Hedging Our Bets Muni-Style

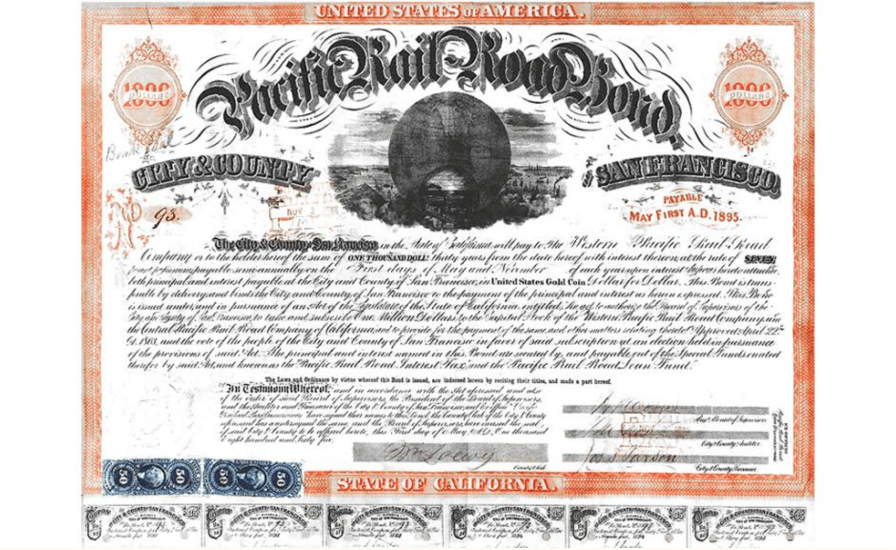
Project Team 3: Alex Domenick, Emmanuel Korlewala, Phillip Millspaugh, Adrienne Stark, Feng Zhang

Agenda

- What are municipal bonds?
- What does it mean to hedge?
- Our research question
- Methodology
- Challenges
- Conclusions

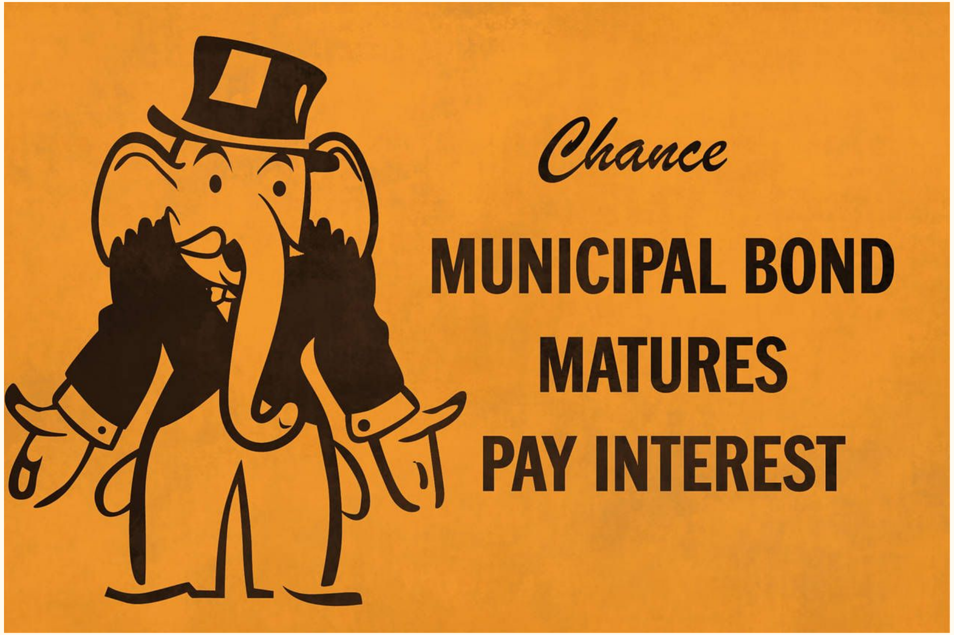
What are municipal bonds?

- Issued by state and local governments
- Raise money for public projects such as bridges, transit systems, and stadiums
- Example: California General Obligation 5% coupon due to mature 1/1/30



Why invest in municipal bonds?

- Typically high-quality
- Support local projects
- Receive interest that is tax-exempt at federal level and often at state level



What is hedging?

Municipal bond arbitrage involves building a leveraged portfolio of tax-exempt municipal bonds and simultaneously hedging the **duration risk of the portfolio**. The hedging takes place through the short sale of equivalent taxable corporate bonds of the same maturity, generally via interest rate swaps. (source: Investopedia)

Problems with hedging

- Municipal bonds are a long-only market
- Tax-exempt securities need to be hedge with taxable securities
- Highly fragmented market with 50,000+ issuers
- OTC traded security with possibly thin price discovery

Our research question:

How can we choose the best hedge for a given municipal bond?

Methodology

- Identify a sample of 816 out of 6,711 total municipal bonds on the market and pull daily data for the last seven years 1/1/2013 - 10/31/2020
- Identify potential hedges (Treasuries, ICE LIBOR swaps, MUB, TLT, JNK, LQD, UDN) and pull price data for them
- Prepare and clean the data
- Calculate daily return data and determine the correlation of bonds versus a given hedge
- Identify the most closely correlated instrument which when shorted would be the most appropriate hedge

Data Preparation

- Muni Bond 7 years daily prices (816 random sampling from 6711 bonds from Bloomberg Barclays Municipal Bonds Index) ----- Bloomberg
- LIBOR swaps ----- Bloomberg
- US Treasury Yield Rate (1yr, 2yr, 3yr, 5yr, 7yr, 10yr, 20yr, 30yrs) ----- Quandl API
- Potential ETFs that could be used to hedge ("MUB", "TLT", "LQD", "JNK", "UDN", "UUP") ----- Alpaca API
- States with latitude and longitude ----- https://developers.google.com/public-data/docs/canonical/states_csv

Data Cleaning

ETFs data format from Alpaca

```
#import alpaca api file for ETF price history
alpaca = tradeapi.REST(
    alpaca_api_key,
    alpaca_secret_key,
    api_version="v2")

# Format current date as ISO format
start_date = pd.Timestamp("2013-01-02", tz="America/New_York").isoformat()
today = pd.Timestamp("2020-11-02", tz="America/New_York").isoformat()

# Set the tickers
tickers = ["MUB", "TLT", "UDN", "UUP","LQD","JNK"]

# Set timeframe to '1D' for Alpaca API
timeframe = "1D"

# Get closing prices
df_ETFs = alpaca.get_barset(
    tickers,
    timeframe,
    start = start_date,
    end = today
).df

# Preview DataFrame
# YOUR CODE HERE!
df_ETFs.head()

# Output the data to CSV
#df_ETFs.to_csv("Resources/ETFs1.csv", encoding='utf-8', index=True)
```

	JNK					LQD					...	UDN					UUP				
	open	high	low	close	volume	open	high	low	close	volume	...	open	high	low	close	volume	open	high	low	close	volume
2013-01-02 00:00:00-05:00	41.05	41.05	40.96	41.03	6801481	120.96	121.310	120.81	121.27	3392382	...	27.28	27.3099	27.150	27.180	33539.0	21.7400	21.85	21.7200	21.82	782706.0
2013-01-03 00:00:00-05:00	40.95	41.03	40.89	40.94	6337005	121.14	121.180	120.56	120.60	2871556	...	27.08	27.1000	26.950	26.960	72918.0	21.9000	22.00	21.8873	21.99	1091846.0
2013-01-04 00:00:00-05:00	40.99	41.05	40.88	40.99	5081128	120.57	120.649	120.26	120.55	3998513	...	26.90	26.9700	26.890	26.950	146499.0	22.0300	22.05	21.9800	21.99	1022712.0
2013-01-07 00:00:00-05:00	40.93	41.09	40.89	41.08	5909999	120.69	120.800	120.51	120.72	2471978	...	26.94	27.0400	26.932	27.040	41209.0	21.9901	22.01	21.9200	21.93	1910335.0
2013-01-08 00:00:00-05:00	41.08	41.10	41.02	41.06	2958198	120.81	120.940	120.76	120.85	2138549	...	26.99	27.0100	26.961	27.005	38866.0	21.9600	21.99	21.9500	21.95	1134138.0

Data Cleaning

Change the Date format and data structure to fit our needs

```
ETF_csv = Path("Resources/ETFs.csv")
ETF_df = pd.read_csv(ETF_csv, index_col='Unnamed: 0', infer_datetime_format=True, parse_dates=True)
#ETF_df.rename(columns={'Unnamed: 0': 'Date'}, inplace = True)
ETF_df = ETF_df.drop(ETF_df.index[0])
#ETF_df['Date'] = pd.to_datetime(ETF_df['Date'], utc=True)
ETF_df.index = pd.to_datetime(ETF_df.index, utc=True).date
#ETF_df['Date']= ETF_df['Date'].dt.strftime('%m/%d/%Y')
ETF_df.sort_index(inplace=True)

#drop unneccessary columns and keep close price columns
ETF_close_prices_df = ETF_df.loc[:, ['JNK.3', 'LQD.3', 'MUB.3', 'TLT.3', 'UDN.3', 'UUP.3']]
ETF_close_prices_df.rename(columns={'JNK.3': 'JNK',
                   'LQD.3': 'LQD',
                   'MUB.3': 'MUB',
                   'TLT.3': 'TLT',
                   'UDN.3': 'UDN',
                   'UUP.3': 'UUP',
                  }, inplace=True)

ETF_close_prices_df[['JNK', 'LQD', 'MUB', 'TLT', 'UDN', 'UUP']] = ETF_close_prices_df[['JNK', 'LQD', 'MUB', 'TLT', 'UDN', 'UUP']].astype(float)
ETF_close_prices_df.dtypes

#calculate daily returns
ETF_daily_returns_df = ETF_close_prices_df.pct_change().dropna()
ETF_daily_returns_df.head()
```

	JNK	LQD	MUB	TLT	UDN	UUP
2013-01-03	-0.002194	-0.005525	0.000002	-0.013632	-0.008094	0.007791
2013-01-04	0.001221	-0.000415	-0.001968	0.003900	-0.000371	0.000000
2013-01-07	0.002196	0.001410	0.001434	0.000210	0.003340	-0.002729
2013-01-08	-0.000487	0.001077	0.001790	0.006714	-0.001294	0.000912
2013-01-09	0.000974	-0.000153	0.002501	-0.001007	-0.003148	0.002733

Define Functions

Functions to calculate bond prices from yield

```
#create new dataframe with maturity values
columns = []
tsy_mty_df = USTREASURY_data.copy()
for column in USTREASURY_data.columns:
    maturity = ''
    for item in column.split():
        if item.isdigit():
            maturity = maturity + item
    #print(int(maturity[0]))
    tsy_mty_df[column] = int(maturity)
    columns.append(maturity)
USTREASURY_data.columns = columns
tsy_mty_df.columns = columns

#USTREASURY_data.head()
#tsy_mty_df.head()

#define function for calculating Treasury bond prices
def bondprice(fv, c, ytm, t, m):
    bondprice = ((fv*c/m*(1-(1+ytm/m)**(-m*t)))/(ytm/m)) + fv*(1+(ytm/m))**(-m*t)
    return(bondprice)

#bondprice(1000,0.06,0.08,9,2)
#bondprice(100,0.0015,0.0014,1,2)

#create new dataframe with treasury bond prices
fv = 100
c = USTREASURY_data.shift(1)/100
ytm = USTREASURY_data/100
m = 2
t = tsy_mty_df
USTREASURY_daily_prices = bondprice(fv, c, ytm, t, m)

USTREASURY_data_daily_returns = USTREASURY_daily_prices.pct_change().dropna()
#USTREASURY_data_daily_returns.head()
USTREASURY_data_daily_returns.columns = USTREASURY_data_daily_returns.columns.astype(int)
USTREASURY_data_daily_returns.head()
#tsy_mty_df.head()
```

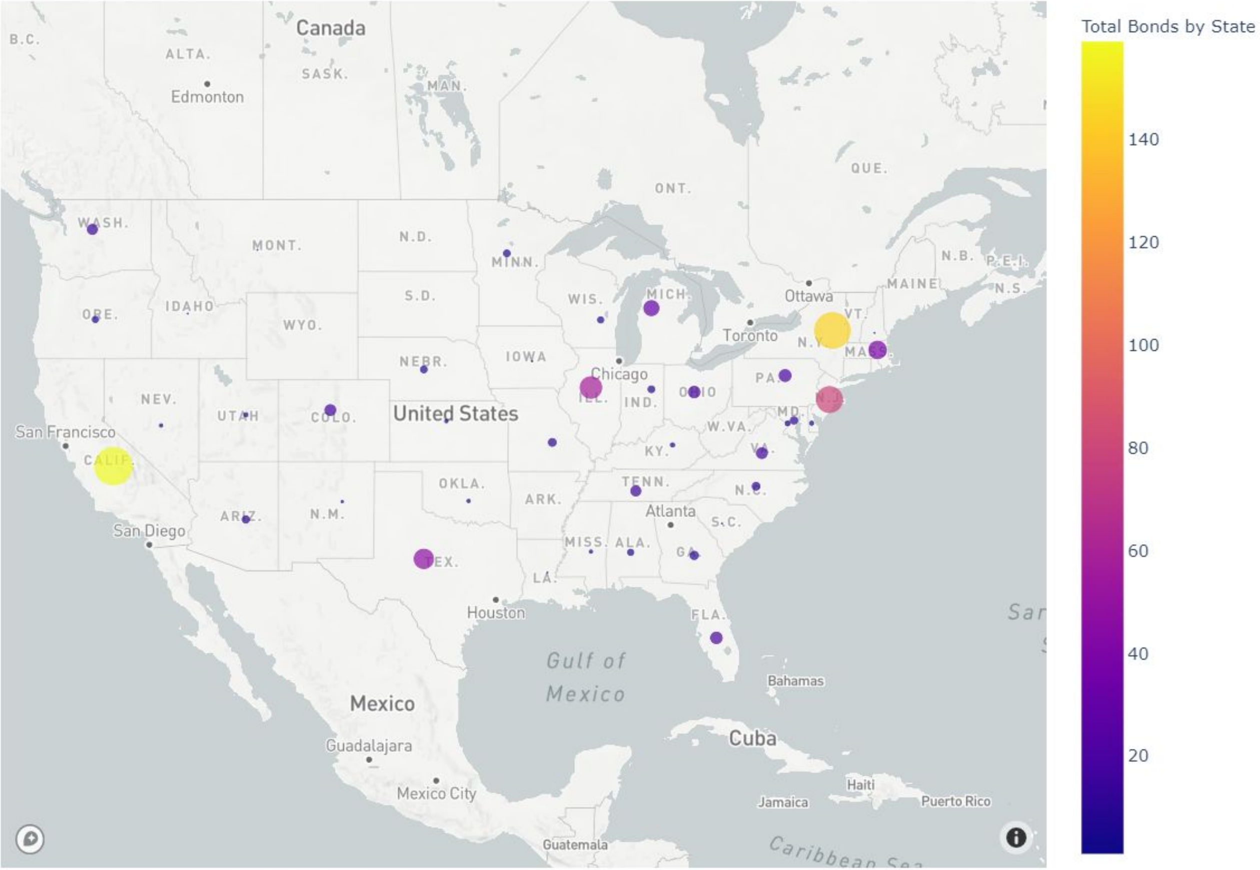
Date	1	2	3	5	7	10	20	30
2013-01-04	0.000000e+00	0.000000	0.000596	0.001961	0.003348	0.004555	0.010880	0.019705
2013-01-07	0.000000e+00	0.000000	0.000298	0.000489	0.001334	0.001813	0.000000	-0.003873
2013-01-08	9.989510e-05	0.000399	0.000894	0.001468	0.001335	0.001814	0.006173	0.007816
2013-01-09	7.485769e-09	-0.000199	-0.000595	-0.000488	-0.001332	-0.001809	-0.004600	-0.007755
2013-01-10	-1.997777e-04	-0.000598	-0.000298	-0.002444	-0.002667	-0.003624	-0.006156	-0.003898

Functions

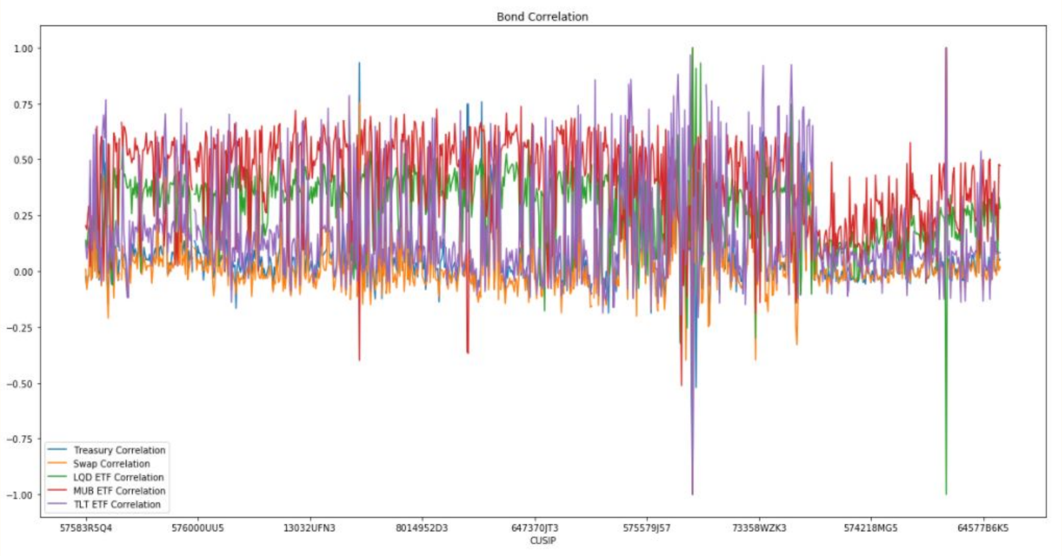
Functions to calculate muni bond prices

```python
#function passing a cusip to put it against hedges
#what are the steps we need our function to do:
    #1.pick bond
    #2. compare versus all hedges
    #3. show correlation of bond vis a vis different hedges
#ETF_daily_returns_df, USTREASURY_data_daily_returns, swaps_daily_returns, bonds_daily_returns
cusip = '677561LN0'
def historical_corr(cusip):
    # find maturity of chosen muni bond and convert to datetime
    bond_mty = bondescr_df.loc[cusip,'Maturity Date']
    bond_mty = dt.datetime.strptime(bond_mty, '%m/%d/%Y')
    bond_mty

    # create series with bond years to maturity for lookup in dataframes
    # relativedelta function: https://dateutil.readthedocs.io/en/stable/relativedelta.html
    ## make output for years more exact by looking at days/360 and rounding?
    mty_years = []
    for return_date in bonds_daily_returns.index:
        difference = relativedelta(bond_mty, return_date)
        mty_years.append(difference.years)
    #len(mty_years)

    # reference closest maturity in tsy and swap df
    # USTREASURY_data_daily_returns, swaps_daily_returns, mty_years
    tsy_index = []
    for item in mty_years:
        min_item = ' '
        min_value = 0
        for col in USTREASURY_data_daily_returns.columns:
            if min_item == ' ':
                min_item = col
                min_value = abs(col - int(item))
            elif abs(col - int(item)) < min_value:
                min_item = col
                min_value = abs(col - int(item))
        tsy_index.append(min_item)

    swap_index = []
    for item in mty_years:
        min_item = ' '
        min_value = 0
        for col in swaps_daily_returns.columns:
            if min_item == ' ':
                min_item = col
                min_value = abs(col - int(item))
            elif abs(col - int(item)) < min_value:
                min_item = col
                min_value = abs(col - int(item))
        swap_index.append(min_item)
    #len(swap_index)
    #len(tsy_index)

    # create new dataframe to combine items
    # USTREASURY_data_daily_returns, swaps_daily_returns, mty_years
    joined_df = pd.DataFrame(bonds_daily_returns[cusip])
    joined_df['tsy_index'] = tsy_index
    joined_df['swap_index'] = swap_index

    joined_df = pd.merge(joined_df, USTREASURY_data_daily_returns, left_index=True, right_index=True)

    tsy_change = []
    for indx in joined_df.index:
        change = joined_df.loc[indx, joined_df['tsy_index'][indx]]
        tsy_change.append(change)
    joined_df['tsy_change'] = tsy_change
    tsy_col_drop = ['tsy_index',1,2,3,5,7,10,20,30]
    joined_df.drop(columns=tsy_col_drop,inplace=True)

    joined_df = pd.merge(joined_df, swaps_daily_returns, left_index=True, right_index=True)

    swap_change = []
    for indx in joined_df.index:
        change = joined_df.loc[indx, joined_df['swap_index'][indx]]
        swap_change.append(change)
    joined_df['swap_change'] = swap_change
    swap_col_drop = ['swap_index',1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30]
    joined_df.drop(columns=swap_col_drop,inplace=True)

    ##concat separate dataframes (muni etf, treasury etf)
    #ETF_daily_returns_df
    joined_df = pd.merge(joined_df, ETF_daily_returns_df, left_index=True, right_index=True)
    #joined_df.head()
```

Muni Bond Sample Set by State

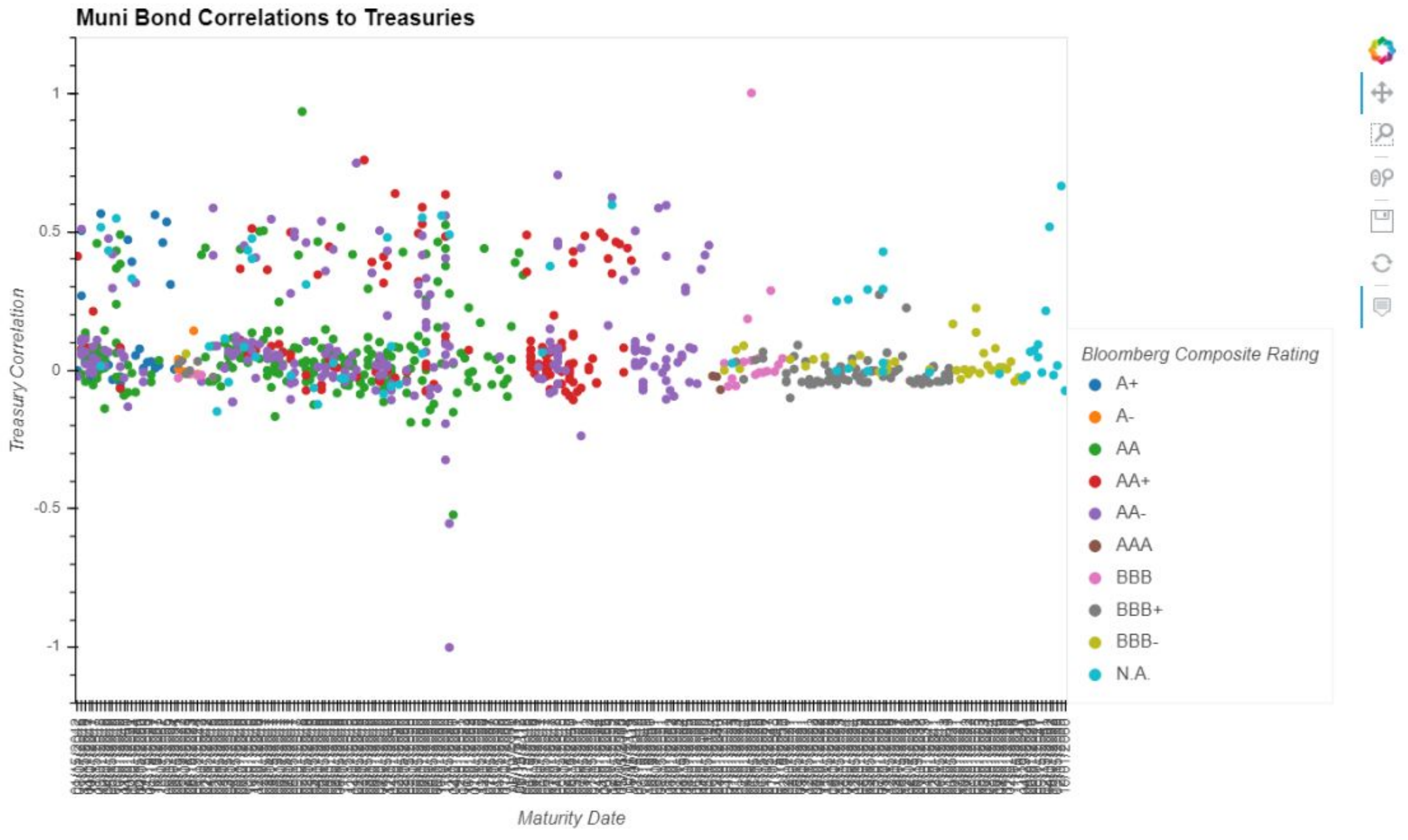


	677561LN0	tsy_change	swap_change	JNK	LQD	MUB	TLT	UDN	UUP
677561LN0	1	0.416049	0.311006	-0.394449	0.132931	0.02003	0.495735	0.00767703	-0.0357252
tsy_change	0.416049	1	0.83515	-0.0193425	0.369592	0.193514	0.66243	0.113587	-0.112852
swap_change	0.311006	0.83515	1	-0.0125036	0.254821	0.153117	0.617693	0.110226	-0.136813
JNK	-0.394449	-0.0193425	-0.0125036	1	0.0533749	0.0503592	-0.0173925	0.0121156	-0.00493745
LQD	0.132931	0.369592	0.254821	0.0533749	1	0.686187	0.548531	0.246054	-0.198178
MUB	0.02003	0.193514	0.153117	0.0503592	0.686187	1	0.382497	0.228248	-0.186376
TLT	0.495735	0.66243	0.617693	-0.0173925	0.548531	0.382497	1	0.158435	-0.179354
UDN	0.00767703	0.113587	0.110226	0.0121156	0.246054	0.228248	0.158435	1	-0.959371
UUP	-0.0357252	-0.112852	-0.136813	-0.00493745	-0.198178	-0.186376	-0.179354	-0.959371	1

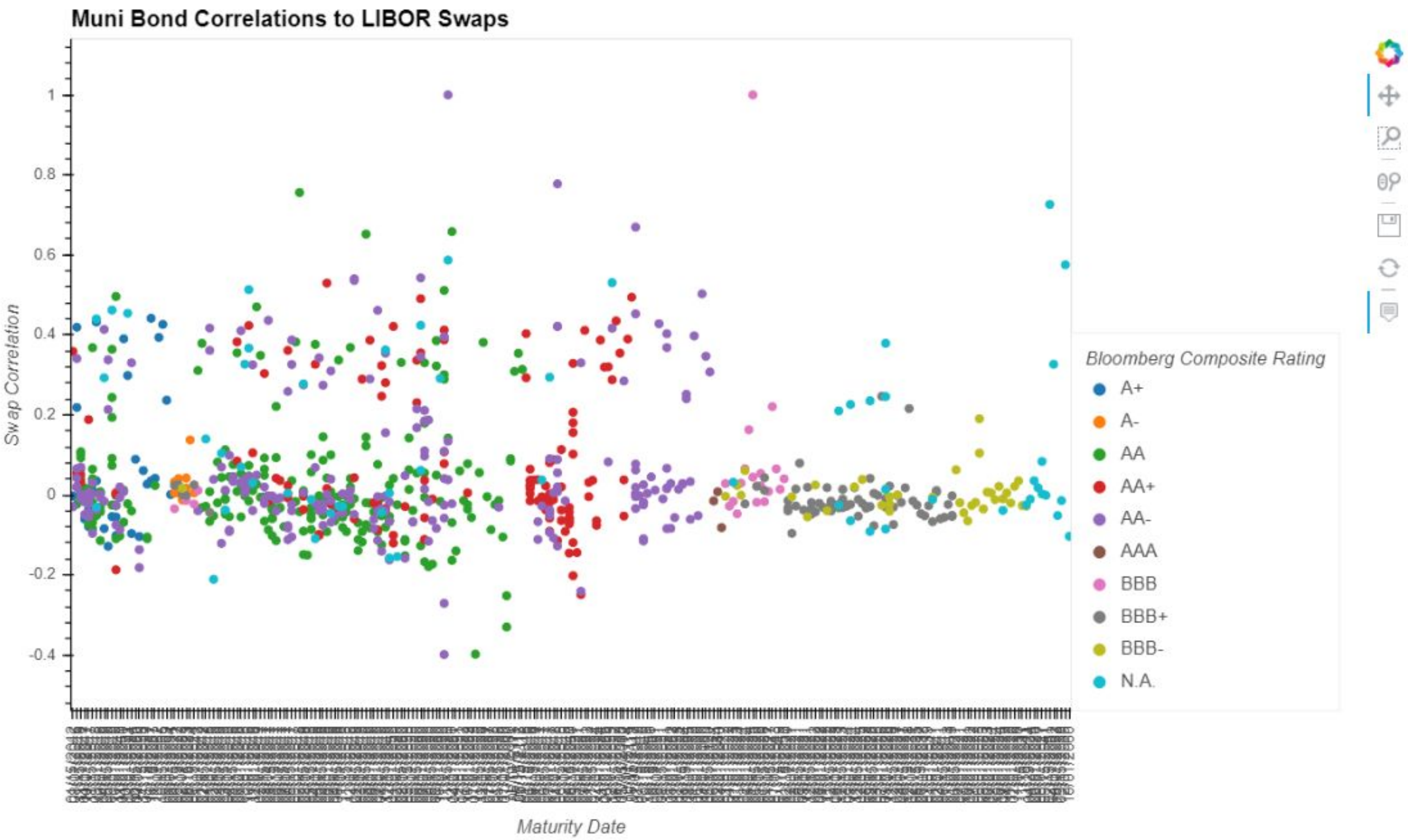


- Muni Bond “677561LNO” Correlation heat map to other hedgers

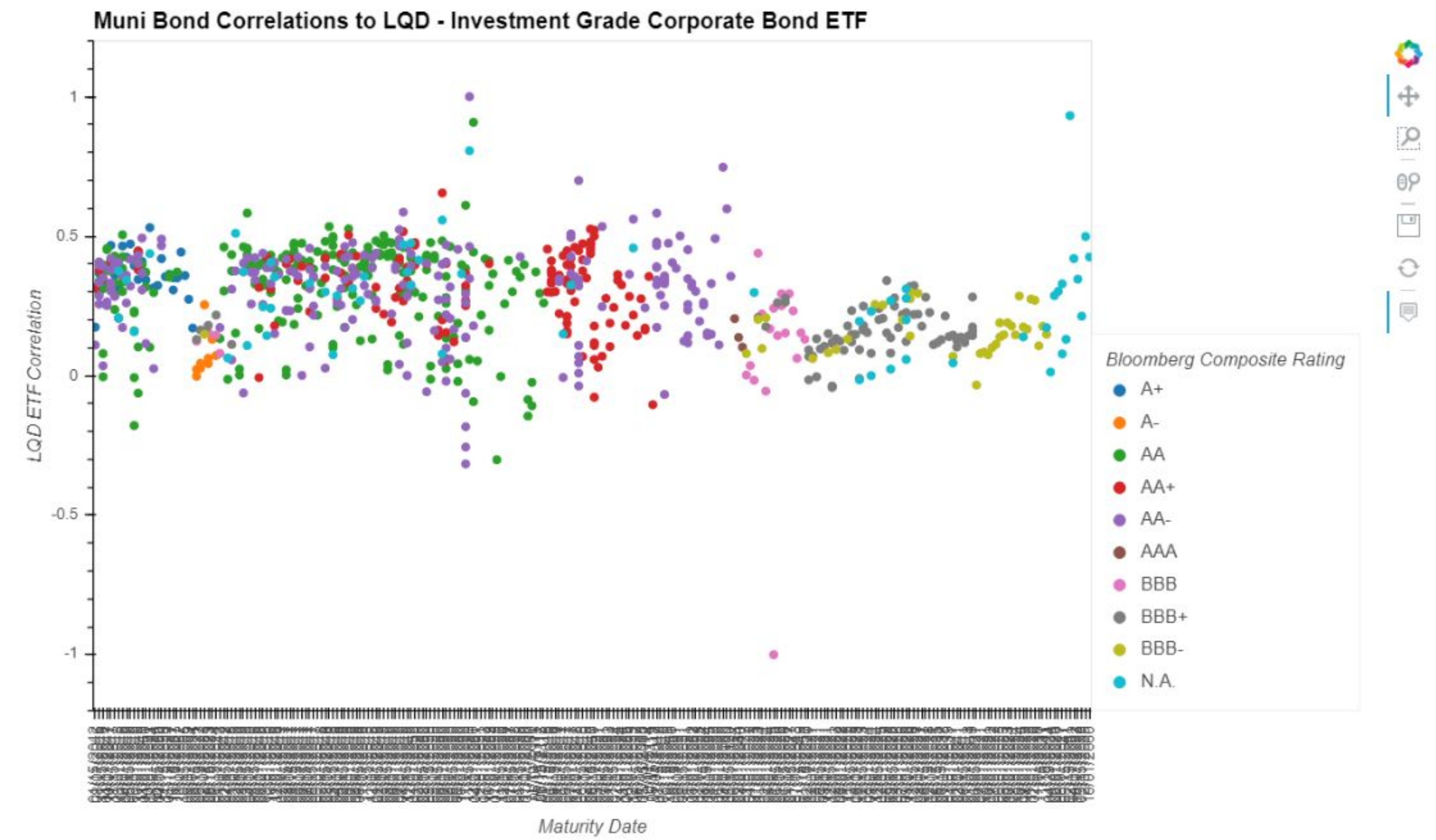
- Overall Muni Bond Correlation to other hedgers

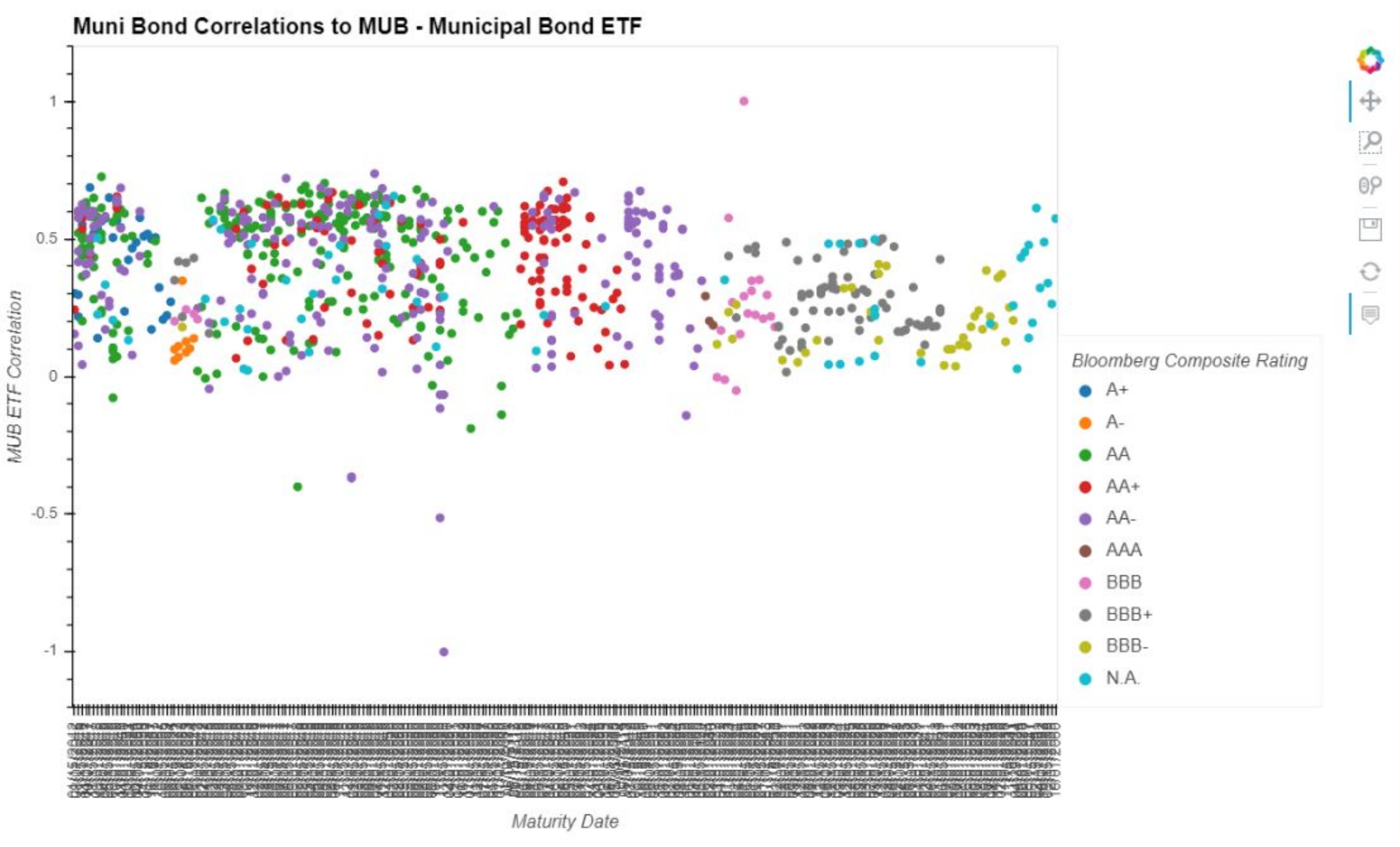


Muni Bond Correlations to Treasuries
Treasury Correlation
1
0.5
0
-0.5
-1
Maturity Date
Bloomberg Composite Rating
A+
A-
AA
AA+
AA-
AAA
BBB
BBB+
BBB-
N.A.

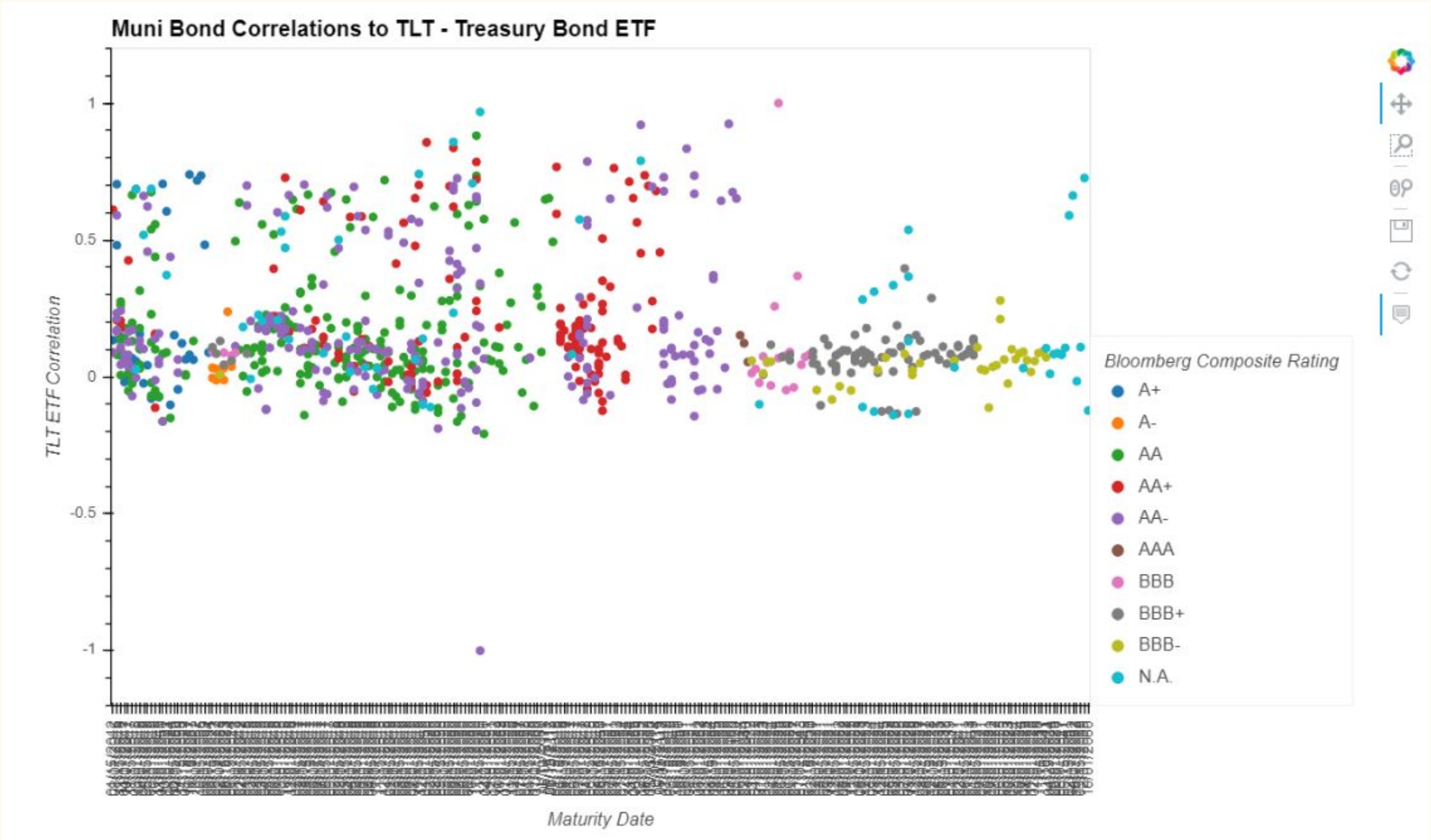


Muni Bond Correlations to LIBOR Swaps
Swap Correlation
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
Maturity Date
Bloomberg Composite Rating
A+
A-
AA
AA+
AA-
AAA
BBB
BBB+
BBB-
N.A.

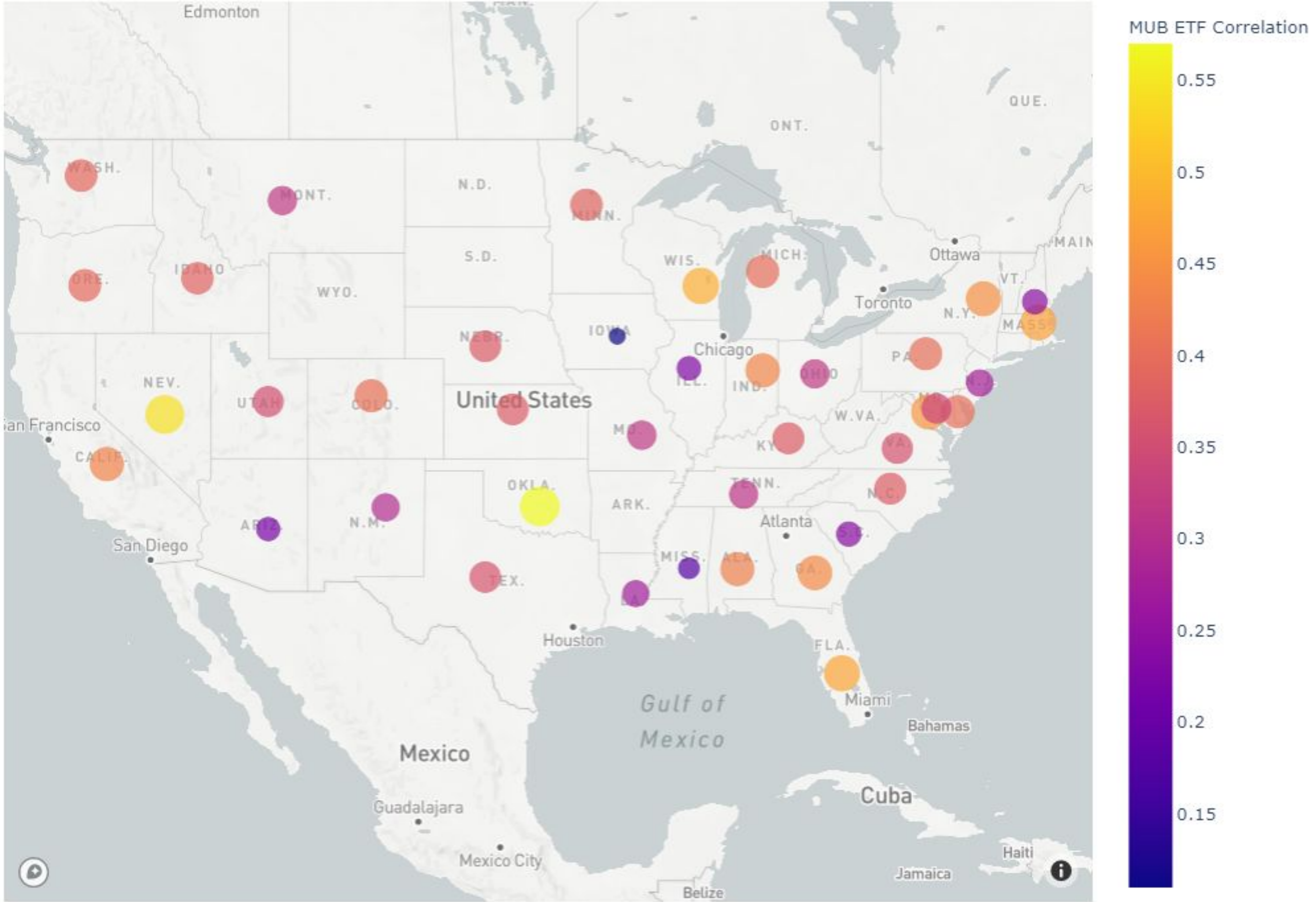

Muni Bond Correlations to LQD - Investment Grade Corporate Bond ETF
LQD ETF Correlation
1
0.5
0
-0.5
-1
Maturity Date
Bloomberg Composite Rating
A+
A-
AA
AA+
AA-
AAA
BBB
BBB+
BBB-
N.A.



Muni Bond Correlations to MUB - Municipal Bond ETF
MUB ETF Correlation
1
0.5
0
-0.5
-1
Maturity Date
Bloomberg Composite Rating
A+
A-
AA
AA+
AA-
AAA
BBB
BBB+
BBB-
N.A.


Muni Bond Correlations to TLT - Treasury Bond ETF
TLT ETF Correlation
1
0.5
0
-0.5
-1
Maturity Date
Bloomberg Composite Rating
A+
A-
AA
AA+
AA-
AAA
BBB
BBB+
BBB-
N.A.

Average MUB ETF Correlation by State



Challenges

- Identifying appropriate hedges
- Building in transaction costs
- Factoring in total return

Conclusions

- Municipal bond traders and relative value managers hedge systematic risk with shorting Treasury bonds
- Shorting MUB may be better than the current industry standard

```
avg_treasury_correlation = muni_bond_characteristics_correlations_df['Treasury Correlation'].mean()
avg_treasury_correlation
```

0.08749208794786831

```
avg_swap_correlation = muni_bond_characteristics_correlations_df['Swap Correlation'].mean()
avg_swap_correlation
```

0.054004128028149585

```
avg_lqd_correlation = muni_bond_characteristics_correlations_df['LQD ETF Correlation'].mean()
avg_lqd_correlation
```

0.28241858163082073

```
avg_mub_correlation = muni_bond_characteristics_correlations_df['MUB ETF Correlation'].mean()
avg_mub_correlation
```

0.3852561171521304

```
avg_tlt_correlation = muni_bond_characteristics_correlations_df['TLT ETF Correlation'].mean()
avg_tlt_correlation
```

0.17200482184276683

	Treasury Correlation	Swap Correlation	LQD ETF Correlation	MUB ETF Correlation	TLT ETF Correlation
count	812.000000	812.000000	812.000000	812.000000	812.000000
mean	0.087492	0.054004	0.282419	0.385256	0.172005
std	0.187325	0.172877	0.166117	0.209231	0.234057
min	-1.000000	-0.398898	-1.000000	-1.000000	-1.000000
25%	-0.017830	-0.039372	0.163549	0.223813	0.032594
50%	0.028647	-0.000602	0.314290	0.419963	0.100384
75%	0.094066	0.068950	0.401622	0.561598	0.210272
max	1.000000	1.000000	1.000000	1.000000	1.000000