[FE800] Group 4: Phase 1:3 - Total and Residual Momentum Spillover

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

Functions

Ranking Equity Return function

```
In [2]: def rank_port_decile(equity_data , formation_date, num_month, strat_type = 0):

            form_date = datetime.strptime(formation_date, '%Y-%m-%d')
            avg_date =  form_date - relativedelta.relativedelta(months = num_month)
            check_last_day = calendar.monthrange(avg_date.year, avg_date.month)
            avg_date = datetime(avg_date.year, avg_date.month, check_last_day[-1])

            equity_data_ = equity_data[(equity_data.date >= avg_date) & (equity_data.date <= form_date)]
            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')

            #residual
            if strat_type != 0:
                equity_data_.RETX = equity_data_.RETX - equity_data_.RF

            equity_data_.RETX = equity_data_.RETX.add(1) #cumulative

            #equity_group_mean_ret = equity_data_.groupby('TICKER').agg({'RETX': 'mean'})
            equity_group_mean_ret = equity_data_.groupby('TICKER').agg({'RETX': 'prod'}) #cumulative
            equity_group_mean_ret = equity_group_mean_ret.subtract(1) #cumulative
            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)
```

Generating Ranking Summary table function

```
In [3]: def rank_table(rank_port, universe, strat_type = 0):
    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':'mean'})
    rank_port_sort_sum.loc['Total'] = rank_port_sort_sum.sum()

    strat_name = np.where(strat_type == 0, 'Total', 'Residual')

    fig, ax = plt.subplots()
    fig.set_figheight(5)
    fig.set_figwidth(8)
    plt.subplots_adjust(left=0.5, top=0.8)
    ax.set_title('Ranking {0} Equity Returns \n From Winner to Loser Portfolios - {1}'.format(strat_name,
universe), fontsize=15, weight='bold')
    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='center')
    fig.tight_layout()
```

Running Total Momentum Spillover Strategy function

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

    bond_data = pd.merge(bond_data, rank_port[['GROUP']], how = 'left', left_on = 'company_symbol', right_
index=True)
    bond_data = bond_data[bond_data.TMT >= TMT]
    bond_data = bond_data.dropna(subset=['RET_EOM', 'DURATION'])

    bond_average_month = bond_data.groupby(['month', 'company_symbol', 'GROUP']).agg({'PRICE_EOM':'mean',
'RET_EOM':'mean', 'RATING_NUM':'mean', 'DURATION':'mean'})

    bond_month = pd.DataFrame(columns = ['month', 'ticker', 'port_group', 'price', 'return', 'avg_rating',
 'duration'])
    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['return'] = bond_average_month['RET_EOM'].values
    bond_month['avg_rating'] = bond_average_month['RATING_NUM'].values
    bond_month['duration'] = bond_average_month['DURATION'].values
    bond_hold = bond_month[(bond_month.month >= start_month) & (bond_month.month <= end_month)]


    bond_sum_price = bond_hold.groupby(['port_group', 'month']).agg({'price':'sum'})
    bond_sum_price['port_group'] = [i[0] for i in bond_sum_price.index]
    bond_sum_price['month'] = [i[1] for i in bond_sum_price.index]

    bond_hold_value = pd.merge(bond_hold, bond_sum_price, how = 'left', left_on = ['port_group', 'month'],
 right_on=['port_group', 'month'])
    bond_hold_value.columns = ['month', 'ticker', 'port_group', 'price', 'return', 'avg_rating', 'duratio
n','port_total_weight']
    bond_hold_value['port_weight'] = bond_hold_value['price'].divide(bond_hold_value['port_total_weight'])
    bond_hold_value['value_return'] = bond_hold_value['return'].mul(bond_hold_value['port_weight'])
    bond_hold_value['value_price'] = bond_hold_value['price'].mul(bond_hold_value['port_weight'])
    bond_hold_value['value_rating'] = bond_hold_value['avg_rating'].mul(bond_hold_value['port_weight'])
    bond_hold_value['value_duration'] = bond_hold_value['duration'].mul(bond_hold_value['port_weight'])

    # Value portfolio
    # value portfolio
    bond_hold_value_ = bond_hold_value.groupby(['port_group', 'month', 'ticker']).agg({'value_return':'su
m', 'value_price':'sum', 'value_duration':'sum', 'value_rating':'sum'})
    bond_hold_value_['port_group'] = [i[0] for i in bond_hold_value_.index]
    bond_hold_value_['month'] = [i[1] for i in bond_hold_value_.index]
    bond_hold_value_['ticker'] = [i[2] for i in bond_hold_value_.index]
    bond_hold_value_['value_return'] = bond_hold_value_['value_return'].add(1)

    bond_hold_value__ = bond_hold_value_.groupby(['port_group', 'ticker']).agg({'value_return':'prod', 'va
lue_price':lambda x: x.iloc[-1],'value_duration':lambda x: x.iloc[-1], 'value_rating':lambda x: x.iloc[-1
]})
    bond_hold_value__['port_group'] = [i[0] for i in bond_hold_value__.index]
    bond_hold_value__['ticker'] = [i[1] for i in bond_hold_value__.index]
    bond_hold_value__['value_return'] = bond_hold_value__['value_return'].subtract(1)

    bond_hold_valueW = bond_hold_value__.groupby(['port_group']).agg({'value_return':'sum', 'value_price':
'sum','value_duration':'sum', 'value_rating':'sum'})
    bond_hold_valueW.loc['P01-P10'] = bond_hold_valueW.ix[0,0] - bond_hold_valueW.ix[-1,0]
    bond_hold_valueW.loc['P01-P05'] = bond_hold_valueW.ix[0,0] - bond_hold_valueW.ix[4,0]

    #Equal portfolio
    bond_hold_value___ = bond_hold_value
    bond_hold_value___['return'] = bond_hold_value___['return'].add(1)
    bond_hold_equalW_ = bond_hold_value___.groupby(['port_group', 'ticker']).agg({'return':'prod', 'price'
:lambda x: x.iloc[-1], 'avg_rating':lambda x: x.iloc[-1], 'duration': lambda x: x.iloc[-1]})
    bond_hold_equalW_['port_group'] = [i[0] for i in bond_hold_equalW_.index]
    bond_hold_equalW_['ticker'] = [i[1] for i in bond_hold_equalW_.index]
    bond_hold_equalW_['return'] = bond_hold_equalW_['return'].subtract(1)
    bond_hold_equalW = bond_hold_equalW_.groupby('port_group').agg({'return':'mean', 'price':'mean', 'avg_
rating':'mean', 'duration': 'mean', 'ticker': lambda x: x.nunique()})
    bond_hold_equalW.loc['P01-P10'] = bond_hold_equalW.ix[0,0] - bond_hold_equalW.ix[-1,0]
    bond_hold_equalW.loc['P01-P05'] = bond_hold_equalW.ix[0,0] - bond_hold_equalW.ix[4,0]

    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__ = pd.DataFrame(columns = [text1, text2, 'value_price', 'equal_price', 'value_rating', 'equal_
rating', 'value_duration', 'equal_duration', 'com_num'])
    result__[text1] = result_.iloc[:, 0]
    result__[text2] = result_.iloc[:, 4]
    result__['value_price'] = result_.iloc[:, 1]
    result__['equal_price'] = result_.iloc[:, 5]
    result__['value_rating'] = result_.iloc[:, 3]
    result__['equal_rating'] = result_.iloc[:, 6]
    result__['value_duration'] = result_.iloc[:, 2]
    result__['equal_duration'] = result_.iloc[:, 7]
    result__['com_num'] = result_.iloc[:, 8]
    result__.ix[10:,2:] = 0

    result = pd.DataFrame(data = 0, columns = result__.columns, index = ['P01', 'P02', 'P03', 'P04', 'P05'
, 'P06', 'P07', 'P08', 'P09', 'P10'])

    for i in result__.index:
        result.loc[i] = result__.loc[i]

    return(result)
```

Generating Total Momentum Spillover Stategy Summary table function

In [5]:

```
def m_performance_table(perf, universe, strat_type = 0):
    strat_name = np.where(strat_type == 0, 'Total', 'Residual')

    fig, ax = plt.subplots()
    fig.set_figheight(5)
    fig.set_figwidth(14)
    plt.subplots_adjust(left=0.5, top=0.8)
    ax.set_title('{0} Momentum Spillover Performance Table - {1}'.format(strat_name, universe), fontsize=1
5, weight='bold')
    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","w","w","w"])
    clcolors.append(["#92a4cd","#92a4cd","#92a4cd","#92a4cd","#92a4cd","#92a4cd", "#92a4cd", "#92a4cd", "#92a4cd"])
    clcolors.append(["#8c95c6","#8c95c6","#8c95c6","#8c95c6","#8c95c6","#8c95c6", "#8c95c6", "#8c95c6", "#8c95c6"])

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

Total Momentum Spillover Strategy Performance Visualization function

In [6]:

```
def performance_plot(perf, avg_period, start, end, tmt, universe, strat_type = 0):

    style.use('seaborn')

    fig = plt.figure(figsize=(18, 15))
    strat_name = np.where(strat_type == 0, 'Total', 'Residual')
    fig.suptitle('{0} Momentum Spillover Performance - {1}'.format(strat_name, universe), fontsize=20, fontweight='bold')

    plt.figtext(0.5,0.95, "Average {0}-month equity returns \n Holding bond portfolios from {1} to {2} \n Time-to-maturity over {3}".format(avg_period, start, end, tmt), ha="center", va="top", fontsize=14, color="black")

    ax1 = plt.subplot2grid((30, 22), (0, 0), rowspan=10, colspan=13)
    ax1.plot(perf.iloc[0:10, 0], label="{0}".format(perf.columns[0]), color='SteelBlue')
    ax1.annotate("{0:0.5f}".format(perf.iloc[0,0]), xy=(0, perf.iloc[0,0]), xytext=(8, 6), xycoords=('axes fraction', 'data'), textcoords='offset points')
    ax1.annotate("{0:0.5f}".format(perf.iloc[-3,0]), xy=(1, perf.iloc[-3,0]), xytext=(8, 6), xycoords=('axes fraction', 'data'), textcoords='offset points')

    ax1.plot(perf.iloc[0:10, 1], label="{0}".format(perf.columns[1]), color='IndianRed')
    ax1.annotate("{0:0.5f}".format(perf.iloc[0,1]), xy=(0, perf.iloc[0,1]), xytext=(8, 6), xycoords=('axes fraction', 'data'), textcoords='offset points')
    ax1.annotate("{0:0.5f}".format(perf.iloc[-3,1]), xy=(1, perf.iloc[-3,1]), xytext=(8, 6), xycoords=('axes fraction', 'data'), textcoords='offset points')
    ax1.set_title('{0} momemtum Spillover Curves from Winner to Loser Portfolios'.format(strat_name), fontsize=15, weight='bold')
    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, perf.iloc[10:12,0].values, bar_width, color='SkyBlue', label="{0}".format(perf.columns[0]))
    for rect in rect1:
        height1 = rect.get_height()
        ax2.text(rect.get_x() + rect.get_width()/2., 1.01*height1,
                      "{0:0.5f}".format(height1),
                      ha='center', va='bottom')

    rect2 = ax2.bar(index + bar_width/2, perf.iloc[10:12,1].values, bar_width, color='IndianRed', alpha=opacity, label="{0}".format(perf.columns[1]))
    for rect in rect2:
        height2 = rect.get_height()
        ax2.text(rect.get_x() + rect.get_width()/2., 1.01*height2,
                      "{0:0.5f}".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.iloc[0:10, 2], 'o-',label="Value price", color='orchid')
    ax3.plot(perf.iloc[0:10, 3], 'o-',label="Equal price", color='turquoise')
    ax3.set_title('Bond Prices from Winner to Loser Portfolios', fontsize=15, weight='bold')
    for i in range(0,10):
        ax3.text(i,1.001*perf.iloc[i, 2], '{0:.2f}'.format(perf.iloc[i, 2]), ha='center', va='bottom')
        ax3.text(i,1.001*perf.iloc[i, 3], '{0:.2f}'.format(perf.iloc[i, 3]), ha='center', va='bottom')
    ax3.legend()

    ax4 = plt.subplot2grid((30, 22), (12, 11), rowspan=6, colspan=10)
    ax4.plot(perf.iloc[0:10, 4], 'o-', label="Value credit rating", color='orchid')
    ax4.plot(perf.iloc[0:10, 5], 'o-', label="Equal credit rating", color='turquoise')
    ax4.set_title('Credit Rating from Winner to Loser Portfolios', fontsize=15, weight='bold')
    for i in range(0,10):
        ax4.text(i,1.001*perf.iloc[i, 4], '{0:.2f}'.format(perf.iloc[i, 4]), ha='center', va='bottom')
        ax4.text(i,1.001*perf.iloc[i, 5], '{0:.2f}'.format(perf.iloc[i, 5]), ha='center', va='bottom')
    ax4.legend()

    ax5 = plt.subplot2grid((30, 22), (20, 0), rowspan=6, colspan=10)
    ax5.plot(perf.iloc[0:10, 6], 'o-', label="Value durations", color='orchid')
    ax5.plot(perf.iloc[0:10, 7], 'o-', label="Equal durations", color='turquoise')
    ax5.set_title('Durations from Winner to Loser Portfolios', fontsize=15, weight='bold')
    for i in range(0,10):
        ax5.text(i,1.001*perf.iloc[i, 6], '{0:.2f}'.format(perf.iloc[i, 6]), ha='center', va='bottom')
        ax5.text(i,1.001*perf.iloc[i, 7], '{0:.2f}'.format(perf.iloc[i, 7]), ha='center', va='bottom')
    ax5.legend()

    ax6 = plt.subplot2grid((30, 22), (20, 11), rowspan=6, colspan=10)
    rect3 = ax6.bar(perf.index[0:10].values, perf.iloc[0:10, 8], bar_width, color='deepskyblue', label="Nu
mber of companies")
    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, weight='bold')
```

Performance comparison function

In [7]:

```
def comparison_performance(equity_data, bond_data, holding_range, formation_date, equity_range, TMT=2, str
at_type = 0):
    rank1_port = rank_port_decile(equity_data, formation_date, equity_range[0], strat_type=strat_type)
    rank2_port = rank_port_decile(equity_data, formation_date, equity_range[1], strat_type=strat_type)
    rank3_port = rank_port_decile(equity_data, formation_date, equity_range[2], strat_type=strat_type)

    avg_period = (equity_range[0], equity_range[1], equity_range[2])

    perf1_1 = momentum_strategy(bond_data, rank1_port, holding_range[0][0], holding_range[0][1], TMT=TMT)
    perf1_2 = momentum_strategy(bond_data, rank1_port, holding_range[1][0], holding_range[1][1], TMT=TMT)
    perf1_3 = momentum_strategy(bond_data, rank1_port, holding_range[2][0], holding_range[2][1], TMT=TMT)
    perf1_4 = momentum_strategy(bond_data, rank1_port, holding_range[3][0], holding_range[3][1], TMT=TMT)
    perf11 = [perf1_1.iloc[-2, 0], perf1_1.iloc[-2, 1], perf1_2.iloc[-2, 0], perf1_2.iloc[-2, 1], perf1_3.
iloc[-2, 0], perf1_3.iloc[-2, 1], perf1_4.iloc[-2, 0], perf1_4.iloc[-2, 1]]
    perf12 = [perf1_1.iloc[-1, 0], perf1_1.iloc[-1, 1], perf1_2.iloc[-1, 0], perf1_2.iloc[-1, 1], perf1_3.
iloc[-1, 0], perf1_3.iloc[-1, 1], perf1_4.iloc[-1, 0], perf1_4.iloc[-1, 1]]

    perf2_1 = momentum_strategy(bond_data, rank2_port, holding_range[0][0], holding_range[0][1], TMT=TMT)
    perf2_2 = momentum_strategy(bond_data, rank2_port, holding_range[1][0], holding_range[1][1], TMT=TMT)
    perf2_3 = momentum_strategy(bond_data, rank2_port, holding_range[2][0], holding_range[2][1], TMT=TMT)
    perf2_4 = momentum_strategy(bond_data, rank2_port, holding_range[3][0], holding_range[3][1], TMT=TMT)
    perf21 = [perf2_1.iloc[-2, 0], perf2_1.iloc[-2, 1], perf2_2.iloc[-2, 0], perf2_2.iloc[-2, 1], perf2_3.
iloc[-2, 0], perf2_3.iloc[-2, 1], perf2_4.iloc[-2, 0], perf2_4.iloc[-2, 1]]
    perf22 = [perf2_1.iloc[-1, 0], perf2_1.iloc[-1, 1], perf2_2.iloc[-1, 0], perf2_2.iloc[-1, 1], perf2_3.
iloc[-1, 0], perf2_3.iloc[-1, 1], perf2_4.iloc[-1, 0], perf2_4.iloc[-1, 1]]

    perf3_1 = momentum_strategy(bond_data, rank3_port, holding_range[0][0], holding_range[0][1], TMT=TMT)
    perf3_2 = momentum_strategy(bond_data, rank3_port, holding_range[1][0], holding_range[1][1], TMT=TMT)
    perf3_3 = momentum_strategy(bond_data, rank3_port, holding_range[2][0], holding_range[2][1], TMT=TMT)
    perf3_4 = momentum_strategy(bond_data, rank3_port, holding_range[3][0], holding_range[3][1], TMT=TMT)
    perf31 = [perf3_1.iloc[-2, 0], perf3_1.iloc[-2, 1], perf3_2.iloc[-2, 0], perf3_2.iloc[-2, 1], perf3_3.
iloc[-2, 0], perf3_3.iloc[-2, 1], perf3_4.iloc[-2, 0], perf3_4.iloc[-2, 1]]
    perf32 = [perf3_1.iloc[-1, 0], perf3_1.iloc[-1, 1], perf3_2.iloc[-1, 0], perf3_2.iloc[-1, 1], perf3_3.
iloc[-1, 0], perf3_3.iloc[-1, 1], perf3_4.iloc[-1, 0], perf3_4.iloc[-1, 1]]
```

```
    perf = [perf11, perf12, perf21, perf22, perf31, perf32]

    table_index = []
    for i in avg_period:
        for j in ['P01-P10', 'P01-P05']:
            table_index.append('Average {0}-month return:{1}'.format(i, j))

    column_name = ['Value {0}'.format(holding_range[0]), 'Equal {0}'.format(holding_range[0]), 'Value {0}'
.format(holding_range[1]) , 'Equal {0}'.format(holding_range[1]), 'Value {0}'.format(holding_range[2]), 'E
qual {0}'.format(holding_range[2]), 'Value {0}'.format(holding_range[3]), 'Equal {0}'.format(holding_range
[3])]
    comparison_table = pd.DataFrame(data = perf,index = table_index, columns = column_name)
    return(comparison_table)
```

Generating performance comparison table function

```
In [8]: def comparison_table(com_perf, universe, strat_type = 0):

            fig, ax = plt.subplots()
            fig.set_figheight(5)
            fig.set_figwidth(12)
            plt.subplots_adjust(left=0.5, top=0.8)
            strat_name = np.where(strat_type == 0, 'Total', 'Residual')
            ax.set_title('{0} Momentum Spillover Comparison Performance Table - {1}'.format(strat_name, universe),
         fontsize=15, weight='bold')
            ax.xaxis.set_visible(False)
            ax.yaxis.set_visible(False)
            ax.axis('off')
            ax.axis('tight')

            data = com_perf.round(5).values
            columns = com_perf.columns
            rows = com_perf.index

            rcolors = plt.cm.BuPu(np.linspace(0, 0.5, len(rows)))
            ccolors = plt.cm.BuPu(np.linspace(0, 0.5, len(columns)))[::-1]
            clcolors = []
            for i in range(0, 6):
                maxval  = max(data[i])
                maxidx = [index for index, val in enumerate(data[i]) if val == maxval]
                clist = ["w","w","w","w","w","w","w","w"]
                clist[maxidx[0]] = "tomato"
                clcolors.append(clist)

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

Yearly holding comparision function

```
In [9]: def comparison_holding(equity_data, bond_data, form_date, num_month, strat_type = 0, TMT = 2):

            start_month = 1
            end_month = 12

            rank_port = rank_port_decile(equity_data, form_date, num_month, strat_type)

            bond_data = pd.merge(bond_data, rank_port[['GROUP']], how = 'left', left_on = 'company_symbol', right_
        index=True)
            bond_data = bond_data[bond_data.TMT >= TMT]
            bond_data = bond_data.dropna(subset=['RET_EOM', 'DURATION'])

            bond_average_month = bond_data.groupby(['month', 'company_symbol', 'GROUP']).agg({'PRICE_EOM':'mean',
        'RET_EOM':'mean', 'RATING_NUM':'mean', 'DURATION':'mean'})

            bond_month = pd.DataFrame(columns = ['month', 'ticker', 'port_group', 'price', 'return', 'avg_rating',
         'duration'])
            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['port_group'] = [i[2] for i in bond_average_month.index]
    bond_month['price'] = bond_average_month['PRICE_EOM'].values
    bond_month['return'] = bond_average_month['RET_EOM'].values
    bond_month['avg_rating'] = bond_average_month['RATING_NUM'].values
    bond_month['duration'] = bond_average_month['DURATION'].values
    bond_hold = bond_month[(bond_month.month >= start_month) & (bond_month.month <= end_month)]


    bond_sum_price = bond_hold.groupby(['port_group', 'month']).agg({'price':'sum'})
    bond_sum_price['port_group'] = [i[0] for i in bond_sum_price.index]
    bond_sum_price['month'] = [i[1] for i in bond_sum_price.index]

    bond_hold_value = pd.merge(bond_hold, bond_sum_price, how = 'left', left_on = ['port_group', 'month'],
 right_on=['port_group', 'month'])
    bond_hold_value.columns = ['month', 'ticker', 'port_group', 'price', 'return', 'avg_rating', 'duratio
n','port_total_weight']
    bond_hold_value['port_weight'] = bond_hold_value['price'].divide(bond_hold_value['port_total_weight'])
    bond_hold_value['value_return'] = bond_hold_value['return'].mul(bond_hold_value['port_weight'])
    bond_hold_value['value_price'] = bond_hold_value['price'].mul(bond_hold_value['port_weight'])
    bond_hold_value['value_rating'] = bond_hold_value['avg_rating'].mul(bond_hold_value['port_weight'])
    bond_hold_value['value_duration'] = bond_hold_value['duration'].mul(bond_hold_value['port_weight'])

    # Value portfolio
    bond_hold_value_ = bond_hold_value.groupby(['port_group', 'month']).agg({'value_return':'sum', 'value_
price':'sum', 'value_duration':'sum', 'value_rating':'sum'})
    bond_hold_value_['port_group'] = [i[0] for i in bond_hold_value_.index]
    bond_hold_value_['month'] = [i[1] for i in bond_hold_value_.index]
    bond_hold_value_['value_return'] = bond_hold_value_['value_return'].add(1)
    bond_hold_value_['cumulative_return'] = bond_hold_value_.groupby(['port_group']).cumprod()['value_retu
rn']
    bond_hold_value_['cumulative_return']  = bond_hold_value_['cumulative_return'] .subtract(1)
    bond_hold_value_['value_return']  = bond_hold_value_['value_return'] .subtract(1)



    #Equal portfolio
    bond_hold_value___ = bond_hold_value
    #bond_hold_value___['return'] = bond_hold_value___['return'].add(1)
    bond_hold_equal = bond_hold_value___.groupby(['port_group', 'month']).agg({'return':'mean', 'price':'m
ean', 'duration':'mean', 'avg_rating':'mean'})
    bond_hold_equal['port_group'] = [i[0] for i in bond_hold_equal.index]
    bond_hold_equal['month'] = [i[1] for i in bond_hold_equal.index]
    bond_hold_equal['return'] = bond_hold_equal['return'].add(1)
    bond_hold_equal['cumulative_return'] = bond_hold_equal.groupby(['port_group']).cumprod()['return']
    bond_hold_equal['cumulative_return']  = bond_hold_equal['cumulative_return'] .subtract(1)
    bond_hold_equal['return']  = bond_hold_equal['return'] .subtract(1)



    col_name = []
    port = ['P01', 'P02', 'P03', 'P04', 'P05', 'P06', 'P07', 'P08', 'P09', 'P10']
    for i in ['value', 'equal']:
        for j in port:
            col_name.append('{0}_{1}'.format(i, j))

    result = pd.DataFrame(data = 0, columns = col_name, index = [i for i in range(1, 13)])

    for i in port:
        result.ix[:, '{0}_{1}'.format('value', i)] = bond_hold_value_[bond_hold_value_['port_group'] == i]
['cumulative_return'].values

    for i in port:
        result.ix[:, '{0}_{1}'.format('equal', i)] = bond_hold_equal[ bond_hold_equal['port_group'] == i][
'cumulative_return'].values

    return(result)
```

Yearly holding comparision visualization function

In [10]:

```
def comparison_holding_plot(com_hold, universe, avg_period, TMT, strat_type = 0):
    style.use('seaborn')

    fig = plt.figure(figsize=(15, 15))
    strat_name = np.where(strat_type == 0, 'Total', 'Residual')
    fig.suptitle('{0} Momentum Spillover Holding Comparison - {1}'.format(strat_name, universe), fontsize=
20, fontweight='bold')
```

```
    plt.figtext(0.5,0.95, "Average {0}-month equity returns \n Time-to-maturity over {1}".format(avg_period, TMT), ha="center", va="top", fontsize=14, color="black")

    ax1 = plt.subplot2grid((30, 22), (0, 0), rowspan=12, colspan=22)
    ax1.plot(com_hold.iloc[:, 0], 'o-', label="{0}".format(com_hold.columns[0]), color='lightsalmon')
    ax1.plot(com_hold.iloc[:, 4], 'o-', label="{0}".format(com_hold.columns[4]), color='greenyellow')
    ax1.plot(com_hold.iloc[:, 9], 'o-', label="{0}".format(com_hold.columns[9]), color='skyblue')

    for i in range(0, 12):
            ax1.text(com_hold.index[i],1.02*com_hold.iloc[i, 0], '{0:.5f}'.format(com_hold.iloc[i, 0]), ha='center', va='bottom')
            ax1.text(com_hold.index[i],1.02*com_hold.iloc[i, 4], '{0:.5f}'.format(com_hold.iloc[i, 4]), ha='center', va='bottom')
            ax1.text(com_hold.index[i],1.02*com_hold.iloc[i, 9], '{0:.5f}'.format(com_hold.iloc[i, 9]), ha='center', va='bottom')

    ax1.set_title('Cumulative Returns of Value Wighted Portfolios Per Year', fontsize=15, weight='bold')
    ax1.set_ylabel('Cumulative return')
    ax1.legend()

    ax2 = plt.subplot2grid((30, 22), (14, 0), rowspan=12, colspan=22, sharex=ax1)
    ax2.plot(com_hold.iloc[:, 10], 'o-', label="{0}".format(com_hold.columns[10]), color='lightsalmon')
    ax2.plot(com_hold.iloc[:, 14], 'o-', label="{0}".format(com_hold.columns[14]), color='greenyellow')
    ax2.plot(com_hold.iloc[:, 19], 'o-', label="{0}".format(com_hold.columns[19]), color='skyblue')

    for i in range(0, 12):
            ax2.text(com_hold.index[i],1.02*com_hold.iloc[i, 10], '{0:.5f}'.format(com_hold.iloc[i, 10]), ha='center', va='bottom')
            ax2.text(com_hold.index[i],1.02*com_hold.iloc[i, 14], '{0:.5f}'.format(com_hold.iloc[i, 14]), ha='center', va='bottom')
            ax2.text(com_hold.index[i],1.02*com_hold.iloc[i, 19], '{0:.5f}'.format(com_hold.iloc[i, 19]), ha='center', va='bottom')

    ax2.set_title('Cumulative Returns of Equal Wighted Portfolios Per Year', fontsize=15, weight='bold')
    ax2.set_ylabel('Cumulative return')
    ax2.set_xlabel('Months')
    ax2.legend()
```

Backtesting

Import equity data step

In [11]:

```
equity_data_IG = pd.read_csv('CRSP 2012-2013 - IG.csv')
equity_data_IG.date = pd.to_datetime(equity_data_IG.date, format = '%Y%m%d')

equity_data_BBB = pd.read_csv('CRSP 2012-2013 - BBB.csv')
equity_data_BBB.date = pd.to_datetime(equity_data_BBB.date, format = '%Y%m%d')

equity_data_HY = pd.read_csv('CRSP 2012-2013 - HY.csv')
equity_data_HY.date = pd.to_datetime(equity_data_HY.date, format = '%Y%m%d')
```

Import bond data step

In [12]:

```
bond_data_IG = pd.read_csv('WRDS bond return 2013 - IG.csv')
bond_data_IG.DATE = pd.to_datetime(bond_data_IG.DATE, format = '%d-%b-%y')

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

bond_data_BBB = pd.read_csv('WRDS bond return 2013 - BBB.csv')
bond_data_BBB.DATE = pd.to_datetime(bond_data_BBB.DATE, format = '%d-%b-%y')

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

bond_data_HY = pd.read_csv('WRDS bond return 2013 - HY.csv')
bond_data_HY.DATE = pd.to_datetime(bond_data_HY.DATE, format = '%d-%b-%y')

bond_data_HY['month'] = [i.month for i in bond_data_HY.DATE]
```

```
bond_data_HY.YIELD = bond_data_HY.YIELD.str.replace('%', '').astype('float').divide(100.0)
bond_data_HY.RET_EOM = bond_data_HY.RET_EOM.str.replace('%', '').astype('float').divide(100.0)
```

Select formation date of strategy

```
In [13]: formation_date = '2012-12-31'
```

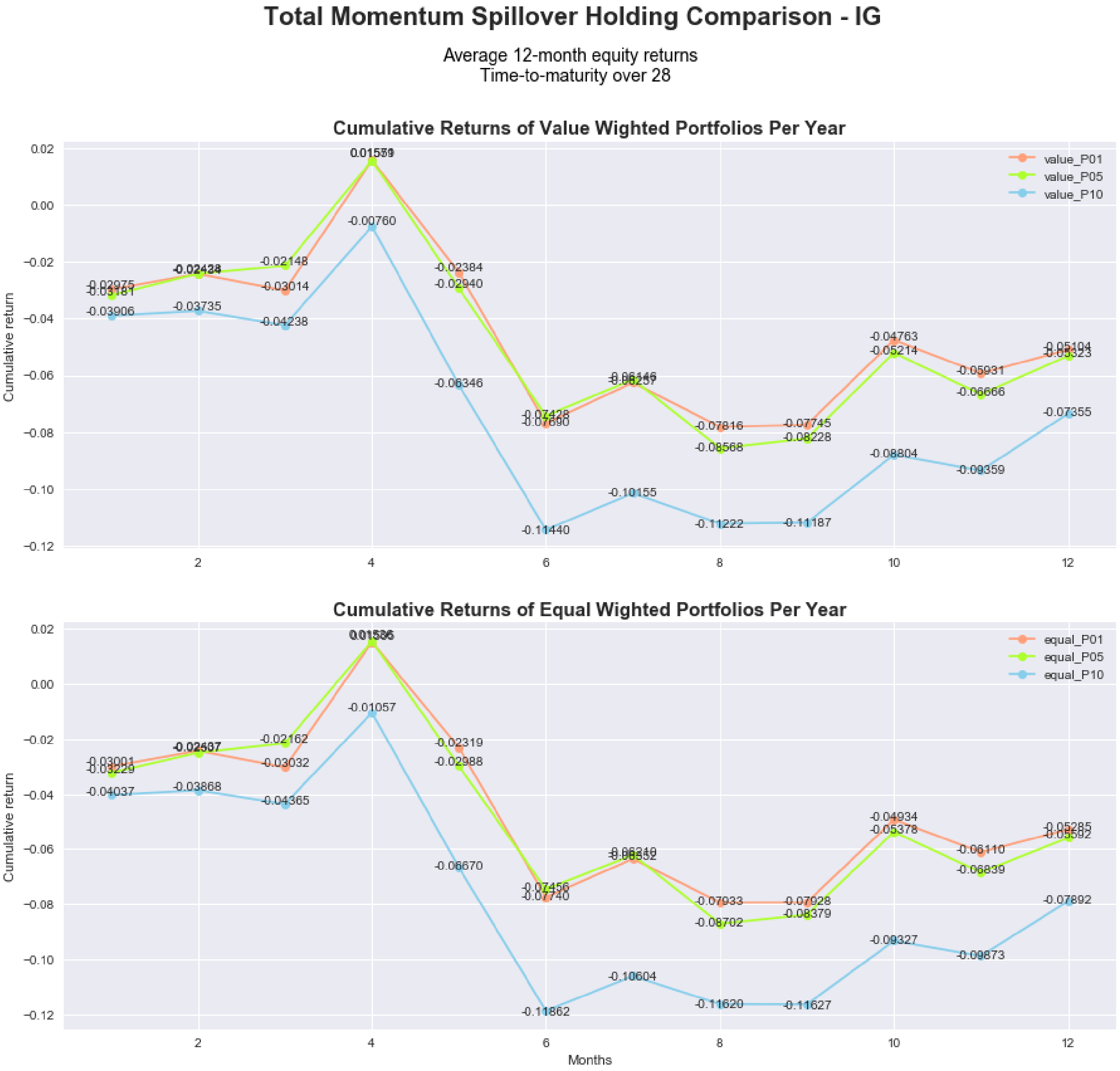
Backtesting on IG

Select time-to-maturity of bonds

```
In [14]: TMT_IG = 28
         universe_IG = 'IG'
```

Observe the effect of momentum to bond portfolios throughout the year

```
In [15]: avg_period_IG = 12
         com_hold_IG = comparison_holding(equity_data_IG, bond_data_IG, formation_date, avg_period_IG, TMT = TMT_IG
         )
         comparison_holding_plot(com_hold_IG, universe_IG, avg_period_IG, TMT = TMT_IG)
```


Total Momentum Spillover Holding Comparison - IG

Average 12-month equity returns
Time-to-maturity over 28

Cumulative Returns of Value Wighted Portfolios Per Year

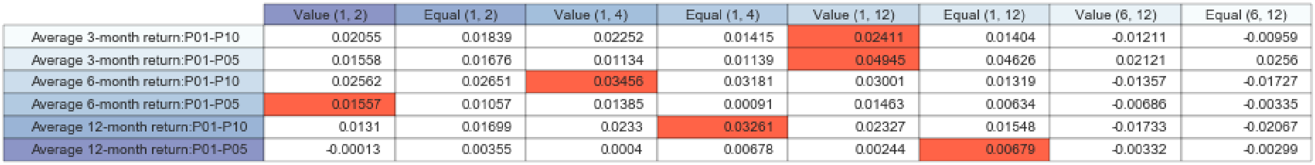
Cumulative Returns of Equal Wighted Portfolios Per Year

As we can see the cumulative returns of Winner portfolio - P01 and loser portfolio - P10 go up from 1 to 4, we will compare the ranges as below

```
In [16]: holding_range_IG = [(1, 2), (1, 4), (1, 12), (6, 12)]
         equity_range = (3, 6, 12)
```

```
In [17]: s_type_IG = 0
         com_perf_IG = comparison_performance(equity_data_IG, bond_data_IG, holding_range_IG, formation_date, equit
         y_range, TMT=TMT_IG, strat_type = s_type_IG)
         comparison_table(com_perf_IG, universe_IG, s_type_IG)
```

Total Momentum Spillover Comparison Performance Table - IG

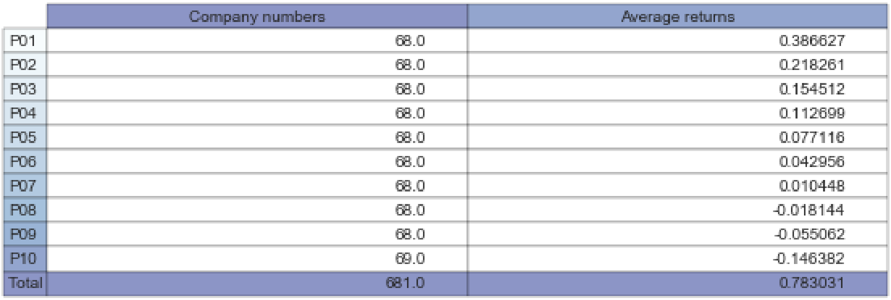


	Value (1, 2)	Equal (1, 2)	Value (1, 4)	Equal (1, 4)	Value (1, 12)	Equal (1, 12)	Value (6, 12)	Equal (6, 12)
Average 3-month return:P01-P10	0.02055	0.01839	0.02252	0.01415	0.02411	0.01404	-0.01211	-0.00959
Average 3-month return:P01-P05	0.01558	0.01676	0.01134	0.01139	0.04945	0.04626	0.02121	0.0256
Average 6-month return:P01-P10	0.02562	0.02651	0.03456	0.03181	0.03001	0.01319	-0.01357	-0.01727
Average 6-month return:P01-P05	0.01557	0.01057	0.01385	0.00091	0.01463	0.00634	-0.00686	-0.00335
Average 12-month return:P01-P10	0.0131	0.01699	0.0233	0.03261	0.02327	0.01548	-0.01733	-0.02067
Average 12-month return:P01-P05	-0.00013	0.00355	0.0004	0.00678	0.00244	0.00679	-0.00332	-0.00299

Using average 6-month equity return and holding bond porfolios from 1 to 4 with over 28-year time-to-maturity bonds outperforms the others. We will see more detail about it.

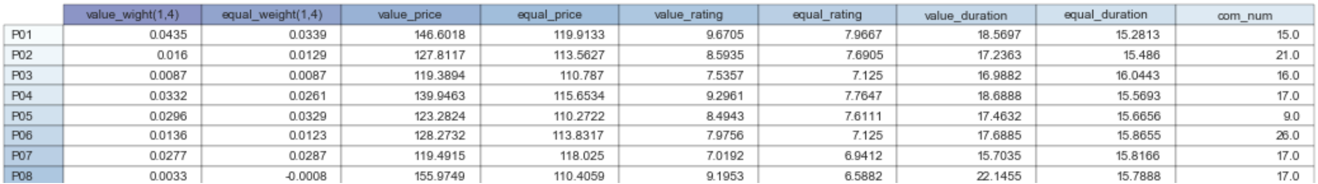
```
In [18]: avg_period_IG = 6

         rank_port_IG = rank_port_decile(equity_data_IG, formation_date, avg_period_IG, s_type_IG)
         rank_table(rank_port_IG, universe_IG, s_type_IG)

         hfrom_IG = 1
         hend_IG = 4
         tmt_IG = 28

         perf_IG = momentum_strategy(bond_data_IG, rank_port_IG, hfrom_IG, hend_IG, TMT=TMT_IG)
         m_performance_table(perf_IG, universe_IG, s_type_IG)

         performance_plot(perf_IG, avg_period_IG, hfrom_IG, hend_IG, tmt_IG, universe_IG, s_type_IG)
```

Ranking Total Equity Returns
From Winner to Loser Portfolios - IG



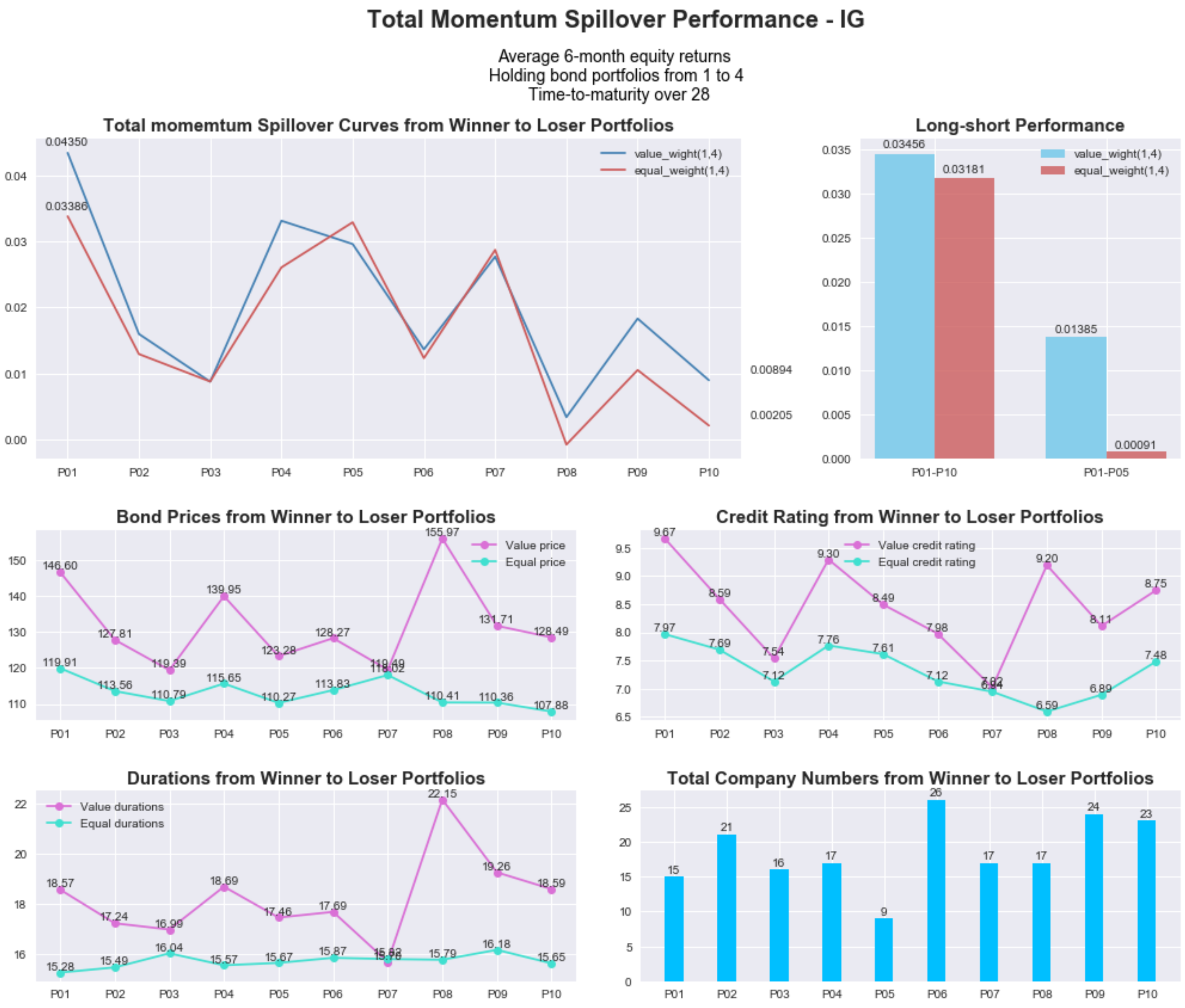
	Company numbers	Average returns
P01	68.0	0.386627
P02	68.0	0.218261
P03	68.0	0.154512
P04	68.0	0.112699
P05	68.0	0.077116
P06	68.0	0.042956
P07	68.0	0.010448
P08	68.0	-0.018144
P09	68.0	-0.055062
P10	69.0	-0.146382
Total	681.0	0.783031

Total Momentum Spillover Performance Table - IG



	value_wight(1,4)	equal_weight(1,4)	value_price	equal_price	value_rating	equal_rating	value_duration	equal_duration	com_num
P01	0.0435	0.0339	146.6018	119.9133	9.6705	7.9667	18.5697	15.2813	15.0
P02	0.016	0.0129	127.8117	113.5627	8.5935	7.6905	17.2363	15.486	21.0
P03	0.0087	0.0087	119.3894	110.787	7.5357	7.125	16.9882	16.0443	16.0
P04	0.0332	0.0261	139.9463	115.6534	9.2961	7.7647	18.6888	15.5693	17.0
P05	0.0296	0.0329	123.2824	110.2722	8.4943	7.6111	17.4632	15.6656	9.0
P06	0.0136	0.0123	128.2732	113.8317	7.9756	7.125	17.6885	15.8655	26.0
P07	0.0277	0.0287	119.4915	118.025	7.0192	6.9412	15.7035	15.8166	17.0
P08	0.0033	-0.0008	155.9749	110.4059	9.1953	6.5882	22.1455	15.7888	17.0

P09	0.0183	0.0105	131.7052	110.362	8.1124	6.8869	19.2561	16.1818	24.0
P10	0.0089	0.0021	128.4866	107.8779	8.749	7.4783	18.5927	15.6465	23.0
P01-P10	0.0346	0.0318	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P01-P05	0.0139	0.0009	0.0	0.0	0.0	0.0	0.0	0.0	0.0


Total Momentum Spillover Performance - IG

Average 6-month equity returns
Holding bond portfolios from 1 to 4
Time-to-maturity over 28

Total momemtum Spillover Curves from Winner to Loser Portfolios

Long-short Performance

Bond Prices from Winner to Loser Portfolios

Credit Rating from Winner to Loser Portfolios

Durations from Winner to Loser Portfolios

Total Company Numbers from Winner to Loser Portfolios

The performance on IG universe shows fair momentum with gaining around 3.2% - 3.5% return.

Backtesting on BBB

Select time-to-maturity of bonds

```
In [19]: TMT_BBB = 5
         universe_BBB = 'BBB'
```

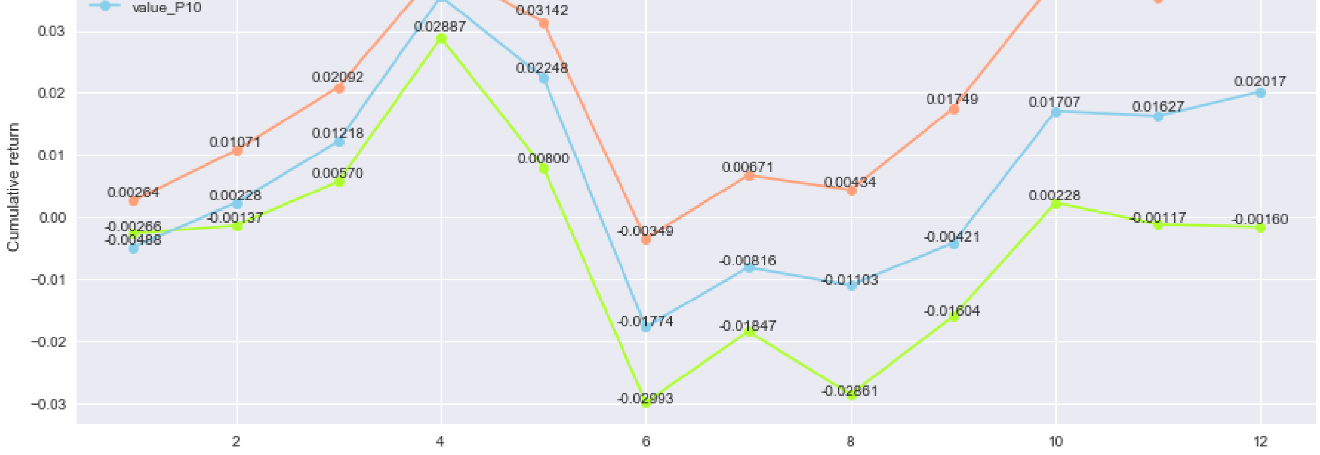
Observe the effect of momentum to bond portfolios throughout the year

```
In [20]: avg_period_BBB = 6
         com_hold_BBB = comparison_holding(equity_data_BBB, bond_data_BBB, formation_date, avg_period_BBB, strat_ty
         pe = 1, TMT = TMT_BBB)
         comparison_holding_plot(com_hold_BBB, universe_BBB, avg_period_BBB, TMT = TMT_BBB, strat_type = 1)
```

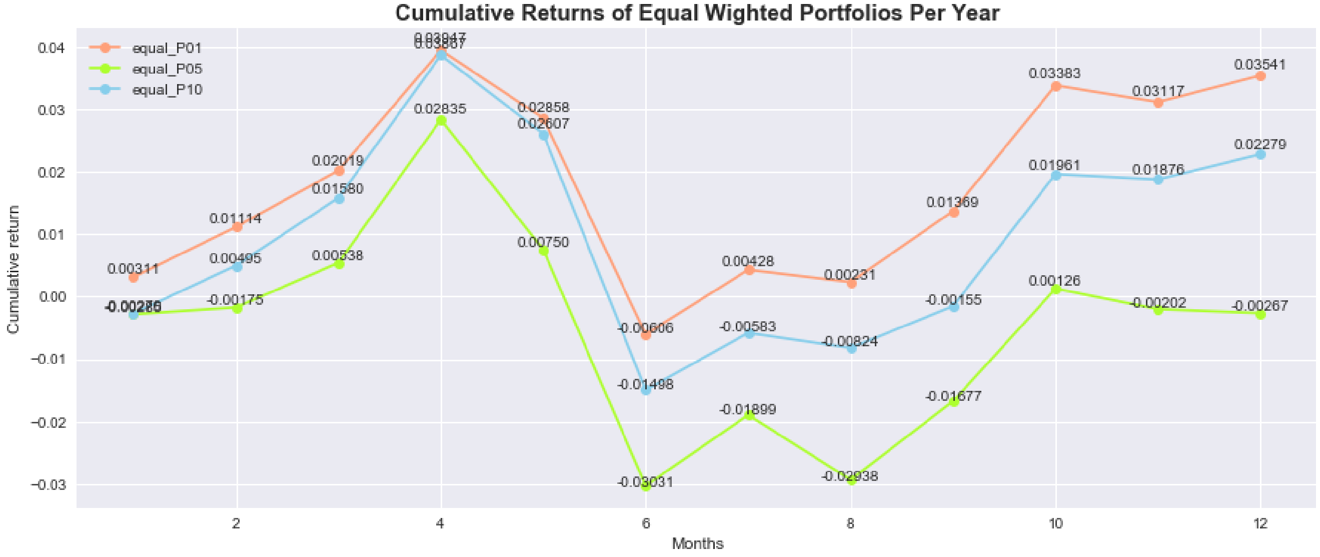
Residual Momentum Spillover Holding Comparison - BBB

Average 6-month equity returns
Time-to-maturity over 5

Cumulative Returns of Value Wighted Portfolios Per Year
value_P01
value_P05
0.04
0.04029
0.03556
0.03830
0.03546
0.03947



Cumulative Returns of Equal Wighted Portfolios Per Year



As we can see the cumulative returns of Winner portfolio - P01 and loser portfolio - P10 go up from 1 to 4 as weel as from 1 to 12 is still looking-good, we will compare the ranges as below

In [21]:
```
holding_range_BBB = [(2, 2), (1, 3), (1, 4), (1, 12)]
```

In [22]:
```
s_type_BBB = 1
com_perf_BBB = comparison_performance(equity_data_BBB, bond_data_BBB, holding_range_BBB, formation_date, e
quity_range, TMT=TMT_BBB, strat_type = s_type_BBB)
comparison_table(com_perf_BBB, universe_BBB, s_type_BBB)
```

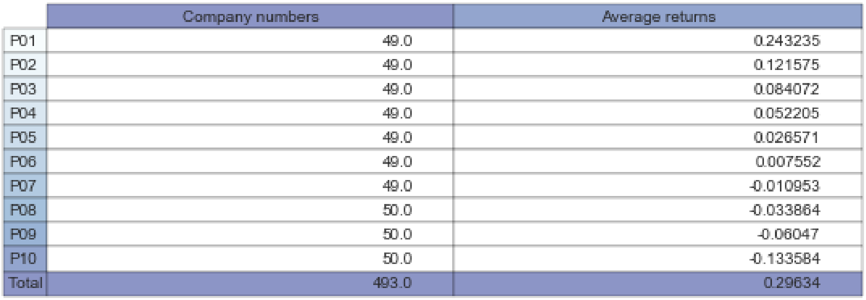
Residual Momentum Spillover Comparison Performance Table - BBB

	Value (2, 2)	Equal (2, 2)	Value (1, 3)	Equal (1, 3)	Value (1, 4)	Equal (1, 4)	Value (1, 12)	Equal (1, 12)
Average 3-month return:P01-P10	0.00361	0.00334	0.01835	0.01517	0.01836	0.01397	0.03781	0.02726
Average 3-month return:P01-P05	0.00385	0.00392	0.00603	0.00104	0.00222	-0.00278	0.03073	0.01737
Average 6-month return:P01-P10	0.00086	0.00032	0.00859	0.00282	0.00448	-0.00192	0.01848	0.00909
Average 6-month return:P01-P05	0.00676	0.0069	0.01508	0.01373	0.01102	0.00904	0.04001	0.03506
Average 12-month return:P01-P10	0.00269	0.00186	0.00729	0.00278	0.00616	0.00073	0.01235	0.00709
Average 12-month return:P01-P05	0.00312	0.00314	0.01052	0.01078	0.00897	0.00797	0.02673	0.02815

Using average 3-month equity return and holding bond porfolios from 1 to 12 with over 28-year time-to-maturity bonds outperforms and shows stronger momentum than the others. We will see more detail about it.

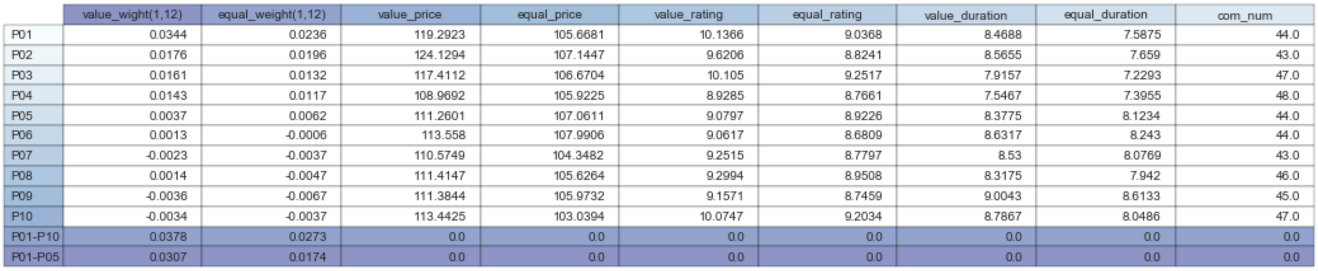
In [23]:
```
avg_period_BBB = 3

rank_port_BBB = rank_port_decile(equity_data_BBB, formation_date, avg_period_BBB, s_type_BBB)
rank_port_BBB = rank_port_decile(equity_data_BBB, formation_date, avg_period_BBB, s_type_BBB)
rank_table(rank_port_BBB, universe_BBB, s_type_BBB)

hfrom_BBB = 1
hend_BBB = 12
tmt_BBB = 5

perf_BBB = momentum_strategy(bond_data_BBB, rank_port_BBB, hfrom_BBB, hend_BBB, TMT=TMT_BBB)
m_performance_table(perf_BBB, universe_BBB, s_type_BBB)

performance_plot(perf_BBB, avg_period_BBB, hfrom_BBB, hend_BBB, tmt_BBB, universe_BBB, s_type_BBB)
```

Ranking Residual Equity Returns
From Winner to Loser Portfolios - BBB



	Company numbers	Average returns
P01	49.0	0.243235
P02	49.0	0.121575
P03	49.0	0.084072
P04	49.0	0.052205
P05	49.0	0.026571
P06	49.0	0.007552
P07	49.0	-0.010953
P08	50.0	-0.033864
P09	50.0	-0.06047
P10	50.0	-0.133584
Total	493.0	0.29634

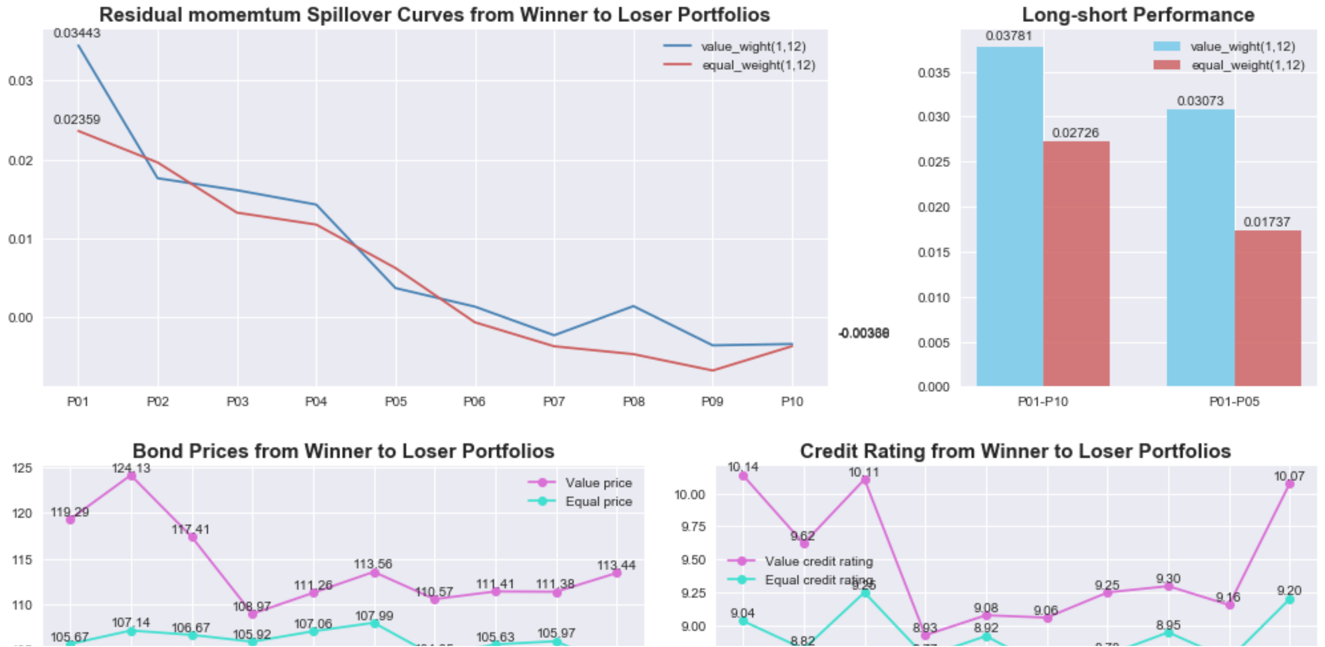
Residual Momentum Spillover Performance Table - BBB

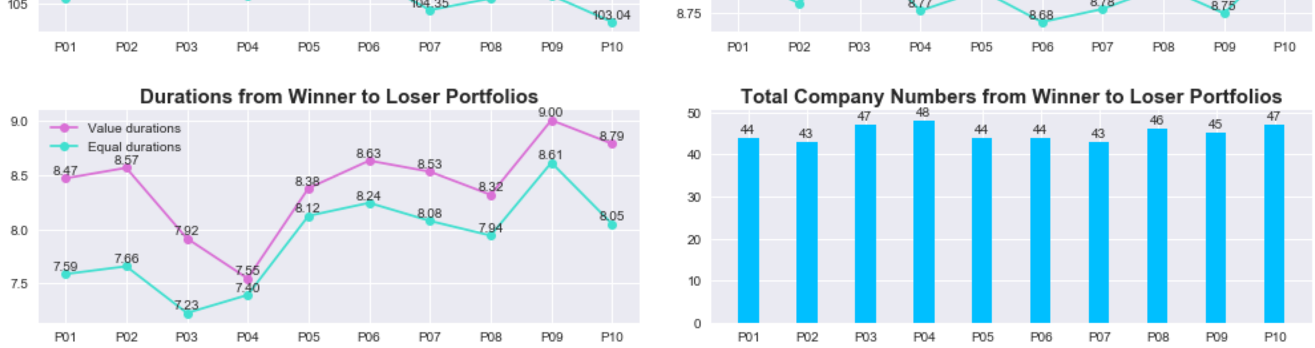


	value_wight(1,12)	equal_weight(1,12)	value_price	equal_price	value_rating	equal_rating	value_duration	equal_duration	com_num
P01	0.0344	0.0236	119.2923	105.6681	10.1366	9.0368	8.4688	7.5875	44.0
P02	0.0176	0.0196	124.1294	107.1447	9.6206	8.8241	8.5655	7.659	43.0
P03	0.0161	0.0132	117.4112	106.6704	10.105	9.2517	7.9157	7.2293	47.0
P04	0.0143	0.0117	108.9692	105.9225	8.9285	8.7661	7.5467	7.3955	48.0
P05	0.0037	0.0062	111.2601	107.0611	9.0797	8.9226	8.3775	8.1234	44.0
P06	0.0013	-0.0006	113.558	107.9906	9.0617	8.6809	8.6317	8.243	44.0
P07	-0.0023	-0.0037	110.5749	104.3482	9.2515	8.7797	8.53	8.0769	43.0
P08	0.0014	-0.0047	111.4147	105.6264	9.2994	8.9508	8.3175	7.942	46.0
P09	-0.0036	-0.0067	111.3844	105.9732	9.1571	8.7459	9.0043	8.6133	45.0
P10	-0.0034	-0.0037	113.4425	103.0394	10.0747	9.2034	8.7867	8.0486	47.0
P01-P10	0.0378	0.0273	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P01-P05	0.0307	0.0174	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Residual Momentum Spillover Performance - BBB

Average 3-month equity returns
Holding bond portfolios from 1 to 12
Time-to-maturity over 5





The performance on BBB universe shows strong momentum with gaining around 2.7% - 3.8% return.

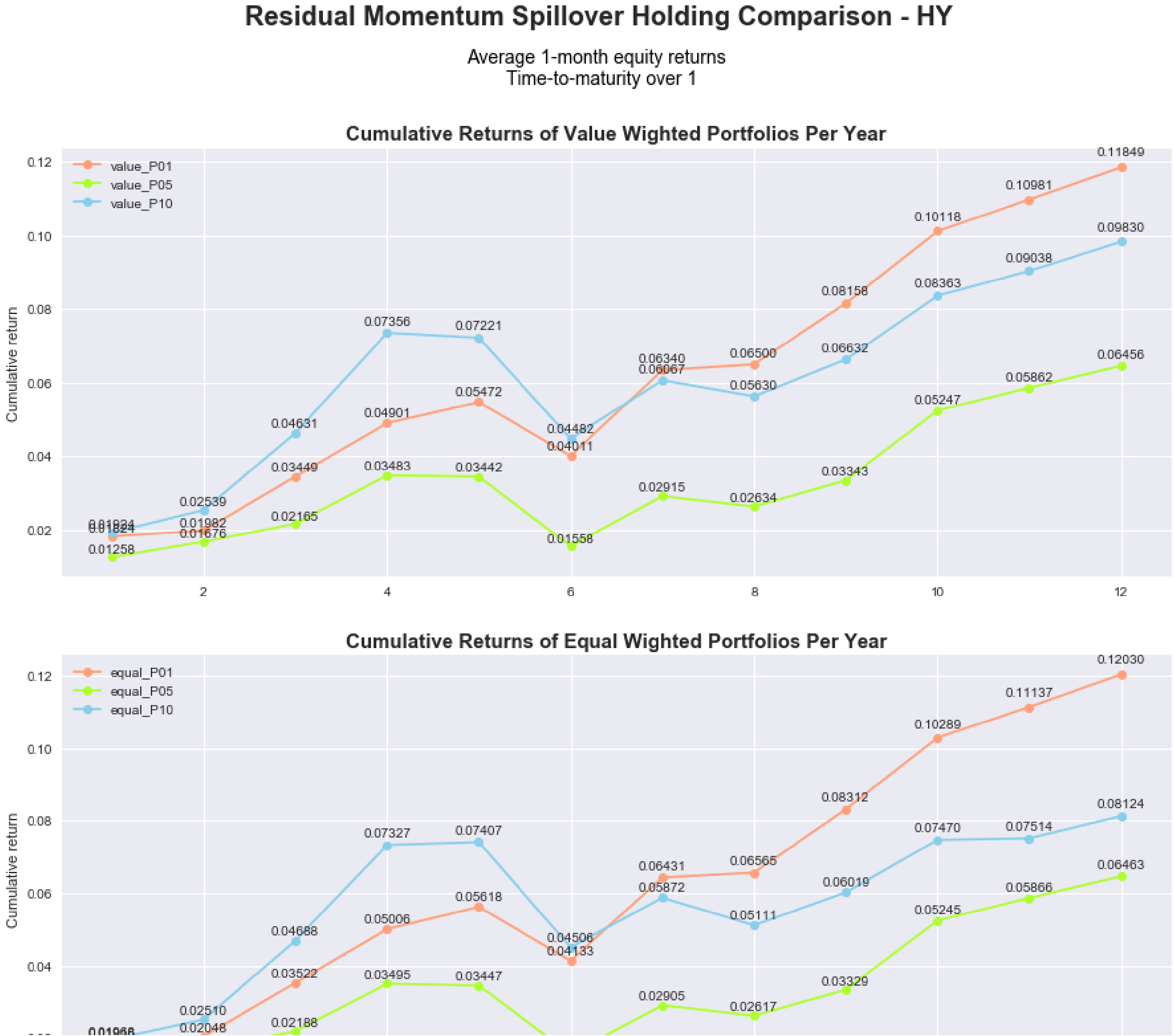
Backtesting on HY

Select time-to-maturity of bonds

In [24]:
```
TMT_HY = 1
universe_HY = 'HY'
```

Observe the effect of momentum to bond portfolios throughout the year

In [25]:
```
avg_period_HY = 1
com_hold_HY = comparison_holding(equity_data_HY, bond_data_HY, formation_date, avg_period_HY, strat_type =
 1, TMT = TMT_HY)
comparison_holding_plot(com_hold_HY, universe_HY, avg_period_HY,TMT = TMT_HY, strat_type = 1)
```

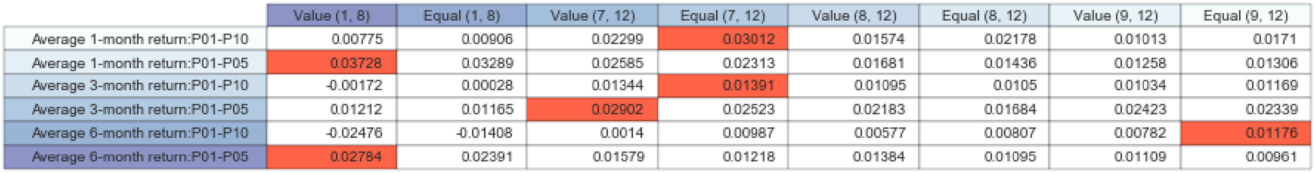


As we can see the cumulative returns of Winner portfolio - P01 goes higher than loser portfolio - P10 from 7 to 12, we will compare the ranges as below

In [26]:
```
holding_range_HY = [(1, 8), (7, 12), (8, 12), (9, 12)]
equity_range_HY = (1, 3, 6)
```

In [27]:
```
s_type_HY = 1
com_perf_HY = comparison_performance(equity_data_HY, bond_data_HY, holding_range_HY, formation_date, equit
y_range_HY, TMT=TMT_HY, strat_type = s_type_HY)
comparison_table(com_perf_HY, universe_HY, s_type_HY)
```

Residual Momentum Spillover Comparison Performance Table - HY

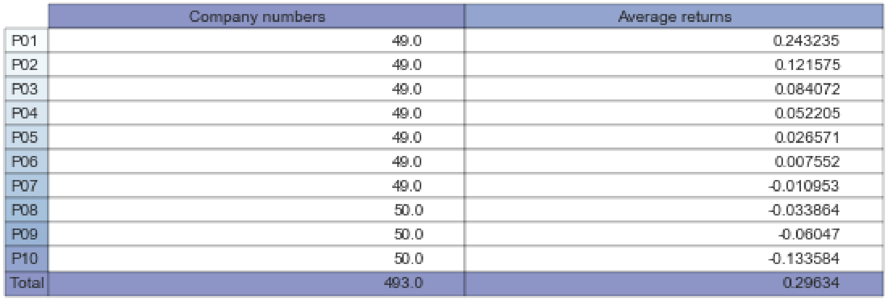


	Value (1, 8)	Equal (1, 8)	Value (7, 12)	Equal (7, 12)	Value (8, 12)	Equal (8, 12)	Value (9, 12)	Equal (9, 12)
Average 1-month return:P01-P10	0.00775	0.00906	0.02299	0.03012	0.01574	0.02178	0.01013	0.0171
Average 1-month return:P01-P05	0.03728	0.03289	0.02585	0.02313	0.01681	0.01436	0.01258	0.01306
Average 3-month return:P01-P10	-0.00172	0.00028	0.01344	0.01391	0.01095	0.0105	0.01034	0.01169
Average 3-month return:P01-P05	0.01212	0.01165	0.02902	0.02523	0.02183	0.01684	0.02423	0.02339
Average 6-month return:P01-P10	-0.02476	-0.01408	0.0014	0.00987	0.00577	0.00807	0.00782	0.01176
Average 6-month return:P01-P05	0.02784	0.02391	0.01579	0.01218	0.01384	0.01095	0.01109	0.00961

Using average 1-month equity return and holding bond porfolios from 7 to 12 with over 1-year time-to-maturity bonds outperforms and shows stronger momentum than the others. We will see more detail about it.

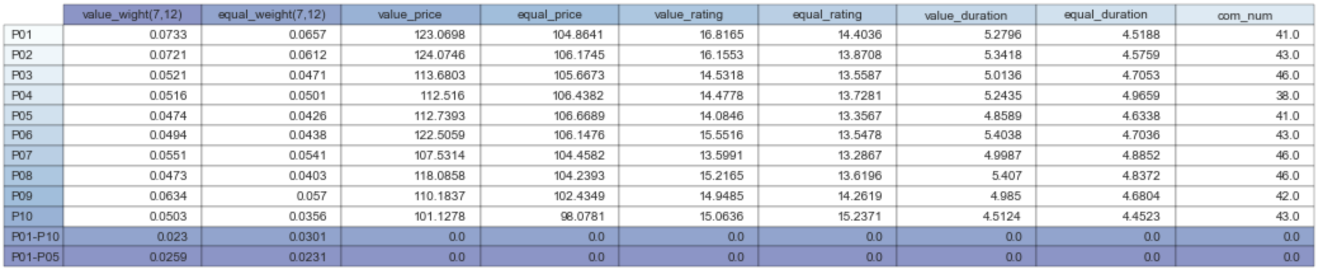
In [28]:
```
avg_period_HY = 1

rank_port_HY = rank_port_decile(equity_data_HY, formation_date, avg_period_HY, s_type_HY)
rank_table(rank_port_BBB, universe_HY, s_type_HY)

hfrom_HY = 7
hend_HY = 12
tmt_HY = 1

perf_HY = momentum_strategy(bond_data_HY, rank_port_HY, hfrom_HY, hend_HY, TMT=TMT_HY)
m_performance_table(perf_HY, universe_HY, s_type_HY)

performance_plot(perf_HY, avg_period_HY, hfrom_HY, hend_HY, tmt_HY, universe_HY, s_type_HY)
```

Ranking Residual Equity Returns
From Winner to Loser Portfolios - HY

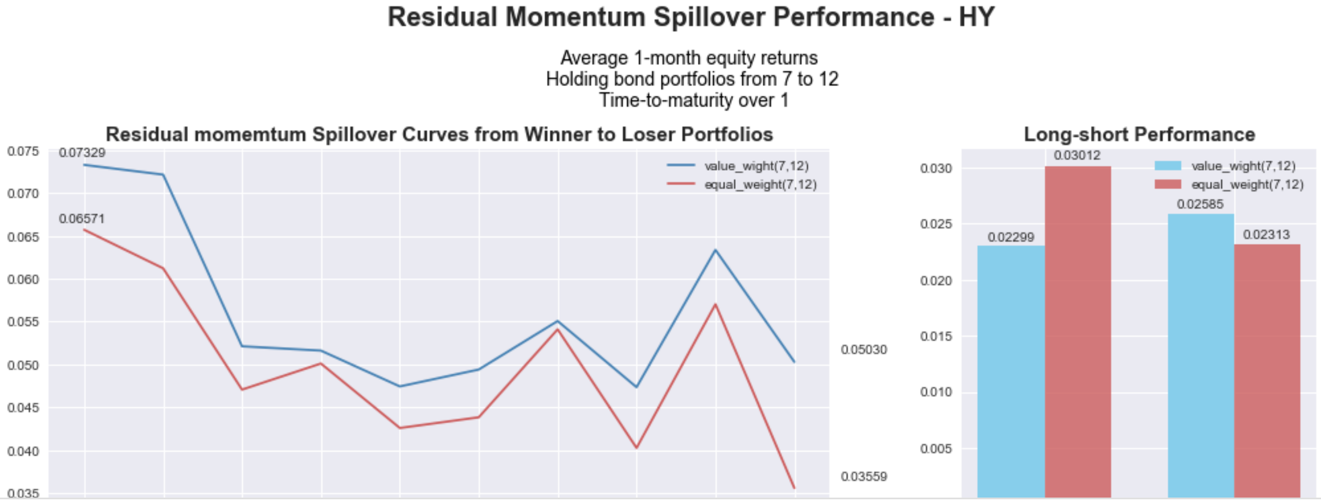


	Company numbers	Average returns
P01	49.0	0.243235
P02	49.0	0.121575
P03	49.0	0.084072
P04	49.0	0.052205
P05	49.0	0.026571
P06	49.0	0.007552
P07	49.0	-0.010953
P08	50.0	-0.033864
P09	50.0	-0.06047
P10	50.0	-0.133584
Total	493.0	0.29634

Residual Momentum Spillover Performance Table - HY



	value_wight(7,12)	equal_weight(7,12)	value_price	equal_price	value_rating	equal_rating	value_duration	equal_duration	com_num
P01	0.0733	0.0657	123.0698	104.8641	16.8165	14.4036	5.2796	4.5188	41.0
P02	0.0721	0.0612	124.0746	106.1745	16.1553	13.8708	5.3418	4.5759	43.0
P03	0.0521	0.0471	113.6803	105.6673	14.5318	13.5587	5.0136	4.7053	46.0
P04	0.0516	0.0501	112.516	106.4382	14.4778	13.7281	5.2435	4.9659	38.0
P05	0.0474	0.0426	112.7393	106.6689	14.0846	13.3567	4.8589	4.6338	41.0
P06	0.0494	0.0438	122.5059	106.1476	15.5516	13.5478	5.4038	4.7036	43.0
P07	0.0551	0.0541	107.5314	104.4582	13.5991	13.2867	4.9987	4.8852	46.0
P08	0.0473	0.0403	118.0858	104.2393	15.2165	13.6196	5.407	4.8372	46.0
P09	0.0634	0.057	110.1837	102.4349	14.9485	14.2619	4.985	4.6804	42.0
P10	0.0503	0.0356	101.1278	98.0781	15.0636	15.2371	4.5124	4.4523	43.0
P01-P10	0.023	0.0301	0.0	0.0	0.0	0.0	0.0	0.0	0.0
P01-P05	0.0259	0.0231	0.0	0.0	0.0	0.0	0.0	0.0	0.0



Residual Momentum Spillover Performance - HY

Average 1-month equity returns
Holding bond portfolios from 7 to 12
Time-to-maturity over 1