Dynamics of skewness in asset returns

Econophysics Research Project

Hypothesis : Predicting skew movements of assets would give an advantage is stock pricking

Definition:

Skewness of returns: *Skewness is "a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative, or even undefined." - Wikipedia*

Motivation

Nassim Taleb & Black Swan Theory:

- Black swan theory, first theoretisiezed by mathematicien Nassim Taleb is defined as "an event that comes as a surprise, has a major effect, and is often inappropriately rationalized after the fact with the benefit of hindsight." - Wikipedia

Why indexing Works by Heaton:

- Six page paper explaining why active equity managers tend to underperform a benchmark index, mainly due to the natural detency of having positivly skewed returns.
  https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2673262 (https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2673262)

Data Collecting

We start the research by collecting data from 2002 to 2017. We will use data for SPY - the ETF for the S&P500. Splitting the daily returns into months, we will end up with about 3816 return datapoints. Enough to hopefully state something conclusive. Of those datapoints, we will calculate the skewness of the returns as well as the mean.

The Data

In [25]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from statsmodels import regression
import statsmodels.api as sm

import scipy as sp
from scipy import stats
import itertools
```

In [26]:

```
start = '2002-01-01'
end   = '2017-01-01'

prices = get_pricing('SPY', fields = 'price',
                     start_date = start,
                     end_date= end,
                     frequency = 'daily')

spy = prices.pct_change()[1:]
```

In [27]:

```
#Returns.
m02_1, m02_2, m02_3, m02_4, m02_5, m02_6, m02_7, m02_8, m02_9, m02_10, m02_11, m
02_12 = (
spy.loc['2002-1-1':'2002-2-1'], spy.loc['2002-2-1':'2002-3-1'], spy.loc['2002-3-
1':'2002-4-1'],
spy.loc['2002-4-1':'2002-5-1'], spy.loc['2002-5-1':'2002-6-1'], spy.loc['2002-6-
1':'2002-7-1'],
spy.loc['2002-7-1':'2002-8-1'], spy.loc['2002-8-1':'2002-9-1'], spy.loc['2002-9-
1':'2002-10-1'],
spy.loc['2002-10-1':'2002-11-1'], spy.loc['2002-11-1':'2002-12-1'], spy.loc['200
2-12-1':'2003-1-1'])

m03_1, m03_2, m03_3, m03_4, m03_5, m03_6, m03_7, m03_8, m03_9, m03_10, m03_11, m
03_12 = (
spy.loc['2003-1-1':'2003-2-1'], spy.loc['2003-2-1':'2003-3-1'], spy.loc['2003-3-
1':'2003-4-1'],
spy.loc['2003-4-1':'2003-5-1'], spy.loc['2003-5-1':'2003-6-1'], spy.loc['2003-6-
1':'2003-7-1'],
spy.loc['2003-7-1':'2003-8-1'], spy.loc['2003-8-1':'2003-9-1'], spy.loc['2003-9-
1':'2003-10-1'],
spy.loc['2003-10-1':'2003-11-1'], spy.loc['2003-11-1':'2003-12-1'], spy.loc['200
3-12-1':'2004-1-1'])

m04_1, m04_2, m04_3, m04_4, m04_5, m04_6, m04_7, m04_8, m04_9, m04_10, m04_11, m
04_12 = (
spy.loc['2004-1-1':'2004-2-1'], spy.loc['2004-2-1':'2004-3-1'], spy.loc['2004-3-
```

```
1':'2004-4-1'],

spy.loc['2004-4-1':'2004-5-1'], spy.loc['2004-5-1':'2004-6-1'], spy.loc['2004-6-
1':'2004-7-1'],
spy.loc['2004-7-1':'2004-8-1'], spy.loc['2004-8-1':'2004-9-1'], spy.loc['2004-9-
1':'2004-10-1'],
spy.loc['2004-10-1':'2004-11-1'], spy.loc['2004-11-1':'2004-12-1'], spy.loc['200
4-12-1':'2005-1-1'])

m05_1, m05_2, m05_3, m05_4, m05_5, m05_6, m05_7, m05_8, m05_9, m05_10, m05_11, m
05_12 = (
spy.loc['2005-1-1':'2005-2-1'], spy.loc['2005-2-1':'2005-3-1'], spy.loc['2005-3-
1':'2005-4-1'],
spy.loc['2005-4-1':'2005-5-1'], spy.loc['2005-5-1':'2005-6-1'], spy.loc['2005-6-
1':'2005-7-1'],
spy.loc['2005-7-1':'2005-8-1'], spy.loc['2005-8-1':'2005-9-1'], spy.loc['2005-9-
1':'2005-10-1'],
spy.loc['2005-10-1':'2005-11-1'], spy.loc['2005-11-1':'2005-12-1'], spy.loc['200
5-12-1':'2006-1-1'])

m06_1, m06_2, m06_3, m06_4, m06_5, m06_6, m06_7, m06_8, m06_9, m06_10, m06_11, m
06_12 = (
spy.loc['2006-1-1':'2006-2-1'], spy.loc['2006-2-1':'2006-3-1'], spy.loc['2006-3-
1':'2006-4-1'],
spy.loc['2006-4-1':'2006-5-1'], spy.loc['2006-5-1':'2006-6-1'], spy.loc['2006-6-
1':'2006-7-1'],
spy.loc['2006-7-1':'2006-8-1'], spy.loc['2006-8-1':'2006-9-1'], spy.loc['2006-9-
1':'2006-10-1'],
spy.loc['2006-10-1':'2006-11-1'], spy.loc['2006-11-1':'2006-12-1'], spy.loc['200
6-12-1':'2007-1-1'])

m07_1, m07_2, m07_3, m07_4, m07_5, m07_6, m07_7, m07_8, m07_9, m07_10, m07_11, m
07_12 = (
spy.loc['2007-1-1':'2007-2-1'], spy.loc['2007-2-1':'2007-3-1'], spy.loc['2007-3-
1':'2007-4-1'],
spy.loc['2007-4-1':'2007-5-1'], spy.loc['2007-5-1':'2007-6-1'], spy.loc['2007-6-
1':'2007-7-1'],
spy.loc['2007-7-1':'2007-8-1'], spy.loc['2007-8-1':'2007-9-1'], spy.loc['2007-9-
1':'2007-10-1'],
spy.loc['2007-10-1':'2007-11-1'], spy.loc['2007-11-1':'2007-12-1'], spy.loc['200
7-12-1':'2008-1-1'])

m08_1, m08_2, m08_3, m08_4, m08_5, m08_6, m08_7, m08_8, m08_9, m08_10, m08_11, m
08_12 = (
spy.loc['2008-1-1':'2008-2-1'], spy.loc['2008-2-1':'2008-3-1'], spy.loc['2008-3-
1':'2008-4-1'],
spy.loc['2008-4-1':'2008-5-1'], spy.loc['2008-5-1':'2008-6-1'], spy.loc['2008-6-
1':'2008-7-1'],
spy.loc['2008-7-1':'2008-8-1'], spy.loc['2008-8-1':'2008-9-1'], spy.loc['2008-9-
1':'2008-10-1'],
spy.loc['2008-10-1':'2008-11-1'], spy.loc['2008-11-1':'2008-12-1'], spy.loc['200
8-12-1':'2009-1-1'])

m09_1, m09_2, m09_3, m09_4, m09_5, m09_6, m09_7, m09_8, m09_9, m09_10, m09_11, m
```

```
09_12 = (

spy.loc['2009-1-1':'2009-2-1'], spy.loc['2009-2-1':'2009-3-1'], spy.loc['2009-3-
1':'2009-4-1'],
spy.loc['2009-4-1':'2009-5-1'], spy.loc['2009-5-1':'2009-6-1'], spy.loc['2009-6-
1':'2009-7-1'],
spy.loc['2009-7-1':'2009-8-1'], spy.loc['2009-8-1':'2009-9-1'], spy.loc['2009-9-
1':'2009-10-1'],
spy.loc['2009-10-1':'2009-11-1'], spy.loc['2009-11-1':'2009-12-1'], spy.loc['200
9-12-1':'2010-1-1'])

m10_1, m10_2, m10_3, m10_4, m10_5, m10_6, m10_7, m10_8, m10_9, m10_10, m10_11, m
10_12 = (
spy.loc['2010-1-1':'2010-2-1'], spy.loc['2010-2-1':'2010-3-1'], spy.loc['2010-3-
1':'2010-4-1'],
spy.loc['2010-4-1':'2010-5-1'], spy.loc['2010-5-1':'2010-6-1'], spy.loc['2010-6-
1':'2010-7-1'],
spy.loc['2010-7-1':'2010-8-1'], spy.loc['2010-8-1':'2010-9-1'], spy.loc['2010-9-
1':'2010-10-1'],
spy.loc['2010-10-1':'2010-11-1'], spy.loc['2010-11-1':'2010-12-1'], spy.loc['201
0-12-1':'2011-1-1'])

m11_1, m11_2, m11_3, m11_4, m11_5, m11_6, m11_7, m11_8, m11_9, m11_10, m11_11, m
11_12 = (
spy.loc['2011-1-1':'2011-2-1'], spy.loc['2011-2-1':'2011-3-1'], spy.loc['2011-3-
1':'2011-4-1'],
spy.loc['2011-4-1':'2011-5-1'], spy.loc['2011-5-1':'2011-6-1'], spy.loc['2011-6-
1':'2011-7-1'],
spy.loc['2011-7-1':'2011-8-1'], spy.loc['2011-8-1':'2011-9-1'], spy.loc['2011-9-
1':'2011-10-1'],
spy.loc['2011-10-1':'2011-11-1'], spy.loc['2011-11-1':'2011-12-1'], spy.loc['201
1-12-1':'2012-1-1'])

m12_1, m12_2, m12_3, m12_4, m12_5, m12_6, m12_7, m12_8, m12_9, m12_10, m12_11, m
12_12 = (
spy.loc['2012-1-1':'2012-2-1'], spy.loc['2012-2-1':'2012-3-1'], spy.loc['2012-3-
1':'2012-4-1'],
spy.loc['2012-4-1':'2012-5-1'], spy.loc['2012-5-1':'2012-6-1'], spy.loc['2012-6-
1':'2012-7-1'],
spy.loc['2012-7-1':'2012-8-1'], spy.loc['2012-8-1':'2012-9-1'], spy.loc['2012-9-
1':'2012-10-1'],
spy.loc['2012-10-1':'2012-11-1'], spy.loc['2012-11-1':'2012-12-1'], spy.loc['201
2-12-1':'2013-1-1'])

m13_1, m13_2, m13_3, m13_4, m13_5, m13_6, m13_7, m13_8, m13_9, m13_10, m13_11, m
13_12 = (
spy.loc['2013-1-1':'2013-2-1'], spy.loc['2013-2-1':'2013-3-1'], spy.loc['2013-3-
1':'2013-4-1'],
spy.loc['2013-4-1':'2013-5-1'], spy.loc['2013-5-1':'2013-6-1'], spy.loc['2013-6-
1':'2013-7-1'],
spy.loc['2013-7-1':'2013-8-1'], spy.loc['2013-8-1':'2013-9-1'], spy.loc['2013-9-
1':'2013-10-1'],
spy.loc['2013-10-1':'2013-11-1'], spy.loc['2013-11-1':'2013-12-1'], spy.loc['201
3-12-1':'2014-1-1'])
```

```
m14_1, m14_2, m14_3, m14_4, m14_5, m14_6, m14_7, m14_8, m14_9, m14_10, m14_11, m
14_12 = (
spy.loc['2014-1-1':'2014-2-1'], spy.loc['2014-2-1':'2014-3-1'], spy.loc['2014-3-
1':'2014-4-1'],
spy.loc['2014-4-1':'2014-5-1'], spy.loc['2014-5-1':'2014-6-1'], spy.loc['2014-6-
1':'2014-7-1'],
spy.loc['2014-7-1':'2014-8-1'], spy.loc['2014-8-1':'2014-9-1'], spy.loc['2014-9-
1':'2014-10-1'],
spy.loc['2014-10-1':'2014-11-1'], spy.loc['2014-11-1':'2014-12-1'], spy.loc['201
4-12-1':'2015-1-1'])

m15_1, m15_2, m15_3, m15_4, m15_5, m15_6, m15_7, m15_8, m15_9, m15_10, m15_11, m
15_12 = (
spy.loc['2015-1-1':'2015-2-1'], spy.loc['2015-2-1':'2015-3-1'], spy.loc['2015-3-
1':'2015-4-1'],
spy.loc['2015-4-1':'2015-5-1'], spy.loc['2015-5-1':'2015-6-1'], spy.loc['2015-6-
1':'2015-7-1'],
spy.loc['2015-7-1':'2015-8-1'], spy.loc['2015-8-1':'2015-9-1'], spy.loc['2015-9-
1':'2015-10-1'],
spy.loc['2015-10-1':'2015-11-1'], spy.loc['2015-11-1':'2015-12-1'], spy.loc['201
5-12-1':'2016-1-1'])

m16_1, m16_2, m16_3, m16_4, m16_5, m16_6, m16_7, m16_8, m16_9, m16_10, m16_11, m
16_12 = (
spy.loc['2016-1-1':'2016-2-1'], spy.loc['2016-2-1':'2016-3-1'], spy.loc['2016-3-
1':'2016-4-1'],
spy.loc['2016-4-1':'2016-5-1'], spy.loc['2016-5-1':'2016-6-1'], spy.loc['2016-6-
1':'2016-7-1'],
spy.loc['2016-7-1':'2016-8-1'], spy.loc['2016-8-1':'2016-9-1'], spy.loc['2016-9-
1':'2016-10-1'],
spy.loc['2016-10-1':'2016-11-1'], spy.loc['2016-11-1':'2016-12-1'], spy.loc['201
6-12-1':'2017-1-1'])
```

In [28]:

```
# Skew
s02_1, s02_2, s02_3, s02_4, s02_5, s02_6, s02_7, s02_8, s02_9, s02_10, s02_11, s
02_12 = (
stats.skew(m02_1), stats.skew(m02_2), stats.skew(m02_3), stats.skew(m02_4), stat
s.skew(m02_5),
stats.skew(m02_6), stats.skew(m02_7), stats.skew(m02_8), stats.skew(m02_9), stat
s.skew(m02_10),
stats.skew(m02_11), stats.skew(m02_12))

s03_1, s03_2, s03_3, s03_4, s03_5, s03_6, s03_7, s03_8, s03_9, s03_10, s03_11, s
03_12 = (
stats.skew(m03_1), stats.skew(m03_2), stats.skew(m03_3), stats.skew(m03_4), stat
s.skew(m03_5),
stats.skew(m03_6), stats.skew(m03_7), stats.skew(m03_8), stats.skew(m03_9), stat
s.skew(m03_10),
stats.skew(m03_11), stats.skew(m03_12))
```

```
s04_1, s04_2, s04_3, s04_4, s04_5, s04_6, s04_7, s04_8, s04_9, s04_10, s04_11, s
04_12 = (
stats.skew(m04_1), stats.skew(m04_2), stats.skew(m04_3), stats.skew(m04_4), stat
s.skew(m04_5),
stats.skew(m04_6), stats.skew(m04_7), stats.skew(m04_8), stats.skew(m04_9), stat
s.skew(m04_10),
stats.skew(m04_11), stats.skew(m04_12))

s05_1, s05_2, s05_3, s05_4, s05_5, s05_6, s05_7, s05_8, s05_9, s05_10, s05_11, s
05_12 = (
stats.skew(m05_1), stats.skew(m05_2), stats.skew(m05_3), stats.skew(m05_4), stat
s.skew(m05_5),
stats.skew(m05_6), stats.skew(m05_7), stats.skew(m05_8), stats.skew(m05_9), stat
s.skew(m05_10),
stats.skew(m05_11), stats.skew(m05_12))

s06_1, s06_2, s06_3, s06_4, s06_5, s06_6, s06_7, s06_8, s06_9, s06_10, s06_11, s
06_12 = (
stats.skew(m06_1), stats.skew(m06_2), stats.skew(m06_3), stats.skew(m06_4), stat
s.skew(m06_5),
stats.skew(m06_6), stats.skew(m06_7), stats.skew(m06_8), stats.skew(m06_9), stat
s.skew(m06_10),
stats.skew(m06_11), stats.skew(m06_12))

s07_1, s07_2, s07_3, s07_4, s07_5, s07_6, s07_7, s07_8, s07_9, s07_10, s07_11, s
07_12 = (
stats.skew(m07_1), stats.skew(m07_2), stats.skew(m07_3), stats.skew(m07_4), stat
s.skew(m07_5),
stats.skew(m07_6), stats.skew(m07_7), stats.skew(m07_8), stats.skew(m07_9), stat
s.skew(m07_10),
stats.skew(m07_11), stats.skew(m07_12))

s08_1, s08_2, s08_3, s08_4, s08_5, s08_6, s08_7, s08_8, s08_9, s08_10, s08_11, s
08_12 = (
stats.skew(m08_1), stats.skew(m08_2), stats.skew(m08_3), stats.skew(m08_4), stat
s.skew(m08_5),
stats.skew(m08_6), stats.skew(m08_7), stats.skew(m08_8), stats.skew(m08_9), stat
s.skew(m08_10),
stats.skew(m08_11), stats.skew(m08_12))

s09_1, s09_2, s09_3, s09_4, s09_5, s09_6, s09_7, s09_8, s09_9, s09_10, s09_11, s
09_12 = (
stats.skew(m09_1), stats.skew(m09_2), stats.skew(m09_3), stats.skew(m09_4), stat
s.skew(m09_5),
stats.skew(m09_6), stats.skew(m09_7), stats.skew(m09_8), stats.skew(m09_9), stat
s.skew(m09_10),
stats.skew(m09_11), stats.skew(m09_12))

s10_1, s10_2, s10_3, s10_4, s10_5, s10_6, s10_7, s10_8, s10_9, s10_10, s10_11, s
10_12 = (
stats.skew(m10_1), stats.skew(m10_2), stats.skew(m10_3), stats.skew(m10_4), stat
s.skew(m10_5),
stats.skew(m10_6), stats.skew(m10_7), stats.skew(m10_8), stats.skew(m10_9), stat
```

```
s.skew(m10_10),

stats.skew(m10_11), stats.skew(m10_12))

s11_1, s11_2, s11_3, s11_4, s11_5, s11_6, s11_7, s11_8, s11_9, s11_10, s11_11, s
11_12 = (
stats.skew(m11_1), stats.skew(m11_2), stats.skew(m11_3), stats.skew(m11_4), stat
s.skew(m11_5),
stats.skew(m11_6), stats.skew(m11_7), stats.skew(m11_8), stats.skew(m11_9), stat
s.skew(m11_10),
stats.skew(m11_11), stats.skew(m11_12))

s12_1, s12_2, s12_3, s12_4, s12_5, s12_6, s12_7, s12_8, s12_9, s12_10, s12_11, s
12_12 = (
stats.skew(m12_1), stats.skew(m12_2), stats.skew(m12_3), stats.skew(m12_4), stat
s.skew(m12_5),
stats.skew(m12_6), stats.skew(m12_7), stats.skew(m12_8), stats.skew(m12_9), stat
s.skew(m12_10),
stats.skew(m12_11), stats.skew(m12_12))

s13_1, s13_2, s13_3, s13_4, s13_5, s13_6, s13_7, s13_8, s13_9, s13_10, s13_11, s
13_12 = (
stats.skew(m13_1), stats.skew(m13_2), stats.skew(m13_3), stats.skew(m13_4), stat
s.skew(m13_5),
stats.skew(m13_6), stats.skew(m13_7), stats.skew(m13_8), stats.skew(m13_9), stat
s.skew(m13_10),
stats.skew(m13_11), stats.skew(m13_12))

s14_1, s14_2, s14_3, s14_4, s14_5, s14_6, s14_7, s14_8, s14_9, s14_10, s14_11, s
14_12 = (
stats.skew(m14_1), stats.skew(m14_2), stats.skew(m14_3), stats.skew(m14_4), stat
s.skew(m14_5),
stats.skew(m14_6), stats.skew(m14_7), stats.skew(m14_8), stats.skew(m14_9), stat
s.skew(m14_10),
stats.skew(m14_11), stats.skew(m14_12))

s15_1, s15_2, s15_3, s15_4, s15_5, s15_6, s15_7, s15_8, s15_9, s15_10, s15_11, s
15_12 = (
stats.skew(m15_1), stats.skew(m15_2), stats.skew(m15_3), stats.skew(m15_4), stat
s.skew(m15_5),
stats.skew(m15_6), stats.skew(m15_7), stats.skew(m15_8), stats.skew(m15_9), stat
s.skew(m15_10),
stats.skew(m15_11), stats.skew(m15_12))

s16_1, s16_2, s16_3, s16_4, s16_5, s16_6, s16_7, s16_8, s16_9, s16_10, s16_11, s
16_12 = (
stats.skew(m16_1), stats.skew(m16_2), stats.skew(m16_3), stats.skew(m16_4), stat
s.skew(m16_5),
stats.skew(m16_6), stats.skew(m16_7), stats.skew(m16_8), stats.skew(m16_9), stat
s.skew(m16_10),
stats.skew(m16_11), stats.skew(m16_12))

s02, s03, s04, s05, s06, s07, s08, s09, s10, s11, s12, s13, s14, s15, s16 = (
    [s02_1, s02_2, s02_3, s02_4, s02_5, s02_6, s02_7, s02_8, s02_9, s02_10, s02_
```

```
11, s02_12],

    [s03_1, s03_2, s03_3, s03_4, s03_5, s03_6, s03_7, s03_8, s03_9, s03_10, s03_
11, s03_12],
    [s04_1, s04_2, s04_3, s04_4, s04_5, s04_6, s04_7, s04_8, s04_9, s04_10, s04_
11, s04_12],
    [s05_1, s05_2, s05_3, s05_4, s05_5, s05_6, s05_7, s05_8, s05_9, s05_10, s05_
11, s05_12],
    [s06_1, s06_2, s06_3, s06_4, s06_5, s06_6, s06_7, s06_8, s06_9, s06_10, s06_
11, s06_12],
    [s07_1, s07_2, s07_3, s07_4, s07_5, s07_6, s07_7, s07_8, s07_9, s07_10, s07_
11, s07_12],
    [s08_1, s08_2, s08_3, s08_4, s08_5, s08_6, s08_7, s08_8, s08_9, s08_10, s08_
11, s08_12],
    [s09_1, s09_2, s09_3, s09_4, s09_5, s09_6, s09_7, s09_8, s09_9, s09_10, s09_
11, s09_12],
    [s10_1, s10_2, s10_3, s10_4, s10_5, s10_6, s10_7, s10_8, s10_9, s10_10, s10_
11, s10_12],
    [s11_1, s11_2, s11_3, s11_4, s11_5, s11_6, s11_7, s11_8, s11_9, s11_10, s11_
11, s11_12],
    [s12_1, s12_2, s12_3, s12_4, s12_5, s12_6, s12_7, s12_8, s12_9, s12_10, s12_
11, s12_12],
    [s13_1, s13_2, s13_3, s13_4, s13_5, s13_6, s13_7, s13_8, s13_9, s13_10, s13_
11, s13_12],
    [s14_1, s14_2, s14_3, s14_4, s14_5, s14_6, s14_7, s14_8, s14_9, s14_10, s14_
11, s14_12],
    [s15_1, s15_2, s15_3, s15_4, s15_5, s15_6, s15_7, s15_8, s15_9, s15_10, s15_
11, s15_12],
    [s16_1, s16_2, s16_3, s16_4, s16_5, s16_6, s16_7, s16_8, s16_9, s16_10, s16_
11, s16_12])

totals = []
totals.extend([s02, s03, s04, s05, s06, s07, s08, s09, s10, s11, s12, s13, s14,
s15, s16])
flattened_totals = [item for sublist in totals for item in sublist]
```

In [29]:

```
# Means

mean02_1, mean02_2, mean02_3, mean02_4, mean02_5, mean02_6, mean02_7, mean02_8,
mean02_9, mean02_10, mean02_11, mean02_12 = (m02_1.mean(), m02_2.mean(), m02_3.m
ean(), m02_4.mean(), m02_5.mean(),
m02_6.mean(), m02_7.mean(), m02_8.mean(), m02_9.mean(), m02_10.mean(),
m02_11.mean(), m02_12.mean())

mean03_1, mean03_2, mean03_3, mean03_4, mean03_5, mean03_6, mean03_7, mean03_8,
mean03_9, mean03_10, mean03_11, mean03_12 = (
m03_1.mean(), m03_2.mean(), m03_3.mean(), m03_4.mean(), m03_5.mean(),
m03_6.mean(), m03_7.mean(), m03_8.mean(), m03_9.mean(), m03_10.mean(),
m03_11.mean(), m03_12.mean())

mean04_1, mean04_2, mean04_3, mean04_4, mean04_5, mean04_6, mean04_7, mean04_8,
mean04_9, mean04_10, mean04_11, mean04_12 = (
```

```
m04_1.mean(), m04_2.mean(), m04_3.mean(), m04_4.mean(), m04_5.mean(),

m04_6.mean(), m04_7.mean(), m04_8.mean(), m04_9.mean(), m04_10.mean(),
m04_11.mean(), m04_12.mean())

mean05_1, mean05_2, mean05_3, mean05_4, mean05_5, mean05_6, mean05_7, mean05_8,
mean05_9, mean05_10, mean05_11, mean05_12 = (
m05_1.mean(), m05_2.mean(), m05_3.mean(), m05_4.mean(), m05_5.mean(),
m05_6.mean(), m05_7.mean(), m05_8.mean(), m05_9.mean(), m05_10.mean(),
m05_11.mean(), m05_12.mean())

mean06_1, mean06_2, mean06_3, mean06_4, mean06_5, mean06_6, mean06_7, mean06_8,
mean06_9, mean06_10, mean06_11, mean06_12 = (
m06_1.mean(), m06_2.mean(), m06_3.mean(), m06_4.mean(), m06_5.mean(),
m06_6.mean(), m06_7.mean(), m06_8.mean(), m06_9.mean(), m06_10.mean(),
m06_11.mean(), m06_12.mean())

mean07_1, mean07_2, mean07_3, mean07_4, mean07_5, mean07_6, mean07_7, mean07_8,
mean07_9, mean07_10, mean07_11, mean07_12 = (
m07_1.mean(), m07_2.mean(), m07_3.mean(), m07_4.mean(), m07_5.mean(),
m07_6.mean(), m07_7.mean(), m07_8.mean(), m07_9.mean(), m07_10.mean(),
m07_11.mean(), m07_12.mean())

mean08_1, mean08_2, mean08_3, mean08_4, mean08_5, mean08_6, mean08_7, mean08_8,
mean08_9, mean08_10, mean08_11, mean08_12 = (
m08_1.mean(), m08_2.mean(), m08_3.mean(), m08_4.mean(), m08_5.mean(),
m08_6.mean(), m08_7.mean(), m08_8.mean(), m08_9.mean(), m08_10.mean(),
m08_11.mean(), m08_12.mean())

mean09_1, mean09_2, mean09_3, mean09_4, mean09_5, mean09_6, mean09_7, mean09_8,
mean09_9, mean09_10, mean09_11, mean09_12 =(
m09_1.mean(), m09_2.mean(), m09_3.mean(), m09_4.mean(), m09_5.mean(),
m09_6.mean(), m09_7.mean(), m09_8.mean(), m09_9.mean(), m09_10.mean(),
m09_11.mean(), m09_12.mean())

mean10_1, mean10_2, mean10_3, mean10_4, mean10_5, mean10_6, mean10_7, mean10_8,
mean10_9, mean10_10, mean10_11, mean10_12 = (
m10_1.mean(), m10_2.mean(), m10_3.mean(), m10_4.mean(), m10_5.mean(),
m10_6.mean(), m10_7.mean(), m10_8.mean(), m10_9.mean(), m10_10.mean(),
m10_11.mean(), m10_12.mean())

mean11_1, mean11_2, mean11_3, mean11_4, mean11_5, mean11_6, mean11_7, mean11_8,
mean11_9, mean11_10, mean11_11, mean11_12 = (
m11_1.mean(), m11_2.mean(), m11_3.mean(), m11_4.mean(), m11_5.mean(),
m11_6.mean(), m11_7.mean(), m11_8.mean(), m11_9.mean(), m11_10.mean(),
m11_11.mean(), m11_12.mean())

mean12_1, mean12_2, mean12_3, mean12_4, mean12_5, mean12_6, mean12_7, mean12_8,
mean12_9, mean12_10, mean12_11, mean12_12 = (
m12_1.mean(), m12_2.mean(), m12_3.mean(), m12_4.mean(), m12_5.mean(),
m12_6.mean(), m12_7.mean(), m12_8.mean(), m12_9.mean(), m12_10.mean(),
m12_11.mean(), m12_12.mean())

mean13_1, mean13_2, mean13_3, mean13_4, mean13_5, mean13_6, mean13_7, mean13_8,
```

```
mean13_9, mean13_10, mean13_11, mean13_12 = (

m13_1.mean(), m13_2.mean(), m13_3.mean(), m13_4.mean(), m13_5.mean(),
m13_6.mean(), m13_7.mean(), m13_8.mean(), m13_9.mean(), m13_10.mean(),
m13_11.mean(), m13_12.mean())

mean14_1, mean14_2, mean14_3, mean14_4, mean14_5, mean14_6, mean14_7, mean14_8,
mean14_9, mean14_10, mean14_11, mean14_12 = (
m14_1.mean(), m14_2.mean(), m14_3.mean(), m14_4.mean(), m14_5.mean(),
m14_6.mean(), m14_7.mean(), m14_8.mean(), m14_9.mean(), m14_10.mean(),
m14_11.mean(), m14_12.mean())

mean15_1, mean15_2, mean15_3, mean15_4, mean15_5, mean15_6, mean15_7, mean15_8,
mean15_9, mean15_10, mean15_11, mean15_12 = (
m15_1.mean(), m15_2.mean(), m15_3.mean(), m15_4.mean(), m15_5.mean(),
m15_6.mean(), m15_7.mean(), m15_8.mean(), m15_9.mean(), m15_10.mean(),
m15_11.mean(), m15_12.mean())

mean16_1, mean16_2, mean16_3, mean16_4, mean16_5, mean16_6, mean16_7, mean16_8,
mean16_9, mean16_10, mean16_11, mean16_12 = (
m16_1.mean(), m16_2.mean(), m16_3.mean(), m16_4.mean(), m16_5.mean(),
m16_6.mean(), m16_7.mean(), m16_8.mean(), m16_9.mean(), m16_10.mean(),
m16_11.mean(), m16_12.mean())

mean02, mean03, mean04, mean05, mean06, mean07, mean08, mean09, mean10, mean11,
mean12, mean13, mean14, mean15, mean16 = (
    [mean02_1, mean02_2, mean02_3, mean02_4, mean02_5, mean02_6, mean02_7, mean0
2_8, mean02_9, mean02_10, mean02_11, mean02_12],
    [mean03_1, mean03_2, mean03_3, mean03_4, mean03_5, mean03_6, mean03_7, mean0
3_8, mean03_9, mean03_10, mean03_11, mean03_12],
    [mean04_1, mean04_2, mean04_3, mean04_4, mean04_5, mean04_6, mean04_7, mean0
4_8, mean04_9, mean04_10, mean04_11, mean04_12],
    [mean05_1, mean05_2, mean05_3, mean05_4, mean05_5, mean05_6, mean05_7, mean0
5_8, mean05_9, mean05_10, mean05_11, mean05_12],
    [mean06_1, mean06_2, mean06_3, mean06_4, mean06_5, mean06_6, mean06_7, mean0
6_8, mean06_9, mean06_10, mean06_11, mean06_12],
    [mean07_1, mean07_2, mean07_3, mean07_4, mean07_5, mean07_6, mean07_7, mean0
7_8, mean07_9, mean07_10, mean07_11, mean07_12],
    [mean08_1, mean08_2, mean08_3, mean08_4, mean08_5, mean08_6, mean08_7, mean0
8_8, mean08_9, mean08_10, mean08_11, mean08_12],
    [mean09_1, mean09_2, mean09_3, mean09_4, mean09_5, mean09_6, mean09_7, mean0
9_8, mean09_9, mean09_10, mean09_11, mean09_12],
    [mean10_1, mean10_2, mean10_3, mean10_4, mean10_5, mean10_6, mean10_7, mean1
0_8, mean10_9, mean10_10, mean10_11, mean10_12],
    [mean11_1, mean11_2, mean11_3, mean11_4, mean11_5, mean11_6, mean11_7, mean1
1_8, mean11_9, mean11_10, mean11_11, mean11_12],
    [mean12_1, mean12_2, mean12_3, mean12_4, mean12_5, mean12_6, mean12_7, mean1
2_8, mean12_9, mean12_10, mean12_11, mean12_12],
    [mean13_1, mean13_2, mean13_3, mean13_4, mean13_5, mean13_6, mean13_7, mean1
3_8, mean13_9, mean13_10, mean13_11, mean13_12],
    [mean14_1, mean14_2, mean14_3, mean14_4, mean14_5, mean14_6, mean14_7, mean1
4_8, mean14_9, mean14_10, mean14_11, mean14_12],
    [mean15_1, mean15_2, mean15_3, mean15_4, mean15_5, mean15_6, mean15_7, mean1
5_8, mean15_9, mean15_10, mean15_11, mean15_12],
```

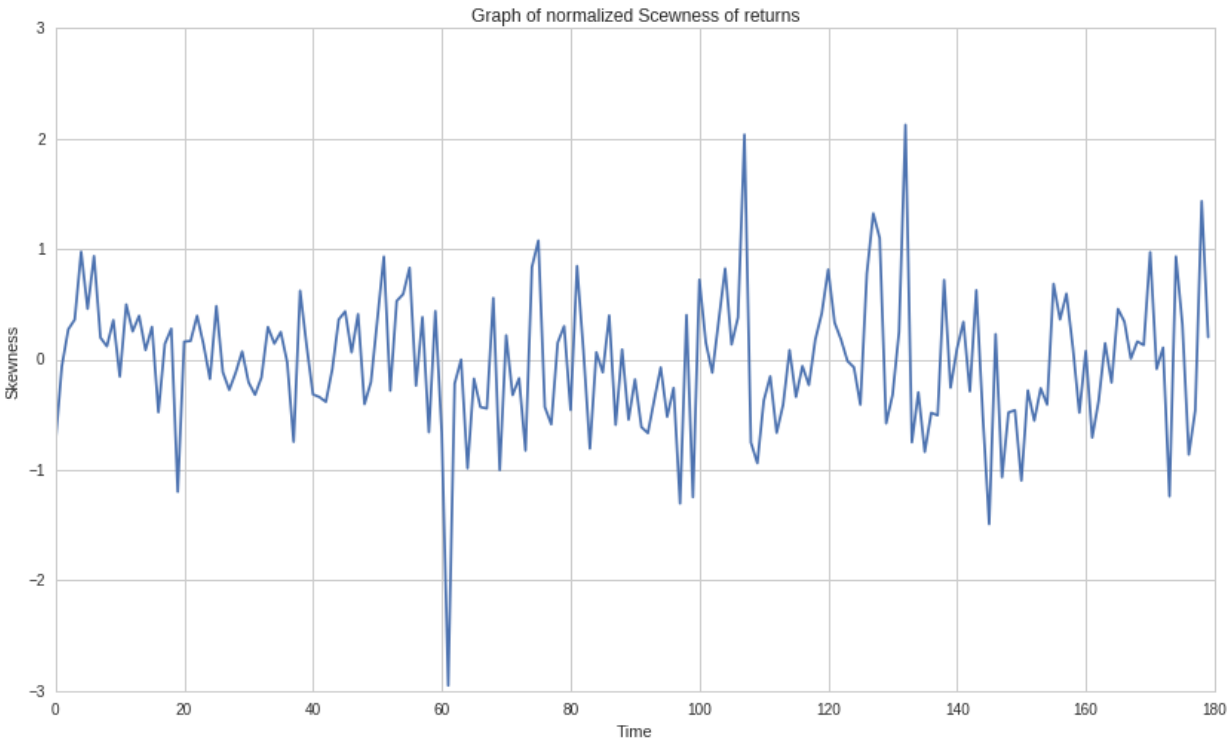
```
    [mean16_1, mean16_2, mean16_3, mean16_4, mean16_5, mean16_6, mean16_7, mean1
6_8, mean16_9, mean16_10, mean16_11, mean16_12])

totalm = []
totalm.extend([mean02, mean03, mean04, mean05, mean06, mean07, mean08, mean09, m
ean10, mean11, mean12, mean13, mean14, mean15, mean16])
flattened_totalm = [item for sublist in totalm for item in sublist]
```

In [6]:

```
# Normalization of Means and Skew

smini = min(flattened_totals)
smaxi = max(flattened_totals)

mmini = min(flattened_totalm)
mmaxi = max(flattened_totalm)

# Convert List to PD DataFrame in order to index with time.
df_flattened_totals = pd.DataFrame(flattened_totals)
df_flattened_totalm = pd.DataFrame(flattened_totalm)

# New Index
new_index = pd.date_range(start, periods=len(df_flattened_totals), freq="M")
df_flattened_totals.index = new_index
df_flattened_totalm.index = new_index

# DataFrame normals
df_norm_s = ( (df_flattened_totals - df_flattened_totals.mean()) /
(df_flattened_totals.max() - df_flattened_totals.min()) )

df_norm_m = ( (df_flattened_totalm - df_flattened_totalm.mean()) /
             (df_flattened_totalm.max() - df_flattened_totalm.min()) )

pd_serie_s = pd.Series(flattened_totals)
pd_serie_m = pd.Series(flattened_totalm)

S_norm_s = ( (pd_serie_s - pd_serie_s.mean()) /
(pd_serie_s.max() - pd_serie_s.min()) )

S_norm_m = ( (pd_serie_m - pd_serie_m.mean()) /
             (pd_serie_m.max() - pd_serie_m.min()) )

S_norm_s.index = new_index
S_norm_m.index = new_index

df_norm_s.name = 'Normalized Skewness'
df_norm_m.name = 'Normalized Mean'
```

In [14]:

```
plt.plot(pd_serie_s)
plt.title('Graph of normalized Scewness of returns')
plt.xlabel('Time')
plt.ylabel('Skewness')
plt.show();
```



In [8]:

```
negative_skewing = pd_serie_s.loc[pd_serie_s < 0]
positive_skewing = pd_serie_s.loc[pd_serie_s > 0]

print 'Number of negativly skewed:', len(negative_skewing)
print 'Number of positivly skewed:', len(positive_skewing)
percentage_diff = 100 - (len(positive_skewing)*100 / len(negative_skewing))
print 'There are about',percentage_diff,'% more Negativly skewed returns in the
time between 2002 and 2017'
```

```
Number of negativly skewed: 91
Number of positivly skewed: 89
There are about 3 % more Negativly skewed returns in the time betwee
n 2002 and 2017
```

We can find a natural tendency in the SPY ETF of having more **Negativly skewed** returns than postive once. This means that at any given time, it is more likely that we will have a higher probability of having higher returns over a probability of having lower returns.
Next we will try and find what the average return was.

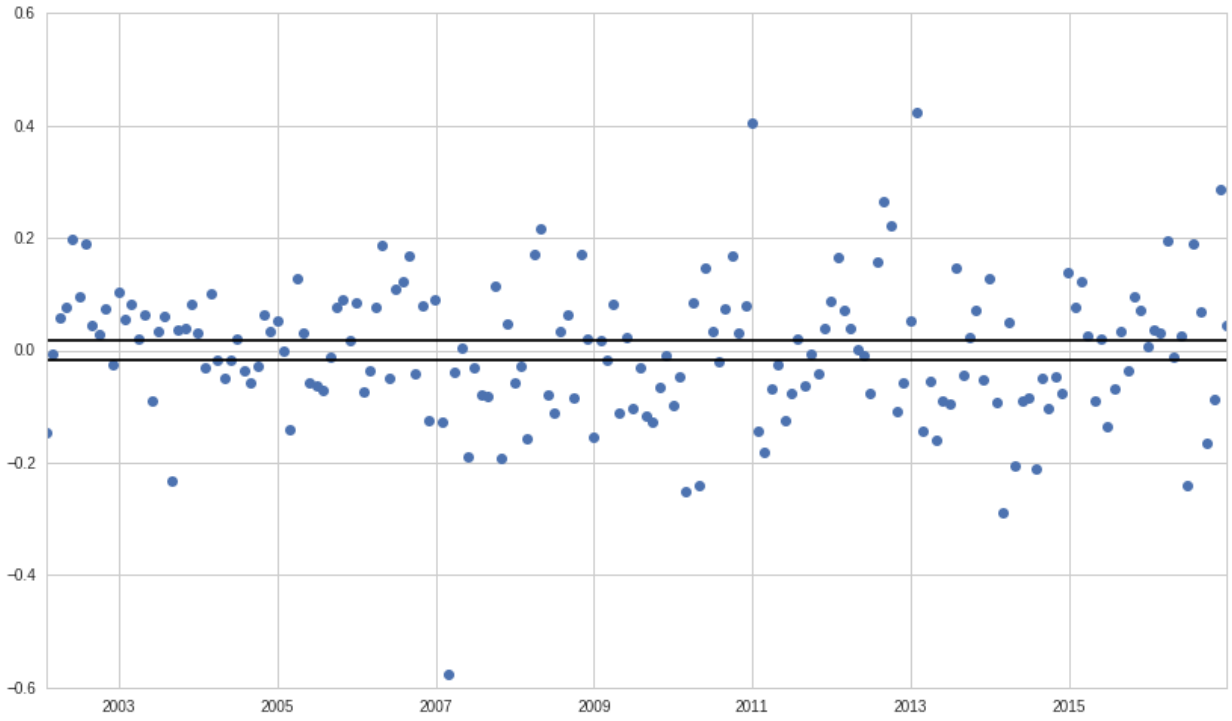
In [9]:

```
confidence = .95
a = np.array(S_norm_s)
n = len(a)
m = np.mean(a)
se = stats.sem(a)
h = se * sp.stats.t._ppf((1 + confidence) / 2., n-1)
```

In [15]:

```
#Plotting using confidence levels
plt.plot(S_norm_s, 'o');
plt.axhline(y = m-h, color='k');
plt.axhline(y = m+h, color='k');

print 'Variance =', pd.Series.var(S_norm_s)
```

```
Variance = 0.0150528904734
```



Skewness above one is considered to be significant.

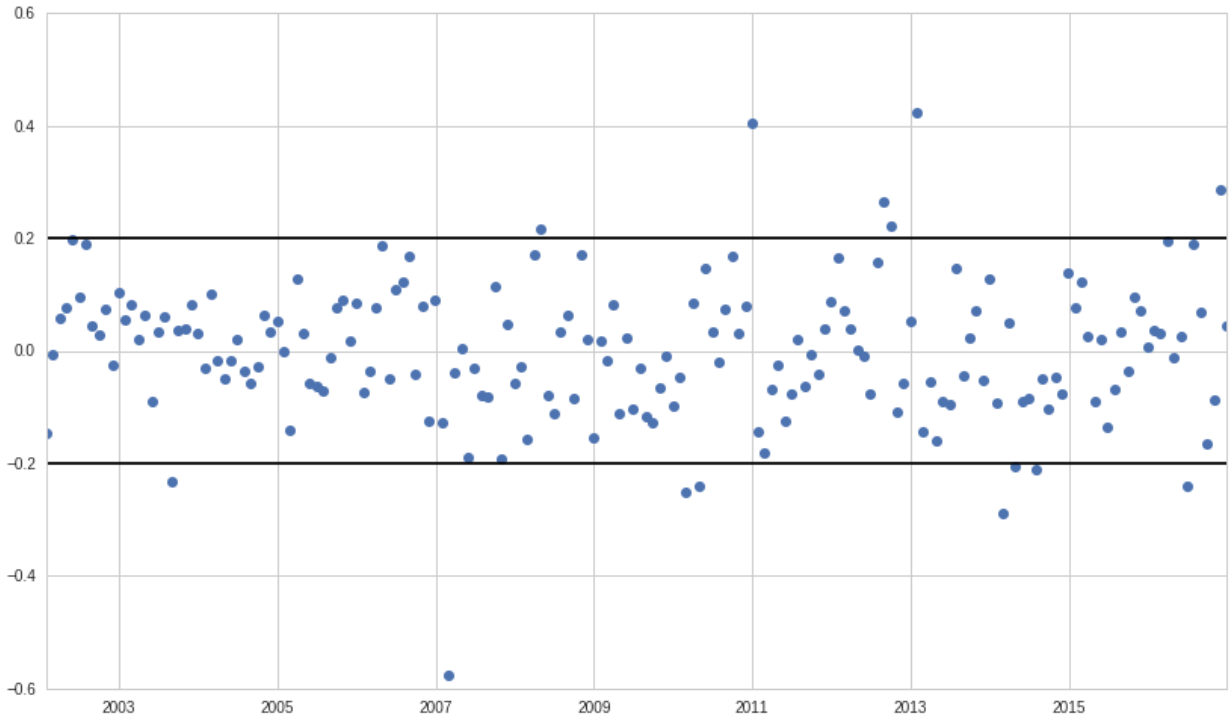
In [11]:

```
# Noramlizing one.
normalone = ( (1 - pd_serie_s.mean()) / (pd_serie_s.max() - pd_serie_s.min()) )
print normalone
```

```
0.201666280188
```

In [12]:

```
# Plotting using +/- 0.15 level.
plt.plot(S_norm_s, 'o');
plt.axhline(y = 0.2, color='k');
plt.axhline(y = -0.2, color='k');
```



In [14]:

```
above = S_norm_s.loc[S_norm_s > 0.2]
print above
```

```
2008-04-30    0.216920
2010-12-31    0.405843
2012-08-31    0.265348
2012-09-30    0.221328
2013-01-31    0.423309
2016-11-30    0.287109
dtype: float64
```

In [15]:

```
bellow = S_norm_s.loc[S_norm_s < -0.2]
print bellow
```

```
2003-08-31   -0.231045
2007-02-28   -0.576691
2010-02-28   -0.251899
2010-04-30   -0.240572
2014-02-28   -0.288777
2014-04-30   -0.205193
2014-07-31   -0.211200
2016-06-30   -0.239086
dtype: float64
```

In [16]:

```
total_outliers = len(above) + len(bellow)
total_numbers = len(S_norm_s)
print 'percentage of outliers in the data:', total_outliers*100 / float(total_nu
mbers), '%'
```

```
percentage of outliers in the data: 7.77777777778 %
```

We can find that from 2002 to 2017, there is only about 8% of outliers.
We will next try to find a regression model to fit the data.

For instances of negative Skewing:

- We first look for the moment when there was maximum negative skewing.
- Next we plot a histogram of the returns to determine the most likely value of returns, when it occured and how many times that value was repeated.

In [17]:

```
# Let's look for moments where the skewness is negative.

# Converting negative_skew into a DataFrame will allow us to use indexes to find
# Dates and times associated with skewness levels.

df_flattened_totals = pd.DataFrame(flattened_totals)
df_flattened_totals.columns = ['Price']
df_flattened_totals.index = pd.date_range(start, periods=len(df_flattened_totals
), freq="M")
print pd.DataFrame.min(df_flattened_totals)
```

```
Price   -2.952178
dtype: float64
```

In [18]:

```
# Use < mini because numbers are rounded and exact values cannot be found.
df_flattened_totals[df_flattened_totals['Price'] < -2.95217].index.tolist()
```

Out[18]:

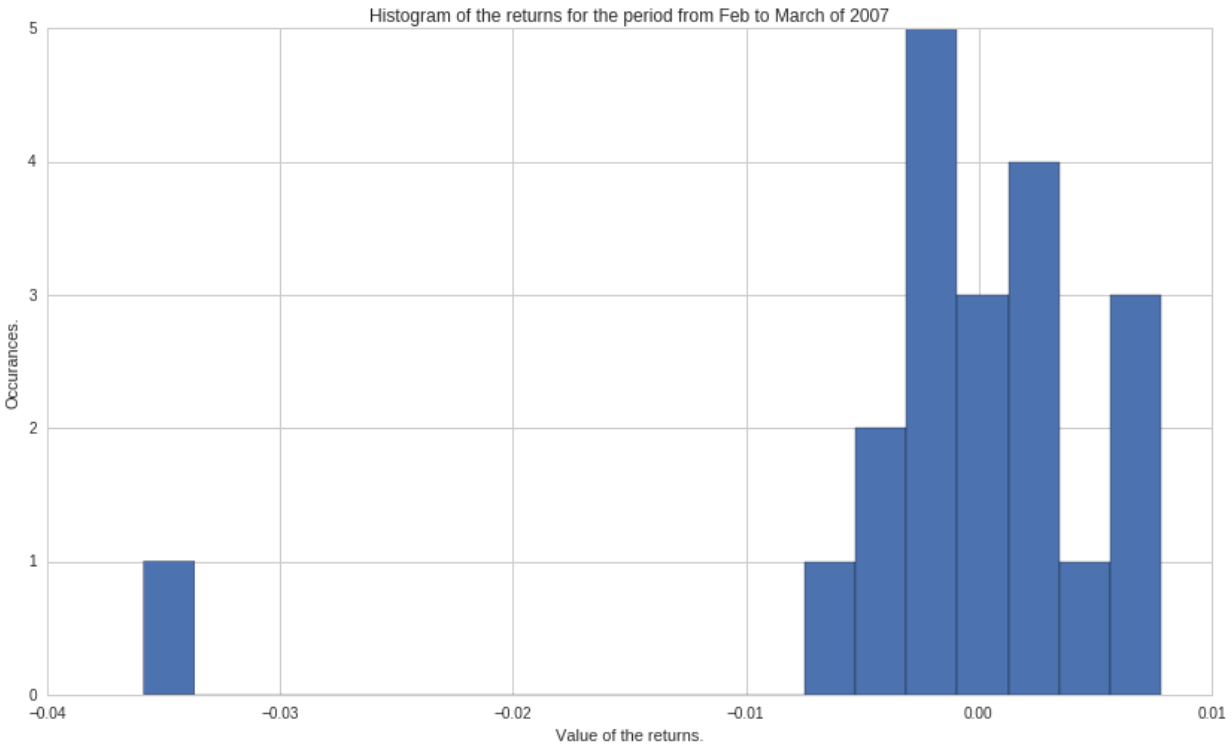
```
[Timestamp('2007-02-28 00:00:00', offset='M')]
```

We find the maximum negative skew value on the $28^{th} of February 2007$. We will then plot the histogram of the skewness for the returns of February, and see the returns with highest probability.

In [19]:

```
returns = spy['2007-02-01':'2007-03-01']

hist = plt.hist(returns, bins=len(returns));
plt.xlabel('Value of the returns.')
plt.ylabel('Occurances.')
plt.title('Histogram of the returns for the period from Feb to March of 2007')

y, x, _ = hist

print 'Maximum value of returns is found on :', np.argmax(returns)
print 'Return with the most frequency was:', x.max(), 'with a occurance of :', y
.max(), 'times in the month'
```

```
Maximum value of returns is found on : 2007-02-14 00:00:00+00:00
Return with the most frequency was: 0.00781792029783 with a occuranc
e of : 5.0 times in the month
```



Doing the same with Positive Skewness this time:

- We first look for the moment when there was maximum Postive skewing.
- Next we plot a histogram of the returns to determine the most likely value of returns, when it occured and how many times that value was repeated.

In [20]:

```
# Doing the same thing for when the returns are positivly skewed.
print  'Maximum value is found as:', pd.DataFrame.max(df_flattened_totals)
```

```
Maximum value is found as: Price    2.125411
dtype: float64
```

In [21]:

```
# Use > mini because numbers are rounded and exact values cannot be found.
df_flattened_totals[df_flattened_totals['Price'] > 2.12541].index.tolist()
```

Out[21]:

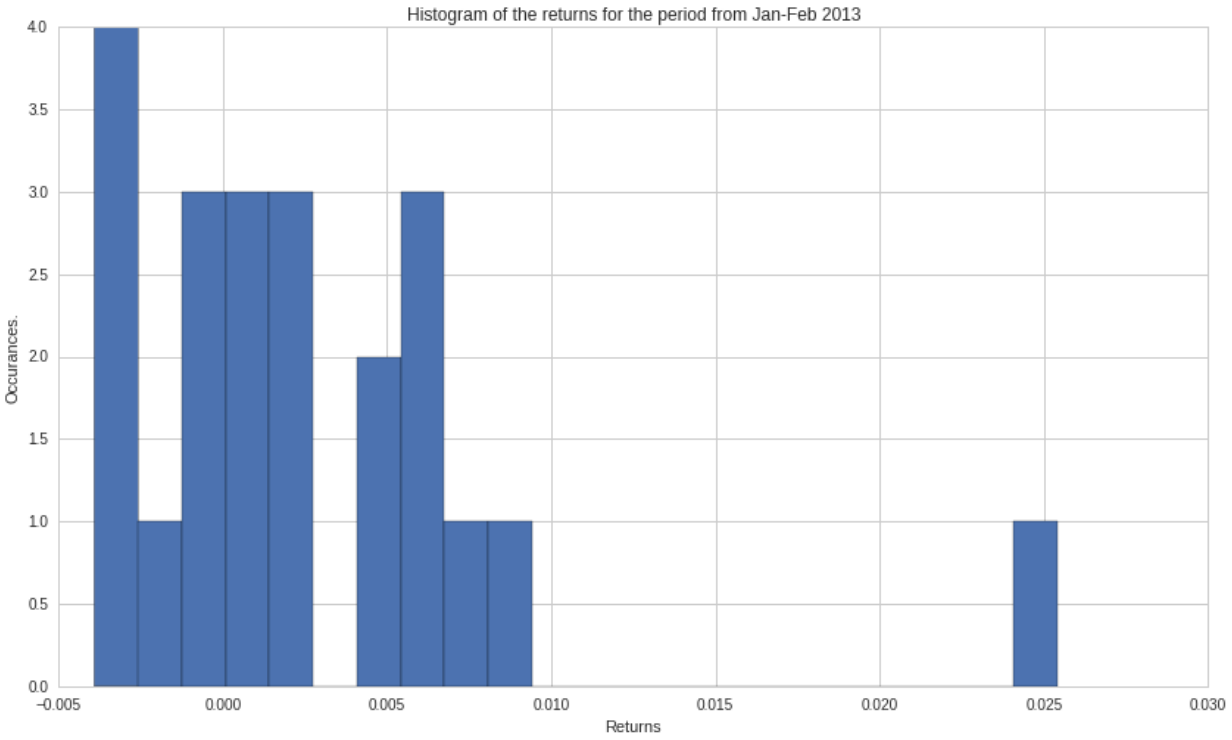
```
[Timestamp('2013-01-31 00:00:00', offset='M')]
```

We find the maximum positive skew value on the $31^{st} of January 2013$. We will then plot the histogram of the skewness for the returns of February, and see the returns with highest probability.

In [22]:

```
returns2 = spy['2013-01-01':'2013-02-01']

hist2 = plt.hist(returns2, bins=len(returns2));
plt.title('Histogram of the returns for the period from Jan-Feb 2013')
plt.xlabel('Returns')
plt.ylabel('Occurances.')

y2, x2, _ = hist2

print 'Maximum value of returns is found on :', np.argmax(returns2)
print 'Return with the most frequency was:', x2.max(), 'with a value of :', y2.m
ax(), 'occurances'
```

```
Maximum value of returns is found on : 2013-01-02 00:00:00+00:00
Return with the most frequency was: 0.0254177292277 with a value of
: 4.0 occurances
```

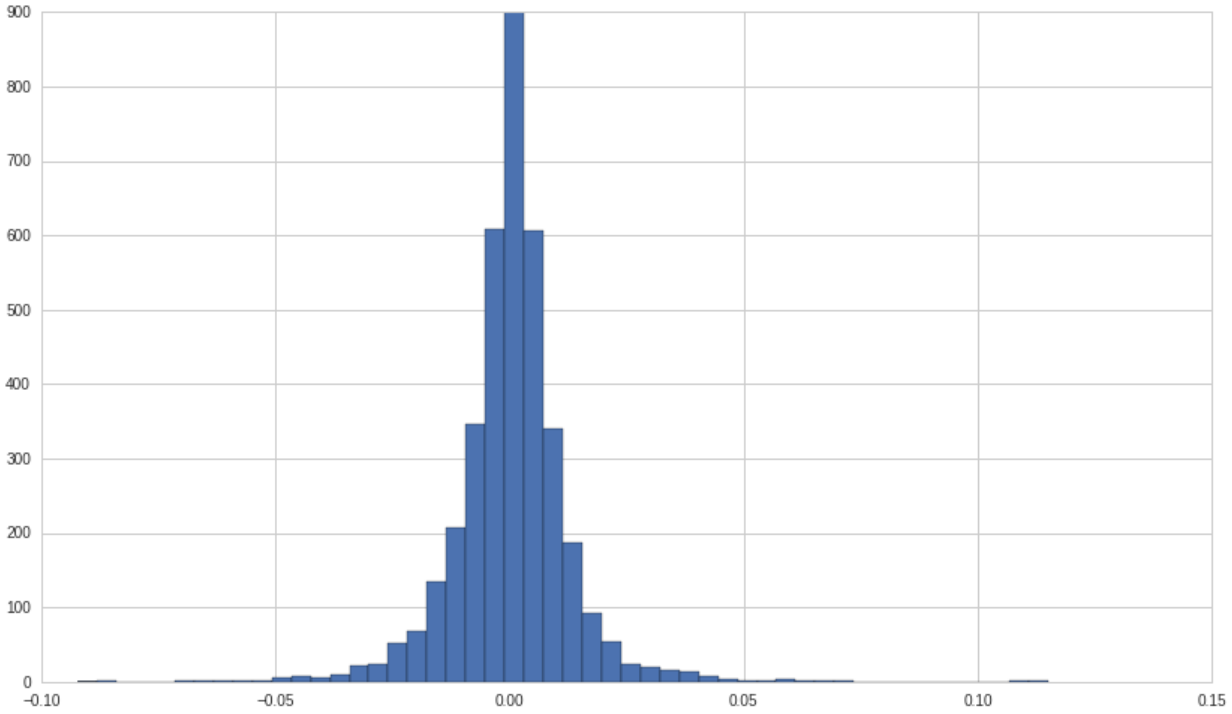


We find there is a delay between the day at which the maximum value of the return is found, and the value of greatest skewing. In the case of negative skewing, there is a 14 day delay, and in the case of positive skewing, there is a 30 day delay.

In [23]:

```
print 'skewing of the SPY index:', stats.skew(spy)
plt.hist(spy, bins=50);
```

```
skewing of the SPY index: 0.0287814439145
```

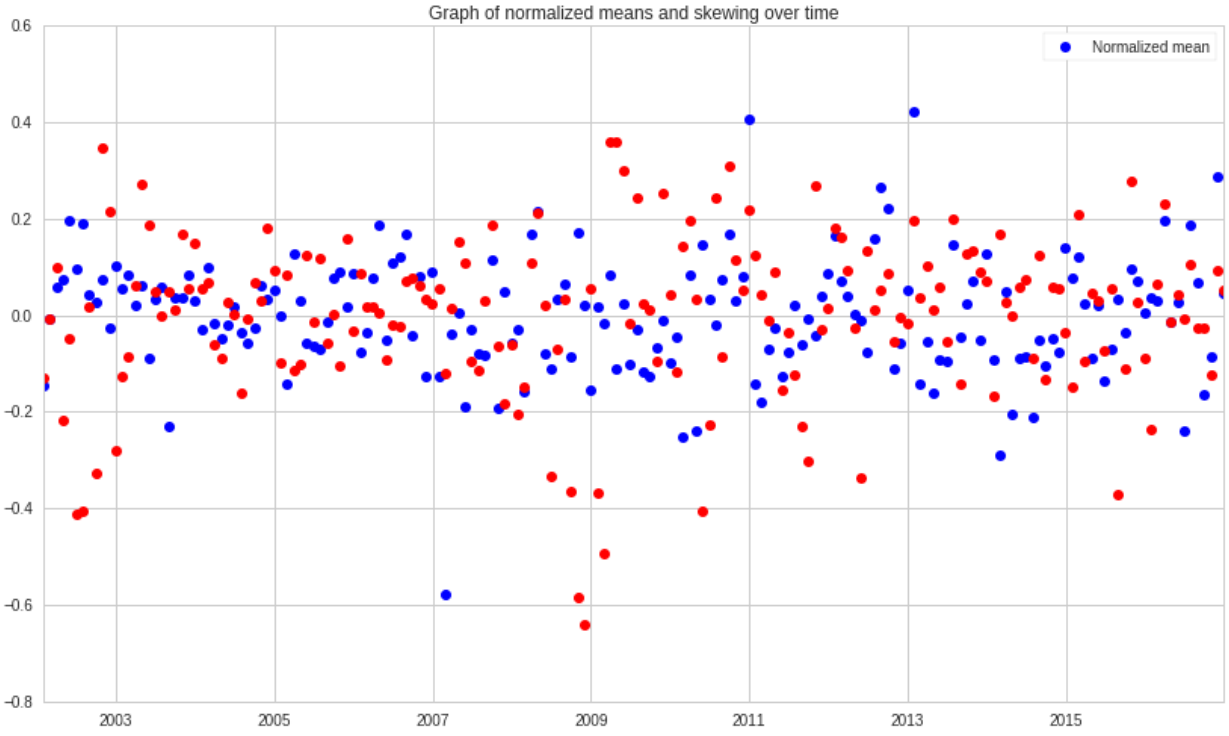


It is interesting to note that although there exists Skewing in the daily returns analyses over month, the S&P500 has a natural tendency to be very slightly positivly skewed, to the point where we can treat it as a normal distribution.

Autocorrelation, Regression & Error analysis

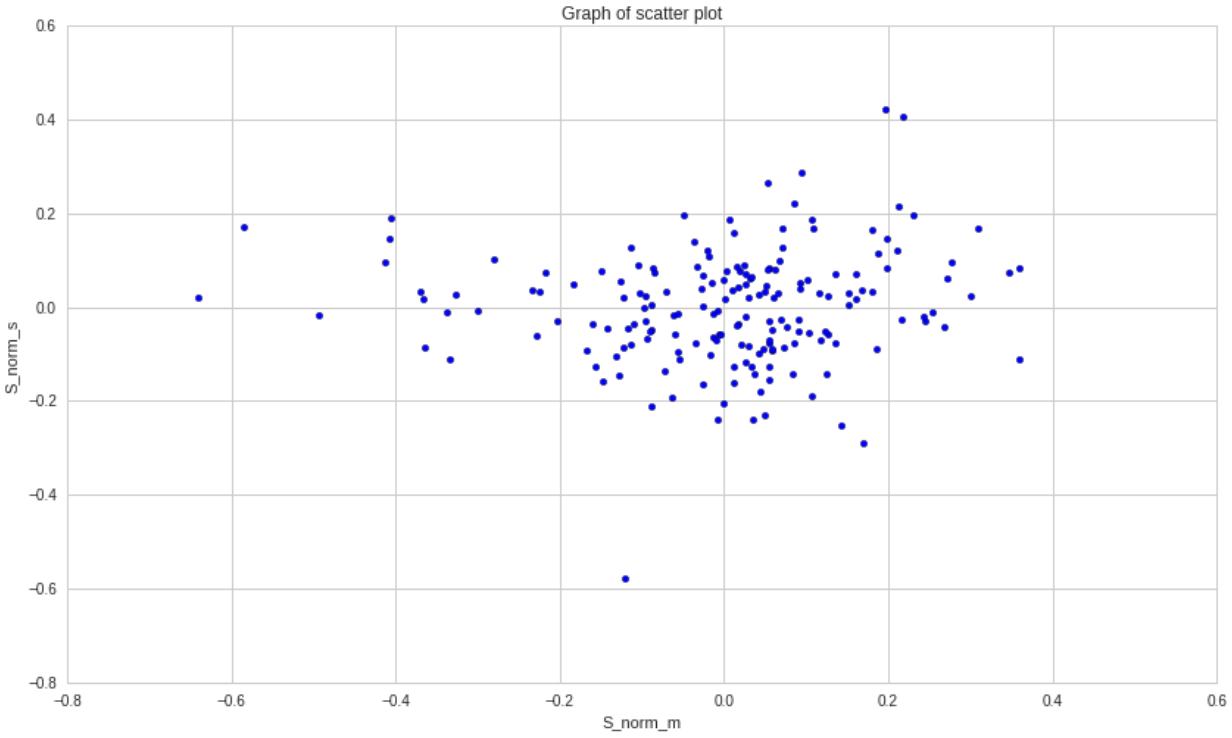
In [24]:

```
S_norm_s.name = 'Normalized Skewing'
S_norm_s.name = 'Normalized mean'

plt.plot(S_norm_s, 'bo')
plt.plot(S_norm_m, 'ro')
plt.title('Graph of normalized means and skewing over time')
plt.legend();
```



In [25]:

```
plt.scatter(S_norm_m, S_norm_s)
plt.title('Graph of scatter plot')
plt.xlabel('S_norm_m')
plt.ylabel('S_norm_s');
```



Autocorrelation

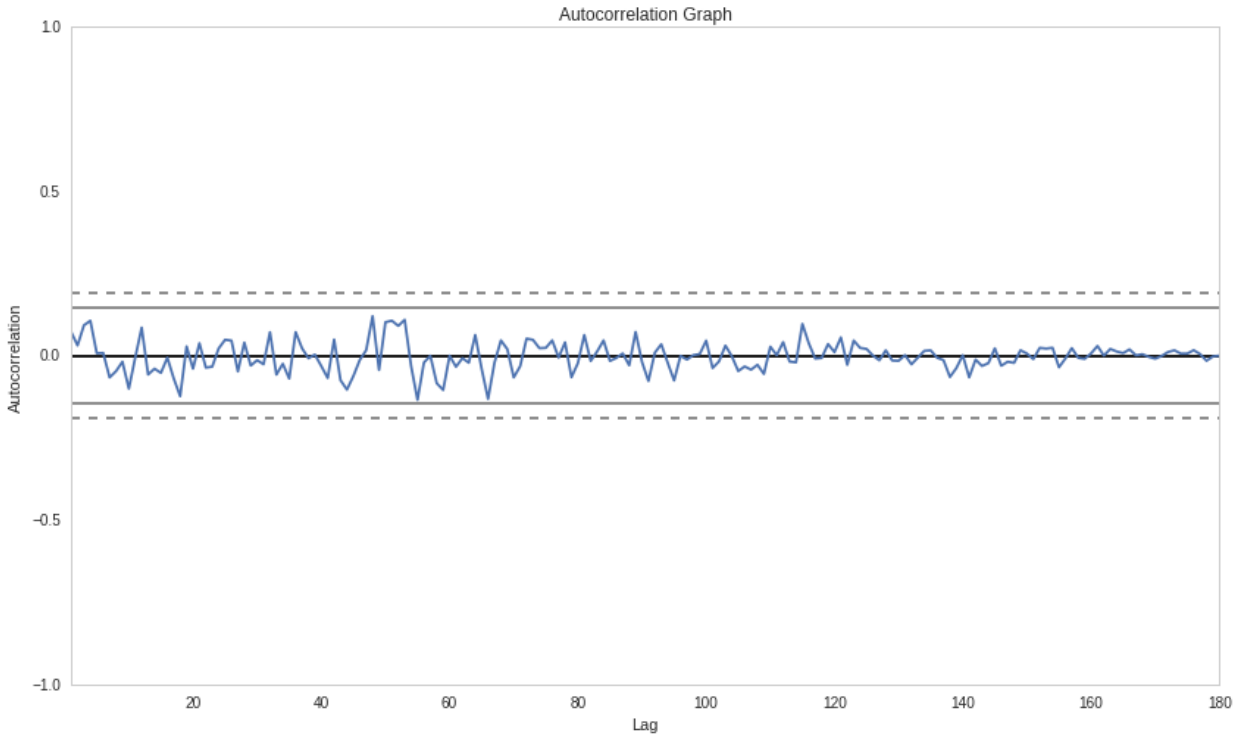
In [26]:

```
print 'Value of autocorrelation with lag 5:', S_norm_s.autocorr(5)
```

```
Value of autocorrelation with lag 5: 0.00745740707789
```

In [27]:

```
from pandas.tools.plotting import autocorrelation_plot
autocorrelation_plot(S_norm_s);
plt.title('Autocorrelation Graph');
```



We can see in this autocorrelation plot that over time the autocorrelation gets closer and closer to 0. This would indicate that there is near randomness in the value of our Skewing.

Linear Regression

In [28]:

```
slr = regression.linear_model.OLS(S_norm_s, sm.add_constant(S_norm_m)).fit()
print 'First parameter of the slr:', slr.params[0]
print 'Second parameter of the slr:', slr.params[1]
```

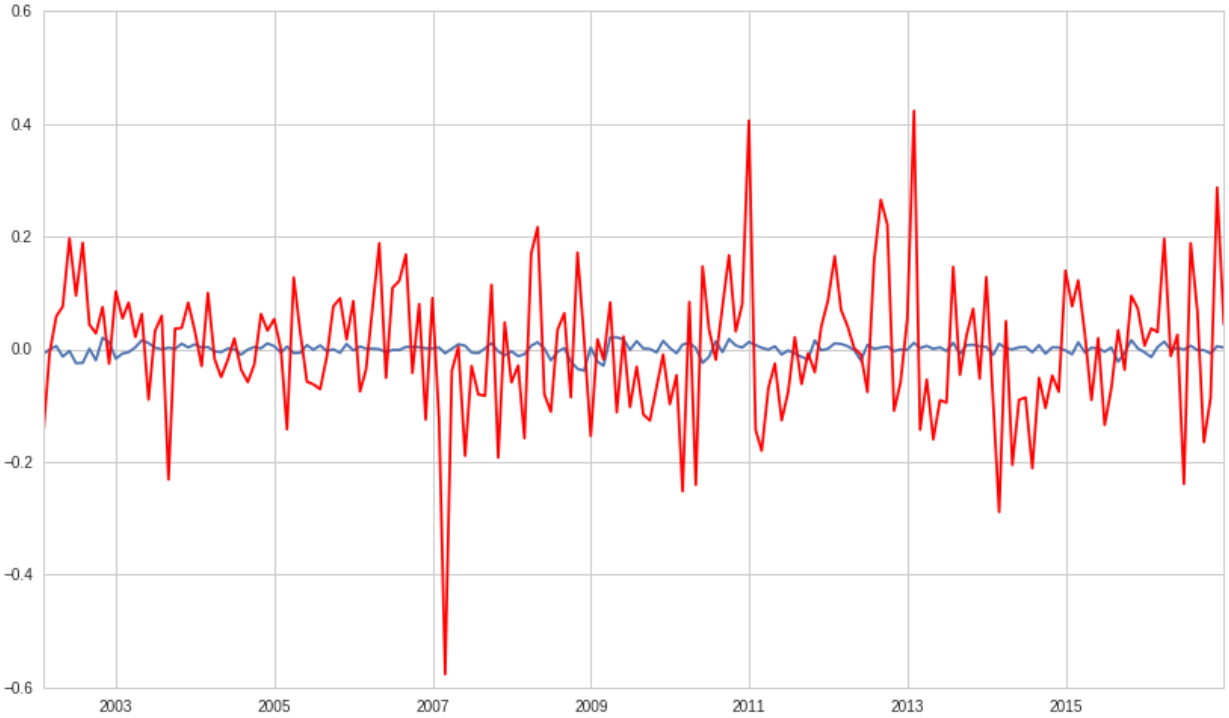
```
First parameter of the slr: -1.73472347598e-18
Second parameter of the slr: 0.0592810025521
```

In [29]:

```
predictions = slr.params[0] + slr.params[1]*S_norm_m
```

In [30]:

```
plt.plot(predictions)
plt.plot(S_norm_s, color = 'red');
```

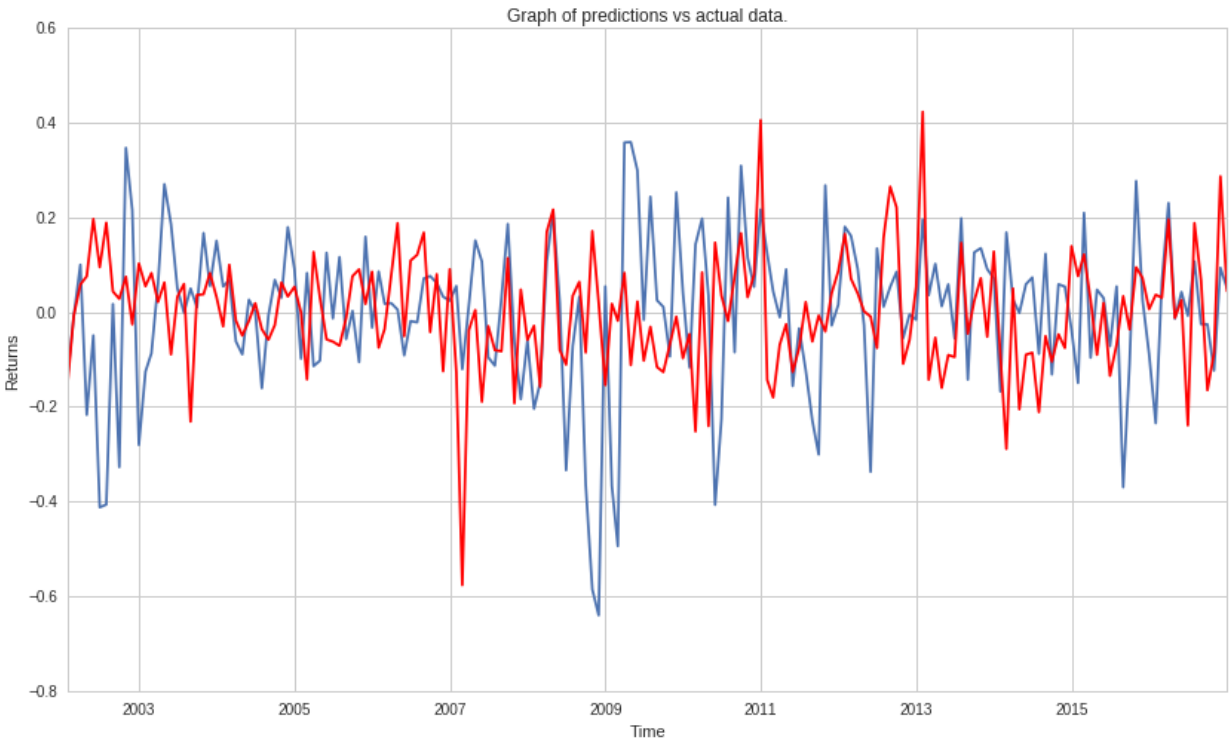


In [31]:

```
P_norm_s = ( (predictions - predictions.mean()) / (predictions.max() - predictio
ns.min()))
```

In [32]:

```
plt.plot(P_norm_s)
plt.plot(S_norm_s, color = 'red')
plt.xlabel('Time')
plt.ylabel('Returns')
plt.title('Graph of predictions vs actual data.');
```



Conclusion

In this research project, we attempted to find a way to predict the skew dynamics in returns of SPY the S&P 500 ETF index. To start with, we set upon finding the skew monthly values. This is because we needed enough data points to have valuable skew values, while at the same time needing the right amount of data points. Once that was done, we set about finding the monthly means. This was for comparasion from monthly returns to skew values. A linear regression was done to try and find an equation that best fited the monthly skew values. Along with scatter plot of skew values and monthly values. We attempted to find the autocorrealtion in the data, to find if the previous months skew value could impact the present one. This was based on the paper earlier sited, indicating that markets have a natural tendency to be positivly skewed.

We find that on a monthly scale, there are more negativly skewed Datpoints, and that there isn't much correlation between previous and present skew values. While our linear regression seemed to provide a somewhat similar plot, we were unable to take the error into consideration given the nomralization.

We concluded that while it is not possible follwing this research to accuratly predict skewing movements, it would be interesting to further it by running Mulitple linear regression model on three variables (variance perhaps) and to try and find more accurate instances of outling skewness in the returns in other assets as well. Another interesting note would be to take minute data, and find the daily skew values. This would give the added advantage of providing us with more datapoints, and we could perhaps see the transitions from positive to negative skewing more accuratly.

Acknowledgment

Big thanks to Quantopian for their platform and free access to market data.