# The Practice of QCF

## Liquidity Cost Analysis

In [ ]:
```
import wrds
import datetime
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from matplotlib.dates import DateFormatter
import seaborn as sns
import pandas as pd

# Handle date time conversions between pandas and matplotlib
from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
import statsmodels.formula.api as smf
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/statsmodels/tools/
_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use t
he functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

## Getting Trace data from wrds

In [7]:
```
db = wrds.Connection()
```

```
Enter your WRDS username [deepsha]:madhur5
Enter your password:••••••••
WRDS recommends setting up a .pgpass file.
You can find more info here:
https://www.postgresql.org/docs/9.5/static/libpq-pgpass.html.
Loading library list...
Done
```

In [149]:
```
sql_query = """
SELECT *
FROM trace.trace_enhanced
WHERE CUSIP_ID in ('931142EH2','931142BF9','931142AU7','931142CK7','9311
42CM3','931142CH4','931142DB6','931142CS0','931142CB7','931142CY7','9311
42CV3','931142DK6','931142DQ3','931142DD2','931142EC3','931142DG5','9311
42EB5','931142EE9','931142DW0','931142ED1','931142EK5','931142DP5','9311
42EN9','931142CZ4','931142EJ8','931142EM1','931142EP4','931142EL3','9311
42DV2','931142DH3','931142EQ2','931142DU4','931142EA7')
AND TRD_EXCTN_DT BETWEEN '2009-01-01' AND '2019-12-31';
"""
data_query = db.raw_sql(sql_query)
```

In [151]:
```
len(data_query)
```

Out[151]: 471077

# Dick - Nielson Filtering

In [ ]:
```
data_query['trd_exctn_tm'] = data_query['trd_exctn_tm'].apply(lambda x:
datetime.timedelta(seconds=x))
data_query['trd_rpt_tm'] = data_query['trd_rpt_tm'].apply(lambda x: date
time.timedelta(seconds=x))

### Dick-Nielsen Filtering
# Cancellations, reversals and corrections
data_query = data_query[~data_query['trc_st'].isin(['C','Y','X'])]
#After Market Hours
data_query = data_query[data_query.sale_cndtn2_cd != 'A']
# Filter below 100 percentage pts
data_query = data_query[data_query['rptd_pr'] >= 100]
# Remove agency transactions without commissions
sell_index = data_query[(data_query['rpt_side_cd'] == 'S') & (data_query
['sell_cpcty_cd'] == 'A') & (data_query['cntra_mp_id'] == 'C') & (data_q
uery['cmsn_trd'] == 'N')].index
data_query.drop(sell_index, inplace=True)
buy_index = data_query[(data_query['rpt_side_cd'] == 'B') & (data_query[
'buy_cpcty_cd'] == 'A') & (data_query['cntra_mp_id'] == 'C') & (data_que
ry['cmsn_trd'] == 'N')].index
data_query.drop(buy_index, inplace=True)
# Remove interdealer transactions
interdealer_txn = data_query[(data_query['cntra_mp_id'] == 'D') & (data_
query['rpt_side_cd'] == 'B')].index
data_query.drop(interdealer_txn, inplace=True)
```

In [162]:
```
data_query_subset_rolls = data_query[['cusip_id','bond_sym_id','trd_exct
n_dt','trd_exctn_tm','rptd_pr','rpt_side_cd']]
```

In [164]:
```
data_query_subset_rolls.to_csv(r'data_query_subset_rolls')
```

## finding the cusip with the most transactions

In [8]:
```
most_txn_bond_cusip = data_query.groupby('bond_sym_id').size().sort_values(ascending=False).index[:5].values
# most_txn_bond_cusip
bond_transactions.groupby('bond_sym_id').size().sort_values(ascending=False).index
```

Out[8]:
```
Index(['AFL.GC', 'ADT4117044', 'ADT4104761', 'ADT4192152', 'AES.HU', 'AFL.GB',
       'ADT3997702', 'ADSW4074463', 'ADT3991008', 'ADT3991009', 'AEPI.GG',
       'AESO.GA', 'AEPI.GD', 'AFC.GA', 'AES.GN', 'AES.GQ', 'AES.GM', 'ADCT.GC',
       'AEPI.GB', 'THG.GA', 'AES.GR', 'AES.GO', 'ADCT.GF', 'CG3706278',
       'FNHO3862006', 'AEPI.GE', 'AES.GL', 'AES.IS', 'AEEC.AA', 'ADST.AA',
       'ADCT.GE', 'CG3706277', 'TYC.GA', 'ACLI3862006', 'TEL3672364',
       'TEL3672363', 'AEPI.GC', 'IRWL3706275', 'AFL.GD', 'AVY.IE',
       'PPL3706275', 'ADCT.GD'],
      dtype='object', name='bond_sym_id')
```

## FISD data

In [658]:
```
fisd_data = pd.read_csv('/users/Deepsha/Downloads/fisd.csv')
fisd_data.head()
```

Out[658]:

| | Unnamed: 0 | COMPLETE_CUSIP | MATURITY | COUPON_TYPE | COUPON | PROSPECTUS_ISSUER_N |
|---|---|---|---|---|---|---|
| **0** | 0 | 000361AA3 | 2001-11-01 | F | 9.50 | AAR C |
| **1** | 1 | 000361AB1 | 2003-10-15 | F | 7.25 | AAR C |
| **2** | 2 | 00077DAB5 | 1996-01-12 | F | 4.15 | ABN AMRO BK N V N Y |
| **3** | 3 | 00077DAF6 | 2009-08-01 | F | 8.25 | ABN AMRO BK N V N Y |
| **4** | 4 | 00077TAA2 | 2023-05-15 | F | 7.75 | ABN AMRO BK N V N Y |

5 rows × 33 columns

In [23]:
```
sql_query = """
SELECT CUSIP_ID as "CUSIP_ID",TRD_EXCTN_DT as "TRD_EXCTN_DT",
    max(TRD_EXCTN_TM) as "TRD_EXCTN_TM",
    max(RPTD_PR) as "RPTD_PR",
    sum(ENTRD_VOL_QT) as "ENTRD_VOL_QT",
    max(RPT_SIDE_CD) as "RPT_SIDE_CD",
    max(TRC_ST) as "TRC_ST",
    max(ASOF_CD) as "ASOF_CD",
    max(MSG_SEQ_NB) as "MSG_SEQ_NB",
    max(ORIG_MSG_SEQ_NB) as "ORIG_MSG_SEQ_NB"
    from trace.trace_enhanced
    WHERE TRD_EXCTN_DT>'2010-01-01'
    group by CUSIP_ID,TRD_EXCTN_DT
    order by CUSIP_ID,TRD_EXCTN_DT;
"""
q2 = db.raw_sql(sql_query)
```

In [24]:
```
q2.head()
```

Out[24]:

| | CUSIP_ID | TRD_EXCTN_DT | TRD_EXCTN_TM | RPTD_PR | ENTRD_VOL_QT | RPT_SIDE_CD | TRC |
|---|---|---|---|---|---|---|---|
| 0 | 000305AB8 | 2010-01-22 | 44651.0 | 100.00 | 64000.0 | S | |
| 1 | 000305AB8 | 2010-01-29 | 52684.0 | 100.00 | 64000.0 | B | |
| 2 | 000305AB8 | 2010-02-09 | 32311.0 | 98.50 | 60000.0 | B | |
| 3 | 000305AB8 | 2010-04-16 | 33784.0 | 99.25 | 20000.0 | B | |
| 4 | 000305AB8 | 2010-09-22 | 34440.0 | 98.50 | 347900.0 | B | |

# Walmart Bond Liquidity Analysis

In [312]:
```
cleaned_wmt = pd.read_csv("Cleaned_wmt.csv")
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/IPython/core/inter
activeshell.py:3057: DtypeWarning:

Columns (8,13,16,24) have mixed types.Specify dtype option on import or
set low_memory=False.
```

In [313]:
```
cleaned_wmt
```

Out[313]:

| | cusip_id | bond_sym_id | company_symbol | trd_exctn_dt | trd_exctn_tm | trd_rpt_dt | trd_rp |
|---|---|---|---|---|---|---|---|
| **0** | 931142DB6 | WMT.AB | WMT | 20110412 | 8:51:13 | 20110412 | 8:5 |
| **1** | 931142DB6 | WMT.AB | WMT | 20110412 | 10:25:08 | 20110412 | 10:3 |
| **2** | 931142DB6 | WMT.AB | WMT | 20110412 | 11:24:00 | 20110412 | 11:2 |
| **3** | 931142DB6 | WMT.AB | WMT | 20110412 | 11:24:00 | 20110412 | 11:2 |
| **4** | 931142DB6 | WMT.AB | WMT | 20110412 | 11:27:30 | 20110412 | 11:2 |
| **...** | ... | ... | ... | ... | ... | ... | |
| **290276** | 931142EP4 | WMT4887055 | WMT | 20200331 | 12:58:58 | 20200331 | 12:5 |
| **290277** | 931142EP4 | WMT4887055 | WMT | 20200331 | 12:58:58 | 20200331 | 12:5 |
| **290278** | 931142EP4 | WMT4887055 | WMT | 20200331 | 16:20:36 | 20200331 | 16:2 |
| **290279** | 931142EP4 | WMT4887055 | WMT | 20200331 | 16:41:55 | 20200331 | 16:4 |
| **290280** | 931142EP4 | WMT4887055 | WMT | 20200331 | 17:21:51 | 20200331 | 17:2 |

290281 rows × 26 columns

# Calculation of Rolls Measure

In [314]:
```
def lagged_auto_cov(Xi,t):

    N = len(Xi)
    Xs = np.mean(Xi)
    end_padded_series = np.zeros(N+t)
    end_padded_series[:N] = Xi - Xs
    start_padded_series = np.zeros(N+t)
    start_padded_series[t:] = Xi - Xs

    auto_cov = 1./(N-1) * np.sum( start_padded_series*end_padded_series
)
    return auto_cov
```

In [315]:
```
cleaned_wmt["trd_exctn_dt"] = pd.to_datetime(cleaned_wmt["trd_exctn_dt"
], format="%Y%m%d")
```

In [317]:
```
uniques_cusips = cleaned_wmt.cusip_id.unique()
df_rs_wmt = pd.DataFrame(columns = ['cusip', 'date', 'auto_cov','rolls_s
pread'])
for k in uniques_cusips:
    x = cleaned_wmt[cleaned_wmt.cusip_id == k]
    auto_cov = []
    rolls_spread = []
    uniques_dates = x.trd_exctn_dt.unique()
    for i in range(0,len(uniques_dates)):
        p = x[x.trd_exctn_dt == uniques_dates[i]]['rptd_pr']
        d = np.diff(p)
        if len(d) == 0 or len(d) == 1 :
            a = np.nan
            r = np.nan
            auto_cov.append(np.nan)
            rolls_spread.append(np.nan)
        else:
            a = lagged_auto_cov(d,1)
            r = 2*np.sqrt(-a)
            auto_cov.append(a)
            rolls_spread.append(r)
        data = pd.DataFrame({'cusip':k ,'date':uniques_dates[i],'auto_co
v': a,'rolls_spread' : r },index=[0])
        df_rs_wmt = df_rs_wmt.append(data)
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:18: RuntimeWarning:

invalid value encountered in sqrt
```

In [659]:
```
df_rs_wmt.head()
```

Out[659]:

| | COMPLETE_CUSIP | trd_exctn_dt | auto_cov | rolls_spread | year | month | year_month |
|---|---|---|---|---|---|---|---|
| **0** | 931142DB6 | 2011-04-12 | -0.117809 | 0.686466 | 2011 | 4 | 20114 |
| **0** | 931142DB6 | 2011-04-13 | -0.008013 | 0.179029 | 2011 | 4 | 20114 |
| **0** | 931142DB6 | 2011-04-14 | -0.467709 | 1.367784 | 2011 | 4 | 20114 |
| **0** | 931142DB6 | 2011-04-15 | -0.013290 | 0.230564 | 2011 | 4 | 20114 |
| **0** | 931142DB6 | 2011-04-18 | -0.982859 | 1.982784 | 2011 | 4 | 20114 |

In [319]:
```
df_rs_wmt['year'] = pd.DatetimeIndex(df_rs_wmt['date']).year
df_rs_wmt['month'] = pd.DatetimeIndex(df_rs_wmt['date']).month
df_rs_wmt["year_month"] = df_rs_wmt['year'].astype(str) + df_rs_wmt['mon
th'].astype(str)
```

In [320]:
```
df_rs_wmt.dtypes
```

Out[320]:
```
cusip                    object
date            datetime64[ns]
auto_cov                float64
rolls_spread            float64
year                      int64
month                     int64
year_month               object
dtype: object
```

In [321]:
```
g = df_rs_wmt.groupby(["year_month"])
monthly_averages = g.aggregate({"rolls_spread":np.mean})
```

In [322]:
```
monthly_averages.head()
```

Out[322]:

| year_month | rolls_spread |
|---|---|
| **20071** | 2.054803 |
| **200710** | 1.957956 |
| **200711** | 2.065796 |
| **200712** | 2.342166 |
| **20072** | 2.071908 |

In [324]:
```
monthly_averages['date'] = monthly_averages.index
monthly_averages['Date']  = pd.to_datetime(monthly_averages['date'], for
mat='%Y%m')
monthly_averages = monthly_averages.sort_values(by="Date")
```

In [327]:
```
monthly_averages.unstack(level =1).dropna()
```

Out[327]:
```
              year_month
rolls_spread  20071                     2.0548
              20072                    2.07191
              20073                     1.5568
              20074                     1.6639
              20075                    1.68826
                                 ...
Date          201911        2019-11-01 00:00:00
              201912        2019-12-01 00:00:00
              20201         2020-01-01 00:00:00
              20202         2020-02-01 00:00:00
              20203         2020-03-01 00:00:00
Length: 477, dtype: object
```

In [191]:
```
df_rs
```

Out[191]:

| | rolls_spread | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| month | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| year | | | | | | | | | |
| 2007 | 2.054803 | 2.071908 | 1.556803 | 1.663898 | 1.688256 | 1.912134 | 2.608851 | 1.615786 | 1.985052 |
| 2008 | 2.029783 | 2.244261 | 2.453255 | 2.219971 | 2.406434 | 2.092986 | 2.745232 | 2.866189 | 2.318209 |
| 2009 | 2.514892 | 2.605601 | 2.611524 | 2.780327 | 2.650243 | 2.611532 | 2.088986 | 1.983868 | 1.776198 |
| 2010 | 1.341853 | 1.740805 | 1.321628 | 1.802177 | 2.018981 | 1.979389 | 1.814921 | 1.645156 | 1.425124 |
| 2011 | 1.640544 | 1.426926 | 1.497672 | 1.294161 | 1.294624 | 1.387083 | 1.552728 | 1.645759 | 1.889168 |
| 2012 | 1.581219 | 1.527863 | 1.279952 | 1.068730 | 1.257127 | 1.274706 | 1.304365 | 1.090537 | 1.023637 |
| 2013 | 0.901313 | 0.939662 | 0.941415 | 0.925082 | 1.020712 | 1.310737 | 0.976045 | 1.023553 | 1.020343 |
| 2014 | 0.842588 | 1.105697 | 0.803572 | 0.881554 | 0.796003 | 0.943337 | 0.925418 | 0.891254 | 0.826172 |
| 2015 | 0.849146 | 0.856662 | 0.913300 | 0.817295 | 0.758596 | 0.797849 | 1.192001 | 0.885466 | 0.851538 |
| 2016 | 0.746235 | 0.913059 | 0.887744 | 0.812485 | 0.854331 | 0.849817 | 0.903953 | 0.868819 | 0.808108 |
| 2017 | 0.682234 | 0.893176 | 0.598404 | 0.709409 | 0.645541 | 0.670067 | 0.704391 | 0.630452 | 0.575980 |
| 2018 | 0.606588 | 1.004431 | 0.635912 | 0.910718 | 0.862186 | 0.425905 | 0.455956 | 0.504976 | 0.510923 |
| 2019 | 0.578460 | 0.520910 | 0.551238 | 0.511739 | 0.466513 | 0.394595 | 0.462972 | 0.445645 | 0.484216 |
| 2020 | 0.429784 | 0.390116 | 1.326876 | NaN | NaN | NaN | NaN | NaN | NaN |

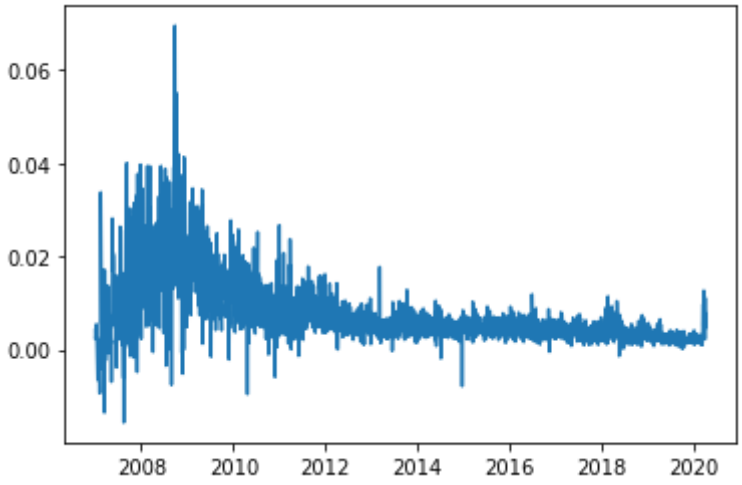
# Bid - Ask Liquidity Cost

In [ ]:
```
data_query_subset_lc = cleaned_wmt.groupby(['cusip_id','trd_exctn_dt','r
pt_side_cd'])['rptd_pr'].mean().unstack(level = 2).dropna()
data_query_subset_lc['LC'] = (data_query_subset_lc['S']- data_query_subs
et_lc['B'])/(0.5*(data_query_subset_lc['S']+ data_query_subset_lc['B']))
```

In [508]:
```
df_lc  = data_query_subset_lc.groupby(['trd_exctn_dt']).mean()
```

In [509]:
```
plt.plot(df_lc.LC)
```

Out[509]:
```
[<matplotlib.lines.Line2D at 0x168b99668>]
```




In [332]:
```
df_lc['date'] = df_lc.index
df_lc['year'] = pd.DatetimeIndex(df_lc['date']).year
df_lc['month'] = pd.DatetimeIndex(df_lc['date']).month
df_lc["year_month"] = df_lc['year'].astype(str) + df_lc['month'].astype(
str)
```

In [333]:
```
df_lc
```

Out[333]:

| rpt_side_cd<br>trd_exctn_dt | B | S | LC | date | year | month | year_month |
|---|---|---|---|---|---|---|---|
| **2007-01-03** | 121.338000 | 121.838380 | 0.004115 | 2007-01-03 | 2007 | 1 | 20071 |
| **2007-01-04** | 116.006250 | 116.276385 | 0.002209 | 2007-01-04 | 2007 | 1 | 20071 |
| **2007-01-05** | 121.933000 | 122.259000 | 0.002670 | 2007-01-05 | 2007 | 1 | 20071 |
| **2007-01-10** | 121.890000 | 122.370600 | 0.003935 | 2007-01-10 | 2007 | 1 | 20071 |
| **2007-01-12** | 120.209999 | 120.864500 | 0.005430 | 2007-01-12 | 2007 | 1 | 20071 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **2020-03-25** | 109.806382 | 110.059774 | 0.002182 | 2020-03-25 | 2020 | 3 | 20203 |
| **2020-03-26** | 113.377161 | 114.695204 | 0.011080 | 2020-03-26 | 2020 | 3 | 20203 |
| **2020-03-27** | 115.707856 | 116.294219 | 0.004962 | 2020-03-27 | 2020 | 3 | 20203 |
| **2020-03-30** | 113.312589 | 114.162816 | 0.007273 | 2020-03-30 | 2020 | 3 | 20203 |
| **2020-03-31** | 114.946111 | 115.504779 | 0.004742 | 2020-03-31 | 2020 | 3 | 20203 |

3190 rows × 7 columns

# Liquidity Cost Monthly Average

In [334]:
```
df_lc = df_lc.groupby(["year_month"])
df_lc_ma = df_lc.aggregate({"LC":np.mean})
```

In [335]:
```
df_lc_ma
```

Out[335]:

| year_month | LC |
|---|---|
| 20071 | 0.000511 |
| 200710 | 0.012864 |
| 200711 | 0.019669 |
| 200712 | 0.025485 |
| 20072 | 0.004763 |
| ... | ... |
| 20198 | 0.002376 |
| 20199 | 0.001749 |
| 20201 | 0.002083 |
| 20202 | 0.001887 |
| 20203 | 0.005801 |

159 rows × 1 columns

# Merging Rolls spread and Liquidity Cost

In [336]:
```
df_lc_rs = df_lc_ma.join(monthly_averages, how='inner')
```

In [339]:
```
df_lc_rs.head()
```

Out[339]:

| year_month | LC | rolls_spread | date | Date |
|---|---|---|---|---|
| 20071 | 0.000511 | 2.054803 | 20071 | 2007-01-01 |
| 20072 | 0.004763 | 2.071908 | 20072 | 2007-02-01 |
| 20073 | 0.002092 | 1.556803 | 20073 | 2007-03-01 |
| 20074 | 0.005565 | 1.663898 | 20074 | 2007-04-01 |
| 20075 | 0.009949 | 1.688256 | 20075 | 2007-05-01 |

In [49]:
```
df_lc_rs['_year_month'] = df_lc_rs.index
df_lc_rs['Date']  = pd.to_datetime(df_lc_rs['_year_month'], format='%Y%m')
```

In [338]:
```
df_lc_rs = df_lc_rs.sort_values(by="Date")
```

In [60]:
```
df_lc_ma['_year_month'] = df_lc_ma.index
df_lc_ma['Date']  = pd.to_datetime(df_lc_ma['_year_month'], format='%Y%m')
df_lc_ma = df_lc_ma.sort_values(by="Date")
```

In [62]:
```
df_lc_ma.head()
```

Out[62]:

| year_month | LC | _year_month | Date |
|---|---|---|---|
| **20071** | 0.000511 | 20071 | 2007-01-01 |
| **20072** | 0.004763 | 20072 | 2007-02-01 |
| **20073** | 0.002092 | 20073 | 2007-03-01 |
| **20074** | 0.005565 | 20074 | 2007-04-01 |
| **20075** | 0.009949 | 20075 | 2007-05-01 |

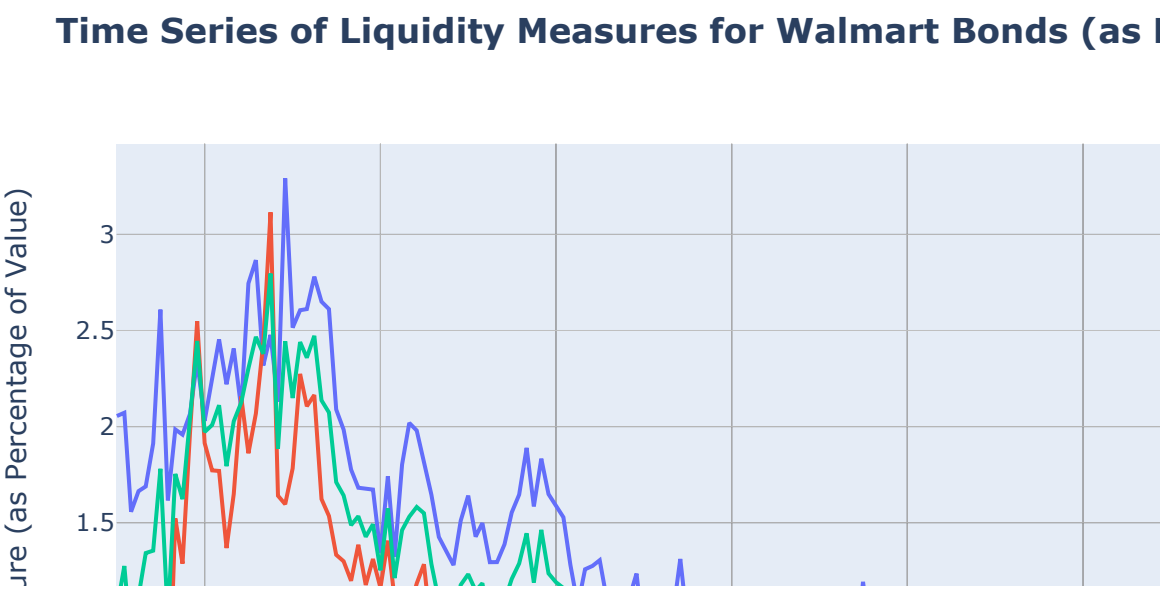
# Average of Rolls Spread and Liquidity cost

In [661]:
```
df_lc_rs["rs_and_LC_average"] = (df_lc_rs['rolls_spread'] + df_lc_rs['LC']*100)/2
df_lc_rs.head()
```

Out[661]:

| year_month | LC | rolls_spread | date | Date | rs_and_LC_average |
|---|---|---|---|---|---|
| **20071** | 0.000511 | 2.054803 | 20071 | 2007-01-01 | 1.052958 |
| **20072** | 0.004763 | 2.071908 | 20072 | 2007-02-01 | 1.274095 |
| **20073** | 0.002092 | 1.556803 | 20073 | 2007-03-01 | 0.883022 |
| **20074** | 0.005565 | 1.663898 | 20074 | 2007-04-01 | 1.110200 |
| **20075** | 0.009949 | 1.688256 | 20075 | 2007-05-01 | 1.341571 |

In [485]:
```
import plotly.express as px
import plotly.graph_objects as go
fig = go.Figure(data=go.Scatter(x=df_lc_rs['Date'], y=df_lc_rs['rolls_sp
read'],name = 'Rolls Spread'),layout=go.Layout(
        title=go.layout.Title(text='<b>Time Series of Liquidity Measures
for Walmart Bonds (as Percentage of Value)</b>')))
fig.add_scatter(x=df_lc_rs['Date'], y=df_lc_rs['LC']*100, mode='lines',n
ame = 'Liquidity Cost(Bid-Ask Spread)')
fig.add_scatter(x=df_lc_rs['Date'], y=df_lc_rs["rs_and_LC_average"], mod
e='lines',name = 'Average of Rolls Spread and Liquidity Cost')
fig.update_layout(legend_title_text = "<b>Liquidity Measures</b>")
fig.update_xaxes(title_text="years -->")
fig.update_yaxes(title_text="Liquidity Measure (as Percentage of Value)"
)
fig.show("notebook")
```




## Correlation of Rolls spread and Liquidity Spread

In [223]:
```
df_lc_rs['rolls_spread'].corr(df_lc_rs['LC'])
```

Out[223]: 0.8615707694938298

# Bond Characteristics Calculations

In [359]:
```
cleaned_wmt = cleaned_wmt.rename(columns={"cusip_id": "COMPLETE_CUSIP"})
cleaned_wmt_fisd = pd.merge(cleaned_wmt,fisd_data, how = "inner", on =
"COMPLETE_CUSIP")
cleaned_wmt_fisd["trd_exctn_dt"] = pd.to_datetime(cleaned_wmt_fisd ["trd
_exctn_dt"], format="%Y%m%d")
```

In [362]:
```
import numpy as np
import scipy
cleaned_wmt_fisd['MATURITY'] = cleaned_wmt_fisd['MATURITY'].apply(pd.to_
datetime)
cleaned_wmt_fisd['Days_to_Maturity'] = (cleaned_wmt_fisd['MATURITY']- cl
eaned_wmt_fisd['trd_exctn_dt'])
cleaned_wmt_fisd['N_MATURITY'] = ((cleaned_wmt_fisd['Days_to_Maturity']
/ 365) * 2)/np.timedelta64(1, 'D')
cleaned_wmt_fisd = cleaned_wmt_fisd[cleaned_wmt_fisd['Days_to_Maturity']
> pd.Timedelta(0,'D')]


yld = {}
for i in range(0, len(cleaned_wmt_fisd)):
    try:
        yld.update({i:YieldCalc(cleaned_wmt_fisd['COUPON'].iloc[i]/(clea
ned_wmt_fisd['PRINCIPAL_AMT'].iloc[i]/10),cleaned_wmt_fisd['rptd_pr'].il
oc[i],
                                cleaned_wmt_fisd['PRINCIPAL_AMT'].iloc[i
]/10, 2.0,
                                cleaned_wmt_fisd['N_MATURITY'].iloc[i],
cleaned_wmt_fisd['COUPON'].iloc[i])})
    except(RuntimeError):
        pass
    else:
        pass

l = []
for i in yld:
    l.append(yld[i])

cleaned_wmt_fisd['YTM'] = pd.Series(l, index=cleaned_wmt_fisd.index)
```

In [363]:
```
cleaned_wmt_fisd.head()
```

Out[363]:

| | COMPLETE_CUSIP | bond_sym_id | company_symbol | trd_exctn_dt | trd_exctn_tm | trd_rpt_dt | trd_ |
|---|---|---|---|---|---|---|---|
| **0** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 8:51:13 | 20110412 | |
| **1** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 10:25:08 | 20110412 | 1 |
| **2** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:24:00 | 20110412 | 1 |
| **3** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:24:00 | 20110412 | 1 |
| **4** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:27:30 | 20110412 | 1 |

5 rows × 61 columns

# Merging Fisd and Walmart data

In [665]:
```
cleaned_wmt_fisd.head()
```

Out[665]:

| | COMPLETE_CUSIP | bond_sym_id | company_symbol | trd_exctn_dt | trd_exctn_tm | trd_rpt_dt | trd_ |
|---|---|---|---|---|---|---|---|
| **0** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 8:51:13 | 2011-04-12 | |
| **1** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 10:25:08 | 2011-04-12 | 1 |
| **2** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:24:00 | 2011-04-12 | 1 |
| **3** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:24:00 | 2011-04-12 | 1 |
| **4** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:27:30 | 2011-04-12 | 1 |

5 rows × 66 columns

# Duration Calculation

In [364]:
```
def duration(coupon,YTM,freq,years,Price,Princple):
    period=1/freq
    N=np.floor(years)/period+1
    T=years-np.floor(years)+0.5*np.arange(N)
    M=np.sum([coupon*t/pow(1+YTM,t) for t in T])+Princple*years/pow(1+YTM,years)
    return M/(Price*10)

def days2years(d):
    return d.days/365.25

def calculate_duration(data):
    # Calculate Duration
    #data1 = pd.read_csv('trace_fisd_merged.csv')  #Change directory as necessary

    data['trd_rpt_dt']=pd.to_datetime(data['trd_rpt_dt'],format='%Y%m%d')
    data['MATURITY']=pd.to_datetime(data['MATURITY'])
    data['diff'] = data['MATURITY']-data['trd_rpt_dt']
    data['years_TM']=data.apply(lambda row: days2years(row['diff']),axis=1)
    data['semicoupon']=data['COUPON']/100*data['PRINCIPAL_AMT']/data['INTEREST_FREQUENCY']
    data['Duration_MAC']=data.apply(lambda row:duration(row['semicoupon'],row['YTM'],\
                                                        row['INTEREST_FREQUENCY'],row['years_TM'],\
                                                        row['rptd_pr'],row['PRINCIPAL_AMT']),axis=1)

    data['Duration_MOD']=data['Duration_MAC']/(1+data['YTM'])
    return data
```

In [365]:
```
test = calculate_duration(cleaned_wmt_fisd)
```

In [666]:
```
merged_df.head()
```

Out[666]:

| | COMPLETE_CUSIP | bond_sym_id | company_symbol | trd_exctn_dt | trd_exctn_tm | trd_rpt_dt | trd_ |
|---|---|---|---|---|---|---|---|
| **0** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 8:51:13 | 2011-04-12 | |
| **1** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 10:25:08 | 2011-04-12 | 1 |
| **2** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:24:00 | 2011-04-12 | 1 |
| **3** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:24:00 | 2011-04-12 | 1 |
| **4** | 931142DB6 | WMT.AB | WMT | 2011-04-12 | 11:27:30 | 2011-04-12 | 1 |

5 rows × 71 columns

In [366]:
```
df_rs_wmt = df_rs_wmt.rename(columns={"cusip": "COMPLETE_CUSIP", "date":
"trd_exctn_dt"})
merged_df = test.merge(df_rs_wmt, how='inner', on=["COMPLETE_CUSIP", "tr
d_exctn_dt"])
```

In [ ]:
```
test1 = merged_df.groupby(["trd_exctn_dt"]).mean()
test1 = test1.dropna()
test1 = test1[["rolls_spread"]]
```

# 30 yr Treasury Yield

In [90]:
```
tr30 = pd.read_csv("DGS30.csv")
tr30 = tr30.rename(columns={"DATE": "trd_exctn_dt"})
tr30["trd_exctn_dt"] = pd.to_datetime(tr30["trd_exctn_dt"])
```

In [98]:
```
tr30 = tr30[tr30['DGS30'] != "."]
tr30
```

Out[98]:

| | trd_exctn_dt | DGS30 |
|---|---|---|
| **0** | 2007-01-02 | 4.79 |
| **1** | 2007-01-03 | 4.77 |
| **2** | 2007-01-04 | 4.72 |
| **3** | 2007-01-05 | 4.74 |
| **4** | 2007-01-08 | 4.74 |
| **...** | ... | ... |
| **3601** | 2020-10-21 | 1.62 |
| **3602** | 2020-10-22 | 1.67 |
| **3603** | 2020-10-23 | 1.64 |
| **3604** | 2020-10-26 | 1.59 |
| **3605** | 2020-10-27 | 1.57 |

3460 rows × 2 columns

In [99]:
```
test1_tr30= pd.merge(test1,tr30, how ="inner", on = "trd_exctn_dt")
test1_tr30["spread"] = (test1_tr30["DGS30"].astype(float)/100) - test1_t
r30["YTM"]
test1_tr30["spread_Duration"] = test1_tr30["spread"]*test1_tr30["Duratio
n_MAC"]
```

In [100]:
```
test1_tr30_sub = test1_tr30[["rolls_spread", "spread", "spread_Duration"
]]
```

# Age Calculation

In [ ]:
```
#import datetime
merged_df['OFFERING_DATE'] = pd.to_datetime(merged_df['OFFERING_DATE'])
merged_df['Age'] = (merged_df['trd_exctn_dt'] - merged_df['OFFERING_DAT
E']).apply(lambda x: (x.days)/365)
merged_df_ = merged_df.dropna(subset=['rolls_spread'])
merged_df_age = merged_df_.groupby(["COMPLETE_CUSIP"])['rolls_spread',
"Age"].mean()
```

In [113]:
```
merged_df['trd_exctn_dt'] = pd.to_datetime(merged_df['trd_exctn_dt'])
```

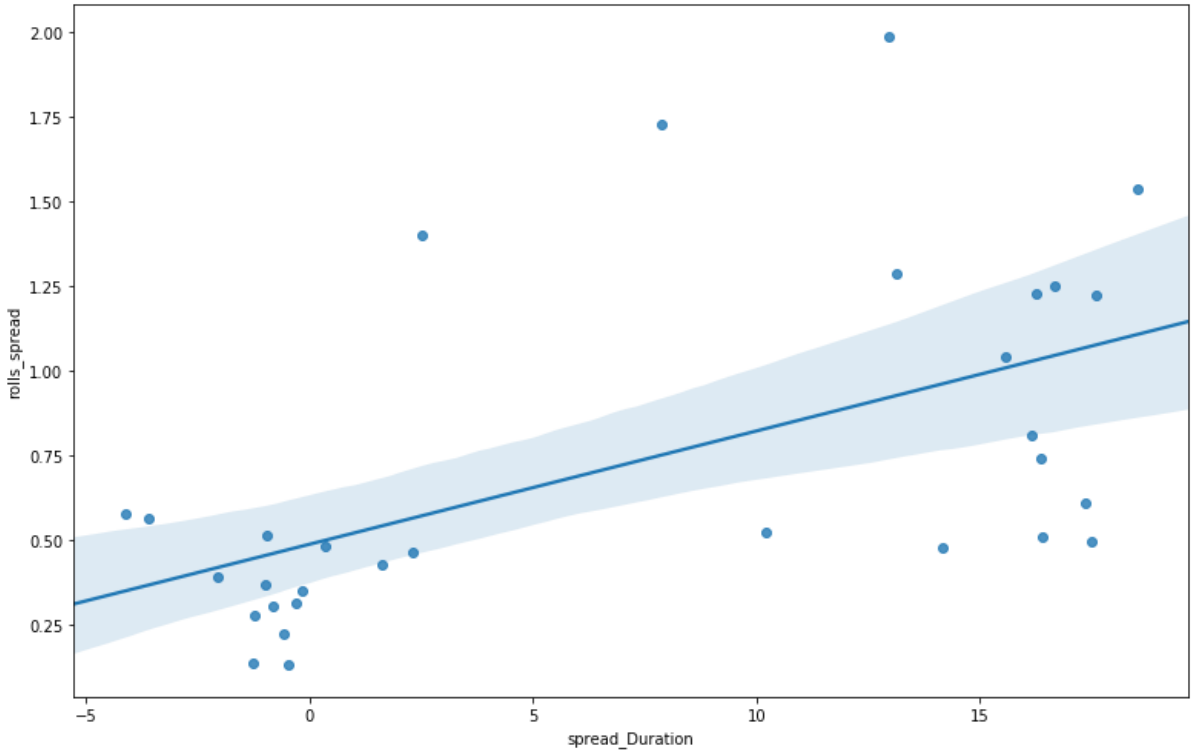
# Spread Duration and Issue size Calculation

In [123]:
```
merged_df_tr30= pd.merge(merged_df_,tr30, how ="inner", on = "trd_exctn_
dt")
merged_df_tr30["spread"] =  merged_df_tr30["YTM"] - (merged_df_tr30["DGS
30"].astype(float)/100)
merged_df_tr30["spread_Duration"] = merged_df_tr30["spread"]*merged_df_t
r30["Duration_MAC"]*100
merged_df_tr30["issue_size"] = merged_df_tr30["OFFERING_AMT"]*merged_df_
tr30["PRINCIPAL_AMT"]
merged_df_spreadD = merged_df_tr30.groupby(["COMPLETE_CUSIP"])['rolls_sp
read', "spread_Duration"].mean()
```

In [ ]:
```
merged_df_issue_size = merged_df_tr30.groupby(["COMPLETE_CUSIP"])['rolls
_spread', "issue_size"].mean()
merged_df_issue_size["issue_size"] = merged_df_issue_size["issue_size"]/
1000000000
sns.jointplot(x=merged_df_issue_size["issue_size"], height = 8, y=merged
_df_issue_size["rolls_spread"],color= "black", kind="reg", stat_func=r2)
```

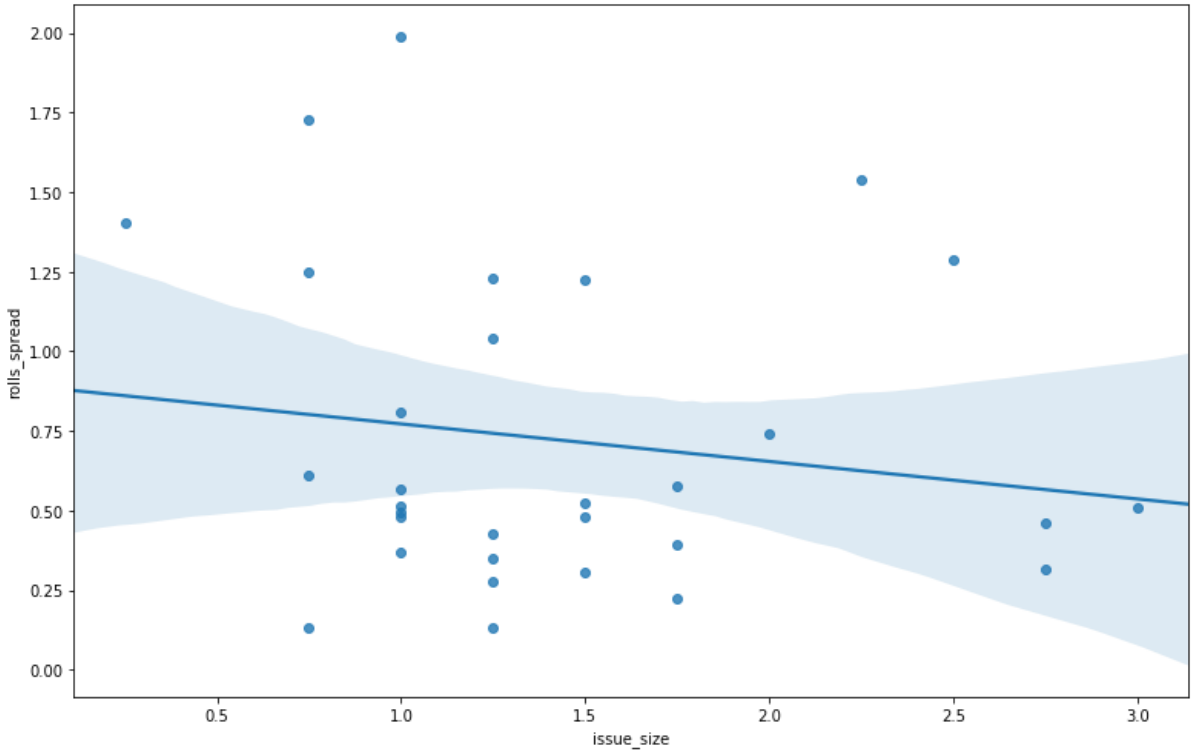
In [195]:
```
a4_dims = (13, 8.27)
fig, ax = pyplot.subplots(figsize=a4_dims)
sns.regplot(x=merged_df_spreadD["spread_Duration"], y=merged_df_spreadD[
"rolls_spread"])
```

Out[195]: <matplotlib.axes._subplots.AxesSubplot at 0x173dc74a8>



In [216]:
```
a4_dims = (13, 8.27)
fig, ax = pyplot.subplots(figsize=a4_dims)
sns.regplot(x=merged_df_issue_size["issue_size"], y=merged_df_issue_size
["rolls_spread"])
```

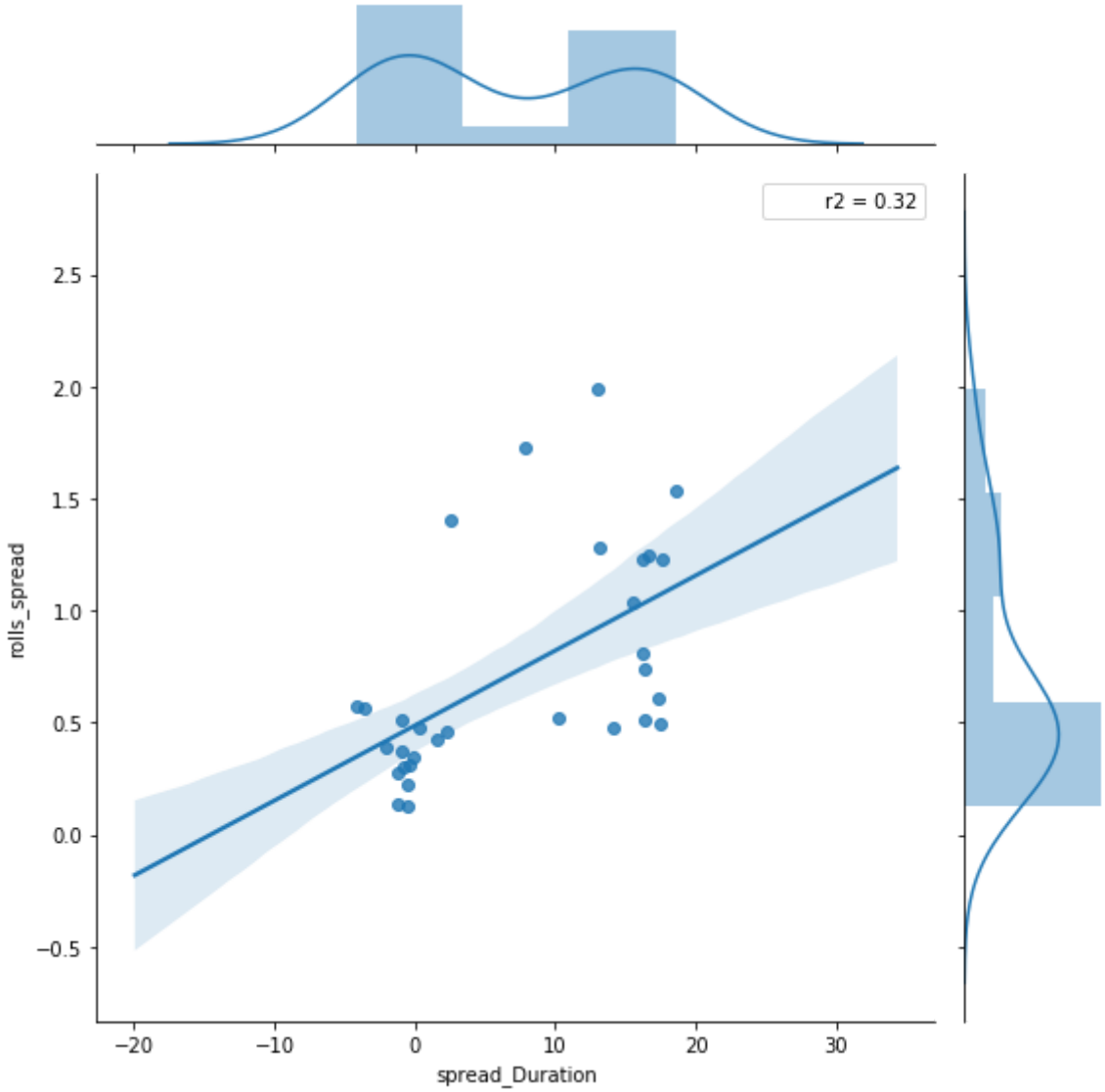
Out[216]: <matplotlib.axes._subplots.AxesSubplot at 0x177786f60>




## Joinplots with R square

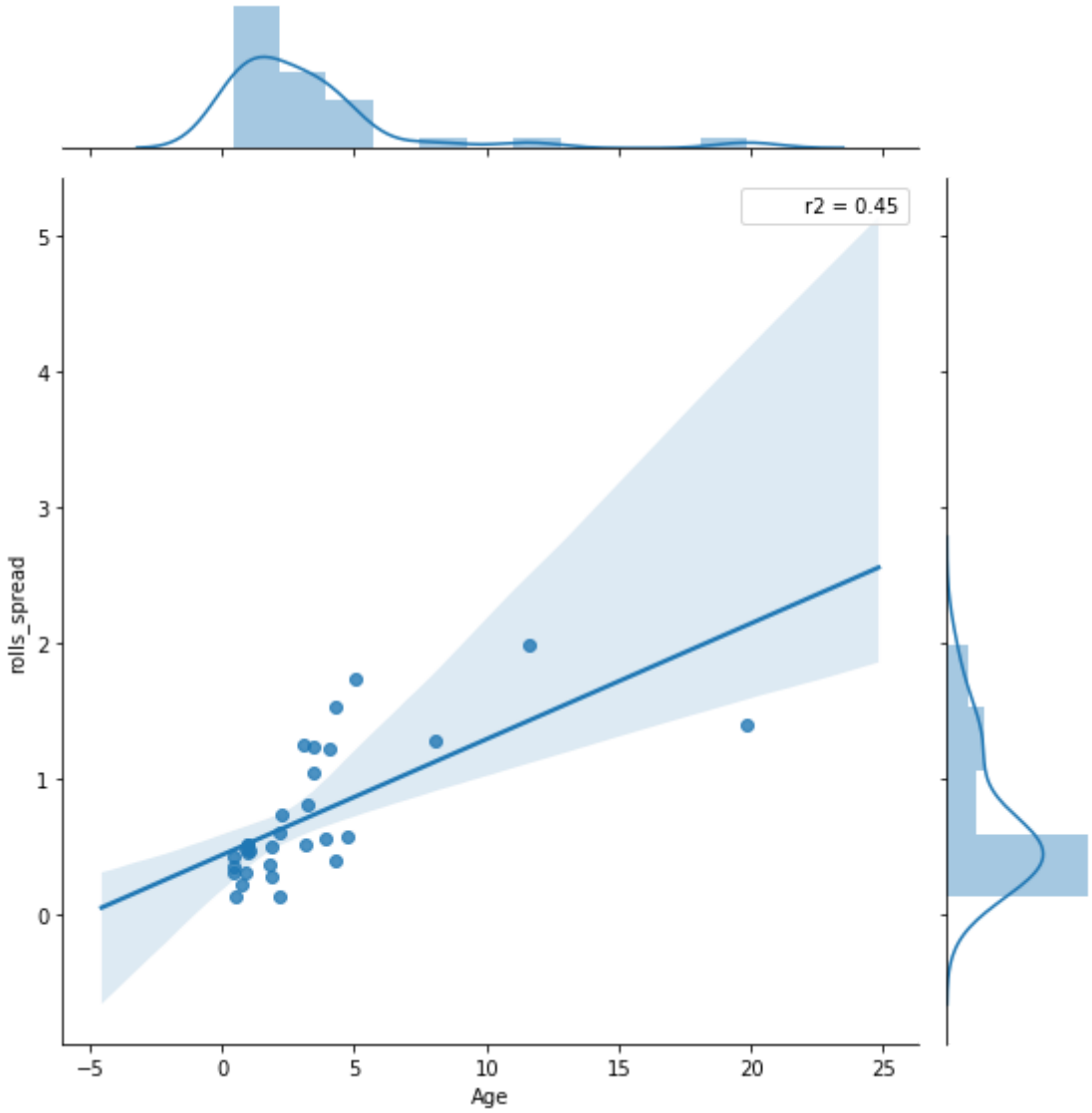
In [330]:
```
from scipy import stats
def r2(x, y):
    return stats.pearsonr(x, y)[0] ** 2
sns.jointplot(x=merged_df_spreadD["spread_Duration"], y=merged_df_spread
D["rolls_spread"], height = 8,kind="reg", stat_func=r2)
```

Out[330]: <seaborn.axisgrid.JointGrid at 0x1633a7c18>



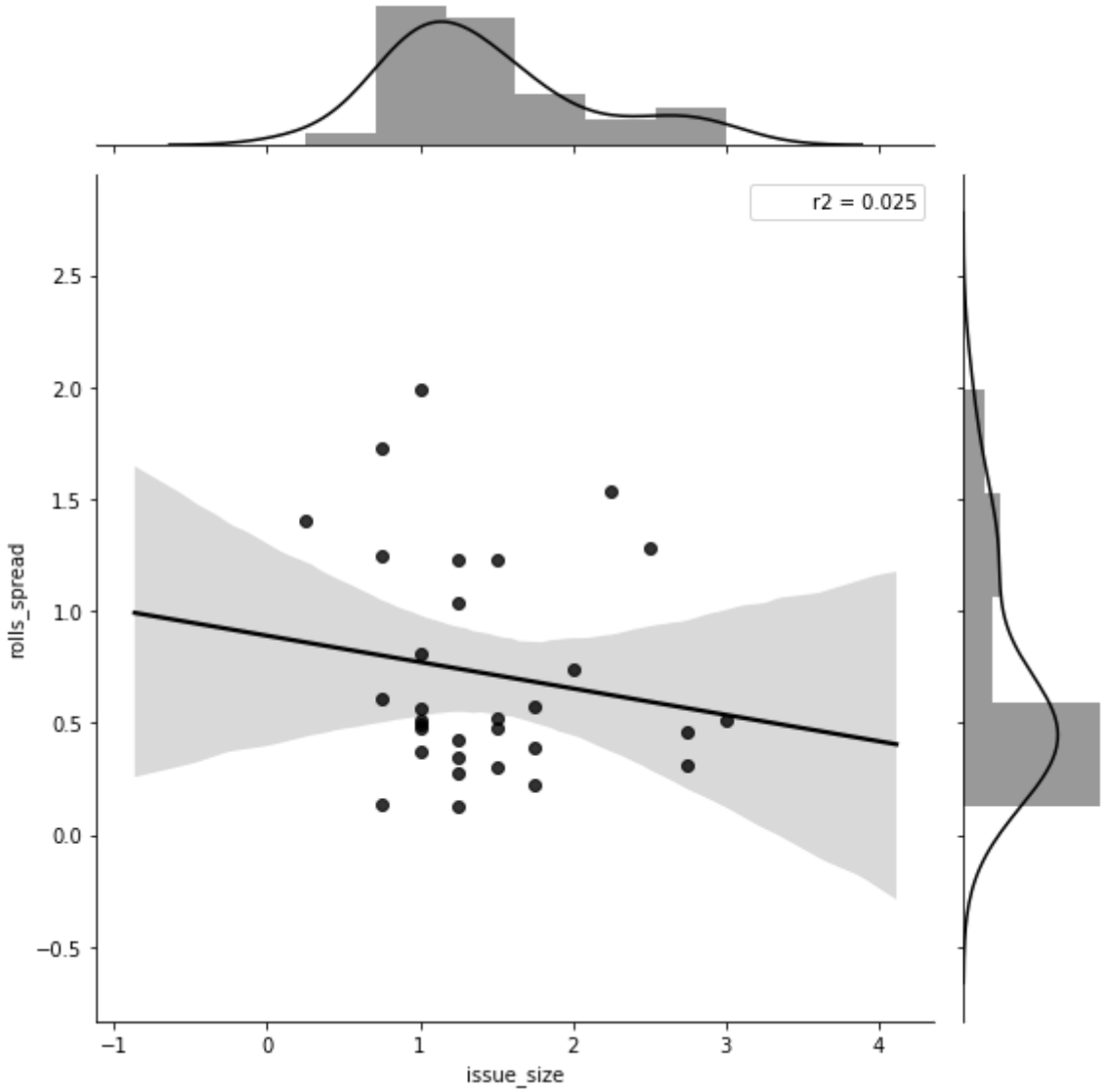
In [329]:
```
sns.jointplot(x=merged_df_age["Age"], y=merged_df_age["rolls_spread"],he
ight = 8, kind="reg", stat_func=r2)
```

Out[329]: <seaborn.axisgrid.JointGrid at 0x16e44f320>



In [333]:
```
sns.jointplot(x=merged_df_issue_size["issue_size"], height = 8, y=merged
_df_issue_size["rolls_spread"],color= "black", kind="reg", stat_func=r2)
```

Out[333]: <seaborn.axisgrid.JointGrid at 0x16fa74898>




## Regression Summary for Issue size, Age and Spread Duration

In [222]:
```
import statsmodels.formula.api as smf
import statsmodels.api as sm

results = smf.ols('rolls_spread ~ issue_size', data=merged_df_issue_size
).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:           rolls_spread   R-squared:
0.025
Model:                            OLS   Adj. R-squared:
-0.009
Method:                 Least Squares   F-statistic:
0.7413
Date:                Thu, 29 Oct 2020   Prob (F-statistic):
0.396
Time:                        18:01:34   Log-Likelihood:
-21.240
No. Observations:                  31   AIC:
46.48
Df Residuals:                      29   BIC:
49.35
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept      0.8898      0.215      4.143      0.000       0.451
1.329
issue_size    -0.1179      0.137     -0.861      0.396      -0.398
0.162
==============================================================================
=======
Omnibus:                        4.909   Durbin-Watson:
0.354
Prob(Omnibus):                  0.086   Jarque-Bera (JB):
4.304
Skew:                           0.909   Prob(JB):
0.116
Kurtosis:                       2.834   Cond. No.
5.12
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [224]:
```
results = smf.ols('rolls_spread ~ Age', data= merged_df_age).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:           rolls_spread   R-squared:
0.452
Model:                            OLS   Adj. R-squared:
0.433
Method:                 Least Squares   F-statistic:
23.89
Date:                Thu, 29 Oct 2020   Prob (F-statistic):
3.47e-05
Time:                        18:03:15   Log-Likelihood:
-12.316
No. Observations:                  31   AIC:
28.63
Df Residuals:                      29   BIC:
31.50
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=======
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
Intercept      0.4395      0.088      4.976      0.000       0.259
0.620
Age            0.0852      0.017      4.888      0.000       0.050
0.121
==============================================================================
=======
Omnibus:                        2.511   Durbin-Watson:
1.113
Prob(Omnibus):                  0.285   Jarque-Bera (JB):
1.859
Skew:                           0.599   Prob(JB):
0.395
Kurtosis:                       2.955   Cond. No.
6.81
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [225]:
```
results = smf.ols('rolls_spread ~ spread_Duration', data=merged_df_spread_D).fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           rolls_spread   R-squared:                       0.319
Model:                            OLS   Adj. R-squared:                  0.296
Method:                 Least Squares   F-statistic:                     13.61
Date:                Thu, 29 Oct 2020   Prob (F-statistic):           0.000924
Time:                        18:04:10   Log-Likelihood:                -15.666
No. Observations:                  31   AIC:                             35.33
Df Residuals:                      29   BIC:                             38.20
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      0.4873      0.098      4.979      0.000       0.287       0.687
spread_Duration     0.0335      0.009      3.689      0.001       0.015       0.052
==============================================================================
Omnibus:                        8.540   Durbin-Watson:                   0.739
Prob(Omnibus):                  0.014   Jarque-Bera (JB):                7.015
Skew:                           1.086   Prob(JB):                       0.0300
Kurtosis:                       3.845   Cond. No.                         14.2
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

# Trailing One month Volume Calculation

In [134]:
```
uniques_cusips = merged_df_.COMPLETE_CUSIP.unique()
merged_df_TV = pd.DataFrame(columns = ['entrd_vol_qt','rolls_spread','CO
MPLETE_CUSIP','_year_month', 'date_monthly','trailing_vol'])
for k in uniques_cusips:
    x = merged_df_[merged_df_.COMPLETE_CUSIP == k]
    x['year'] = pd.DatetimeIndex(x['trd_exctn_dt']).year
    x['month'] = pd.DatetimeIndex(x['trd_exctn_dt']).month
    x["year_month"] =x['year'].astype(str) + x['month'].astype(str)
    x = x.groupby(["year_month"])
    x_s = x.aggregate({"entrd_vol_qt":np.sum,"rolls_spread": np.mean})
    x_s['COMPLETE_CUSIP'] = k
    x_s['_year_month'] = x_s.index
    x_s['date_monthly']  = pd.to_datetime(x_s['_year_month'], format='%
Y%m')
    x_s = x_s.sort_values(by="date_monthly")
    x_s['trailing_vol'] = x_s['entrd_vol_qt'].shift(1)
    merged_df_TV = merged_df_TV.append(x_s)
```

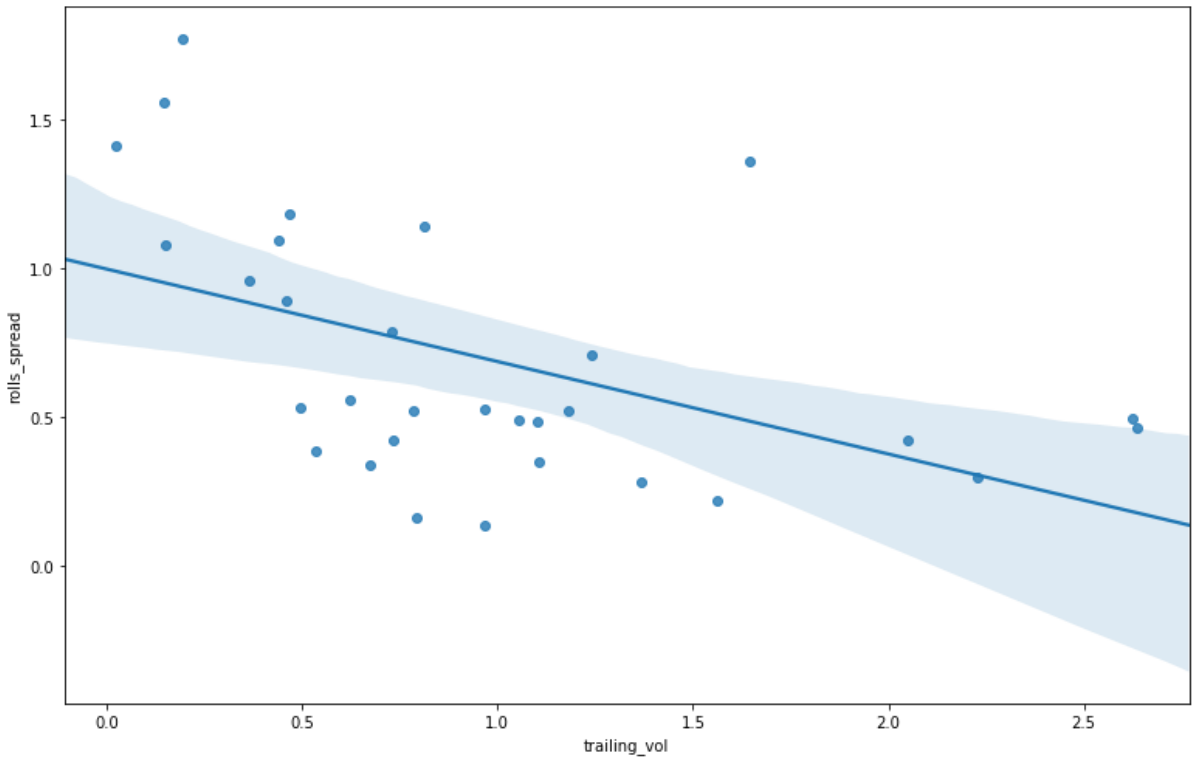
```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:7: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [138]:
```
merged_df_TV_main = merged_df_TV.groupby(["COMPLETE_CUSIP"])['rolls_spre
ad', "trailing_vol"].mean()
merged_df_TV_main["trailing_vol"] = merged_df_TV_main["trailing_vol"]/10
000000
```

# Regplot of Trailing One Month Volume

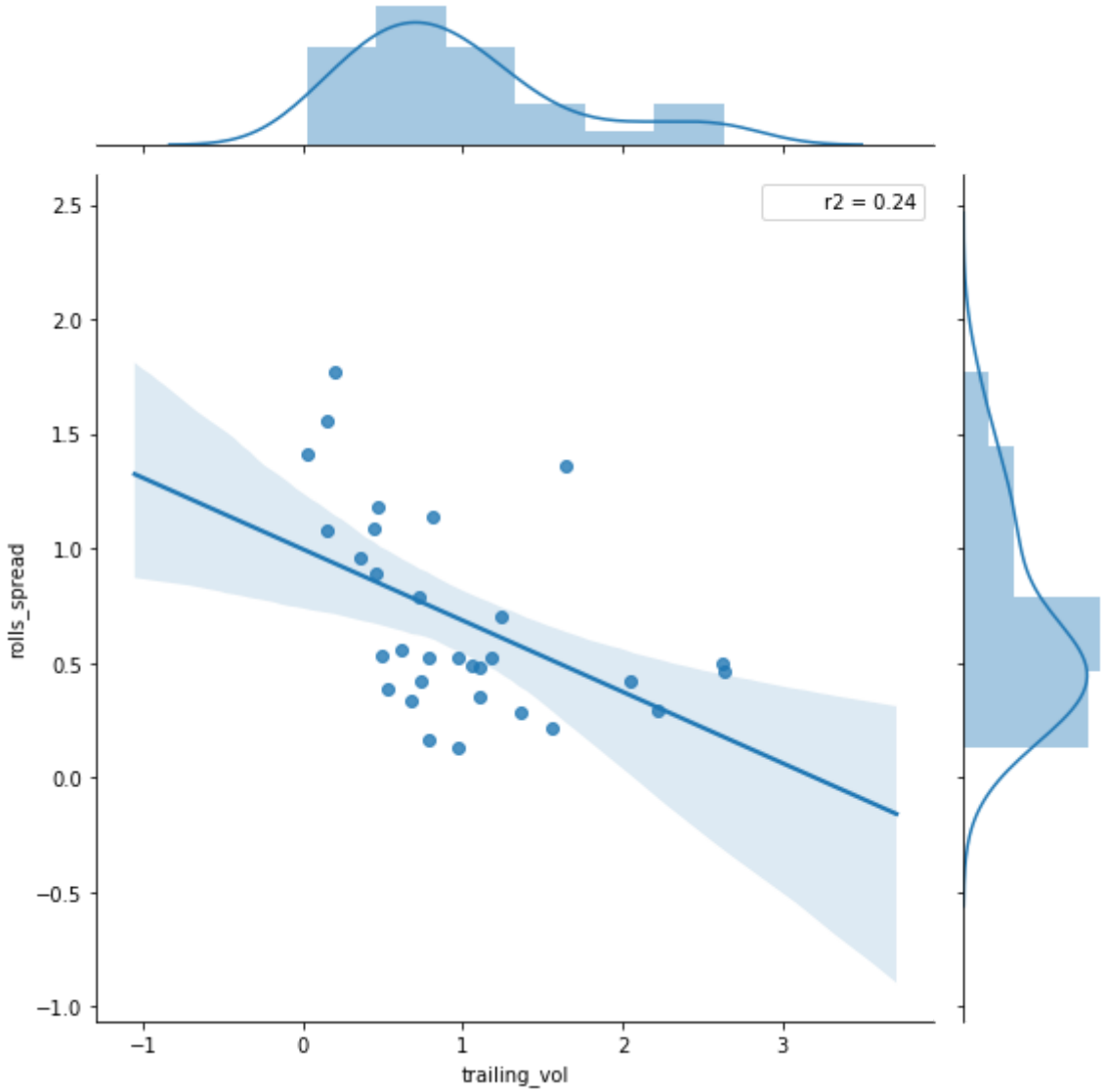
In [283]:
```
a4_dims = (13, 8.27)
fig, ax = pyplot.subplots(figsize=a4_dims)
sns.regplot(x=merged_df_TV_main["trailing_vol"], y=merged_df_TV_main["ro
lls_spread"])
```

Out[283]: <matplotlib.axes._subplots.AxesSubplot at 0x171168160>



In [327]:
```
sns.jointplot(x=merged_df_TV_main["trailing_vol"], y = merged_df_TV_main
["rolls_spread"], height = 8 ,kind="reg", stat_func=r2)
```

Out[327]: <seaborn.axisgrid.JointGrid at 0x170177b38>




## regression Summary for One month trailing Volume

In [287]:
```
results = smf.ols('rolls_spread ~ trailing_vol', data=merged_df_TV_main)
.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:           rolls_spread   R-squared:
0.242
Model:                            OLS   Adj. R-squared:
0.216
Method:                 Least Squares   F-statistic:
9.275
Date:                Thu, 29 Oct 2020   Prob (F-statistic):
0.00491
Time:                        20:22:54   Log-Likelihood:
-13.344
No. Observations:                  31   AIC:
30.69
Df Residuals:                      29   BIC:
33.56
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
Intercept      0.9965      0.121      8.239      0.000       0.749
1.244
trailing_vol  -0.3107      0.102     -3.045      0.005      -0.519
-0.102
==============================================================================
Omnibus:                        2.332   Durbin-Watson:
1.007
Prob(Omnibus):                  0.312   Jarque-Bera (JB):
1.928
Skew:                           0.599   Prob(JB):
0.381
Kurtosis:                       2.760   Cond. No.
3.24
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [141]:
```
merged_df_spreadD = merged_df_tr30.groupby(["COMPLETE_CUSIP"])['rolls_spread', "spread_Duration"].mean()
merged_df_issue_size = merged_df_tr30.groupby(["COMPLETE_CUSIP"])['rolls_spread', "issue_size"].mean()
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:1: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.

/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

In [148]:
```
merged_1 = pd.merge(merged_df_issue_size,merged_df_spreadD,how ="inner", on = "COMPLETE_CUSIP")
merged_2 = pd.merge(merged_1,merged_df_TV_main,how ="inner", on = "COMPLETE_CUSIP")
merged_3 = pd.merge(merged_2,merged_df_age,how ="inner", on = "COMPLETE_CUSIP")
```

In [151]:
```
merged_3 = merged_3[['rolls_spread_x','issue_size','spread_Duration','trailing_vol','Age']]
```

# Generic Model

In [169]:
```
merged_3 = merged_3.loc[:,~merged_3.columns.duplicated()]
merged_3['log_issue_size'] = merged_3['issue_size'].apply(lambda x: np.log(x))
results_m = smf.ols('rolls_spread_x ~ spread_Duration + trailing_vol + Age ', data=merged_3).fit()
print(results_m.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:         rolls_spread_x   R-squared:
0.714
Model:                            OLS   Adj. R-squared:
0.683
Method:                 Least Squares   F-statistic:
22.51
Date:                Tue, 10 Nov 2020   Prob (F-statistic):
1.64e-07
Time:                        15:55:07   Log-Likelihood:
-2.2091
No. Observations:                  31   AIC:
12.42
Df Residuals:                      27   BIC:
18.15
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
============
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------------
Intercept           0.3539      0.138      2.572      0.016       0.072
0.636
spread_Duration     0.0296      0.006      4.821      0.000       0.017
0.042
trailing_vol       -0.0079      0.009     -0.911      0.370      -0.026
0.010
Age                 0.0718      0.015      4.678      0.000       0.040
0.103
==============================================================================
=======
Omnibus:                        6.152   Durbin-Watson:
1.421
Prob(Omnibus):                  0.046   Jarque-Bera (JB):
4.517
Skew:                           0.721   Prob(JB):
0.105
Kurtosis:                       4.191   Cond. No.
38.9
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

In [174]:
```
merged_3.to_csv("merged_3.csv")
```

In [176]:
```
merged_df_tr30.to_csv("wmt_data4Regression.csv")
```

## NEW CUSIP( Investment grade Bonds) Liquidity ANALYSIS

In [8]:
```
sql_query = """
SELECT *
FROM trace.trace_enhanced
WHERE CUSIP_ID in ('92343VFT6','92343VBR4','110122DQ8','110122DR6','4664
7PBB1','46625HHA1','6174468U6','6174468T9','J57160DZ3','404280CL1','2003
0NBX8','035240AN0','097023CZ6','63254AAE8','92976GAG6','012725AD9','1127
1LAB8','097023BX2','26969PAA6','343498AA9','88732JBB3','37045VAT7','7847
10AB1','418056AS6','059438AG6','251526BR9','04621XAF5','38376AAB9','0970
23CM5','25470DBF5','25272KAK9','071813BW8','05526DBP9', '281020AM9','292
480AM2','674599CW3','614810AB5')
AND TRD_EXCTN_DT BETWEEN '2007-01-01' AND '2019-12-31';
"""
new_data_query1 = db.raw_sql(sql_query)
```

In [9]:
```
new_data_query2 = pd.read_csv("extra_wmt.csv")
new_data_query2.dtypes
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/IPython/core/inter
activeshell.py:3057: DtypeWarning: Columns (8,13,16,23) have mixed type
s.Specify dtype option on import or set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[9]:
```
cusip_id                 object
bond_sym_id              object
company_symbol           object
trd_exctn_dt              int64
trd_exctn_tm             object
trd_rpt_dt                int64
trd_rpt_tm               object
trc_st                   object
scrty_type_cd            object
wis_fl                   object
cmsn_trd                 object
entrd_vol_qt            float64
rptd_pr                 float64
yld_sign_cd              object
yld_pt                  float64
asof_cd                  object
sale_cndtn_cd            object
rpt_side_cd              object
buy_cmsn_rt             float64
buy_cpcty_cd             object
sell_cmsn_rt            float64
sell_cpcty_cd            object
cntra_mp_id              object
agu_qsr_id               object
trdg_mkt_cd              object
stlmnt_dt               float64
trd_mod_3                object
rptg_party_type          object
pr_trd_dt               float64
first_trade_ctrl_date   float64
dtype: object
```

In [11]:
```
new_data_query2["trd_exctn_dt"] = pd.to_datetime(new_data_query2['trd_ex
ctn_dt'], format='%Y%m%d').dt.date
new_data_query2["trd_rpt_dt"] = pd.to_datetime(new_data_query2['trd_rpt_
dt'], format='%Y%m%d').dt.date
```

In [12]:
```
new_data_query2["trd_rpt_tm"] = pd.to_datetime(new_data_query2["trd_rpt_
tm"], format='%H:%M:%S').dt.time
```

In [13]:
```
new_data_query1['trd_exctn_tm'] = new_data_query1['trd_exctn_tm'].apply(
lambda x: datetime.timedelta(seconds=x))
new_data_query1['trd_rpt_tm'] = new_data_query1['trd_rpt_tm'].apply(lamb
da x: datetime.timedelta(seconds=x))
```

In [14]:
```
frames = [new_data_query1, new_data_query2]
new_data_query = pd.concat(frames)
```

In [15]:
```
### Dick-Nielsen Filtering
# Cancellations, reversals and corrections
new_data_query = new_data_query[~new_data_query['trc_st'].isin(['C','Y',
'X'])]
#After Market Hours
new_data_query = new_data_query[new_data_query.sale_cndtn2_cd != 'A']
# Filter below 100 percentage pts
new_data_query = new_data_query[new_data_query['rptd_pr'] >= 100]
# Remove agency transactions without commissions
sell_index = new_data_query[(new_data_query['rpt_side_cd'] == 'S') & (ne
w_data_query['sell_cpcty_cd'] == 'A') & (new_data_query['cntra_mp_id'] =
= 'C') & (new_data_query['cmsn_trd'] == 'N')].index
new_data_query.drop(sell_index, inplace=True)
buy_index = new_data_query[(new_data_query['rpt_side_cd'] == 'B') & (new
_data_query['buy_cpcty_cd'] == 'A') & (new_data_query['cntra_mp_id'] ==
'C') & (new_data_query['cmsn_trd'] == 'N')].index
new_data_query.drop(buy_index, inplace=True)
# Remove interdealer transactions
interdealer_txn = new_data_query[(new_data_query['cntra_mp_id'] == 'D')
& (new_data_query['rpt_side_cd'] == 'B')].index
new_data_query.drop(interdealer_txn, inplace=True)
```

In [18]:
```
new_data_query = new_data_query.rename(columns={"cusip_id": "COMPLETE_CU
SIP"})
new_data_query_fisd = pd.merge(new_data_query,fisd_data, how = "inner",
on = "COMPLETE_CUSIP")
new_data_query_fisd["trd_exctn_dt"] = pd.to_datetime(new_data_query_fisd
["trd_exctn_dt"], format="%Y-%m-%d")
```

In [20]:
```
new_data_query_fisd['MATURITY'] = new_data_query_fisd['MATURITY'].apply(
pd.to_datetime)
new_data_query_fisd['Days_to_Maturity'] = (new_data_query_fisd['MATURIT
Y']- new_data_query_fisd['trd_exctn_dt'])
new_data_query_fisd['N_MATURITY'] = ((new_data_query_fisd['Days_to_Matur
ity'] / 365) * 2)/np.timedelta64(1, 'D')
new_data_query_fisd = new_data_query_fisd[new_data_query_fisd['Days_to_M
aturity'] > pd.Timedelta(0,'D')]


yld = {}
for i in range(0, len(new_data_query_fisd)):
    try:
        yld.update({i:YieldCalc(new_data_query_fisd['COUPON'].iloc[i]/(n
ew_data_query_fisd['PRINCIPAL_AMT'].iloc[i]/10),new_data_query_fisd['rpt
d_pr'].iloc[i],
                                new_data_query_fisd['PRINCIPAL_AMT'].ilo
c[i]/10, 2.0,
                                new_data_query_fisd['N_MATURITY'].iloc[i
], new_data_query_fisd['COUPON'].iloc[i])})
    except(RuntimeError):
        pass
    else:
        pass

l = []
for i in yld:
    l.append(yld[i])

new_data_query_fisd['YTM'] = pd.Series(l, index=new_data_query_fisd.inde
x)
```

In [23]:
```
new_data_query_fisd = calculate_duration(new_data_query_fisd)
```

In [27]:
```
uniques_cusips = new_data_query.COMPLETE_CUSIP.unique()
Ncs_rs_all = pd.DataFrame(columns = ['COMPLETE_CUSIP', 'date', 'auto_co
v','rolls_spread'])
for k in uniques_cusips:
    x = new_data_query[new_data_query.COMPLETE_CUSIP == k]
    auto_cov = []
    rolls_spread = []
    uniques_dates = x.trd_exctn_dt.unique()
    for i in range(0,len(uniques_dates)):
        p = x[x.trd_exctn_dt == uniques_dates[i]]['rptd_pr']
        d = np.diff(p)
        if len(d) == 0 or len(d) == 1 :
            a = np.nan
            r = np.nan
            auto_cov.append(np.nan)
            rolls_spread.append(np.nan)
        else:
            a = lagged_auto_cov(d,1)
            r = 2*np.sqrt(-a)
            auto_cov.append(a)
            rolls_spread.append(r)
        data = pd.DataFrame({'COMPLETE_CUSIP':k ,'date':uniques_dates[i
],'auto_cov': a,'rolls_spread' : r },index=[0])
        Ncs_rs_all = Ncs_rs_all.append(data)
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:18: RuntimeWarning: invalid value encountered in sqrt
```

In [28]:
```
len(uniques_cusips)
```

Out[28]: 35

In [29]:
```
Ncs_rs_all
```

Out[29]:

| | COMPLETE_CUSIP | date | auto_cov | rolls_spread |
|---|---|---|---|---|
| **0** | 012725AD9 | 2014-11-17 | -0.002260 | 0.095077 |
| **0** | 012725AD9 | 2014-11-19 | NaN | NaN |
| **0** | 012725AD9 | 2014-11-20 | -0.326059 | 1.142031 |
| **0** | 012725AD9 | 2014-11-21 | NaN | NaN |
| **0** | 012725AD9 | 2014-11-24 | -0.311787 | 1.116757 |
| **...** | ... | ... | ... | ... |
| **0** | 00206RAS1 | 2020-03-24 | -1.810418 | 2.691036 |
| **0** | 00206RAS1 | 2020-03-25 | 0.444078 | NaN |
| **0** | 00206RAS1 | 2020-03-26 | 0.004771 | NaN |
| **0** | 00206RAS1 | 2020-03-30 | NaN | NaN |
| **0** | 00206RAS1 | 2020-03-31 | NaN | NaN |

19222 rows × 4 columns

In [30]:
```
Ncs_rs_all['year'] = pd.DatetimeIndex(Ncs_rs_all['date']).year
Ncs_rs_all['month'] = pd.DatetimeIndex(Ncs_rs_all['date']).month
Ncs_rs_all["year_month"] = Ncs_rs_all['year'].astype(str) + Ncs_rs_all[
'month'].astype(str)
```

In [31]:
```
Ncs_rs_all_ma = Ncs_rs_all.groupby(["year_month"])
Ncs_rs_all_ma = Ncs_rs_all_ma.aggregate({"rolls_spread":np.mean}).dropna
()
```

In [34]:
```
Ncs_rs_all_ma
```

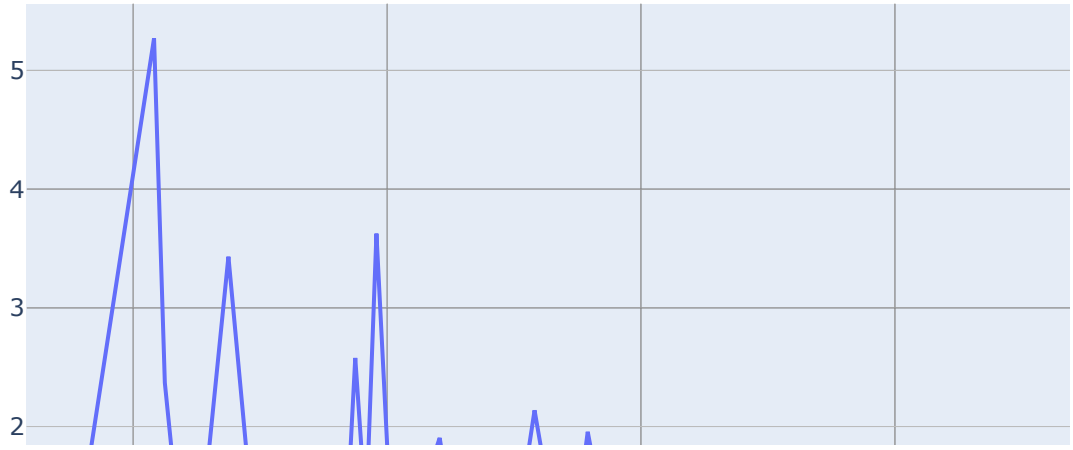
Out[34]:

| year_month | rolls_spread | _year_month | Date |
|---|---|---|---|
| **20073** | 0.245000 | 20073 | 2007-03-01 |
| **20074** | 0.798000 | 20074 | 2007-04-01 |
| **20076** | 0.055540 | 20076 | 2007-06-01 |
| **20083** | 5.271000 | 20083 | 2008-03-01 |
| **20084** | 2.368246 | 20084 | 2008-04-01 |
| **...** | ... | ... | ... |
| **201911** | 0.540249 | 201911 | 2019-11-01 |
| **201912** | 0.514775 | 201912 | 2019-12-01 |
| **20201** | 0.626234 | 20201 | 2020-01-01 |
| **20202** | 0.567962 | 20202 | 2020-02-01 |
| **20203** | 1.280718 | 20203 | 2020-03-01 |

139 rows × 3 columns

In [33]:
```
Ncs_rs_all_ma['_year_month'] = Ncs_rs_all_ma.index
Ncs_rs_all_ma['Date']  = pd.to_datetime(Ncs_rs_all_ma['_year_month'], fo
rmat='%Y%m')
Ncs_rs_all_ma = Ncs_rs_all_ma.sort_values(by="Date")
```

In [35]:
```
import plotly.express as px
import plotly.graph_objects as go
fig = go.Figure(data=go.Scatter(x=Ncs_rs_all_ma['Date'], y=Ncs_rs_all_ma
['rolls_spread'],name = 'rolls_spread'))
fig.show("notebook")
```




In [36]:
```
Ncs_rs_all = Ncs_rs_all.rename(columns={"date":"trd_exctn_dt"})
Ncs_rs_all['trd_exctn_dt']  = pd.to_datetime(Ncs_rs_all['trd_exctn_dt'],
format='%Y-%m-%d')
merged_df_cusips_all = new_data_query_fisd.merge(Ncs_rs_all, how='inner'
, on=["COMPLETE_CUSIP", "trd_exctn_dt"])
```

In [333]:
```
merged_df_cusips_all.to_csv("merged_df_cusips_all.csv")
```

# Bond Characteristic Analysis for Investment Grade Bonds

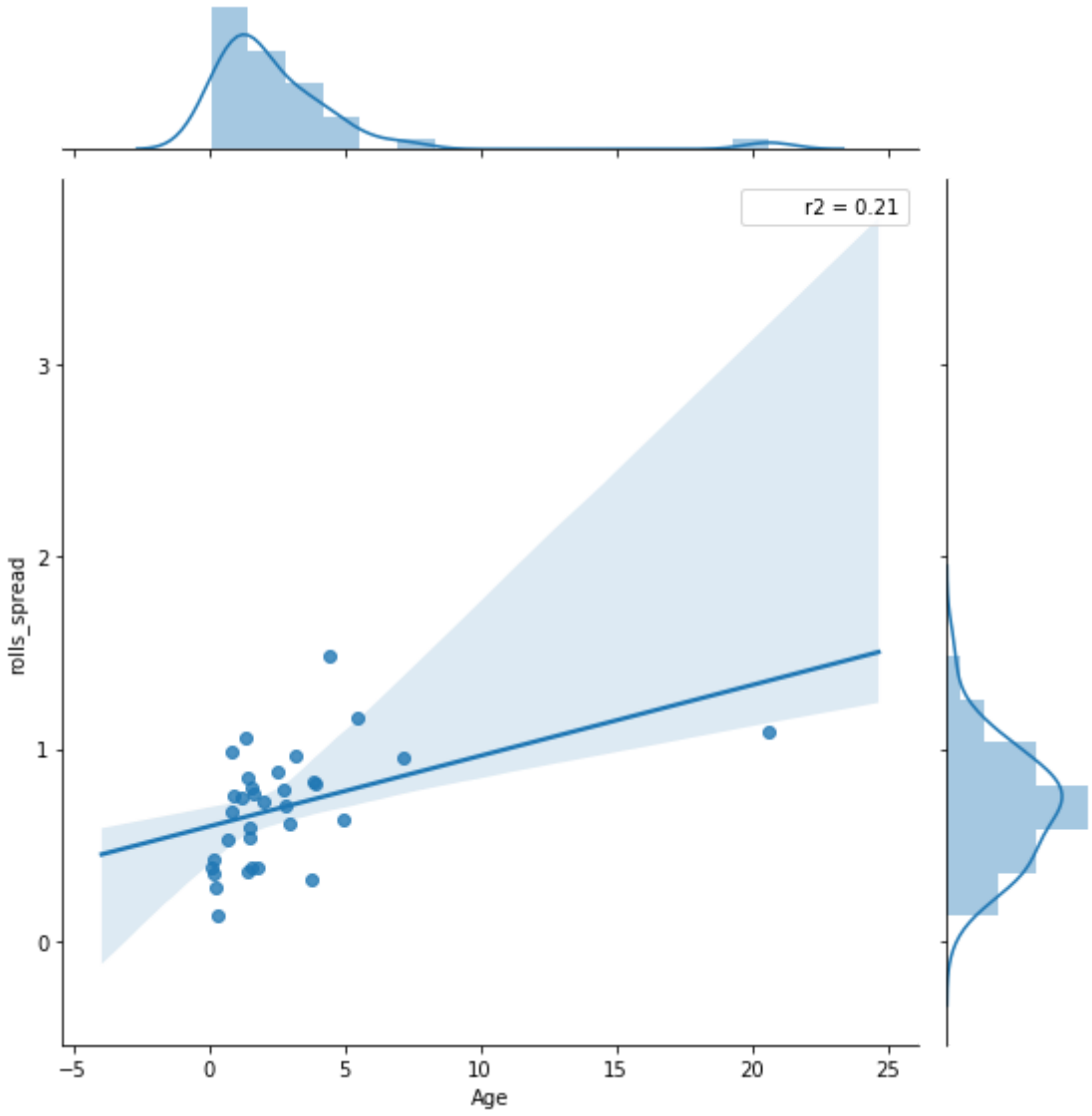
In [38]:
```
merged_df_cusips_all['OFFERING_DATE'] = pd.to_datetime(merged_df_cusips_
all['OFFERING_DATE'])
merged_df_cusips_all['Age'] = (merged_df_cusips_all['trd_exctn_dt'] - me
rged_df_cusips_all['OFFERING_DATE']).apply(lambda x: (x.days)/365)
merged_df_cusips_all_= merged_df_cusips_all.dropna(subset=['rolls_sprea
d'])
merged_df_cusips_all_age = merged_df_cusips_all_.groupby(["COMPLETE_CUSI
P"])['rolls_spread', "Age"].mean()
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:4: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) w
ill be deprecated, use a list instead.
```

In [40]:
```
sns.jointplot(x=merged_df_cusips_all_age['Age'], y=merged_df_cusips_all_
age["rolls_spread"], height = 8,kind="reg", stat_func=r2)
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.
py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated;
use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be
interpreted as an array index, `arr[np.array(seq)]`, which will result
either in an error or a different result.

/Users/deepsha/anaconda3/lib/python3.7/site-packages/seaborn/axisgrid.p
y:1847: UserWarning:

JointGrid annotation is deprecated and will be removed in a future rele
ase.
```

Out[40]: <seaborn.axisgrid.JointGrid at 0x139d18860>

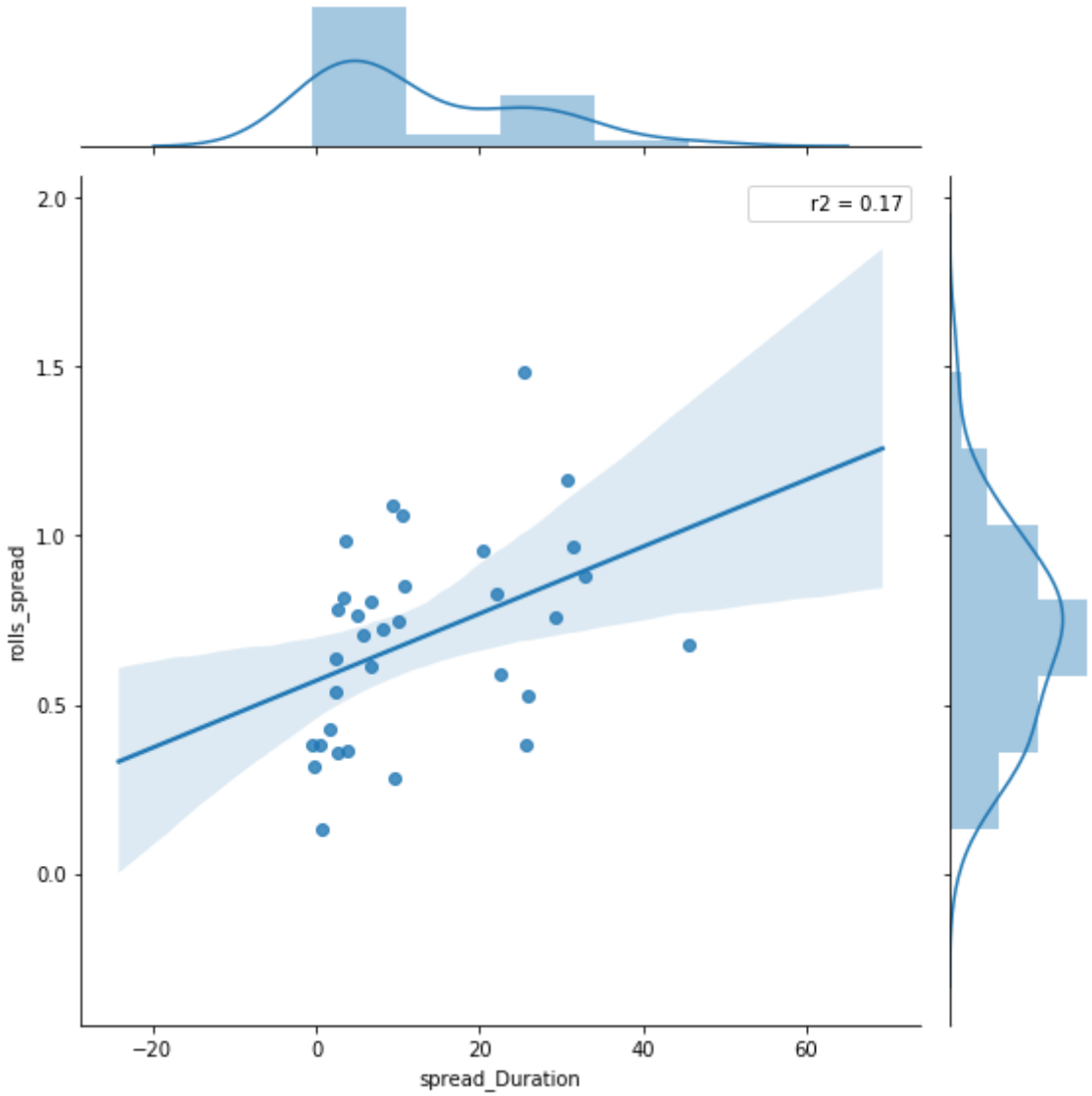


In [42]:
```
merged_df_cusips_all_tr30= pd.merge(merged_df_cusips_all_,tr30, how ="inner", on = "trd_exctn_dt")
merged_df_cusips_all_tr30["spread"] =  merged_df_cusips_all_tr30["YTM"] - (merged_df_cusips_all_tr30["DGS30"].astype(float)/100)
merged_df_cusips_all_tr30["spread_Duration"] = merged_df_cusips_all_tr30["spread"]*merged_df_cusips_all_tr30["Duration_MAC"]*100
merged_df_cusips_all_tr30["issue_size"] = merged_df_cusips_all_tr30["OFFERING_AMT"]*merged_df_cusips_all_tr30["PRINCIPAL_AMT"]
merged_df_cusips_all_spreadD = merged_df_cusips_all_tr30.groupby(["COMPLETE_CUSIP"])['rolls_spread', "spread_Duration"].mean()
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launcher.py:5: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) will be deprecated, use a list instead.
```

In [43]:
```
sns.jointplot(x=merged_df_cusips_all_spreadD['spread_Duration'], y=merged_df_cusips_all_spreadD["rolls_spread"], height = 8,kind="reg", stat_func=r2)
```

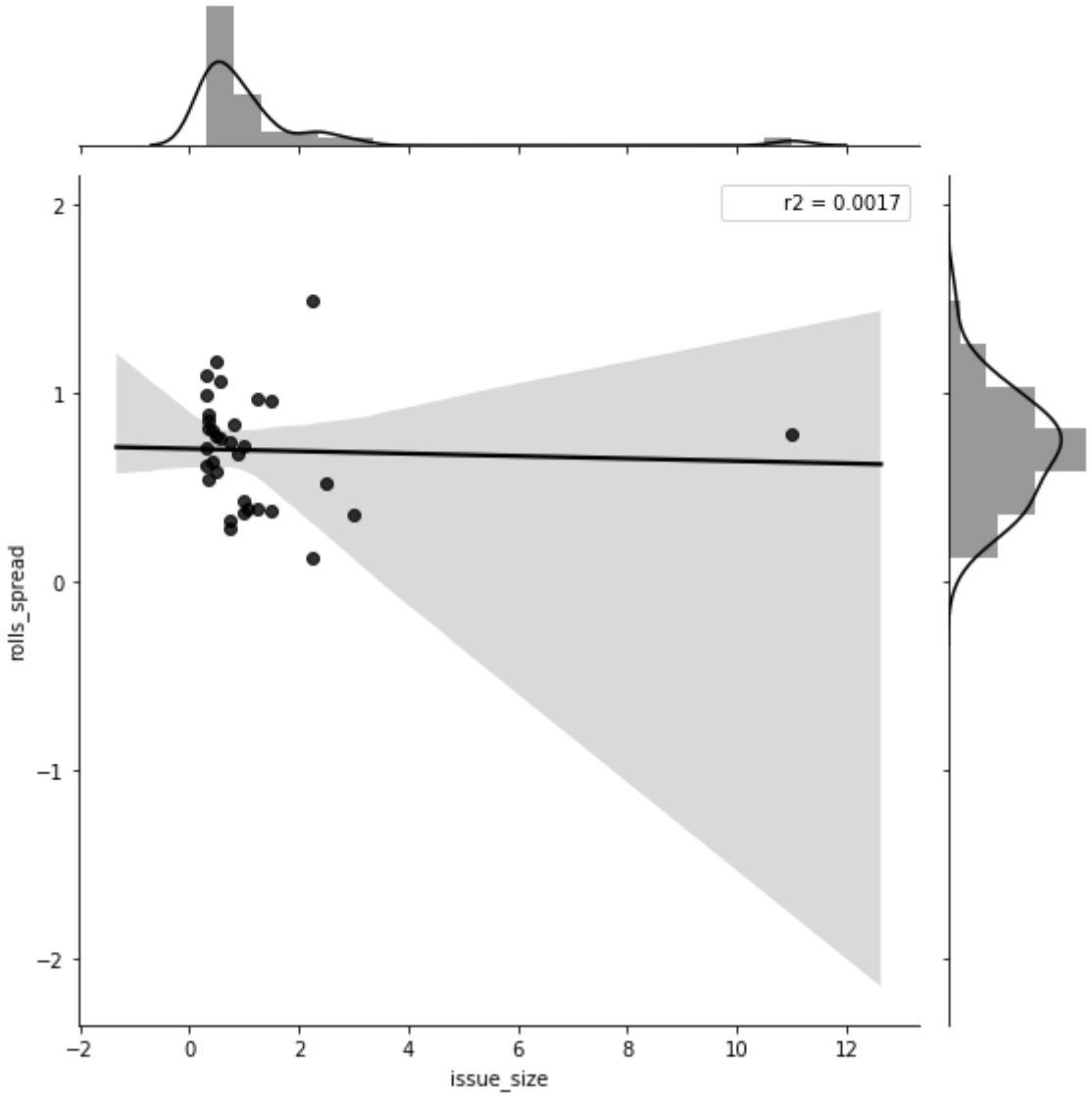
Out[43]: <seaborn.axisgrid.JointGrid at 0x1278387b8>



In [44]:
```
merged_df_cusips_all_issue_size = merged_df_cusips_all_tr30.groupby(["CO
MPLETE_CUSIP"])['rolls_spread', "issue_size"].mean()
merged_df_cusips_all_issue_size["issue_size"] = merged_df_cusips_all_iss
ue_size["issue_size"]/1000000000
sns.jointplot(x=merged_df_cusips_all_issue_size["issue_size"], height =
8, y=merged_df_cusips_all_issue_size["rolls_spread"],color= "black", kin
d="reg", stat_func=r2)
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:1: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) w
ill be deprecated, use a list instead.
```

Out[44]: <seaborn.axisgrid.JointGrid at 0x12853c550>



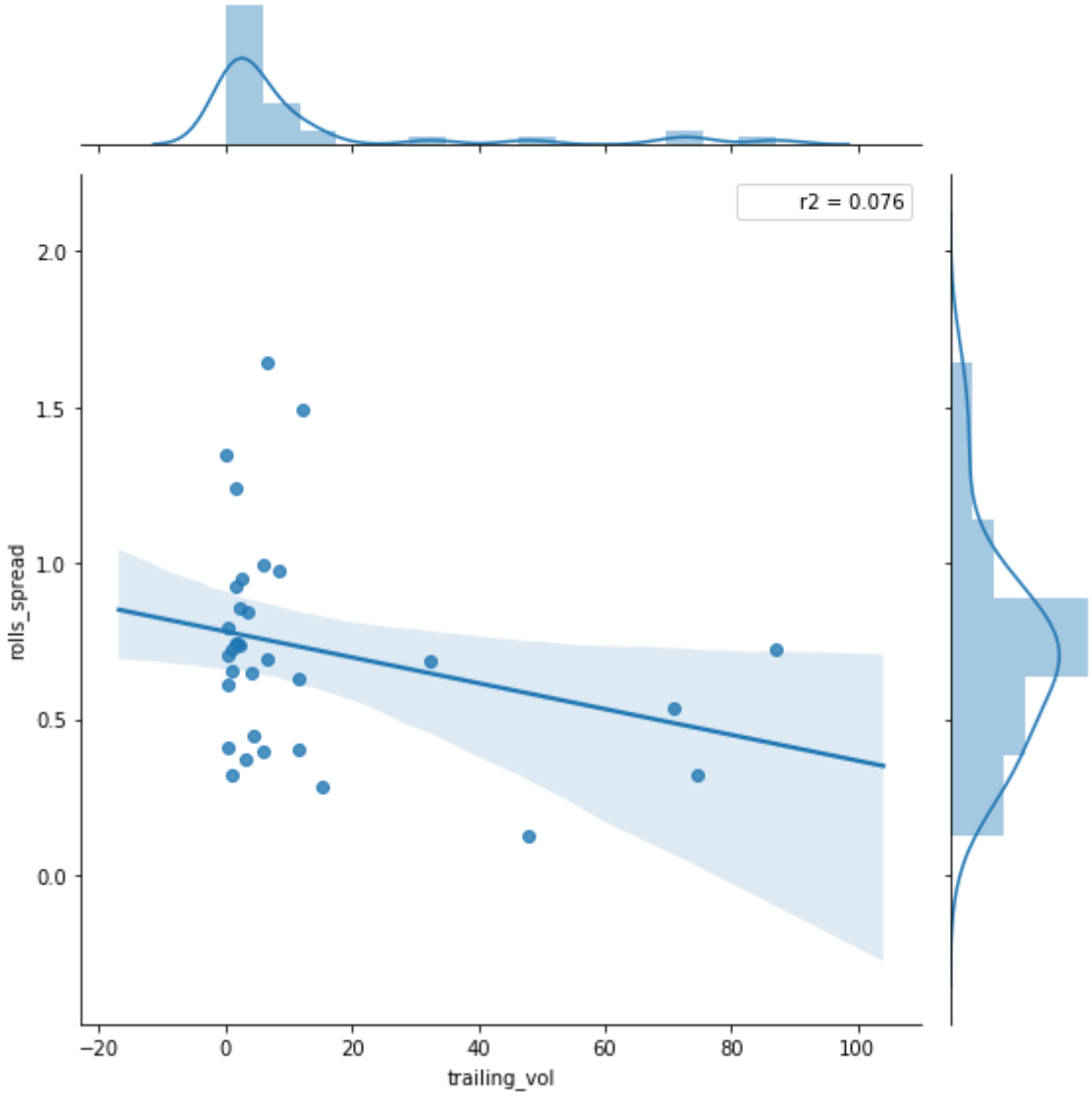
In [45]:
```
uniques_cusips = merged_df_cusips_all_.COMPLETE_CUSIP.unique()
merged_df_cusips_all_TV = pd.DataFrame(columns = ['entrd_vol_qt','rolls_
spread','COMPLETE_CUSIP','_year_month', 'date_monthly','trailing_vol'])
for k in uniques_cusips:
    x = merged_df_cusips_all_[merged_df_cusips_all_.COMPLETE_CUSIP == k]
    x['year'] = pd.DatetimeIndex(x['trd_exctn_dt']).year
    x['month'] = pd.DatetimeIndex(x['trd_exctn_dt']).month
    x["year_month"] =x['year'].astype(str) + x['month'].astype(str)
    x = x.groupby(["year_month"])
    x_s = x.aggregate({"entrd_vol_qt":np.sum,"rolls_spread": np.mean})
    x_s['COMPLETE_CUSIP'] = k
    x_s['_year_month'] = x_s.index
    x_s['date_monthly']  = pd.to_datetime(x_s['_year_month'], format='%
Y%m')
    x_s = x_s.sort_values(by="date_monthly")
    x_s['trailing_vol'] = x_s['entrd_vol_qt'].shift(1)
    merged_df_cusips_all_TV = merged_df_cusips_all_TV.append(x_s)
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:5: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:6: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:7: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [47]:
```
merged_df_cusips_all_TV_main = merged_df_cusips_all_TV.groupby(["COMPLETE_CUSIP"])['rolls_spread', "trailing_vol"].mean()
merged_df_cusips_all_TV_main["trailing_vol"] = merged_df_cusips_all_TV_main["trailing_vol"]/10000000

sns.jointplot(x=merged_df_cusips_all_TV_main["trailing_vol"], y = merged_df_cusips_all_TV_main["rolls_spread"], height = 8 ,kind="reg", stat_func=r2)
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:1: FutureWarning:

Indexing with multiple keys (implicitly converted to a tuple of keys) w
ill be deprecated, use a list instead.

/Users/deepsha/anaconda3/lib/python3.7/site-packages/scipy/stats/stats.
py:1713: FutureWarning:

Using a non-tuple sequence for multidimensional indexing is deprecated;
use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be
interpreted as an array index, `arr[np.array(seq)]`, which will result
either in an error or a different result.

/Users/deepsha/anaconda3/lib/python3.7/site-packages/seaborn/axisgrid.p
y:1847: UserWarning:

JointGrid annotation is deprecated and will be removed in a future rele
ase.
```

Out[47]: <seaborn.axisgrid.JointGrid at 0x12b2f98d0>



In [48]:
```
merged_1 = pd.merge(merged_df_cusips_all_issue_size,merged_df_cusips_all
_spreadD,how ="inner", on = "COMPLETE_CUSIP")
merged_2 = pd.merge(merged_1,merged_df_cusips_all_TV_main,how ="inner",
on = "COMPLETE_CUSIP")
merged_3_all = pd.merge(merged_2,merged_df_cusips_all_age,how ="inner",
on = "COMPLETE_CUSIP")
```

# Final Investment grade Bonds

In [49]:
```
merged_3_all
```

Out[49]:

| COMPLETE_CUSIP | rolls_spread_x | issue_size | rolls_spread_y | spread_Duration | rolls_spread_x | tra |
|---|---|---|---|---|---|---|
| 00206RAS1 | 1.485341 | 2.250000 | 1.485341 | 25.371667 | 1.495307 | 12 |
| 012725AD9 | 0.880805 | 0.350000 | 0.880805 | 32.956884 | 0.742098 | 1 |
| 02406PAU4 | 0.588226 | 0.499760 | 0.588226 | 22.647077 | 0.649296 | 4 |
| 035240AN0 | 0.525740 | 2.500000 | 0.525740 | 25.796990 | 0.532878 | 70 |
| 04621XAF5 | 0.817231 | 0.350000 | 0.817231 | 3.402616 | 0.791784 | 0 |
| 059438AG6 | 1.090580 | 0.300000 | 1.090580 | 9.403359 | 1.348438 | 0 |
| 097023BX2 | 0.538606 | 0.350000 | 0.538606 | 2.450760 | 0.611648 | 0 |
| 097023CM5 | 0.426020 | 1.000000 | 0.426020 | 1.539362 | 0.402830 | 11 |
| 11271LAB8 | 0.760152 | 0.550000 | 0.760152 | 29.159131 | 1.641362 | 6 |
| 126307AY3 | 0.383804 | 1.045882 | 0.383804 | 25.589729 | 0.409313 | 0 |
| 177376AE0 | 0.744194 | 0.750000 | 0.744194 | 10.153748 | 0.695913 | 6 |
| 20030NAY7 | 0.828751 | 0.800000 | 0.828751 | 22.110255 | 0.953500 | 2 |
| 20030NBX8 | 0.381215 | 1.250000 | 0.381215 | -0.453488 | 0.396264 | 6 |
| 251526BR9 | 0.361262 | 1.000000 | 0.361262 | 3.785005 | 0.446092 | 4 |
| 25470DBF5 | 0.283568 | 0.750000 | 0.283568 | 9.622756 | 0.286386 | 15 |
| 269246BQ6 | 0.803162 | 0.400000 | 0.803162 | 6.665016 | 0.845828 | 3 |
| 26969PAA6 | 0.851833 | 0.350000 | 0.851833 | 10.780075 | 0.858604 | 2 |
| 281020AM9 | 1.061425 | 0.550000 | 1.061425 | 10.428552 | 0.992845 | 5 |
| 30212PAP0 | 0.723679 | 1.000000 | 0.723679 | 8.238717 | 0.630828 | 11 |
| 31428XBP0 | 0.766469 | 0.500000 | 0.766469 | 4.973166 | 0.733916 | 2 |
| 343498AA9 | 0.636902 | 0.400000 | 0.636902 | 2.255282 | 0.653576 | 0 |
| 37045VAT7 | 0.677915 | 0.900000 | 0.677915 | 45.606817 | 0.685273 | 32 |
| 38376AAB9 | 0.987394 | 0.300000 | 0.987394 | 3.521339 | 0.744221 | 2 |
| 418056AS6 | 1.162382 | 0.500000 | 1.162382 | 30.806409 | 1.237482 | 1 |
| 46647PBB1 | 0.131441 | 2.250000 | 0.131441 | 0.694481 | 0.129313 | 47 |
| 52107QAH8 | 0.705176 | 0.300000 | 0.705176 | 5.791111 | 0.708517 | 0 |
| 614810AB5 | 0.615543 | 0.300000 | 0.615543 | 6.742356 | 0.722599 | 1 |
| 63254AAE8 | 0.318886 | 0.750000 | 0.318886 | -0.201531 | 0.321282 | 1 |
| 674599CW3 | 0.356788 | 3.000000 | 0.356788 | 2.621921 | 0.321816 | 74 |
| 713448DY1 | 0.379390 | 1.500000 | 0.379390 | 0.359984 | 0.370328 | 3 |
| 88732JBB3 | 0.965914 | 1.250000 | 0.965914 | 31.478948 | 0.978918 | 8 |
| 92343VBR4 | 0.783273 | 11.000000 | 0.783273 | 2.563271 | 0.723661 | 87 |
| 92976GAG6 | 0.953884 | 1.500000 | 0.953884 | 20.411201 | 0.928018 | 1 |

# Generic Model for Investment grade Bonds

In [50]:
```
merged_3_all = merged_3_all.loc[:,~merged_3_all.columns.duplicated()]
merged_3_all['log_issue_size'] = merged_3_all['issue_size'].apply(lambda
x: np.log(x))
results_m = smf.ols('rolls_spread_x ~ spread_Duration + trailing_vol + A
ge ', data = merged_3_all).fit()
print(results_m.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:         rolls_spread_x   R-squared:
0.390
Model:                            OLS   Adj. R-squared:
0.327
Method:                 Least Squares   F-statistic:
6.187
Date:                Sun, 29 Nov 2020   Prob (F-statistic):
0.00221
Time:                        14:39:04   Log-Likelihood:
2.1619
No. Observations:                  33   AIC:
3.676
Df Residuals:                      29   BIC:
9.662
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
============
                     coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------------
Intercept           0.5094      0.076      6.722      0.000       0.354
0.664
spread_Duration     0.0096      0.003      2.775      0.010       0.003
0.017
trailing_vol       -0.0019      0.002     -1.016      0.318      -0.006
0.002
Age                 0.0332      0.012      2.754      0.010       0.009
0.058
==============================================================================
=======
Omnibus:                        3.189   Durbin-Watson:
1.623
Prob(Omnibus):                  0.203   Jarque-Bera (JB):
2.307
Skew:                           0.645   Prob(JB):
0.316
Kurtosis:                       3.110   Cond. No.
49.3
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:2: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/Users/deepsha/anaconda3/lib/python3.7/site-packages/statsmodels/compa
t/pandas.py:23: FutureWarning:

The Panel class is removed from pandas. Accessing it from the top-level
namespace will also be removed in the next version
```

# New Liquidity Indicators Calculation on Walmart Bonds and Investment grade Bonds

In [54]:
```
# Investnment grade
Daily_range = merged_df_cusips_all.groupby(['trd_exctn_dt'])[['rptd_pr'
]].max()-merged_df_cusips_all.groupby(['trd_exctn_dt'])[['rptd_pr']].min
()
```

In [486]:
```
# Walmart Bonds
Daily_range_wmt = merged_df.groupby(['trd_exctn_dt'])[['rptd_pr']].max()
-merged_df.groupby(['trd_exctn_dt'])[['rptd_pr']].min()
```

In [488]:
```
Daily_range_wmt['date'] = Daily_range_wmt.index
Daily_range_wmt
```

Out[488]:

| trd_exctn_dt | rptd_pr | date |
|---|---|---|
| **2007-01-02** | 10.68700 | 2007-01-02 |
| **2007-01-03** | 2.17800 | 2007-01-03 |
| **2007-01-04** | 16.15100 | 2007-01-04 |
| **2007-01-05** | 13.70800 | 2007-01-05 |
| **2007-01-08** | 1.71900 | 2007-01-08 |
| **...** | ... | ... |
| **2020-03-25** | 46.50100 | 2020-03-25 |
| **2020-03-26** | 47.56801 | 2020-03-26 |
| **2020-03-27** | 47.63700 | 2020-03-27 |
| **2020-03-30** | 52.28100 | 2020-03-30 |
| **2020-03-31** | 54.87600 | 2020-03-31 |

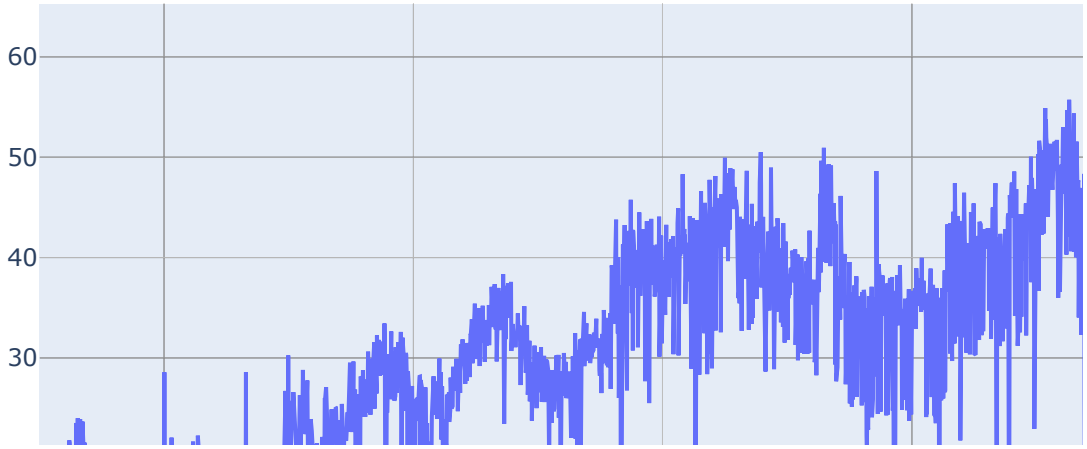
3340 rows × 2 columns

In [371]:
```
Daily_range_wmt['date'] = Daily_range_wmt.index
Daily_range_wmt['year'] = pd.DatetimeIndex(Daily_range_wmt['date']).year
Daily_range_wmt['month'] = pd.DatetimeIndex(Daily_range_wmt['date']).mon
th
Daily_range_wmt["year_month"] = Daily_range_wmt['year'].astype(str) + Da
ily_range_wmt['month'].astype(str)
```

In [372]:
```
Daily_range_wmt = Daily_range_wmt.groupby(['year_month']).mean()
Daily_range_wmt['_year_month'] = Daily_range_wmt.index
Daily_range_wmt['Date']  = pd.to_datetime(Daily_range_wmt['_year_month'
], format='%Y%m')
Daily_range_wmt = Daily_range_wmt.sort_values(by="Date")
```

# Daily RANGE OF Prices of Walmart Bonds

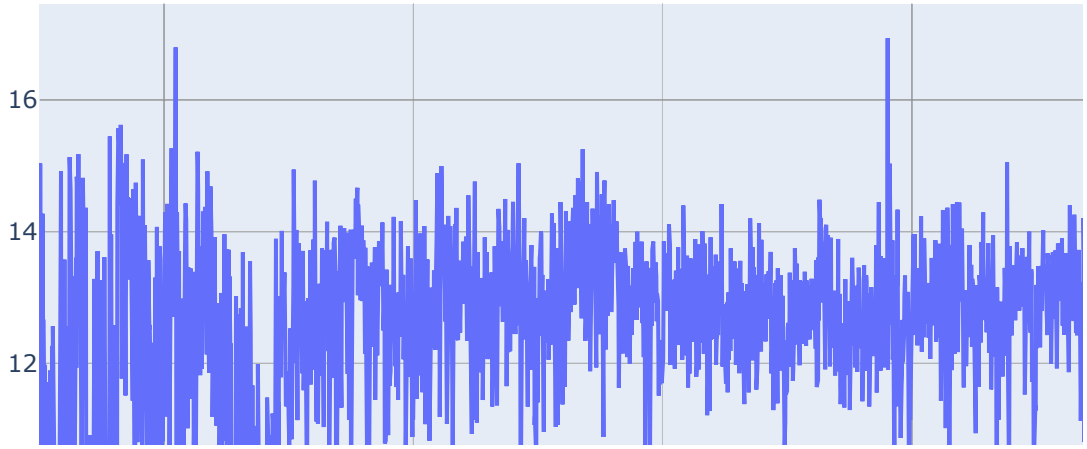
In [489]:
```
fig = go.Figure(data=go.Scatter(x=Daily_range_wmt['date'], y=Daily_range
_wmt['rptd_pr'],name = 'Daily range'))
fig.show("notebook")
```




## Trading Volume of Walmart Bonds

In [526]:
```
trading_vol_WMT = merged_df[["COMPLETE_CUSIP","trd_exctn_dt","entrd_vol_
qt"]]
trading_vol_WMT = trading_vol_WMT.groupby(['trd_exctn_dt']).mean()
trading_vol_WMT['date'] = trading_vol_WMT.index
trading_vol_WMT["inverse_vol"] = -np.log(1/trading_vol_WMT["entrd_vol_q
t"])
trading_vol_WMT["Date"] = trading_vol_WMT["date"]
```

In [502]:
```
fig = go.Figure(data=go.Scatter(x=trading_vol_WMT['date'], y=trading_vol
_WMT["inverse_vol"],name = 'trading_vol_WMT'))
fig.show("notebook")
```




## trading_volume for Investment grade Bonds

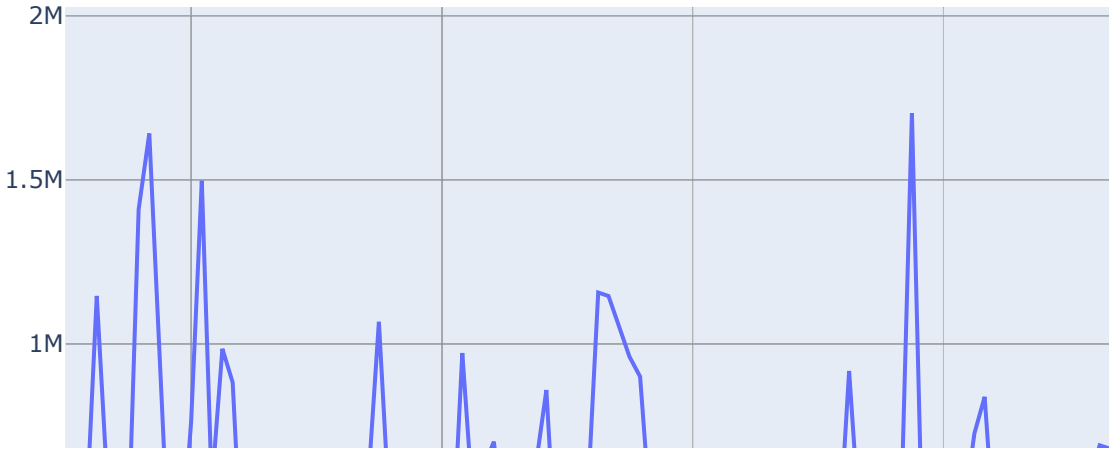
In [374]:
```
trading_vol = merged_df_cusips_all[["COMPLETE_CUSIP","trd_exctn_dt","ent
rd_vol_qt"]]
trading_vol_m = trading_vol.groupby(['trd_exctn_dt']).mean()
trading_vol_m['_year_month'] = trading_vol_m.index
trading_vol_m['Date']  = pd.to_datetime(trading_vol_m['_year_month'], fo
rmat='%Y%m')
trading_vol_m = trading_vol_m.sort_values(by="Date")
trading_vol_m['year'] = pd.DatetimeIndex(trading_vol_m['Date']).year
trading_vol_m['month'] = pd.DatetimeIndex(trading_vol_m['Date']).month
trading_vol_m["year_month"] = trading_vol_m['year'].astype(str) + tradin
g_vol_m['month'].astype(str)
trading_vol_m = trading_vol_m.groupby(['year_month']).mean()
trading_vol_m["inverse_vol"] = 1/trading_vol_m["entrd_vol_qt"]
```

In [386]:
```
trading_vol_mWMT['_year_month'] = trading_vol_mWMT.index
trading_vol_mWMT['Date']  = pd.to_datetime(trading_vol_mWMT['_year_mont
h'], format='%Y%m')
trading_vol_mWMT  = trading_vol_mWMT.sort_values(by="Date")
```

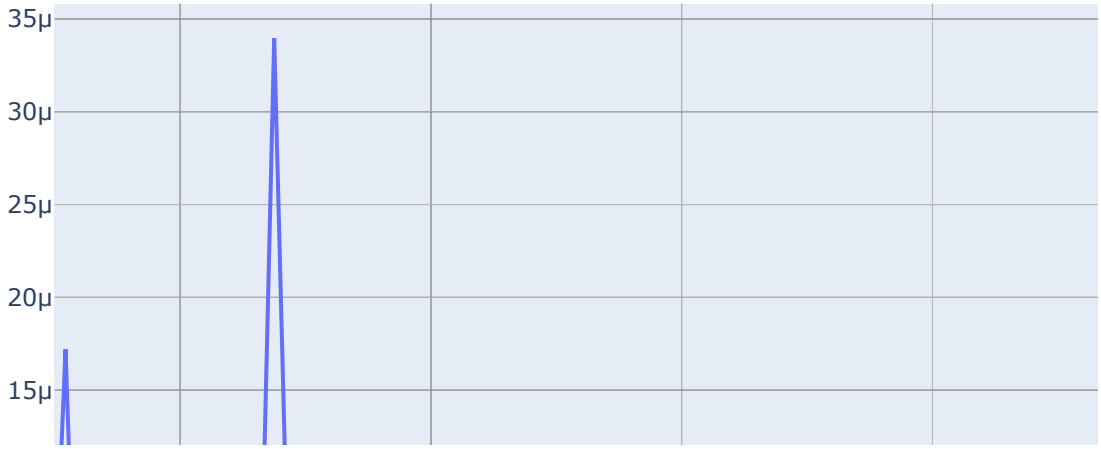
In [388]:
```
trading_vol_mWMT ["inverse_vol"] = 1/trading_vol_mWMT["entrd_vol_qt"]
```

# Plots of Monthly trading Volume and Inverse Volume for Walmart Bonds

In [390]:
```
fig = go.Figure(data=go.Scatter(x=trading_vol_mWMT['Date'], y=trading_vo
l_mWMT['entrd_vol_qt'],name = 'Trading Vol'))
fig.show("notebook")
```



In [392]:
```
fig = go.Figure(data=go.Scatter(x=trading_vol_mWMT['Date'], y=trading_vo
l_mWMT["inverse_vol"] ,name = 'Trading Vol'))
fig.show("notebook")
```

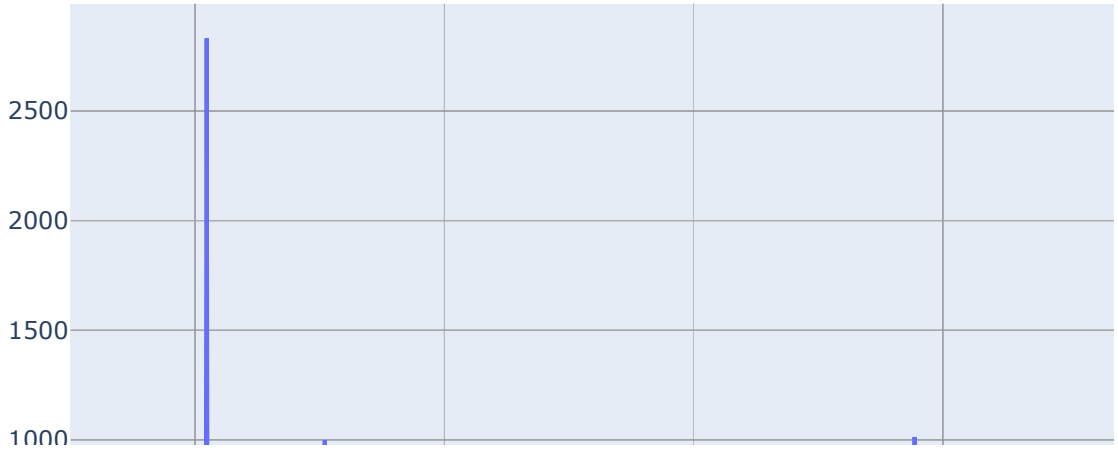



# Amihud Ratio Calculation for Walmart Bonds

In [ ]:
```
uniques_cusips_WMT = merged_df.COMPLETE_CUSIP.unique()
amihud_all_WMT = pd.DataFrame(columns = ['COMPLETE_CUSIP', 'date', 'amih
ud'])
for k in uniques_cusips_WMT:
    x = merged_df[merged_df.COMPLETE_CUSIP == k]
    un = [str(i)[:10] for i in x.trd_exctn_dt.unique()]
    for i in range(0,len(un)):
        p = x[x.trd_exctn_dt == un[i]]['rptd_pr'].values.tolist()
        v = x[x.trd_exctn_dt == un[i]]['entrd_vol_qt'].values.tolist()
        d = np.diff(p)
        if len(d) == 0:
            s = np.nan
        else:
            s = sum([(abs(d[i])/p[i+1]*v[i+1]) for i in range(0,len(d
))])/len(d)
        data = pd.DataFrame({'COMPLETE_CUSIP':k ,'date':un[i],'amihud':
s},index=[0])
        amihud_all_WMT = amihud_all_WMT.append(data)
```

## Plot for Daily Amihud ratio

In [516]:
```
amihud_all_WMT_DAILY = amihud_all_WMT.groupby(['date']).mean()
amihud_all_WMT_DAILY["Date"] = amihud_all_WMT_DAILY.index
fig = go.Figure(data=go.Scatter(x=amihud_all_WMT_DAILY['Date'], y=amihud
_all_WMT_DAILY['amihud']/100,name = 'amihud_all'))
fig.show("notebook")
```



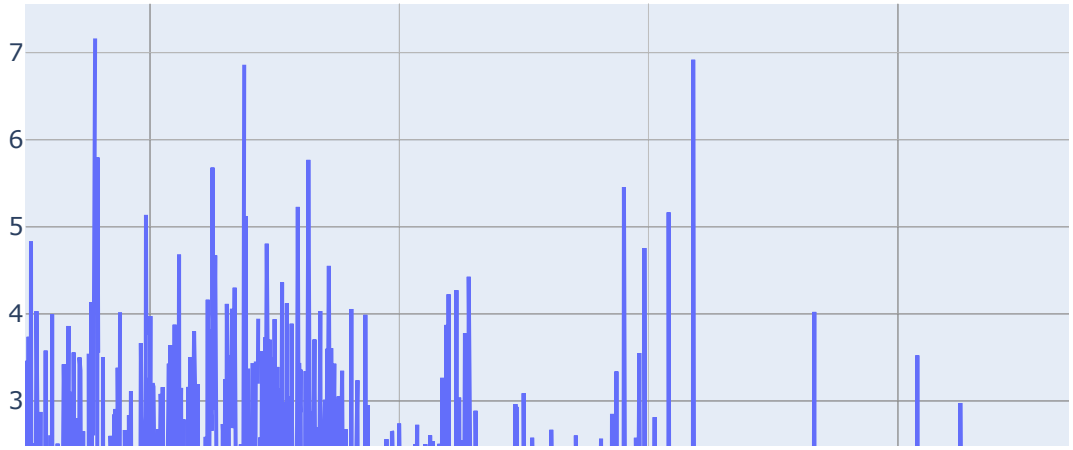

## Monthly Amihud Ratio Calculation

In [ ]:
```
amihud_all_WMT['year'] = pd.DatetimeIndex(amihud_all_WMT['date']).year
amihud_all_WMT['month'] = pd.DatetimeIndex(amihud_all_WMT['date']).month
amihud_all_WMT["year_month"] = amihud_all_WMT['year'].astype(str) + amih
ud_all_WMT['month'].astype(str)
amihud_all_WMT = amihud_all_WMT.groupby(['year_month']).mean()
amihud_all_WMT['_year_month'] = amihud_all_WMT.index
amihud_all_WMT['Date']  = pd.to_datetime(amihud_all_WMT['_year_month'],
format='%Y%m')
amihud_all_WMT = amihud_all_WMT.sort_values(by="Date")
```

# Rolls spread Plot for Walmart Bonds

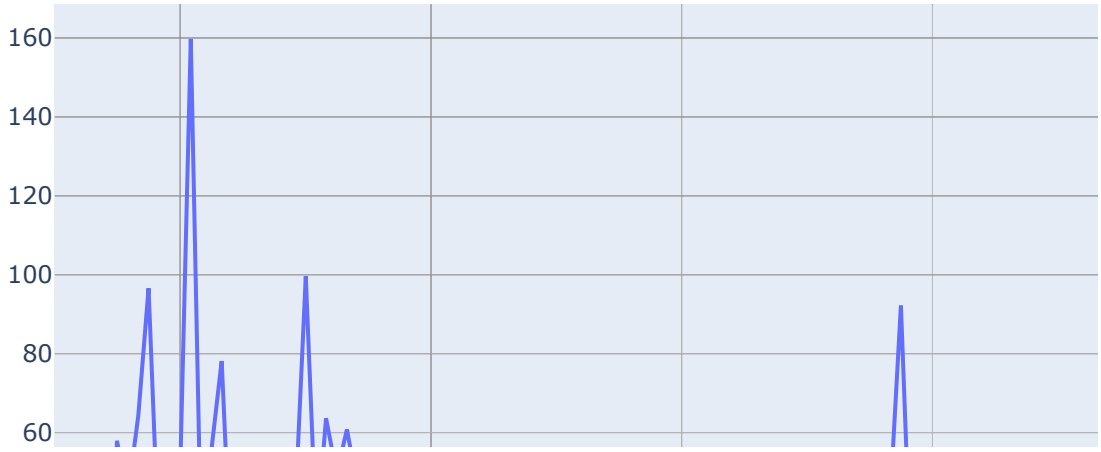
In [523]:
```
test1["Date"] = test1.index
fig = go.Figure(data=go.Scatter(x=test1['Date'], y=test1['rolls_spread'
],name = 'rolls_spread'))
fig.show("notebook")
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```




# Monthly Amihud plot for Walmart Bonds

In [397]:
```
fig = go.Figure(data=go.Scatter(x=amihud_all_WMT['Date'], y=amihud_all_W
MT['amihud']/100,name = 'amihud_all'))
fig.show("notebook")
```




In [400]:
```
amihud_all_s_wmt = amihud_all_WMT[["amihud", "Date"]]
trading_vol_m_s_wmt = trading_vol_mWMT[["entrd_vol_qt", "Date", "inverse
_vol"]]
Daily_range_s_wmt = Daily_range_wmt[["rptd_pr", "Date"]]
```

In [474]:
```
amihud_all_WMT
```

Out[474]:

| year_month | amihud | year | month | _year_month | Date |
|---|---|---|---|---|---|
| **20071** | 1739.829906 | 2007 | 1 | 20071 | 2007-01-01 |
| **20072** | 252.663188 | 2007 | 2 | 20072 | 2007-02-01 |
| **20073** | 1309.790088 | 2007 | 3 | 20073 | 2007-03-01 |
| **20074** | 2218.378820 | 2007 | 4 | 20074 | 2007-04-01 |
| **20075** | 1353.411753 | 2007 | 5 | 20075 | 2007-05-01 |
| **...** | ... | ... | ... | ... | ... |
| **201911** | 630.805226 | 2019 | 11 | 201911 | 2019-11-01 |
| **201912** | 506.733101 | 2019 | 12 | 201912 | 2019-12-01 |
| **20201** | 742.501362 | 2020 | 1 | 20201 | 2020-01-01 |
| **20202** | 715.183241 | 2020 | 2 | 20202 | 2020-02-01 |
| **20203** | 4156.077972 | 2020 | 3 | 20203 | 2020-03-01 |

159 rows × 5 columns

In [404]:
```
merged_pca_wmt = pd.merge(amihud_all_s_wmt,df_lc_rs,how ="inner", on = "Date")
#merged__pca2 = pd.merge(merged_pca_wmt,bonds_d_s,how ="inner", on = "Date")
merged__pca2_wmt = pd.merge(merged_pca_wmt,trading_vol_m_s_wmt,how ="inner", on = "Date")
merged__pca43_wmt = pd.merge(merged__pca2_wmt,Daily_range_s_wmt,how ="inner", on = "Date")
```

In [559]:
```
#Daily data merge

merged_pca_wmt_d = pd.merge(amihud_all_WMT_DAILY,test1,how ="inner", on = "Date")
merged__pca2 = pd.merge(merged_pca_wmt,bonds_d_s,how ="inner", on = "Date")
merged__pca2_wmt_d = pd.merge(merged_pca_wmt_d ,trading_vol_WMT,how ="inner", on = "Date")
merged__pca43_wmt_d= pd.merge(merged__pca2_wmt,df_lc,how ="inner", on = "Date")
merged__pca5 = pd.merge(merged__pca43_wmt_d,Daily_range_wmt,how ="inner", on = "Date")
```

In [558]:
```
Daily_range_wmt["Date"] = pd.to_datetime(Daily_range_wmt["date"]).dt.date
```

In [560]:
```
## daily data for PCA

merged__pca5
```

Out[560]:

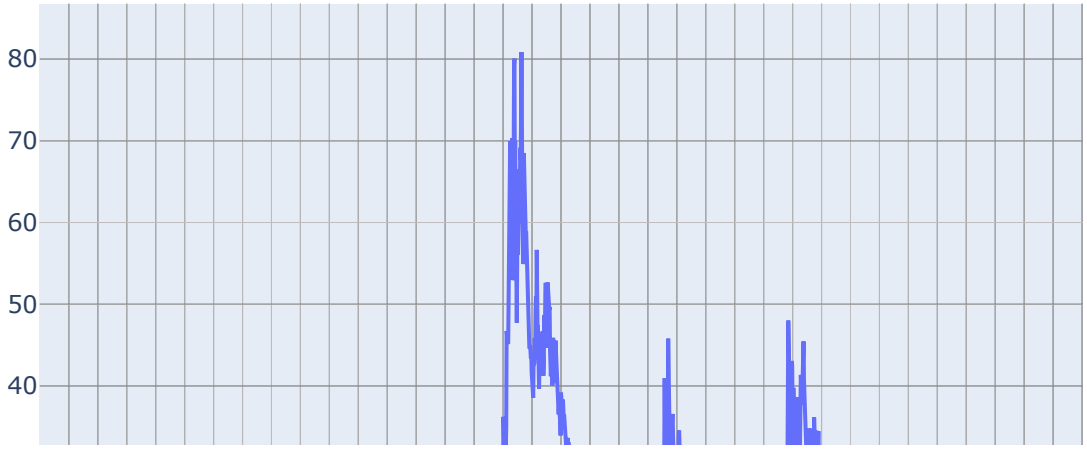
| | amihud | Date | LC_x | rolls_spread | date_x | rs_and_LC_average | entrd_vol_qt | inver |
|---|---|---|---|---|---|---|---|---|
| **0** | 252.663188 | 2007-02-01 | 0.004763 | 2.071908 | 20072 | 1.274095 | 5.815561e+04 | 1.71 |
| **1** | 1309.790088 | 2007-03-01 | 0.002092 | 1.556803 | 20073 | 0.883022 | 4.631617e+05 | 2.15 |
| **2** | 4370.334760 | 2007-08-01 | 0.003869 | 1.615786 | 20078 | 1.001340 | 1.409888e+06 | 7.09 |
| **3** | 2825.686234 | 2007-11-01 | 0.019669 | 2.065796 | 200711 | 2.016327 | 2.709361e+05 | 3.69 |
| **4** | 15974.072014 | 2008-02-01 | 0.017720 | 2.244261 | 20082 | 2.008130 | 1.497256e+06 | 6.67 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **89** | 986.526755 | 2019-05-01 | 0.002442 | 0.466513 | 20195 | 0.355375 | 4.745336e+05 | 2.10 |
| **90** | 934.742874 | 2019-07-01 | 0.002462 | 0.462972 | 20197 | 0.354570 | 4.397811e+05 | 2.27 |
| **91** | 575.729444 | 2019-08-01 | 0.002376 | 0.445645 | 20198 | 0.341646 | 4.112111e+05 | 2.43 |
| **92** | 515.030591 | 2019-10-01 | 0.001932 | 0.422847 | 201910 | 0.308048 | 3.941469e+05 | 2.53 |
| **93** | 630.805226 | 2019-11-01 | 0.002191 | 0.419778 | 201911 | 0.319454 | 3.062157e+05 | 3.26 |

94 rows × 11 columns

## VIX index

In [177]:
```
vix = pd.read_csv("vixcurrent.csv",header=1)
```

In [82]:
```
fig = go.Figure(data=go.Scatter(x=vix['Date'], y=vix['VIX Close'],name =
'amihud_all '))
fig.show("notebook")
```




# Liquidity Indicator Calcultion for Investment grade Bonds

In [87]:
```
amihud_all_s = amihud_all[["amihud", "Date"]]
trading_vol_m_s = trading_vol_m[["entrd_vol_qt", "Date", "inverse_vol"]]
Daily_range_s = Daily_range[["rptd_pr", "Date"]]
```

In [102]:
```
ns_lc = merged_df_cusips_all.groupby(['trd_exctn_dt','rpt_side_cd'])['rp
td_pr'].mean().unstack(level = 1).dropna()
ns_lc['LC'] = (ns_lc['S']- ns_lc['B'])/(0.5*(ns_lc['S']+ ns_lc['B']))
```

In [91]:
```
bonds_d["Date"] = bonds_d.index
bonds_d_s = bonds_d[["rolls_spread", "Date"]]
```

In [106]:
```
ns_lc_s = ns_lc[["LC","Date"]]
```

In [104]:
```
ns_lc['Date'] = ns_lc.index
ns_lc['year'] = pd.DatetimeIndex(ns_lc['Date']).year
ns_lc['month'] = pd.DatetimeIndex(ns_lc['Date']).month
ns_lc["year_month"] = ns_lc['year'].astype(str) + ns_lc['month'].astype(
str)
```

In [108]:
```
ns_lc = ns_lc.groupby(["year_month"])
ns_lc_ma = ns_lc.aggregate({"LC":np.mean})
ns_lc_ma['_year_month'] = ns_lc_ma.index
ns_lc_ma['Date']  = pd.to_datetime(ns_lc_ma['_year_month'], format='%Y%
m')
ns_lc_ma = ns_lc_ma.sort_values(by="Date")
```

## Data Preparation for PCA of Investment grade Bonds

In [118]:
```
merged_pca = pd.merge(amihud_all_s,ns_lc_s,how ="inner", on = "Date")
merged__pca2 = pd.merge(merged_pca ,bonds_d_s,how ="inner", on = "Date")
merged__pca3 = pd.merge(merged__pca2,trading_vol_m_s,how ="inner", on =
"Date")
merged__pca4 = pd.merge(merged__pca3,Daily_range_s,how ="inner", on = "D
ate")
```

In [561]:
```
merged__pca5["amihud"] = merged__pca5["amihud"]/100
pca4 = merged__pca4[["amihud","LC","rolls_spread","inverse_vol","rptd_p
r"]]
pca5 = merged__pca5[["amihud","LC_x","rolls_spread","inverse_vol","rptd_
pr"]]
```

In [407]:
```
pca4_wmt = merged__pca43_wmt[["amihud","LC","rolls_spread","inverse_vol"
,"rptd_pr"]]
```

## PCA on Walmart Bonds

In [128]:
```
from sklearn.preprocessing import StandardScaler
pca4_wmt = StandardScaler().fit_transform(pca4_wmt)
pca_ = PCA(n_components=2)
principalComponents_ = pca_.fit_transform(pca4_wmt)
```

In [ ]:
```
feat_cols = ['feature'+str(i) for i in range(pca4_wmt.shape[1])]
normalised_feat = pd.DataFrame(pca4_wmt,columns=feat_cols)
principal_Df = pd.DataFrame(data = principalComponents_
             , columns = ['principal component 1', 'principal component
 2'])
```

In [ ]:
```
feature_weights = pca_.components_
squared_factor_weights = np.array([[i*i for i in k] for k in feature_wei
ghts ]).T
max_el = [{list(i).index(max(i)): max(i)} for i in squared_factor_weight
s]
```

# PCA on Investment grade Bonds

In [567]:
```
from sklearn.decomposition import PCA

pca5 = StandardScaler().fit_transform(pca5)

pca_ = PCA(n_components=2)
principalComponents_ = pca_.fit_transform(pca5)
```

In [568]:
```
np.mean(pca5),np.std(pca5)
```

Out[568]:
```
(-1.209434443846979e-16, 1.0)
```

In [569]:
```
feat_cols = ['feature'+str(i) for i in range(pca5.shape[1])]
normalised_feat = pd.DataFrame(pca5,columns=feat_cols)
```

In [571]:
```
normalised_feat.head()
```

Out[571]:

| | feature0 | feature1 | feature2 | feature3 | feature4 |
|---|---|---|---|---|---|
| **0** | -1.046565 | -0.495242 | 1.349736 | 8.018346 | -3.019824 |
| **1** | -0.532159 | -1.070150 | 0.542736 | -0.091347 | -1.613716 |
| **2** | 0.957127 | -0.687684 | 0.635143 | -0.873288 | -2.964957 |
| **3** | 0.205489 | 2.713797 | 1.340160 | 0.734841 | -1.559897 |
| **4** | 6.603598 | 2.294287 | 1.619756 | -0.895610 | -1.513260 |

In [573]:
```
principal_Df = pd.DataFrame(data = principalComponents_
             , columns = ['principal component 1', 'principal component
 2'])
```

In [574]:
```
principal_Df.head()
```

Out[574]:

| | principal component 1 | principal component 2 |
|---|---|---|
| **0** | 1.996762 | 8.063796 |
| **1** | 0.287687 | 0.669545 |
| **2** | 1.754688 | -0.368914 |
| **3** | 3.088195 | 0.918712 |
| **4** | 5.480346 | -3.528773 |

In [575]:
```
print('Explained variation per principal component: {}'.format(pca_.explained_variance_ratio_))
```

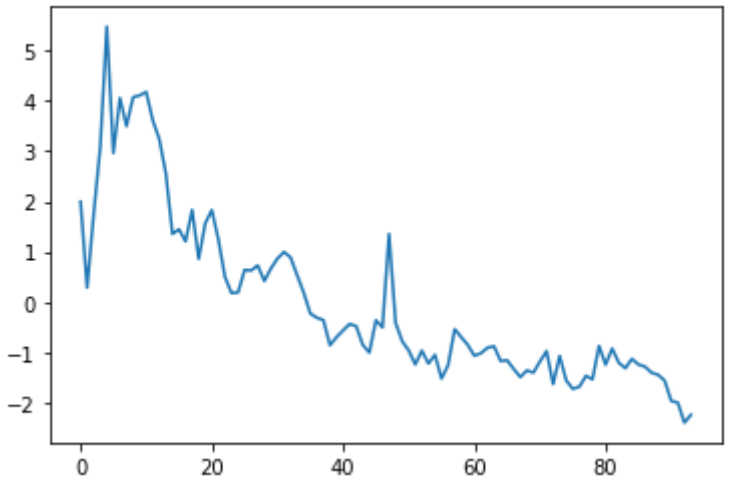
```
Explained variation per principal component: [0.58483393 0.25791861]
```

In [576]:
```
pca_.explained_variance_ratio_[0]
```

Out[576]:
```
0.5848339336975913
```

In [577]:
```
#x_std = pca.transform(transformed)plt.figure()
plt.plot(principal_Df.iloc[:,0])
```

Out[577]:
```
[<matplotlib.lines.Line2D at 0x16bdda390>]
```




# Partial and Final Weight Calculation

In [578]:
```
feature_weights = pca_.components_
```

In [579]:
```
feature_weights
```

Out[579]:
```
array([[ 0.40340911,  0.53835853,  0.55592833,  0.0588107 , -0.48468153],
       [-0.48686855, -0.03335338,  0.07653984,  0.83167913, -0.25357059]])
```

In [582]:
```
squared_factor_weights
```

Out[582]:
```
array([[0.16273891, 0.23704098],
       [0.2898299 , 0.00111245],
       [0.3090563 , 0.00585835],
       [0.0034587 , 0.69169018],
       [0.23491619, 0.06429805]])
```

In [581]:
```
squared_factor_weights = np.array([[i*i for i in k] for k in feature_wei
ghts ]).T
max_el = [{list(i).index(max(i)): max(i)} for i in squared_factor_weight
s]
```

In [585]:
```
import collections
counter = collections.Counter()
for d in max_el:
    counter.update(d)
m = dict(counter)
```

In [586]:
```
p_final = [[v/m[k]   for k, v in max_el[i].items()]for i in range(0, len
(max_el))]
flat_p_final = [item for sublist in p_final for item in sublist]
flat_w_final = [i*pca_.explained_variance_ratio_[0] for i in flat_p_fina
l]
flat_w_final[0] = (flat_w_final[0]/pca_.explained_variance_ratio_[0])*pc
a_.explained_variance_ratio_[1]
flat_w_final[3] = (flat_w_final[3]/pca_.explained_variance_ratio_[0])*pc
a_.explained_variance_ratio_[1]
```

In [590]:
```
sum(flat_w_final)
```

Out[590]:
```
0.842752545977197
```

In [591]:
```
normalised_feat.loc[:, ["feature0", "feature1", "feature2", "feature3",
"feature4"]] *= np.array(flat_w_final)
normalised_feat["new_index" ] = normalised_feat["feature0"]+normalised_f
eat["feature1"]+normalised_feat["feature2"]+normalised_feat["feature3"]+
normalised_feat["feature4"]
```

In [592]:
```
merged__pca5["new_index"]= normalised_feat["new_index"]
```

In [593]:
```
merged__pca5
```

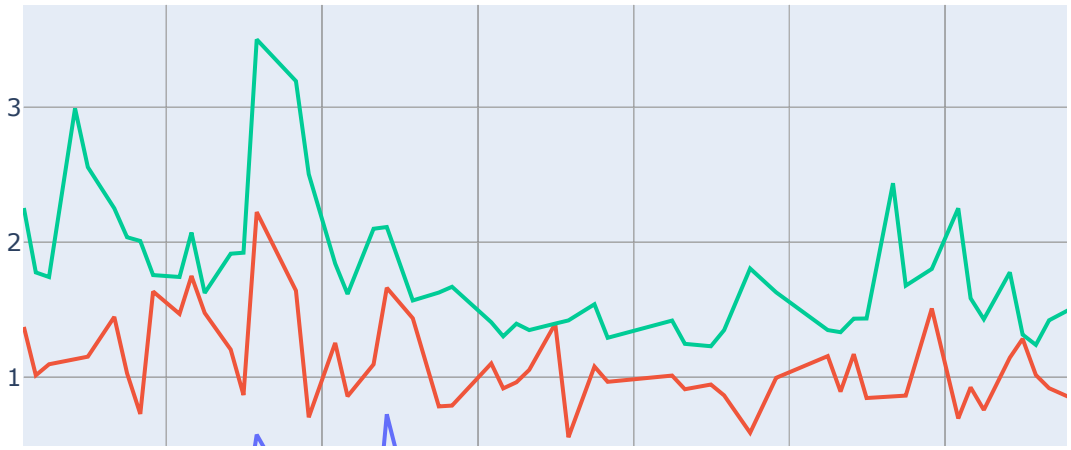
Out[593]:

| | amihud | Date | LC_x | rolls_spread | date_x | rs_and_LC_average | entrd_vol_qt | inverse |
|---|---|---|---|---|---|---|---|---|
| **0** | 2.526632 | 2007-02-01 | 0.004763 | 2.071908 | 20072 | 1.274095 | 5.815561e+04 | 1.7195 |
| **1** | 13.097901 | 2007-03-01 | 0.002092 | 1.556803 | 20073 | 0.883022 | 4.631617e+05 | 2.1590 |
| **2** | 43.703348 | 2007-08-01 | 0.003869 | 1.615786 | 20078 | 1.001340 | 1.409888e+06 | 7.0927 |
| **3** | 28.256862 | 2007-11-01 | 0.019669 | 2.065796 | 200711 | 2.016327 | 2.709361e+05 | 3.6909 |
| **4** | 159.740720 | 2008-02-01 | 0.017720 | 2.244261 | 20082 | 2.008130 | 1.497256e+06 | 6.6788 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **89** | 9.865268 | 2019-05-01 | 0.002442 | 0.466513 | 20195 | 0.355375 | 4.745336e+05 | 2.1073 |
| **90** | 9.347429 | 2019-07-01 | 0.002462 | 0.462972 | 20197 | 0.354570 | 4.397811e+05 | 2.2738 |
| **91** | 5.757294 | 2019-08-01 | 0.002376 | 0.445645 | 20198 | 0.341646 | 4.112111e+05 | 2.4318 |
| **92** | 5.150306 | 2019-10-01 | 0.001932 | 0.422847 | 201910 | 0.308048 | 3.941469e+05 | 2.5371 |
| **93** | 6.308052 | 2019-11-01 | 0.002191 | 0.419778 | 201911 | 0.319454 | 3.062157e+05 | 3.2656 |

94 rows × 12 columns

# Comparison of Monthly Liquidity Indicators with VIX for Investment grade Bonds

In [275]:
```
fig = go.Figure(data=go.Scatter(x=merged__pca4['Date'], y=merged__pca4[
'new_index'],name = 'syn'))
#fig.add_scatter(x=merged__pca4['Date'], y=merged__pca4['LC'],name = 'L
C')
fig.add_scatter(x=merged__pca4['Date'], y=merged__pca4['rolls_spread'],n
ame = 'rolls_spread')
#Fig.add_scatter(x=merged__pca4['Date'], y=merged__pca4['inverse_vol'],n
ame = 'inverse_vol')
fig.add_scatter(x=mmm['Date'], y=mmm['VIX Close']/10,name = 'VIX Close')
fig.show("notebook")
```




## FINAL Walmart Data with Liquidity Measures

In [600]:
```
mmm_wmt = pd.merge(merged__pca43_wmt,vix_ma, on = "Date")
mmm_wmt
```

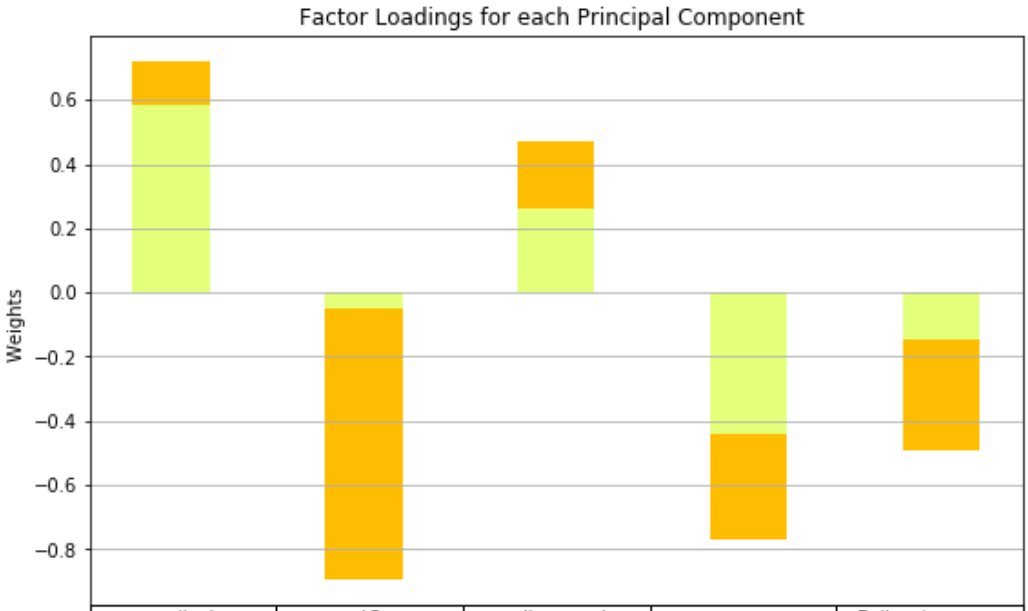
Out[600]:

| | amihud | Date | LC | rolls_spread | date | rs_and_LC_average | entrd_vol_qt | inver |
|---|---|---|---|---|---|---|---|---|
| **0** | 1739.829906 | 2007-01-01 | 0.000511 | 2.054803 | 20071 | 1.052958 | 2.891318e+05 | 3.45 |
| **1** | 252.663188 | 2007-02-01 | 0.004763 | 2.071908 | 20072 | 1.274095 | 5.815561e+04 | 1.71 |
| **2** | 1309.790088 | 2007-03-01 | 0.002092 | 1.556803 | 20073 | 0.883022 | 4.631617e+05 | 2.15 |
| **3** | 2218.378820 | 2007-04-01 | 0.005565 | 1.663898 | 20074 | 1.110200 | 1.146345e+06 | 8.72 |
| **4** | 1353.411753 | 2007-05-01 | 0.009949 | 1.688256 | 20075 | 1.341571 | 5.863556e+05 | 1.70 |
| **...** | ... | ... | ... | ... | ... | ... | ... | |
| **154** | 630.805226 | 2019-11-01 | 0.002191 | 0.419778 | 201911 | 0.319454 | 3.062157e+05 | 3.26 |
| **155** | 506.733101 | 2019-12-01 | 0.002401 | 0.389722 | 201912 | 0.314898 | 3.307363e+05 | 3.02 |
| **156** | 742.501362 | 2020-01-01 | 0.002083 | 0.429784 | 20201 | 0.319032 | 4.090183e+05 | 2.44 |
| **157** | 715.183241 | 2020-02-01 | 0.001887 | 0.390116 | 20202 | 0.289405 | 4.179085e+05 | 2.39 |
| **158** | 4156.077972 | 2020-03-01 | 0.005801 | 1.326876 | 20203 | 0.953508 | 5.548511e+05 | 1.80 |

159 rows × 12 columns

# Factor Loading Plots for Walmart Daily Data

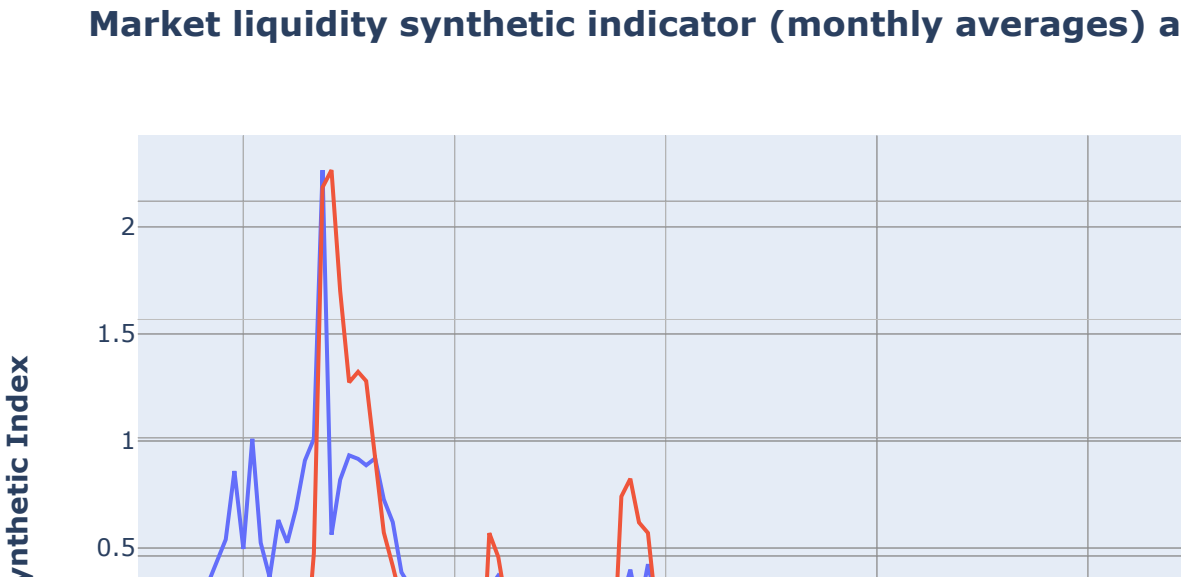
In [399]:
```
columns = ('amihud', 'LC', 'rolls_spread', 'inverse_vol', 'Daily price r
ange')
rows = ['Principal Component %d' % x for x in (1,2)]
plt.figure(figsize=(8,5))

values = np.arange(-1.0, 1, 0.1)
value_increment = 1

# Get some pastel shades for the colors
colors = plt.cm.Wistia(np.linspace(0, 0.5, len(rows)))
n_rows = len(feature_weights)
index = np.arange(len(columns)) + 0.3
bar_width = 0.4

y_offset = np.zeros(len(columns))

cell_text = []
for row in range(n_rows):
    plt.bar(index, feature_weights[row], bar_width, bottom=y_offset, col
or=colors[row])
    y_offset = y_offset + feature_weights[row]
    cell_text.append(['%1.2f' % (x) for x in y_offset])
# Reverse colors and text labels to display the last value at the top.
colors = colors[::-1]
#cell_text.reverse()



# Add a table at the bottom of the axes
the_table = plt.table(cellText=cell_text,
                      rowLabels=rows,
                      rowColours=colors,
                      colLabels=columns,
                      loc='bottom')

# Adjust layout to make room for the table:
plt.subplots_adjust(left=0.01, bottom=0.01)

plt.ylabel("Weights ")
#plt.yticks(values * value_increment, ['%d' % val for val in values])
plt.xticks([])
plt.title('Factor Loadings for each Principal Component ')
plt.grid(True)
plt.show()
```




| | amihud | LC | rolls_spread | inverse_vol | Daily price range |
|---|---|---|---|---|---|
| Principal Component 1 | 0.58 | -0.05 | 0.47 | -0.44 | -0.49 |
| Principal Component 2 | 0.72 | -0.89 | 0.26 | -0.77 | -0.15 |

# New Liquidity Indec vs VIX index

In [484]:
```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
fig = make_subplots(specs=[[{"secondary_y": True}]])



fig.add_trace(
    go.Scatter(x=merged__pca43_wmt['Date'], y=merged__pca43_wmt['new_ind
ex'], name='synthetic Index'),
    secondary_y=False,
)

fig.add_trace(
    go.Scatter(x=mmm_wmt['Date'], y=mmm_wmt['VIX Close'],name = 'VIX'),
    secondary_y=True,
)

fig.update_layout(
    title_text="<b> Market liquidity synthetic indicator (monthly averag
es) and VIX index</b> "
)

# Set x-axis title
fig.update_xaxes(title_text="years -->")

# Set y-axes titles
fig.update_yaxes(title_text="<b>Synthetic Index</b> ", secondary_y=False
)
fig.update_yaxes(title_text="<b>VIX</b>", secondary_y=True)


fig.show("notebook")
```




# New Liquidity Index Created

In [598]:
```
fig = go.Figure(data=go.Scatter(x=merged__pca5['Date'], y=(merged__pca5[
'new_index']),name = 'rolls_spread'))
fig.show("notebook")
```


In [606]:
```
plot_d = mmm_wmt[["Date","amihud","rolls_spread","inverse_vol","rptd_pr"
,"LC"]]
plot_d.head()
```

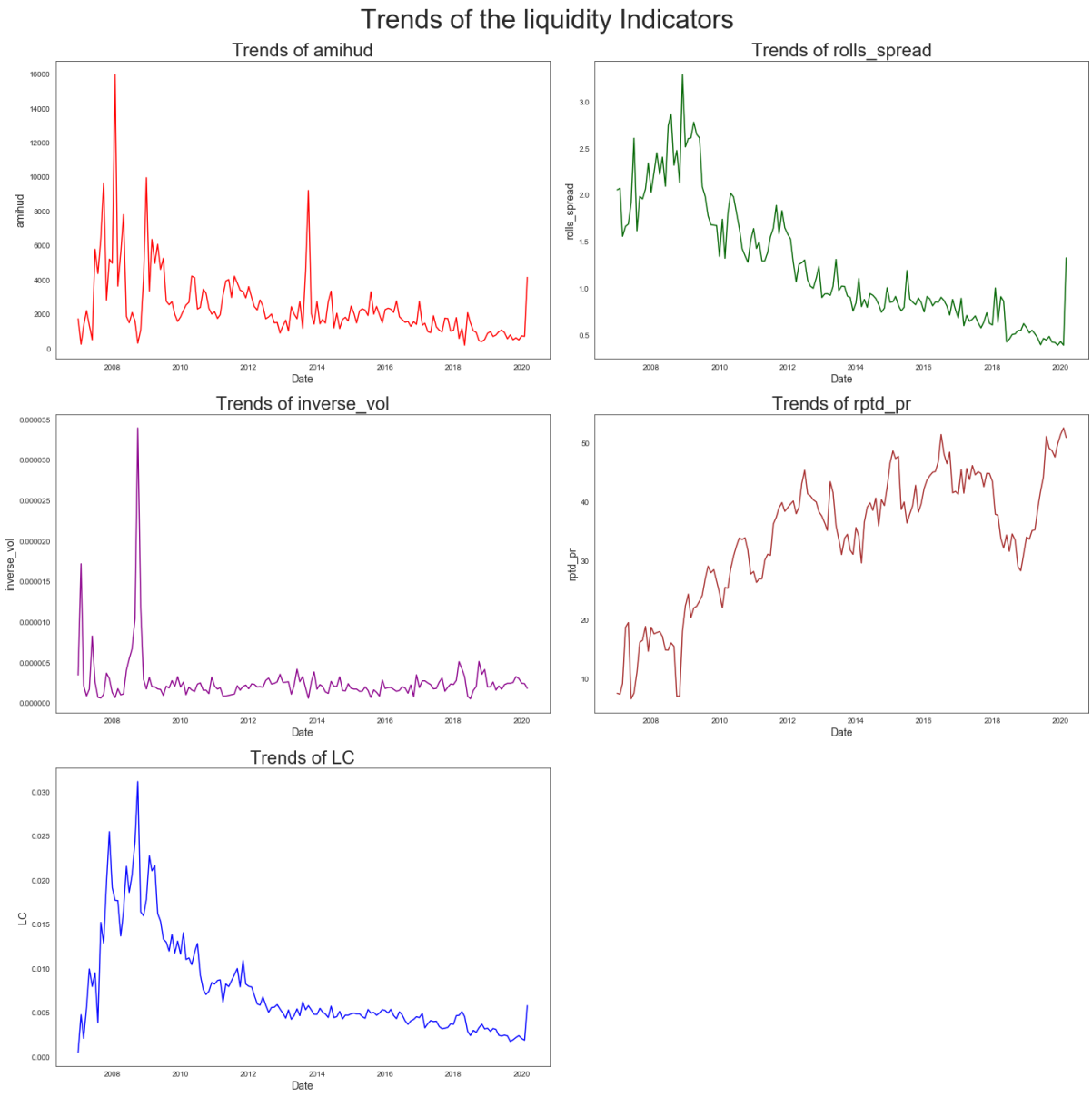
Out[606]:

| | Date | amihud | rolls_spread | inverse_vol | rptd_pr | LC |
|---|---|---|---|---|---|---|
| **0** | 2007-01-01 | 1739.829906 | 2.054803 | 3.458630e-06 | 7.480328 | 0.000511 |
| **1** | 2007-02-01 | 252.663188 | 2.071908 | 1.719525e-05 | 7.343895 | 0.004763 |
| **2** | 2007-03-01 | 1309.790088 | 1.556803 | 2.159073e-06 | 9.173682 | 0.002092 |
| **3** | 2007-04-01 | 2218.378820 | 1.663898 | 8.723377e-07 | 18.696334 | 0.005565 |
| **4** | 2007-05-01 | 1353.411753 | 1.688256 | 1.705450e-06 | 19.519186 | 0.009949 |

# Plot trends for Liquidity Indicators

In [621]:
```
keylist = list(plot_d.columns[1:])
colors = ['Red', 'darkgreen', 'darkmagenta', 'brown', 'blue'][:len(keyli
st)]
colors_val = dict(zip(keylist, colors))
```

In [622]:
```
req_rows = int(len(keylist)/2) if len(keylist)%2==0 else int(len(keylis
t)/2) + 1
fig, axs = plt.subplots(nrows=req_rows, ncols=2, figsize = (20,20), cons
trained_layout = True)
i = 0
index = 0
while i < len(keylist):
    j=0
    while j < 2:
        sns.set_style('white')
        axs[index,j].plot(plot_d['Date'], plot_d[keylist[i]], color = co
lors_val[keylist[i]], label = keylist[i][:])
        axs[index,j].set_xlabel('Date', size = 14)
        axs[index,j].set_ylabel(keylist[i][:], size = 14)
        axs[index,j].set_title('Trends of ' + keylist[i], size = 24)
        # For Legend
        ln_1, lab_1 = axs[index,j].get_legend_handles_labels()
        ax.legend(ln_1, lab_1, loc=0)

        j = j + 1
        i = i + 1
        if (i == len(keylist)) & (j != 2):
            axs[index,j].axis("off")
            break
    index = index + 1
plt.suptitle('Trends of the liquidity Indicators', size = 35)
fig.savefig('Trends of the liquidity Indicators.png')
plt.show()
```




In [627]:
```
merged_3.head()
```

Out[627]:

| | COMPLETE_CUSIP | rolls_spread_x | issue_size | spread_Duration | trailing_vol | Age | log_ |
|---|---|---|---|---|---|---|---|
| **0** | 931142AU7 | 1.403100 | 2.500000e+08 | 2.508518 | 0.247132 | 19.865254 | |
| **1** | 931142BF9 | 1.988638 | 1.000000e+09 | 12.974996 | 1.959841 | 11.598015 | |
| **2** | 931142CB7 | 1.285224 | 2.500000e+09 | 13.131005 | 4.685502 | 8.029763 | |
| **3** | 931142CH4 | 1.728408 | 7.500000e+08 | 7.866904 | 1.452575 | 5.045386 | |
| **4** | 931142CK7 | 1.536798 | 2.250000e+09 | 18.528089 | 16.459191 | 4.314677 | |

In [656]:
```
keylist = list(merged_3.columns[2:])
colors = ['darkorchid', 'seagreen', 'darkmagenta', 'orangered', 'navy']
colors_val = dict(zip(keylist, colors))
```

# Grid Plot for Bond Characteristics

In [ ]:
```
req_rows = int(len(keylist)/2) if len(keylist)%2==0 else int(len(keylis
t)/2) + 1
fig, axs = plt.subplots(nrows=req_rows, ncols=2, figsize = (20,20), cons
trained_layout = True)
i = 0
index = 0
while i < len(keylist):
    j=0
    while j < 2:
        sns.set_style('white')
        sns.jointplot(x = merged_3[keylist[i][:]], y =merged_3["rolls_sp
read_x"], color = colors_val[keylist[i]], ax = axs[index,j],
                      label=keylist[i][:],kind="reg")
        axs[index,j].set_xlabel(keylist[i][:], size = 14)
        axs[index,j].set_ylabel('rolls_spread', size = 14)
        axs[index,j].set_title('rolls_spread vs ' + keylist[i][:], size
= 24)


        # For Legend
        ln_1, lab_1 = axs[index,j].get_legend_handles_labels()

        ax.legend(ln_1, lab_1, loc=0)

        j = j + 1
        i = i + 1
        if (i == len(keylist)) & (j != 2):
            axs[index,j].axis("off")
            break
    index = index + 1
plt.suptitle('Regression plot wrt Bond Characteristics', size = 35)
fig.savefig('Regression plot wrt Bond Characteristics.png')
```

# Dealer's run

In [675]:
```
import numpy as np
import scipy
import scipy.optimize as optimize
import datetime
import pandas as pd
import pandas_datareader.data as web
import matplotlib.pyplot as plt
```

In [686]:
```
Data_RUN = pd.read_csv('Temp_Data_Runs.csv')
Data_RUN.head()
```

Out[686]:

| | Time | ISIN | Price | Size | Axe | Dealer | Side | PriceType | Vendor | t |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2020-06-15T08:11:37Z | US92343VBR42 | 113.614 | 0.0 | I | BANCO BILBAO VIZCAYA | Offer | Percentage | Dealer Runs | |
| **1** | 2020-06-03T07:01:38Z | US92343VBR42 | 114.750 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | |
| **2** | 2020-06-17T11:10:33Z | US92343VBR42 | 113.922 | 0.0 | I | BANCO BILBAO VIZCAYA | Offer | Percentage | Dealer Runs | |
| **3** | 2020-06-08T15:16:47Z | US92343VBR42 | 114.300 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | |
| **4** | 2020-06-02T09:00:18Z | US92343VBR42 | 114.430 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | |

In [685]:
```
Data_RUN['trd_exctn_dt'] = pd.to_datetime(Data_RUN['Time'], format='%Y-%m-%dT%H:%M:%S.%f')
Data_RUN['trd_exctn_dt'] = Data_RUN['trd_exctn_dt'].dt.date
```

In [687]:
```
start = datetime.datetime(2007, 1, 30)
end = datetime.datetime(2020, 3, 31)
oas_percent = web.DataReader("BAMLC0A4CBBB", "fred", start, end)
yield_10y = web.DataReader("DGS10", "fred", start, end)
```

In [707]:
```
# filtering data to actionable prices:
Data_RUN_A = Data_RUN[Data_RUN.Axe == "A"]
Data_RUN_A = Data_RUN_A[Data_RUN_A.PriceType != "Spread"]
Data_RUN_A.head()
```

Out[707]:

| | Time | ISIN | Price | Size | Axe | Dealer | Side | PriceType | Vendor | tr |
|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 2020-06-03T07:01:38Z | US92343VBR42 | 114.75 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | 2 |
| **3** | 2020-06-08T15:16:47Z | US92343VBR42 | 114.30 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | 2 |
| **4** | 2020-06-02T09:00:18Z | US92343VBR42 | 114.43 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | 2 |
| **5** | 2020-06-10T10:19:12Z | US92343VBR42 | 113.91 | 660000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | 2 |
| **7** | 2020-06-10T09:34:08Z | US92343VBR42 | 113.91 | 660000.0 | A | JP MORGAN | Offer | Percentage | Dealer Runs | 2 |

In [719]:
```
def Px(Rate,Mkt_Price,Face,Freq,N,C):
    return Mkt_Price - (Face * ( 1 + Rate / Freq ) ** ( - N ) + ( C / Ra
te ) * ( 1 - (1 + ( Rate / Freq )) ** -N ) )

def YieldCalc(guess,Mkt_Price,Face,Freq,N,C):
    x = scipy.optimize.newton(Px, guess,args = (Mkt_Price,Face,Freq,N,C
), tol=.0000001, maxiter=100)
    return x
Data_RUN_A['price_diff'] = Data_RUN_A.groupby(['trd_exctn_dt','ISIN'])[
'Price'].diff().fillna(0)
```

In [709]:
```
Data_RUN_A['cusip_id'] = Data_RUN_A["ISIN"].apply(lambda x: x[2:-1])
```

In [711]:
```
fisd = pd.read_csv('fisd.csv')
fisd.drop(fisd.columns[0], axis=1, inplace=True)
fisd_offer = fisd.loc[fisd['COMPLETE_CUSIP'].isin(Data_RUN_A['cusip_id']
.unique())][['COMPLETE_CUSIP','OFFERING_DATE', 'COUPON', 'MATURITY']]
fisd_offer
```

Out[711]:

| | COMPLETE_CUSIP | OFFERING_DATE | COUPON | MATURITY |
|---|---|---|---|---|
| **309710** | 92343VBR4 | 2013-09-11 | 5.15 | 2023-09-15 |

In [717]:
```
Data_RUN_A_merge = pd.merge(Data_RUN_A, fisd_offer, left_on='cusip_id',
right_on='COMPLETE_CUSIP')
Data_RUN_A_merge['MATURITY'] = pd.to_datetime(Data_RUN_A_merge['MATURIT
Y'], format='%Y-%m-%d')
Data_RUN_A_merge['OFFERING_DATE'] = pd.to_datetime(Data_RUN_A_merge['OFF
ERING_DATE'], format='%Y-%m-%d')
Data_RUN_A_merge['trd_exctn_dt'] = pd.to_datetime(Data_RUN_A_merge['trd_
exctn_dt'], format='%Y-%m-%d')

Data_RUN_A_merge['N_PERIODS'] = (((Data_RUN_A_merge['MATURITY'] - Data_R
UN_A_merge['trd_exctn_dt'])/ np.timedelta64(1, 'Y'))*2).astype(int)

Data_RUN_A_merge['YTM'] = Data_RUN_A_merge.apply(lambda x: YieldCalc(x[
'COUPON']/100.0, x['Price'], 100, 2, x['N_PERIODS'], x['COUPON']), axis=
1)
Data_RUN_A_merge['Age'] = (Data_RUN_A_merge['trd_exctn_dt'] - Data_RUN_A
_merge['OFFERING_DATE'])/ np.timedelta64(1, 'D')
```

In [722]:
```
Data_RUN_A_merge
```

Out[722]:

| | Time | ISIN | Price | Size | Axe | Dealer | Side | PriceType | Vend |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 2020-06-03T07:01:38Z | US92343VBR42 | 114.750 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Deal Rur |
| **1** | 2020-06-08T15:16:47Z | US92343VBR42 | 114.300 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Deal Rur |
| **2** | 2020-06-02T09:00:18Z | US92343VBR42 | 114.430 | 1000000.0 | A | JP MORGAN | Offer | Percentage | Deal Rur |
| **3** | 2020-06-10T10:19:12Z | US92343VBR42 | 113.910 | 660000.0 | A | JP MORGAN | Offer | Percentage | Deal Rur |
| **4** | 2020-06-10T09:34:08Z | US92343VBR42 | 113.910 | 660000.0 | A | JP MORGAN | Offer | Percentage | Deal Rur |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **38312** | 2020-03-12T13:32:59Z | US92343VBR42 | 110.206 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPM F |
| **38313** | 2020-03-12T13:33:36Z | US92343VBR42 | 110.188 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPM F |
| **38314** | 2020-03-12T13:33:54Z | US92343VBR42 | 110.201 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPM F |
| **38315** | 2020-03-12T13:33:58Z | US92343VBR42 | 110.210 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPM F |
| **38316** | 2020-03-12T13:37:44Z | US92343VBR42 | 110.221 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPM F |

38317 rows × 19 columns

In [723]:
```
def duration(c, y, m, n):
    macaulay_duration = ((1+y) / (m*y)) - ( (1 + y + n*(c-y)) / ((m*c*
((1+y)**n - 1)) + m*y) )
    modified_duration = macaulay_duration / (1 + y)
    return macaulay_duration, modified_duration
```

In [724]:
```
duration_ = duration(Data_RUN_A_merge['COUPON'].astype('float')/100.0, D
ata_RUN_A_merge['YTM']/2.0, 2.0, Data_RUN_A_merge['N_PERIODS'])
```

In [725]:
```
Data_RUN_A_merge['mac_duration'] = duration_[0]
Data_RUN_A_merge['mod_duration'] = duration_[1]
Data_RUN_A_merge = pd.merge(Data_RUN_A_merge, yield_10y, left_on='trd_ex
ctn_dt', right_on='DATE')
Data_RUN_A_merge = pd.merge(Data_RUN_A_merge, oas_percent, left_on='trd_
exctn_dt', right_on='DATE')
```

In [726]:
```
Data_RUN_A_merge = Data_RUN_A_merge.rename(columns={'BAMLC0A4CBBB': 'BAM
L_OAS', 'DGS10': 'Yield_10YT'})
Data_RUN_A_merge['spread'] = Data_RUN_A_merge['YTM']*100 - Data_RUN_A_me
rge['Yield_10YT']

t_diff = Data_RUN_A_merge.groupby(['trd_exctn_dt', 'cusip_id'])['Yield_1
0YT'].mean().diff().fillna(0).reset_index()#['Yield_10YT'].diff()
t_diff = t_diff.rename(columns={'Yield_10YT':'T_diff'})
Data_RUN_A_merge_1 = Data_RUN_A_merge.merge(t_diff, left_on=['trd_exctn_
dt','cusip_id'], right_on = ['trd_exctn_dt','cusip_id'], how='left')
```

In [728]:
```
Data_RUN_A_merge_1['yield_change'] = Data_RUN_A_merge_1['mod_duration']*
Data_RUN_A_merge_1['T_diff']

Data_RUN_A_merge_1['spread_duration'] = Data_RUN_A_merge_1['spread']*Dat
a_RUN_A_merge_1['mod_duration']
Data_RUN_A_merge_1['spread_change'] = Data_RUN_A_merge_1.groupby(['trd_e
xctn_dt','cusip_id'])['spread_duration'].diff()


Data_RUN_A_merge_2 = Data_RUN_A_merge_1[Data_RUN_A_merge_1['spread_chang
e'].notna()]

Data_RUN_A_merge_2['p_ret'] = Data_RUN_A_merge_2.groupby(['trd_exctn_dt'
,'cusip_id'])['Price'].pct_change()*100
Data_RUN_A_merge_2 = Data_RUN_A_merge_2[Data_RUN_A_merge_2['p_ret'].notn
a()]

Data_RUN_A_merge_2.to_csv('Data_RUN_A_merge_2.csv')
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:9: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [729]:
```
Data_RUN_A_merge_2
```

Out[729]:

| | Time | ISIN | Price | Size | Axe | Dealer | Side | PriceType | Vend |
|---|---|---|---|---|---|---|---|---|---|
| **2** | 2020-02-07T06:15:13Z | US92343VBR42 | 111.450 | 10000000.0 | A | JP MORGAN | Offer | Percentage | Dea Ru |
| **3** | 2020-02-07T08:34:24Z | US92343VBR42 | 111.510 | 10000000.0 | A | JP MORGAN | Offer | Percentage | Dea Ru |
| **4** | 2020-02-07T07:32:48Z | US92343VBR42 | 111.370 | 10000000.0 | A | JP MORGAN | Offer | Percentage | Dea Ru |
| **5** | 2020-02-07T08:15:16Z | US92343VBR42 | 111.470 | 10000000.0 | A | JP MORGAN | Offer | Percentage | Dea Ru |
| **6** | 2020-02-07T06:14:39Z | US92343VBR42 | 111.450 | 10000000.0 | A | JP MORGAN | Offer | Percentage | Dea Ru |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | |
| **26291** | 2020-03-27T21:05:12Z | US92343VBR42 | 108.735 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPI |
| **26292** | 2020-03-27T21:08:39Z | US92343VBR42 | 108.739 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPI |
| **26293** | 2020-03-27T21:10:40Z | US92343VBR42 | 108.753 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPI |
| **26294** | 2020-03-27T21:12:54Z | US92343VBR42 | 108.748 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPI |
| **26295** | 2020-03-27T21:13:15Z | US92343VBR42 | 108.721 | 1001000.0 | A | JP MORGAN | Bid | Percentage | JPI |

26078 rows × 29 columns

In [745]:
```
results_m = smf.ols('p_ret ~ yield_change + spread_change ', data = Data
_RUN_A_merge_2).fit()
print(results_m.summary())
```

```
                            OLS Regression Results
==============================================================================
=======
Dep. Variable:                    p_ret   R-squared:
1.000
Model:                            OLS   Adj. R-squared:
1.000
Method:                 Least Squares   F-statistic:                3.
586e+07
Date:                Thu, 10 Dec 2020   Prob (F-statistic):
0.00
Time:                        02:55:58   Log-Likelihood:             1.1
936e+05
No. Observations:               26078   AIC:                        -2.
387e+05
Df Residuals:                   26075   BIC:                        -2.
387e+05
Df Model:                           2
Covariance Type:            nonrobust
==============================================================================
==========
                   coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
----------
Intercept      9.632e-05   1.57e-05      6.143      0.000    6.56e-05
0.000
yield_change      0.0001   5.22e-05      2.235      0.025    1.43e-05
0.000
spread_change    -1.0722      0.000  -8469.196      0.000      -1.072
-1.072
==============================================================================
=======
Omnibus:                    75129.280   Durbin-Watson:
2.006
Prob(Omnibus):                  0.000   Jarque-Bera (JB):       3883645
284.029
Skew:                          39.509   Prob(JB):
0.00
Kurtosis:                    1891.898   Cond. No.
8.23
==============================================================================
=======

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is
correctly specified.
```

# Liquidity cost for the data Run

In [733]:
```
sql_query = """
SELECT *
FROM trace.trace_enhanced
WHERE CUSIP_ID in ('92343VBR4','110122DQ8','110122DR6','92343VFT6')
AND TRD_EXCTN_DT BETWEEN '2007-01-01' AND '2019-12-31';
"""
new_data_query1 = db.raw_sql(sql_query)
```

In [732]:
```
db = wrds.Connection()
```

```
Enter your WRDS username [deepsha]:madhur5
Enter your password:••••••••
WRDS recommends setting up a .pgpass file.
You can find more info here:
https://www.postgresql.org/docs/9.5/static/libpq-pgpass.html.
Loading library list...
Done
```

In [734]:
```
new_data_query1.to_csv('new_data_query1.csv')
```

In [736]:
```
def lagged_auto_cov(Xi,t):

    N = len(Xi)
    Xs = np.mean(Xi)
    end_padded_series = np.zeros(N+t)
    end_padded_series[:N] = Xi - Xs
    start_padded_series = np.zeros(N+t)
    start_padded_series[t:] = Xi - Xs

    auto_cov = 1./(N-1) * np.sum( start_padded_series*end_padded_series
)
    return auto_cov

#  Data Run
new_data_query = pd.read_csv('new_data_query1.csv')
new_data_query = new_data_query.rename(columns={"cusip_id": "COMPLETE_CU
SIP"})
new_data_query["trd_exctn_dt"] = pd.to_datetime(new_data_query["trd_exct
n_dt"], format="%Y-%m-%d")

# rolls spread calculation

uniques_cusips = new_data_query.COMPLETE_CUSIP.unique()
Ncs_rs_all = pd.DataFrame(columns = ['COMPLETE_CUSIP', 'date', 'auto_co
v','rolls_spread'])
for k in uniques_cusips:
    x = new_data_query[new_data_query.COMPLETE_CUSIP == k]
    auto_cov = []
    rolls_spread = []
    uniques_dates = x.trd_exctn_dt.unique()
    for i in range(0,len(uniques_dates)):
        p = x[x.trd_exctn_dt == uniques_dates[i]]['rptd_pr']
        d = np.diff(p)
        if len(d) == 0 or len(d) == 1 :
            a = np.nan
            r = np.nan
            auto_cov.append(np.nan)
            rolls_spread.append(np.nan)
        else:
            a = lagged_auto_cov(d,1)
            r = 2*np.sqrt(-a)
            auto_cov.append(a)
            rolls_spread.append(r)
        data = pd.DataFrame({'COMPLETE_CUSIP':k ,'date':uniques_dates[i
],'auto_cov': a,'rolls_spread' : r },index=[0])
        Ncs_rs_all = Ncs_rs_all.append(data)

#rename date
Ncs_rs_all = Ncs_rs_all.rename(columns={"date":"trd_exctn_dt"})



## Ncs_rs_all contains rolls  spread for all cusips for all dates. You c
an merge this with your predicted price data.

Ncs_rs_all.to_csv('new_data_query14regression.csv')
```

```
/Users/deepsha/anaconda3/lib/python3.7/site-packages/ipykernel_launche
r.py:37: RuntimeWarning:

invalid value encountered in sqrt
```

In [737]:
```
Ncs_rs_all
```

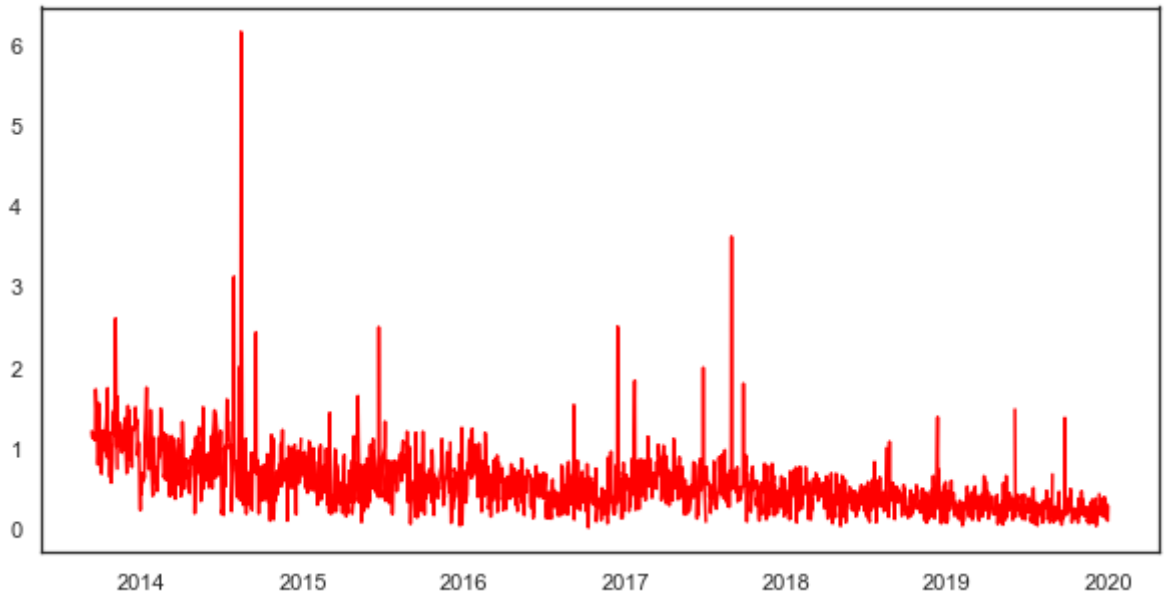
Out[737]:

| | COMPLETE_CUSIP | trd_exctn_dt | auto_cov | rolls_spread |
|---|---|---|---|---|
| **0** | 92343VBR4 | 2013-09-11 | -0.359518 | 1.199196 |
| **0** | 92343VBR4 | 2013-09-12 | -0.317982 | 1.127799 |
| **0** | 92343VBR4 | 2013-09-13 | -0.309822 | 1.113234 |
| **0** | 92343VBR4 | 2013-09-16 | -0.330675 | 1.150087 |
| **0** | 92343VBR4 | 2013-09-17 | -0.339058 | 1.164573 |
| **...** | ... | ... | ... | ... |
| **0** | 92343VBR4 | 2019-12-24 | -0.004041 | 0.127141 |
| **0** | 92343VBR4 | 2019-12-26 | -0.023100 | 0.303974 |
| **0** | 92343VBR4 | 2019-12-27 | -0.012442 | 0.223092 |
| **0** | 92343VBR4 | 2019-12-30 | -0.001810 | 0.085078 |
| **0** | 92343VBR4 | 2019-12-31 | -0.018010 | 0.268402 |

1587 rows × 4 columns

In [744]:
```
fig, ax = plt.subplots(figsize = (10,5))
sns.set_style('white')
ax.plot(Ncs_rs_all['trd_exctn_dt'], Ncs_rs_all['rolls_spread'], color =
"red", label = 'rolls_spread')
```

Out[744]:
```
[<matplotlib.lines.Line2D at 0x14a167c88>]
```



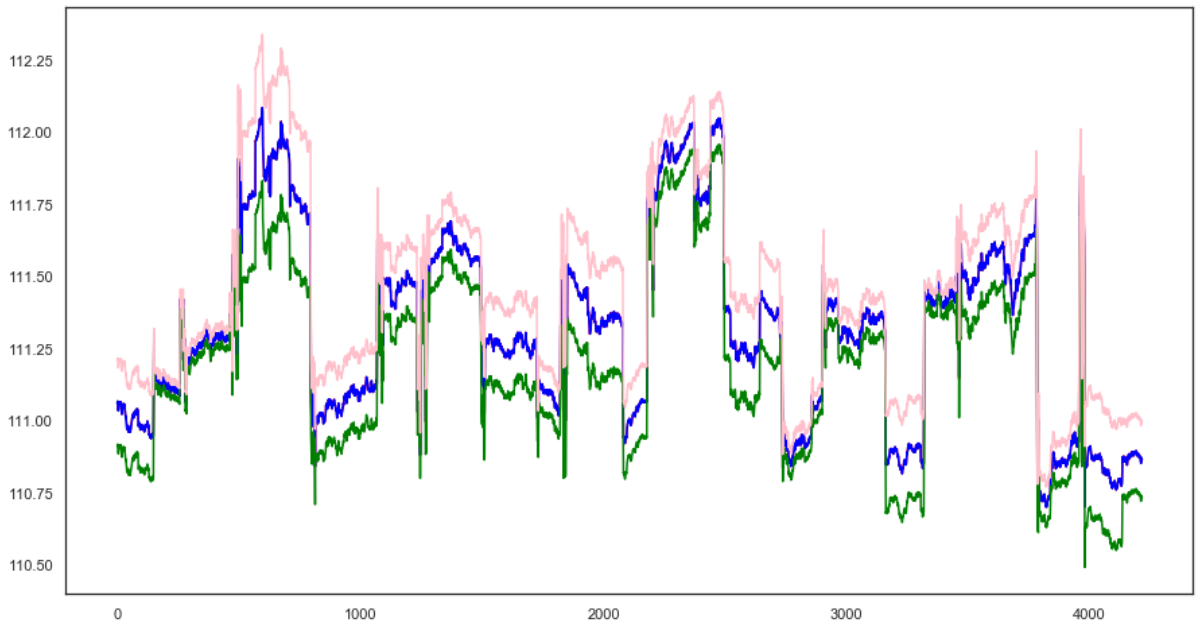
In [747]:
```
Data_RUN_price = pd.read_csv('Data_RUN_A_merge_2.csv')
Data_RUN_price = Data_RUN_price.iloc[:, 1:]

Data_RUN_price['deltaP'] = (9.632e-05 + 0.0001*Data_RUN_price['yield_cha
nge']-1.0722*Data_RUN_price['spread_change'])
Data_RUN_price['prevP'] = Data_RUN_price.groupby(['trd_exctn_dt','cusip_
id'])['Price'].shift(1)
Data_RUN_price['predictedP'] = Data_RUN_price['prevP'] * (1 + (0.01 * Da
ta_RUN_price['deltaP']))

Data_RUN_pred_Price  = Data_RUN_price[['cusip_id', 'trd_exctn_dt', 'Pric
e', 'predictedP']]

new_data_query14regression = pd.read_csv('new_data_query14regression.cs
v')
new_data_query14regression = new_data_query14regression.iloc[:, 1:]

new_data_rollSpread = new_data_query14regression[['COMPLETE_CUSIP', 'trd
_exctn_dt', 'rolls_spread']]
#wmt_rollSpread["trd_exctn_dt"] = pd.to_datetime(wmt_rollSpread["trd_exc
tn_dt"], format="%Y%m%d")
new_data_rollSpread.trd_exctn_dt = new_data_rollSpread.trd_exctn_dt.asty
pe(str)
#print(wmt_rollSpread.head())
Data_RUN_band = Data_RUN_pred_Price.merge(new_data_rollSpread, left_on=[
'cusip_id', 'trd_exctn_dt'], right_on=['COMPLETE_CUSIP', 'trd_exctn_dt'
])

Data_RUN_band['price_lower_bound'] = Data_RUN_band['predictedP'] - 0.5*(
Data_RUN_band['rolls_spread']*Data_RUN_band['predictedP'] /100)
Data_RUN_band['price_upper_bound'] = Data_RUN_band['predictedP'] + 0.5*(
Data_RUN_band['rolls_spread']*Data_RUN_band['predictedP'] /100)

good_trade = Data_RUN_band.apply(lambda x: 0 if (x['Price'] >= x['price_
lower_bound'] and x['Price'] <= x['price_upper_bound']) else 1, axis=1)

print(sum(good_trade)/len(good_trade))
```

```
0.010431484115694643

/Users/deepsha/anaconda3/lib/python3.7/site-packages/pandas/core/generi
c.py:5165: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

In [755]:
```
Data_RUN_band_ana = Data_RUN_band.dropna()
```

In [756]:
```
fig, ax = plt.subplots(figsize = (15,8))
sns.set_style('white')
ax.plot( Data_RUN_band_ana['predictedP'], color = "red", label = 'rolls_
spread')
ax.plot( Data_RUN_band_ana['Price'], color = "blue", label = 'rolls_spre
ad')
ax.plot( Data_RUN_band_ana['price_lower_bound'], color = "green", label
= 'rolls_spread')
ax.plot( Data_RUN_band_ana['price_upper_bound'], color = "pink", label =
'rolls_spread')
```

Out[756]:
```
[<matplotlib.lines.Line2D at 0x16de40748>]
```




# ADF testing for Mean Reversion

In [757]:
```
import statsmodels.tsa.stattools as ts
```

In [759]:
```
ts.adfuller(Data_RUN_band_ana['predictedP'])
```

Out[759]:
```
(-3.269212196423111,
 0.0163120474923516,
 25,
 4148,
 {'1%': -3.431927471404566,
  '5%': -2.862237040274829,
  '10%': -2.567141040789254},
 -12958.644444157177)
```

### Since the Test statistic value is smaller than the 5 and 10 levels, it is likely to reject the null hypothesis and the series is mean reverting

In [766]:
```
adf_statistic = ts.adfuller(Data_RUN_band_ana['predictedP'], 1)
adf_statistic
```

Out[766]:
```
(-4.357451443554643,
 0.00035275248743890143,
 1,
 4172,
 {'1%': -3.431918391192134,
  '5%': -2.8622330290415605,
  '10%': -2.5671389053926603},
 -13000.908382996844)
```

In [771]:
```
import os
import sys
import warnings
from datetime import date
import pandas as pd
import pandas_datareader.data as web
import numpy as np
from numpy.linalg import LinAlgError

import statsmodels.tsa.api as tsa
from statsmodels.graphics.tsaplots import plot_acf, acf, plot_pacf, pacf
from statsmodels.tsa.stattools import acf, q_stat, adfuller
import statsmodels.api as sm
from scipy.stats import probplot, moment
from sklearn.metrics import mean_squared_error

import quandl
import matplotlib.pyplot as plt
import matplotlib as mpl
```

In [772]:
```
%matplotlib inline
warnings.filterwarnings('ignore')
plt.style.use('ggplot')
```

In [773]:
```
def plot_correlogram(x, lags=None, title=None):
    lags = min(10, int(len(x)/5)) if lags is None else lags
    fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(14, 8))
    x.plot(ax=axes[0][0])
    q_p = np.max(q_stat(acf(x, nlags=lags), len(x))[1])
    stats = f'Q-Stat: {np.max(q_p):>8.2f}\nADF: {adfuller(x)[1]:>11.2f}'
    axes[0][0].text(x=.02, y=.85, s=stats, transform=axes[0][0].transAxe
s)
    probplot(x, plot=axes[0][1])
    mean, var, skew, kurtosis = moment(x, moment=[1, 2, 3, 4])
    s = f'Mean: {mean:>12.2f}\nSD: {np.sqrt(var):>16.2f}\nSkew: {skew:1
2.2f}\nKurtosis:{kurtosis:9.2f}'
    axes[0][1].text(x=.02, y=.75, s=s, transform=axes[0][1].transAxes)
    plot_acf(x=x, lags=lags, zero=False, ax=axes[1][0])
    plot_pacf(x, lags=lags, zero=False, ax=axes[1][1])
    axes[1][0].set_xlabel('Lag')
    axes[1][1].set_xlabel('Lag')
    fig.suptitle(title, fontsize=20)
    fig.tight_layout()
    fig.subplots_adjust(top=.9)
```

In [774]:
```
ts_p = Data_RUN_band_ana['predictedP']
```

In [776]:
```
(ts_p == 0).any()
```

Out[776]: False

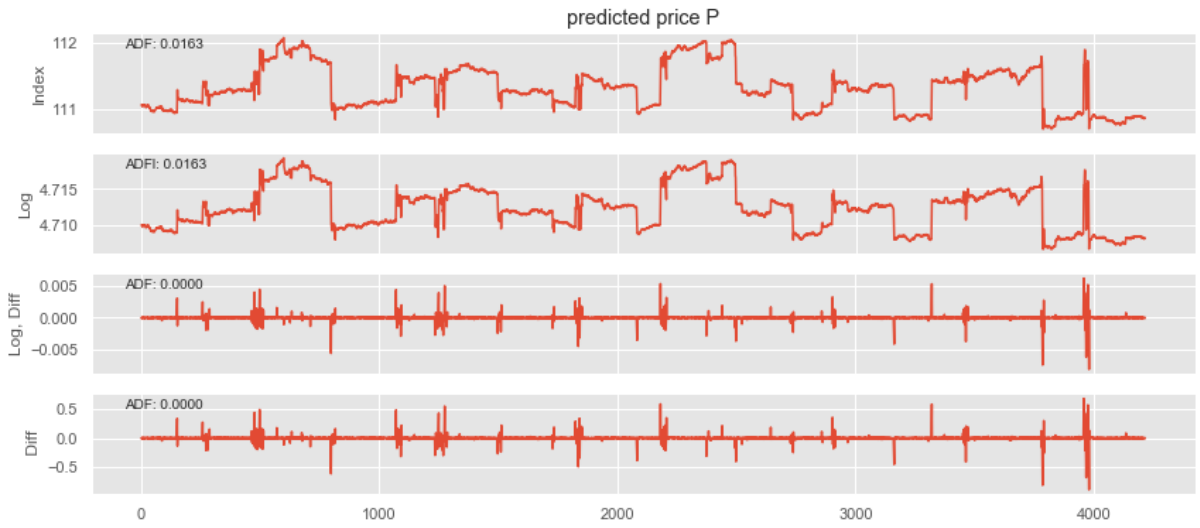
In [783]:
```
ts_p_diff = ts_p.diff().dropna()
```

In [777]:
```
ts_p_log = np.log(ts_p)
```

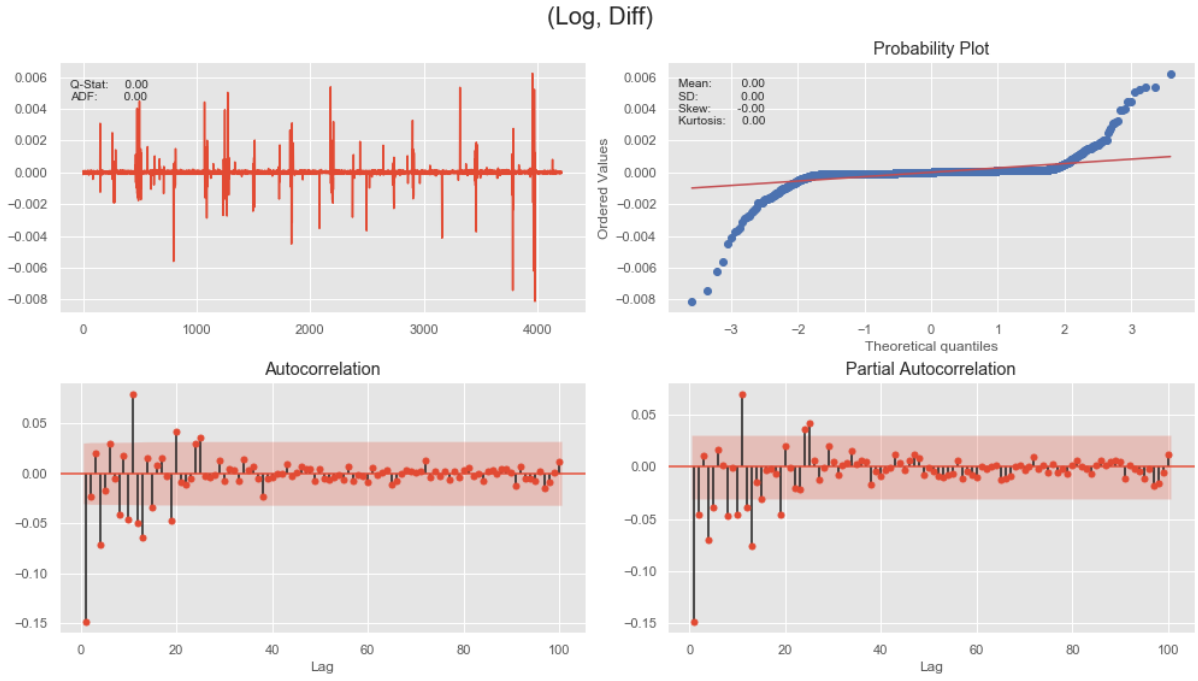
In [778]:
```
ts_p_log_diff = ts_p_log.diff().dropna()
```

In [784]:
```
fig, axes = plt.subplots(nrows=4, ncols=1, figsize=(14,6))

ts_p.plot(ax=axes[0], title='predicted price P')
axes[0].text(x=.03, y=.85, s=f'ADF: {tsa.adfuller(ts_p.dropna())[1]:.4
f}', transform=axes[0].transAxes)
axes[0].set_ylabel('Index')

ts_p_log.plot(ax=axes[1], sharex=axes[0])
axes[1].text(x=.03, y=.85, s=f'ADFl: {tsa.adfuller(ts_p_log.dropna())
[1]:.4f}', transform=axes[1].transAxes)
axes[1].set_ylabel('Log')

ts_p_log_diff.plot(ax=axes[2], sharex=axes[0])
axes[2].text(x=.03, y=.85, s=f'ADF: {tsa.adfuller(ts_p_log_diff.dropna
())[1]:.4f}', transform=axes[2].transAxes)
axes[2].set_ylabel('Log, Diff')

ts_p_diff.plot(ax=axes[3], sharex=axes[0])
axes[3].text(x=.03, y=.85, s=f'ADF: {tsa.adfuller(ts_p_diff.dropna())
[1]:.4f}', transform=axes[3].transAxes)
axes[3].set_ylabel(' Diff')
```

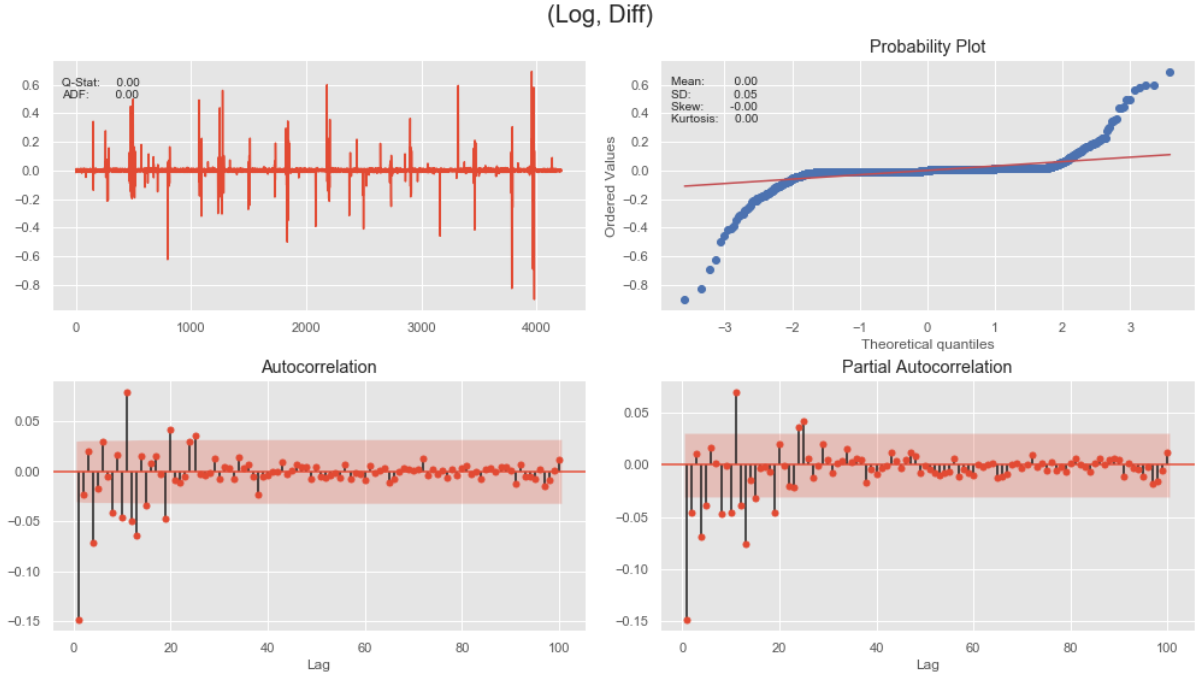
Out[784]: Text(0, 0.5, ' Diff')



In [782]:
```
plot_correlogram(ts_p_log_diff, lags=100, title='(Log, Diff)')
```




In [785]:
```
plot_correlogram(ts_p_diff, lags=100, title='(Log, Diff)')
```




In [ ]:
```

```