total_momemtum_spillover_ver2

October 1, 2018

1 Total Mementum Spillover ver.2

```
In [1]: import pandas as pd
        import numpy as np
        from datetime import datetime
        import math
        import matplotlib.pyplot as plt
        from matplotlib import style
```

2 Functions

2.1 Ranking Equity Return function

```
In [2]: def rank_port_decile_t(equity_data , start_date, end_date):

            start = datetime.strptime(start_date, '%Y-%m-%d')
            end = datetime.strptime(end_date, '%Y-%m-%d')

            equity_data = equity_data[(equity_data.date >= start) & (equity_data.date <= end)]
            equity_data = equity_data.drop_duplicates()
            try:
                equity_data = equity_data[equity_data.RETX != 'C']
            except:
                print('There is no C')
            equity_data.RETX = equity_data.RETX.astype('float64')
            equity_data.RET = equity_data.RET.astype('float64')

            equity_group_mean_ret = equity_data.groupby('TICKER').agg({'RETX': 'mean'})
            equity_group_mean_ret_sort = equity_group_mean_ret.sort_values(by=['RETX'])
            equity_group_mean_ret_sort['GROUP'] = 0

            frac = math.floor(equity_group_mean_ret_sort.shape[0]/10)
            num_line = equity_group_mean_ret_sort.shape[0]
            left = num_line%frac

            group_num = [frac+1]*(left)+[frac]*(10-left)
```

```
    group = ['P10', 'P09', 'P08', 'P07', 'P06', 'P05', 'P04', 'P03', 'P02', 'P01']

    i, j = 0, group_num[0]
    while j <= num_line:

        equity_group_mean_ret_sort.ix[(j-group_num[i]):j , 'GROUP'] = group[i]
        i +=1
        if(i == 10):
            break
        j = j + group_num[i]

    return(equity_group_mean_ret_sort)
```

2.2 Generating Ranking Summary table function

```
In [3]: def rank_total_table(rank_port):

    rank_port_sort = rank_port
    rank_port_sort['TICKER'] = rank_port_sort.index
    rank_port_sort = rank_port.sort_values(by=['GROUP','TICKER'])
    rank_port_sort_sum = rank_port_sort.groupby('GROUP').agg({'TICKER':'count', 'RETX'
    rank_port_sort_sum.loc['Total'] = rank_port_sort_sum.sum()

    fig, ax = plt.subplots()
    fig.set_figheight(5)
    fig.set_figwidth(7)
    plt.subplots_adjust(left=0.5, top=0.8)
    ax.set_title('Ranking Equity Returns \n From Winner to Loser Portfolios', fontsize=
    ax.xaxis.set_visible(False)
    ax.yaxis.set_visible(False)
    ax.axis('off')
    ax.axis('tight')

    data = rank_port_sort_sum.round(6).values
    columns = ['Company numbers', 'Average returns']
    rows = rank_port_sort_sum.index

    rcolors = plt.cm.BuPu(np.linspace(0, 0.5, len(rows)))
    ccolors = rcolors[::-1]
    clcolors = []
    for i in range(0, 10):
        clcolors.append(["w","w"])
    #clcolors.append(["#92a4cd","#92a4cd"])
    clcolors.append(["#8c95c6","#8c95c6"])

    plt.table(cellText = data, cellColours=clcolors,
                          rowLabels=rows,
```

```
                                colLabels=columns,rowColours=rcolors, colColours=ccolors, loc
        fig.tight_layout()


        #return(fig)
```

2.3 Running Total Momentum Spillover Strategy function

```
In [4]: def tmomentum_strategy(bond_data, rank_port, start_month, end_month, TMT = 2):

        bond_data = pd.merge(bond_data, rank_port[['GROUP']], how = 'left', left_on = 'comp

        bond_data = bond_data[bond_data.TMT >= TMT]

        #equity_group_mean_ret_sort[equity_group_mean_ret_sort.index == "TRP"]

        bond_average_month = bond_data.groupby(['month', 'company_symbol', 'GROUP']).agg({
        bond_month = pd.DataFrame(columns = ['month', 'ticker', 'port_group', 'price', 'yie
        bond_month['month'] = [i[0] for i in bond_average_month.index]
        bond_month['ticker'] = [i[1] for i in bond_average_month.index]
        bond_month['port_group'] = [i[2] for i in bond_average_month.index]
        bond_month['price'] = bond_average_month['PRICE_EOM'].values
        bond_month['yield'] = bond_average_month['YIELD'].values
        bond_month['avg_rating'] = bond_average_month['RATING_NUM'].values
        bond_month['bond_num'] = bond_average_month['CUSIP'].values

        bond_hold = bond_month[(bond_month.month >= start_month) & (bond_month.month <= end

        bond_sum_price = bond_hold.groupby('port_group').agg({'price':'sum'})
        bond_hold_value = pd.merge(bond_hold, bond_sum_price, how = 'left', left_on = 'port
        bond_hold_value.columns = ['month', 'ticker', 'port_group', 'price', 'yield', 'avg_
        bond_hold_value['port_weight'] = bond_hold_value['price'].divide(bond_hold_value['p
        bond_hold_value['port_weighted_bond'] = bond_hold_value['yield'].mul(bond_hold_valu

        bond_hold_valueW = bond_hold_value.groupby('port_group').agg({'port_weighted_bond'
        bond_hold_valueW.loc['P01-P10'] = bond_hold_valueW.ix[0,0] - bond_hold_valueW.ix[9
        bond_hold_valueW.loc['P01-P05'] = bond_hold_valueW.ix[0,0] - bond_hold_valueW.ix[4

        bond_hold_equalW = bond_hold.groupby('port_group').agg({'yield':'mean', 'price':'me
        bond_hold_equalW.loc['P01-P10'] = bond_hold_equalW.ix[0,0] - bond_hold_equalW.ix[9
        bond_hold_equalW.loc['P01-P05'] = bond_hold_equalW.ix[0,0] - bond_hold_equalW.ix[4

        result = bond_hold_valueW.merge(bond_hold_equalW, left_index=True, right_index=True
        text1 = 'value_wight'+'('+str(start_month)+','+str(end_month)+')'
        text2 = 'equal_weight'+'('+str(start_month)+','+str(end_month)+')'
        result.columns = [text1, text2, 'avg_price', 'avg_rating', 'bond_num', 'com_num']
        result.ix[10:,2:] = 0
```

```
            return(result)
```

2.4 Generating Total Momentum Spillover Stategy Summary table function

```
In [5]: def tm_performance_table(perf):
            fig, ax = plt.subplots()
            fig.set_figheight(5)
            fig.set_figwidth(10)
            plt.subplots_adjust(left=0.5, top=0.8)
            ax.set_title('Total Momentum Spillover Performance Table', fontsize=15, weight='bo
            ax.xaxis.set_visible(False)
            ax.yaxis.set_visible(False)
            ax.axis('off')
            ax.axis('tight')

            data = perf.round(4).values
            columns = perf.columns
            rows = perf.index

            rcolors = plt.cm.BuPu(np.linspace(0, 0.5, len(rows)))
            ccolors = rcolors[::-1]
            clcolors = []
            for i in range(0, 10):
                clcolors.append(["w","w","w","w","w", "w"])
            clcolors.append(["#92a4cd","#92a4cd","#92a4cd","#92a4cd","#92a4cd", "#92a4cd"])
            clcolors.append(["#8c95c6","#8c95c6","#8c95c6","#8c95c6","#8c95c6", "#8c95c6"])

            plt.table(cellText = data, cellColours=clcolors,
                                  rowLabels=rows,
                                  colLabels=columns,rowColours=rcolors, colColours=ccolors, lo
            fig.tight_layout()

            #return(fig)
```

2.5 Total Momentum Spillover Strategy Performance Visualization function

```
In [6]: def performance_plot(perf, holding, start, end, tmt):

            style.use('seaborn')

            fig = plt.figure(figsize=(18, 15))
            fig.suptitle('Total Momentum Spillover Performance', fontsize=20, fontweight='bold

            plt.figtext(0.5,0.95, "Average {0}-month of equity returns \n Holding bond portfol

            ax1 = plt.subplot2grid((30, 22), (0, 0), rowspan=10, colspan=13)
            ax1.plot(perf[[0]].iloc[0:10], label="{0}".format(perf.columns[0]))
            ax1.annotate("{0:0.4f}".format(perf[[0]].values[0][0]), xy=(0, perf[[0]].values[0]
```

```
ax1.annotate("{0:0.4f}".format(perf[[0]].values[-3][0]), xy=(1, perf[[0]].values[-3

ax1.plot(perf[[1]].iloc[0:10], label="{0}".format(perf.columns[1]))
ax1.annotate("{0:0.4f}".format(perf[[1]].values[0][0]), xy=(0, perf[[1]].values[0]
ax1.annotate("{0:0.4f}".format(perf[[1]].values[-3][0]), xy=(1, perf[[1]].values[-3
ax1.set_title('Total momemtum Spillover Curves from Winner to Loser Portfolios', fo
ax1.legend()

ax2 = plt.subplot2grid((30, 22), (0, 15), rowspan=10, colspan=6)

index = np.arange(2)
bar_width = 0.35
opacity = 0.8

rect1 = ax2.bar(index - bar_width/2, [i[0] for i in perf[[0]].values[10:12]], bar_w
for rect in rect1:
    height1 = rect.get_height()
    ax2.text(rect.get_x() + rect.get_width()/2., 1.05*height1,
                "{0:0.4f}".format(height1),
                ha='center', va='bottom')

rect2 = ax2.bar(index + bar_width/2, [i[0] for i in perf[[1]].values[10:12]], bar_w
for rect in rect2:
    height2 = rect.get_height()
    ax2.text(rect.get_x() + rect.get_width()/2., 1.01*height2,
                "{0:0.4f}".format(height2),
                ha='center', va='bottom')

ax2.set_xticks(index)
ax2.set_xticklabels((perf.index[-2], perf.index[-1]))
ax2.set_title('Long-short Performance', fontsize=15, weight='bold')
ax2.legend()

ax3 = plt.subplot2grid((30, 22), (12, 0), rowspan=6, colspan=10)
ax3.plot(perf[[2]].iloc[0:10], 'o-',label="Avergae market price", color='turquoise
ax3.set_title('Average Bond Price from Winner to Loser Portfolios', fontsize=15, we
for i in range(0,10):
    ax3.text(i,1.001*[j for j in perf[[2]].iloc[i]][0], '{0:.2f}'.format([j for j i
ax3.legend()

ax4 = plt.subplot2grid((30, 22), (12, 11), rowspan=6, colspan=10)
ax4.plot(perf[[3]].iloc[0:10], 'o-', label="Avergae credit rating", color='turquois
ax4.set_title('Average Credit Rating from Winner to Loser Portfolios', fontsize=15
for i in range(0,10):
    ax4.text(i,1.001*[j for j in perf[[3]].iloc[i]][0], '{0:.2f}'.format([j for j i
ax4.legend()

ax5 = plt.subplot2grid((30, 22), (20, 0), rowspan=6, colspan=10)
```

```
    ax5.plot(perf[[4]].iloc[0:10], 'o-', label="Number of bonds", color='turquoise')
    ax5.set_title('Total Bond Numbers from Winner to Loser Portfolios', fontsize=15, w
    for i in range(0,10):
        ax5.text(i,1.001*[j for j in perf[[4]].iloc[i]][0], '{0:.2f}'.format([j for j
    ax4.legend()
    ax5.legend()

    ax6 = plt.subplot2grid((30, 22), (20, 11), rowspan=6, colspan=10)
    rect3 = ax6.bar(perf.index[0:10].values, [i[0] for i in perf[[5]].values[0:10]], ba
    for rect in rect3:
        height3 = rect.get_height()
        ax6.text(rect.get_x() + rect.get_width()/2., 1.01*height3,
                    "{0}".format(int(height3)),
                    ha='center', va='bottom')

    ax6.set_title('Total Company Numbers from Winner to Loser Portfolios', fontsize=15

    #return(fig)
```

3 Backtesting

3.1 Import equity data step

```
In [7]: equity_data = pd.read_csv('CRSP 2013 - BBB.csv')
        equity_data.date = pd.to_datetime(equity_data.date, format = '%Y%m%d')
```

3.2 Import bond data step

```
In [8]: bond_data = pd.read_csv('WRDS bond return 2013 - BBB.csv')
        bond_data.DATE = pd.to_datetime(bond_data.DATE, format = '%d-%b-%y')

        bond_data['month'] = [i.month for i in bond_data.DATE]
        bond_data.YIELD = bond_data.YIELD.str.replace('%', '').astype('float').divide(100.0)
```

3.3 Ranking equity returns step

```
In [9]: efrom = '2012-06-01'
        eend = '2012-12-31'
        rank_port = rank_port_decile_t(equity_data, efrom, eend)
        holding = datetime.strptime(eend, '%Y-%m-%d').month - datetime.strptime(efrom, '%Y-%m-

In [10]: rank_total_table(rank_port)
```

Ranking Equity Returns
From Winner to Loser Portfolios

	Company numbers	Average returns
P01	49.0	0.062519
P02	49.0	0.036055
P03	49.0	0.027007
P04	49.0	0.021286
P05	49.0	0.01659
P06	49.0	0.01098
P07	50.0	0.006509
P08	50.0	0.002477
P09	50.0	-0.003332
P10	50.0	-0.017843
Total	494.0	0.162249

3.4 Perform backtesting step

Inputs: hfrom = 'Holding from', hend = 'Holding end', tmt = lower 'Time-to-maturity' that we don't want

```
In [11]: hfrom = 2
         hend = 2
         tmt = 2
         perf = tmomentum_strategy(bond_data, rank_port, hfrom, hend, TMT=tmt)

In [12]: tm_performance_table(perf)
```

Total Momentum Spillover Performance Table

	value_wight(2,2)	equal_weight(2,2)	avg_price	avg_rating	bond_num	com_num
P01	0.0514	0.066	112.0426	9.1554	235.0	36.0
P02	0.0346	0.0346	112.4361	8.9243	169.0	37.0
P03	0.0337	0.0345	112.8028	8.986	181.0	40.0
P04	0.0313	0.0335	110.7737	9.0143	186.0	42.0
P05	0.0321	0.0344	110.5579	9.177	264.0	45.0
P06	0.0294	0.0313	112.1986	8.7961	173.0	43.0
P07	0.0313	0.0325	111.4153	9.0318	286.0	44.0
P08	0.0311	0.0335	112.4836	9.0117	143.0	37.0
P09	0.0328	0.0336	114.0543	8.9779	247.0	38.0
P10	0.0362	0.0379	109.7427	9.0385	237.0	39.0
P01-P10	0.0152	0.0281	0.0	0.0	0.0	0.0
P01-P05	0.0193	0.0315	0.0	0.0	0.0	0.0

In [13]: `performance_plot(perf, holding, hfrom, hend, perf)`