32.632977	33.124495	32.372761	32.62
2016-01-05	33.74	34.22	33.28	34.10	1138798.0	0.0	1.0	32.517325	32.979931	32.073995	32.86
2016-01-06	33.78	34.18	33.66	34.03	554998.0	0.0	1.0	32.555876	32.941380	32.440224	32.79
2016-01-07	33.41	33.76	33.12	33.25	664306.0	0.0	1.0	32.199284	32.536600	31.919793	32.04
2016-01-08	33.17	33.68	33.05	33.43	1206743.0	0.0	1.0	31.967981	32.459499	31.852329	32.21

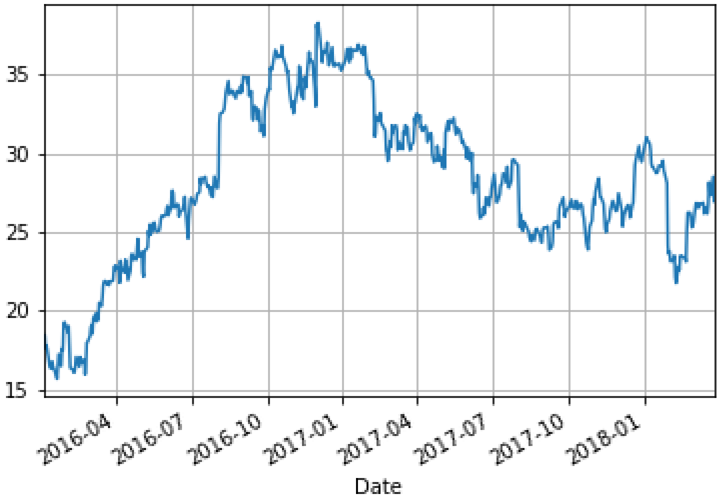
```
In [8]: first_energy["Adj. Close"].plot(grid=True)
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0xc16e150>
```



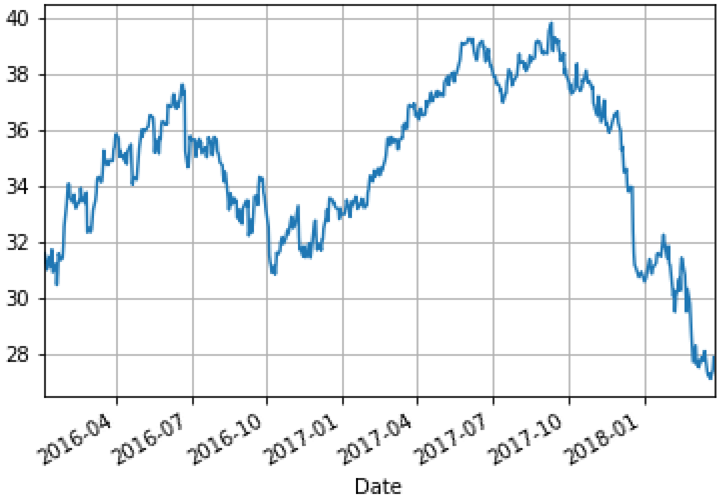
```
In [9]: peco_energy["Adj. Close"].plot(grid=True)
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0xc5c2a50>
```



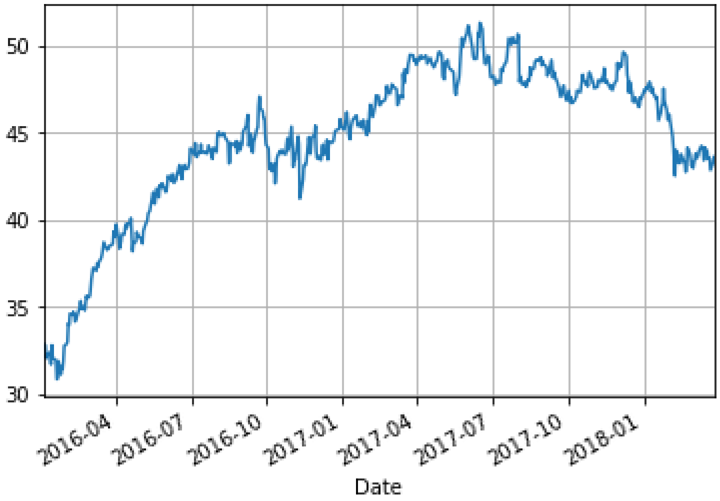
```
In [10]: ppl_electric["Adj. Close"].plot(grid=True)
```

```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0xc678450>
```



```
In [11]: united_gas["Adj. Close"].plot(grid=True)
```

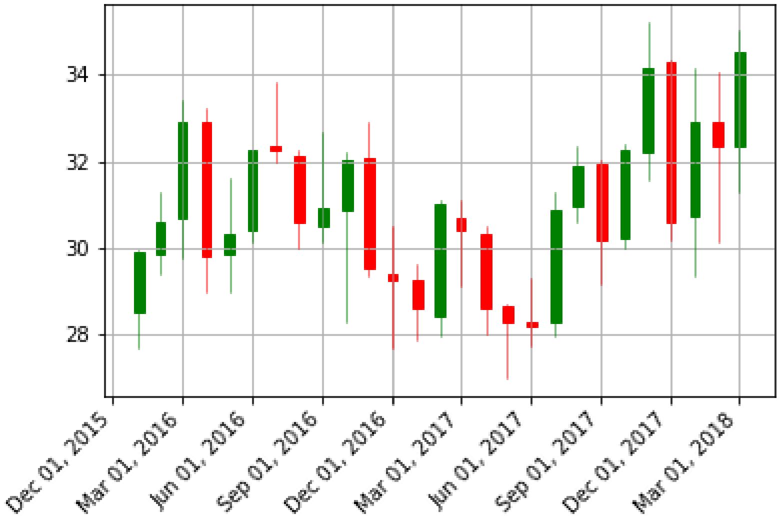
```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0xc6b69f0>
```



```
In [12]: def pandas_candlestick_ohlc(dat, stick = "day", adj = False, otherseries = None):
    """
    :param dat: pandas DataFrame object with datetime64 index, and float columns "Open", "High", "Low", and "Close", likely created via DataReader from "yahoo"
    :param stick: A string or number indicating the period of time covered by a single candlestick. Valid string inputs include "day", "week", "month", and "year", ("day" default), and any numeric input indicates the number of trading days included in a period
    :param adj: A boolean indicating whether to use adjusted prices
    :param otherseries: An iterable that will be coerced into a list, containing the columns of dat that hold other series to be plotted as lines

    This will show a Japanese candlestick plot for stock data stored in dat, also plotting other series if passed.
    """
    mondays = WeekdayLocator(MONDAY)        # major ticks on the mondays
    alldays = DayLocator()              # minor ticks on the days
    dayFormatter = DateFormatter('%d')      # e.g., 12

    # Create a new DataFrame which includes OHLC data for each period specified by stick input
    fields = ["Open", "High", "Low", "Close"]
    if adj:
        fields = ["Adj. " + s for s in fields]
    transdat = dat.loc[:,fields]
    transdat.columns = pd.Index(["Open", "High", "Low", "Close"])
    if (type(stick) == str):
        if stick == "day":
            plotdat = transdat
            stick = 1 # Used for plotting
        elif stick in ["week", "month", "year"]:
            if stick == "week":
                transdat["week"] = pd.to_datetime(transdat.index).map(lambda x: x.isocalendar()[1]) # Identify weeks
            elif stick == "month":
                transdat["month"] = pd.to_datetime(transdat.index).map(lambda x: x.month) # Identify months
            transdat["year"] = pd.to_datetime(transdat.index).map(lambda x: x.isocalendar()[0]) # Identify years
            grouped = transdat.groupby(list(set(["year",stick]))) # Group by year and other appropriate variable
            plotdat = pd.DataFrame({"Open": [], "High": [], "Low": [], "Close": []}) # Create empty data frame containing what will be plotted
            for name, group in grouped:
                plotdat = plotdat.append(pd.DataFrame({"Open": group.iloc[0,0],
                                            "High": max(group.High),
                                            "Low": min(group.Low),
                                            "Close": group.iloc[-1,3]},
                                           index = [group.index[0]]))
            if stick == "week": stick = 5
            elif stick == "month": stick = 30
            elif stick == "qtr": stick = 91
            elif stick == "year": stick = 365

    elif (type(stick) == int and stick >= 1):
        transdat["stick"] = [np.floor(i / stick) for i in range(len(transdat.index))]
        grouped = transdat.groupby("stick")
        plotdat = pd.DataFrame({"Open": [], "High": [], "Low": [], "Close": []}) # Create empty data frame containing what will be plotted
        for name, group in grouped:
            plotdat = plotdat.append(pd.DataFrame({"Open": group.iloc[0,0],
                                        "High": max(group.High),
                                        "Low": min(group.Low),
                                        "Close": group.iloc[-1,3]},
                                       index = [group.index[0]]))

    else:
        raise ValueError('Valid inputs to argument "stick" include the strings "day", "week", "month", "year", or a positive integer')
    # Set plot parameters, including the axis object ax used for plotting
    fig, ax = plt.subplots()
    fig.subplots_adjust(bottom=0.2)
    if plotdat.index[-1] - plotdat.index[0] < pd.Timedelta('730 days'):
        weekFormatter = DateFormatter('%b %d')  # e.g., Jan 12
        ax.xaxis.set_major_locator(mondays)
        ax.xaxis.set_minor_locator(alldays)
    else:
        weekFormatter = DateFormatter('%b %d, %Y')
    ax.xaxis.set_major_formatter(weekFormatter)

    ax.grid(True)

    # Create the candelstick chart
    candlestick_ohlc(ax, list(zip(list(date2num(plotdat.index.tolist())),
plotdat["Open"].tolist(), plotdat["High"].tolist(),
                      plotdat["Low"].tolist(), plotdat["Close"].tolist()))
,
                      colorup = "green", colordown = "red", width = stick
* .4)

    # Plot other series (such as moving averages) as lines
    if otherseries != None:
        if type(otherseries) != list:
            otherseries = [otherseries]
        dat.loc[:,otherseries].plot(ax = ax, lw = 1.3, grid = True)

    ax.xaxis_date()
    ax.autoscale_view()
    plt.setp(plt.gca().get_xticklabels(), rotation=45, horizontalalignment
='right')

    plt.show()
```

In [13]:

```
pandas_candlestick_ohlc(first_energy, adj=True, stick="month")
```



In [14]:

```
stocks = pd.DataFrame({"FE": first_energy["Adj. Close"],
                              "PE/PA": peco_energy["Adj. Close"],
                              "PPL": ppl_electric["Adj. Close"],
                              "UGI": united_gas["Adj. Close"]})
```

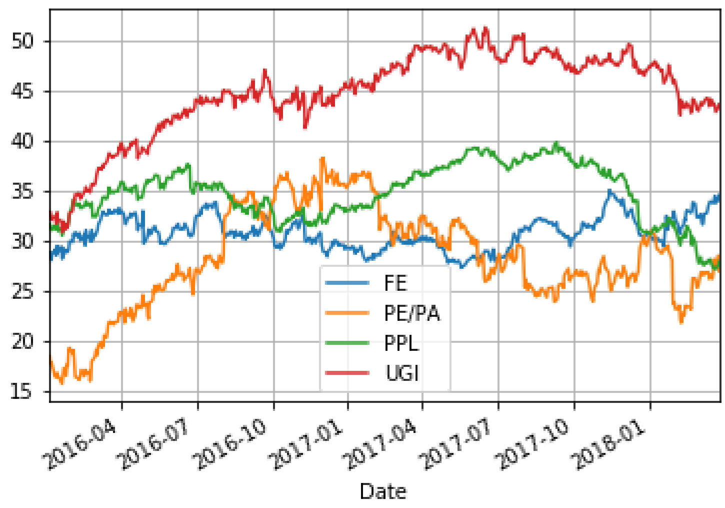
In [15]:
```
stocks.head()
```

Out[15]:

	FE	PE/PA	PPL	UGI
Date				
2016-01-04	28.494992	18.47	31.296717	32.623339
2016-01-05	28.531176	18.44	31.631243	32.864279
2016-01-06	28.811603	17.61	31.529027	32.796816
2016-01-07	28.042690	17.78	31.166624	32.045082
2016-01-08	28.576406	17.52	31.027239	32.218559

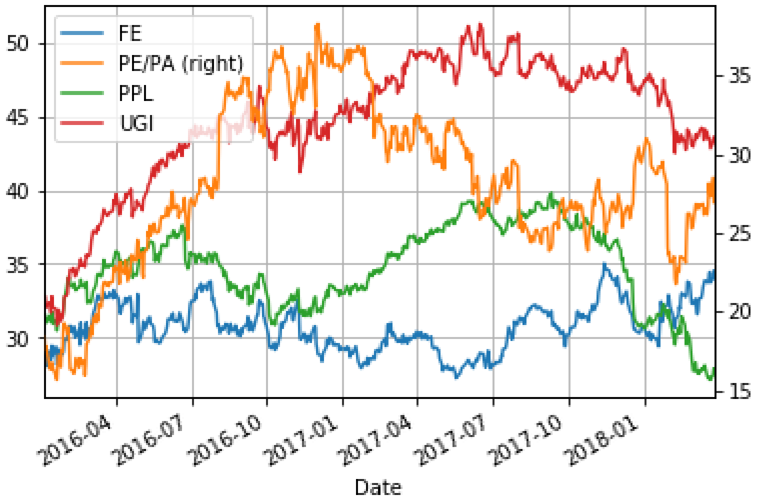
In [16]:
```
stocks.plot(grid=True)
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xc7a1890>
```



In [17]:
```
stocks.plot(secondary_y = ["PE/PA"], grid = True)
```

Out[17]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xc623dd0>
```



In [18]:

```
#return: (x1 - x0) - 1 | x1 - x0 / x0
stock_return = stocks.apply(lambda x: (x/x[0]) - 1)
```

In [19]:

```
stock_return.head()
```

Out[19]:

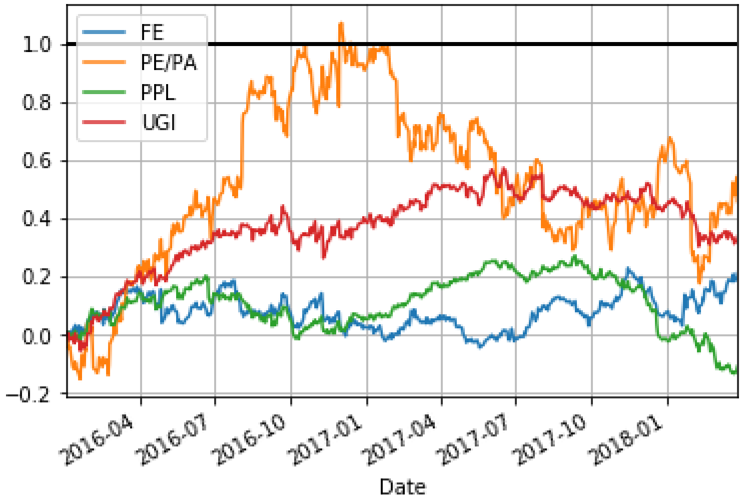
	FE	PE/PA	PPL	UGI
Date				
2016-01-04	0.000000	0.000000	0.000000	0.000000
2016-01-05	0.001270	-0.001624	0.010689	0.007386
2016-01-06	0.011111	-0.046562	0.007423	0.005318
2016-01-07	-0.015873	-0.037358	-0.004157	-0.017725
2016-01-08	0.002857	-0.051435	-0.008610	-0.012408

In [20]:

```
stock_return.plot(grid = True).axhline(y = 1, color = "black", lw = 2)
```

Out[20]:

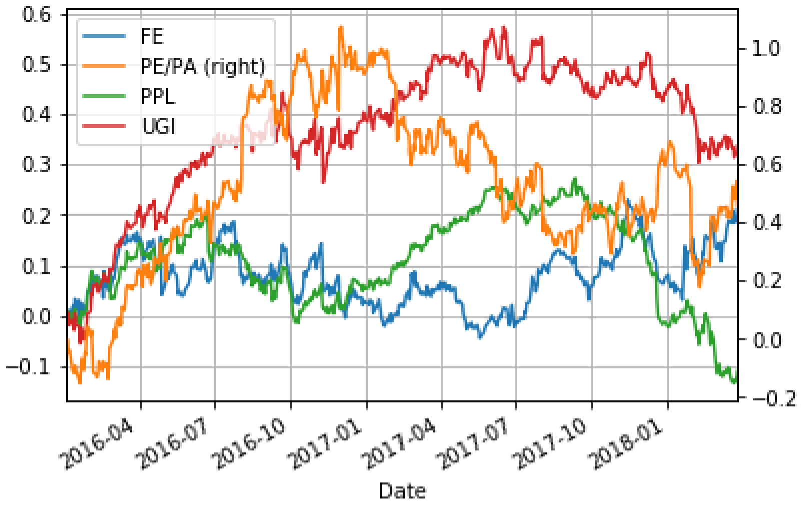
```
<matplotlib.lines.Line2D at 0xda73610>
```



In [21]:

```
stock_return.plot(secondary_y = ["PE/PA"], grid = True)
```

Out[21]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xdaa8a10>
```



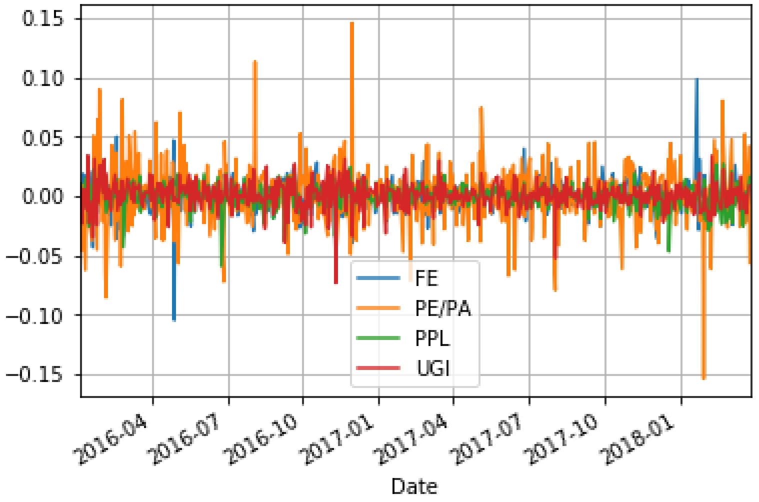
In [22]:
```
stock_change = stocks.apply(lambda x: np.log(x) - np.log(x.shift(1))) # shift moves dates back by 1.
stock_change.head()
```

Out[22]:

Date	FE	PE/PA	PPL	UGI
2016-01-04	NaN	NaN	NaN	NaN
2016-01-05	0.001269	-0.001626	0.010632	0.007358
2016-01-06	0.009781	-0.046055	-0.003237	-0.002055
2016-01-07	-0.027050	0.009607	-0.011561	-0.023188
2016-01-08	0.018853	-0.014731	-0.004482	0.005399

In [23]:
```
stock_change.plot(grid=True)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xc657930>
```



In [24]:
```
spyderdat = pd.read_csv(r"C:\Users\rurobbins\source\repos\classic_metrics_
notebook\spdr.csv")
```

In [25]:
```
spyderdat.head()
```

Out[25]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2015-12-31	205.130005	205.889999	203.869995	203.869995	191.285904	114877900
1	2016-01-04	200.490005	201.029999	198.589996	201.020004	188.611862	222353500
2	2016-01-05	201.399994	201.899994	200.050003	201.360001	188.930862	110845800
3	2016-01-06	198.339996	200.059998	197.600006	198.820007	186.547623	152112600
4	2016-01-07	195.330002	197.440002	193.589996	194.050003	182.072052	213436100

In [26]:
```
spyderdat = pd.DataFrame(spyderdat.loc[:, ["Open", "High", "Low", "Close",
 "Adj Close"]].iloc[1:].as_matrix(),
                        index=pd.DatetimeIndex(spyderdat.iloc[1:, 0]),
                     columns=["Open","High","Low","Close","Adj Close"]).s
ort_index()
```

```
C:\Program Files (x86)\Microsoft Visual Studio\Shared\Anaconda2_86\lib\sit
e-packages\ipykernel_launcher.py:1: FutureWarning: Method .as_matrix will
be removed in a future version. Use .values instead.
  """Entry point for launching an IPython kernel.
```

In [27]:
```
spyderdat.head()
```

Out[27]:

	Open	High	Low	Close	Adj Close
Date					
2016-01-04	200.490005	201.029999	198.589996	201.020004	188.611862
2016-01-05	201.399994	201.899994	200.050003	201.360001	188.930862
2016-01-06	198.339996	200.059998	197.600006	198.820007	186.547623
2016-01-07	195.330002	197.440002	193.589996	194.050003	182.072052
2016-01-08	195.190002	195.850006	191.580002	191.919998	180.073517

In [28]:
```
spyderdat.index.name = 'Date'
```

In [29]:
```
spyderdat.head()
```

Out[29]:

	Open	High	Low	Close	Adj Close
Date					
2016-01-04	200.490005	201.029999	198.589996	201.020004	188.611862
2016-01-05	201.399994	201.899994	200.050003	201.360001	188.930862

2016-01-06	198.339996	200.059998	197.600006	198.820007	186.547623
2016-01-07	195.330002	197.440002	193.589996	194.050003	182.072052
2016-01-08	195.190002	195.850006	191.580002	191.919998	180.073517

In [30]:
```
stocks = stocks.join(spyderdat.loc[:, "Close"]).rename(columns={"Close": "SPY"})
```

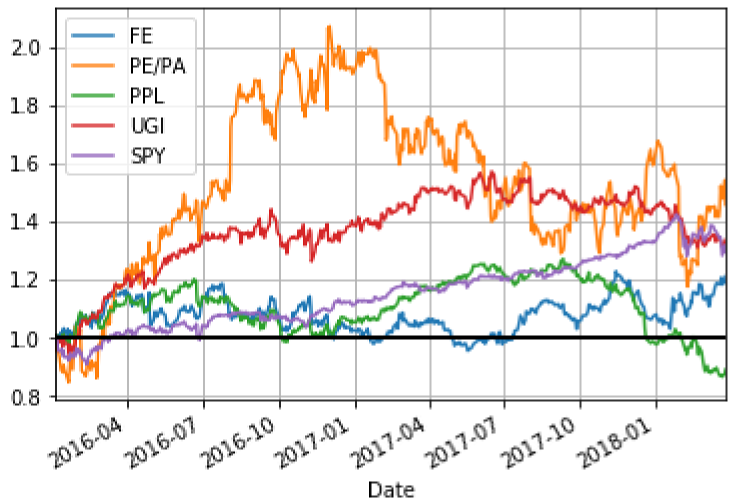
In [31]:
```
stocks.head()
```

Out[31]:

	FE	PE/PA	PPL	UGI	SPY
Date					
2016-01-04	28.494992	18.47	31.296717	32.623339	201.020004
2016-01-05	28.531176	18.44	31.631243	32.864279	201.360001
2016-01-06	28.811603	17.61	31.529027	32.796816	198.820007
2016-01-07	28.042690	17.78	31.166624	32.045082	194.050003
2016-01-08	28.576406	17.52	31.027239	32.218559	191.919998

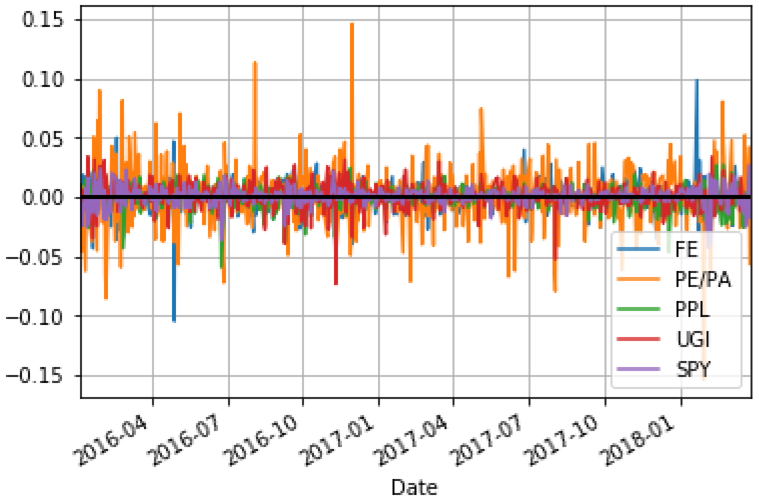
In [32]:
```
stock_return = stocks.apply(lambda x: x / x[0])
stock_return.plot(grid = True).axhline(y = 1, color = "black", lw = 2)
```

Out[32]:
```
<matplotlib.lines.Line2D at 0xc5f88d0>
```



In [33]:
```
stock_change = stocks.apply(lambda x: np.log(x) - np.log(x.shift(1)))
stock_change.plot(grid=True).axhline(y = 0, color = "black", lw = 2)
```

Out[33]:
```
<matplotlib.lines.Line2D at 0xc77cf30>
```



In [34]:
```
stock_change_apr = stock_change * 252 * 100    # There are 252 trading day
s in a year; the 100 converts to percentages
stock_change_apr.tail()
```

Out[34]:

Date	FE	PE/PA	PPL	UGI	SPY
2018-03-21	237.877873	1323.717854	-37.004412	40.519131	-48.411403
2018-03-22	286.910120	-579.752385	92.409344	-52.108002	-637.935186
2018-03-23	-524.787993	-165.487643	-175.868482	-361.924220	-542.935322
2018-03-26	186.033622	1065.583325	295.496240	222.440469	680.184780
2018-03-27	397.206122	-1418.556291	409.765126	214.716325	-432.383866

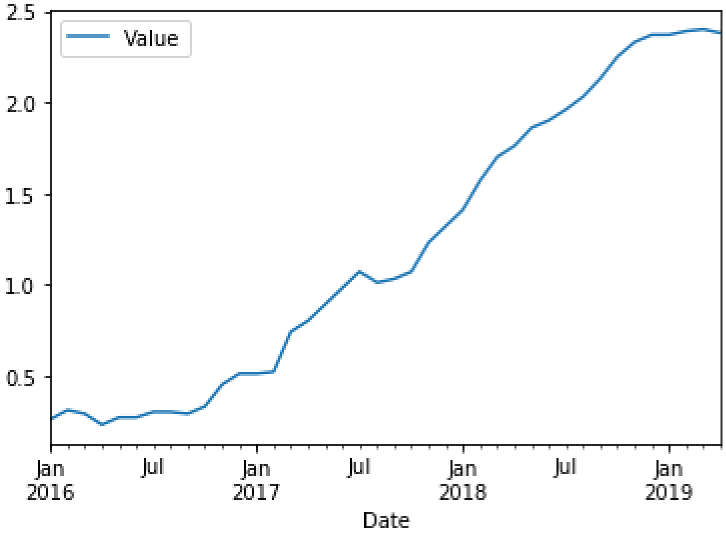
In [35]:
```
tbill = quandl.get("FRED/TB3MS", start_date=start, end_date=end)
tbill.tail()
```

Out[35]:

Date	Value
2018-12-01	2.37
2019-01-01	2.37
2019-02-01	2.39
2019-03-01	2.40
2019-04-01	2.38

In [36]:
```
tbill.plot()
```

Out[36]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xe1e5d0>
```



```
In [43]: rrf = tbill.iloc[-1, 0];rrf
```

```
Out[43]: 2.38
```

```
In [44]: smcorr = stock_change_apr.drop("SPY", 1).corrwith(stock_change_apr.SPY); s
         mcorr
```

```
Out[44]: FE       0.251570
         PE/PA    0.403995
         PPL      0.347617
         UGI      0.380425
         dtype: float64
```

```
In [46]: sy = stock_change_apr.drop("SPY", 1).std();
         sx = stock_change_apr.SPY.std();
         sy
```

```
Out[46]: FE       353.846198
         PE/PA    648.748158
         PPL      260.571441
         UGI      284.418180
         dtype: float64
```

```
In [47]: sx
```

```
Out[47]: 188.0163410157881
```

```
In [48]: ybar = stock_change_apr.drop("SPY", 1).mean() - rrf
         xbar = stock_change_apr.SPY.mean() - rrf
         ybar
```

```
Out[48]: FE        6.264508
         PE/PA    14.605897
         PPL      -7.549894
         UGI      10.681634
         dtype: float64
```

```
In [49]: xbar
```

```
Out[49]: 9.301201799874118
```

```
In [50]: beta = smcorr * sy / sx
         alpha = ybar - beta * xbar
         beta
```

```
Out[50]: FE        0.473453
         PE/PA     1.393980
         PPL       0.481762
         UGI       0.575481
         dtype: float64
```

```
In [51]: alpha
```

```
Out[51]: FE         1.860822
         PE/PA      1.640206
         PPL      -12.030856
         UGI        5.328969
         dtype: float64
```

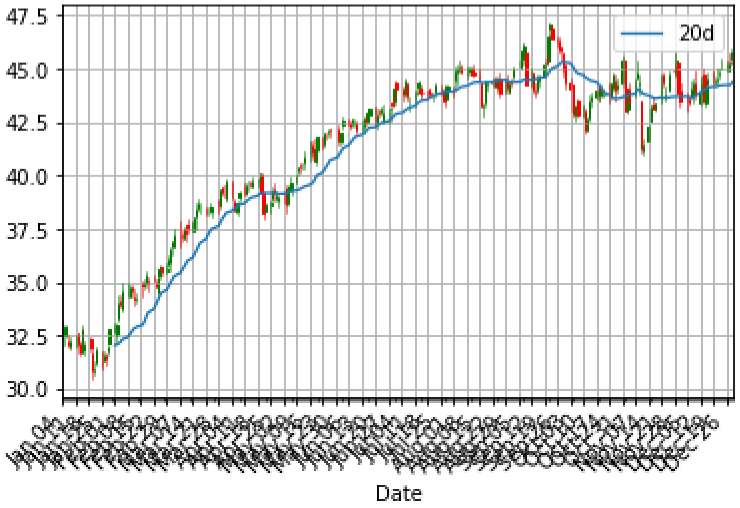
```
In [54]: sharpe = (ybar - rrf)/sy; sharpe
```

```
Out[54]: FE        0.010978
         PE/PA     0.018845
         PPL      -0.038108
         UGI       0.029188
         dtype: float64
```

```
In [55]: (xbar - rrf)/sx
```

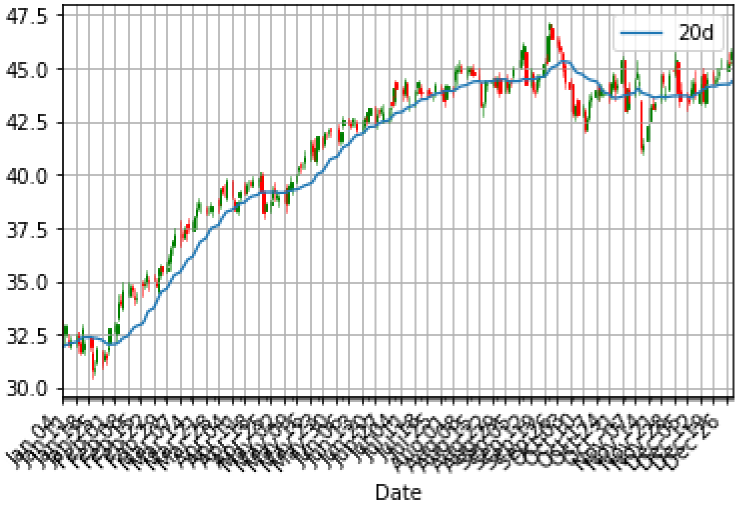
```
Out[55]: 0.03681170350662728
```

```
In [58]: united_gas["20d"] = np.round(united_gas["Adj. Close"].rolling(window = 20,
          center = False).mean(), 2)
         pandas_candlestick_ohlc(united_gas.loc['2016-01-04':'2016-12-31',:], other
         series = "20d", adj=True)
```



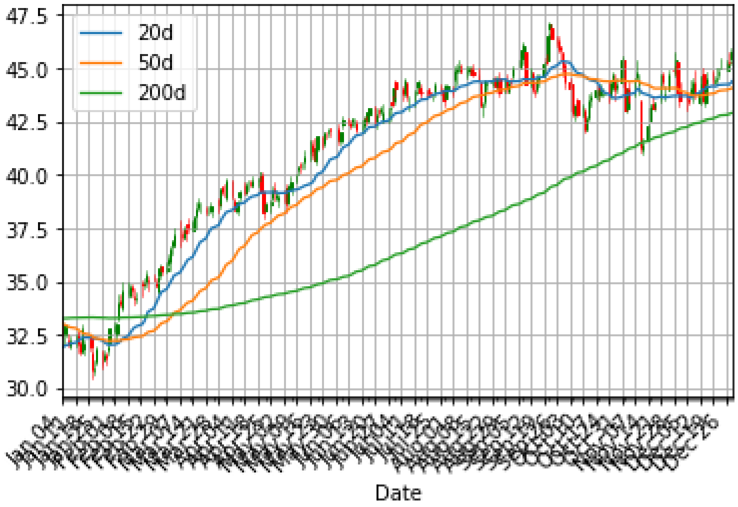
```
In [59]: start = datetime.datetime(2010,1,1)
         united_gas = quandl.get("WIKI/UGI", start_date=start, end_date=end)
```

```
united_gas["20d"] = np.round(united_gas["Adj. Close"].rolling(window = 20,
 center = False).mean(), 2)

pandas_candlestick_ohlc(united_gas.loc['2016-01-04':'2016-12-31',:], other
series = "20d", adj=True)
```



In [60]:

```
united_gas["50d"] = np.round(united_gas["Adj. Close"].rolling(window = 50,
 center = False).mean(), 2)
united_gas["200d"] = np.round(united_gas["Adj. Close"].rolling(window = 20
0, center = False).mean(), 2)

pandas_candlestick_ohlc(united_gas.loc['2016-01-04':'2016-12-31',:], other
series = ["20d", "50d", "200d"], adj=True)
```



In [ ]: