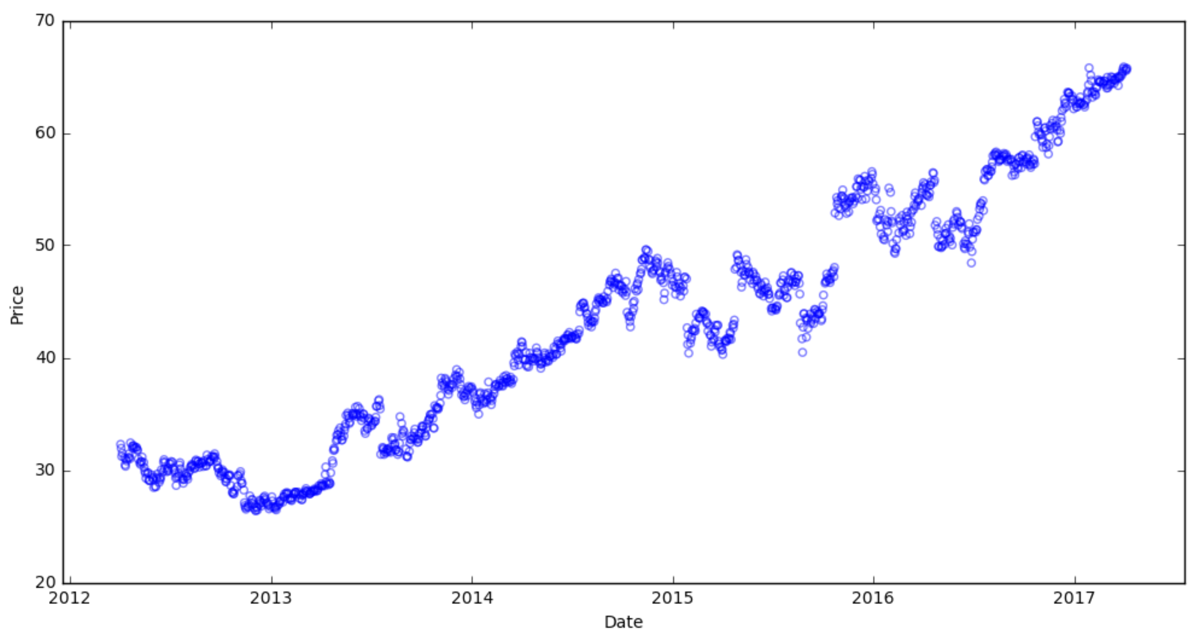
Big Data Tools and Techniques Mining Financial, Operational and Social Network Data TERM PROJECT

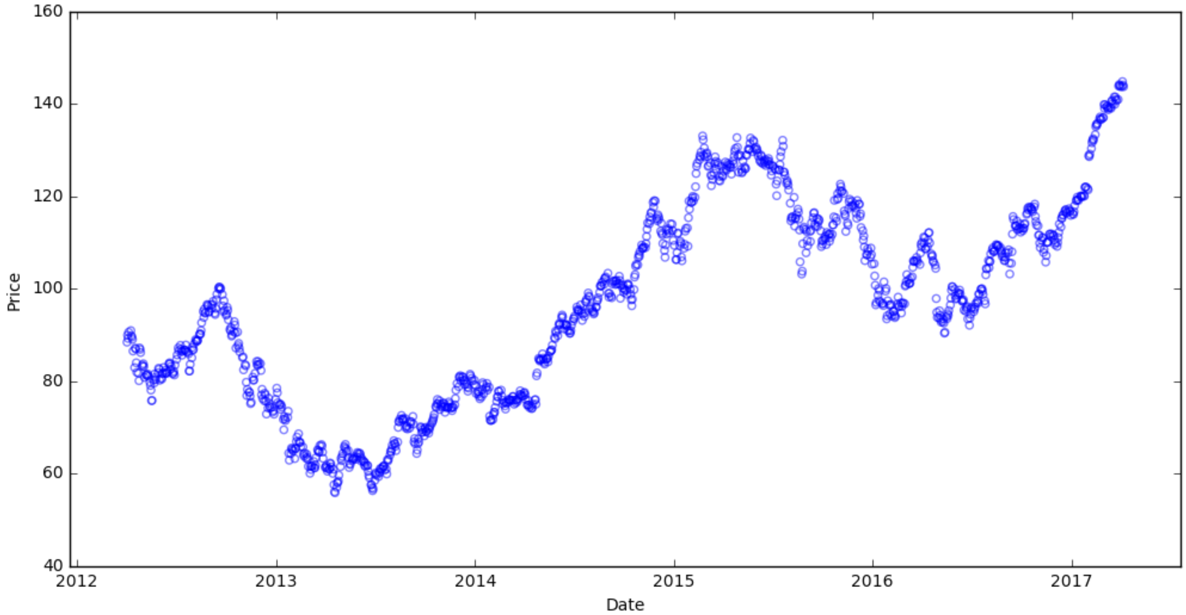
In [321]:
```
%matplotlib inline
from __future__ import division
from pandas import Series, DataFrame
import pandas as pd
from numpy.random import randn
import numpy as np
np.set_printoptions(precision=4, suppress=True)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(12, 6))
import pandas.io.data as web
from matplotlib.pyplot import scatter,xlabel,ylabel,savefig
import statsmodels.api as sm
from datetime import datetime
```

In [329]:
```
aapl = web.get_data_google('AAPL', '2012-04-01')
aapl['Rows'] = (aapl.index - datetime.strptime('2012-04-01',date_format)).days
msft = web.get_data_google('MSFT', '2012-04-01')
msft['Rows'] = (msft.index - datetime.strptime('2012-04-01',date_format)).days
```

In [330]:
```
scatter(msft.index,msft.Close,marker='o',edgecolor='b',facecolor='none',alpha=
0.5)
xlabel('Date')
ylabel('Price')
savefig('Microsoft.png',fmt='png',dpi = 100)
```



In [331]:
```
scatter(aapl.index,aapl.Close,marker='o',edgecolor='b',facecolor='none',alpha=
0.5)
xlabel('Date')
ylabel('Price')
savefig('APPLE.png',fmt='png',dpi = 100)
```



In [332]:

```
date_format = "%Y-%m-%d"
yaapl = aapl.Close
Xaapl= aapl['Rows']
Xaapl = sm.add_constant(Xaapl)
estaapl = sm.OLS(yaapl, Xaapl).fit()
estaapl.summary()
```

Out[332]:

OLS Regression Results

Dep. Variable:	Close	**R-squared:**	0.541
Model:	OLS	**Adj. R-squared:**	0.541
Method:	Least Squares	**F-statistic:**	1487.
Date:	Fri, 07 Apr 2017	**Prob (F-statistic):**	1.61e-215
Time:	01:17:33	**Log-Likelihood:**	-5165.4
No. Observations:	1262	**AIC:**	1.033e+04
Df Residuals:	1260	**BIC:**	1.035e+04
Df Model:	1		
Covariance Type:	nonrobust		

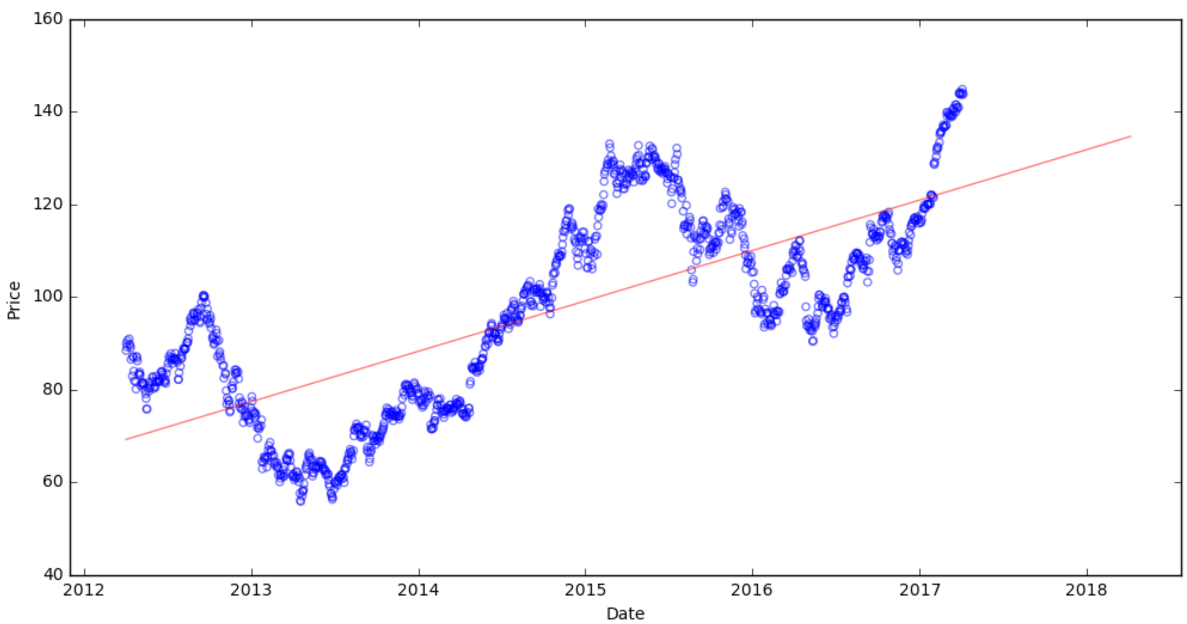
	coef	std err	t	P>|t|	[95.0% Conf. Int.]
const	69.1448	0.818	84.573	0.000	67.541 70.749
Rows	0.0298	0.001	38.566	0.000	0.028 0.031

Omnibus:	570.188	**Durbin-Watson:**	0.011
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	79.713
Skew:	0.251	**Prob(JB):**	4.90e-18
Kurtosis:	1.876	**Cond. No.**	2.12e+03

In [334]:
```
%pylab inline
plt.rc('figure', figsize=(12, 6))
Xaaplpred = Xaapl.copy()
#print Xaaplpred.xs(datetime.date(2016, 1, 26))
daterange = pd.date_range(datetime.datetime.now(), datetime.datetime.now() + d
atetime.timedelta(days=365))
for d in daterange:
    Xaaplpred.loc[d.date()]= [1,(d - datetime.datetime.strptime('2012-04-01',d
ate_format)).days]
yaaplpred = estaapl.predict(Xaaplpred)
scatter(aapl.index,aapl.Close,marker='o',edgecolor='b',facecolor='none',alpha=
0.5)
xlabel('Date')
ylabel('Price')
savefig('APPLE.png',fmt='png',dpi = 100)
plt.plot(Xaaplpred.index, yaaplpred, 'r', alpha=0.5)  # Add the regression lin
e
```

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

Out[334]:
```
[<matplotlib.lines.Line2D at 0x125a2630>]
```



In [335]:

```python
ymsft = msft.Close
Xmsft= msft['Rows']
Xmsft = sm.add_constant(Xmsft)
estmsft = sm.OLS(ymsft, Xmsft).fit()
estaapl.summary()
```

Out[335]:

OLS Regression Results

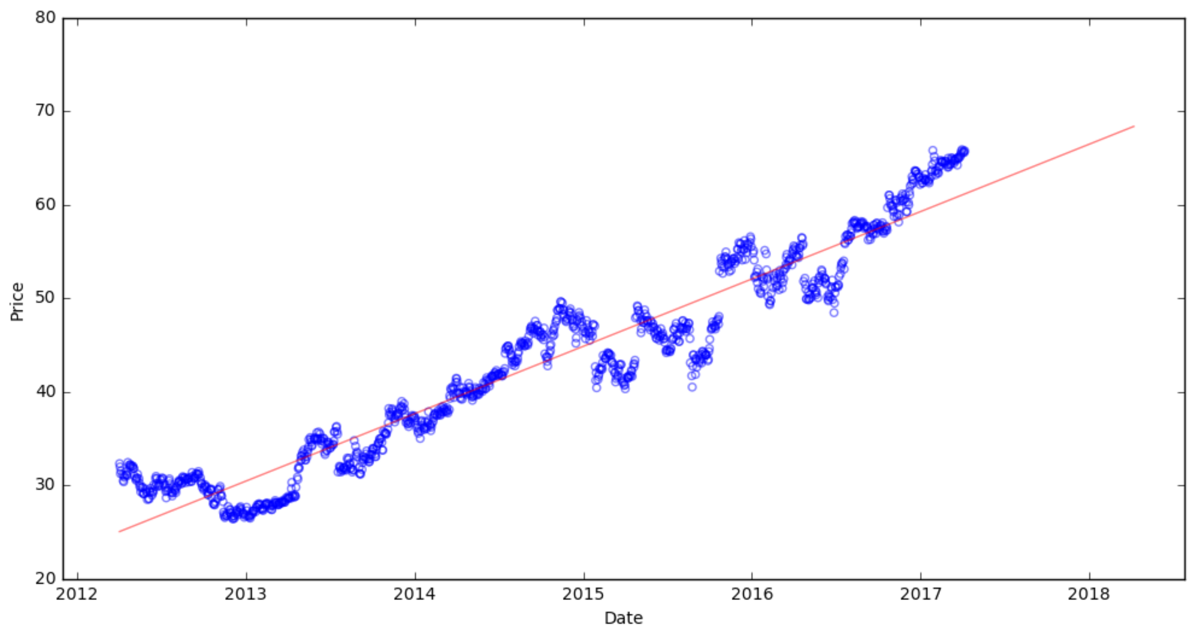
Dep. Variable:	Close	**R-squared:**	0.541
Model:	OLS	**Adj. R-squared:**	0.541
Method:	Least Squares	**F-statistic:**	1487.
Date:	Fri, 07 Apr 2017	**Prob (F-statistic):**	1.61e-215
Time:	01:18:26	**Log-Likelihood:**	-5165.4
No. Observations:	1262	**AIC:**	1.033e+04
Df Residuals:	1260	**BIC:**	1.035e+04
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf. Int.]
const	69.1448	0.818	84.573	0.000	67.541 70.749
Rows	0.0298	0.001	38.566	0.000	0.028 0.031

Omnibus:	570.188	**Durbin-Watson:**	0.011
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	79.713
Skew:	0.251	**Prob(JB):**	4.90e-18
Kurtosis:	1.876	**Cond. No.**	2.12e+03

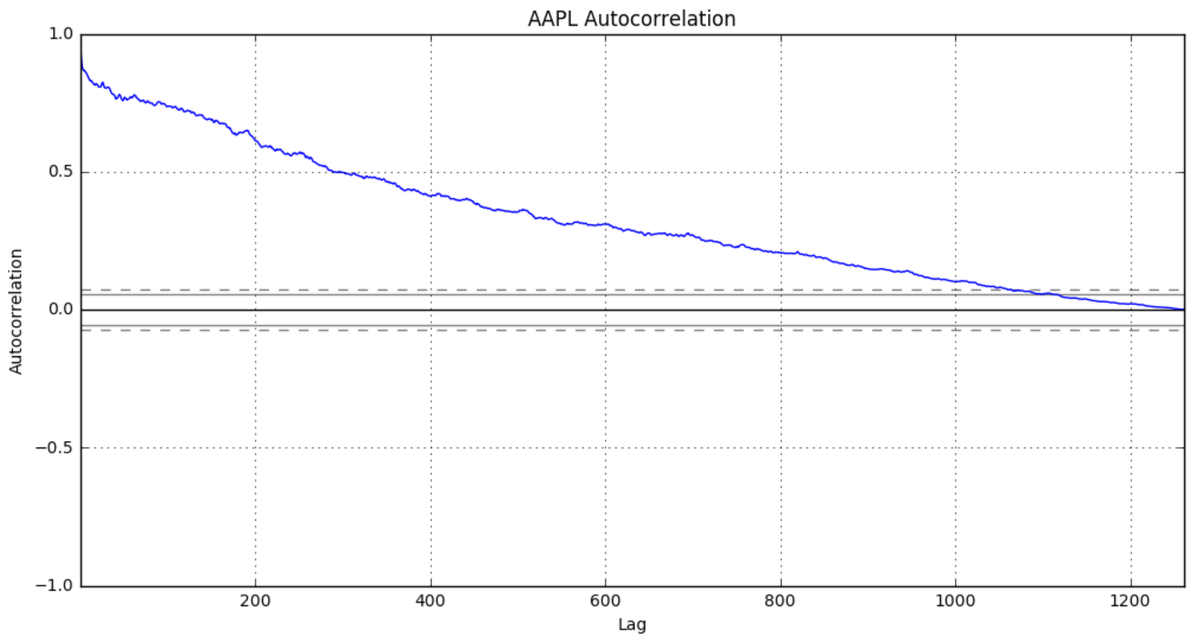
In [336]:
```
Xmsftpred = Xmsft.copy()
#print Xmsftpred.xs(datetime.date(2016, 1, 26))
daterange = pd.date_range(datetime.datetime.now(), datetime.datetime.now() + d
atetime.timedelta(days=365))
for d in daterange:
    Xmsftpred.loc[d.date()]= [1,(d - datetime.datetime.strptime('2012-04-01',d
ate_format)).days]
ymsftpred = estmsft.predict(Xmsftpred)
scatter(msft.index,msft.Close,marker='o',edgecolor='b',facecolor='none',alpha=
0.5)
xlabel('Date')
ylabel('Price')
savefig('Microsoft.png',fmt='png',dpi = 100)
plt.plot(Xmsftpred.index, ymsftpred, 'r', alpha=0.5)  # Add the regression lin
e
```

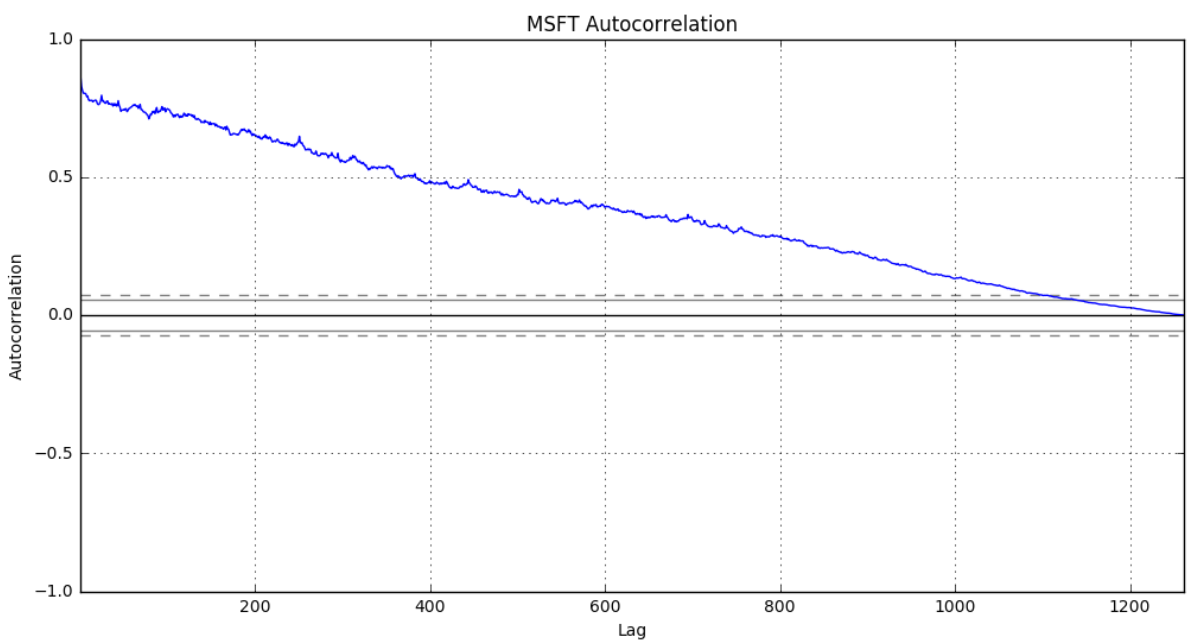
Out[336]:
```
[<matplotlib.lines.Line2D at 0x16b1ac50>]
```



In [338]:
```
#import matplotlib.pyplot as plt
from pandas.tools.plotting import autocorrelation_plot
```

In [339]:
```python
# Plot the auto-correlation graph for each stock using Pandas' autocorrelation_plot() function
plt.figure()
autocorrelation_plot(aapl)
plt.title('AAPL Autocorrelation')
plt.figure()
autocorrelation_plot(msft)
plt.title('MSFT Autocorrelation')
plt.figure()
plt.show()
```

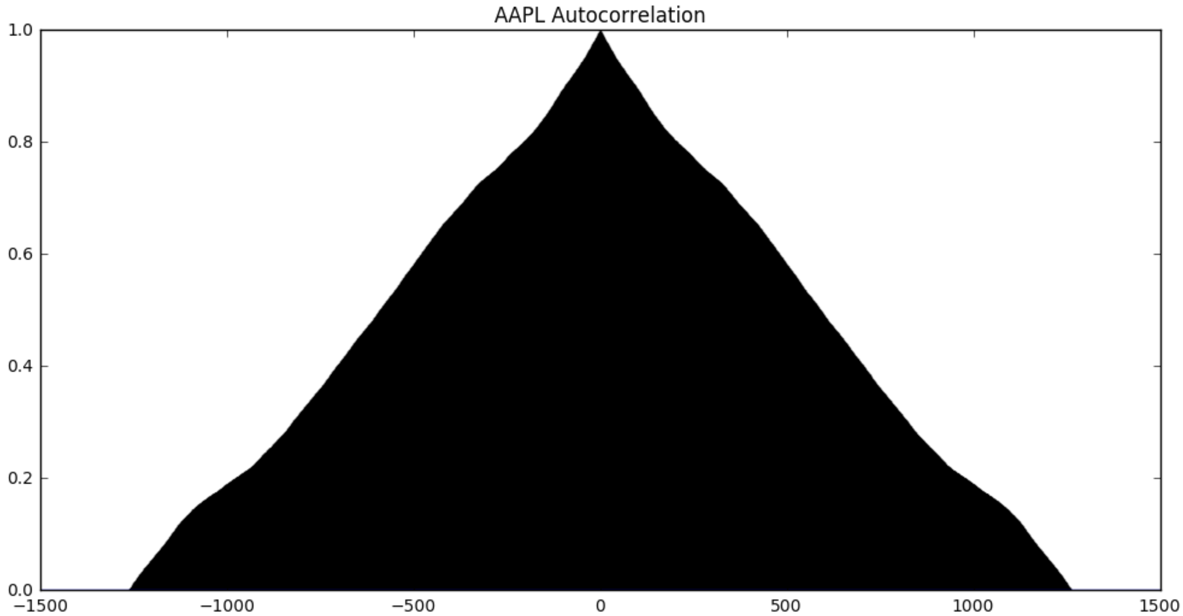





```
<matplotlib.figure.Figure at 0x15df04e0>
```

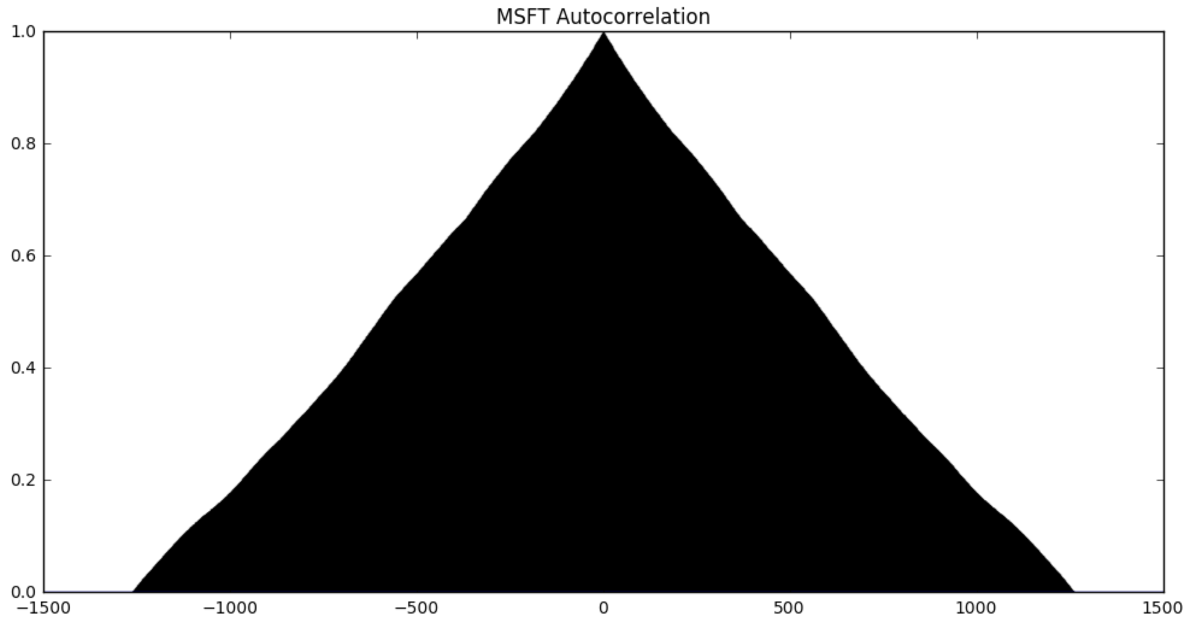
In [355]:
```python
# An alternative way of plotting correlation using matplotlib instead:
plt.title('AAPL Autocorrelation')
plt.acorr(aapl.Close, maxlags=len(aapl)-1)
# This plot will look different from the one generated by Pandas because:
#   (1) the correlation is calculated relative to 0 rather than to the average
of the data
#   (2) the lag is shown from -1200 to 1200 rather than 0 to 1200
```

Out[355]:
```
(array([-1261, -1260, -1259, ...,  1259,  1260,  1261]),
 array([ 0.001 ,  0.0021,  0.0031, ...,  0.0031,  0.0021,  0.001 ]),
 <matplotlib.collections.LineCollection at 0x2359f470>,
 <matplotlib.lines.Line2D at 0x23c88c88>)
```



In [356]:
```
plt.title('MSFT Autocorrelation')
plt.acorr(msft.Close, maxlags=len(aapl)-1)
```

Out[356]:
```
(array([-1261, -1260, -1259, ...,  1259,  1260,  1261]),
 array([ 0.0009,  0.0017,  0.0025, ...,  0.0025,  0.0017,  0.0009]),
 <matplotlib.collections.LineCollection at 0x1fbe5b38>,
 <matplotlib.lines.Line2D at 0x240a9b00>)
```



In [37]:
```
# All stocks are highly autocorrelated.  This is to be expected because yester
day's stock price
# gives you a lot of insight into today's: it's likely very close.  If they we
ren't autocorrelated
# that would mean that the prices jump all over the place from day to day: $5
 one day, $47 the next,
# $0.02 the day after that.
```

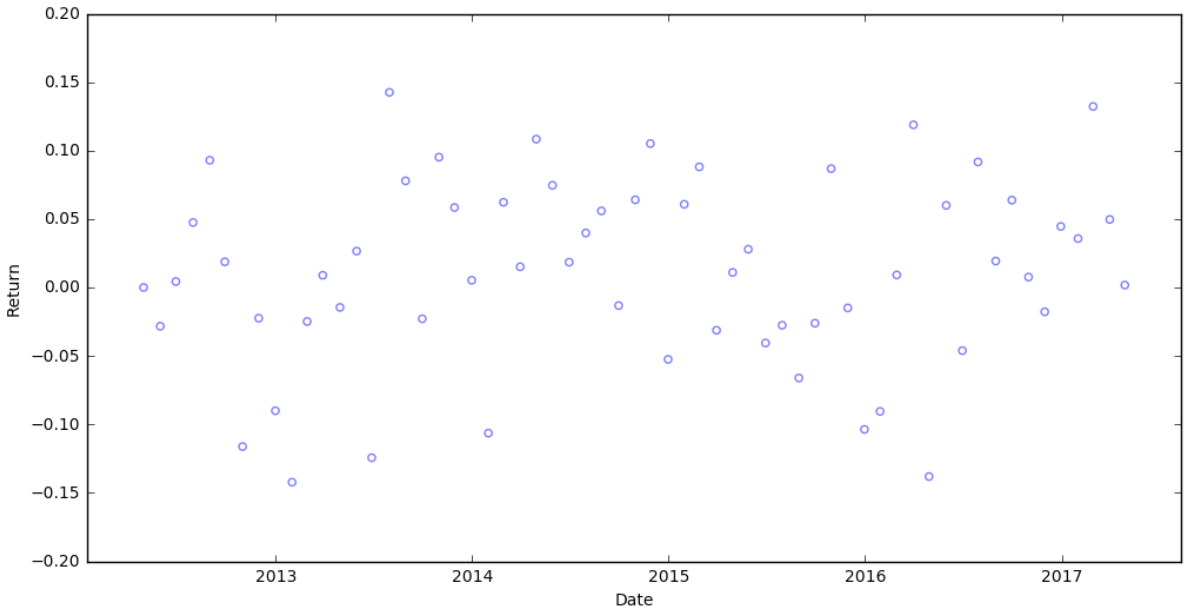
Task 3: "Calculate the monthly return over the period for each stock using the “shift trick” on the page titled Shifting in the Working with Time Series in Pandas deck"

In [428]:
```
# Resampling the data to get end of business month data points
aapl_monthend = aapl.Close.resample('BM').last()
msft_monthend = msft.Close.resample('BM').last()
Xaapl_monthend = Xaapl.resample('BM').last()
Xmsft_monthend = Xmsft.resample('BM').last()
# Check to make sure we got what we expected
#aapl_monthend
```

In [422]:

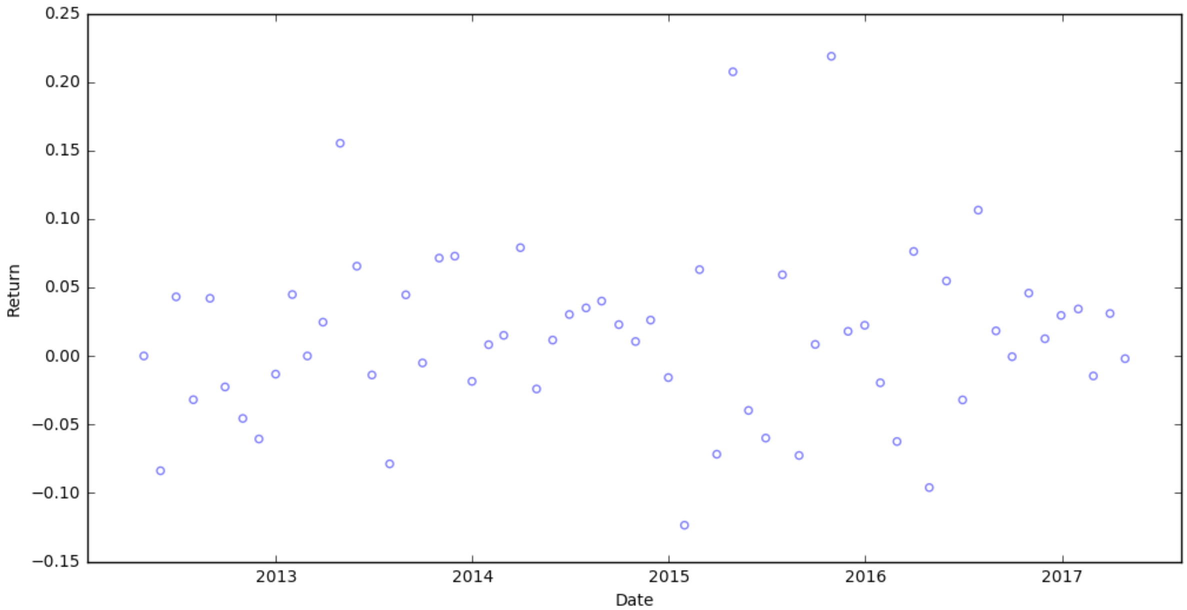
```
# calculating monthly returns for each stock
aapl_returns = aapl_monthend / aapl_monthend.shift(1) - 1
aapl_returns.name = 'AAPL'
aapl_returns.loc[datetime.date(2012,4,30)] = 0
scatter(aapl_returns.index,aapl_returns,marker='o',edgecolor='b',facecolor='no
ne',alpha=0.5)
xlabel('Date')
ylabel('Return')
savefig('APPLE Return.png',fmt='png',dpi = 100)

#aapl_returns
```



In [423]:

```
msft_returns = msft_monthend / msft_monthend.shift(1) - 1
msft_returns.name = 'MSFT'
#msft_returns
msft_returns.loc[datetime.date(2012,4,30)] = 0
scatter(msft_returns.index,msft_returns,marker='o',edgecolor='b',facecolor='no
ne',alpha=0.5)
xlabel('Date')
ylabel('Return')
savefig('Microsoft Return.png',fmt='png',dpi = 100)
```



In [424]:
```
# print aapl_returns
estaaplreturn = sm.OLS(aapl_returns , Xaapl_monthend).fit()
# Check the results
estaaplreturn.summary()
#estaapl.params
```

Out[424]:

OLS Regression Results

Dep. Variable:	AAPL	**R-squared:**	0.016
Model:	OLS	**Adj. R-squared:**	-0.001
Method:	Least Squares	**F-statistic:**	0.9351
Date:	Fri, 07 Apr 2017	**Prob (F-statistic):**	0.337
Time:	03:08:04	**Log-Likelihood:**	77.879
No. Observations:	61	**AIC:**	-151.8
Df Residuals:	59	**BIC:**	-147.5
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
const	-0.0040	0.018	-0.223	0.824	-0.040 0.032
Rows	1.588e-05	1.64e-05	0.967	0.337	-1.7e-05 4.87e-05

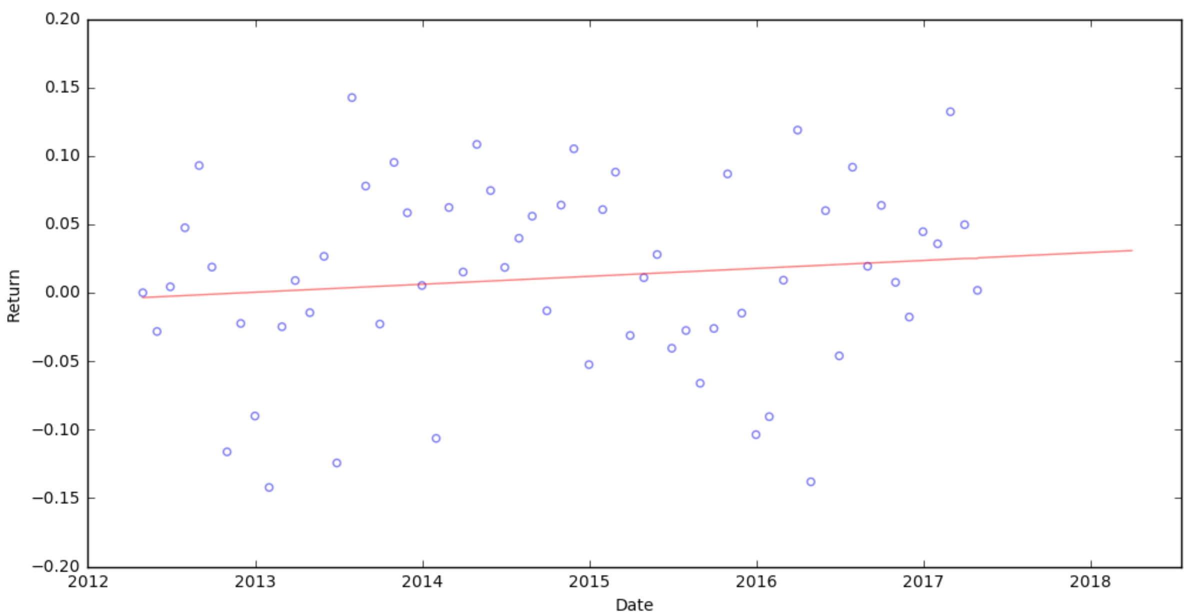
Omnibus:	1.573	**Durbin-Watson:**	1.954
Prob(Omnibus):	0.456	**Jarque-Bera (JB):**	1.584
Skew:	-0.338	**Prob(JB):**	0.453
Kurtosis:	2.594	**Cond. No.**	2.19e+03

In [425]:

```
Xaaplpred_me = Xaapl_monthend.copy()
#print Xaaplpred.xs(datetime.date(2016, 1, 26))
daterange = pd.date_range(datetime.datetime.now(), datetime.datetime.now() + d
atetime.timedelta(days=365), freq='M')
for d in daterange:
    Xaaplpred_me.loc[d.date()]= [1,(d - datetime.datetime.strptime('2012-04-0
1',date_format)).days]
yaaplpred_me = estaaplreturn.predict(Xaaplpred_me)
scatter(aapl_returns.index,aapl_returns,marker='o',edgecolor='b',facecolor='no
ne',alpha=0.5)
xlabel('Date')
ylabel('Return')
savefig('APPLE RETURN.png',fmt='png',dpi = 100)
plt.plot(Xaaplpred_me.index, yaaplpred_me, 'r', alpha=0.5)  # Add the regressi
on line
```

Out[425]:

```
[<matplotlib.lines.Line2D at 0x23c96da0>]
```



In [426]:

```
# print aapl_returns
estmsftreturn = sm.OLS(msft_returns , Xmsft_monthend).fit()
# Check the results
estmsftreturn.summary()
#estaapl.params
```

Out[426]:

OLS Regression Results

Dep. Variable:	MSFT	**R-squared:**	0.006
Model:	OLS	**Adj. R-squared:**	-0.011
Method:	Least Squares	**F-statistic:**	0.3706
Date:	Fri, 07 Apr 2017	**Prob (F-statistic):**	0.545
Time:	03:08:20	**Log-Likelihood:**	82.560
No. Observations:	61	**AIC:**	-161.1
Df Residuals:	59	**BIC:**	-156.9
Df Model:	1		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[95.0% Conf. Int.]**
const	0.0051	0.016	0.308	0.759	-0.028 0.038
Rows	9.257e-06	1.52e-05	0.609	0.545	-2.12e-05 3.97e-05

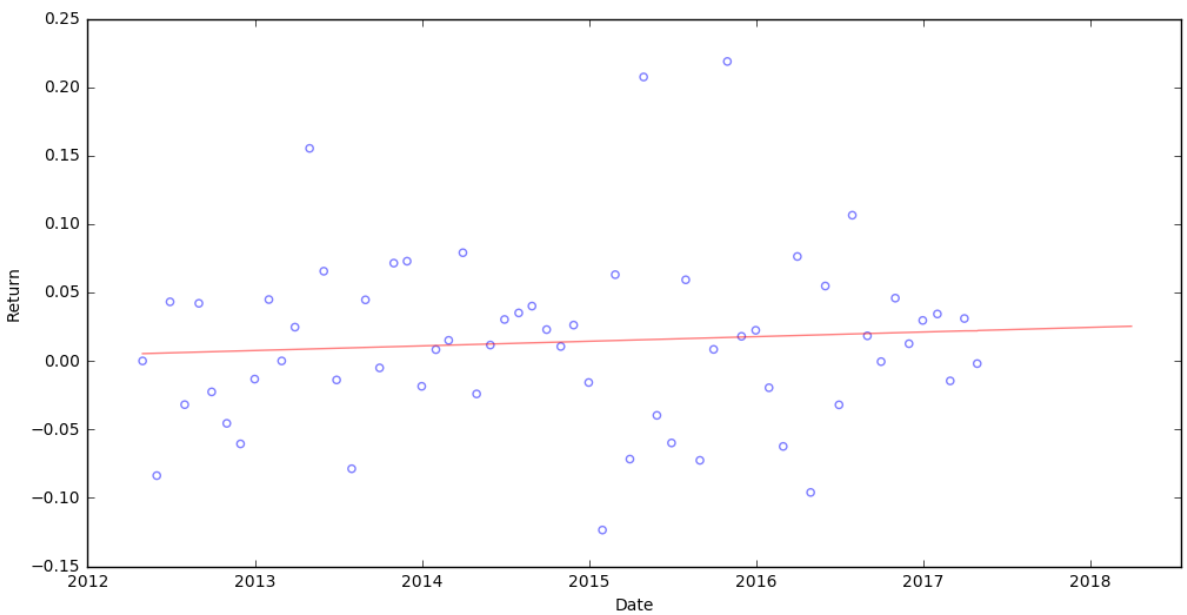
Omnibus:	12.122	**Durbin-Watson:**	2.435
Prob(Omnibus):	0.002	**Jarque-Bera (JB):**	15.103
Skew:	0.785	**Prob(JB):**	0.000525
Kurtosis:	4.864	**Cond. No.**	2.19e+03

In [427]:

```
Xmsftpred_me = Xmsft_monthend.copy()
#print Xaaplpred.xs(datetime.date(2016, 1, 26))
daterange = pd.date_range(datetime.datetime.now(), datetime.datetime.now() + d
atetime.timedelta(days=365), freq='M')
for d in daterange:
    Xmsftpred_me.loc[d.date()]= [1,(d - datetime.datetime.strptime('2012-04-0
1',date_format)).days]
ymsftpred_me = estmsftreturn.predict(Xmsftpred_me)
scatter(msft_returns.index,msft_returns,marker='o',edgecolor='b',facecolor='no
ne',alpha=0.5)
xlabel('Date')
ylabel('Return')
savefig('MICROSOFT RETURN.png',fmt='png',dpi = 100)
plt.plot(Xmsftpred_me.index, ymsftpred_me, 'r', alpha=0.5)  # Add the regressi
on line
```

Out[427]:

```
[<matplotlib.lines.Line2D at 0x1f2b95c0>]
```



In [374]:

```
# concatenating all 4 time series into a datafame for further analysis
df_to_analyse = pd.concat([aapl_returns, msft_returns], join='outer', axis = 1
)
```

In [375]:

```
# dropping first row with NAs
df_to_analyse.dropna()
```

Out[375]:

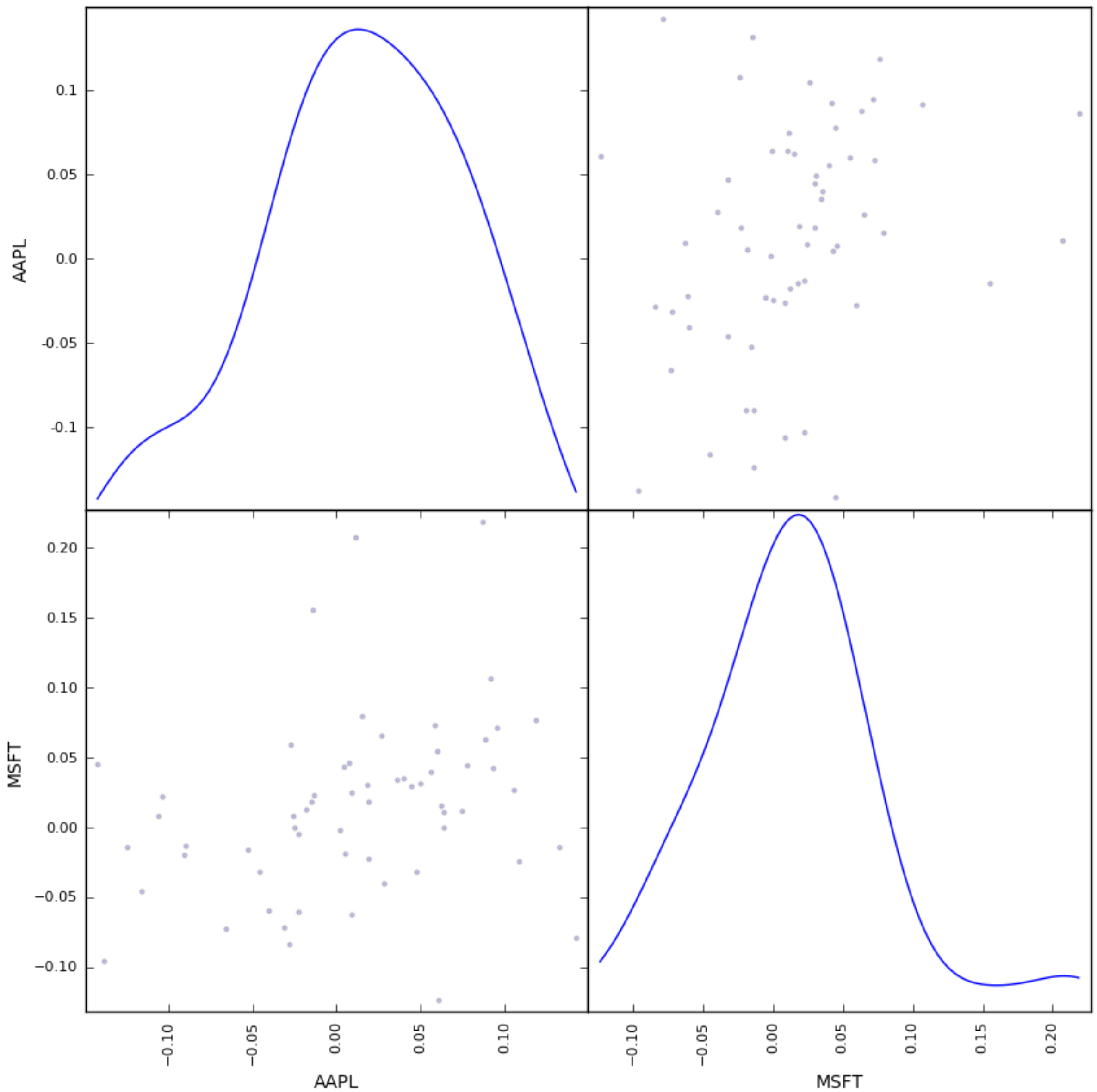
Date	AAPL	MSFT
2012-05-31	-0.028307	-0.083775
2012-06-29	0.004334	0.043168
2012-07-31	0.047465	-0.031932
2012-08-31	0.092917	0.042073
2012-09-28	0.018741	-0.022610
2012-10-31	-0.116238	-0.045605
2012-11-30	-0.022444	-0.060596
2012-12-31	-0.090055	-0.013269
2013-01-31	-0.142241	0.044826
2013-02-28	-0.024844	0.000000
2013-03-29	0.008909	0.024669
2013-04-30	-0.014562	0.155269
2013-05-31	0.026568	0.065539
2013-06-28	-0.124349	-0.013889
2013-07-31	0.142532	-0.078758
2013-08-30	0.077912	0.044618
2013-09-30	-0.022863	-0.005078
2013-10-31	0.095190	0.071450
2013-11-29	0.058386	0.072850
2013-12-31	0.005266	-0.018543
2014-01-31	-0.106386	0.008249
2014-02-28	0.062247	0.015044
2014-03-31	0.015110	0.079043
2014-04-30	0.108335	-0.024096
2014-05-30	0.074623	0.011605
2014-06-30	0.018474	0.030266
2014-07-31	0.039799	0.035063
2014-08-29	0.055926	0.040055
2014-09-30	-0.013217	0.022887
2014-10-31	0.064014	0.010542
2014-11-28	0.105146	0.026187
2014-12-31	-0.052513	-0.015768
2015-01-30	0.060727	-0.123524
2015-02-27	0.088083	0.063011
2015-03-31	-0.031248	-0.071719

Date	AAPL	MSFT
2015-04-30	0.010989	0.207409
2015-05-29	0.027917	-0.039766
2015-06-30	-0.040548	-0.059912
2015-07-31	-0.027593	0.059258
2015-08-31	-0.066129	-0.072620
2015-09-30	-0.026107	0.008422
2015-10-30	0.086785	0.218736
2015-11-30	-0.014932	0.017966
2015-12-31	-0.103676	0.022380
2016-01-29	-0.090535	-0.019576
2016-02-29	0.009143	-0.062443
2016-03-31	0.118803	0.076283
2016-04-29	-0.138126	-0.096060
2016-05-31	0.059966	0.054726
2016-06-30	-0.046116	-0.032075
2016-07-29	0.091678	0.106433
2016-08-31	0.019321	0.018323
2016-09-30	0.063808	-0.000519
2016-10-31	0.007586	0.045872
2016-11-30	-0.017771	0.012579
2016-12-30	0.044563	0.029585
2017-01-31	0.035751	0.034291
2017-02-28	0.132218	-0.014582
2017-03-31	0.049694	0.030997
2017-04-28	0.001733	-0.001964

In [376]:
```
from pandas.tools.plotting import scatter_matrix

#Scatter plot matrix
scatter_matrix(df_to_analyse.dropna(), alpha=0.2, figsize=(10, 10), diagonal=
'kde')
```

Out[376]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x0000000026059EB8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000002618EC88
>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000000002629BBE0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000000002634D048
>]], dtype=object)
```



In [46]:
```
# Any answer to this is right as long as the "why" makes sense.  Stocks with l
ow or negative
# correlation but similar returns over the period would have made a good combi
nation as they would
# provide equivalent return to one or the other but with less month-to-month v
ariability and so, in
# a sense, less risk.  Note however that this doesn't really say anything abou
t whether they will
# continue to be low- or anti-correlated in the future.
```

In [33]:
```
# Transpose the dataframe in preparation for plotting
df_to_analyse_t = df_to_analyse.T
df_to_analyse_t.dropna(axis=1,how='all')
```

Out[33]:

Date	2012-05-31 00:00:00	2012-06-29 00:00:00	2012-07-31 00:00:00	2012-08-31 00:00:00	2012-09-28 00:00:00	2012-10-31 00:00:00	2012-11-30 00:00:00	2012-12-31 00:00:00	201 00:(
AAPL	-0.010787	0.010905	0.045787	0.089169	0.002841	-0.107555	-0.016931	-0.090779	-0.14
MSFT	-0.088382	0.047962	-0.036613	0.045809	-0.034393	-0.040995	-0.067274	0.003381	0.02

2 rows × 60 columns

In [34]:
```
corr_plt = df_to_analyse_t.dropna(axis=1,how='all').values
corr_plt
```

Out[34]:
```
array([[-0.0108,  0.0109,  0.0458,  0.0892,  0.0028, -0.1076, -0.0169,
        -0.0908, -0.144 , -0.0309,  0.0029,  0.0002,  0.0158, -0.1183,
         0.1412,  0.0766, -0.0214,  0.0963,  0.0639,  0.0089, -0.1078,
         0.0513,  0.02  ,  0.0994,  0.0727,  0.0276,  0.0287,  0.0722,
        -0.0171,  0.072 ,  0.1012, -0.0719,  0.0614,  0.0964, -0.0314,
         0.0058,  0.041 , -0.0373, -0.0328, -0.0704, -0.0218,  0.0834,
        -0.01  , -0.1102, -0.0752, -0.0067,  0.1272, -0.1399,  0.0653,
        -0.0427,  0.0901,  0.0181,  0.0655,  0.0043, -0.0266,  0.048 ,
         0.0477,  0.1289,  0.0487,  0.    ],
       [-0.0884,  0.048 , -0.0366,  0.0458, -0.0344, -0.041 , -0.0673,
         0.0034,  0.0277,  0.0128,  0.0288,  0.1573,  0.0544, -0.0103,
        -0.0782,  0.049 , -0.0036,  0.0637,  0.0771, -0.0189,  0.0115,
         0.0124,  0.07  , -0.0144,  0.0134,  0.0186,  0.035 ,  0.0526,
         0.0205,  0.0127,  0.0183, -0.0284, -0.1302,  0.0854, -0.0727,
         0.1963, -0.0366, -0.0578,  0.0578, -0.0681,  0.017 ,  0.1893,
         0.0325,  0.0208, -0.007 , -0.0764,  0.0855, -0.097 ,  0.0628,
        -0.0345,  0.1077,  0.0138,  0.0024,  0.0403,  0.0057,  0.0312,
         0.0404, -0.0104,  0.0294, -0.002 ]])
```

In [377]:

```
# Plot the correlation of the returns
from numpy import corrcoef, sum, log, arange
from pylab import pcolor, show, colorbar, xticks, yticks

R = corrcoef(corr_plt)
pcolor(R)
colorbar()
yticks(arange(0.5,2),['AAPL', 'MSFT'])
xticks(arange(0.5,2),['AAPL', 'MSFT'])
show()
```