Rebalancing Portfolios II

In [1]:

```
import pandas as pd
import numpy  as np

from tiingo import TiingoClient
tiingo = TiingoClient({'api_key':'XXXX'})

import matplotlib.pyplot as plt                              # Basic plot library.
plt.style.use('ggplot')                                      # Make plots look nice.
```

Get S&P500 sector ETFs for technology, consumer staples and financials:

In [2]:

```
PRICE       = tiingo.get_dataframe(['XLK', 'XLP', 'XLF'],'2000-01-01', metric_na
PRICE.index = pd.to_datetime(PRICE.index).tz_convert(None)

RET = PRICE.pct_change()
RET[-3:]
```

Out[2]:

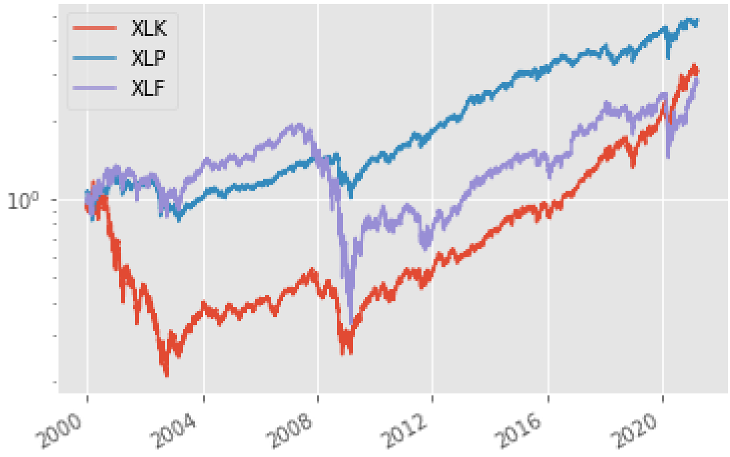
	XLK	XLP	XLF
2021-03-22	0.019623	0.010499	-0.012822
2021-03-23	-0.006218	0.004197	-0.013976
2021-03-24	-0.012133	-0.003881	0.003619

In [3]:

```
RET.add(1).cumprod().plot(logy=True)
```

Out[3]:

```
<AxesSubplot:>
```



Rebalance Loop

Equal-weight portfolio:

In [4]:

```
weights = pd.Series({'XLK':1/3, 'XLP':1/3, 'XLF':1/3})
```

Suppose we rebalance at the end of each month:

In [5]:

```
def get_rebalance_dates(frequency):
    group = getattr(PRICE.index, frequency)
    return PRICE[:1].index.union(PRICE.groupby([PRICE.index.year, group]).tail(1

rebalance_dates = get_rebalance_dates('month')
rebalance_dates
```

Out[5]:

```
DatetimeIndex(['2000-01-03', '2000-01-31', '2000-02-29', '2000-03-31',
               '2000-04-28', '2000-05-31', '2000-06-30', '2000-07-31',
               '2000-08-31', '2000-09-29',
               ...
               '2020-06-30', '2020-07-31', '2020-08-31', '2020-09-30',
               '2020-10-30', '2020-11-30', '2020-12-31', '2021-01-29',
               '2021-02-26', '2021-03-24'],
              dtype='datetime64[ns]', length=256, freq=None)
```

We use a series to store the daily portfolio values.

Our portfolio value equals $1 on the first trading date:

In [6]:

```
portfolio_value = pd.Series(1, index=[rebalance_dates[0]])
portfolio_value
```

Out[6]:

```
2000-01-03    1
dtype: int64
```

First holding period:

In [7]:

```
start_date = rebalance_dates[0]
end_date   = rebalance_dates[1]

print('start:', start_date, 'end:', end_date)
```

```
start: 2000-01-03 00:00:00 end: 2000-01-31 00:00:00
```

Compound return of the assets during this holding period:

In [8]:

```
cum_ret = RET[start_date:end_date][1:].add(1).cumprod()
cum_ret
```

Out[8]:

	XLK	XLP	XLF
2000-01-04	0.949315	0.971868	0.956294
2000-01-05	0.949856	0.986374	0.956731
2000-01-06	0.904221	1.007473	0.990385
2000-01-07	0.919913	1.071648	1.006556
2000-01-10	0.954906	1.047473	0.989073
2000-01-11	0.927850	1.047473	0.972465
2000-01-12	0.923882	1.052747	0.991696
2000-01-13	0.932359	1.058462	1.021853
2000-01-14	0.953824	1.061978	1.045455
2000-01-18	0.958874	1.043956	1.010927

	XLK	XLP	XLF
2000-01-19	0.960498	1.051429	1.007430
2000-01-20	0.966089	1.036923	0.989073
2000-01-21	0.967352	1.025495	0.976399
2000-01-24	0.935786	0.996044	0.958916
2000-01-25	0.955628	0.983736	0.968969
2000-01-26	0.924423	0.992527	1.005245
2000-01-27	0.915404	1.010989	1.016171
2000-01-28	0.882756	1.002637	0.980769
2000-01-31	0.911977	1.021099	1.007867

At the start of each period, our current portfolio value equals the most recent value of the last period:

In [9]:

```
portfolio_value.iloc[-1]
```

Out[9]:

```
1
```

Dollar amounts we invest in the assets at the start of the current holding period:

In [10]:

```
new_positions = portfolio_value.iloc[-1] * weights
new_positions
```

Out[10]:

```
XLK    0.333333
XLP    0.333333
XLF    0.333333
dtype: float64
```

Dollar amounts we have in these assets during the current holding period:

In [11]:

```
start_to_end_positions = new_positions  * cum_ret
start_to_end_positions
```

Out[11]:

	XLK	XLP	XLF
2000-01-04	0.316438	0.323956	0.318765
2000-01-05	0.316619	0.328791	0.318910
2000-01-06	0.301407	0.335824	0.330128
2000-01-07	0.306638	0.357216	0.335519
2000-01-10	0.318302	0.349158	0.329691
2000-01-11	0.309283	0.349158	0.324155
2000-01-12	0.307961	0.350916	0.330565
2000-01-13	0.310786	0.352821	0.340618
2000-01-14	0.317941	0.353993	0.348485

	XLK	XLP	XLF
2000-01-18	0.319625	0.347985	0.336976
2000-01-19	0.320166	0.350476	0.335810
2000-01-20	0.322030	0.345641	0.329691
2000-01-21	0.322451	0.341832	0.325466
2000-01-24	0.311929	0.332015	0.319639
2000-01-25	0.318543	0.327912	0.322990
2000-01-26	0.308141	0.330842	0.335082
2000-01-27	0.305135	0.336996	0.338724
2000-01-28	0.294252	0.334212	0.326923
2000-01-31	0.303992	0.340366	0.335956

Total portfolio value during the current holding period:

In [12]:

```
start_to_end_positions.sum('columns')
```

Out[12]:

```
2000-01-04    0.959159
2000-01-05    0.964320
2000-01-06    0.967359
2000-01-07    0.999373
2000-01-10    0.997151
2000-01-11    0.982596
2000-01-12    0.989442
2000-01-13    1.004225
2000-01-14    1.020419
2000-01-18    1.004586
2000-01-19    1.006452
2000-01-20    0.997362
2000-01-21    0.989748
2000-01-24    0.963582
2000-01-25    0.969444
2000-01-26    0.974065
2000-01-27    0.980855
2000-01-28    0.955388
2000-01-31    0.980314
dtype: float64
```

Now append these values to the previous portfolio value:

In [13]:

```
portfolio_value = portfolio_value.append( start_to_end_positions.sum('columns')
portfolio_value
```

Out[13]:

```
2000-01-03    1.000000
2000-01-04    0.959159
2000-01-05    0.964320
2000-01-06    0.967359
2000-01-07    0.999373
2000-01-10    0.997151
2000-01-11    0.982596
2000-01-12    0.989442
2000-01-13    1.004225
2000-01-14    1.020419
2000-01-18    1.004586
```

```
2000-01-19    1.006452
2000-01-20    0.997362
2000-01-21    0.989748
2000-01-24    0.963582
2000-01-25    0.969444
2000-01-26    0.974065
2000-01-27    0.980855
2000-01-28    0.955388
2000-01-31    0.980314
dtype: float64
```

And now repeat this procedure for the next holding period:

In [14]:

```
start_date = rebalance_dates[1]
end_date   = rebalance_dates[2]

print('start:', start_date, 'end:', end_date)
```

```
start: 2000-01-31 00:00:00 end: 2000-02-29 00:00:00
```

In [15]:

```
cum_ret = RET[start_date:end_date][1:].add(1).cumprod()
```

Previous portfolio value:

In [16]:

```
portfolio_value.iloc[-1]
```

Out[16]:

```
0.9803143153152252
```

Previous positions:

In [17]:

```
start_to_end_positions[-1:]
```

Out[17]:

	XLK	XLP	XLF
2000-01-31	0.303992	0.340366	0.335956

Now we rebalance these positions to the new positions:

In [18]:

```
new_positions = portfolio_value.iloc[-1] * weights  # dollars invested at start
new_positions
```

Out[18]:

```
XLK    0.326771
XLP    0.326771
XLF    0.326771
dtype: float64
```

In [19]:

```
start_to_end_positions = new_positions  * cum_ret
start_to_end_positions
```

Out[19]:

	XLK	XLP	XLF
2000-02-01	0.333687	0.323536	0.328614
2000-02-02	0.336078	0.324239	0.325213
2000-02-03	0.345773	0.319738	0.323229

	XLK	XLP	XLF
2000-02-04	0.349780	0.321285	0.319970
2000-02-07	0.353852	0.319175	0.316569
2000-02-08	0.357083	0.323114	0.320111
2000-02-09	0.352624	0.320301	0.312601
2000-02-10	0.361026	0.309469	0.303957
2000-02-11	0.351460	0.303561	0.304665
2000-02-14	0.351654	0.305109	0.307358
2000-02-15	0.352624	0.311861	0.307783
2000-02-16	0.350814	0.307500	0.301406
2000-02-17	0.358699	0.302858	0.297580
2000-02-18	0.346613	0.299341	0.286952
2000-02-22	0.345773	0.304265	0.289645
2000-02-23	0.356308	0.297653	0.288511
2000-02-24	0.360702	0.291464	0.282560
2000-02-25	0.357277	0.284149	0.280717
2000-02-28	0.353076	0.289635	0.289786
2000-02-29	0.361155	0.288650	0.291770

Sum the positions and append them to the portfolio value:

In [20]:

```
portfolio_value = portfolio_value.append(start_to_end_positions.sum('columns'))
portfolio_value
```

Out[20]:

```
2000-01-03    1.000000
2000-01-04    0.959159
2000-01-05    0.964320
2000-01-06    0.967359
2000-01-07    0.999373
2000-01-10    0.997151
2000-01-11    0.982596
2000-01-12    0.989442
2000-01-13    1.004225
2000-01-14    1.020419
2000-01-18    1.004586
2000-01-19    1.006452
2000-01-20    0.997362
2000-01-21    0.989748
2000-01-24    0.963582
2000-01-25    0.969444
2000-01-26    0.974065
2000-01-27    0.980855
2000-01-28    0.955388
2000-01-31    0.980314
2000-02-01    0.985837
2000-02-02    0.985530
2000-02-03    0.988740
2000-02-04    0.991035
```

```
2000-02-07    0.989596
2000-02-08    1.000308
2000-02-09    0.985525
2000-02-10    0.974452
2000-02-11    0.959687
2000-02-14    0.964121
2000-02-15    0.972267
2000-02-16    0.959720
2000-02-17    0.959137
2000-02-18    0.932907
2000-02-22    0.939682
2000-02-23    0.942472
2000-02-24    0.934726
2000-02-25    0.922143
2000-02-28    0.932498
2000-02-29    0.941576
dtype: float64
```

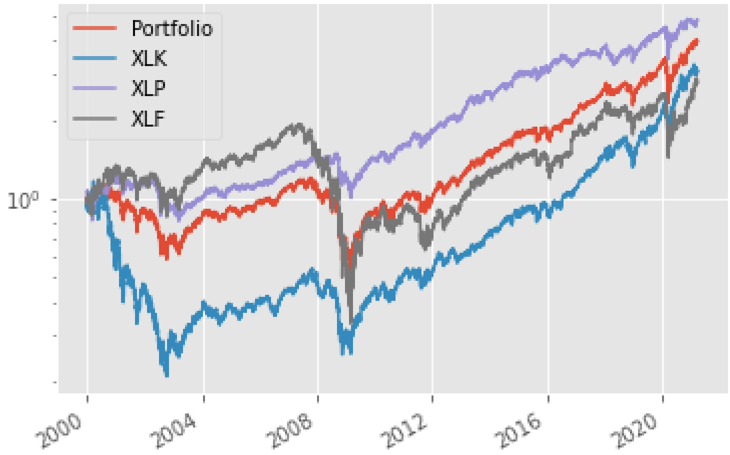
And now loop over all rebalance dates:

In [21]:

```
def run_backtest(frequency):
    rebalance_dates = get_rebalance_dates(frequency)
    weights         = pd.Series({'XLK':1/3, 'XLP':1/3, 'XLF':1/3})
    portfolio_value = pd.Series(1,index=[rebalance_dates[0]])

    for i in range(len(rebalance_dates)-1):
        start_date = rebalance_dates[i]
        end_date   = rebalance_dates[i+1]

        cum_ret = RET[start_date:end_date][1:].add(1).cumprod()

        new_positions          = portfolio_value.iloc[-1] * weights
        start_to_end_positions = new_positions  * cum_ret

        portfolio_value = portfolio_value.append(start_to_end_positions.sum('col

    return portfolio_value


value = run_backtest('month')

value.to_frame('Portfolio').join(RET.add(1).cumprod()).plot(logy=True)
```

Out[21]:

```
<AxesSubplot:>
```

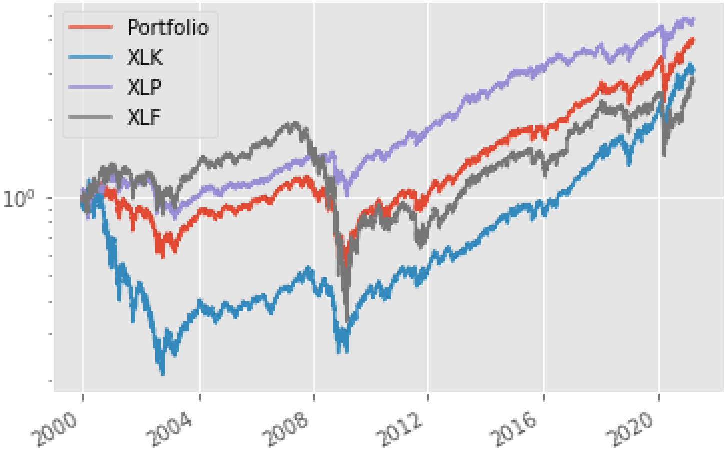


Portfolio turnover

In [22]:

```
def run_backtest(frequency):
    rebalance_dates = get_rebalance_dates(frequency)
    weights          = pd.Series({'XLK':1/3, 'XLP':1/3, 'XLF':1/3})
    portfolio_value = pd.Series(1,                        index=[rebalance_dates
    trades           = pd.DataFrame(columns=weights.index, index=[rebalance_dates
    previous_positions = weights

    for i in range(len(rebalance_dates)-1):
        start_date = rebalance_dates[i]
        end_date   = rebalance_dates[i+1]

        cum_ret = RET[start_date:end_date][1:].add(1).cumprod()

        new_positions           = portfolio_value.iloc[-1] * weights
        start_to_end_positions = new_positions  * cum_ret

        portfolio_value = portfolio_value.append(start_to_end_positions.sum('col

        trades.loc[start_date]  = new_positions - previous_positions
        previous_positions      = start_to_end_positions.iloc[-1]      # Previou

    return portfolio_value, trades



value, trades = run_backtest('month')

value.to_frame('Portfolio').join(RET.add(1).cumprod()).plot(logy=True)
```

Out[22]: <AxesSubplot:>



Daily trades:

In [23]:

```
trades
```

Out[23]:

	XLK	XLP	XLF
2000-01-03	0.0	0.0	0.0
2000-01-31	0.022779	-0.013595	-0.009184

	XLK	XLP	XLF
2000-02-29	-0.047296	0.025208	0.022088
2000-03-31	0.005545	0.020073	-0.025618
2000-04-28	0.028332	-0.021651	-0.006681
...	...	...	...
2020-10-30	0.023382	-0.000506	-0.022876
2020-11-30	0.005614	0.048167	-0.053781
2020-12-31	-0.012661	0.034533	-0.021872
2021-01-29	-0.021618	0.031041	-0.009423
2021-02-26	0.03156	0.063738	-0.095298

255 rows × 3 columns

Turnover example:

In [119…

```
# Before rebalancing:
position_A = 80
position_B = 40
position_C = 30

# After rebalancing:
position_A = 50
position_B = 50
position_C = 50
```

Total dollar amount traded:

In [24]:

```
total_trade = abs(-30) + 10 + 20
total_trade
```

Out[24]:

```
60
```

Turnover:

In [25]:

```
total_trade/2
```

Out[25]:

```
30.0
```

We measure turnover as $30, since 30 is reallocated from stock A to stock B and C. (Every dollar we reallocate results in 2 dollars trade, since it involves a buy and a sell).

Daily portfolio turnover (total dollar amount realloacted):

In [26]:

```
turnover = trades.abs().sum('columns').div(2)
turnover
```

Out[26]:

```
2000-01-03    0.000000
2000-01-31    0.022779
```

```
2000-02-29    0.047296
2000-03-31    0.025618
2000-04-28    0.028332
                ...
2020-10-30    0.023382
2020-11-30    0.053781
2020-12-31    0.034533
2021-01-29    0.031041
2021-02-26    0.095298
Length: 255, dtype: float64
```

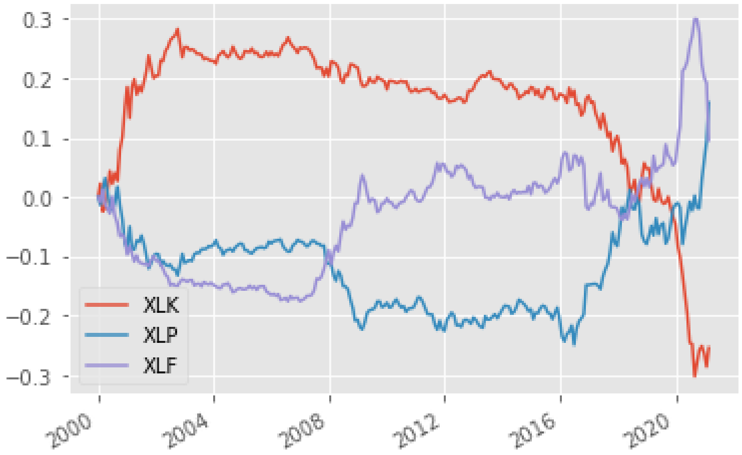
Cumulative trades:

In [27]:

```
trades.cumsum().plot()
```

Out[27]:

```
<AxesSubplot:>
```



This graph shows the total (cumulative) dollars that we put into or take out of each asset.

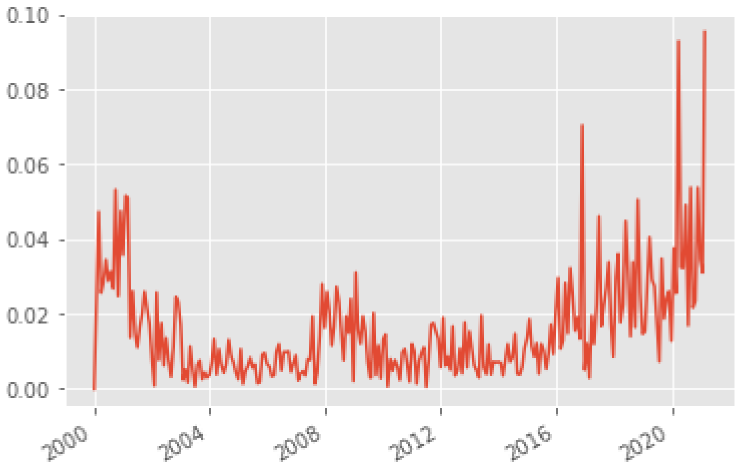
Plot the turnover:

In [28]:

```
turnover.plot()
```

Out[28]:

```
<AxesSubplot:>
```

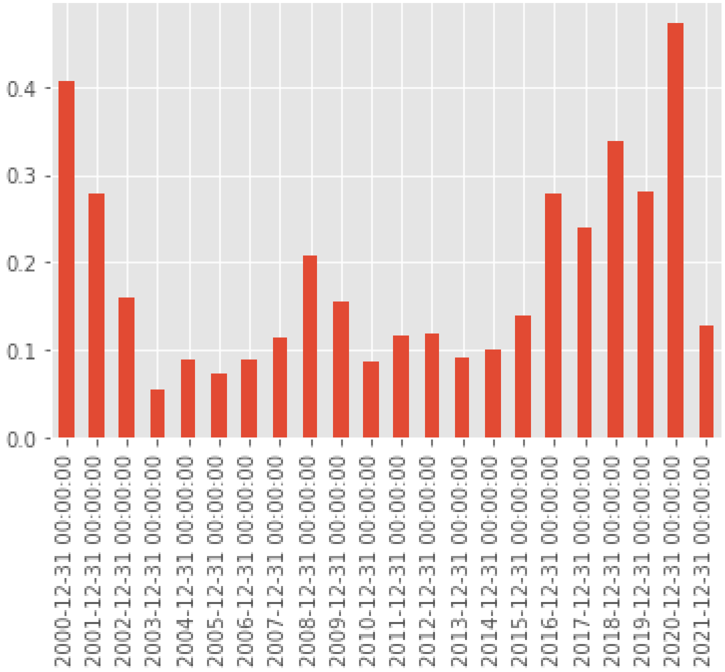


Total turnover per year:

In [29]:

```
turnover.resample('A').sum().plot.bar()
```

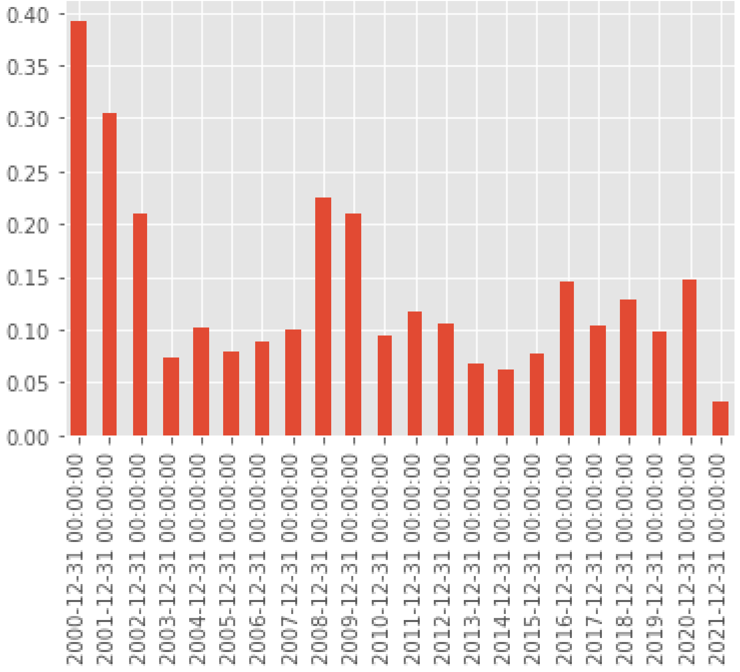
```
Out[29]: <AxesSubplot:>
```



We calculate the **turnover ratio** as the total annual turnover divided by the average annual portfolio value:

```
In [30]: turnover.resample('A').sum().div( value.resample('A').mean() ).plot.bar()
```

```
Out[30]: <AxesSubplot:>
```


0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
2000-12-31 00:00:00
2001-12-31 00:00:00
2002-12-31 00:00:00
2003-12-31 00:00:00
2004-12-31 00:00:00
2005-12-31 00:00:00
2006-12-31 00:00:00
2007-12-31 00:00:00
2008-12-31 00:00:00
2009-12-31 00:00:00
2010-12-31 00:00:00
2011-12-31 00:00:00
2012-12-31 00:00:00
2013-12-31 00:00:00
2014-12-31 00:00:00
2015-12-31 00:00:00
2016-12-31 00:00:00
2017-12-31 00:00:00
2018-12-31 00:00:00
2019-12-31 00:00:00
2020-12-31 00:00:00
2021-12-31 00:00:00