Markov Process in Finance - Credit Rating

1. Introduction

The objective for this project is to analyze the credit ratings phenomenon. In the following section, we will explore the single notch and multiple notch migrations by companies and sectors. We will also observe the changes over time and how it is affected during a financial crisis. The data we use for our analysis is the North America S&P credit ratings monthly data from WRDS.

Initial Setup to include dataframe package

In [1]:

```
using Pkg;
Pkg.add("DataFrames");
```

```
  Updating registry at `/home/jrun/.julia/registries/JuliaPro`
  Updating git-repo `https://pkg.juliacomputing.com/registry/JuliaPro`
[1mFetching: [========================================>]  100.0 %.0 %13.7 %>
                             ]  27.4 %                  ]  54.7 %>       ]
 80.7 %=====================================>  ]  94.4 % Resolving package
 versions...
  Updating `~/.julia/Project.toml`
 [no changes]
  Updating `~/.julia/Manifest.toml`
 [no changes]
```

In [2]:

```
using DataFrames
using Dates
using Plots
using LinearAlgebra
```

In [3]:

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

2. Company (1987 ~ 2017)

For this section, we will discuss the single and multi notch and transition matrix based on the ticker name and company names with all time data. Unlike the third section, we create a new dataframe here. The reason why we restrcuture the dataframe is because we don't want to fall into a survival bias when we drop all missing values.

2.1 Data Processing

In [4]:

```
raw_data = readtable("data.csv")
head(raw_data)
```

Out[4]:

	gvkey	splticrm	datadate	gsector	conm	tic
	Int64⍰	String⍰	Int64⍰	Int64⍰	String⍰	String⍰
1	1003	missing	20040630	25	A.A. IMPORTING CO INC	ANTQ
2	1003	missing	20040731	25	A.A. IMPORTING CO INC	ANTQ
3	1003	missing	20040831	25	A.A. IMPORTING CO INC	ANTQ
4	1003	missing	20040930	25	A.A. IMPORTING CO INC	ANTQ
5	1003	missing	20041031	25	A.A. IMPORTING CO INC	ANTQ
6	1003	missing	20041130	25	A.A. IMPORTING CO INC	ANTQ

In [5]:

```
data = dropmissing(raw_data)
head(data)
```

Out[5]:

	gvkey	splticrm	datadate	gsector	conm	tic
	Int64⍰	String⍰	Int64⍰	Int64⍰	String⍰	String⍰
1	1004	BBB	19870531	20	AAR CORP	AIR
2	1004	BBB	19870630	20	AAR CORP	AIR
3	1004	BBB	19870731	20	AAR CORP	AIR
4	1004	BBB	19870831	20	AAR CORP	AIR
5	1004	BBB	19870930	20	AAR CORP	AIR
6	1004	BBB	19871031	20	AAR CORP	AIR

In [6]:

```
ticker = unique(data[:1])
ticker_name = unique(data[:6])
```

Out[6]:

```
5947-element Array{Union{Missing, String},1}:
 "AIR"
 "4165A"
 "5548C"
 "AGS.2"
 "ALO.2"
 "UDI."
 "AEN.2"
 "AAL"
 "4267A"
 "ASTA"
 "ARXX"
 "4328B"
 "AVX"
 ⋮
 "KRG"
 "TAM.2"
 "ATBIF"
 "EVGPF"
 "PLZLY"
 "EXXIQ"
 "MODL"
 "EC"
 "CNHI"
 "PACDD"
 "TAM"
 "ALLE"
```

We can easily find that now the time is from 1987 to 2017. The number of companies is now 5947 rather than 4987. In this section, we try to do the analysis in a different way. Here, we try to join all companies into one dataframe, where each columns shows the rating for each company. (Note that the process to join all companies will take a while)

In [7]:

```
all = DataFrame(datadate=0)
for itr in enumerate(ticker)
#     println(itr[2])
    temp1 = data[data[:1].==itr[2],:][2:3]
    all=join(all, temp1, on= :datadate, kind= :outer)
end
```

In [8]:

```
real_all = all[2:end,:]
sort!(real_all);
head(real_all)
```

Out[8]:

	datadate	splticrm	splticrm_1	splticrm_2	splticrm_3	splticrm_4	splticrm_5	splticrm_6	s
	Int64⍰	String⍰	String⍰	String⍰	String⍰	String⍰	String⍰	String⍰	
1	19810131	missing	missing	missing	missing	missing	missing	missing	
2	19810228	missing	missing	missing	missing	missing	missing	missing	
3	19810331	missing	missing	missing	missing	missing	missing	missing	
4	19810430	missing	missing	missing	missing	missing	missing	missing	
5	19810531	missing	missing	missing	missing	missing	missing	missing	
6	19810630	missing	missing	missing	missing	missing	missing	missing	

Next step, we try to convert string ratings to numerical ratings. Unlike the second section, we have a different scale here because the increasing data set. We've also create a dictionary of the numerical ratings for later analysis.

In [9]:

```
data=DataFrame()
ratingcount=zeros(22)
for j in 2:size(real_all)[2]
    temp=[]
    for k in 1:size(real_all)[1]
        if ismissing(real_all[j][k])
            append!(temp,-2)
        elseif real_all[j][k]=="AAA"
            append!(temp,23)
            ratingcount[22]+=1
        elseif real_all[j][k]=="AA+"
            append!(temp,22)
            ratingcount[21]+=1
        elseif real_all[j][k]=="AA"
            append!(temp,21)
            ratingcount[20]+=1
        elseif real_all[j][k]=="AA-"
            append!(temp,20)
            ratingcount[19]+=1
        elseif real_all[j][k]=="A+"
            append!(temp,19)
            ratingcount[18]+=1
        elseif real_all[j][k]=="A"
            append!(temp,18)
            ratingcount[17]+=1
        elseif real_all[j][k]=="A-"
            append!(temp,17)
            ratingcount[16]+=1
        elseif real_all[j][k]=="BBB+"
            append!(temp,16)
            ratingcount[15]+=1
        elseif real_all[j][k]=="BBB"
            append!(temp,15)
            ratingcount[14]+=1
        elseif real_all[j][k]=="BBB-"
            append!(temp,14)
            ratingcount[13]+=1
        elseif real_all[j][k]=="BB+"
            append!(temp,13)
            ratingcount[12]+=1
        elseif real_all[j][k]=="BB"
            append!(temp,12)
            ratingcount[11]+=1
        elseif real_all[j][k]=="BB-"
            append!(temp,11)
            ratingcount[10]+=1
        elseif real_all[j][k]=="B+"
            append!(temp,10)
            ratingcount[9]+=1
        elseif real_all[j][k]=="B"
            append!(temp,9)
            ratingcount[8]+=1
        elseif real_all[j][k]=="B-"
            append!(temp,8)
            ratingcount[7]+=1
        elseif real_all[j][k]=="CCC+"
            append!(temp,7)
```

```
            ratingcount[6]+=1
        elseif real_all[j][k]=="CCC"
            append!(temp,6)
            ratingcount[5]+=1
        elseif real_all[j][k]=="CCC-"
            append!(temp,5)
            ratingcount[4]+=1
        elseif real_all[j][k]=="CC"
            append!(temp,4)
            ratingcount[3]+=1
        elseif real_all[j][k]=="C"
            append!(temp,3)
            ratingcount[2]+=1
        elseif real_all[j][k]=="SD"
            append!(temp,2)
            ratingcount[1]+=1
        elseif real_all[j][k]=="D"
            append!(temp,2)
            ratingcount[1]+=1
        elseif real_all[j][k]=="N.M."
            append!(temp,0)
        else
            append!(temp,0)
        end
    end
    data=hcat(data,temp)
end
```

In [10]:

```
ref_list = Dict("AAA"=>23,"AA+"=>22,"AA"=>21,"AA-"=>20,"A+"=>19,"A"=>18,"A-"=>17,
                "BBB+"=>16,"BBB"=>15,"BBB-"=>14,"BB+"=>13,"BB"=>12,"BB-"=>11,"B+"=>10,"B"=>
                "CCC+"=>7,"CCC"=>6,"CCC-"=>5,"CC"=>4,"C"=>3,"D"=>2,
                "NM"=>0,"missing"=>-2)
sort(ref_list)
all_ticks = DataFrame(ticker=ticker, tickername=ticker_name);
```

In [11]:

```
#creating the list of names of rating
sortlist = sort(collect(zip(values(ref_list),keys(ref_list))))
ratinglist = []
for i in 3:length(sortlist)
    ratinglist = vcat(ratinglist, sortlist[i][2])
end
```

In [12]:

```
ratingcounter = DataFrame(ratinglist = ratinglist, ratingcount=ratingcount)
sort(ratingcounter, :ratingcount, rev=true)
```

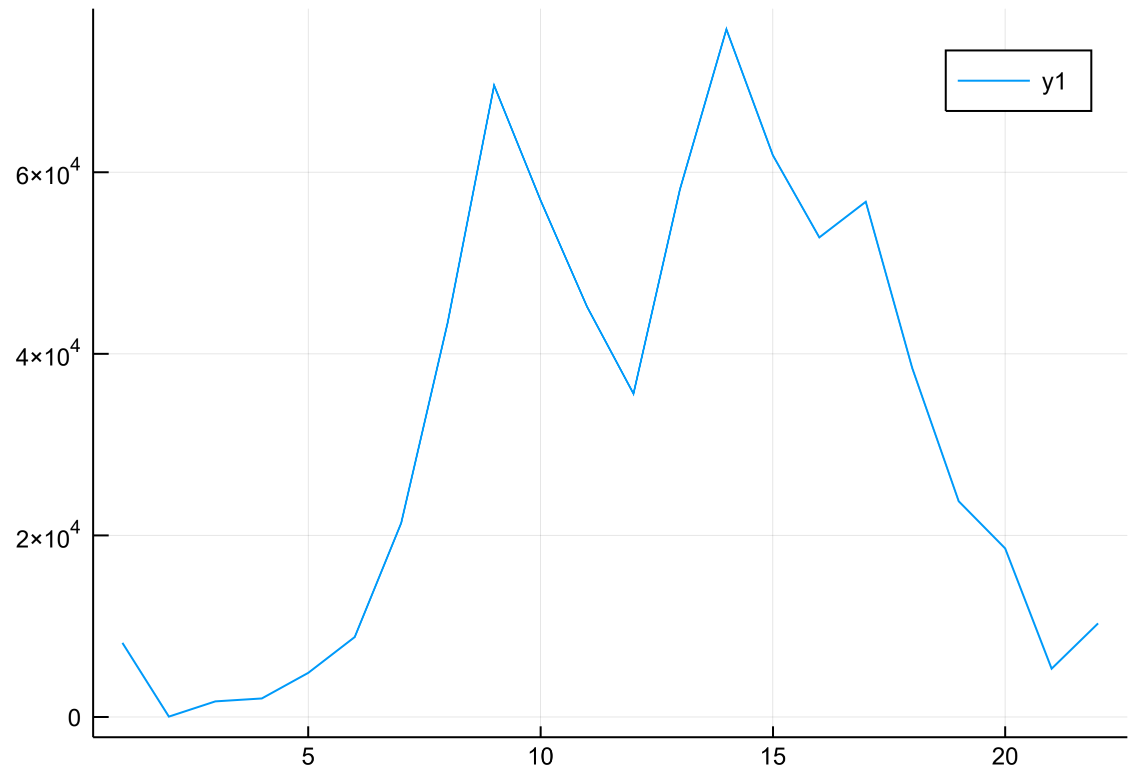
Out[12]:

	ratinglist	ratingcount
	Any	Float64
1	BBB	75748.0000
2	B+	69567.0000
3	BBB+	61873.0000
4	BBB-	58117.0000
5	BB-	56913.0000
6	A	56752.0000
7	A-	52823.0000
8	BB	45194.0000
9	B	43433.0000
10	A+	38430.0000
11	BB+	35605.0000
12	AA-	23763.0000
13	B-	21364.0000
14	AA	18569.0000
15	AAA	10311.0000
16	CCC+	8807.0000
17	D	8164.0000
18	AA+	5321.0000
19	CCC	4863.0000
20	CCC-	2042.0000
21	CC	1720.0000
22	C	42.0000

In [13]:

```
plot(ratingcounter.ratingcount)
```

Out[13]:



We can find the number of each credit rating. The most credit rating in the data set is 75,748, which is BBB, and the least credit rating in the data set is 42, which is C. Then, we plot the distribution of the data set to have a easier check for each credit rating.

In [14]:

```
timeframe = DataFrame(Date=[])
for i in 1:size(real_all)[1]
    timeframe = vcat(timeframe, Dates.DateTime(string(real_all[i,1]),"yyyymmdd"))
end
timeframe = timeframe[2:end];
```

In [15]:

```
size(data)
```

Out[15]:

```
(434, 5947)
```

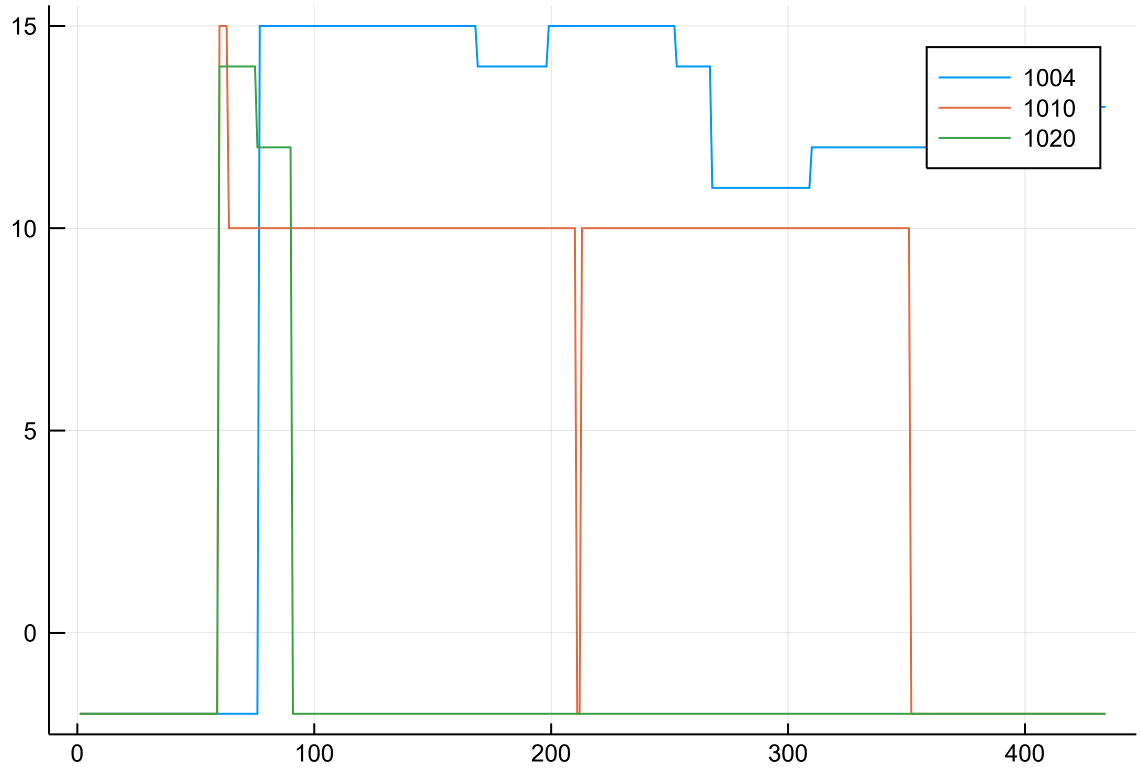
2.2 Rating Change Graph Demonstration

For this part, we have two ways to plot credit ratings by using ticker names and the name of companies. By doing so, we can compare with each other to see how the changes occur for different company. Based on the new dataframe we use, we can have some default data set back, which can be seen in the graph we plot.

In [16]:

```
plot()
start = 1
en_d = 3
plot!([data[i] for i in start:en_d], label = [ticker[k] for k in start:en_d])
```

Out[16]:

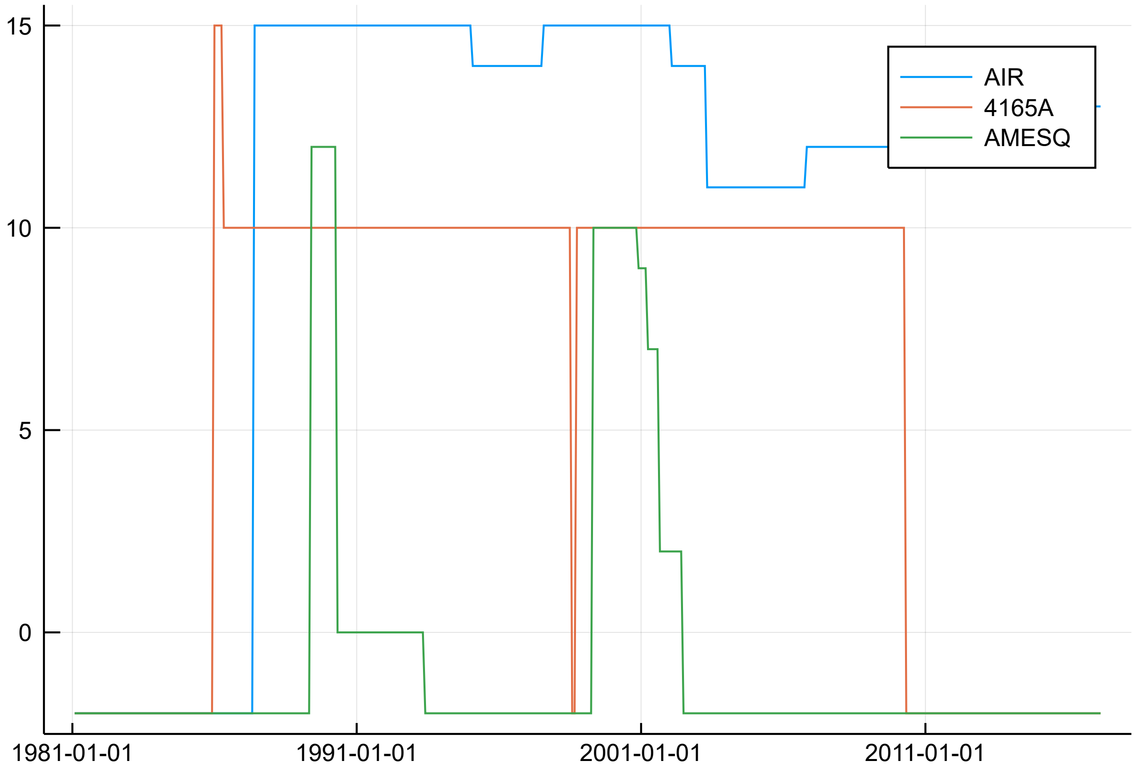


In [17]:

```
#plot using tick
wantick = ["AIR","4165A","AMESQ"]
ranges = []
for i in wantick
    append!(ranges, findall(all_ticks.tickername.==i))
end

plot()
plot!(timeframe,[data[i] for i in ranges],label=[all_ticks.tickername[k] for k in ranges])
```

Out[17]:



2.3 Single Notch Tansitions, Multiple Notch Transitions Percentage

For this section, we will discuss about the single-notch and multi-notch percentage. We can find that the percentage of single notch percentage is 38.70% and the percentage of multi notch percentage 61.30%. Then, we can find the percentage of single and multi notch by using ticker names or companies name for a deeper discussion.

In [18]:

```
sing_mult_notch=DataFrame(single=[],multi=[])
sing_percentage=[]
mult_percentage=[]
for j in 1:size(data)[2]
    notch=[0,0]
    for k in 1:size(data)[1]-1
        if abs(data[k,j]-data[k+1,j])==1
            if data[k,j]!=-1
                notch[1] = notch[1]+1
            end
        elseif abs(data[k,j]-data[k+1,j])>1
            if data[k,j]!=-1
                notch[2] =notch[2]+1
            end
        end
    end
    push!(sing_mult_notch,notch)
    append!(sing_percentage,notch[1]/sum(notch))
    append!(mult_percentage,notch[2]/sum(notch))
end
sing_mult_notch=hcat(sing_mult_notch,sing_percentage)
sing_mult_notch=hcat(sing_mult_notch,mult_percentage)
sing_mult_notch=hcat(sing_mult_notch,ticker_name)
rename!(sing_mult_notch,:x1,:single_percentage)
rename!(sing_mult_notch,:x1_1,:multi_percentage)
rename!(sing_mult_notch,:x1_2,:tickername)
println("single notch percentage: ",sum(sing_mult_notch[1])/(sum(sing_mult_notch[1])+sum(si
println("multi  notch percentage: ",1-sum(sing_mult_notch[1])/(sum(sing_mult_notch[1])+sum(
```

```
single notch percentage: 0.3870
multi  notch percentage: 0.6130
```

In [19]:

```
#dataframe of single_multi_notch percentage
head(sing_mult_notch)
```

Out[19]:

	single	multi	single_percentage	multi_percentage	tickername
	Any	Any	Any	Any	String⍰
1	5	2	0.7143	0.2857	AIR
2	0	5	0.0000	1.0000	4165A
3	0	3	0.0000	1.0000	5548C
4	0	2	0.0000	1.0000	AGS.2
5	3	6	0.3333	0.6667	ALO.2
6	0	2	0.0000	1.0000	UDI.

In [20]:

```
#sorted by single notch percentage
sort(sing_mult_notch, :single_percentage, rev=true)
```

Out[20]:

	single	multi	single_percentage	multi_percentage	tickername
	Any	Any	Any	Any	String⍰
1	14	1	0.9333	0.0667	AKS
2	12	1	0.9231	0.0769	WFC
3	11	1	0.9167	0.0833	LPX
4	11	1	0.9167	0.0833	COT
5	11	1	0.9167	0.0833	TTM
6	10	1	0.9091	0.0909	HSC
7	10	1	0.9091	0.0909	BZH
8	10	1	0.9091	0.0909	FCH
9	9	1	0.9000	0.1000	PNW1
10	9	1	0.9000	0.1000	PYX
11	9	1	0.9000	0.1000	WHR
12	9	1	0.9000	0.1000	RCL
13	8	1	0.8889	0.1111	BA
14	8	1	0.8889	0.1111	AON
15	8	1	0.8889	0.1111	CMCSA
16	8	1	0.8889	0.1111	MAS
17	8	1	0.8889	0.1111	MCK
18	8	1	0.8889	0.1111	BAC
19	8	1	0.8889	0.1111	JWN
20	8	1	0.8889	0.1111	OXY
21	8	1	0.8889	0.1111	PCH
22	8	1	0.8889	0.1111	SNE
23	8	1	0.8889	0.1111	T
24	8	1	0.8889	0.1111	KSS
25	8	1	0.8889	0.1111	8135A
26	7	1	0.8750	0.1250	BK
27	7	1	0.8750	0.1250	BAX
28	7	1	0.8750	0.1250	CRS
29	7	1	0.8750	0.1250	KO
30	7	1	0.8750	0.1250	SO2
⋮	⋮	⋮	⋮	⋮	⋮

In [21]:

```
#sorted by multi notch percentage
sort(sing_mult_notch, :multi_percentage, rev=true)
```

Out[21]:

	single	multi	single_percentage	multi_percentage	tickername
	Any	Any	Any	Any	String⍰
1	0	5	0.0000	1.0000	4165A
2	0	3	0.0000	1.0000	5548C
3	0	2	0.0000	1.0000	AGS.2
4	0	2	0.0000	1.0000	UDI.
5	0	4	0.0000	1.0000	ARXX
6	0	3	0.0000	1.0000	4328B
7	0	2	0.0000	1.0000	AVX
8	0	2	0.0000	1.0000	ACRA.
9	0	2	0.0000	1.0000	ATVI.1
10	0	2	0.0000	1.0000	ASY.2
11	0	2	0.0000	1.0000	AHM.1
12	0	2	0.0000	1.0000	AEIC
13	0	6	0.0000	1.0000	ABF
14	0	2	0.0000	1.0000	ABC1
15	0	4	0.0000	1.0000	AAL.1
16	0	2	0.0000	1.0000	ALX
17	0	5	0.0000	1.0000	APNI
18	0	4	0.0000	1.0000	AT4
19	0	2	0.0000	1.0000	AT1
20	0	3	0.0000	1.0000	5714B
21	0	3	0.0000	1.0000	AMKKQ
22	0	2	0.0000	1.0000	2388B
23	0	2	0.0000	1.0000	ADT.2
24	0	1	0.0000	1.0000	ECOL
25	0	4	0.0000	1.0000	2551A
26	0	2	0.0000	1.0000	AHL.2
27	0	3	0.0000	1.0000	DIVC.1
28	0	2	0.0000	1.0000	AMO.1
29	0	2	0.0000	1.0000	FI.2
30	0	4	0.0000	1.0000	TT.2
⋮	⋮	⋮	⋮	⋮	⋮

In [22]:

```
#number of total single notch and multi notch changes
println("single: ",sum(sing_mult_notch[1]),"\nmulti : ",sum(sing_mult_notch[2]))
```

```
single: 11644
multi : 18443
```

2.4 Transition Probability

In [23]:

```
#Checking the size of data for creating the nested for loop to calculate transition matrix
size(data)
```

Out[23]:

```
(434, 5947)
```

In [24]:

```
trans=zeros(22,22)
for j in 1:size(data)[2]
    for k in 1:size(data)[1]-1
        if (data[k,j]>0) && (data[k+1,j]>0)
            trans[data[k,j]-1,data[k+1,j]-1]=trans[data[k,j]-1,data[k+1,j]-1]+1
        end
    end
end
transframe=DataFrame(trans)
names!(transframe, [Symbol("$i") for i in ratinglist])
```

Out[24]:

	D	C	CC	CCC-	CCC	CCC+	B-	B	
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	
1	7285.0000	0.0000	14.0000	16.0000	21.0000	38.0000	27.0000	24.0000	
2	2.0000	33.0000	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000	
3	213.0000	1.0000	1318.0000	21.0000	23.0000	18.0000	31.0000	5.0000	
4	113.0000	2.0000	68.0000	1749.0000	11.0000	13.0000	4.0000	2.0000	
5	173.0000	4.0000	84.0000	46.0000	4329.0000	49.0000	35.0000	15.0000	
6	120.0000	1.0000	91.0000	99.0000	132.0000	8010.0000	123.0000	41.0000	
7	68.0000	0.0000	73.0000	45.0000	184.0000	283.0000	20041.0000	268.0000	
8	52.0000	0.0000	35.0000	36.0000	65.0000	205.0000	507.0000	41291.0000	
9	17.0000	1.0000	13.0000	10.0000	35.0000	73.0000	205.0000	745.0000	66
10	9.0000	0.0000	4.0000	5.0000	4.0000	18.0000	42.0000	147.0000	
11	1.0000	0.0000	3.0000	5.0000	6.0000	3.0000	17.0000	34.0000	
12	4.0000	0.0000	0.0000	1.0000	5.0000	1.0000	4.0000	14.0000	
13	3.0000	0.0000	0.0000	0.0000	2.0000	1.0000	4.0000	6.0000	
14	3.0000	0.0000	0.0000	0.0000	2.0000	1.0000	0.0000	7.0000	
15	0.0000	0.0000	1.0000	0.0000	0.0000	1.0000	0.0000	3.0000	
16	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	2.0000	
17	2.0000	0.0000	1.0000	1.0000	0.0000	0.0000	0.0000	1.0000	
18	2.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	3.0000	
19	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	3.0000	
20	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	
22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	

In [25]:

```
trans2=zeros(22,22)
for j in 1:22
    trans2[j,:]=trans[j,:]/sum(trans[j,:])
end
transframe2=DataFrame(trans2)
names!(transframe2, [Symbol("$i") for i in ratinglist])
```

Out[25]:

	D	C	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Flo
1	0.9764	0.0000	0.0019	0.0021	0.0028	0.0051	0.0036	0.0032	0.0024	0.0008	0.
2	0.0541	0.8919	0.0000	0.0000	0.0270	0.0000	0.0000	0.0270	0.0000	0.0000	0.
3	0.1300	0.0006	0.8041	0.0128	0.0140	0.0110	0.0189	0.0031	0.0037	0.0012	0.
4	0.0574	0.0010	0.0345	0.8878	0.0056	0.0066	0.0020	0.0010	0.0020	0.0015	0.
5	0.0365	0.0008	0.0177	0.0097	0.9121	0.0103	0.0074	0.0032	0.0006	0.0004	0.
6	0.0139	0.0001	0.0105	0.0114	0.0153	0.9262	0.0142	0.0047	0.0022	0.0005	0.
7	0.0032	0.0000	0.0035	0.0021	0.0087	0.0135	0.9526	0.0127	0.0027	0.0004	0.
8	0.0012	0.0000	0.0008	0.0008	0.0015	0.0048	0.0119	0.9655	0.0115	0.0013	0.
9	0.0002	0.0000	0.0002	0.0001	0.0005	0.0011	0.0030	0.0108	0.9728	0.0089	0.
10	0.0002	0.0000	0.0001	0.0001	0.0001	0.0003	0.0007	0.0026	0.0109	0.9732	0.
11	0.0000	0.0000	0.0001	0.0001	0.0001	0.0001	0.0004	0.0008	0.0031	0.0107	0.
12	0.0001	0.0000	0.0000	0.0000	0.0001	0.0000	0.0001	0.0004	0.0009	0.0037	0.
13	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0002	0.0007	0.
14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0003	0.0002	0.
15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.
16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.
17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.
18	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0000	0.
19	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.
20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
21	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.

We implement two types of transition matrix. One transition matrix shows the exactly time and one shows the probability. We have another dataframe to see the single notch, multi notch, and stationary probability for each credit rating, which enable us to have a more clear understanding for our transition matrix.

In [26]:

```
transum=[]
singsum=[]
for j in 1:size(trans2)[1]
    if j==1
        tempsum=1-(trans2[j,j]+trans2[j,j+1])
        tempsum2=trans2[j,j+1]
    elseif j==size(trans2)[1]
        tempsum=1-(trans2[j,j]+trans2[j,j-1])
        tempsum2=trans2[j,j-1]
    else
        tempsum=1-(trans2[j,j]+trans2[j,j+1]+trans2[j,j-1])
        tempsum2=trans2[j,j+1]+trans2[j,j-1]
    end
    append!(transum,tempsum)
    append!(singsum,tempsum2)
end
```

In [27]:

```
DataFrame(ratinglist=ratinglist,multiprob=transum,singprob=singsum)
```

Out[27]:

	ratinglist	multiprob	singprob
	Any	Any	Any
1	D	0.0236	0.0000
2	C	0.0541	0.0541
3	CC	0.1824	0.0134
4	CCC-	0.0721	0.0401
5	CCC	0.0678	0.0200
6	CCC+	0.0443	0.0295
7	B-	0.0212	0.0262
8	B	0.0112	0.0234
9	B+	0.0075	0.0197
10	BB-	0.0065	0.0203
11	BB	0.0075	0.0205
12	BB+	0.0081	0.0202
13	BBB-	0.0057	0.0153
14	BBB	0.0038	0.0143
15	BBB+	0.0040	0.0153
16	A-	0.0041	0.0150
17	A	0.0037	0.0118
18	A+	0.0041	0.0127
19	AA-	0.0035	0.0144
20	AA	0.0044	0.0118
21	AA+	0.0034	0.0151
22	AAA	0.0030	0.0040

In [28]:

```
#migration of 100 steps starting from AAA and stationary distribution
initial=zeros(22)
initial[22]=1
migration=transpose(initial)*(trans2^100)
M = trans2 - one(trans2)
M[:,22] = ones(22)
stationary=transpose(initial)*inv(M)
DataFrame(ratinglist=ratinglist,migration_100=transpose(migration),stationary=transpose(sta
```

Out[28]:

	ratinglist	migration_100	stationary
	Any	Float64	Float64
1	D	0.0007	0.1244
2	C	0.0000	0.0003
3	CC	0.0001	0.0075
4	CCC-	0.0001	0.0096
5	CCC	0.0002	0.0183
6	CCC+	0.0003	0.0293
7	B-	0.0006	0.0516
8	B	0.0014	0.0762
9	B+	0.0024	0.0954
10	BB-	0.0044	0.0782
11	BB	0.0043	0.0640
12	BB+	0.0023	0.0538
13	BBB-	0.0037	0.0856
14	BBB	0.0095	0.1015
15	BBB+	0.0114	0.0750
16	A-	0.0194	0.0517
17	A	0.0303	0.0408
18	A+	0.0498	0.0191
19	AA-	0.0854	0.0090
20	AA	0.1298	0.0044
21	AA+	0.1255	0.0013
22	AAA	0.5185	0.0029

Last, we combine all credit ratings into 4 categories to have a more simple transition matrix as the following blocks show.

In [29]:

```
#group up A B C D
basket=zeros(4,4)
for j in enumerate([1,2:6,7:15,16:22])
    for k in enumerate([1,2:6,7:15,16:22])
        basket[j[1],k[1]]=sum(trans[j[2],k[2]])
    end
    basket[j[1],:]=basket[j[1],:]/sum(basket[j[1],:])
end
```

In [30]:

```
bask=DataFrame(basket) #D C B A
names!(bask,[:D,:C,:B,:A])
```

Out[30]:

	D	C	B	A
	Float64	Float64	Float64	Float64
1	0.9764	0.0119	0.0115	0.0001
2	0.0364	0.9450	0.0183	0.0003
3	0.0003	0.0024	0.9959	0.0013
4	0.0000	0.0000	0.0046	0.9954

3. Company (1997~2017)

In [31]:

```
raw_data = readtable("raw_data.csv");
size(raw_data)
```

Out[31]:

```
(2387466, 6)
```

In [32]:

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

Out[32]:

```
(503108, 6)
```

In [33]:

```
size(unique(data.gvkey),1)
```

Out[33]:

4984

In [34]:

```
by(data, :gvkey, size)
```

Out[34]:

	gvkey	x1
	Int64⍰	Tuple…
1	1004	(242, 6)
2	1010	(157, 6)
3	1034	(118, 6)
4	1036	(16, 6)
5	1038	(97, 6)
6	1045	(242, 6)
7	1048	(159, 6)
8	1055	(7, 6)
9	1056	(5, 6)
10	1075	(242, 6)
11	1078	(242, 6)
12	1081	(121, 6)
13	1095	(45, 6)
14	1111	(41, 6)
15	1161	(242, 6)
16	1164	(84, 6)
17	1166	(224, 6)
18	1177	(242, 6)
19	1186	(74, 6)
20	1194	(22, 6)
21	1203	(18, 6)
22	1209	(242, 6)
23	1213	(82, 6)
24	1224	(242, 6)
25	1225	(242, 6)
26	1230	(242, 6)
27	1238	(39, 6)
28	1239	(172, 6)
29	1240	(113, 6)
30	1243	(132, 6)
⋮	⋮	⋮

First of all, we can easily find that the data set decreases significantly after adjusting for missing values (from

2,387,466 to 503,108), which is quite reasonable because many companies had no ratings at the beginning.

The data above also shows that using tic symbols 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. To prevent survivor bias/distortion on transition matrix, the inconsistencies are moved.

As the following block shows, there are only 529 companies have complete data from January 1997 to February 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 [35]:

```
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[35]:

```
529
```

In [36]:

```
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[36]:

```
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 [37]:

```
unique(selected_data.datadate)
```

Out[37]:

```
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 [38]:

```
unique(selected_data.splticrm)
```

Out[38]:

```
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 [39]:

```
sort!(selected_data, :datadate)
```

Out[39]:

	gvkey	splticrm	datadate	gsector	conm	tic
	Int64⍰	String⍰	Int64⍰	Int64⍰	String⍰	String⍰
1	220940	AAA	19970131	50	ORANGE	ORAN
2	212782	BB	19970131	35	FRESENIUS MEDICAL CARE AG&CO	FMS
3	210216	AA	19970131	50	TELSTRA CORP LTD	TLSYY
4	145348	A-	19970131	55	PPL ELECTRIC UTILITIES CORP	PPL2
5	104831	AA+	19970131	50	SPARK NEW ZEALAND LTD	SPKKY
6	100590	AA+	19970131	55	E.ON SE	EONGY
7	100338	AA-	19970131	20	RELX PLC	RELX
8	100243	A	19970131	15	AMCOR LTD	AMCRY
9	100165	A-	19970131	10	SANTOS LTD	SSLZY
10	66624	B+	19970131	55	TUCSON ELECTRIC POWER CO	UNS1
11	65298	A+	19970131	55	MIDAMERICAN ENERGY CO	MEC1
12	65095	AA+	19970131	55	WISCONSIN PUBLIC SERVICE CP	WPS1
13	65090	A+	19970131	55	SAN DIEGO GAS & ELECTRIC CO	SRE4
14	65089	BBB+	19970131	55	DETROIT EDISON CO	DTE1
15	64389	B+	19970131	15	SILGAN HOLDINGS INC	SLGN
16	64166	BB	19970131	35	QUEST DIAGNOSTICS INC	DGX
17	63759	A+	19970131	40	SCOR SE	SCRYY
18	63639	B+	19970131	40	OCWEN FINANCIAL CORP	OCN
19	63605	B+	19970131	55	CALPINE CORP	CPN
20	63477	A-	19970131	20	BAE SYSTEMS PLC	BAESY
21	62374	B+	19970131	60	IRON MOUNTAIN INC	IRM
22	61739	A	19970131	40	HARTFORD FINANCIAL SERVICES	HIG
23	61409	A-	19970131	10	DIAMOND OFFSHRE DRILLING INC	DO
24	61408	A-	19970131	40	HANOVER INSURANCE GROUP INC	THG
25	61338	BB-	19970131	20	CENVEO INC	CVOVQ
26	61034	BB	19970131	10	TEEKAY CORP	TK
27	60900	B	19970131	50	DISH NETWORK CORP	DISH
28	60800	BB-	19970131	50	SINCLAIR BROADCAST GP -CL A	SBGI
29	31846	BBB	19970131	25	DARDEN RESTAURANTS INC	DRI
30	31596	BBB	19970131	55	COMMONWEALTH EDISON CO	UCM1
⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [40]:

```
sort!(selected_data, :splticrm)
```

Out[40]:

	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 [41]:

```
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
rating_float
```

Out[41]:

```
128018×1 Array{Int64,2}:
 16
 16
 16
 16
 16
```

```
 16
 16
 16
 16
 16
 16
 16
 16
  ⋮
  1
  1
  1
  1
  1
  1
  1
  1
  1
  1
  1
  1
```

In [42]:

```
delete!(selected_data,:splticrm)
```

Out[42]:

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

In [43]:

```
rating_float = convert(DataFrame, rating_float);
rename!(rating_float, :x1 => :rating)
```

Out[43]:

	rating
	Int64
1	16
2	16
3	16
4	16
5	16
6	16
7	16
8	16
9	16
10	16
11	16
12	16
13	16
14	16
15	16
16	16
17	16
18	16
19	16
20	16
21	16
22	16
23	16
24	16
25	16
26	16
27	16
28	16
29	16
30	16
⋮	⋮

In [44]:

```
selected_data = hcat(selected_data, rating_float)
```

Out[44]:

	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 [45]:

```
sort!(selected_data,:tic)
```

Out[45]:

	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
⋮	⋮	⋮	⋮	⋮	⋮	⋮

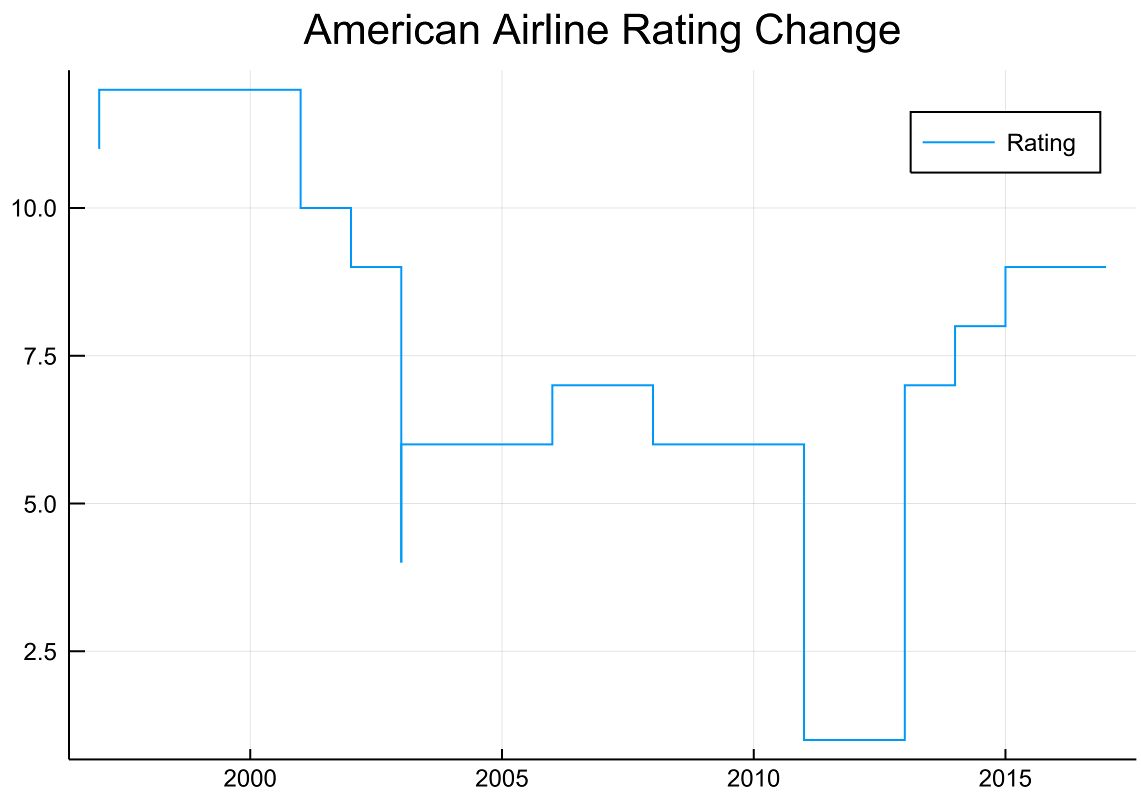
3.2 Rating Change Graph Demonstration

Demonstration of a company's rating changes over the year

In [46]:

```
# Select American Airline as the example
AAL = DataFrame(selected_data[selected_data.tic .=== "AAL",:]);
sort!(AAL,:datadate)
date = trunc.(Int, AAL.datadate / 10000)
plot(date, AAL.rating, label = "Rating", title = "American Airline Rating Change")
```

Out[46]:



In [47]:

```
savefig("AAL_rating.png")
```

3.3 Single Notch Tansitions, Multiple Notch Transitions Percentage

Calculate the percentage of single notch change and multiple notch change

3.3.1 Per Company (transitioning between months)

In [48]:

```
part1_transition_df_with_year_transition = DataFrame(company = String[], singlenotch = Int[
part1_transition_df_without_year_transition = DataFrame(company = String[], singlenotch = I
multiple_notch_counter = 0;
single_notch_counter = 0;
```

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

In [49]:

```
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_counter)
    name = subgroup.tic[1]
    push!(part1_transition_df_with_year_transition, [name, single_notch_counter, multiple_r
end
part1_transition_df_with_year_transition
```

Out[49]:

	company	singlenotch	multinotch	perctmultinotch
	String	Int64	Int64	Float64
1	0141A	6	3	0.3333
2	0176A	2	0	0.0000
3	0191A	4	6	0.6000
4	1231B	7	3	0.3000
5	3NSRGY	0	1	1.0000
6	5672A	3	0	0.0000
7	5946B	6	2	0.2500
8	6120B	5	2	0.2857
9	8135A	8	0	0.0000
10	AAL	6	6	0.5000
11	ABT	0	3	1.0000
12	ADM	2	0	0.0000
13	ADP	1	1	0.5000
14	AEE	6	1	0.1429
15	AEG	2	1	0.3333
16	AEP1	4	0	0.0000
17	AEP12	4	1	0.2000
18	AEP13	5	1	0.1667
19	AEP2	5	0	0.0000
20	AEP4	5	0	0.0000

	company	singlenotch	multinotch	perctmultinotch
	String	Int64	Int64	Float64
21	AEP5	4	0	0.0000
22	AES	6	0	0.0000
23	AES3	4	5	0.5556
24	AET	9	0	0.0000
25	AFG	3	1	0.2500
26	AGC1	1	5	0.8333
27	AGCO	2	0	0.0000
28	AGU	0	0	NaN
29	AIG	4	1	0.2000
30	AIR	4	1	0.2000
⋮	⋮	⋮	⋮	⋮

In [50]:

```
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 [51]:

```
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)
```

Out[51]:

	gvkey	datadate	gsector	conm	tic	rating	year
	Int64⍰	Int64⍰	Int64⍰	String⍰	String⍰	Int64	Int64
1	2316	20061031	15	HEXION INC	0141A	7	2006
2	2316	20061130	15	HEXION INC	0141A	7	2006
3	2316	20061231	15	HEXION INC	0141A	7	2006
4	2316	20070131	15	HEXION INC	0141A	7	2007
5	2316	20070228	15	HEXION INC	0141A	7	2007
6	2316	20070331	15	HEXION INC	0141A	7	2007
7	2316	20070430	15	HEXION INC	0141A	7	2007
8	2316	20070531	15	HEXION INC	0141A	7	2007
9	2316	20070630	15	HEXION INC	0141A	7	2007
10	2316	20070731	15	HEXION INC	0141A	7	2007
11	2316	20070831	15	HEXION INC	0141A	7	2007
12	2316	20070930	15	HEXION INC	0141A	7	2007
13	2316	20071031	15	HEXION INC	0141A	7	2007
14	2316	20071130	15	HEXION INC	0141A	7	2007
15	2316	20071231	15	HEXION INC	0141A	7	2007
16	2316	20080131	15	HEXION INC	0141A	7	2008
17	2316	20080229	15	HEXION INC	0141A	7	2008
18	2316	20080331	15	HEXION INC	0141A	7	2008
19	2316	20080430	15	HEXION INC	0141A	7	2008
20	2316	20080531	15	HEXION INC	0141A	7	2008
21	2316	20080630	15	HEXION INC	0141A	7	2008
22	2316	20080731	15	HEXION INC	0141A	7	2008
23	2316	20080831	15	HEXION INC	0141A	7	2008
24	2316	20080930	15	HEXION INC	0141A	7	2008
25	2316	20081031	15	HEXION INC	0141A	7	2008
26	2316	20040831	15	HEXION INC	0141A	8	2004
27	2316	20040930	15	HEXION INC	0141A	8	2004
28	2316	20041031	15	HEXION INC	0141A	8	2004
29	2316	20041130	15	HEXION INC	0141A	8	2004

	gvkey	datadate	gsector	conm	tic	rating	year
	Int64⍰	Int64⍰	Int64⍰	String⍰	String⍰	Int64	Int64
30	2316	20041231	15	HEXION INC	0141A	8	2004
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [52]:

```
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_year
            multiple_notch_counter = multiple_notch_counter + 1
        elseif abs(temp_rating[i] - previous) == 1 && temp_year[i] == previous_year
            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_counter)
    name = subgroup.tic[1]
    push!(part1_transition_df_without_year_transition, [name, single_notch_counter, multipl
end
part1_transition_df_without_year_transition
```

Out[52]:

	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
8	6120B	5	2	0.2857
9	8135A	8	0	0.0000
10	AAL	6	6	0.5000
11	ABT	0	2	1.0000
12	ADM	1	0	0.0000
13	ADP	1	1	0.5000
14	AEE	5	0	0.0000
15	AEG	2	1	0.3333
16	AEP1	4	0	0.0000
17	AEP12	4	1	0.2000
18	AEP13	5	1	0.1667

	company	singlenotch	multinotch	perctmultinotch
	String	Int64	Int64	Float64
19	AEP2	5	0	0.0000
20	AEP4	5	0	0.0000
21	AEP5	4	0	0.0000
22	AES	5	0	0.0000
23	AES3	4	4	0.5000
24	AET	9	0	0.0000
25	AFG	3	1	0.2500
26	AGC1	1	5	0.8333
27	AGCO	2	0	0.0000
28	AGU	0	0	NaN
29	AIG	3	1	0.2500
30	AIR	3	1	0.2500
⋮	⋮	⋮	⋮	⋮

In [53]:

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

```
1963
468
```

3.3.2 Over time, Per year

In [54]:

```
part2_transition_df_without_year_transition = DataFrame(year = Int[1997, 1998, 1999, 2000, 
part2_transition_df_with_year_transition = DataFrame(year = Int[1997, 1998, 1999, 2000, 20(
overtime_multinotch_counter = 0;
overtime_singlenotch_counter = 0;
```

For easier processing, create a column of the year

In [55]:

```
# Still use tic as the separating factor, then slowly accumulate the counter
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 + 1
            elseif abs(temp_rating[i] - previous) == 1
                overtime_singlenotch_counter = overtime_singlenotch_counter + 1
            end
            previous = temp_rating[i]
        end
        part2_transition_df_without_year_transition[counter,2] = part2_transition_df_withou
        part2_transition_df_without_year_transition[counter,3] = part2_transition_df_withou
        counter = counter + 1
    end
end
part2_transition_df_without_year_transition.perctmultinotch = part2_transition_df_without_y
part2_transition_df_without_year_transition
```

Out[55]:

	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

	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
17	2013	104	11	0.0957
18	2014	80	12	0.1304
19	2015	101	13	0.1140
20	2016	101	22	0.1789
21	2017	10	0	0.0000

In [56]:

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

```
1963
468
```

In [57]:

```
# Testing ground
for subgroup in groupby(part2_data, :gvkey)
    subgroup = sort(subgroup,:datadate)
    A = subgroup[subgroup.year .=== 2017,:]
    println(A)
    break
end
```

```
2×7 DataFrame
│ Row │ gvkey  │ datadate │ gsector │ conm        │ tic     │ rating │ year
│
│     │ Int64⍰ │ Int64⍰   │ Int64⍰  │ String⍰     │ String⍰ │ Int64  │ In
t64 │
├─────┼────────┼──────────┼─────────┼─────────────┼─────────┼────────┼──────
┤
│ 1   │ 2316   │ 20170131 │ 15      │ HEXION INC │ 0141A   │ 5      │ 2017
 │
│ 2   │ 2316   │ 20170228 │ 15      │ HEXION INC │ 0141A   │ 5      │ 2017
 │
```

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 [58]:

```
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 + 1
            elseif abs(previous - previous_year) == 1
                overtime_singlenotch_counter = overtime_singlenotch_counter + 1
            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 + 1
            elseif abs(temp_rating[i] - previous) == 1
                overtime_singlenotch_counter = overtime_singlenotch_counter + 1
            end
            previous = temp_rating[i]
        end
        part2_transition_df_with_year_transition[counter,2] = part2_transition_df_with_year
        part2_transition_df_with_year_transition[counter,3] = part2_transition_df_with_year
        counter = counter + 1
        init_flag = 1
    end
end
part2_transition_df_with_year_transition.perctmultinotch = part2_transition_df_with_year_tr
part2_transition_df_with_year_transition
```

Out[58]:

	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
1	1997	84	9	0.0968
2	1998	98	17	0.1478
3	1999	94	24	0.2034
4	2000	81	36	0.3077
5	2001	98	40	0.2899
6	2002	126	40	0.2410
7	2003	118	31	0.2081

	year	singlenotch	multinotch	perctmultinotch
	Int64	Int64	Int64	Float64
8	2004	79	14	0.1505
9	2005	107	16	0.1301
10	2006	115	21	0.1544
11	2007	112	31	0.2168
12	2008	123	40	0.2454
13	2009	126	59	0.3189
14	2010	103	25	0.1953
15	2011	121	18	0.1295
16	2012	96	19	0.1652
17	2013	108	12	0.1000
18	2014	82	12	0.1277
19	2015	110	13	0.1057
20	2016	104	23	0.1811
21	2017	14	2	0.1250

In [59]:

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

```
2099
502
```

Note that the sum checks for 3.3.1 and 3.3.2 (without year and with year transition) match perfectly

3.4 Transition Probability

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 [60]:

```
transition_matrix_array = [];
```

In [61]:

```
transition_data = selected_data;
temp_month = trunc.(Int, (transition_data.datadate - (trunc.(Int, (transition_data.datadate
month = DataFrame();
month = hcat(year, temp_month);
rename!(month, :x1 => :month);
transition_data = hcat(transition_data, month)
```

Out[61]:

	gvkey	datadate	gsector	conm	tic	rating	year	month
	Int64⍰	Int64⍰	Int64⍰	String⍰	String⍰	Int64	Int64	Int64
1	2316	20061031	15	HEXION INC	0141A	7	2006	10
2	2316	20061130	15	HEXION INC	0141A	7	2006	11
3	2316	20061231	15	HEXION INC	0141A	7	2006	12
4	2316	20070131	15	HEXION INC	0141A	7	2007	1
5	2316	20070228	15	HEXION INC	0141A	7	2007	2
6	2316	20070331	15	HEXION INC	0141A	7	2007	3
7	2316	20070430	15	HEXION INC	0141A	7	2007	4
8	2316	20070531	15	HEXION INC	0141A	7	2007	5
9	2316	20070630	15	HEXION INC	0141A	7	2007	6
10	2316	20070731	15	HEXION INC	0141A	7	2007	7
11	2316	20070831	15	HEXION INC	0141A	7	2007	8
12	2316	20070930	15	HEXION INC	0141A	7	2007	9
13	2316	20071031	15	HEXION INC	0141A	7	2007	10
14	2316	20071130	15	HEXION INC	0141A	7	2007	11
15	2316	20071231	15	HEXION INC	0141A	7	2007	12
16	2316	20080131	15	HEXION INC	0141A	7	2008	1
17	2316	20080229	15	HEXION INC	0141A	7	2008	2
18	2316	20080331	15	HEXION INC	0141A	7	2008	3
19	2316	20080430	15	HEXION INC	0141A	7	2008	4
20	2316	20080531	15	HEXION INC	0141A	7	2008	5
21	2316	20080630	15	HEXION INC	0141A	7	2008	6
22	2316	20080731	15	HEXION INC	0141A	7	2008	7
23	2316	20080831	15	HEXION INC	0141A	7	2008	8
24	2316	20080930	15	HEXION INC	0141A	7	2008	9
25	2316	20081031	15	HEXION INC	0141A	7	2008	10
26	2316	20040831	15	HEXION INC	0141A	8	2004	8
27	2316	20040930	15	HEXION INC	0141A	8	2004	9
28	2316	20041031	15	HEXION INC	0141A	8	2004	10

	gvkey	datadate	gsector	conm	tic	rating	year	month
	Int64⍰	Int64⍰	Int64⍰	String⍰	String⍰	Int64	Int64	Int64
29	2316	20041130	15	HEXION INC	0141A	8	2004	11
30	2316	20041231	15	HEXION INC	0141A	8	2004	12
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

In [62]:

```
# 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 => Symbol("CC"), :x1_2 =
```

In [63]:

```
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("CC"), :x1_2 =>
    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[63]:

21

In [64]:

```
total_transition_matrix
```

Out[64]:

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

In [65]:

```
transition_matrix_array[7]
```

Out[65]:

	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	Int6
1	30	0	0	0	1	0	0	0	0	0	0	0	0	
2	1	1	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	
4	0	0	0	4	0	2	0	0	0	0	0	0	0	
5	0	0	0	0	19	1	0	0	0	0	0	0	0	
6	0	0	0	2	1	40	1	0	1	0	0	0	0	
7	0	0	1	0	1	0	46	2	0	0	0	0	0	
8	0	0	0	0	0	0	4	142	2	0	0	0	0	
9	0	0	0	0	0	2	1	5	213	4	0	0	0	
10	0	0	0	0	0	0	1	2	5	237	1	0	1	
11	0	0	0	0	0	0	1	0	1	3	262	3	0	
12	0	0	0	0	0	0	0	0	1	1	3	484	10	
13	0	0	0	0	0	0	0	0	0	1	1	18	1021	
14	0	0	0	0	0	0	0	0	0	0	0	1	10	63
15	0	0	0	0	0	0	0	0	0	0	0	1	1	
16	0	0	0	0	0	0	0	0	0	0	0	0	0	
17	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	
19	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	
21	0	0	0	0	0	0	0	0	0	0	0	0	0	

In [66]:

```
# Testing ground
result = 0
transition = []
for i in 1:21
    for row in 1:size(transition_matrix_array[i],1)
        for column in 1:size(transition_matrix_array[i],1)
            if row != column
                result += transition_matrix_array[i][row,column]
            end
        end
    end
    push!(transition, result)
    result = 0
end
transition
```

Out[66]:

```
21-element Array{Any,1}:
  93
 107
 105
 108
 123
 155
 142
  86
 119
 123
 139
 151
 163
 121
 133
 109
 115
  92
 114
 123
  10
```

The result above matches the transition numbers without cross-year transitions found in prvious section

In [67]:

```
total_transition_probability_matrix = DataFrame(x1 = [], x1_1 = [], x1_2 = [], x1_3 = [], x
rename!(total_transition_probability_matrix, Dict(:x1 => Symbol("D/SD"), :x1_1 => Symbol("C
for row = 1:size(total_transition_matrix,1)
    temp_row = convert(Array, total_transition_matrix[row,:]) / sum(convert(Array, total_tr
    push!(total_transition_probability_matrix, temp_row)
end
total_transition_probability_matrix
```

Out[67]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	BB+	BBI
	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Ar
1	0.9266	0.0000	0.0000	0.0092	0.0183	0.0000	0.0321	0.0000	0.0046	0.0046	0.0000	0.004
2	0.1282	0.7179	0.0256	0.0256	0.0513	0.0256	0.0256	0.0000	0.0000	0.0000	0.0000	0.000
3	0.0137	0.0274	0.9315	0.0000	0.0137	0.0137	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
4	0.0252	0.0420	0.0000	0.8487	0.0420	0.0420	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
5	0.0048	0.0048	0.0048	0.0145	0.9229	0.0386	0.0096	0.0000	0.0000	0.0000	0.0000	0.000
6	0.0017	0.0009	0.0000	0.0061	0.0165	0.9455	0.0208	0.0061	0.0017	0.0000	0.0000	0.000
7	0.0004	0.0009	0.0004	0.0009	0.0013	0.0125	0.9633	0.0173	0.0030	0.0000	0.0000	0.000
8	0.0000	0.0003	0.0000	0.0003	0.0007	0.0038	0.0177	0.9570	0.0173	0.0024	0.0003	0.000
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0013	0.0018	0.0139	0.9663	0.0139	0.0029	0.000
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0002	0.0004	0.0021	0.0107	0.9701	0.0138	0.002
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0008	0.0008	0.0033	0.0097	0.9683	0.014
12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0003	0.0027	0.0065	0.978
13	0.0001	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0003	0.0006	0.007
14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001	0.0002	0.001
15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.000
16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0000	0.0000	0.0000	0.0001	0.000
17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
18	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
19	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
21	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000

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

In [68]:

```
transition_probability_matrix_array = []
for year = 1:size(transition_matrix_array,1)
    temp_transition_probability_matrix = DataFrame(x1 = [], x1_1 = [], x1_2 = [], x1_3 = []
    rename!(temp_transition_probability_matrix, Dict(:x1 => Symbol("D/SD"), :x1_1 => Symbol
    for row = 1:size(transition_matrix_array[year],1)
        temp_row = convert(Array, transition_matrix_array[year][row,:]) / sum(convert(Array
        if sum(convert(Array, transition_matrix_array[year][row,:])) == 0
            temp_row = Array([0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.0000,0.000
        end
        push!(temp_transition_probability_matrix, temp_row)
    end
    push!(transition_probability_matrix_array, temp_transition_probability_matrix)
end
```

In [69]:

```
transition_probability_matrix_array[6]
```

Out[69]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	BB+	BBI
	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	An
1	0.8333	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1667	0.0000	0.000
2	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
3	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
4	0.2500	0.0000	0.0000	0.7500	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
5	0.0000	0.0000	0.0000	0.0000	0.7500	0.0000	0.2500	0.0000	0.0000	0.0000	0.0000	0.000
6	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
7	0.0143	0.0000	0.0000	0.0143	0.0000	0.0286	0.9286	0.0000	0.0143	0.0000	0.0000	0.000
8	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0253	0.9620	0.0127	0.0000	0.0000	0.000
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0105	0.9895	0.0000	0.0000	0.000
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0038	0.0000	0.0038	0.0226	0.9547	0.0151	0.000
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0040	0.0202	0.9676	0.008
12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0041	0.0104	0.981
13	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0021	0.0011	0.0032	0.0000	0.009
14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0013	0.0013	0.002
15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
18	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
19	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000
21	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.000

3.5 Stationary Distribution

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 [70]:

```
M = convert(Array, total_transition_probability_matrix) - Matrix{Float64}(I, 21, 21);
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 0 0 0] * inv(M))
rename!(stationary_distribution, Dict(:x1 => Symbol("D/SD"), :x2 => Symbol("CC"), :x3 => Sy
```

Out[70]:

	D/SD	CC	CCC-	CCC	CCC+	B-	B	B+	BB-	BB	E
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Floa
1	0.0067	0.0015	0.0020	0.0041	0.0148	0.0372	0.0688	0.0634	0.0762	0.0802	0.0

4. Sector

For this section, we will discuss about the single and multi notch and transition matrix based on the sector. The reason why we use sector is because there might be possibility that two companies share the same ticker name. We will use the same data file as the second section does. There are totally 11 unique sector based on the data set we use.

4.1 Data Processing

In [71]:

```
raw_data = readtable("data.csv");
data = dropmissing(raw_data);
```

In [72]:

```
sector = unique(data.gsector)
print("Number of Unique Sector: ", size(sector, 1), "\n")
print("Unique Sector: ", "\n", sort(sector))
```

```
Number of Unique Sector: 11
Unique Sector:
Union{Missing, Int64}[10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60]
```

In [73]:

```
sector_10 = data[data[:4] .== 10,:]
sector_15 = data[data[:4] .== 15,:]
sector_20 = data[data[:4] .== 20,:]
sector_25 = data[data[:4] .== 25,:]
sector_30 = data[data[:4] .== 30,:]
sector_35 = data[data[:4] .== 35,:]
sector_40 = data[data[:4] .== 40,:]
sector_45 = data[data[:4] .== 45,:]
sector_50 = data[data[:4] .== 50,:]
sector_55 = data[data[:4] .== 55,:]
sector_60 = data[data[:4] .== 60,:]

sector_dict = Dict("sector_10"=>sector_10,"sector_15"=>sector_15,"sector_20"=>sector_20,"se
```

Out[73]:

```
Dict{String,DataFrame} with 11 entries:
  "sector_10" => 60689×6 DataFrame. Omitted printing of 1 columns…
  "sector_35" => 39882×6 DataFrame. Omitted printing of 1 columns…
  "sector_20" => 93300×6 DataFrame…
  "sector_30" => 42590×6 DataFrame. Omitted printing of 1 columns…
  "sector_60" => 17543×6 DataFrame. Omitted printing of 1 columns…
  "sector_55" => 85314×6 DataFrame. Omitted printing of 1 columns…
  "sector_15" => 61993×6 DataFrame. Omitted printing of 1 columns…
  "sector_50" => 41973×6 DataFrame. Omitted printing of 2 columns…
  "sector_25" => 109659×6 DataFrame. Omitted printing of 1 columns…
  "sector_45" => 41830×6 DataFrame…
  "sector_40" => 106484×6 DataFrame. Omitted printing of 1 columns…
```

In [74]:

```
head(sector_dict["sector_60"])
```

Out[74]:

	gvkey	splticrm	datadate	gsector	conm	tic
	Int64⍰	String⍰	Int64⍰	Int64⍰	String⍰	String⍰
1	1257	BB-	19851231	60	ALEXANDER'S INC	ALX
2	1257	BB-	19860131	60	ALEXANDER'S INC	ALX
3	1257	BB-	19860228	60	ALEXANDER'S INC	ALX
4	1257	BB-	19860331	60	ALEXANDER'S INC	ALX
5	1257	BB-	19860430	60	ALEXANDER'S INC	ALX
6	1257	BB-	19860531	60	ALEXANDER'S INC	ALX

Here, we group the data by 11 sectors and encode the string credit ratings into numerical credit ratings.

In [75]:

```
sector_data_dict = Dict()
for sector in ["sector_10","sector_15","sector_20","sector_25", "sector_30","sector_35","se
    sector_data = sector_dict[sector]
    #take out the gvkey
    gvkey=unique(sector_data[:1])
    #join all companies into dataframe
    all_data=DataFrame(datadate=0)
    for itr in enumerate(gvkey)
        #println(itr[2])
        temp1=sector_data[sector_data[:1].==itr[2],:][2:3]
        all_data=join(all_data,temp1,on= :datadate, kind= :outer,makeunique = true)
    end
    key = sector*"_data"
    sector_data_dict[key] = all_data[2:end,:]
    colnames = vcat(["datadate"],gvkey)
    names!(sector_data_dict[key],Symbol.(colnames))
end
```

In [76]:

```
head(sector_data_dict["sector_60_data"])
```

Out[76]:

	datadate	1257	4605	4842	5149	5543	5862	7063	8363	86
	Int64⍰	String⍰	String⍰	String⍰	String⍰	String⍰	String⍰	String⍰	String⍰	String
1	19851231	BB-	missing	missing	B+	missing	A	A-	missing	BBI
2	19860131	BB-	missing	missing	B+	missing	A	A-	missing	BBI
3	19860228	BB-	missing	missing	B+	missing	A	A-	missing	BBI
4	19860331	BB-	A	missing	B+	missing	A	A-	missing	BBI
5	19860430	BB-	A	missing	B+	missing	A	A-	missing	BBI
6	19860531	BB-	A	missing	B+	missing	A	A-	missing	BBI

In [77]:

```
numerical_data_dict = Dict()
for sector in ["sector_10","sector_15","sector_20","sector_25", "sector_30","sector_35","se
    key = sector*"_data"
    sector_data = sector_data_dict[key]
    data=DataFrame()
    for j in 2:size(sector_data)[2]
        temp=[]
        for k in 1:size(sector_data)[1]
            if ismissing(sector_data[j][k])
                append!(temp,-2)
            elseif sector_data[j][k]=="AAA"
                append!(temp,23)
            elseif sector_data[j][k]=="AA+"
                append!(temp,22)
            elseif sector_data[j][k]=="AA"
                append!(temp,21)
            elseif sector_data[j][k]=="AA-"
                append!(temp,20)
            elseif sector_data[j][k]=="A+"
                append!(temp,19)
            elseif sector_data[j][k]=="A"
                append!(temp,18)
            elseif sector_data[j][k]=="A-"
                append!(temp,17)
            elseif sector_data[j][k]=="BBB+"
                append!(temp,16)
            elseif sector_data[j][k]=="BBB"
                append!(temp,15)
            elseif sector_data[j][k]=="BBB-"
                append!(temp,14)
            elseif sector_data[j][k]=="BB+"
                append!(temp,13)
            elseif sector_data[j][k]=="BB"
                append!(temp,12)
            elseif sector_data[j][k]=="BB-"
                append!(temp,11)
            elseif sector_data[j][k]=="B+"
                append!(temp,10)
            elseif sector_data[j][k]=="B"
                append!(temp,9)
            elseif sector_data[j][k]=="B-"
                append!(temp,8)
            elseif sector_data[j][k]=="CCC+"
                append!(temp,7)
            elseif sector_data[j][k]=="CCC"
                append!(temp,6)
            elseif sector_data[j][k]=="CCC-"
                append!(temp,5)
            elseif sector_data[j][k]=="CC"
                append!(temp,4)
            elseif sector_data[j][k]=="C"
                append!(temp,3)
            elseif sector_data[j][k]=="SD"
                append!(temp,2)
            elseif sector_data[j][k]=="D"
                append!(temp,2)
            elseif sector_data[j][k]=="N.M."
```

```
                append!(temp, 0)
            else
                append!(temp, 0)
            end

        end

        data=hcat(data,temp,makeunique = true)
    end
    sector_data = sector_dict[sector]
    gvkey=unique(sector_data[:1])
    names!(data,Symbol.(gvkey))
    key_new = sector*"_num"
    numerical_data_dict[key_new] = data
end
```

In [78]:

```
head(numerical_data_dict["sector_60_num"])
```

Out[78]:

	1257	4605	4842	5149	5543	5862	7063	8363	8692	8824	10096	10894	11220	11301
	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any	Any
1	11	-2	-2	10	-2	18	17	-2	16	2	-2	-2	-2	-2
2	11	-2	-2	10	-2	18	17	-2	16	2	-2	-2	-2	-2
3	11	-2	-2	10	-2	18	17	-2	16	2	-2	-2	-2	-2
4	11	18	-2	10	-2	18	17	-2	16	2	-2	-2	-2	-2
5	11	18	-2	10	-2	18	17	-2	16	2	-2	-2	-2	-2
6	11	18	-2	10	-2	18	17	-2	16	2	-2	-2	-2	-2

4.2 Single Notch Tansitions, Multiple Notch Transitions Percentage

In [79]:

```
notch_dict = Dict()
for key in ["sector_10_num","sector_15_num","sector_20_num","sector_25_num","sector_30_num"
    data = numerical_data_dict[key]
    sing_mult_notch=DataFrame(single=[],multi=[])
    sing_percentage=[]
    mult_percentage=[]
    for j in 1:size(data)[2]
        notch=[0,0]
        for k in 1:size(data)[1]-1
            if abs(data[k,j]-data[k+1,j])==1
                if data[k,j]!=-2 # missing = -2
                    notch[1] = notch[1]+1
                end
            elseif abs(data[k,j]-data[k+1,j])>1
                if data[k,j]!=-2 # missing = -2
                    notch[2] =notch[2]+1
                end
            end
        end
        push!(sing_mult_notch,notch)
        append!(sing_percentage,notch[1]/sum(notch))
        append!(mult_percentage,notch[2]/sum(notch))
    end
    sing_mult_notch=hcat(sing_mult_notch,sing_percentage)
    sing_mult_notch=hcat(sing_mult_notch,mult_percentage,makeunique=true)
    rename!(sing_mult_notch,:x1,:single_percentage)
    rename!(sing_mult_notch,:x1_1,:multi_percentage)
    notch_dict[key] = sing_mult_notch
    print(key,"\n")
    println("single notch percentage: ",sum(sing_mult_notch[1])/(sum(sing_mult_notch[1])+su
    println("multi  notch percentage: ",1-sum(sing_mult_notch[1])/(sum(sing_mult_notch[1])+
end
```

```
sector_10_num
single notch percentage: 0.4694
multi  notch percentage: 0.5306
sector_15_num
single notch percentage: 0.4305
multi  notch percentage: 0.5695
sector_20_num
single notch percentage: 0.4234
multi  notch percentage: 0.5766
sector_25_num
single notch percentage: 0.4699
multi  notch percentage: 0.5301
sector_30_num
single notch percentage: 0.4286
multi  notch percentage: 0.5714
sector_35_num
single notch percentage: 0.3987
multi  notch percentage: 0.6013
sector_40_num
single notch percentage: 0.4920
multi  notch percentage: 0.5080
sector_45_num
```

```
single notch percentage: 0.3708
multi  notch percentage: 0.6292
sector_50_num
single notch percentage: 0.3880
multi  notch percentage: 0.6120
sector_55_num
single notch percentage: 0.5590
multi  notch percentage: 0.4410
sector_60_num
single notch percentage: 0.7584
multi  notch percentage: 0.2416
```

The code above shows the single notch and multi notch percentage for each sector. We can easily see that the real estate industry, also known as the sector 60, has the highest single notch percentage 75.84%. On the other hand, the information technology industry, also known as the sector 45, has the lowest single notch percentage 37.08%. This phenomenon shows the property of each industry. The information technology industry is often considered as a more risky industry and thus the multi notch percentage is higher compared with other industries.

In [80]:

```
notch_dict["sector_60_num"]
```

Out[80]:

	single	multi	single_percentage	multi_percentage
	Any	Any	Any	Any
1	0	1	0.0000	1.0000
2	5	1	0.8333	0.1667
3	3	3	0.5000	0.5000
4	1	1	0.5000	0.5000
5	2	0	1.0000	0.0000
6	6	2	0.7500	0.2500
7	8	2	0.8000	0.2000
8	0	1	0.0000	1.0000
9	8	0	1.0000	0.0000
10	0	1	0.0000	1.0000
11	3	1	0.7500	0.2500
12	4	0	1.0000	0.0000
13	1	0	1.0000	0.0000
14	2	2	0.5000	0.5000
15	4	1	0.8000	0.2000
16	5	1	0.8333	0.1667
17	2	0	1.0000	0.0000
18	4	0	1.0000	0.0000
19	5	0	1.0000	0.0000
20	0	0	NaN	NaN
21	5	1	0.8333	0.1667
22	2	4	0.3333	0.6667
23	3	4	0.4286	0.5714
24	2	0	1.0000	0.0000
25	0	0	NaN	NaN
26	0	0	NaN	NaN
27	0	0	NaN	NaN
28	0	0	NaN	NaN
29	1	0	1.0000	0.0000
30	0	0	NaN	NaN
⋮	⋮	⋮	⋮	⋮

4.3 Transition Probability

In [81]:

```
tran_mat_dict = Dict()
tran_prob_dict = Dict()
for key in ["sector_10_num","sector_15_num","sector_20_num","sector_25_num","sector_30_num"
    data = numerical_data_dict[key]
    trans=zeros(23,23)
    trans2 = zeros(23,23)
    for j in 1:size(data)[2]
        for k in 1:size(data)[1]-1
            if data[k,j]>0 && data[k+1,j]>0
                trans[data[k,j],data[k+1,j]]=trans[data[k,j],data[k+1,j]]+1
            end
        end
    end
    transframe=DataFrame(trans)
    tran_mat_dict[key] = transframe
    for l in 1:22
        trans2[l,:]=trans[l,:]/sum(trans[l,:])
    end
    transframe2=DataFrame(trans2)
    tran_prob_dict[key] = transframe2
end
```

In [82]:

```
replace_nan(v) = map(x -> isnan(x) ? zero(x) : x, v)
```

Out[82]:

```
replace_nan (generic function with 1 method)
```

In [83]:

```
tran_prob_dict["sector_60_num"] = map(replace_nan, eachcol(tran_prob_dict["sector_60_num"])
```

Out[83]:

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Flo
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
2	0.0000	0.9756	0.0000	0.0244	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
4	0.0000	0.0270	0.0000	0.9189	0.0270	0.0270	0.0000	0.0000	0.0000	0.0000	0.
5	0.0000	0.0000	0.0000	0.1111	0.7778	0.0000	0.1111	0.0000	0.0000	0.0000	0.
6	0.0000	0.0109	0.0000	0.0109	0.0000	0.9239	0.0109	0.0217	0.0109	0.0109	0.
7	0.0000	0.0000	0.0000	0.0000	0.0256	0.0256	0.8974	0.0000	0.0256	0.0256	0.
8	0.0000	0.0000	0.0000	0.0037	0.0000	0.0147	0.0000	0.9632	0.0184	0.0000	0.
9	0.0000	0.0000	0.0000	0.0000	0.0000	0.0020	0.0000	0.0099	0.9622	0.0219	0.
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0009	0.0036	0.0054	0.9682	0.
11	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0008	0.0000	0.0111	0.
12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0010	0.0010	0.0000	0.0031	0.
13	0.0000	0.0000	0.0000	0.0000	0.0000	0.0007	0.0000	0.0000	0.0000	0.0007	0.
14	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
15	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
16	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
17	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0009	0.0000	0.
18	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
19	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
20	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
21	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
22	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.
23	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.

In [84]:

```
head(tran_mat_dict["sector_60_num"])
```

Out[84]:

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Flo
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.(
2	0.0000	40.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.(
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.(
4	0.0000	1.0000	0.0000	34.0000	1.0000	1.0000	0.0000	0.0000	0.0000	0.0000	0.(
5	0.0000	0.0000	0.0000	1.0000	7.0000	0.0000	1.0000	0.0000	0.0000	0.0000	0.(
6	0.0000	1.0000	0.0000	1.0000	0.0000	85.0000	1.0000	2.0000	1.0000	1.0000	0.(

In [85]:

```
head(tran_prob_dict["sector_60_num"])
```

Out[85]:

	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	
	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Float64	Floa
1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0
2	0.0000	0.9756	0.0000	0.0244	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0
3	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0
4	0.0000	0.0270	0.0000	0.9189	0.0270	0.0270	0.0000	0.0000	0.0000	0.0000	0.0
5	0.0000	0.0000	0.0000	0.1111	0.7778	0.0000	0.1111	0.0000	0.0000	0.0000	0.0
6	0.0000	0.0109	0.0000	0.0109	0.0000	0.9239	0.0109	0.0217	0.0109	0.0109	0.0

Next, we try to visualize the transition matrix by exact times and probability. Take the sector 60 for example, there isn't any for data in the second row, which means there isn't any rating 2 (C) in this sector.

In [ ]:

```

```