

In [1]:

```
import pandas as pd
```

In [2]:

```
def weight_ew(r):
    n=len(r.columns)
    return pd.Series(1/n,index=r.columns)
```

In [7]:

```
def backtest(r,estimation_window=5,weighting=weight_ew):
    n_periods=r.shape[0]
    windows=[(start,start+estimation_window) for start in range(n_periods-estimation_window+1)]
    weights=[weighting(r.iloc[win[0]:win[1]]) for win in windows]
    weights=pd.DataFrame(weights, index=r.iloc[estimation_window-1:].index,columns=r.columns)
    returns=(weights*r).sum(axis="columns")[estimation_window:]
    return returns
```

In [9]:

```
ind=local_csv("ind49_m_ret.csv",header=0,index_col=0)/100
ind.index=pd.to_datetime(ind.index,format="%Y%m").to_period('M')
ind.columns=ind.columns.str.strip()
ind=ind["1970":]
```

In [11]:

```
ind.head()
```

Out[11]:

	Agric	Food	Soda	Beer	Smoke	Toys	Fun	Books	Hshld	Clths
1970-01	-0.0334	-0.0206	-0.0328	-0.0041	-0.0167	-0.0776	-0.0525	-0.0851	-0.0448	-0.0236
1970-02	-0.0251	0.0381	0.0337	0.0616	0.0670	0.0334	0.0739	-0.0021	0.0074	0.0168
1970-03	-0.0575	-0.0089	0.0437	-0.0035	0.0454	-0.0431	-0.0352	-0.0217	-0.0297	-0.0211
1970-04	-0.1790	-0.1113	-0.0941	-0.1305	-0.0329	-0.2225	-0.2416	-0.1692	-0.1772	-0.1626
1970-05	-0.2589	-0.0988	-0.0489	-0.0600	-0.0066	-0.1003	-0.0989	-0.1661	-0.1077	-0.0703

5 rows × 49 columns



In [12]:

```
ew=backtest(ind,weighting=weight_ew)
```

In [14]:

```
wealth=(1+ew).cumprod()
wealth.plot(title="Equal Weighted Industry Portfolios")
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f15bc99f128>
```



In [17]:

```
ind_nfirms=local_csv("ind49_m_nfirms.csv",header=0,index_col=0)
ind_nfirms.index=pd.to_datetime(ind_nfirms.index,format="%Y%m").to_period('M')
ind_nfirms.columns=ind_nfirms.columns.str.strip()
ind_nfirms=ind_nfirms["1970":]
```

In [18]:

```
ind_size=local_csv("ind49_m_size.csv",header=0,index_col=0)
ind_size.index=pd.to_datetime(ind_size.index,format="%Y%m").to_period('M')
ind_size.columns=ind_size.columns.str.strip()
ind_size=ind_nfirms["1970":]
```

In [19]:

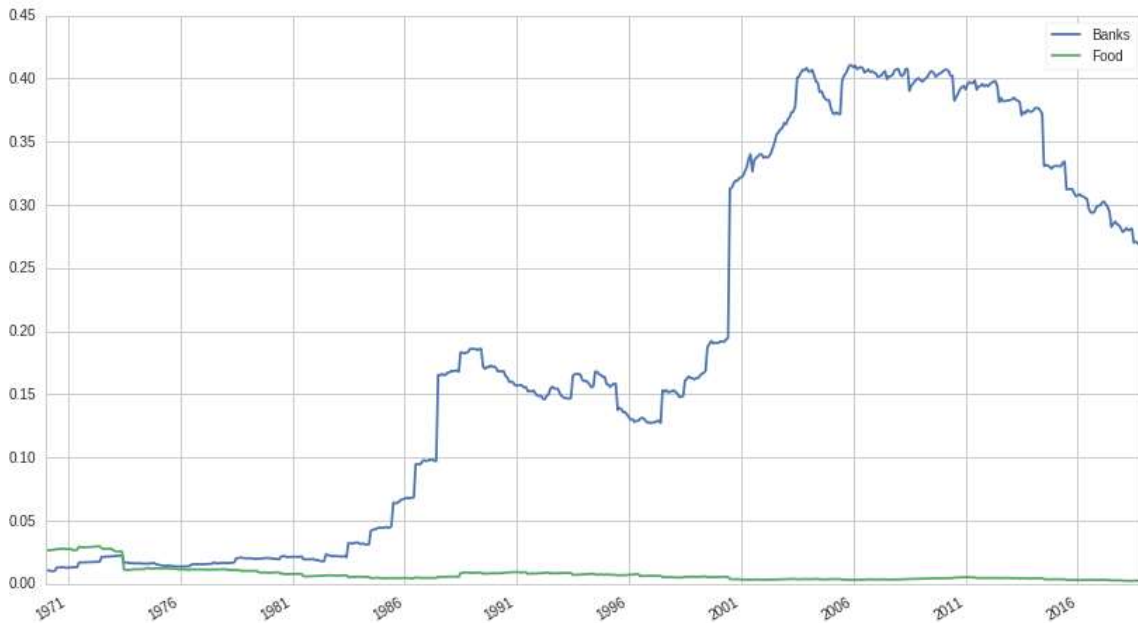
```
ind_cap=ind_nfirms*ind_size
total_cap=ind_cap.sum(axis=1)
ind_cap_weight=ind_cap.divide(total_cap,axis="rows")
```

In [20]:

```
ind_cap_weight[["Banks", "Food"]].plot.line()
```

Out[20]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f15bc90ed68>
```



In [21]:

```
def weight_ew(r,**kwargs):
    n=len(r.columns)
    return pd.Series(1/n,index=r.columns)
```

In [22]:

```
def weight_cw(r,cap_weights,**kwargs):
    return cap_weights.loc[r.index[0]]
```

In [23]:

```
def backtest(r,estimation_window=5,weighting=weight_ew,**kwargs):
    n_periods=r.shape[0]
    windows=[(start,start+estimation_window) for start in range(n_periods-estimation_window+1)]
    weights=[weighting(r.iloc[win[0]:win[1]],**kwargs) for win in windows]
    weights=pd.DataFrame(weights, index=r.iloc[estimation_window-1:].index,columns=r.columns)
    returns=(weights*r).sum(axis="columns")[estimation_window:]
    return returns
```

In [24]:

```
equal_weighted=backtest(ind)
```

In [26]:

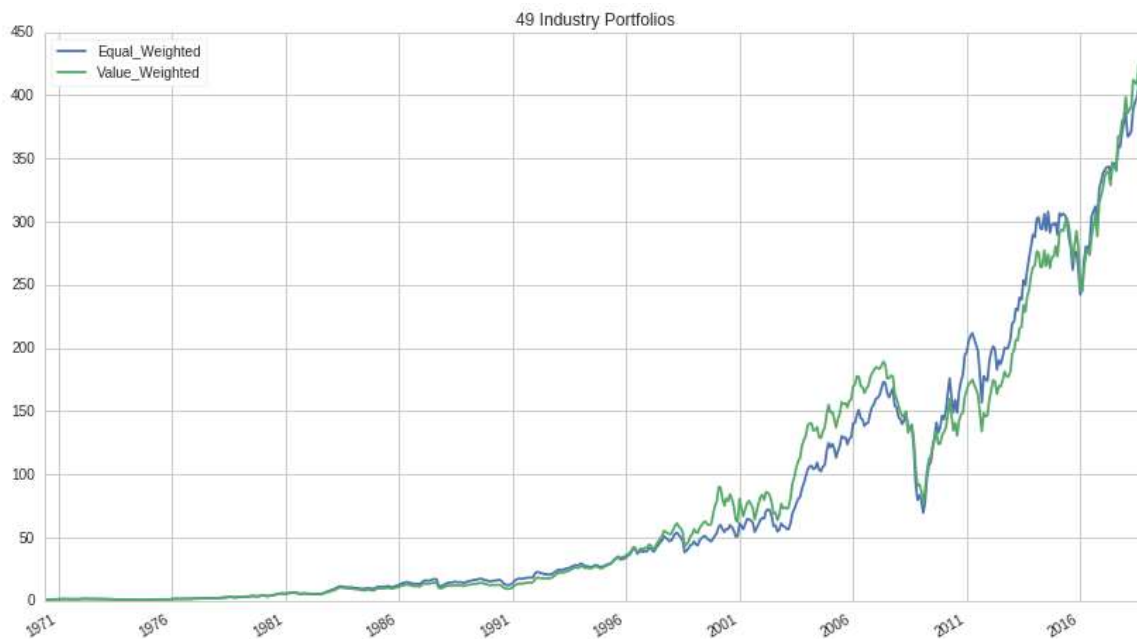
```
value_weighted=backtest(ind,weighting=weight_cw,cap_weights=ind_cap_weight)
```

In [27]:

```
ret=pd.DataFrame({
    "Equal_Weighted": equal_weighted,
    "Value_Weighted": value_weighted
})
```

In [29]:

```
wealth=(1+ret).cumprod().plot(title="49 Industry Portfolios")
```



In [32]:

```
def summary_stats(r,risk_free_rate=0.003):
    return pd.DataFrame({
        "Expected Return":r.mean(),
        "Volatility": r.std(),
        "Sharpe Ratio": (r.mean()-risk_free_rate)/r.std()
    })
```

In [33]:

```
summary_stats(ret)
```

Out[33]:

	Expected Return	Sharpe Ratio	Volatility
Equal_Weighted	0.011568	0.150422	0.056962
Value_Weighted	0.011581	0.149391	0.057439

## Othe libraries

In [34]:

```
import pyfolio as pf
```

In [35]:

```
import empyrical as em
```

In [36]:

```
market=pf.utils.get_symbol_rets('SPY')
```

In [37]:

```
stock=pf.utils.get_symbol_rets('FB')
```

In [40]:

```
print "Sharpe ratio:", em.sharpe_ratio(stock)
```

Sharpe ratio: 0.8469224381090443

In [41]:

```
print "market beta is:", em.beta(stock,market)
```

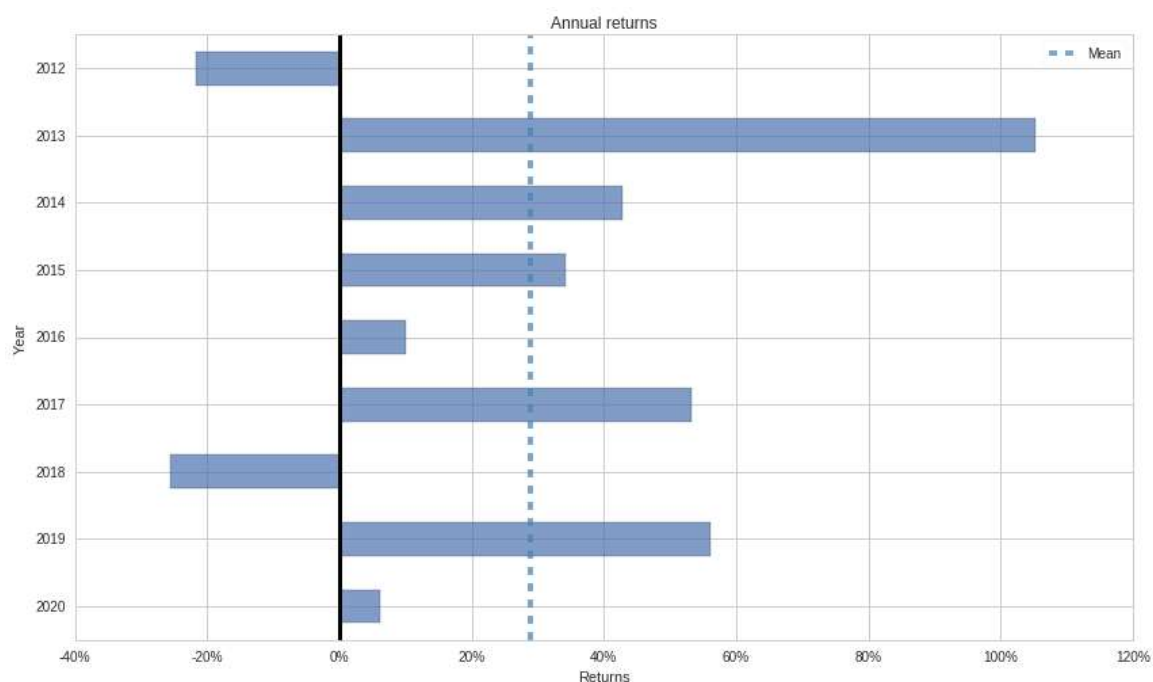
market beta is: 1.1499549635815556

In [42]:

```
pf.plot_annual_returns(stock)
```

Out[42]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f15c00a94a8>

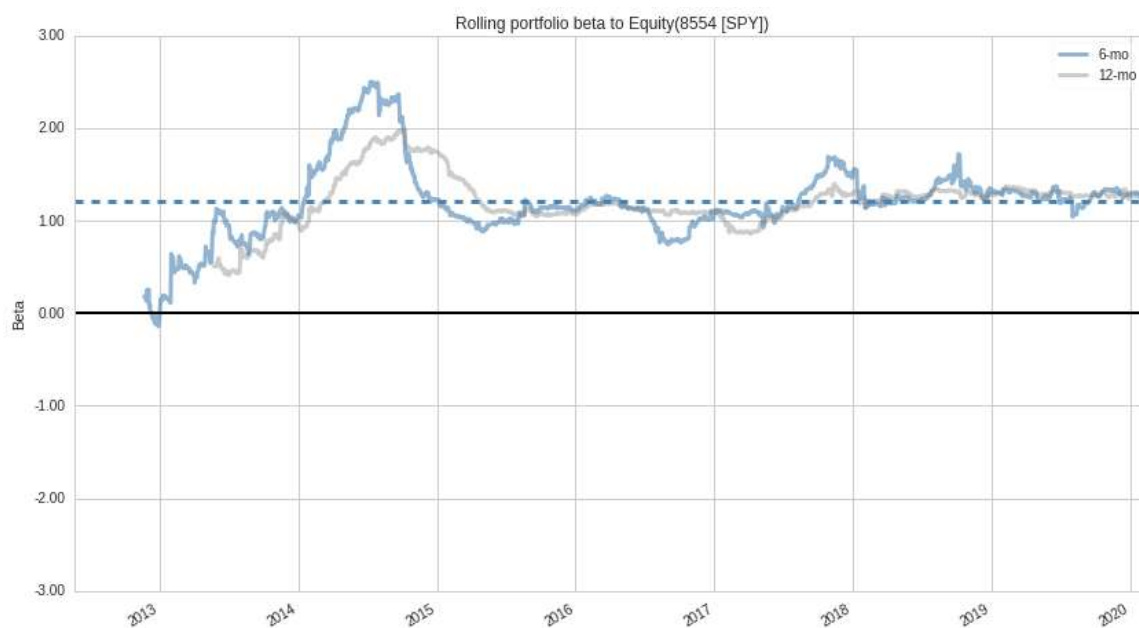


In [43]:

```
ax=pf.plot_rolling_beta(stock, market)
ax.set_ylim([-3,+3])
```

Out[43]:

(-3, 3)



In [ ]: