In [1]:
```
import pandas as pd
import gc
from tqdm import tqdm

# plotly visualization
import plotly.plotly as py
from plotly.offline import init_notebook_mode, plot,
import plotly.graph_objs as go
import cufflinks
# offline mode
cufflinks.go_offline()
# Set the global theme for cufflinks
cufflinks.set_config_file(world_readable=True, theme=
%matplotlib inline
```

1 Data Load

```
# get rid of duplicated data
df_US=pd.read_csv("csv/wrds_US.csv",index_col='DATE')
df_US =
df_US.loc[~df_US.index.duplicated(keep='first')]
df_US.to_csv("csv/wrds_US_1.csv")
```

In [2]:
```
df_US=pd.read_csv("csv/wrds_US_1.csv",index_col='DATE
df_US.index = pd.to_datetime(df_US.index)
```

In [73]:
```

```

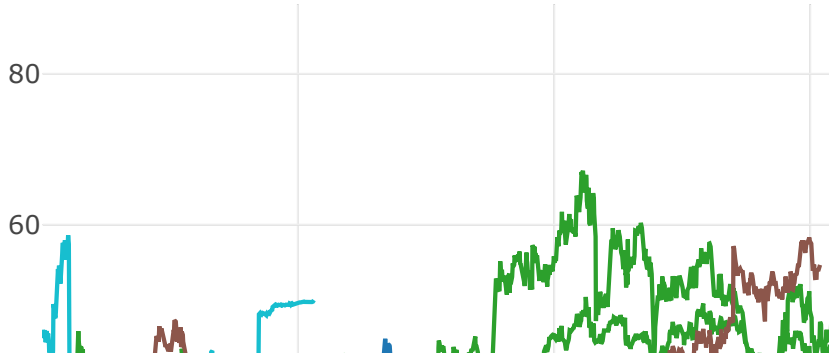
In [3]:
```
df_US.iloc[:5,:10]
```

Out[3]:

DATE	INTEL CORP	SUPERTEX INC	LINEAR TECHNOLOGY CORP	ON SEMICONDUCTOR CORP	ZILOG INC
2010-01-04	20.879999	29.000000	30.940001	8.87	3.53
2010-01-05	20.870001	28.299999	30.959999	8.79	3.53
2010-01-06	20.799999	28.190001	30.480000	8.90	3.53
2010-01-07	20.600000	27.870001	30.500000	8.89	3.54
2010-01-08	20.830000	28.540001	30.889999	8.89	3.53

1.1 Visualization

In [4]:
```
# number of companies to be displayed
n=10
trace=[]
for i in range(len(df_US.columns[n])):
    trace_tmp=go.Scatter(x=df_US.index,y=df_US[df_US.
    #trace_tmp= go.Scatter(x=df_temp.index,y=df_temp[
    trace.append(trace_tmp)
layout= go.Layout(images= [dict(
                    xref= "x",
                    yref= "y",
                    x= 'Jan 2010',
                    y= 0,
                    opacity= 0.5,
                    layer= "below")])

fig=go.Figure(data=trace,layout=layout)
#py.plot(fig,filename='EXAMPLES/background')
#fig = go.Figure(data=data, layout=layout)
iplot(fig, filename='economic.html')
```



2 Var Model

We are interested in modeling a $T \times K$ multivariate time series Y, where T denotes the number of observations and K the number of variables. One way of estimating relationships between the time series and their lagged values is the vector autoregression process:

$$Y_t = A_1 Y_{t-1} + \ldots + A_p Y_{t-p} + u_t$$
$$u_t \sim \text{Normal}(0, \Sigma_u)$$

where A_i is $K \times K$ matrix and

$$Y_t = [Y_{1,t}, Y_{2,t}, \cdots, Y_{K,t}]^T$$
$$Y_{i,t} = i \text{ target's value at time } t$$

2.1 Stationary Checking

The VAR class assumes that the passed time series are stationary. Non-stationary or trending data can often be transformed to be stationary by first-differencing or some other method.

Use Johnanson test to do Stationary Checking,

Reference

1. Lütkepohl, H. 2005. New Introduction to Multiple Time Series Analysis. Springer.

In [7]:
```
import numpy as np
#fit the model
from statsmodels.tsa.vector_ar.var_model import VAR
from statsmodels.tsa.vector_ar.vecm import coint_joha
from statsmodels.tsa.vector_ar.var_model import VARRe
```

In [17]:
```
n=4
df_VAR=df_US.iloc[:,:n]
```

While the company was **delisted**, we fill the nan values by the last non-nan value:

In [22]:
```
# fill nan with the last non-nan value
df_VAR.fillna(method='ffill',inplace=True)
```

In [43]:
```
def check_stationary(df,n=4):
    for lag in range(1,n):
          eigs=coint_johansen(df,-1,lag).eig
          print("Lag, %s: %s " %(lag,eigs))
```

In [44]:
```
check_stationary(df_VAR)
```

```
Lag, 1: [0.00673444 0.00478626 0.00210777 0.00037067]
Lag, 2: [0.00686409 0.00473238 0.00211952 0.00034491]
Lag, 3: [0.00718108 0.00481975 0.00212579 0.00035774]
```

In [45]:
```
# take the difference over logarithm of consective cl
df_diff = np.log(df_VAR).diff().dropna()
```

In [46]:
```
check_stationary(df_diff)
```

```
Lag, 1: [0.37255351 0.35859158 0.34566792 0.32255974]
Lag, 2: [0.29620016 0.27689655 0.26848356 0.23154389]
Lag, 3: [0.25009186 0.23262647 0.22714355 0.19162413]
```

Conclusion

The native prices data are **not** stationary for lag period being 1,2,3, $p < 0.01$; but are stationary for any lag, $p > 0.05$.

2.2 Model Fitting

In [47]:
```
#creating the train and validation set
train = df_VAR[:int(0.8*(len(df_VAR)))]
valid = df_VAR[int(0.8*(len(df_VAR))):]

train.index = pd.DatetimeIndex(train.index.values,
                               freq=train.index.infer

# Var Model
model = VAR(endog=train)
model_fit = model.fit()

# make prediction on validation
prediction = model_fit.forecast(model_fit.y, steps=le
```

```
/Users/cch/anaconda36/anaconda/lib/python3.6/site-pac
kages/statsmodels/tsa/base/tsa_model.py:225: ValueWar
ning:

A date index has been provided, but it has no associa
ted frequency information and so will be ignored when
e.g. forecasting.
```

In [48]:
```
model_fit.summary()
```

Out[48]:
```
  Summary of Regression Results
==================================
Model:                         VAR
Method:                        OLS
Date:           Sun, 02, Jun, 2019
Time:                     08:45:20
```

```
---------------------------------------------------
---------------
No. of Equations:         4.00000    BIC:
-8.55513
Nobs:                     1810.00    HQIC:
-8.59348
Log likelihood:          -2455.71    FPE:
0.000181200
AIC:                     -8.61591    Det(Omega_mle):
0.000179211
---------------------------------------------------
---------------
Results for equation INTEL CORP
===================================================
=======================================
                               coefficient       std.
error           t-stat            prob
---------------------------------------------------
---------------------------------------
const                           0.083863         0.
050869           1.649          0.099
L1.INTEL CORP                   0.993620         0.
003128         317.671          0.000
L1.SUPERTEX INC                 0.003405         0.
002558           1.331          0.183
L1.LINEAR TECHNOLOGY CORP       0.003358         0.
002048           1.640          0.101
L1.ON SEMICONDUCTOR CORP        -0.014256        0.
007257          -1.964          0.049
===================================================
=======================================

Results for equation SUPERTEX INC
===================================================
=======================================
                               coefficient       std.
error           t-stat            prob
---------------------------------------------------
---------------------------------------
const                           0.041519         0.
052736           0.787          0.431
L1.INTEL CORP                   0.000582         0.
003243           0.179          0.858
L1.SUPERTEX INC                 0.993992         0.
002652         374.771          0.000
L1.LINEAR TECHNOLOGY CORP       0.004083         0.
002123           1.923          0.054
L1.ON SEMICONDUCTOR CORP        -0.006355        0.
007523          -0.845          0.398
===================================================
=======================================

Results for equation LINEAR TECHNOLOGY CORP
===================================================
=======================================
                               coefficient       std.
error           t-stat            prob
```

```
------------------------------------------------------
---------------------------------------
const                                0.116366         0.
080320            1.449           0.147
L1.INTEL CORP                        0.000777         0.
004939            0.157           0.875
L1.SUPERTEX INC                      0.001888         0.
004040            0.467           0.640
L1.LINEAR TECHNOLOGY CORP            1.002852         0.
003234          310.141           0.000
L1.ON SEMICONDUCTOR CORP            -0.031299         0.
011458           -2.732           0.006
======================================================
=======================================

Results for equation ON SEMICONDUCTOR CORP
======================================================
=======================================
                                 coefficient       std.
error           t-stat            prob
------------------------------------------------------
---------------------------------------
const                                0.017427         0.
026486            0.658           0.511
L1.INTEL CORP                       -0.000634         0.
001629           -0.389           0.697
L1.SUPERTEX INC                     -0.000958         0.
001332           -0.719           0.472
L1.LINEAR TECHNOLOGY CORP            0.003373         0.
001066            3.164           0.002
L1.ON SEMICONDUCTOR CORP             0.988301         0.
003778          261.572           0.000
======================================================
=======================================

Correlation matrix of residuals
                          INTEL CORP  SUPERTEX INC  L
INEAR TECHNOLOGY CORP  ON SEMICONDUCTOR CORP
INTEL CORP                 1.000000      0.247750
0.524582               0.499451
SUPERTEX INC               0.247750      1.000000
0.270413               0.252339
LINEAR TECHNOLOGY CORP     0.524582      0.270413
1.000000               0.571318
ON SEMICONDUCTOR CORP      0.499451      0.252339
0.571318               1.000000
```

2.3 Result of VAR(1) Model

$$\begin{pmatrix} \text{INTEL CORP} \\ \text{SUPERTEX INC} \\ \text{LINEAR TECHNOLOGY CORP} \\ \text{ON SEMICONDUCTOR CORP} \end{pmatrix}_t = \begin{pmatrix} 0.083863 \\ 0.041519 \\ 0.116366 \\ 0.017427 \end{pmatrix} + \mathbf{A} \begin{pmatrix} \text{LINEA} \\ \text{ON SE} \end{pmatrix}$$

$$A = \begin{pmatrix} 0.993620, 0.003405, 0.0 \\ 0.000582, 0.993992, 0.0 \\ 0.000777, 0.001888, 1.0 \\ -0.000634, -0.000958, (\end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 1.000000, 0.247750, 0.52 \\ 0.247750, 1.000000, 0.27 \\ 0.524582, 0.270413, 1.00 \\ 0.499451, 0.252339, 0.57 \end{pmatrix}$$

2.4 Predition

In [99]:
```
#make final predictions
days=3
model = VAR(endog=df_VAR)
model_fit = model.fit()
yhat = model_fit.forecast(model_fit.y, steps=days)
print(yhat)
```

```
[[46.9046344  33.00529444 65.02776061 16.52268892]
 [46.87967838 33.0304349  65.05530362 16.53539307]
 [46.8551278  33.05542154 65.08263028 16.54811152]]

/Users/cch/anaconda36/anaconda/lib/python3.6/site-pac
kages/statsmodels/tsa/base/tsa_model.py:225: ValueWar
ning:

A date index has been provided, but it has no associa
ted frequency information and so will be ignored when
e.g. forecasting.
```

In [96]:
```
yhat[:3,1]
```

Out[96]:
```
array([33.00529444, 33.0304349 , 33.05542154])
```

In [61]:
```
# Print one day prediction
print(" Prediction end at %10s\n  ---" %(df_VAR.index
print("%26s  %7s  %11s "  %('Company',  'Close','Pred
for i in range(len(df_VAR.columns)):
    print("%26s  %8.3f  %8.2f" %(df_VAR.columns[i],df
```

```
Prediction end at 2018-12-31 00:00:00
 ---
                   Company    Close    Predition
                INTEL CORP    46.930      46.90
              SUPERTEX INC    32.980      33.01
  LINEAR TECHNOLOGY CORP      65.000      65.03
   ON SEMICONDUCTOR CORP      16.510      16.52
```

In [ ]:
```
```

In [131]:
```
# Preditions during period
def predict_pr(df,p):
    days=p.shape[0]
    print(" Prediction end at %10s\n  ---" %(df_VAR.i
    day=' '+str(days)+'-days Prediction'
    print("%26s  %7s  %11s "  %('Company',  'Close',d
    for i in range(len(df.columns)):
        print("%26s  %8.3f  %s" %(df.columns[i],df.il
```

In [132]:
```
predict_pr(df_VAR,yhat)
```

```
Prediction end at 2018-12-31 00:00:00
 ---
                   Company    Close   3-days Predicti
on
                INTEL CORP    46.930  [46.9  46.88 46
.86]
              SUPERTEX INC    32.980  [33.01 33.03 33
.06]
  LINEAR TECHNOLOGY CORP      65.000  [65.03 65.06 65
.08]
   ON SEMICONDUCTOR CORP      16.510  [16.52 16.54 16
.55]
```

2.5 PyPortolioOpt

The defaulted structure of dataframe is in the format:

- index, date
- closed price of each company is listed in column with respect to the date index

In [5]:
```
from pypfopt.efficient_frontier import EfficientFront
from pypfopt import risk_models
from pypfopt import expected_returns
```

In [6]:
```
# Calculate expected returns and sample covariance
mu = expected_returns.mean_historical_return(df_US)
S = risk_models.sample_cov(df_US)
```

In [26]:
```
def portfolio(data):
    print("Company      ","Weight","\n----\n")
    for x in data:
        if (data[x]>0):
            print (x,": ",data[x])
    print("----")
```

In [28]:
```
# Optimise for maximal Sharpe ratio
ef = EfficientFrontier(mu, S)
raw_weights = ef.max_sharpe()
cleaned_weights = ef.clean_weights()
#print(cleaned_weights)
portfolio(cleaned_weights)
ef.portfolio_performance(verbose=True)
```

```
Company       Weight
----

TOWER SEMICONDUCTOR LTD :  0.00389
AUTHENTEC INC :  0.06456
TRIQUINT SEMICONDUCTOR INC :  0.02684
TECHWELL INC :  0.03704
HANWHA Q CELLS CO LTD :  0.00857
KIMBALL ELECTRONICS INC :  0.0608
NATIONAL SEMICONDUCTOR CORP :  0.03092
SEMILEDS CORP :  0.00596
EMCORE CORP :  0.0089
VIRAGE LOGIC CORP :  0.16809
MERCURY SYSTEMS INC :  0.09263
MONTAGE TECHNOLOGY GROUP LTD :  0.13146
ACTEL CORP :  0.05244
PLUG POWER INC :  0.00829
FREESCALE SEMICONDUCTOR LTD :  0.07264
SMART GLOBAL HOLDINGS INC :  0.07918
IKANOS COMMUNICATIONS INC :  0.00667
OCLARO INC :  0.00965
DAQO NEW ENERGY CORP :  0.01004
AMBARELLA INC :  0.04187
MOSYS INC :  0.00424
SILICON IMAGE INC :  0.00908
SOLAREDGE TECHNOLOGIES INC :  0.01769
EVERGREEN SOLAR INC :  0.00311
RAMTRON INTERNATIONAL CORP :  0.02924
NETLIST INC :  0.00031
APPLIED OPTOELECTRONICS INC :  0.00143
GIGPEAK INC :  0.01448
----
Expected annual return: 28.5%
Annual volatility: 13.8%
Sharpe Ratio: 1.93
```

Out[28]:
```
(0.2849944945066098, 0.13765278160784092, 1.925093640
7631247)
```

In [31]:
```
from pypfopt import discrete_allocation

latest_prices = discrete_allocation.get_latest_prices
allocation, leftover = discrete_allocation.portfolio(
    cleaned_weights, latest_prices, total_portfolio_v
)
#print(allocation)
portfolio(allocation)
print("Funds remaining: ${:.2f}".format(leftover))
```

```
104 out of 120 tickers were removed
Funds remaining: 871.28
Company       Weight
----

VIRAGE LOGIC CORP :  140
MONTAGE TECHNOLOGY GROUP LTD :  58
MERCURY SYSTEMS INC :  19
SMART GLOBAL HOLDINGS INC :  26
FREESCALE SEMICONDUCTOR LTD :  19
AUTHENTEC INC :  80
KIMBALL ELECTRONICS INC :  39
ACTEL CORP :  25
AMBARELLA INC :  11
TECHWELL INC :  20
NATIONAL SEMICONDUCTOR CORP :  12
RAMTRON INTERNATIONAL CORP :  94
TRIQUINT SEMICONDUCTOR INC :  9
SOLAREDGE TECHNOLOGIES INC :  5
GIGPEAK INC :  47
DAQO NEW ENERGY CORP :  4
----
Funds remaining: $871.28
```

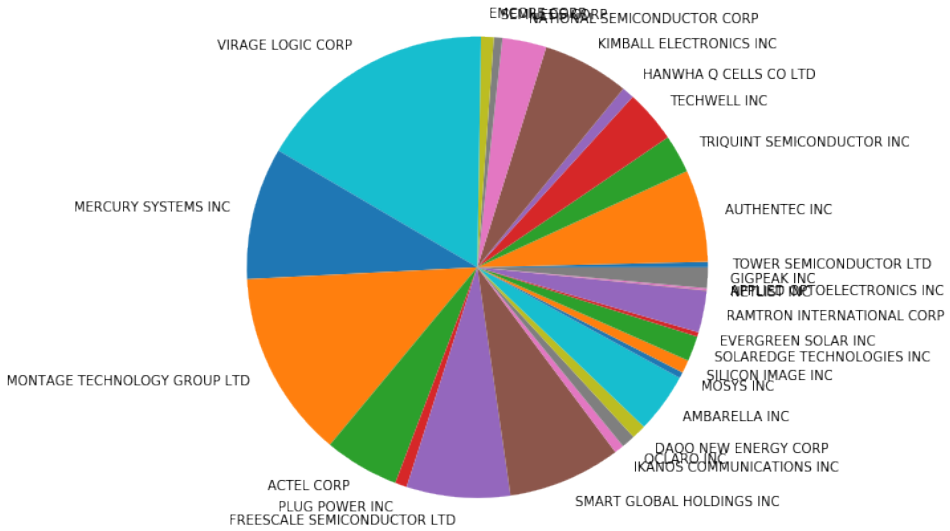
In [81]:
```
import matplotlib.pyplot as plt
def vis_portfolio(data):
    keys,values=[],[]
    new= { k:v for k, v in data.items() if v > 0 }

    for key, value in new.items():
        keys.append(key)
        values.append(float(value))

    plt.figure(figsize=(12,8))
    plt.pie([float(v) for v in values], labels=[k for
    plt.show()
```

In [88]:

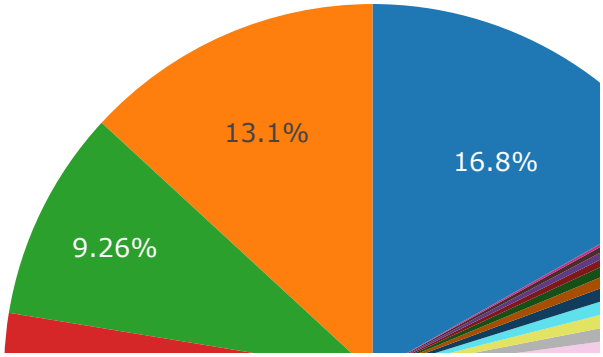
```
def vis_portfolio_plotly(data):
    keys,values=[],[]
    new= { k:v for k, v in data.items() if v > 0 }

    for key, value in new.items():
        keys.append(key)
        values.append(float(value))

    trace = go.Pie(go.Pie(labels=keys, values=values)
    iplot([trace], filename='basic_pie_chart')
```

In [89]:

```
vis_portfolio(cleaned_weights)
```



In [90]:
```
vis_portfolio_plotly(cleaned_weights)
```



In [ ]:
```
counts = Counter(cleaned_weights)
plt.pie([float(v) for v in values], labels=[float(k)
           autopct=None)
```

3 References