Simulation

In [1]:
```
import pandas as pd
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
#Import charting library
import plotly
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, ipl

# Generate graphs offline inside a Jupyter Notebook Environment
init_notebook_mode(connected=True)
```

In [2]:
```
stocks =pd.read_csv('../csv/stocks.csv', index_col=0)
true_values = pd.read_csv('../csv/stocks_true.csv')
predictions = pd.read_csv('../csv/stocks_preds.csv', index_col=0)
true_values.rename( columns={'Unnamed: 0':'index'}, inplace=True )
true_values.head()
```

Out[2]:

	index	delta_7_np
0	419	1
1	420	1
2	421	1
3	422	1
4	423	1

In [3]:
```
true_values = true_values.set_index('index')
```

In [4]:
```
true_values.head(1)
```

Out[4]:

index	delta_7_np
419	1

In [5]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
tickers = stocks.ticker
ticker_encoder = LabelEncoder()
ticker_encoded = ticker_encoder.fit_transform(tickers)
```

Graph predictions

In [6]:
```
ticker_codes = dict(zip(ticker_encoder.transform(ticker_encoder.classes_),t
```

In [7]:
```
predictions['ticker'] = predictions['ticker'].apply(lambda x: ticker_codes[
```

In [8]:
```
predictions.ticker.unique()
```

Out[8]:
```
array(['amzn', 'ebay', 'aapl', 'msft', 'fcbk'], dtype=object)
```

In [9]:
```
predictions.shape[0]
```

Out[9]:
```
105
```

In [10]:
```
dates = stocks.date.tail(predictions.shape[0])
```

In [11]:
```
dates = dates.tolist()
```

In [12]:
```
predictions['date'] = dates
```

In [13]:
```
predictions.head()
```

Out[13]:

	pred	true	ticker	date
0	0	1	amzn	2017-10-23
1	1	1	amzn	2017-11-15
2	1	1	amzn	2017-12-12
3	1	1	amzn	2018-04-12
4	1	1	amzn	2018-05-01

In [14]:
```
from yahoo_fin.stock_info import *
yfin_a = get_data('aapl', start_date = '2010/07/21', end_date = '2019/05/10
yfin_z = get_data('amzn', start_date = '2010/07/21', end_date = '2019/05/10
yfin_c = get_data('csco', start_date = '2010/07/21', end_date = '2019/05/10
yfin_e = get_data('ebay', start_date = '2010/07/21', end_date = '2019/05/10
yfin_f = get_data('fb', start_date = '2010/07/21', end_date = '2019/05/10',
yfin_m = get_data('msft', start_date = '2010/07/21', end_date = '2019/05/10
```

In [15]:
```
yfin = yfin_a
yfin = yfin.append(yfin_z, ignore_index=False, sort=None)
yfin = yfin.append(yfin_c, ignore_index=False, sort=None)
yfin = yfin.append(yfin_e, ignore_index=False, sort=None)
yfin = yfin.append(yfin_f, ignore_index=False, sort=None)
yfin = yfin.append(yfin_m, ignore_index=False, sort=None)
```

In [16]:
```
yfin.sort_values('date').head()
```

Out[16]:

date	open	high	low	close	adjclose	volume	ticker
2010-07-21	37.869999	37.878571	36.285713	36.320000	24.125383	296417800.0	AAPL
2010-07-21	120.620003	121.250000	117.260002	117.430000	117.430000	5011700.0	AMZN
2010-07-21	25.600000	25.650000	24.980000	25.120001	20.088602	73297300.0	MSFT
2010-07-21	23.059999	23.219999	22.400000	22.559999	17.837521	45752100.0	CSCO
2010-07-21	8.716330	8.817340	8.430135	8.489058	8.457433	56822200.0	EBAY

In [17]:
```
cols = ['ticker', 'adjclose']
yfin = yfin[cols]
yfin['ticker']= yfin['ticker'].str.lower()
yfin.to_csv('../csv/yfin.csv')
```

In [18]:
```
yfin = pd.read_csv('../csv/yfin.csv', index_col=0)
```

In [19]:
```
yfin.index = pd.to_datetime(yfin.index)
```

In [20]:
```
predictions['date'] = pd.to_datetime(predictions['date'])
cols = ['date', 'pred', 'ticker']
predictions = predictions [cols]
```

In [21]:
```
predictions = predictions.sort_values(['date'])
predictions.shape
```

Out[21]:
```
(105, 3)
```

In [22]:
```
yfin.rename({'fcbk':'fb'}, axis =1, inplace=True)
```

In [23]:
```
yfin.tail()
```

Out[23]:

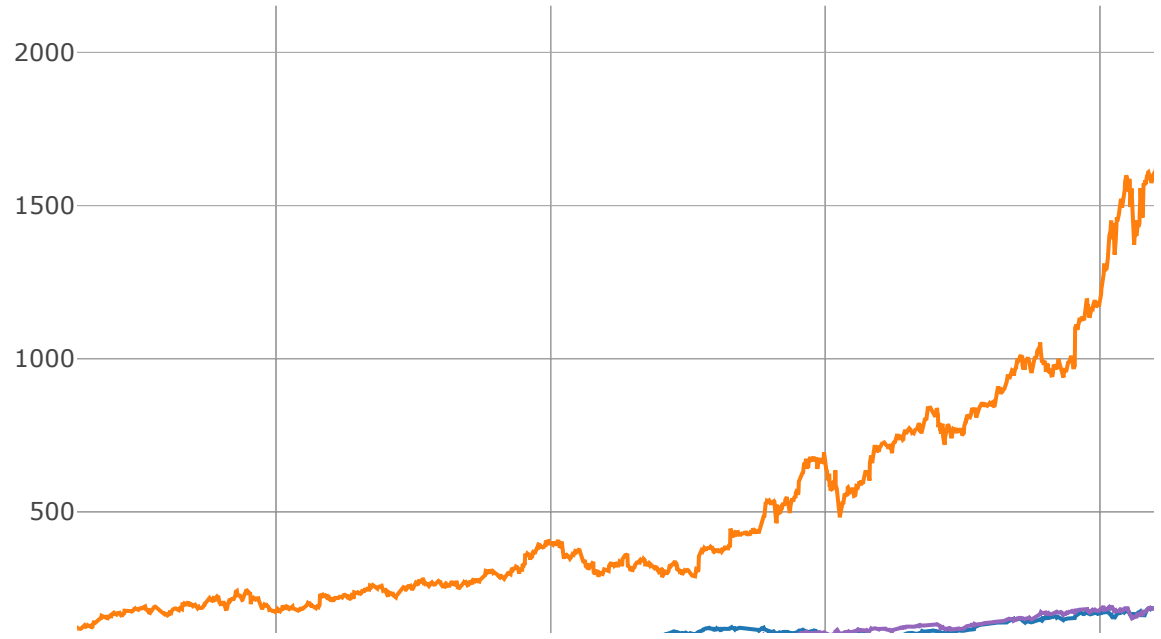
date	ticker	adjclose
2019-05-03	msft	128.424606
2019-05-06	msft	127.677376
2019-05-07	msft	125.057083
2019-05-08	msft	125.047127
2019-05-09	msft	125.037155

In [24]:
```
predictions.ticker.unique()
```

Out[24]:
```
array(['ebay', 'amzn', 'aapl', 'msft', 'fcbk'], dtype=object)
```

In [25]:
```
yfin = yfin.reset_index()
```

In [26]:
```
#Import charting library
import plotly
import plotly.graph_objs as go
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, ipl

#To generate graphs offline inside a Jupyter Notebook Environment
init_notebook_mode(connected=True)

#To plot the adj closing for each stock on the chart
traces=[]
for i in yfin.ticker.unique().tolist():
    df= yfin[yfin['ticker'] == i ]
    traces.append(go.Scatter(
        x = df.date,
        y = df.adjclose,
        mode = 'lines',
        name = i,
    ))

data = traces
iplot(data)
```



In [27]:
```
model = pd.merge(predictions, yfin, on=['date', 'ticker'])
model.tail()
```

Out[27]:

	date	pred	ticker	adjclose
95	2018-10-30	1	ebay	27.317850
96	2018-11-14	1	amzn	1599.010010
97	2018-12-13	1	amzn	1658.380005
98	2019-01-29	1	msft	102.124214
99	2019-02-14	0	ebay	36.184692

In [28]:
```
model = model.sort_values(by=['ticker', 'date'])
```

In [29]:
```
model.loc[model['pred']==0,'cash_flow']=1000
model.loc[model['pred']==1,'cash_flow']=-1000
model.tail()
```

Out[29]:

	date	pred	ticker	adjclose	cash_flow
74	2017-05-22	1	msft	66.066414	-1000.0
76	2017-07-20	1	msft	71.635506	-1000.0
86	2018-05-15	1	msft	95.351631	-1000.0
90	2018-06-14	1	msft	99.799400	-1000.0
98	2019-01-29	1	msft	102.124214	-1000.0

In [30]:
```
model['quantity'] = (-model['cash_flow']/model['adjclose'])
model.head()
```

Out[30]:

	date	pred	ticker	adjclose	cash_flow	quantity
4	2010-10-28	0	aapl	28.964882	1000.0	-34.524567
9	2011-04-27	1	aapl	33.226479	-1000.0	30.096479
12	2011-06-28	1	aapl	31.813543	-1000.0	31.433154
13	2011-07-20	1	aapl	36.713772	-1000.0	27.237735
19	2012-01-18	1	aapl	40.719177	-1000.0	24.558453

In [31]:
```
grouped = model.groupby('ticker')

gains_list= []
for first, model_ticker in grouped:
    model_ticker['gains'] = (model_ticker['adjclose'].shift(1)-model_ticker
    #model_ticker['ticker'] = first
    gains_list.append(model_ticker)
```

In [32]:
```
model = pd.concat(gains_list, axis=0)
model.head()
```

Out[32]:

	date	pred	ticker	adjclose	cash_flow	quantity	gains
4	2010-10-28	0	aapl	28.964882	1000.0	-34.524567	NaN
9	2011-04-27	1	aapl	33.226479	-1000.0	30.096479	-147.129779
12	2011-06-28	1	aapl	31.813543	-1000.0	31.433154	-42.524376
13	2011-07-20	1	aapl	36.713772	-1000.0	27.237735	154.029636
19	2012-01-18	1	aapl	40.719177	-1000.0	24.558453	109.098173

In [33]:
```
pnl = model['gains'].shift(-1).dropna()
round((sum(pnl)/-sum(model.cash_flow))*100,2)
```

Out[33]: 8.36

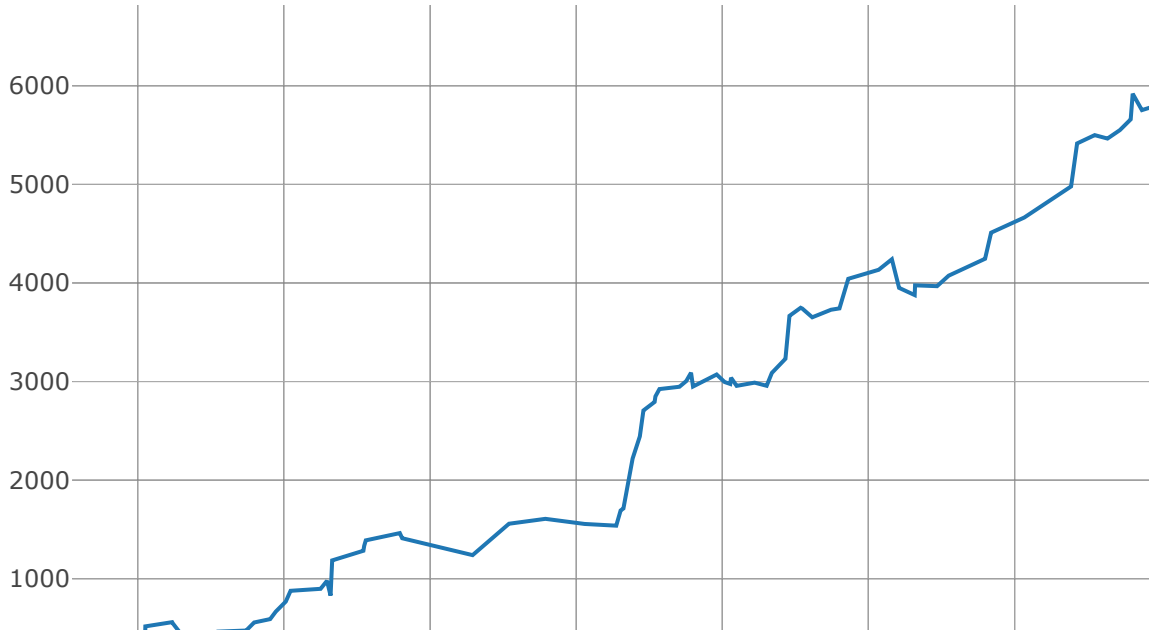
In [34]:
```
model = model.sort_values('date')
```

In [35]:
```
model.head()
```

Out[35]:

	date	pred	ticker	adjclose	cash_flow	quantity	gains
0	2010-07-21	1	ebay	8.457433	-1000.0	118.239190	NaN
1	2010-09-21	1	amzn	150.729996	-1000.0	6.634380	NaN
2	2010-10-05	0	ebay	10.310771	1000.0	-96.985958	219.137213
3	2010-10-20	1	ebay	10.759430	-1000.0	92.941727	-43.513617
4	2010-10-28	0	aapl	28.964882	1000.0	-34.524567	NaN

In [36]:
```
#add a cumulative column to the model dataframe
model['cum_date']= model['gains'].cumsum()

# plot the cumulative gain for all
traces=[]

traces.append(go.Scatter(
    x = model['date'],
    y = model['cum_date'],
    mode = 'lines',
    name = 'model',
    ))

data = traces
iplot(data)
```



In [37]:
```
market = model[['date', 'ticker', 'adjclose']]
```

In [38]:
```
market.tail()
```

Out[38]:

	date	ticker	adjclose
95	2018-10-30	ebay	27.317850
96	2018-11-14	amzn	1599.010010
97	2018-12-13	amzn	1658.380005
98	2019-01-29	msft	102.124214
99	2019-02-14	ebay	36.184692

In [39]:
```
market['pred']=1
market['cash_flow'] = -1000
market.head()
```

Out[39]:

	date	ticker	adjclose	pred	cash_flow
0	2010-07-21	ebay	8.457433	1	-1000
1	2010-09-21	amzn	150.729996	1	-1000
2	2010-10-05	ebay	10.310771	1	-1000
3	2010-10-20	ebay	10.759430	1	-1000
4	2010-10-28	aapl	28.964882	1	-1000

In [40]:
```
market['quantity'] = (-market['cash_flow']/market['adjclose'])
```

In [41]:
```
# model.loc[model['pred']==0,'Quantity']= (-model['cash_flow']/model['adjcl
# model.loc[model['pred']==1,'Quantity']= (model['cash_flow']/model['adjclo
market.head(6)
```

Out[41]:

	date	ticker	adjclose	pred	cash_flow	quantity
0	2010-07-21	ebay	8.457433	1	-1000	118.239190
1	2010-09-21	amzn	150.729996	1	-1000	6.634380
2	2010-10-05	ebay	10.310771	1	-1000	96.985958
3	2010-10-20	ebay	10.759430	1	-1000	92.941727
4	2010-10-28	aapl	28.964882	1	-1000	34.524567
5	2011-01-19	ebay	12.201848	1	-1000	81.954799

In [42]:
```
grouped = market.groupby('ticker')

gains_list= []
for first, market_ticker in grouped:
    market_ticker['gains'] = (market_ticker['adjclose'].shift(1)-market_tic
    #market_ticker['ticker'] = first
    gains_list.append(market_ticker)
```

In [43]:
```
market = pd.concat(gains_list, axis=0)
market.head()
```

Out[43]:

	date	ticker	adjclose	pred	cash_flow	quantity	gains
4	2010-10-28	aapl	28.964882	1	-1000	34.524567	NaN
9	2011-04-27	aapl	33.226479	1	-1000	30.096479	147.129779
12	2011-06-28	aapl	31.813543	1	-1000	31.433154	-42.524376
13	2011-07-20	aapl	36.713772	1	-1000	27.237735	154.029636
19	2012-01-18	aapl	40.719177	1	-1000	24.558453	109.098173

In [44]:
```
pnl = market['gains'].shift(-1).dropna()
round((sum(pnl)/-sum(market.cash_flow))*100,2)
```

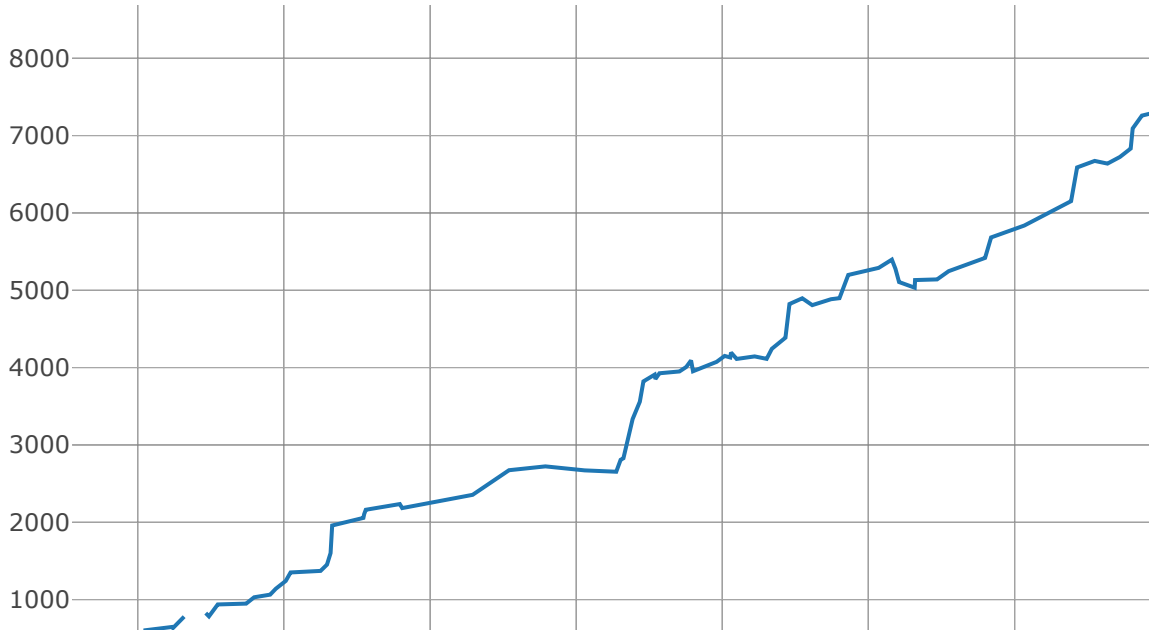
Out[44]: 8.14

In [45]:
```
market = market.sort_values('date')
market.head()
```

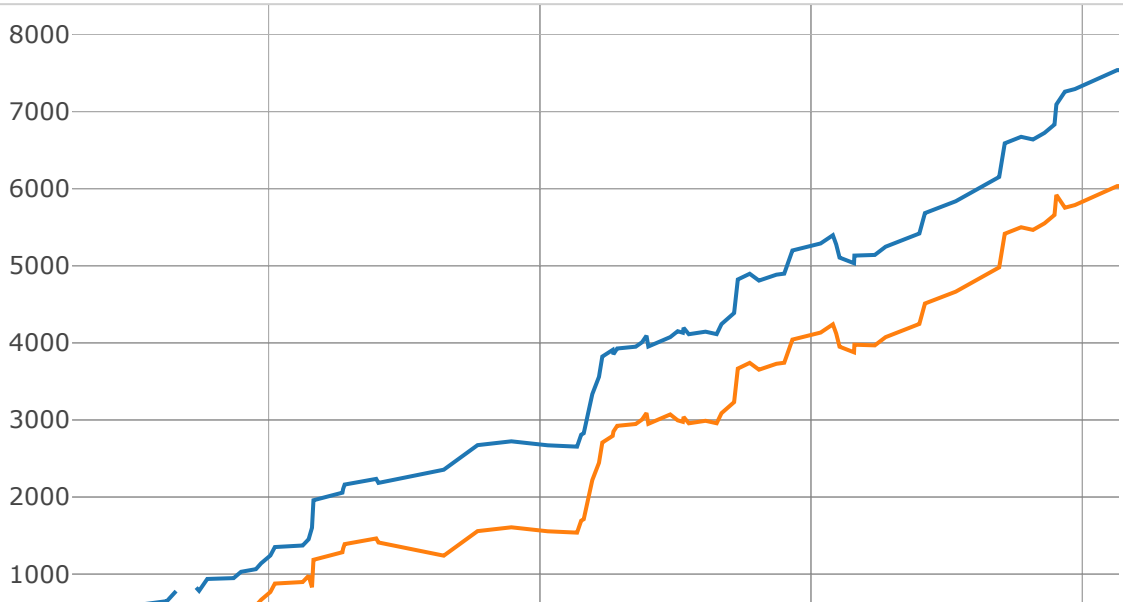
Out[45]:

	date	ticker	adjclose	pred	cash_flow	quantity	gains
0	2010-07-21	ebay	8.457433	1	-1000	118.239190	NaN
1	2010-09-21	amzn	150.729996	1	-1000	6.634380	NaN
2	2010-10-05	ebay	10.310771	1	-1000	96.985958	219.137213
3	2010-10-20	ebay	10.759430	1	-1000	92.941727	43.513617
4	2010-10-28	aapl	28.964882	1	-1000	34.524567	NaN

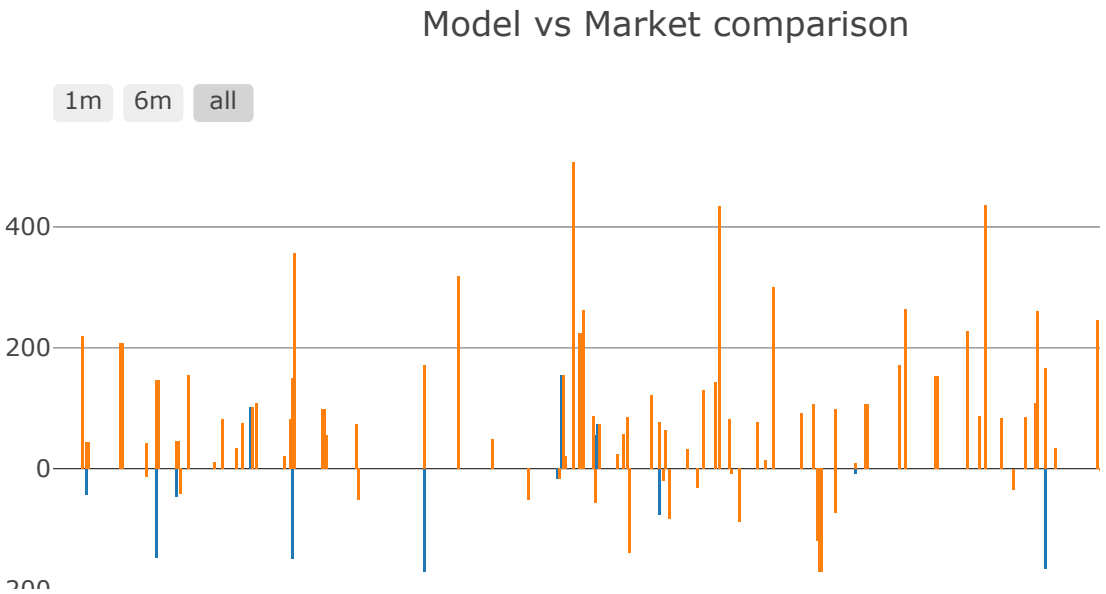
In [46]:
```
# add a cumulative column to the market dataframe
market['cum_date']= market['gains'].cumsum()

# plot the cumulative gain for all
traces=[]

traces.append(go.Scatter(
    x = market['date'],
    y = market['cum_date'],
    mode = 'lines',
    name = 'market',
))

data = traces
iplot(data)
```



In [49]:
```
trace_market = go.Scatter(
    x = market['date'],
    y = market['cum_date'],
    mode = 'lines',
    name = 'market',
)
trace_model = go.Scatter(
    x = model['date'],
    y = model['cum_date'],
    mode = 'lines',
    name = 'model',
)

data = [trace_market, trace_model]
iplot(data)
```



In [48]:
```
# Compare both model and market on a daily basis
trace1=go.Bar(
        x = model['date'],
        y = model['gains'],
        name = 'model',
    )
trace2=go.Bar(
        x = market['date'],
        y = market['gains'],
        name = 'market',
    )
layout = dict(
    title='Model vs Market comparison',
    xaxis=dict(
        rangeselector=dict(
            buttons=list([
                dict(count=1,
                     label='1m',
                     step='month',
                     stepmode='backward'),
                dict(count=6,
                     label='6m',
                     step='month',
                     stepmode='backward'),
                dict(step='all')
            ])
        ),
        rangeslider=dict(
            visible = True
        ),
        type='date'
    )
)

data = [trace1,trace2]
fig = dict(data=data, layout=layout)
iplot(fig)
```



−200

In [ ]: