Capstone Project:
Special Topics in Risk Management:
Markov Theory & Applications

Single and Multi-Notch Credit Rating Analysis Using Julia

Michael Po-Hsuan Lin
Master of Science in Financial and Risk Engineering
N13780093

Overview of the course

Markov theory and application is a special topic course offered to financial engineering master student in 2018 fall semester. This course was taught by Professor Kevin Atteson who is an excellent instructor and a seasoned practitioner. The course can be seen as roughly structured into three portions with the first few lectures establishing the foundation for the rest of the course. Discussed topic ranges from the basics of Markov chain to the more widely used Markov Chain Monte Carlo.

The first few lectures were of great importance as they introduced the properties and graphical representations of Markov chain and how these concepts are used in the industry. The third lecture: Fitting Markov chain and the fourth lecture: Markov chain in Credit Modeling [1] are closely related to the project presented in this paper. Therefore, this material from these two lectures will be discussed in detail below.

These two lectures introduced students to the idea that Markov chain is commonly used to model credit movements. Since default can be seen as an absorbing state in a simple model setup alongside the memoryless property, Markov chain is commonly used to model credit ratings and credit movements. A simplified credit rating Markov chain can be seen below.

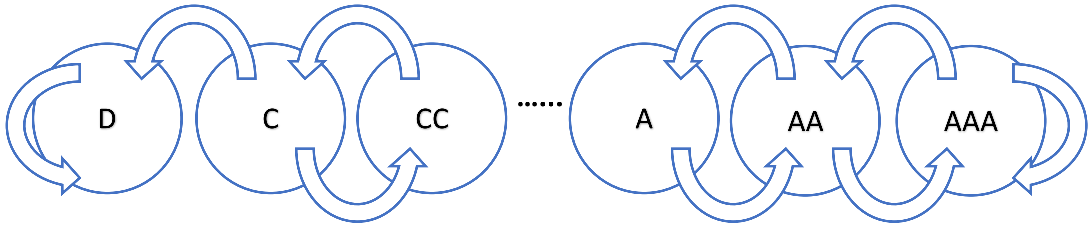


Figure 1, Simplified Credit Rating Markov Chain

The Markov chain above is a simplified version of the credit rating movement model used in the project. The states above are labelled using Standard & Poor credit rating system [2]. D stands for default and as the rating gets closer to AAA, its credit quality improves. It is structured in a similar fashion to the Birth-Death process [3] with default symbolizing death. However, what differentiates the credit rating model from the Birth-Death process is its birth state. Instead of it being another absorbing state, state AAA can't be modelled as an absorbing state. A company that has AAA rating is not guaranteed to have AAA for its remaining life.

As mentioned earlier, the above model is a simplified version. One should be aware of simplifications and assumptions associated with the model. First, the model assumes that default is an absorbing state. However, in reality, when faced with bankruptcy, companies usually get the choices of Chapter 7 liquidation and Chapter 11 restructuring. Under Chapter 7 liquidation, companies are in the absorbing default state. All of its assets are sold off and liabilities resolved. However, this is not the common choice. Chapter 11 restructuring allows for more flexible and value preservation. Under Chapter 11 restructuring, companies are allowed to restructure their debts and move forward with a better capital structure. It preserves value because companies won't cease to exist and its assets won't be liquidated. This situation also implies that default state is not an absorbing state. In fact, it will just be another state that is fully connected to all the other states. Secondly, the model is focused on one step transitions. Each state is only connected to its neighbors but not its neighbors' neighbors. This implies that companies like McDonalds can only transitioned from AA to AA- but not from AA to BBB. Although single step transitions are common, modeling only using single step transitions will miss out on certain phenomenon that is crucial to credit modeling. During recessions, companies tend to experience multistep transitions. For example, during the 2008 financial crisis, many highly sophisticated collateralized debt obligations or CDOs suffer a severe multistep transition downgrades as its underlying collaterals default. The deteriorating credit quality is one of the reasons that exacerbated the financial crisis. Therefore, it is important to also consider multi-notch transitions when building a realistic credit rating Markov model.

As demonstrated in class, once a model is built like the one shown above, extensive calculation can be done to fit the model and results such as transition probability matrix can be calculated. Similar model fitting and calculation process are done for the following project. This section will be concluded with a diagram of probability of default drawn by Professor Atteson [4].

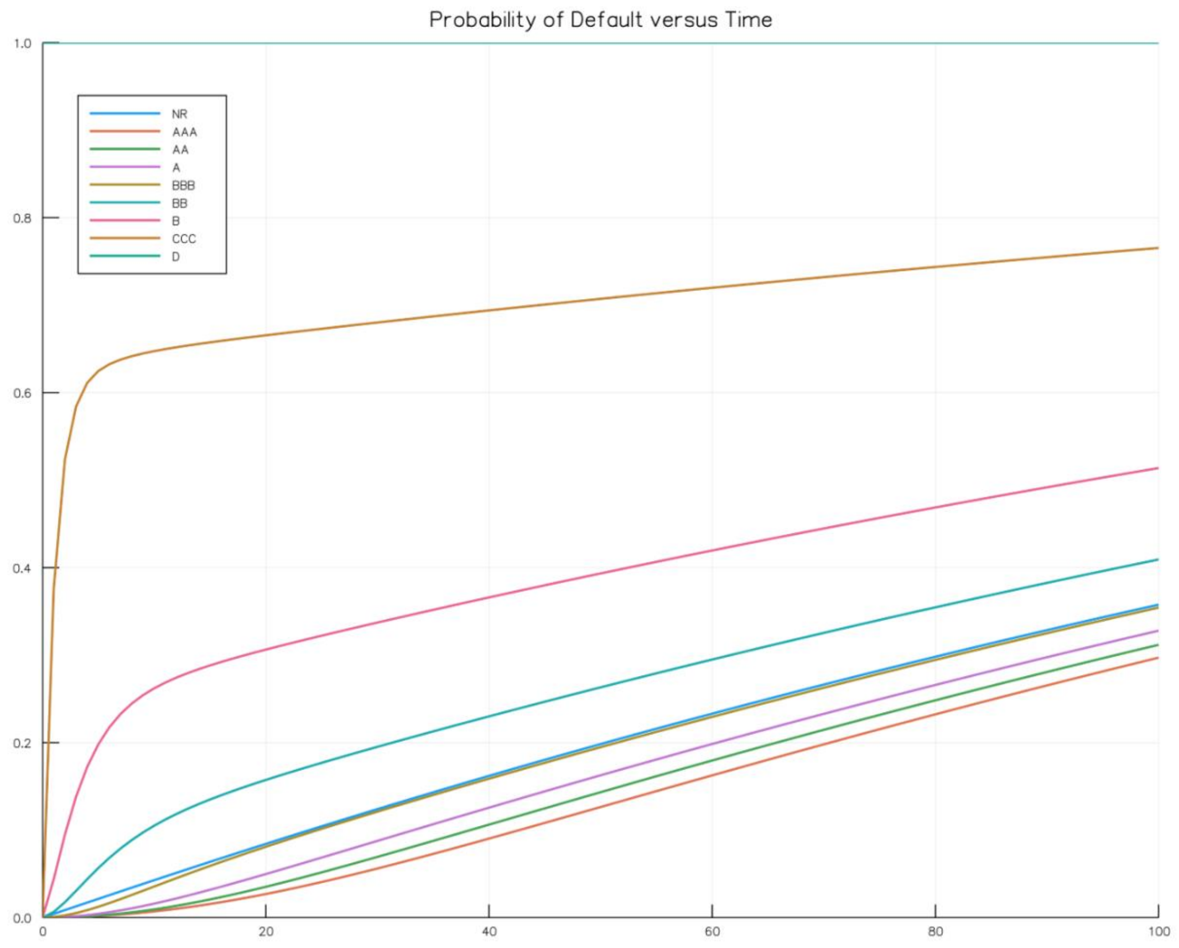


Figure 2, Probability of default versus time

Introduction

The following project and the paper that comes with it is an extension of Professor Atteson's Markov Theory and Application course. In this brief paper, I will be exploring single and multi-notch credit rating transitions using real life data. By exploring the more realistic multi-notch transitions, I was able to confirm two hypotheses: first, multi-notch transitions account for a small portion of all transitions. Second, multi-notch transitions cluster around recession periods with significant spikes in percentage. Aside from confirming these two hypotheses, transition probability matrices across time periods are calculated. The whole data cleaning and analysis is done using **Julia 1.0.3**.

Assumptions of the analysis

The paper will focus on a shorter time period, starting in January 1997 to February 2017. The data is taken from **Wharton Research Data Services**. I want to take a closer look and test out some time-dependent hypotheses in a relatively shorter time period that includes different economic cycles. During this twenty-one years of data, there are two significant economic downturns: the early 2000 internet bubble and the 2008 financial crisis. The assumptions made during the following analysis is slightly different from Professor's analysis during lecture.

First of all, as stated earlier, due to shorter time period, there are 2387466 data entries. After dropping missing data, the data set only contains 4981 companies which is significantly less than the amount of company analyzed in a longer period.

Secondly, data censoring in terms of length is used. Companies that don't have the complete data between 1997 and 2017 are dropped. It makes the analysis and transition matrix calculation easier and more consistent. Normally, this type of data length censoring allows for better accuracy and less noise. However, in this case, the data is skewed to the right. In other words, companies that have problems during this time period and end up in bankruptcy won't be reflected and at the same time, new companies created during this time period won't be included. Therefore, the remaining 529 companies after censoring are companies that are relatively stable and with moderate risk profiles. This bias is reflected in the percentage of multi-notch transitions which will be explained further in the following section. The data set also doesn't contain "not mentioned" (N.M) and "C" rating; as a result, these two ratings are not assigned a specific number in the coding scheme. Because of this difference, transition from a rating of "CC" to "D" or "SD" is considered to be single notch, not multiple notch.

Data analysis code and results are shown below:

Part I: Data Processing

Initial Setup to include dataframe package

```
In [1]: # Initial setup
        using Pkg;
        Pkg.add("DataFrames");
```

```
 Resolving package versions...
  Updating `~/.julia/Project.toml`
 [no changes]
  Updating `~/.julia/Manifest.toml`
 [no changes]
```

```
In [2]: using DataFrames;
        raw_data = readtable("raw_data.csv");
        size(raw_data)
```

```
Out[2]: (2387466, 6)
```

```
In [3]: data = raw_data[raw_data.splticrm .!== missing,:];
        size(data)
```

```
Out[3]: (503108, 6)
```

Data set decreases insignificantly after adjusting for missing values

```
In [4]: size(unique(data.gvkey),1)
```

```
Out[4]: 4984
```

Data shows that using tic symbol to separate companies may cause inconsistency; as shown above, the lengths are not uniform. Considering the data extract taken from January 1997 to February 2017, there should be 242 data entries and each with 6 columns. In the following analysis and prevent survivor bias/distortion on transition matrix, the inconsistencies are removed.

```
In [5]: selected_index = []
        for subdf in groupby(data, :gvkey)
            if size(subdf,1) === 242
                push!(selected_index,subdf.gvkey[1])
            end
        end
        size(selected_index,1)
```

```
Out[5]: 529
```

Only 529 companies have complete data from January 1997 to August 2017 out of the 4981 data. More analysis can be done by pushing the time frame closer to 2017. The amount of company having complete data will greatly increase since a lot of large cap tech companies are created after 2000.

In [6]:
```
selected_data = data[data.gvkey .=== selected_index[1],:]
for i in 2:size(selected_index,1)
    temp = data[data.gvkey .=== selected_index[i],:]
    selected_data = vcat(temp, selected_data)
end
size(selected_data,1)
```

Out[6]:
```
128018
```

After some more data cleaning, we can extract the data from the selected 529 companies. Data is restricted down from 503108 to 128018. Now, we can start analyzing the transitions between each months.

In [7]:
```
unique(selected_data.datadate)
```

Out[7]:
```
242-element Array{Union{Missing, Int64},1}:
 19970131
 19970228
 19970331
 19970430
 19970531
 19970630
 19970731
 19970831
 19970930
 19971031
 19971130
 19971231
 19980131
        ⋮
 20160331
 20160430
 20160531
 20160630
 20160731
 20160831
 20160930
 20161031
 20161130
 20161231
 20170131
 20170228
```

The next step is to replace splticrm ratings with numbers:

AAA => 21 AA+ => 20 AA => 19 AA- => 18 A+ => 17 A => 16 A- => 15 BBB+ => 14 BBB => 13 BBB- => 12 BB+ => 11 BB => 10 BB- => 9 B+ => 8 B => 7 B- => 6 CCC+ => 5 CCC => 4 CCC- => 3 CC => 2 SD => 1 D => 1

```
In [8]: unique(selected_data.splticrm)
```

```
Out[8]: 22-element Array{Union{Missing, String},1}:
         "AAA"
         "AA+"
         "AA"
         "AA-"
         "A"
         "A-"
         "BBB+"
         "BBB"
         "BBB-"
         "BB"
         "BB+"
         "A+"
         "B+"
         "BB-"
         "B"
         "B-"
         "CCC"
         "D"
         "CCC+"
         "CC"
         "SD"
         "CCC-"
```

In [9]:
```
sort!(selected_data, :datadate);
sort!(selected_data, :splticrm)
```

Out[9]:

	gvkey	splticrm	datadate	gsector	conm	tic
	Int64⍰	String⍰	Int64⍰	Int64⍰	String⍰	String⍰
1	100243	A	19970131	15	AMCOR LTD	AMCRY
2	61739	A	19970131	40	HARTFORD FINANCIAL SERVICES	HIG
3	29733	A	19970131	15	MARTIN MARIETTA MATERIALS	MLM
4	28349	A	19970131	40	ALLSTATE CORP	ALL
5	28216	A	19970131	40	REINSURANCE GROUP AMER INC	RGA
6	25157	A	19970131	45	FIRST DATA CORP	FDC
7	16560	A	19970131	15	ALUMINA LTD	AWCMY
8	15620	A	19970131	40	NATIONAL BANK CANADA	NTIOF
9	15305	A	19970131	55	MICHIGAN CONSOLIDATED GAS CO	MCN1
10	14822	A	19970131	40	BERKLEY (W R) CORP	WRB
11	13498	A	19970131	25	CARNIVAL CORP/PLC (USA)	CCL
12	12555	A	19970131	55	INTERSTATE POWER & LIGHT CO	LNT2
13	12428	A	19970131	55	PORTLAND GENERAL ELECTRIC CO	POR
14	12383	A	19970131	15	NORSK HYDRO ASA	NHYDY
15	11636	A	19970131	45	XEROX CORP	XRX
16	11465	A	19970131	25	WHIRLPOOL CORP	WHR
17	11456	A	19970131	60	WEYERHAEUSER CO	WY
18	11304	A	19970131	55	AVISTA CORP	AVA
19	11188	A	19970131	55	VIRGINIA ELECTRIC & POWER CO	D1
20	10614	A	19970131	40	TORCHMARK CORP	TMK
21	10530	A	19970131	35	THERMO FISHER SCIENTIFIC INC	TMO
22	10499	A	19970131	45	TEXAS INSTRUMENTS INC	TXN
23	10405	A	19970131	15	ALLEGHENY TECHNOLOGIES INC	ATI
24	10016	A	19970131	20	STANLEY BLACK & DECKER INC	SWK
25	9860	A	19970131	10	SOUTHERN NATURAL GAS CO	SNT1
26	9850	A	19970131	55	SOUTHERN CO	SO
27	9828	A	19970131	55	SOUTH CAROLINA ELEC & GAS CO	SCG1
28	9818	A	19970131	25	SONY CORP	SNE
29	9815	A	19970131	15	SONOCO PRODUCTS CO	SON
30	8543	A	19970131	30	ALTRIA GROUP INC	MO
⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [10]:
```
rating_string = selected_data.splticrm;
rating_float = Array{Int64}(undef,(size(rating_string,1),1))
for i in 1:size(rating_string,1)
    if rating_string[i] == "AAA"
        rating_float[i] = 21
    elseif rating_string[i] == "AA+"
        rating_float[i] = 20
    elseif rating_string[i] == "AA"
        rating_float[i] = 19
    elseif rating_string[i] == "AA-"
        rating_float[i] = 18
    elseif rating_string[i] == "A+"
        rating_float[i] = 17
    elseif rating_string[i] == "A"
        rating_float[i] = 16
    elseif rating_string[i] == "A-"
        rating_float[i] = 15
    elseif rating_string[i] == "BBB+"
        rating_float[i] = 14
    elseif rating_string[i] == "BBB"
        rating_float[i] = 13
    elseif rating_string[i] == "BBB-"
        rating_float[i] = 12
    elseif rating_string[i] == "BB+"
        rating_float[i] = 11
    elseif rating_string[i] == "BB"
        rating_float[i] = 10
    elseif rating_string[i] == "BB-"
        rating_float[i] = 9
    elseif rating_string[i] == "B+"
        rating_float[i] = 8
    elseif rating_string[i] == "B"
        rating_float[i] = 7
    elseif rating_string[i] == "B-"
        rating_float[i] = 6
    elseif rating_string[i] == "CCC+"
        rating_float[i] = 5
    elseif rating_string[i] == "CCC"
        rating_float[i] = 4
    elseif rating_string[i] == "CCC-"
        rating_float[i] = 3
    elseif rating_string[i] == "CC"
        rating_float[i] = 2
    elseif rating_string[i] == "D" || rating_string[i] == "SD"
        rating_float[i] = 1
    end
end
```

In [11]:
```julia
delete!(selected_data,:splticrm);
rating_float = convert(DataFrame, rating_float);
rename!(rating_float, :x1 => :rating);
selected_data = hcat(selected_data, rating_float)
```

Out[11]:

	gvkey	datadate	gsector	conm	tic	rating
	Int64⍰	Int64⍰	Int64⍰	String⍰	String⍰	Int64
1	100243	19970131	15	AMCOR LTD	AMCRY	16
2	61739	19970131	40	HARTFORD FINANCIAL SERVICES	HIG	16
3	29733	19970131	15	MARTIN MARIETTA MATERIALS	MLM	16
4	28349	19970131	40	ALLSTATE CORP	ALL	16
5	28216	19970131	40	REINSURANCE GROUP AMER INC	RGA	16
6	25157	19970131	45	FIRST DATA CORP	FDC	16
7	16560	19970131	15	ALUMINA LTD	AWCMY	16
8	15620	19970131	40	NATIONAL BANK CANADA	NTIOF	16
9	15305	19970131	55	MICHIGAN CONSOLIDATED GAS CO	MCN1	16
10	14822	19970131	40	BERKLEY (W R) CORP	WRB	16
11	13498	19970131	25	CARNIVAL CORP/PLC (USA)	CCL	16
12	12555	19970131	55	INTERSTATE POWER & LIGHT CO	LNT2	16
13	12428	19970131	55	PORTLAND GENERAL ELECTRIC CO	POR	16
14	12383	19970131	15	NORSK HYDRO ASA	NHYDY	16
15	11636	19970131	45	XEROX CORP	XRX	16
16	11465	19970131	25	WHIRLPOOL CORP	WHR	16
17	11456	19970131	60	WEYERHAEUSER CO	WY	16
18	11304	19970131	55	AVISTA CORP	AVA	16
19	11188	19970131	55	VIRGINIA ELECTRIC & POWER CO	D1	16
20	10614	19970131	40	TORCHMARK CORP	TMK	16
21	10530	19970131	35	THERMO FISHER SCIENTIFIC INC	TMO	16
22	10499	19970131	45	TEXAS INSTRUMENTS INC	TXN	16
23	10405	19970131	15	ALLEGHENY TECHNOLOGIES INC	ATI	16
24	10016	19970131	20	STANLEY BLACK & DECKER INC	SWK	16
25	9860	19970131	10	SOUTHERN NATURAL GAS CO	SNT1	16
26	9850	19970131	55	SOUTHERN CO	SO	16
27	9828	19970131	55	SOUTH CAROLINA ELEC & GAS CO	SCG1	16
28	9818	19970131	25	SONY CORP	SNE	16
29	9815	19970131	15	SONOCO PRODUCTS CO	SON	16
30	8543	19970131	30	ALTRIA GROUP INC	MO	16
⋮	⋮	⋮	⋮	⋮	⋮	⋮

```
In [12]: sort!(selected_data,:tic)
```

```
Out[12]:
        gvkey   datadate  gsector         conm     tic  rating
       Int64⍰     Int64⍰   Int64⍰      String⍰  String⍰  Int64
  1     2316    20061031       15   HEXION INC   0141A       7
  2     2316    20061130       15   HEXION INC   0141A       7
  3     2316    20061231       15   HEXION INC   0141A       7
  4     2316    20070131       15   HEXION INC   0141A       7
  5     2316    20070228       15   HEXION INC   0141A       7
  6     2316    20070331       15   HEXION INC   0141A       7
  7     2316    20070430       15   HEXION INC   0141A       7
  8     2316    20070531       15   HEXION INC   0141A       7
  9     2316    20070630       15   HEXION INC   0141A       7
 10     2316    20070731       15   HEXION INC   0141A       7
 11     2316    20070831       15   HEXION INC   0141A       7
 12     2316    20070930       15   HEXION INC   0141A       7
 13     2316    20071031       15   HEXION INC   0141A       7
 14     2316    20071130       15   HEXION INC   0141A       7
 15     2316    20071231       15   HEXION INC   0141A       7
 16     2316    20080131       15   HEXION INC   0141A       7
 17     2316    20080229       15   HEXION INC   0141A       7
 18     2316    20080331       15   HEXION INC   0141A       7
 19     2316    20080430       15   HEXION INC   0141A       7
 20     2316    20080531       15   HEXION INC   0141A       7
 21     2316    20080630       15   HEXION INC   0141A       7
 22     2316    20080731       15   HEXION INC   0141A       7
 23     2316    20080831       15   HEXION INC   0141A       7
 24     2316    20080930       15   HEXION INC   0141A       7
 25     2316    20081031       15   HEXION INC   0141A       7
 26     2316    20040831       15   HEXION INC   0141A       8
 27     2316    20040930       15   HEXION INC   0141A       8
 28     2316    20041031       15   HEXION INC   0141A       8
 29     2316    20041130       15   HEXION INC   0141A       8
 30     2316    20041231       15   HEXION INC   0141A       8
  ⋮       ⋮         ⋮          ⋮           ⋮        ⋮        ⋮
```

Graphical Representation of Rating Transition

As shown above in data analysis code and results, the data is cleaned and transitioned into a DataFrame that can be easily separated by ticker or by time. Once the DataFrame is set up, one can easily map out the rating transition of any one of 529 companies.

One must be aware of the following encoding scheme:

AAA	AA+	AA	AA-	A+	A	A-	BBB+	BBB	BBB-	BB+	BB	BB-
21	20	19	18	17	16	15	14	13	12	11	10	9
B+	B	B-	CCC+	CCC	CCC-	CC	SD/D					
8	7	6	5	4	3	2	1					

Since the rating hierarchy (for example AAA > AA+) is hard to encode in Julia, a one-hot number encoding scheme is used. With the one-hot number encoding scheme, one can generate graphs like the ones below.

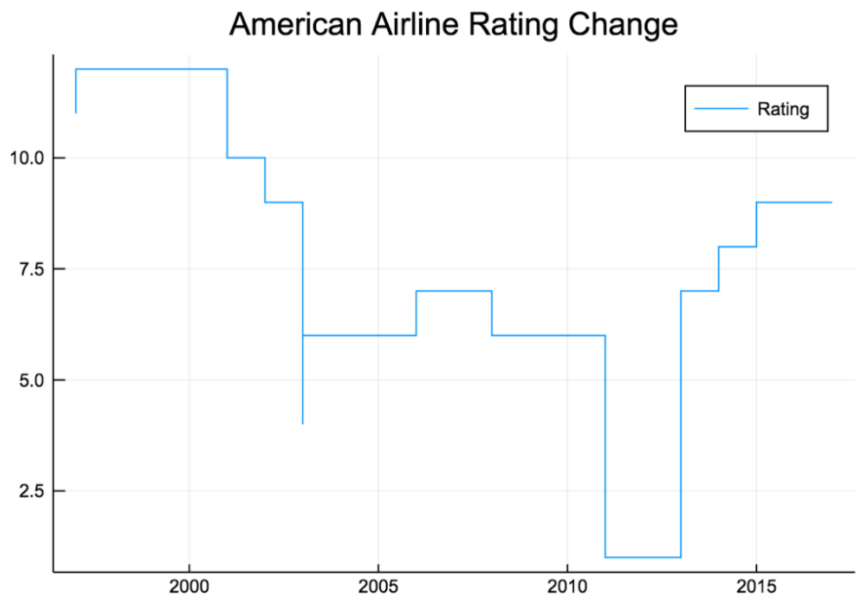


Figure 3, American Airline Credit Transition Diagram

From the graph above, one can see that American Airline has a relatively rocky early 2010s. One can see a significant drop of credit quality from around 6 to default and a quick bounce back in 2013. This suggests a Chapter 11 restructuring is declared in the early 2010s. This is consistent with the news. On November 29th 2011, American Airlines declared Chapter 11 bankruptcy and emerged as a better and more structured airline on December 8th 2013. The subsequent company was in a better position with a stronger capital structure as reflected in the rise in credit rating.

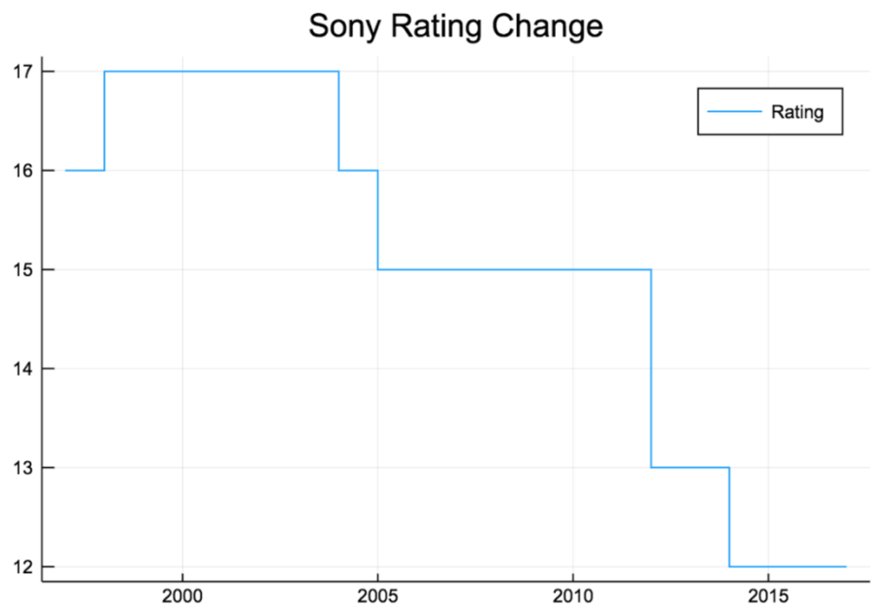


Figure 4, Sony Credit Transition Diagram

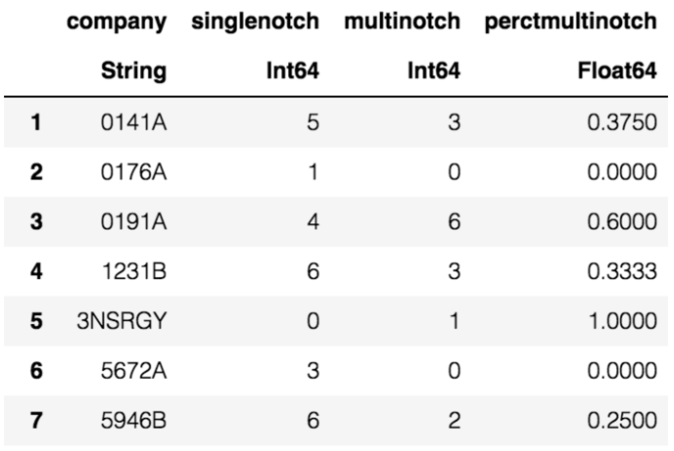
One can also look at another example: Sony Corporation. Unlike American Airlines which is in the more volatile and cyclical airline industry, Sony corporation had a more consistent credit rating. Back in the early 2000s, Sony had a dominant position in the electronics industry. This was reflected by the high investment grade rating of A+. However, as the financial crisis hit in 2008, Sony started seeing a decline in credit rating, dropping from 17 to 12 in 2013. With the intense competition from other electronic companies and lackluster performance in recent years, Sony faced a grim future as it barely hanged on to its investment grade rating. From these two credit rating transition diagrams, readers can easily understand what the company had gone through and perhaps where it is going.

After exploring the graphical representation of credit transitions, one can move on to the hypotheses.

Single Notch, Multiple Notch Transitions Percentage

Hypothesis 1: There are more single notch transitions than multi-notch transitions as companies are less likely to receive multi-notch downgrades or upgrades during non-recession periods

This section is separated into two parts as the data is analyzed from two angles: company and time. The percentage of multi-notch transitions is first calculated by company. This allows me to error check and confirm the calculation in the time section. One subtlety is that there was a significant amount of transitions in-between years. These cross-year transitions are included in the second year. For example, there are multiple transitions between December 1997 and January 1998; they are included in 1998 data. These transitions won't be taken into account if one simply looks at the transitions within the year. Therefore, there are two sets of calculations: one that includes cross-year transitions and one that doesn't.



	company	singlenotch	multinotch	perctmultinotch
	String	Int64	Int64	Float64
1	0141A	5	3	0.3750
2	0176A	1	0	0.0000
3	0191A	4	6	0.6000
4	1231B	6	3	0.3333
5	3NSRGY	0	1	1.0000
6	5672A	3	0	0.0000
7	5946B	6	2	0.2500

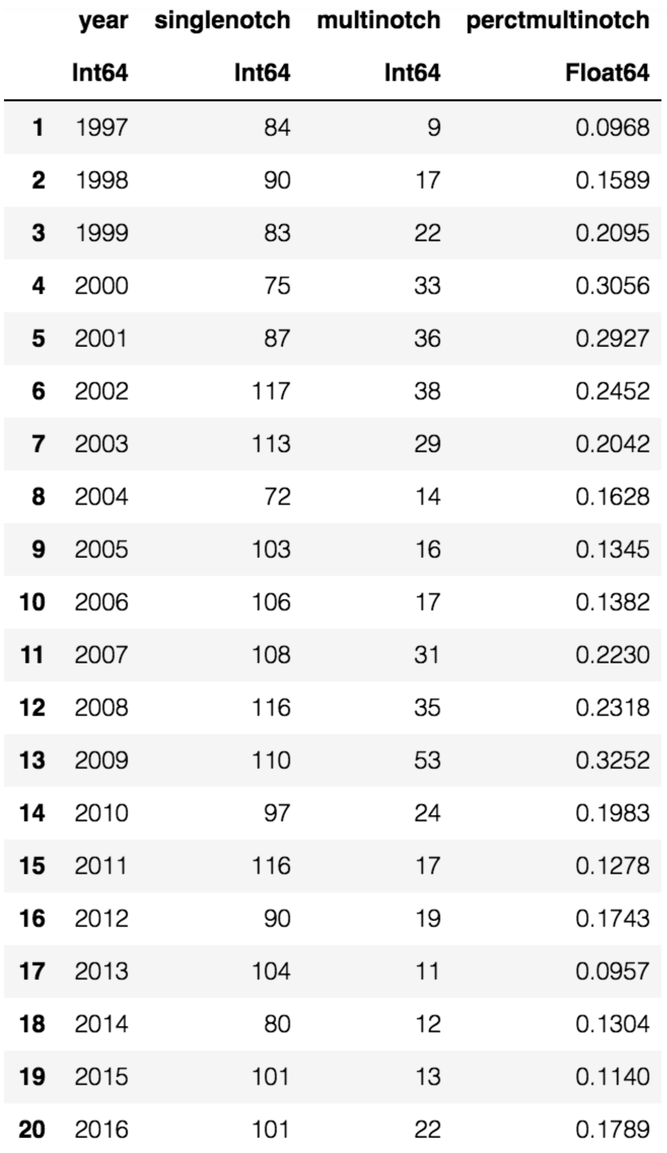
Figure 5, Single and Multi-notch Transition Percentage

One can see that the percentage of multi-notch transition for each company is relatively low baring some exceptions. The total number of multi-notch transitions without accounting cross-year transitions is 468 while the number of single notch is 1963. When cross-year transitions are taken into account, the total number of multi-notch transitions amounts to 502 and the number of single notch is 2099. The percentage of multi-notch transition is approximately 24%. **This supports hypothesis 1 that multi-notch transitions account for a relatively small percentage of all transitions.**

However, one should be aware that this might not be true if a longer period is used or the right censoring is not implemented. Because of the data censoring, companies analyzed here are all relatively stable. They either had a low risk profile or they survived a restructuring and bounced back. Thus, they have a lower possibility of receiving a multi-notch upgrade or downgrade. For example, during the tech bubble in the early 2000s, numerous tech companies went bankrupt and suffered severe multi-notch transitions. These companies are not included in the analysis.

Hypothesis 2: There are more multi-notch transitions during crisis period as companies face harsher economic conditions

After calculating the percentage of multi-notch transitions for each company, one can move on to the time non-homogeneous portion of the calculation. For each year, the number of single notch and multi-notch transitions are amassed, resulting in the following table:



	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
1	1997	84	9	0.0968
2	1998	90	17	0.1589
3	1999	83	22	0.2095
4	2000	75	33	0.3056
5	2001	87	36	0.2927
6	2002	117	38	0.2452
7	2003	113	29	0.2042
8	2004	72	14	0.1628
9	2005	103	16	0.1345
10	2006	106	17	0.1382
11	2007	108	31	0.2230
12	2008	116	35	0.2318
13	2009	110	53	0.3252
14	2010	97	24	0.1983
15	2011	116	17	0.1278
16	2012	90	19	0.1743
17	2013	104	11	0.0957
18	2014	80	12	0.1304
19	2015	101	13	0.1140
20	2016	101	22	0.1789

Figure 6, Single and Multi-Notch Transitions Over the Years

The DataFrame above confirms the hypothesis that during financial downturns, the percentage of multi-notch transitions is significantly higher. The first recession during this period happened in 2000 which is the internet bubble. The percentage of multi-notch transition spikes from 21% to 31%, which is a 46% increase. The following year 2001 also had a high percentage of 29%. Moving onto the most recent 2008 financial crisis, one can also observe a similar pattern. At the end of 2008 and the height of the crisis, Lehman Brothers filed for bankruptcy. The market went into deep recession in the following year. This can be observed from the data as the percentage spiked in percentage in 2009. The percentage shots up by 40%, going from 23% to 33%.

If there is no data censoring, one can expect the phenomenon to be even more extreme. With more companies filing for bankruptcy during recession years, one can expect the see a much larger and persistent spike around the recession. From the data above, one can see that the spike tends to persist over a few years. Using American Airlines as an example, from early analysis, we know that it went through a Chapter 11 in 2011 and emerged two years later in 2013. if there was a multi-notch downgrade, then it was also possible that in the near future, there will be a multi-notch upgrade out of default. This bounce back effect is what bolsters the persistent high percentage of multi-notch transitions during recessions.

The single and multi-notch transition analysis is shown below:

Part II: Single Notch Transitions, Multiple Notch Transitions Percentage

Calculate the percentage of single notch change and multiple notch change

Part 1: Per company (transitioning between months)

In [15]:
```
part1_transition_df_with_year_transition = DataFrame(company = String[], si
part1_transition_df_without_year_transition = DataFrame(company = String[],
multiple_notch_counter = 0;
single_notch_counter = 0;
```

The code below counts the transition between each month, including any transition inbetween years.

In [16]:
```
for subgroup in groupby(selected_data, :gvkey)
    subgroup = sort(subgroup,:datadate)
    temp_rating = subgroup.rating
    previous = temp_rating[1]
    multiple_notch_counter = 0
    single_notch_counter = 0
    for i in 2:size(temp_rating,1)
        if abs(temp_rating[i] - previous) > 1
            multiple_notch_counter = multiple_notch_counter + 1
        elseif abs(temp_rating[i] - previous) == 1
            single_notch_counter = single_notch_counter + 1
        end
        previous = temp_rating[i]
    end
    perct = multiple_notch_counter / (single_notch_counter + multiple_notch
    name = subgroup.tic[1]
    push!(part1_transition_df_with_year_transition, [name, single_notch_cou
end
head(part1_transition_df_with_year_transition)
```

Out[16]:

	company	singlenotch	multinotch	perctmultinotch
	String	Int64	Int64	Float64
1	0141A	6	3	0.333333
2	0176A	2	0	0.0
3	0191A	4	6	0.6
4	1231B	7	3	0.3
5	3NSRGY	0	1	1.0
6	5672A	3	0	0.0

```
In [17]: println(sum(part1_transition_df_with_year_transition[:,2]))
         println(sum(part1_transition_df_with_year_transition[:,3]))
```

```
2099
502
```

This code below counts the transition within the year, excluding cross-year transition; as it turns out, ratings change quite a lot transitioning between years.

```
In [18]: part2_data = selected_data;
         temp_year = trunc.(Int, (part2_data.datadate ./ 10000));
         year = DataFrame();
         year = hcat(year, temp_year);
         rename!(year, :x1 => :year);
         part2_data = hcat(part2_data, year);
```

```
In [19]: for subgroup in groupby(part2_data, :gvkey)
             subgroup = sort(subgroup,:datadate)
             temp_rating = subgroup.rating
             temp_year = subgroup.year
             previous = temp_rating[1]
             previous_year = temp_year[1]
             multiple_notch_counter = 0
             single_notch_counter = 0
             for i in 2:size(temp_rating,1)
                 if abs(temp_rating[i] - previous) > 1 && temp_year[i] == previous_y
                     multiple_notch_counter = multiple_notch_counter + 1
                 elseif abs(temp_rating[i] - previous) == 1 && temp_year[i] == previ
                     single_notch_counter = single_notch_counter + 1
                 end
                 previous = temp_rating[i]
                 previous_year = temp_year[i]
             end
             perct = multiple_notch_counter / (single_notch_counter + multiple_notch
             name = subgroup.tic[1]
             push!(part1_transition_df_without_year_transition, [name, single_notch_
         end
         head(part1_transition_df_without_year_transition)
```

Out[19]:

	company	singlenotch	multinotch	perctmultinotch
	String	Int64	Int64	Float64
1	0141A	5	3	0.375
2	0176A	1	0	0.0
3	0191A	4	6	0.6
4	1231B	6	3	0.333333
5	3NSRGY	0	1	1.0
6	5672A	3	0	0.0

In [20]:
```
println(sum(part1_transition_df_without_year_transition[:,2]))
println(sum(part1_transition_df_without_year_transition[:,3]))
```

```
1963
468
```

Part 2: Over time, Per year

In [21]:
```
sition_df_without_year_transition = DataFrame(year = Int[1997, 1998, 1999, 2
sition_df_with_year_transition = DataFrame(year = Int[1997, 1998, 1999, 200(
ultinotch_counter = 0;
inglenotch_counter = 0;
```

For easier processing, create a column of the year

```
In [22]: # Still use tic as the separating factor, then slowly accumulate the counte
         for subgroup in groupby(part2_data, :gvkey)
             subgroup = sort(subgroup,:datadate)
             temp_data = DataFrame(subgroup)
             counter = 1
             for subtime in groupby(temp_data, :year)
                 temp_rating = subtime.rating
                 previous = temp_rating[1]
                 overtime_multinotch_counter = 0
                 overtime_singlenotch_counter = 0
                 for i in 2:size(temp_rating,1)
                     if abs(temp_rating[i] - previous) > 1
                         overtime_multinotch_counter = overtime_multinotch_counter +
                     elseif abs(temp_rating[i] - previous) == 1
                         overtime_singlenotch_counter = overtime_singlenotch_counter
                     end
                     previous = temp_rating[i]
                 end
                 part2_transition_df_without_year_transition[counter,2] = part2_tran
                 part2_transition_df_without_year_transition[counter,3] = part2_tran
                 counter = counter + 1
             end
         end
         part2_transition_df_without_year_transition.perctmultinotch = part2_transit
         part2_transition_df_without_year_transition
```

Out[22]:

	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
1	1997	84	9	0.0967742
2	1998	90	17	0.158879
3	1999	83	22	0.209524
4	2000	75	33	0.305556
5	2001	87	36	0.292683
6	2002	117	38	0.245161
7	2003	113	29	0.204225
8	2004	72	14	0.162791
9	2005	103	16	0.134454
10	2006	106	17	0.138211
11	2007	108	31	0.223022
12	2008	116	35	0.231788
13	2009	110	53	0.325153
14	2010	97	24	0.198347
15	2011	116	17	0.12782
16	2012	90	19	0.174312
17	2013	104	11	0.0956522

	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
18	2014	80	12	0.130435
19	2015	101	13	0.114035
20	2016	101	22	0.178862
21	2017	10	0	0.0

In [23]:
```
println(sum(part2_transition_df_without_year_transition[:,2]))
println(sum(part2_transition_df_without_year_transition[:,3]))
```

```
1963
468
```

To include the cross-year transitions into the next year's transition: for example, 1997 Dec => 1998 Jan, there is a single notch rating change, it will be recorded as a 1998 transition, not 1997 transition.

In [24]:
```
for subgroup in groupby(part2_data, :gvkey)
    subgroup = sort(subgroup,:datadate)
    temp_data = DataFrame(subgroup)
    counter = 1
    init_flag = 0
    previous_year = 0
    for subtime in groupby(temp_data, :year)
        temp_rating = subtime.rating
        previous = temp_rating[1]
        overtime_multinotch_counter = 0
        overtime_singlenotch_counter = 0

        if init_flag == 0
            previous_year = temp_rating[end]
        elseif init_flag == 1
            if abs(previous - previous_year) > 1
                overtime_multinotch_counter = overtime_multinotch_counter +
            elseif abs(previous - previous_year) == 1
                overtime_singlenotch_counter = overtime_singlenotch_counter
            end
            previous_year = temp_rating[end]
        end

        for i in 2:size(temp_rating,1)
            if abs(temp_rating[i] - previous) > 1
                overtime_multinotch_counter = overtime_multinotch_counter +
            elseif abs(temp_rating[i] - previous) == 1
                overtime_singlenotch_counter = overtime_singlenotch_counter
            end
            previous = temp_rating[i]
        end
        part2_transition_df_with_year_transition[counter,2] = part2_transit
        part2_transition_df_with_year_transition[counter,3] = part2_transit
        counter = counter + 1
        init_flag = 1
    end
end
part2_transition_df_with_year_transition.perctmultinotch = part2_transition
part2_transition_df_with_year_transition
```

Out[24]:

	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
1	1997	84	9	0.0967742
2	1998	98	17	0.147826
3	1999	94	24	0.20339
4	2000	81	36	0.307692
5	2001	98	40	0.289855
6	2002	126	40	0.240964
7	2003	118	31	0.208054
8	2004	79	14	0.150538

	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
9	2005	107	16	0.130081
10	2006	115	21	0.154412
11	2007	112	31	0.216783
12	2008	123	40	0.245399
13	2009	126	59	0.318919
14	2010	103	25	0.195313
15	2011	121	18	0.129496
16	2012	96	19	0.165217
17	2013	108	12	0.1
18	2014	82	12	0.12766
19	2015	110	13	0.105691
20	2016	104	23	0.181102
21	2017	14	2	0.125

In [25]:
```
println(sum(part2_transition_df_with_year_transition[:,2]))
println(sum(part2_transition_df_with_year_transition[:,3]))
```

```
2099
502
```

Transition Probability Matrix and Stationary Distribution

After accumulating the transitions, one can look to fit a Markov chain to each time period and compute the transition probability matrix. The transition matrix is computed using maximum likelihood. The calculation process is very similar to the one mentioned in class and it is annotated with markdowns in the code. The transition probability matrix for each year is stored inside an array; each element in the array is a 21-by-21 matrix.

From observing the array of transition probability matrices, one can see that single notch transitions account for the majority of the transitions. All the transition probability matrices are essentially tridiagonal matrix with the diagonals being close to one. This is consistent with the result from earlier section. Another observation is that multi-notch transitions are cluster around default and CC. This is due to that fact that default is not an absorbing state in this model. Companies that go into default or have lower ratings tend to go into chapter 11 to restructure or selective default some of their debts. Instead of chapter 7 bankruptcy, these companies survive and their ratings are reevaluated, hence the clustering of multi-notch upgrades and downgrades.

The final observation we can make is that ratings above BB- tend to have fatter right tails, indicating that there is a higher probability of upgrading. On the other hand, ratings lower than BB- have fatter left tails. A possible explanation for this dichotomy is that companies that have ratings lower than BB- might be facing escalating managerial or financial problems while companies that are close to investment grade or close to investment grade have the comparative advantage or market share to stay in that region.

Aside from the transition probability matrix, stationary distribution for each Markov chain can be computed. One can see that the stationary distribution is centered around the ratings in the middle with a maximum at BBB.

The transition probability matrix and stationary distribution analysis can be seen below:

Part III: Transition Probabilities and Stationary Distribution

Transition probabilities are calculated by year, so we can see how the transition probabilties grow over the years. The following calculation is without cross-year transitions. Incorporating cross-year transition proves to be problematic.

In [26]:
```
transition_matrix_array = [];
transition_data = selected_data;
temp_month = trunc.(Int, (transition_data.datadate - (trunc.(Int, (transiti
month = DataFrame();
month = hcat(year, temp_month);
rename!(month, :x1 => :month);
transition_data = hcat(transition_data, month);
```

In [27]:
```
# Storing data into a total transition matrix just in case
total_transition_matrix = DataFrame()
# A quick creation of transition matrix using for loop
for i in 1:21
    temp = Array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])
    total_transition_matrix = hcat(total_transition_matrix, temp)
end
rename!(total_transition_matrix, Dict(:x1 => Symbol("D/SD"), :x1_1 => Symbo
In [28]: summation = 0
sort!(transition_data,:datadate)
for subtime in groupby(transition_data, :year)
    subtime = sort(subtime, :datadate)
    temp_data = DataFrame(subtime)

    transition_matrix = DataFrame()
    for i in 1:21
        temp = Array([0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0])
        transition_matrix = hcat(transition_matrix, temp)
    end
    rename!(transition_matrix, Dict(:x1 => Symbol("D/SD"), :x1_1 => Symbol(
    for subgroup in groupby(temp_data, :gvkey)
        temp_rating = subgroup.rating
        previous = temp_rating[1]

        for i in 2:size(temp_rating,1)
            transition_matrix[previous,temp_rating[i]] += 1
            total_transition_matrix[previous,temp_rating[i]] += 1
            previous = temp_rating[i]
        end
    end

    # Add to the transition_matrix_array
    push!(transition_matrix_array, transition_matrix)
end
size(transition_matrix_array,1)

Out[28]: 21
```

In [29]:
```
total_transition_matrix
```

Out[29]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	BB+	BBB-	BBB	BBB+
	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64
1	202	0	0	2	4	0	7	0	1	1	0	1	0	0
2	5	28	1	1	2	1	1	0	0	0	0	0	0	0
3	1	2	68	0	1	1	0	0	0	0	0	0	0	0
4	3	5	0	101	5	5	0	0	0	0	0	0	0	0
5	2	2	2	6	383	16	4	0	0	0	0	0	0	0
6	2	1	0	7	19	1092	24	7	2	0	0	0	1	0
7	1	2	1	2	3	29	2232	40	7	0	0	0	0	0
8	0	1	0	1	2	11	51	2761	50	7	1	0	0	0
9	0	0	0	0	0	5	7	53	3694	53	11	0	0	0
10	0	0	0	0	0	1	2	10	52	4710	67	11	2	0
11	0	0	0	0	0	0	4	4	17	50	4974	75	13	0
12	0	0	0	0	0	0	0	1	3	29	70	10604	116	8
13	1	0	0	0	0	0	0	2	2	6	11	133	18739	141
14	0	0	0	0	0	0	0	0	1	1	3	24	163	15833
15	0	0	0	0	0	0	0	0	0	0	1	3	30	156
16	0	0	0	0	0	0	1	0	0	0	2	0	7	29
17	0	0	0	0	0	0	0	0	0	0	0	0	2	5
18	0	0	0	0	0	0	0	0	0	0	0	0	0	2
19	0	0	0	0	0	0	0	0	0	0	0	0	1	1
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0

In [30]:
```julia
transition_matrix_array[7]
```

Out[30]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	BB+	BBB-	BBB	BBB+	I
	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	Int64	I
1	30	0	0	0	1	0	0	0	0	0	0	0	0	0	
2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
3	1	0	1	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	4	0	2	0	0	0	0	0	0	0	0	
5	0	0	0	0	19	1	0	0	0	0	0	0	0	0	
6	0	0	0	2	1	40	1	0	1	0	0	0	0	0	
7	0	0	1	0	1	0	46	2	0	0	0	0	0	0	
8	0	0	0	0	0	0	4	142	2	0	0	0	0	0	
9	0	0	0	0	0	2	1	5	213	4	0	0	0	0	
10	0	0	0	0	0	0	1	2	5	237	1	0	1	0	
11	0	0	0	0	0	0	1	0	1	3	262	3	0	0	
12	0	0	0	0	0	0	0	0	1	1	3	484	10	1	
13	0	0	0	0	0	0	0	0	0	1	1	18	1021	3	
14	0	0	0	0	0	0	0	0	0	0	0	1	10	634	
15	0	0	0	0	0	0	0	0	0	0	0	1	1	6	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
21	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [31]:
```julia
using Printf
Base.show( io::IO, x::Float64) = @printf(io, "%0.4f", x)
```

```
In [32]: total_transition_probability_matrix = DataFrame(x1 = [], x1_1 = [], x1_2 =
         rename!(total_transition_probability_matrix, Dict(:x1 => Symbol("D/SD"), :x
         for row = 1:size(total_transition_matrix,1)
             temp_row = convert(Array, total_transition_matrix[row,:]) / sum(convert
             push!(total_transition_probability_matrix, temp_row)
         end
         total_transition_probability_matrix
```

Out[32]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	BB+	BBB-
	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any
1	0.9266	0.0000	0.0000	0.0092	0.0183	0.0000	0.0321	0.0000	0.0046	0.0046	0.0000	0.0046
2	0.1282	0.7179	0.0256	0.0256	0.0513	0.0256	0.0256	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0137	0.0274	0.9315	0.0000	0.0137	0.0137	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.0252	0.0420	0.0000	0.8487	0.0420	0.0420	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.0048	0.0048	0.0048	0.0145	0.9229	0.0386	0.0096	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0017	0.0009	0.0000	0.0061	0.0165	0.9455	0.0208	0.0061	0.0017	0.0000	0.0000	0.0000
7	0.0004	0.0009	0.0004	0.0009	0.0013	0.0125	0.9633	0.0173	0.0030	0.0000	0.0000	0.0000
8	0.0000	0.0003	0.0000	0.0003	0.0007	0.0038	0.0177	0.9570	0.0173	0.0024	0.0003	0.0000
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0013	0.0018	0.0139	0.9663	0.0139	0.0029	0.0000
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0002	0.0004	0.0021	0.0107	0.9701	0.0138	0.0023
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0008	0.0008	0.0033	0.0097	0.9683	0.0146
12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0003	0.0027	0.0065	0.9788
13	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0003	0.0006	0.0070
14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0002	0.0015
15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0002
16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0001	0.0000
17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
21	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Each transition matrix in the transition matrix array can be swapped into transition probability matrix using the same conversion process shown above

In [33]:
```
transition_probability_matrix_array = []
for year = 1:size(transition_matrix_array,1)
    temp_transition_probability_matrix = DataFrame(x1 = [], x1_1 = [], x1_2
    rename!(temp_transition_probability_matrix, Dict(:x1 => Symbol("D/SD"),
    for row = 1:size(transition_matrix_array[year],1)
        temp_row = convert(Array, transition_matrix_array[year][row,:]) / s
        if sum(convert(Array, transition_matrix_array[year][row,:])) == 0
            temp_row = Array([0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0
        end
        push!(temp_transition_probability_matrix, temp_row)
    end
    push!(transition_probability_matrix_array, temp_transition_probability_
end
```

In [34]:
```
transition_probability_matrix_array[6]
```

Out[34]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	BB+	BBB-
	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any
1	0.8333	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1667	0.0000	0.0000
2	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
3	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	0.2500	0.0000	0.0000	0.7500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	0.0000	0.0000	0.0000	0.0000	0.7500	0.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.0000
6	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	0.0143	0.0000	0.0000	0.0143	0.0000	0.0286	0.9286	0.0000	0.0143	0.0000	0.0000	0.0000
8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0253	0.9620	0.0127	0.0000	0.0000	0.0000
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0105	0.9895	0.0000	0.0000	0.0000
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0038	0.0000	0.0038	0.0226	0.9547	0.0151	0.0000
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0040	0.0202	0.9676	0.0081
12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0041	0.0104	0.9814
13	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0021	0.0011	0.0032	0.0000	0.0096
14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0013	0.0013	0.0027
15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
21	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Since we can have the transition matrix, we can find the stationary distribution; since the yearly transition probability matrix is too sparse, the total transition probability matrix is used

In [35]:
```
using LinearAlgebra
M = convert(Array, total_transition_probability_matrix) - Matrix{Float64}(I
M[:,1] = ones(21);
stationary_distribution = DataFrame([1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 
rename!(stationary_distribution, Dict(:x1 => Symbol("D/SD"), :x2 => Symbol(
```

Out[35]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	BB+
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64
1	0.0067	0.0015	0.0020	0.0041	0.0148	0.0372	0.0688	0.0634	0.0762	0.0802	0.0744

Conclusion

Using the concepts taught in Markov Theory and Application by Professor Atteson, I was able to analyze the single and multi-notch credit rating transitions. It serves as an extension to the special topic course and I learned a lot in the process of doing this project.

Crucial findings of this project: First, with the data censoring and cleaning, data set suggests that multi-notch rating transitions only accounted for a small percentage of transitions. Second, multi-notch transitions percentage spiked during recessions as more companies received multi-notch downgrades and also subsequent multi-notch upgrades.

Interesting side notes: From the transition probability matrix, one can see that multi-notch transitions cluster around lower half of the rating spectrum. Companies struggling with financials tend to be more volatile and the credit ratings reflected that. Furthermore, transition matrices suggest that companies that have ratings above BB- tend to have fatter right tails.

Future Improvement

A more sophisticated and realistic take on the credit rating model is the hidden Markov model which suggests that there are hidden states that are not observable and what one sees is simply the manifestation of the hidden state transitions. The hidden Markov model is relatively hard to implement. One can assume there is a hidden stable state with a good credit rating and a hidden chaos state with bad credit rating, then the calculation for forward and backward probability can go on from there.

Another direction that also seems interesting is using the probability transition matrix to design a reinforcement learning model. One of the problem in using dynamic programming to implement a reinforcement learning model is not having the probability distribution of state and action pairs. In this case, one can model the situation by connecting state action pairs to the transition matrices. The rows are the states and columns are the actions. If one is in state AAA, it has 21 corresponding actions with corresponding probabilities. The only problem is the reward. Random assignment of heuristics will only skew the game. This is just one of possible projects that can done from this analysis.

References

[1] Atteson, Kevin. *Markov Chains in Credit Modeling*.
http://www.atteson.com/Markov/credit.html. Access: October 20, 2018

[2] Standard & Poor credit rating system
https://www.standardandpoors.com/en_US/web/guest/article/-/view/sourceId/504352. Access: May 11, 2019

[3] Virtamo, Jorma. *Queueing Theory and Birth Death Process*.
https://www.netlab.tkk.fi/opetus/s383143/kalvot/E_bdpros.pdf. Access: May 10, 2019

[4] Atteson, Kevin. *Classification of states, stationary distributions and convergence*.
http://www.atteson.com/Markov/stationary.html. Access: September 15, 2018