U.S. Stock Market IV: Interest & Debt

In [1]:
```
import pandas as pd
import numpy as np
import requests, zipfile, io
import os
from pathlib import Path

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

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

In [2]:
```
def get_items_from_SEC_files(tags, filename=None):                  # Function inp

    directory = 'data/sec/merged/'                                  # Read data fr
    filenames = [filename] if filename else os.listdir(directory) # Supplied fil
    filenames = [f for f in filenames if not f.startswith(".")]   # Exclude hidd

    results   = {t:pd.DataFrame() for t in tags}                    # Dictionary o

    for filename in filenames:                                      # Loop over al
        print(filename)
        data = pd.read_csv(directory+filename, parse_dates=['filed','ddate'])  #

        for t in tags:                                              # Loop over al
            item  = data[data.tag==t]                               # Select all d
            short = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[
            long  = item.sort_values(['cik','filed','ddate','qtrs'], ascending=[
            short = short.groupby(['cik','filed']).last()[['value','qtrs']]
            long  = long .groupby(['cik','filed']).last()[['value','qtrs']]
            short_long = short.join(long, lsuffix='_shortest', rsuffix='_longest
            results[t] = results[t].append( short_long )

    for t in tags:                                                  # Now sort all
        if not results[t].empty: results[t] = results[t].sort_index(level='filed

    return results



def combine_items(tags, items):
    result = items[tags[0]]
    for tag in tags[1:]:  result = result.combine_first( items[tag] )
    return result



def calculate_quarterly_annual_values(item):                        # item: tabl
    result           = pd.DataFrame()                               # Results go
    all_firms        = item.index.get_level_values('cik').unique()  # All CIKs.
    all_filing_dates = pd.read_csv('data/sec/dates/filing_dates.csv', index_col=

    for cik in all_firms:                                           # Loop over
        filing_dates = pd.Series(all_filing_dates.filed[cik])       # All filing
```

```
        # Quarterly values:
        valuesQ = item.loc[cik].value_shortest.reindex(filing_dates) # Values wi
        qtrsQ   = item.loc[cik].qtrs_shortest.astype(int)           # Number of
        for date,q in qtrsQ[qtrsQ>1].iteritems():                   # Loop over
            previous_values = valuesQ[:date][-q:-1]                 # Example: f
            if len(previous_values) == q-1:                         # If all pre
                valuesQ[date] -= previous_values.sum(skipna=False)  # Subtract p
            else:
                valuesQ[date]  = np.nan

        # Annual values:
        valuesA = item.loc[cik].value_longest.reindex(filing_dates) # Values wit
        qtrsA   = item.loc[cik].qtrs_longest.astype(int)            # Number of
        for date,q in qtrsA[qtrsA<4].iteritems():                   # Loop over
            previous_values = valuesQ[:date][-4:-q]                 # Example: f
            if len(previous_values) == 4-q:                         # If all pre
                valuesA[date] += previous_values.sum(skipna=False)  # Add previo
            else:
                valuesA[date]  = np.nan

        result = result.append( pd.DataFrame({'cik':cik, 'filed':filing_dates, '

    return result.set_index(['cik','filed'])                        # Return a t



def ffill_values(item, dates):
    data = item.unstack('cik')
    data = data.reindex(dates.union(data.index)).sort_index()           # Add sp
    filing_dates = pd.read_csv('data/sec/dates/filing_dates.csv', index_col='cik
    last_filing_date_all_firms = filing_dates.max()                     # Most r

    for cik in data.columns:                                            # Loop o
        last_filing_date      = pd.Series(filing_dates[cik]).iloc[-1]   # Last d
        days_since_last_filed = (last_filing_date_all_firms - last_filing_date).
        last_date_this_firm   = dates[-1] if days_since_last_filed < 120 else la
        data.loc[:last_date_this_firm, cik].ffill(inplace=True)         # Forwar

    return data.loc[dates]                                              # Return
```

Get these tags:

In [3]:

```
tags_shortTermDebt              = ['ShortTermBorrowing','DebtCurrent']
tags_longTermDebtCurrent        = ['LongTermDebtAndCapitalLeaseObligationsCurre
tags_longTermDebtNoncurrent     = ['LongTermDebtAndCapitalLeaseObligations', 'L
tags_interest_expense           = ['InterestExpenseDebt','InterestAndDebtExpens
tags_interest_income            = ['InvestmentIncomeInterest','InterestAndOther

all_tags = tags_shortTermDebt + tags_longTermDebtCurrent + tags_longTermDebtNonc

items = get_items_from_SEC_files( all_tags )
```

```
2018q4.csv
2018q3.csv
2018q2.csv
2021_01.csv
2018q1.csv
2020q2.csv
2020q3.csv
```

```
2020q1.csv
2019q4.csv
2019q1.csv
2019q3.csv
2019q2.csv
2013q4.csv
2015q2.csv
2015q3.csv
2017q1.csv
2020_12.csv
2020_10.csv
2017q3.csv
2015q1.csv
2017q2.csv
2011q4.csv
2020_11.csv
2013q2.csv
2015q4.csv
2011q1.csv
2013q3.csv
2013q1.csv
2011q3.csv
2017q4.csv
2011q2.csv
2009q4.csv
2014q1.csv
2016q3.csv
2010q4.csv
2016q2.csv
2014q2.csv
2012q4.csv
2016q1.csv
2014q3.csv
2009q2.csv
2010q3.csv
2012q1.csv
2010q2.csv
2016q4.csv
2009q3.csv
2009q1.csv
2014q4.csv
2012q2.csv
2012q3.csv
2010q1.csv
```

In [4]:

```
items['interest_expense'] = combine_items(tags_interest_expense,  items)
items['interest_income']  = combine_items(tags_interest_income,   items)

interest_expense = calculate_quarterly_annual_values(items['interest_expense'])
interest_income  = calculate_quarterly_annual_values(items['interest_income'])

interest_expense[:3]
```

Out[4]:

		valueQ	valueA
cik	**filed**		
1750	**2010-09-23**	7431000.0	NaN
	2010-12-21	7579000.0	NaN
	2011-03-22	7595000.0	NaN

In [5]:
```
# We don't need to calculate quarterly and annual values for debt, since debt is
shortTermDebt           = combine_items(tags_shortTermDebt,           items)
longTermDebtCurrent     = combine_items(tags_longTermDebtCurrent,     items)
longTermDebtNoncurrent = combine_items(tags_longTermDebtNoncurrent, items)

shortTermDebt[:3]
```

Out[5]:

		value_shortest	qtrs_shortest	value_longest	qtrs_longest
cik	**filed**				
1750	**2012-09-25**	108200000.0	0	108200000.0	0
	2012-12-21	106600000.0	0	106600000.0	0
	2013-03-22	22200000.0	0	22200000.0	0

In [6]:
```
# Save files
interest_income         .to_csv('data/sec/items/InterestIncome.csv')
interest_expense        .to_csv('data/sec/items/InterestExpense.csv')
shortTermDebt           .to_csv('data/sec/items/ShortTermDebt.csv')
longTermDebtCurrent     .to_csv('data/sec/items/LongTermDebtCurrent.csv')
longTermDebtNoncurrent .to_csv('data/sec/items/LongTermDebtNoncurrent.csv')
```

In [7]:
```
# Read files we just saved (units: billion dollars)
interest_income         = pd.read_csv('data/sec/items/InterestIncome.csv',
interest_expense        = pd.read_csv('data/sec/items/InterestExpense.csv',
shortTermDebt           = pd.read_csv('data/sec/items/ShortTermDebt.csv',
longTermDebtCurrent     = pd.read_csv('data/sec/items/LongTermDebtCurrent.csv',
longTermDebtNoncurrent = pd.read_csv('data/sec/items/LongTermDebtNoncurrent.csv'

# Also get operating income
operatingIncome         = pd.read_csv('data/sec/items/OperatingIncome.csv',
operatingIncome[:2]
```

Out[7]:

		valueQ	valueA
filed	**cik**		
2009-04-15	**277948**	0.522	NaN
2009-07-15	**277948**	0.582	NaN

In [8]:
```
# Read our sic codes from last notebook:
sic = pd.read_csv('data/sec/attributes/sic.csv', parse_dates=['filed'], index_co
sic[:2]
```

Out[8]:

		sic
filed	**cik**	
2009-04-15	**277948**	4011.0
2009-04-23	**883984**	3841.0

Fill the tables:

In [9]:
```
trading_days = pd.to_datetime( tiingo.get_dataframe('SPY','2009-04-15').index ).

interestExpenseQ = ffill_values( interest_expense.valueQ, trading_days )
interestExpenseA = ffill_values( interest_expense.valueA, trading_days )

interestIncomeQ = ffill_values( interest_income.valueQ, trading_days )
interestIncomeA = ffill_values( interest_income.valueA, trading_days )

operatingIncomeQ = ffill_values( operatingIncome.valueQ, trading_days )
operatingIncomeA = ffill_values( operatingIncome.valueA, trading_days )

shortTermDebt          = ffill_values( shortTermDebt.value_shortest,          tr
longTermDebtCurrent    = ffill_values( longTermDebtCurrent.value_shortest,    tr
longTermDebtNoncurrent = ffill_values( longTermDebtNoncurrent.value_shortest, tr

sic = ffill_values(sic.sic, trading_days)
```

Calculate total debt:

> debt = shortTermDebt + longTermDebtCurrent + longTermDebtNoncurrent

In [10]:
```
# fill_value=0: if value missing (for example firm does not report shortTermDebt
debt = shortTermDebt.add(longTermDebtCurrent, fill_value=0).add(longTermDebtNonc

debt[-2:]
```

Out[10]:

cik	1750	1800	1961	2034	2098	2488	2491	2969	3116	3197	...	18082
date												
2021-03-08	0.2274	18.355	0.00145	NaN	0.030703	0.33	NaN	7.6029	NaN	0.076755	...	0.4009
2021-03-09	0.2274	18.355	0.00145	NaN	0.030703	0.33	NaN	7.6029	NaN	0.076755	...	0.4009

2 rows × 5290 columns

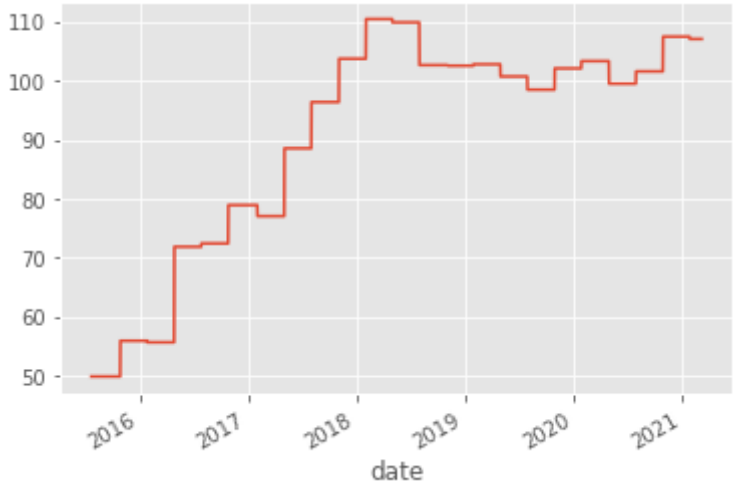
Historical debt for specific firm:

In [11]:
```
symbols = pd.read_csv('data/ticker_symbols/symbols.csv',index_col=0)
```

In [12]:
```
cik = symbols[symbols.ticker=='AAPL'].index[0]

debt[cik].plot()
```

Out[12]:
```
<AxesSubplot:xlabel='date'>
```



Top 10 annual interest expense:

In [13]:
```
sic_current = sic.iloc[-1].to_frame('sic')  # Last row in sic table = most recen
sic_current[:2]
```

Out[13]:

cik	sic
1750	3720.0
1800	2834.0

In [14]:
```
interestExpenseA.iloc[-1].nlargest(10).to_frame('Debt').join(symbols.title).join
```

Out[14]:

cik	Debt	title	sic
310522	88.474	FEDERAL NATIONAL MORTGAGE ASSOCIATION FANNIE MAE	6111.0
1026214	53.588	FEDERAL HOME LOAN MORTGAGE CORP	6111.0
831001	18.525	CITIGROUP INC	6021.0
19617	13.924	JPMORGAN CHASE & CO	6021.0
40545	12.508	GENERAL ELECTRIC CO	3600.0
70858	11.803	BANK OF AMERICA CORP /DE/	6021.0
886982	11.232	GOLDMAN SACHS GROUP INC	6211.0
72971	11.163	WELLS FARGO & COMPANY/MN	6021.0
732717	8.080	AT&T INC.	4813.0
895421	5.994	MORGAN STANLEY	6211.0

Note how the firms with high interest expense are mostly banks.
We cannot directly compare interest for financial and non-financial firms since paying/receiving interest is part of the business of a bank, but not part of the operations of a "regular" company.
Let's exclude financial firms (https://www.osha.gov/data/sic-manual):

In [15]:
```
# Get all 6000s  ("Finance, Insurance, And Real Estate")
codes = sic.div(1000).apply(np.floor)  # Divide by 1000 to get 1st digit
codes[-3:]
```

Out[15]:

cik	1750	1800	1961	2034	2098	2178	2186	2488	2491	2969	...	1824013	1824301
date													
2021-03-05	3.0	2.0	7.0	NaN	3.0	5.0	3.0	3.0	NaN	2.0	...	6.0	6.0
2021-03-08	3.0	2.0	7.0	NaN	3.0	5.0	3.0	3.0	NaN	2.0	...	6.0	6.0
2021-03-09	3.0	2.0	7.0	NaN	3.0	5.0	3.0	3.0	NaN	2.0	...	6.0	6.0

3 rows × 12126 columns

In [16]:
```
financials = codes[codes==6].notnull()  # Select 1st digit == 6
financials[-3:]
```

Out[16]:

cik	1750	1800	1961	2034	2098	2178	2186	2488	2491	2969	...	1824013	1824301
date													
2021-03-05	False	False	False	False	False	False	False	False	False	False	...	True	True
2021-03-08	False	False	False	False	False	False	False	False	False	False	...	True	True
2021-03-09	False	False	False	False	False	False	False	False	False	False	...	True	True

3 rows × 12126 columns

Top 10 financial firms annual interest expense:

In [17]:
```
interestExpenseA[financials].iloc[-1].nlargest(10).to_frame('Debt').join(symbols
```

Out[17]:

cik	Debt	title	sic
310522	88.474	FEDERAL NATIONAL MORTGAGE ASSOCIATION FANNIE MAE	6111.0
1026214	53.588	FEDERAL HOME LOAN MORTGAGE CORP	6111.0
831001	18.525	CITIGROUP INC	6021.0
19617	13.924	JPMORGAN CHASE & CO	6021.0

cik	Debt	title	sic
70858	11.803	BANK OF AMERICA CORP /DE/	6021.0
886982	11.232	GOLDMAN SACHS GROUP INC	6211.0
72971	11.163	WELLS FARGO & COMPANY/MN	6021.0
895421	5.994	MORGAN STANLEY	6211.0
1067983	3.961	BERKSHIRE HATHAWAY INC	6331.0
927628	3.806	CAPITAL ONE FINANCIAL CORP	6021.0

Top 10 non-financial firms annual interest expense:

In [18]:
```
interestExpenseA[~financials].iloc[-1].nlargest(10).to_frame('Debt').join(symbol
```

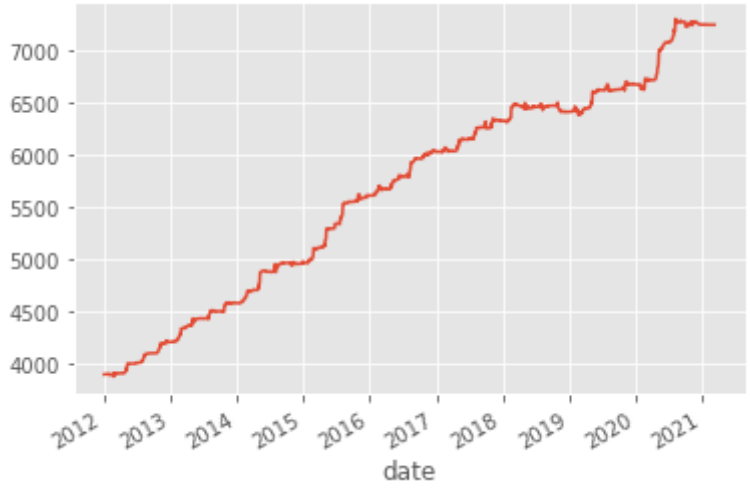
Out[18]:

cik	Debt	title	sic
40545	12.508	GENERAL ELECTRIC CO	3600.0
732717	8.080	AT&T INC.	4813.0
1166691	4.657	COMCAST CORP	4841.0
1467858	4.423	General Motors Co	3711.0
732712	4.326	VERIZON COMMUNICATIONS INC	4813.0
64803	2.963	CVS HEALTH Corp	5912.0
895728	2.802	ENBRIDGE INC	4610.0
320193	2.726	Apple Inc.	3571.0
1571996	2.675	Dell Technologies Inc.	3571.0
789019	2.460	MICROSOFT CORP	7372.0

Total market debt (non-financials):

In [19]:
```
debt[~financials].sum('columns')['2012':].plot()
```

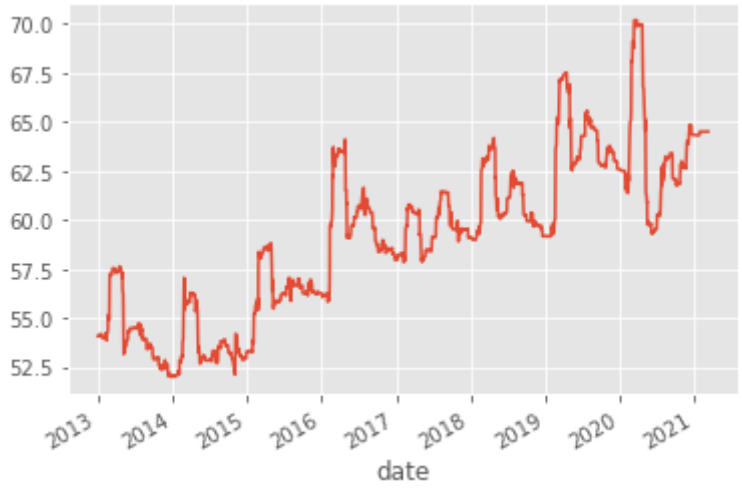
Out[19]:
```
<AxesSubplot:xlabel='date'>
```



Total quarterly interest expense (non-financials):

In [20]:
```
interestExpenseQ[~financials].sum('columns')['2013':].plot()
```

Out[20]: <AxesSubplot:xlabel='date'>

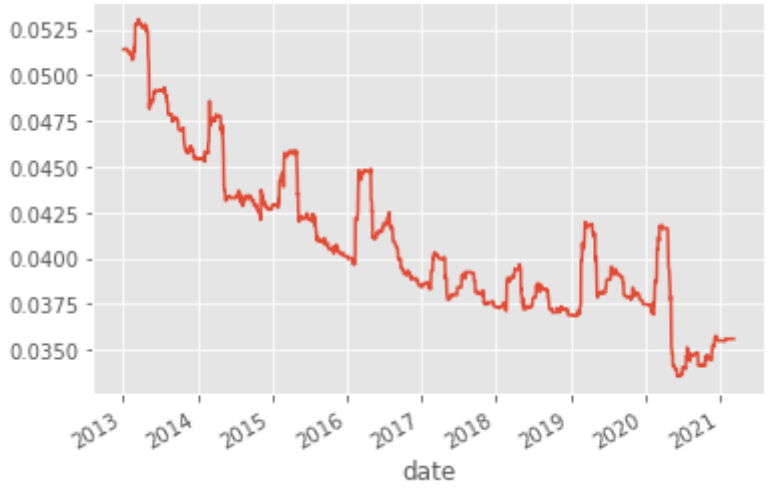


Interest expense relative to debt:

In [21]:
```
total_interestExpsense = interestExpenseQ[~financials].sum('columns')
total_debt             = debt             [~financials].sum('columns')

(total_interestExpsense*4/total_debt)['2013':].plot()   # Multiply quarterly val
```

Out[21]: <AxesSubplot:xlabel='date'>

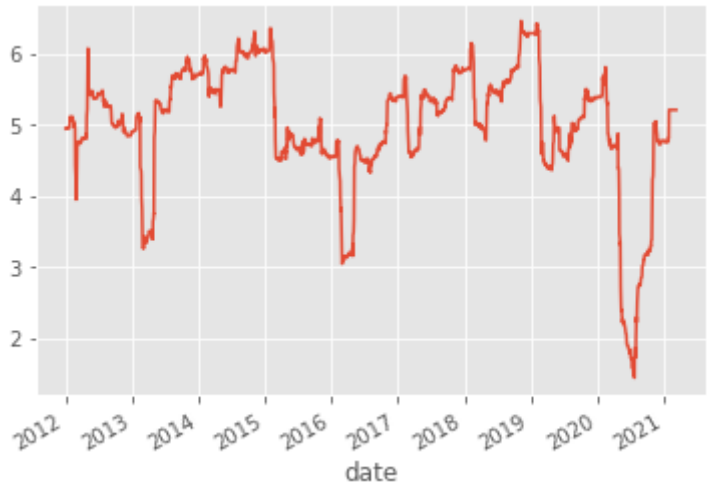


Operating income relative to interest expense:

In [22]:
```
total_operatingIncome = operatingIncomeQ[~financials].sum('columns')

(total_operatingIncome/total_interestExpsense)['2012':].plot()
```

Out[22]:
```
<AxesSubplot:xlabel='date'>
```



This is a type of "interest coverage ratio" (often calculated as EBIT/interest expense). Note how this ratio is somewhat stable over time even though total debt went up. It appears that firms on average target a specific caoverage ratio (about 5 in last 10 years) and they have been issuing more debt to maintain this ratio as earnings went up and interest ratios went down.