Introduction

Consumer Financial Protection Bureau is an government agency,it helps consumers complaints heard by financial companies.Comsumer complaints helps the agnecy to study and identify the inappropriate practices and allowing the government to stop those before it becomes a major issue.This project focuses on the analysis of the complaints over different segments,also providing sentiment analysis of the complaints.

About the data

The Consumer Complaint Database is a collection of complaints on a range of consumer financial products and services, sent to companies for response. It started receiving complaints from July 2011.The database generally updates daily, and contains certain information for each complaint, including the source of the complaint, the date of submission, and the company the complaint was sent to for response. The database also includes information about the actions taken by the company in response to the complaint, such as, whether the company's response was timely and how the company responded.

Data Extraction

Dataset used for analysis is US Consumer Finance Complaints data from Kaggle. Importing and Reading the csv file for further analysis,is the first step in data analysis.

There are 18 variables

1.Date received The date the CFPB received the complaint. For example, "05/25/2013."

2.Product
The type of product the consumer identified in the complaint. For example, "Checking or savings account" or "Student loan."

3.Sub-product
The type of sub-product the consumer identified in the complaint. For example, "Checking account" or "Private student loan."

4.Issue The issue the consumer identified in the complaint. For example, "Managing an account" or "Struggling to repay your loan."

5.Sub-issue The sub-issue the consumer identified in the complaint. For example, "Deposits and withdrawals" or "Problem lowering your monthly payments."

6.Consumer complaint narrative
Consumer complaint narrative is the consumer-submitted description of "what happened" from the complaint. Consumers must opt-in to share their narrative. We will not publish the narrative unless the consumer consents, and consumers can opt-out at any time. The CFPB takes reasonable steps to scrub personal information from each complaint that could be used to identify the consumer.

7.Company public response
The company's optional, public-facing response to a consumer's complaint. Companies can choose to select a response from a pre-set list of options that will be posted on the public database. For example, "Company believes complaint is the result of an isolated error."

8.Company
The complaint is about this company. For example, "ABC Bank."

9.State The state of the mailing address provided by the consumer.

10.ZIP code The mailing ZIP code provided by the consumer. This field may: i) include the first five digits of a ZIP code; ii) include the first three digits of a ZIP code (if the consumer consented to publication of their complaint narrative); or iii) be blank (if ZIP codes have been submitted with non-numeric values, if there are less than 20,000 people in a given ZIP code, or if the complaint has an address outside of the United States).

11.Tags Data that supports easier searching and sorting of complaints submitted by or on behalf of consumers.

For example, complaints where the submitter reports the age of the consumer as 62 years or older are tagged "Older American." Complaints submitted by or on behalf of a servicemember or the spouse or dependent of a servicemember are tagged "Servicemember." Servicemember includes anyone who is active duty, National Guard, or Reservist, as well as anyone who previously served and is a veteran or retiree.

12.Consumer consent provided?
Identifies whether the consumer opted in to publish their complaint narrative. We do not publish the narrative unless the consumer consents, and consumers can opt-out at any time.

13.Submitted via
How the complaint was submitted to the CFPB. For example, "Web" or "Phone."

14.Date sent to company The date the CFPB sent the complaint to the company.

15.Company response to consumer This is how the company responded. For example, "Closed with explanation."

16.Timely response? Whether the company gave a timely response. For example, "Yes" or "No."

17.Consumer disputed?
Whether the consumer disputed the company's response.

18.Complaint ID The unique identification number for a complaint.

As we examine the data most of the variables like company,product,sub_product,issue and sub_issue are categorical variables.

Data Wrangling

```
  Cleaning up of data is a very crucial step in all the data analysis projects.Undersatnding the charac
```

```r
complaint2 <- read_csv("consumer_complaints.csv")
```

```
## Parsed with column specification:
## cols(
##   date_received = col_character(),
##   product = col_character(),
##   sub_product = col_character(),
##   issue = col_character(),
##   sub_issue = col_character(),
##   consumer_complaint_narrative = col_character(),
##   company_public_response = col_character(),
##   company = col_character(),
##   state = col_character(),
##   zipcode = col_character(),
##   tags = col_character(),
##   consumer_consent_provided = col_character(),
##   submitted_via = col_character(),
##   date_sent_to_company = col_character(),
##   company_response_to_consumer = col_character(),
##   timely_response = col_character(),
##   `consumer_disputed?` = col_character(),
##   complaint_id = col_integer()
## )
```

```r
complaint2$date_received <- mdy(complaint2$date_received)
complaint2$date_sent_to_company <- mdy(complaint2$date_sent_to_company)
```

We are very much interested in the factor variables,therby converting product,company,sub_product and issue to factors.

```r
complaint2$product<-as.factor(complaint2$product)
complaint2$company<-as.factor(complaint2$company)
complaint2$sub_product<-as.factor(complaint2$sub_product)
complaint2$issue <- as.factor(complaint2$issue)
```

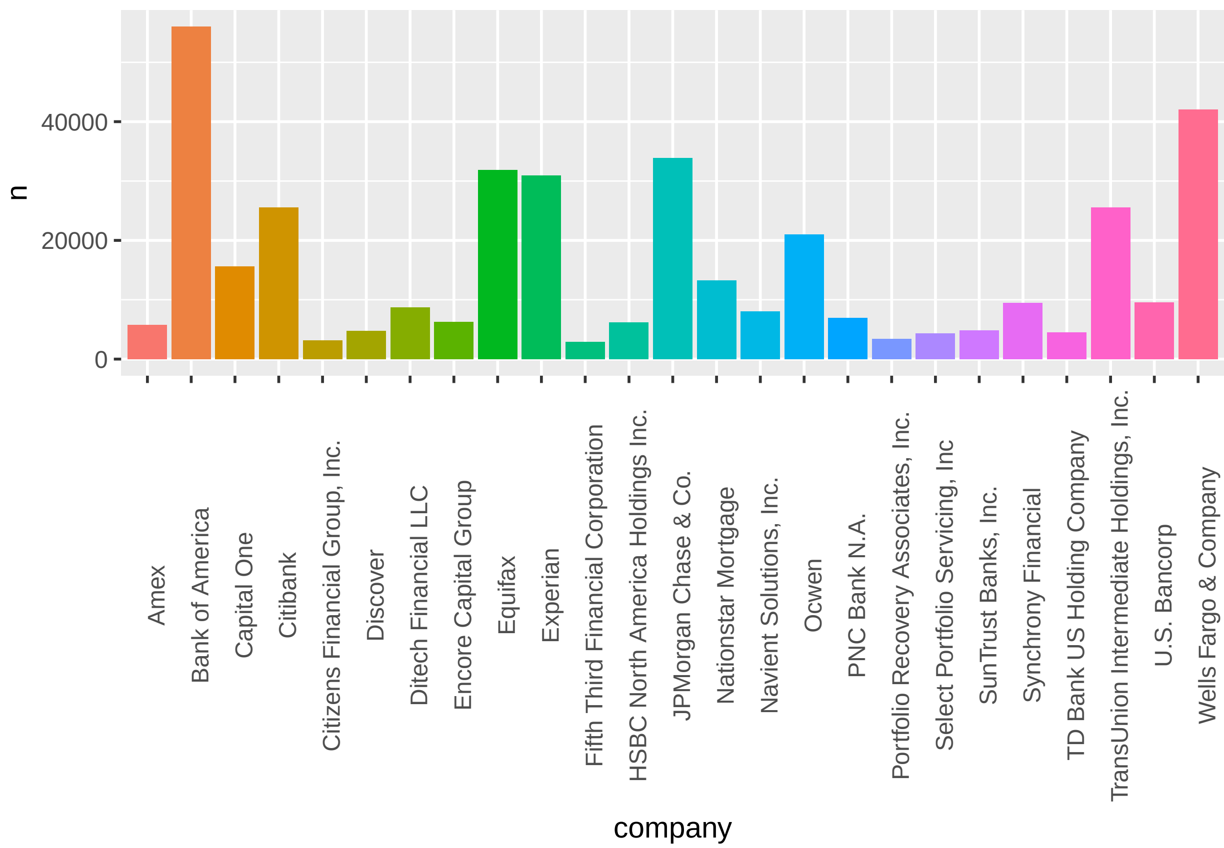
Exploratory Data Analysis

```
EDA helps us to visualize and explore our data deeper. The advantages of EDA are
* Able to visualize better
* Able to ask more questions and refine them
* Able to identify redundancy in data

In our data as complaints are the records of study, we are expnading our questions on categories wi
```

Top 25 companies with highest number of complaints

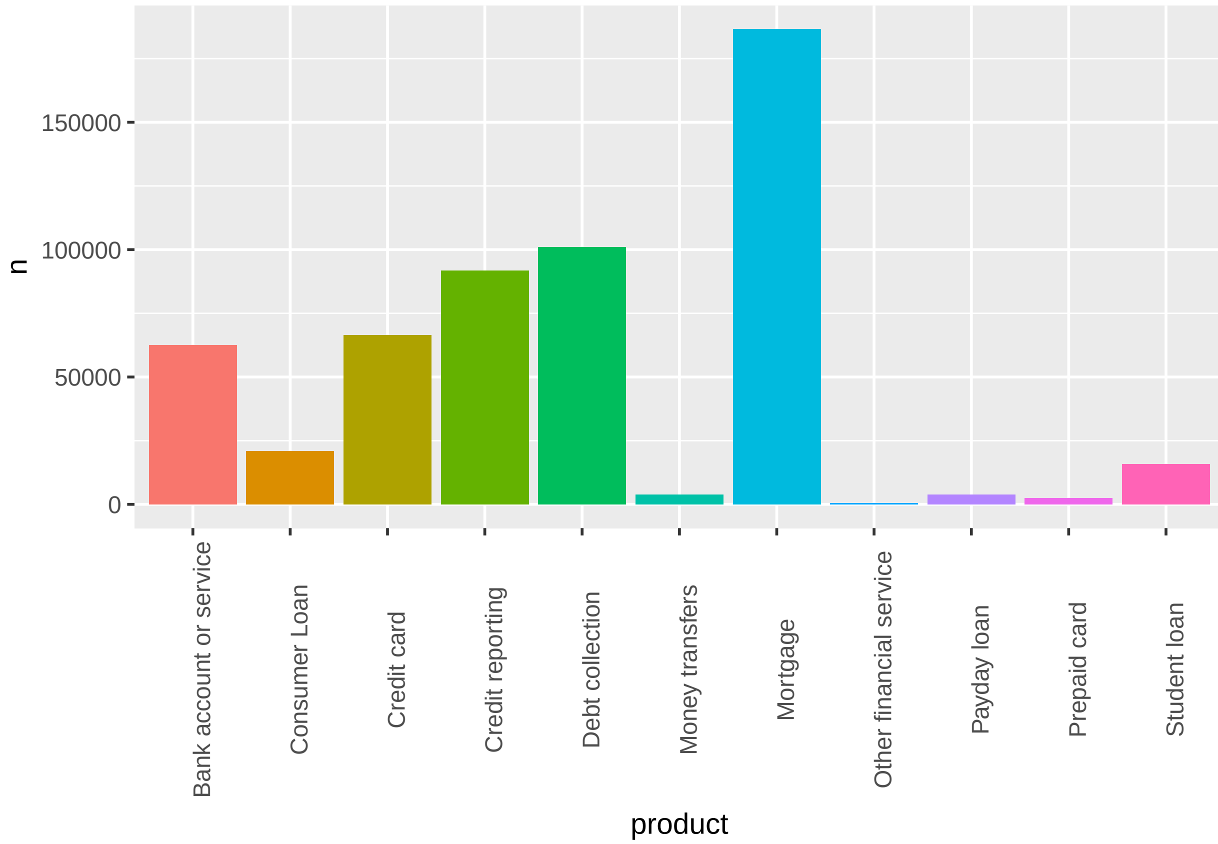
```r
complaint2 %>%
  count(company) %>%
  arrange(desc(n))%>%
  top_n(25) %>%
  ggplot(aes(company,n,fill=company))+
  geom_bar(stat="identity") +
  theme(axis.text.x=element_text(angle=90),legend.position = "none")
```

```
## Selecting by n
```



Products with highest number of complaints

```
complaint2 %>%
  count(product) %>%
  arrange(desc(n))%>%
  ggplot(aes(product,n,fill=product))+
  geom_bar(stat="identity")+
  theme(axis.text.x=element_text(angle=90),legend.position = "none")
```

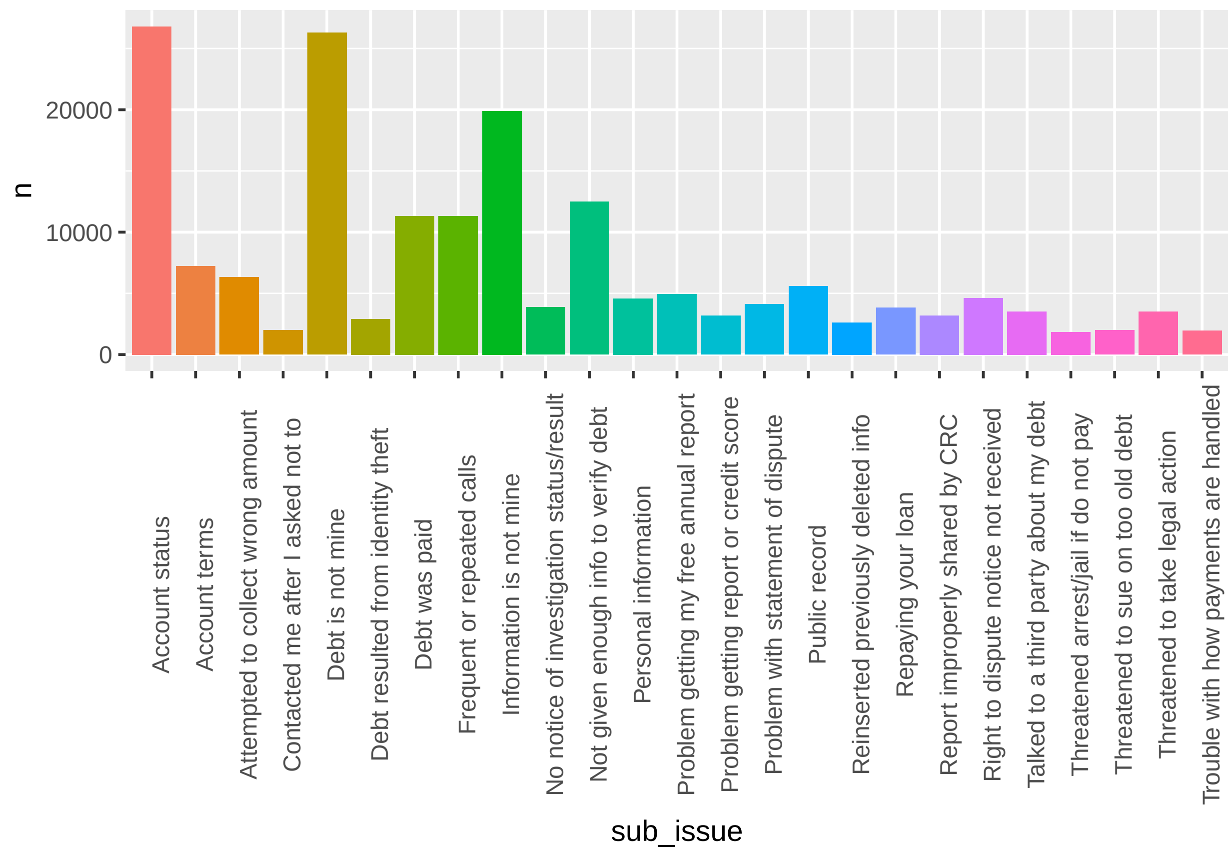



```r
sub_issue_data <- complaint2 %>%
select(complaint_id,sub_issue)%>%
na.omit()
```

Complaint numbers based on the Issue categories

```r
sub_issue_data %>%
  count(sub_issue) %>%
  arrange(desc(n))%>%
  top_n(25) %>%
  ggplot(aes(sub_issue,n,fill=sub_issue))+
  geom_bar(stat="identity")+
  theme(axis.text.x=element_text(angle=90),legend.position = "none")
```

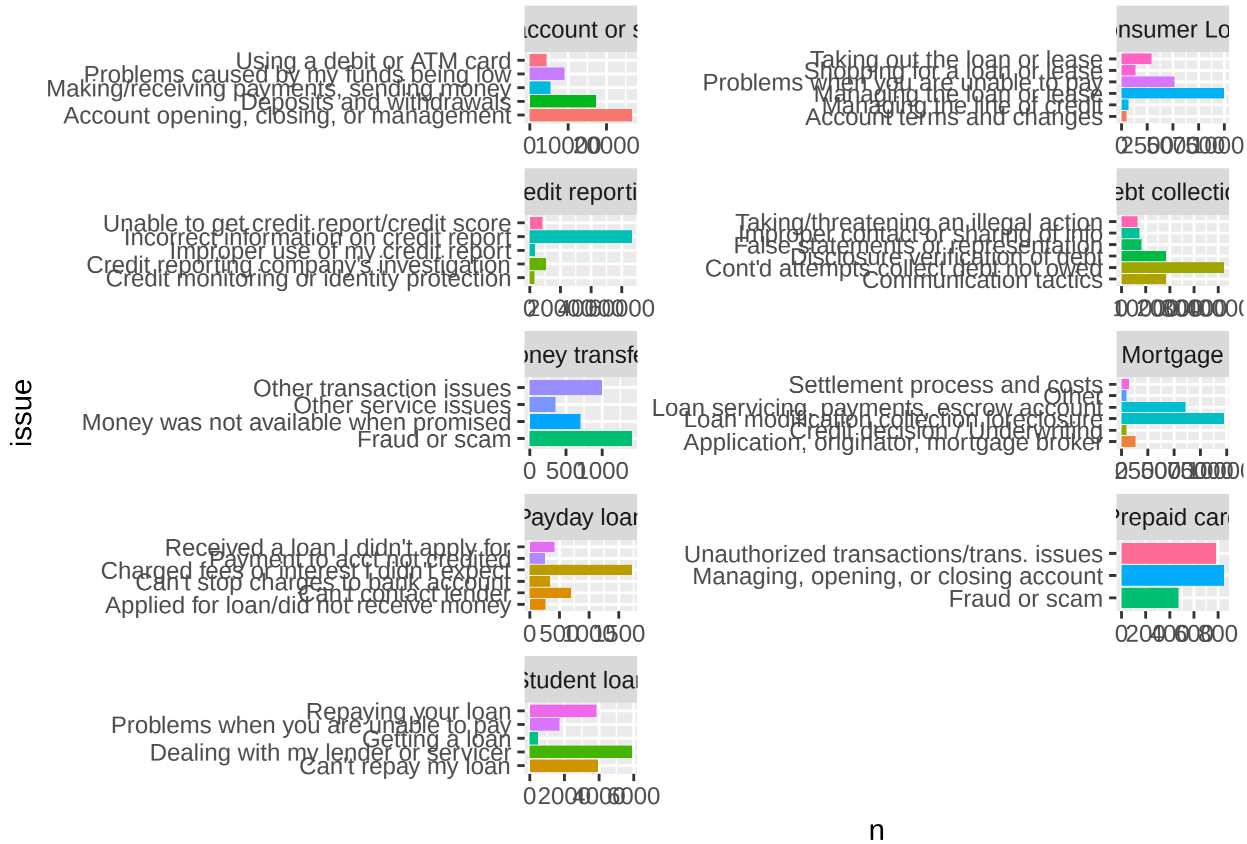
```
## Selecting by n
```




```
product_issue <- complaint2 %>%
  select(product,issue)%>%
  na.omit %>%
  group_by(product,issue)%>%
  count()
```

To identify top issues resported by customers under each product other than Credit card

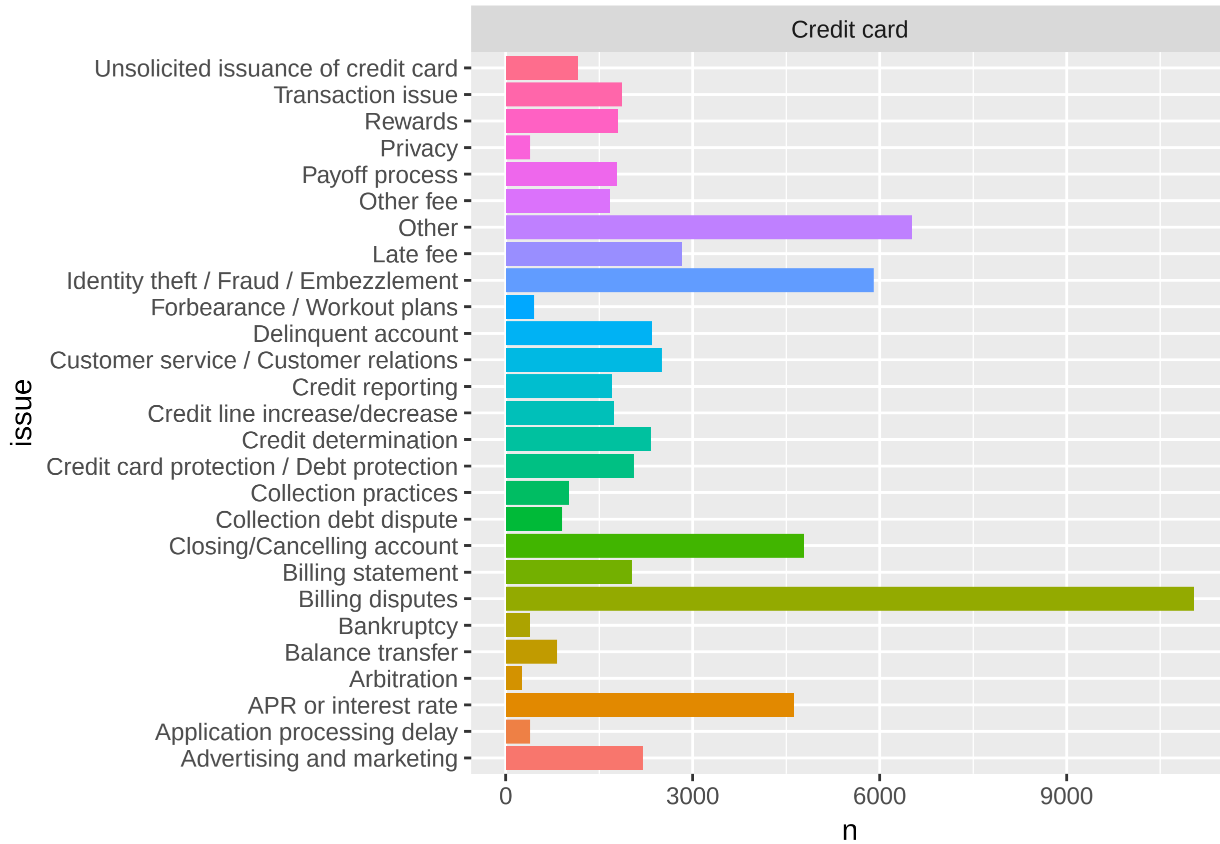
```
product_issue %>%
  filter(n>250)%>%
  filter(product !="Credit card") %>%
  ggplot(aes(issue,n,fill=issue))+
  geom_bar(stat="identity")+
 theme(legend.position = "none")+
 facet_wrap(~product,scale= "free",nrow = 6)+
  coord_flip()
```



In each product we are having one main issue that was reported repeatedly by customers. For example, Bank account product - Account management received more complaints, Incorrect information on credit report is the top issue under credit reporting category.

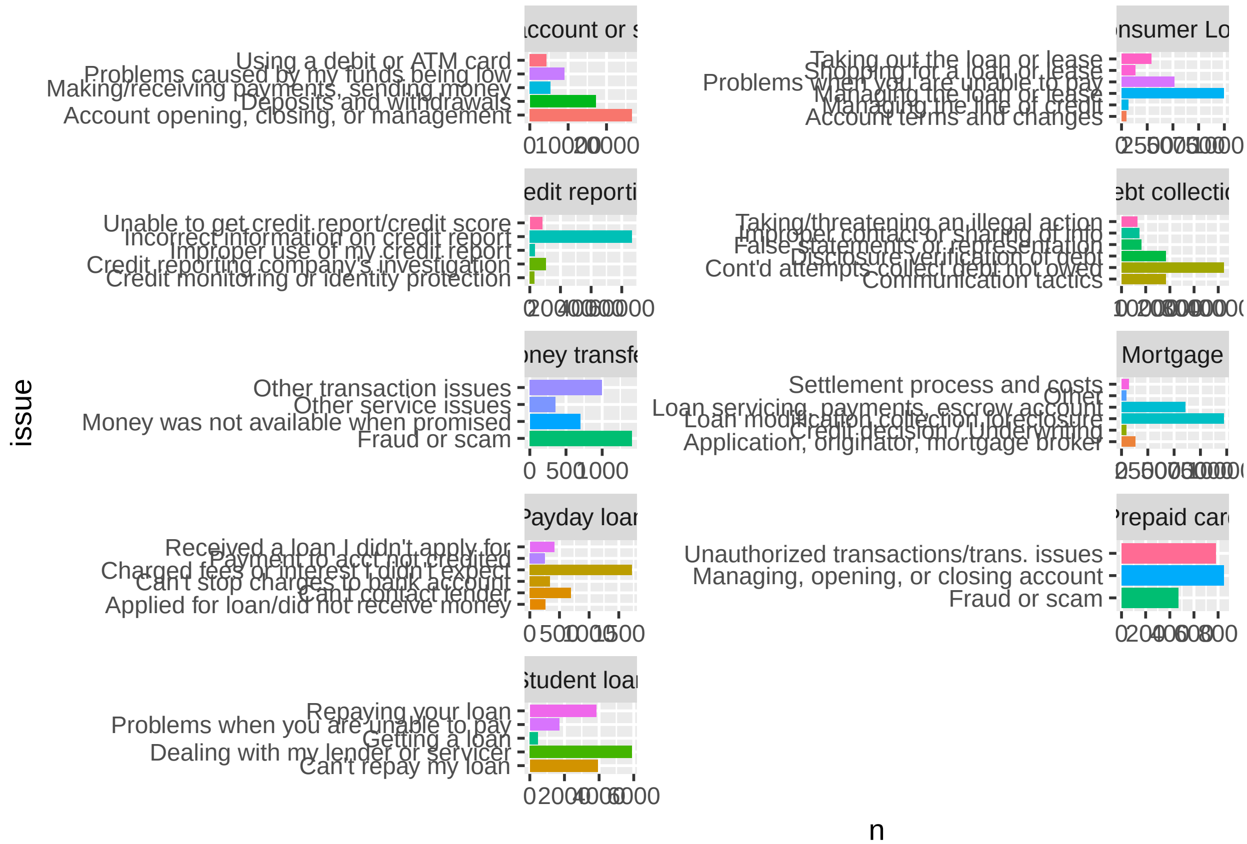
Complaint category under Credit card

```
product_issue %>%
  filter(n>250)%>%
  filter(product =="Credit card") %>%
  ggplot(aes(issue,n,fill=issue))+
  geom_bar(stat="identity")+
  theme(legend.position = "none")+
  facet_wrap(~product,scale= "free",nrow = 6)+
  coord_flip()
```



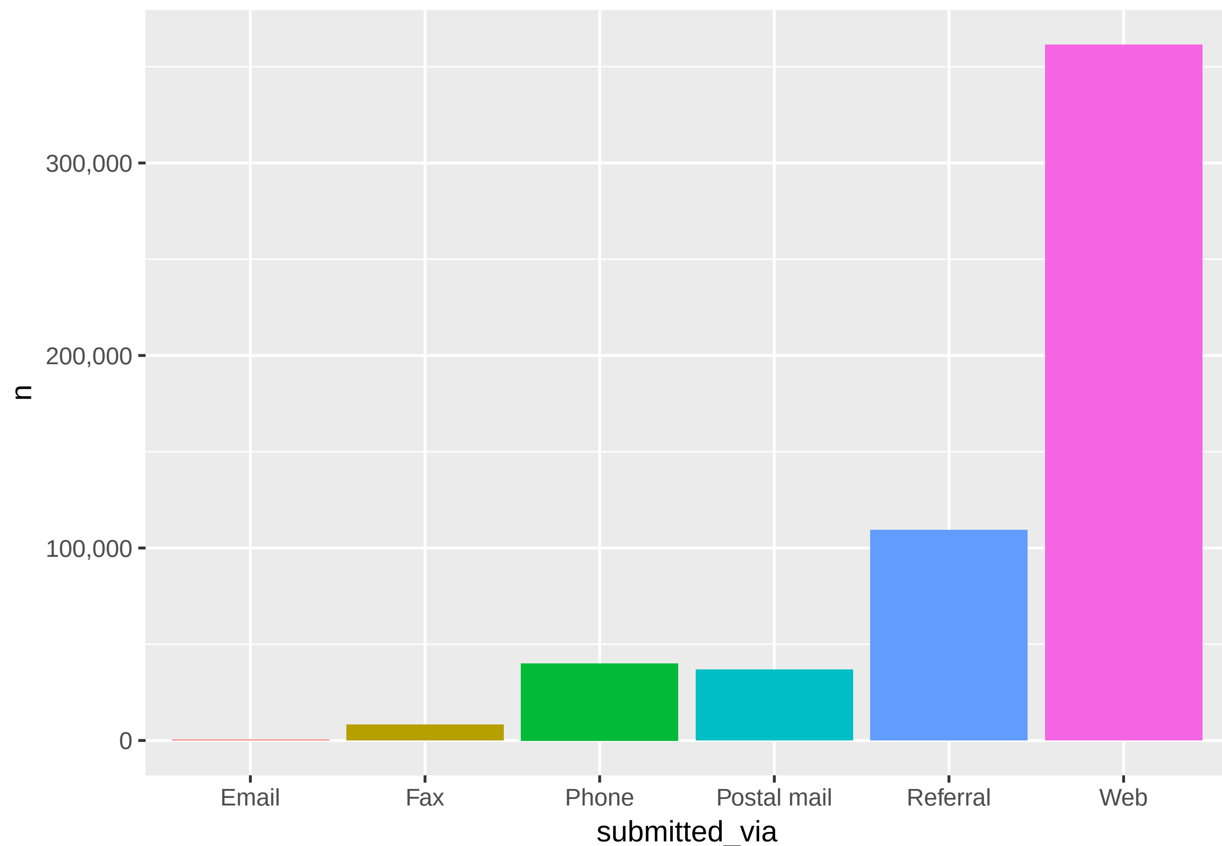
####Complaint category for products other than credit card

```
product_issue %>%
  filter(n>250)%>%
  filter(product !="Credit card") %>%
  ggplot(aes(issue,n,fill=issue))+
  geom_bar(stat="identity")+
  theme(legend.position = "none")+
  facet_wrap(~product,scale= "free",nrow = 6)+
  coord_flip()
```



From which mode more complaints are received

```
complaint2 %>%
  select(company,product,issue,submitted_via)%>%
  na.omit()%>%
  count(submitted_via) %>%
  arrange(desc(n))%>%
  ggplot(aes(submitted_via,n,fill=submitted_via))+
  scale_y_continuous(labels = scales :: comma)+
  geom_bar(stat="identity")+
  theme(legend.position = "none")
```

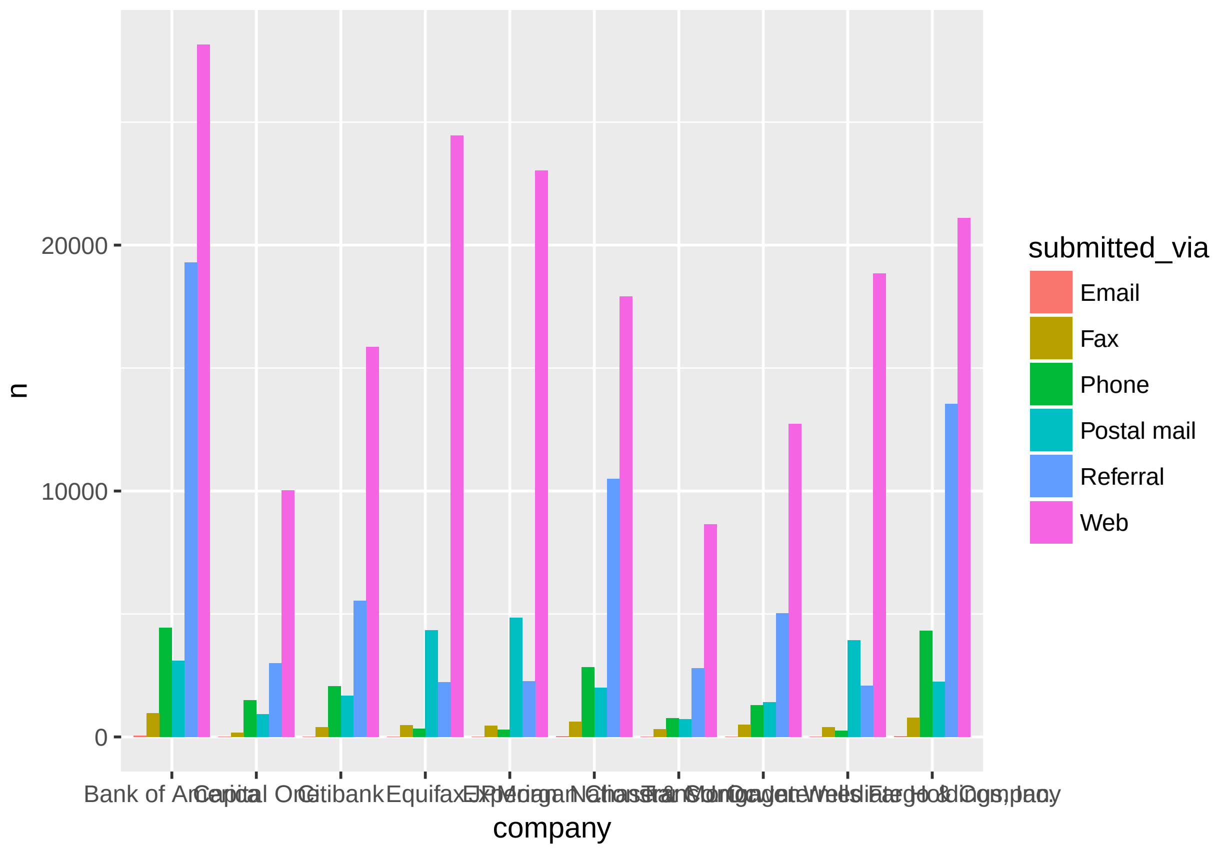



```
top_companies <- complaint2 %>%
  count(company) %>%
  arrange(desc(n))%>%
  top_n(10)
```

```
## Selecting by n
```

Mode by which more complaints are received based on companies

```
complaint2 %>%
  select(company,product,issue,submitted_via)%>%
  filter(company %in% top_companies$company)%>%
  group_by(company)%>%
  na.omit()%>%
  count(submitted_via) %>%
  ggplot(aes(company,n,fill=submitted_via))+
  geom_bar(stat="identity",position = position_dodge())
```



Distribution of complaints over United States

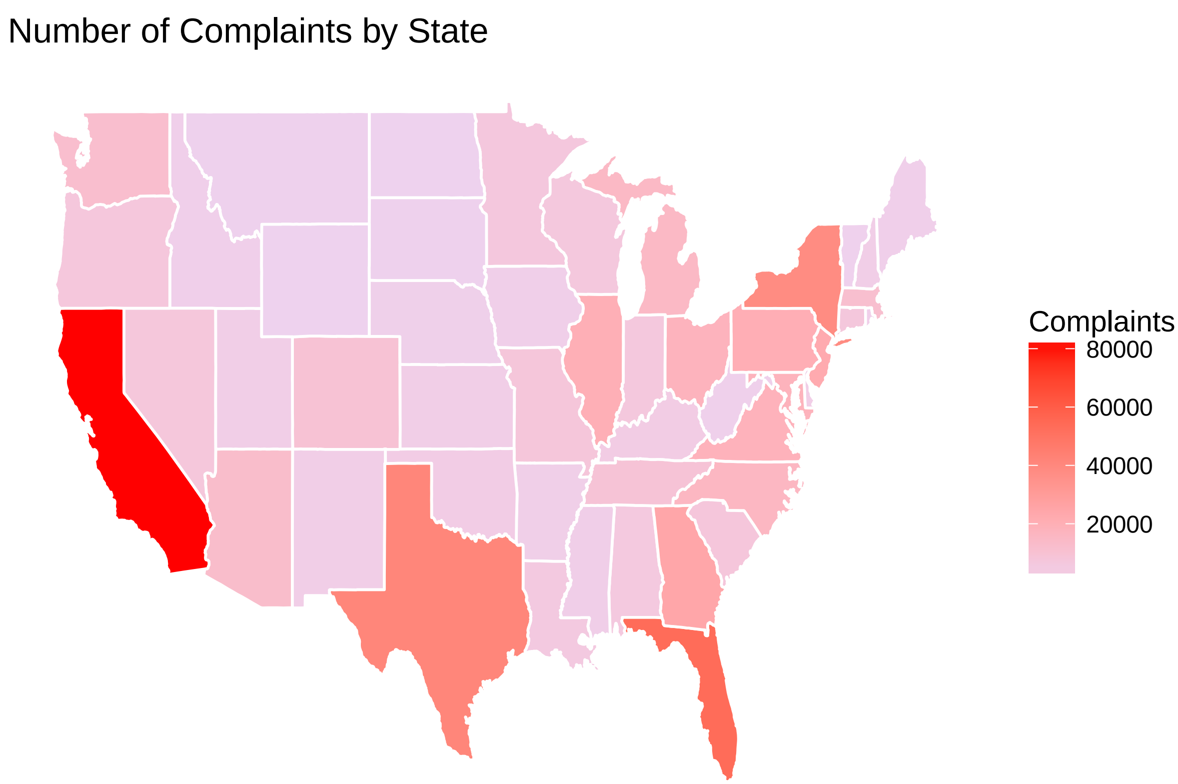
As there is a state information from which the complaints was received, distribution of complaints over United States can be visualized by map packages.

```
all_states <- map_data("state")

complaint2 <- complaint2 %>%
  mutate(region = state.name[match(state,state.abb)] )

complaint2$region[is.na(complaint2$region)] <- "district of columbia"
complaint2$region <- tolower(complaint2$region)
```

```
map_complaint <- complaint2 %>%
  select(company,product,region) %>%
  group_by(region)%>%
  count(region)
```

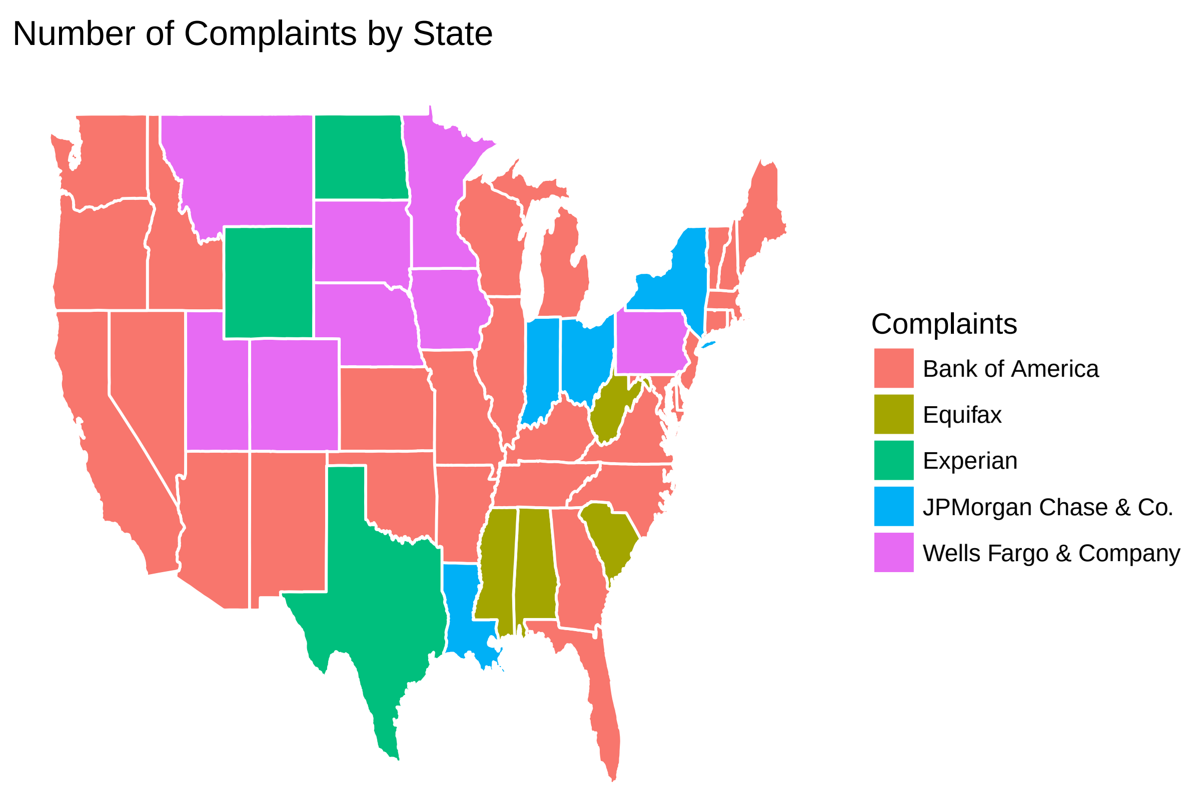
```
map_state <- merge(all_states,map_complaint,by="region")
map_state<- map_state[map_state$region!="district of columbia",]

    map_state %>%
      ggplot(aes(x=long,lat,group=group,fill= n))+
      geom_polygon(color="white")+
      scale_fill_continuous(low = "thistle2",high="red",guide="colorbar")+
      theme_bw()+labs(fill = "Complaints",title="Number of Complaints by State",x="",y="")+
```

```
    scale_y_continuous(breaks=c())+scale_x_continuous(breaks=c())+theme(panel.border=element_blank())
```



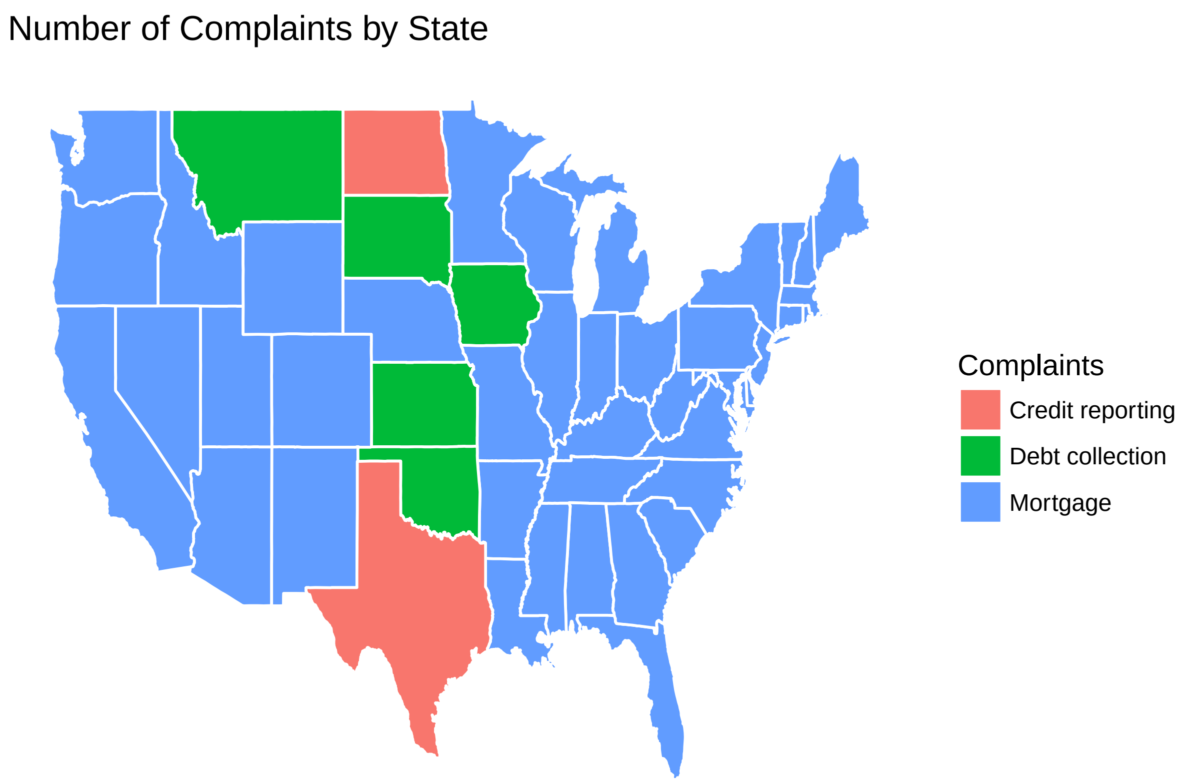
Companies with highest number of complaints distribution based on state

```
comp_company_state <- complaint2 %>%
      select(company,region)%>%
      group_by(region)%>%
      count(company)%>%
      top_n(1,n)%>%
      arrange(desc(n))

   map_company <-merge(all_states,comp_company_state,by="region")
   map_company<- map_company[map_company$region!="district of columbia",]

   map_company %>%
     ggplot(aes(x=long,lat,group=group,fill= company))+
     geom_polygon(color="white")+
     theme_bw()+labs(fill = "Complaints",title="Number of Complaints by State",x="",y="")+
     scale_y_continuous(breaks=c())+scale_x_continuous(breaks=c())+theme(panel.border=element_blank())
```



Products with highest number of complaints distribution based on state

```
comp_product_state <- complaint2 %>%
      select(product,region)%>%
      group_by(region)%>%
      count(product)%>%
      top_n(1,n)%>%
      arrange(desc(n))


    map_product <-merge(all_states,comp_product_state,by="region")
    map_product<- map_product[map_product$region!="district of columbia",]


    map_product %>%
      ggplot(aes(x=long,lat,group=group,fill= product))+
      geom_polygon(color="white")+
      theme_bw()+labs(fill = "Complaints",title="Number of Complaints by State",x="",y="")+
      scale_y_continuous(breaks=c())+scale_x_continuous(breaks=c())+theme(panel.border=element_blank())
```



Data Analysis

Sentimental Analysis helps to understand the emotional intent of words to infer whether the part of text is positive or negative.

The Consumer Complaint Database by name implies is a complaint database. so obviously the expectation is it reveals mainly negative sentiment.Calaculating parameters with variables will provide greater clear picture.

Tidytext package is used for sentimental analysis. Tidytext package have maimly three lexicons,among those "bing" lexicon is used for analysis.

For Sentimental Analysis we need to clean up the text before used for analysis like removing white spaces,unwanted punctuations and removing stopwords.

Function to clean the text

```
tm_clean <- function(corpus){
      tm_clean <- tm_map(corpus,removePunctuation)
      corpus <- tm_map(corpus,stripWhitespace)
      corpus <- tm_map(corpus,removeWords,c(stopwords("en"),"xxxx","xx"))
      return(corpus)
}
```

```
data <- complaint2 %>%
  select(company,product,issue,state,zipcode,submitted_via,company_response_to_consumer,timely_response
  na.omit
```

Function to calculate sentiment

```
GetSentiment <- function(i){
      sentiment1 <- data %>%
        filter(company == i ) %>%
        select(consumer_complaint_narrative) %>%
        VectorSource() %>%
        VCorpus() %>%
        tm_clean() %>%
        DocumentTermMatrix()%>%
        tidy()%>%
        inner_join(get_sentiments("bing"),c(term = "word")) %>% # pull out only sentimen words
        count(sentiment) %>% # count the # of positive & negative words
        spread(sentiment, n, fill = 0) %>% # made data wide rather than narrow
        mutate(sentiment = positive - negative) %>% # # of positive words - # of negative owrds
        mutate(company = i)
      return(sentiment1)
    }
```

```
company_consumer_comp <- complaint2 %>%
      select(company,consumer_complaint_narrative)%>%
      na.omit() %>%
      count(company) %>%
      arrange(desc(n))%>%
      filter(n>100)
```

Calculating overall sentiments for companies

```
comp <- company_consumer_comp$company

    listcomp <- as.list(comp)

    sentiments1 <- data_frame()

    for(i in listcomp )
    {
      sentiments1 <- rbind(sentiments1,GetSentiment(i))
    }

    sentiments1
```

```
## # A tibble: 81 x 4
##    negative positive sentiment company
##       <dbl>    <dbl>     <dbl> <fctr>
##  1      832      405      -427 Equifax
##  2      872      402      -470 Experian
##  3      830      392      -438 TransUnion Intermediate Holdings, Inc.
##  4     1219      551      -668 Bank of America
##  5     1129      556      -573 Wells Fargo & Company
##  6      971      477      -494 Citibank
##  7     1045      497      -548 JPMorgan Chase & Co.
##  8      840      403      -437 Ocwen
##  9      714      342      -372 Capital One
## 10      719      316      -403 Synchrony Financial
## # ... with 71 more rows
```

Complaint percentage calculation function

```
GetPercentage <- function(i){
      d <- data %>%
        filter(company == i) %>%
        count(company) %>%
        mutate(per = (n/66617)*100)
      return(d)
    }
```

Companies complaint percentage.

```
    complaint_percent <- data_frame()
    for(i in listcomp )
    {
      complaint_percent <- rbind(complaint_percent,GetPercentage(i))
    }
    complaint_percent
```

```
## # A tibble: 81 x 3
##    company                                   n   per
##    <fctr>                                <int> <dbl>
##  1 Equifax                                4187  6.29
##  2 Experian                               3929  5.90
##  3 TransUnion Intermediate Holdings, Inc. 3850  5.78
##  4 Bank of America                        3473  5.21
##  5 Wells Fargo & Company                  3058  4.59
##  6 Citibank                               2772  4.16
##  7 JPMorgan Chase & Co.                   2578  3.87
##  8 Ocwen                                  1620  2.43
##  9 Capital One                            1502  2.25
## 10 Synchrony Financial                    1371  2.06
## # ... with 71 more rows
```

Dispute rate Calculation function

```
    disp_rate <- function(i){
      d1 <- data %>%
        filter(company == i) %>%
        count(company,`consumer_disputed?`) %>%
        spread(`consumer_disputed?`,n,fill=0,drop = TRUE) %>%
        mutate(total = Yes + No) %>%
        mutate(YP = (Yes/total)*100) %>%
        mutate(NP = (No/total)*100)
      return(d1)
       }
```

Companies Dispute rate

```
    dispute_rate <- data_frame()

    for(i in listcomp)
    {
      dispute_rate<- rbind(dispute_rate,disp_rate(i))
```

```
}
dispute_rate
```

```
## # A tibble: 81 x 6
##    company                                 No   Yes total    YP    NP
##    <fctr>                               <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Equifax                               2977  1210  4187  28.9  71.1
##  2 Experian                              3308   621  3929  15.8  84.2
##  3 TransUnion Intermediate Holdings, Inc. 3114   736  3850  19.1  80.9
##  4 Bank of America                       2613   860  3473  24.8  75.2
##  5 Wells Fargo & Company                 2207   851  3058  27.8  72.2
##  6 Citibank                              2171   601  2772  21.7  78.3
##  7 JPMorgan Chase & Co.                  1854   724  2578  28.1  71.9
##  8 Ocwen                                 1167   453  1620  28.0  72.0
##  9 Capital One                           1242   260  1502  17.3  82.7
## 10 Synchrony Financial                   1109   262  1371  19.1  80.9
## # ... with 71 more rows
```

Companies response calculation

```
company_response <- function(i){
  d4 <- data %>%
    filter(company == i) %>%
    count(company,timely_response)

  return(d4)
   }
```

Companies timely response

```
tim_resp <- data_frame()

for(i in listcomp )
{
  tim_resp <- rbind(tim_resp,company_response(i))

}

tim_resp
```

```
## # A tibble: 112 x 3
##    company                                timely_response     n
##    <fctr>                                 <chr>           <int>
##  1 Equifax                                Yes              4187
##  2 Experian                               Yes              3929
##  3 TransUnion Intermediate Holdings, Inc. Yes              3850
##  4 Bank of America                        No                  3
##  5 Bank of America                        Yes              3470
##  6 Wells Fargo & Company                  No                 61
##  7 Wells Fargo & Company                  Yes              2997
##  8 Citibank                               No                  1
##  9 Citibank                               Yes              2771
## 10 JPMorgan Chase & Co.                   No                  2
## # ... with 102 more rows
```

Calculating yes and No percentage

```
    resp_percent <- tim_resp %>%
      spread(timely_response,n,fill=0,drop = TRUE) %>%
      mutate(total = Yes + No) %>%
      mutate(YP = (Yes/total)*100) %>%
      mutate(NP = (No/total)*100)%>%
      arrange(desc(total))
```

####Building a DataFrame with all the parameters calculated

```
    result1 <- full_join(complaint_percent,dispute_rate,by = "company")
    result2 <- full_join(result1,sentiments1,by ="company")
    result3 <- full_join(result2,resp_percent,by="company")
```

```
    final_result <- result3%>%
      select(company,n,per,No.x,Yes.x,YP.x,NP.x,negative,positive,sentiment,
             No.y,Yes.y,YP.y,NP.y)%>%
      setNames(c("Company","Total Complaints","Complaint Percent","No Disputes",
                 "Disputes","Dispute Percent","No Dispute Percent","Negative Sentiment",
                 "Positive Sentiment","Sentiment","No Timely Response","Timely Response",
                 "Timely Response Percent","No Timely Response Percent"))

final_result
```

```
## # A tibble: 81 x 14
##    Comp~ `Tot~ `Com~ `No ~ Disp~ `Dis~ `No ~ `Neg~ `Pos~ Sent~ `No ~ `Tim~
##    <fct> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1 Equi~  4187  6.29  2977  1210  28.9  71.1   832   405  -427  0     4187
##  2 Expe~  3929  5.90  3308   621  15.8  84.2   872   402  -470  0     3929
##  3 Tran~  3850  5.78  3114   736  19.1  80.9   830   392  -438  0     3850
##  4 Bank~  3473  5.21  2613   860  24.8  75.2  1219   551  -668  3.00  3470
##  5 Well~  3058  4.59  2207   851  27.8  72.2  1129   556  -573 61.0   2997
##  6 Citi~  2772  4.16  2171   601  21.7  78.3   971   477  -494  1.00  2771
##  7 JPMo~  2578  3.87  1854   724  28.1  71.9  1045   497  -548  2.00  2576
##  8 Ocwen  1620  2.43  1167   453  28.0  72.0   840   403  -437 13.0   1607
##  9 Capi~  1502  2.25  1242   260  17.3  82.7   714   342  -372  3.00  1499
## 10 Sync~  1371  2.06  1109   262  19.1  80.9   719   316  -403  0     1371
## # ... with 71 more rows, and 2 more variables: `Timely Response Percent`
## #   <dbl>, `No Timely Response Percent` <dbl>
```

Text Mining

Case Study of Consumer Complaints based on book Text Mining with R [https://www.tidytextmining.com/]

For any analysis data wrangling is an important step. To do analysis on consumer complaints text_clean() custom function is used to remove whitespace,irrelevant symbols and numberss,converting all characters to lowercase and replacing the contraction words accordingly.

```
text.clean = function(x)                    # text data
{ require("tm")
  x  =  gsub("n't","not",x)
```

```
  x  = gsub("'","",x)
  x  =  gsub("<.*?>", " ", x)              # regex for removing HTML tags
  x  =  iconv(x, "latin1", "ASCII", sub="") # Keep only ASCII characters
  x  =  gsub("[^[:alnum:]]", " ", x)        # keep only alpha numeric
  x  =  tolower(x)                          # convert to lower case characters
  x  =  removeNumbers(x)                    # removing numbers
  x  =  stripWhitespace(x)                  # removing white space
#  x  =  gsub("^\\s+|\\s+$", "", x) # remove leading and trailing white space

  return(x)
}
```

Stopwords are words that are irrelavant or the fillers.Stop_words tibble in tidy text package have the list of stopwords. Real challenge while using stop_words is it contains the words "not","none" and "noone", as this is a complaint database the removal of above words results in more positive sentiment as that is not right way to move on. So creating a custom_stop_word list is important for this dataset. Also negation words like "didn't","can't" etc need to be taken into account.

```
negation_word <- data.frame(negation.words)
negation_word <- setNames(negation_word,c("word"))
negation_word$word <- as.character(negation_word$word)
negation_word <- bind_rows(negation_word,data_frame(word=
                                                     c("non","noone","none",as.character(1:3))))

stopword <- stop_words %>%
  anti_join(negation_word)
```

```
## Joining, by = "word"
```

```
stopword <- data.frame(stopword$word)
stopword <- setNames(stopword,c("word"))
stopword$word <- as.character(stopword$word)

custom_stopword <- data_frame(word =
          c(as.character(1:10),"xxxx","xxx","xxxxx","xx","x","company","companies","said","told",
                                  "however","since","asked","stated","equifax","well","item","items","d
                                  "going","n_t","s"))


comn_stop_word <- bind_rows(stopword,custom_stopword)

comn_stop_word <- list(comn_stop_word$word)
```

From our Explorotary Data Analysis top two products with more complaints are Mortgage and Debt Collection. So considering these two products for our text mining.

Text mining on Product - Mortgage

Considering only complaints based on product Mortgage,cleaning up the data using text_clean data and removing custom stop words.

```
Mortgage_comments <- data%>%
  filter(product =='Mortgage')%>%
  select(issue,consumer_complaint_narrative)
```

```r
Mortgage_comments$consumer_complaint_narrative <- text.clean(Mortgage_comments$consumer_complaint_narra

Mortgage_comments$consumer_complaint_narrative <- removeWords(Mortgage_comments$consumer_complaint_narra
```

Unigram Analysis

Sentences can be tokenize into consective sequence of words called ngrams. By using unnest_tokens this can be achieved. Unigram analysis is done by splitting up the sentence to one ngarm per row and their frequency.

```r
Mortgage_comments_sentiment <- Mortgage_comments %>%
  group_by(issue)%>%
  unnest_tokens(word,consumer_complaint_narrative) %>%
  anti_join(stopword)%>%
  anti_join(custom_stopword)%>%
  count(word,sort = TRUE)%>%
  inner_join(get_sentiments("bing"))%>%
  count(issue,sentiment)%>%
  spread(sentiment, nn, fill = 0) %>%
  mutate(sentiment = positive - negative)
```

```
## Joining, by = "word"
## Joining, by = "word"
## Joining, by = "word"
```

```r
Mortgage_comments_sentiment
```

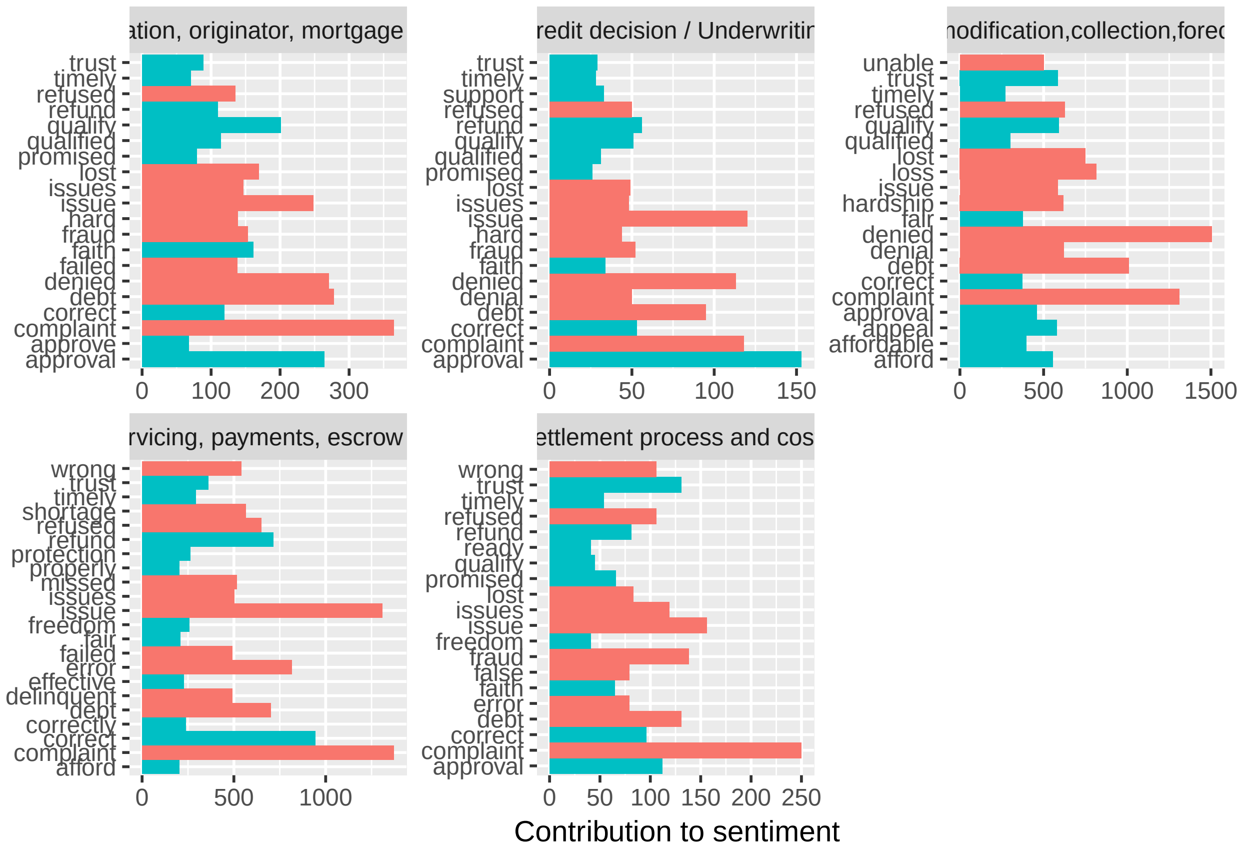
```
## # A tibble: 5 x 4
## # Groups: issue [5]
##   issue                                   negative positive sentiment
##   <fctr>                                     <dbl>    <dbl>     <dbl>
## 1 Application, originator, mortgage broker     990      459      -531
## 2 Credit decision / Underwriting               581      292      -289
## 3 Loan modification,collection,foreclosure    1468      615      -853
## 4 Loan servicing, payments, escrow account    1464      588      -876
## 5 Settlement process and costs                 785      358      -427
```

```r
Mortgage_unigram_sentiment<- Mortgage_comments %>%
  group_by(issue)%>%
  unnest_tokens(word,consumer_complaint_narrative) %>%
  anti_join(stopword)%>%
  anti_join(custom_stopword)%>%
  inner_join(get_sentiments("bing"))%>%
  count(word,sentiment,sort = TRUE)%>%
  ungroup()%>%
  group_by(issue,sentiment)%>%
  top_n(10)%>%
  ggplot(aes(word, n, fill = sentiment)) +
  geom_col(show.legend = FALSE,width = 1) +
  facet_wrap(~issue, scale = "free") +
  labs(y = "Contribution to sentiment",
       x = NULL) +
  coord_flip()
```

```
## Joining, by = "word"
```

```
## Joining, by = "word"
## Joining, by = "word"

## Selecting by n
```

```r
Mortgage_unigram_sentiment
```



Bigram and Trigram Analysis

Though Unigram analysis help us in identifying the neagtive words, further the analysis can be extended towards bigram and trigram that will create more meaningful insights.

```r
Mortgage_bigram <- Mortgage_comments %>%
  # group_by(issue)%>%
  unnest_tokens(bigram,consumer_complaint_narrative,token="ngrams",n=2) %>%
  count(issue,bigram,sort = TRUE)


Mortgage_bigram
```

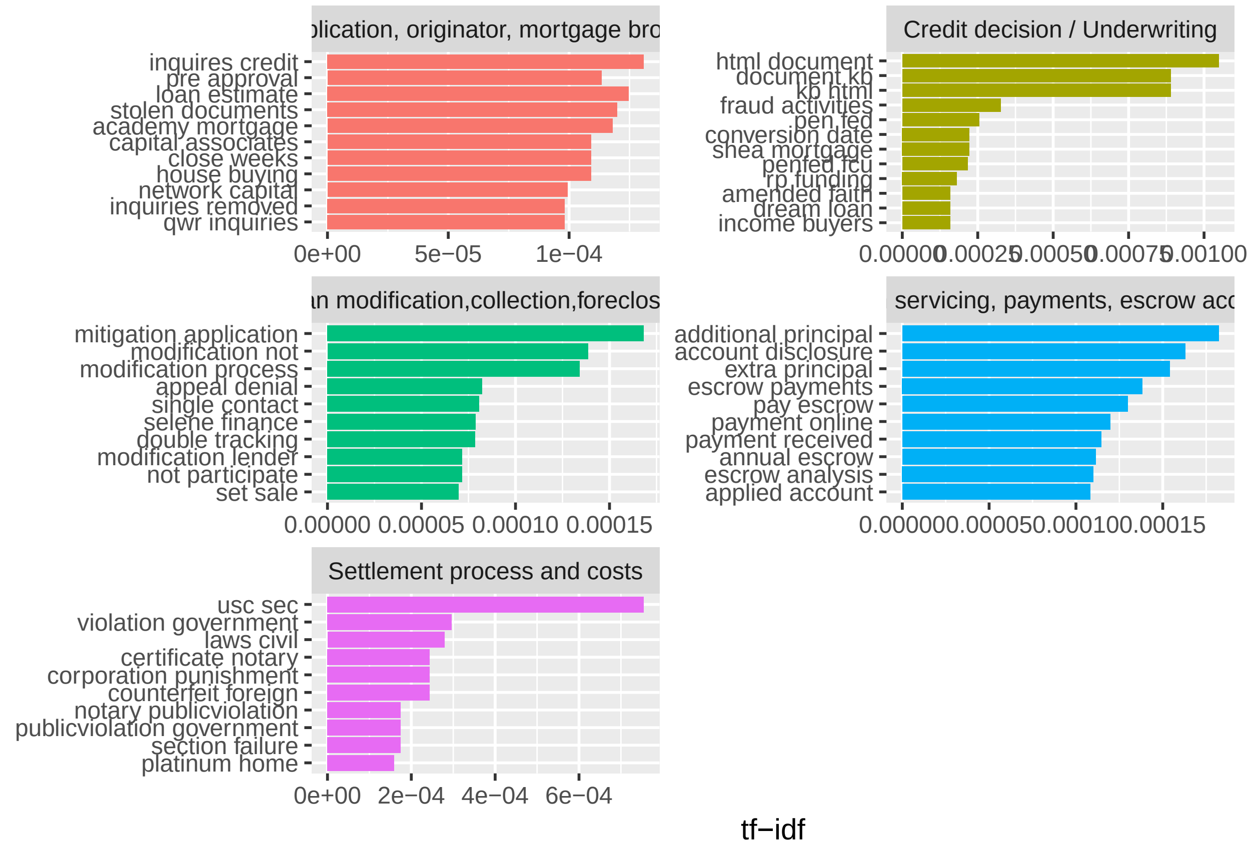
```
## # A tibble: 707,455 x 3
##    issue                                   bigram                 n
##    <fctr>                                  <chr>              <int>
##  1 Loan modification,collection,foreclosure loan modification  3131
##  2 Loan modification,collection,foreclosure short sale         1854
##  3 Loan modification,collection,foreclosure bank america       1423
##  4 Loan servicing, payments, escrow account mortgage payment   1332
```

```
##  5 Loan servicing, payments, escrow account escrow account     1320
##  6 Loan servicing, payments, escrow account customer service   1029
##  7 Loan servicing, payments, escrow account bank america       1027
##  8 Loan servicing, payments, escrow account green tree          930
##  9 Loan servicing, payments, escrow account loan modification   817
## 10 Loan servicing, payments, escrow account payment not         806
## # ... with 707,445 more rows
```

```
Mortgage_bigram_graph <- Mortgage_comments %>%
  # group_by(issue)%>%
  unnest_tokens(bigram,consumer_complaint_narrative,token="ngrams",n=2) %>%
  count(issue,bigram,sort = TRUE)%>%
  ungroup()%>%
  bind_tf_idf(bigram,issue,n)%>%
  arrange(desc(tf_idf))%>%
  mutate(bigram = factor(bigram, levels = rev(unique(bigram)))) %>%
  group_by(issue) %>%
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(bigram, tf_idf, fill = issue)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~issue, ncol = 2, scales = "free") +
  coord_flip()
```

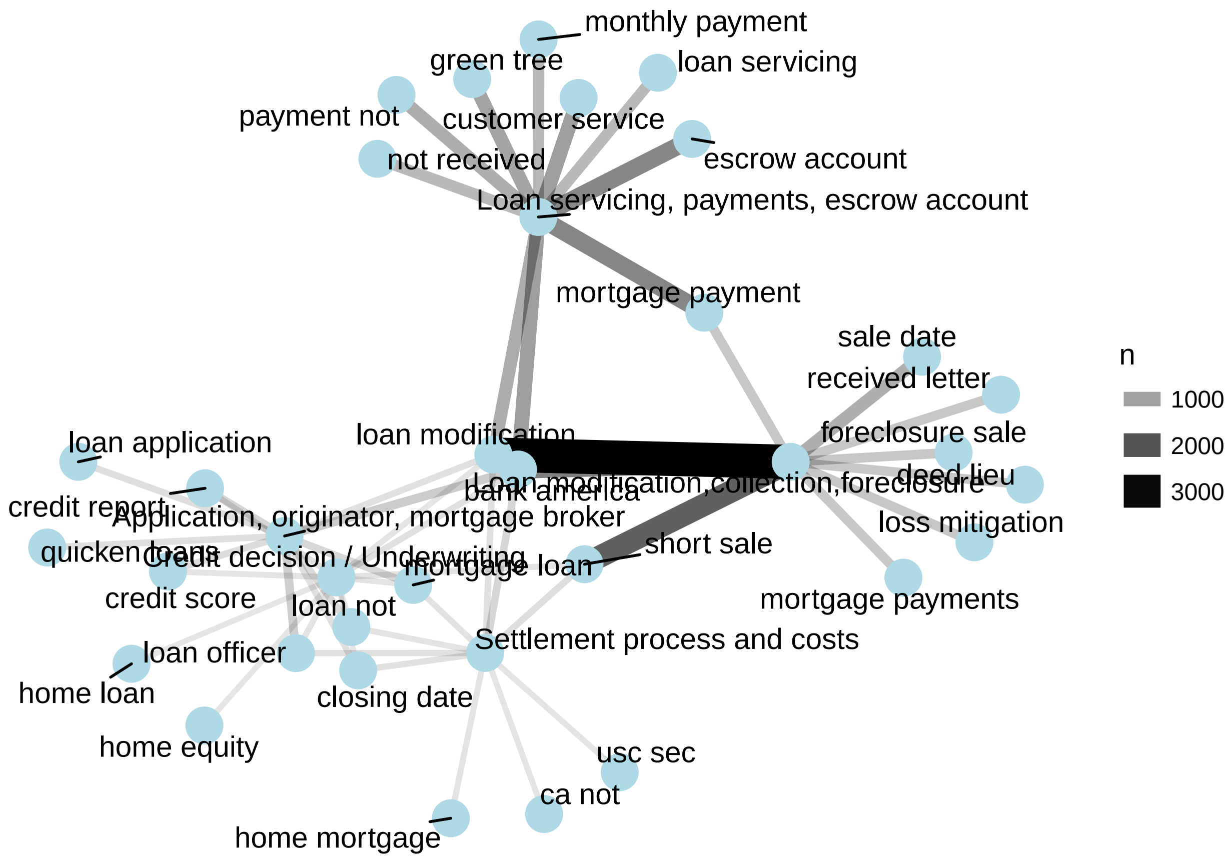
```
## Selecting by tf_idf
```

```
Mortgage_bigram_graph
```



From the above graph it is easy to identify the issues further. For example most complaints have bigram count for "inquires credit","pre approval" and "loan estimate" etc under the main issue Application,originator,mortgage broker.Similarly bigrams related to other issues can also be identified.Bigram gives more meaningful insight than the unigrams.Further it can be represented as a nodal graph as below for better visualization

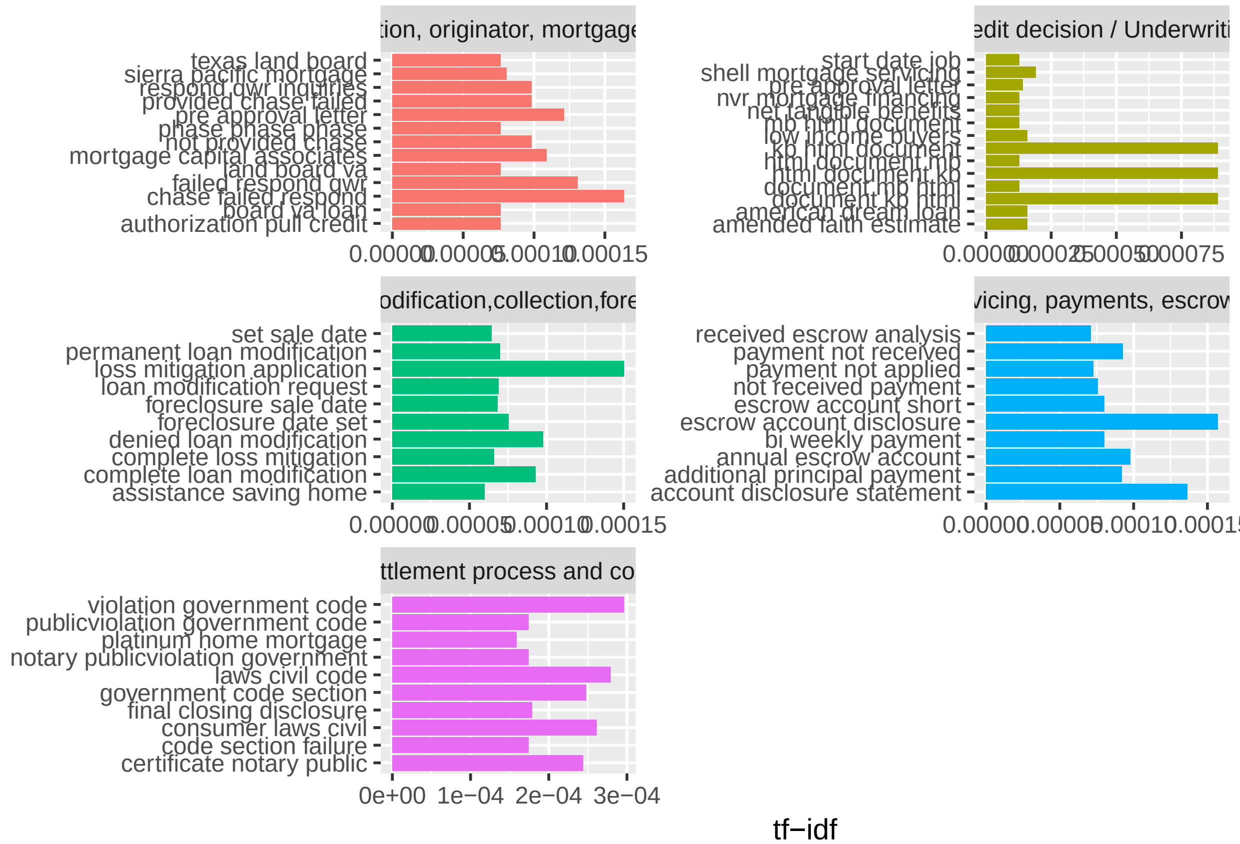
```r
bigram_count <- Mortgage_comments %>%
  group_by(issue)%>%
  unnest_tokens(bigram,consumer_complaint_narrative,token="ngrams",n=2) %>%
  count(issue,bigram)

bigram_count %>%
  group_by(issue)%>%
  #filter(n > 300) %>%
  top_n(10,n)%>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha = n, width = n)) +
  geom_node_point(size = 6, color = "lightblue") +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```




```r
Mortgage_trigram <- Mortgage_comments %>%
  group_by(issue)%>%
  unnest_tokens(trigram,consumer_complaint_narrative,token="ngrams",n=3) %>%
  count(issue,trigram)%>%
  bind_tf_idf(trigram,issue,n)%>%
  arrange(desc(tf_idf))%>%
  #mutate(bigram = factor(bigram, levels = rev(unique(word)))) %>%
  group_by(issue) %>%
  top_n(10) %>%
  ungroup %>%
  ggplot(aes(trigram, tf_idf, fill = issue)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~issue, ncol = 2, scales = "free") +
  coord_flip()
```

```
## Selecting by tf_idf
```

```r
Mortgage_trigram
```



Trigram provides more clear picture,according to the plot under settlement process and cost, "violation of government code" is the top issue and "escrow account analysis" is the to under servicing,payments,escrow accounts.

Correlations

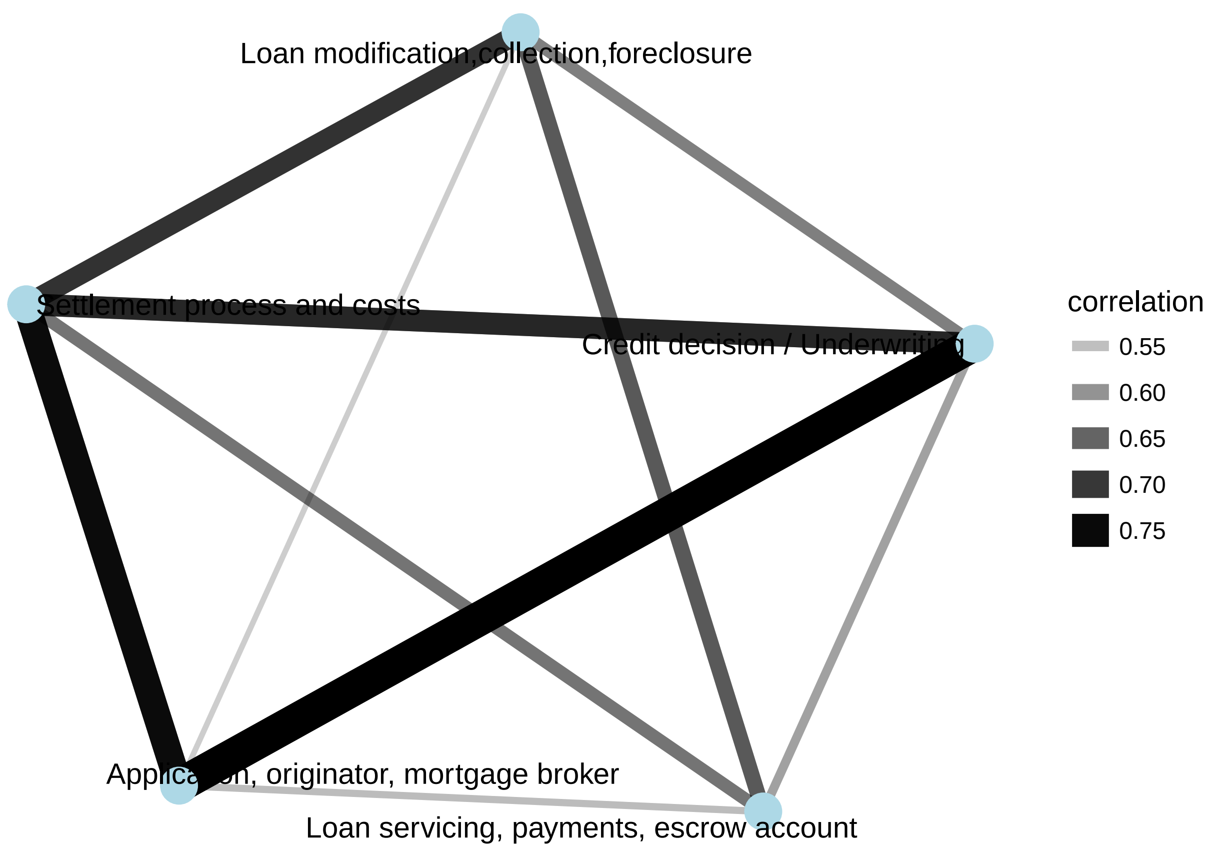
With the help of n-grams we can identify the correlations between the entities

```
cors <- Mortgage_comments %>%
  group_by(issue)%>%
  unnest_tokens(bigram,consumer_complaint_narrative,token="ngrams",n=2) %>%
  count(issue,bigram)%>%
  pairwise_cor(issue,bigram,n,sort=TRUE)

cors
```

```
## # A tibble: 20 x 3
##    item1                                  item2                  correl~
##    <fctr>                                 <fctr>                   <dbl>
##  1 Credit decision / Underwriting         Application, originat~   0.760
##  2 Application, originator, mortgage broker Credit decision / Und~   0.760
##  3 Settlement process and costs           Application, originat~   0.702
##  4 Application, originator, mortgage broker Settlement process an~   0.702
##  5 Settlement process and costs           Credit decision / Und~   0.652
##  6 Credit decision / Underwriting         Settlement process an~   0.652
##  7 Settlement process and costs           Loan modification,col~   0.635
##  8 Loan modification,collection,foreclosure Settlement process an~   0.635
```

```
## 9 Loan servicing, payments, escrow account Loan modification,col~   0.594
## 10 Loan modification,collection,foreclosure Loan servicing, payme~   0.594
## 11 Settlement process and costs             Loan servicing, payme~   0.572
## 12 Loan servicing, payments, escrow account Settlement process an~   0.572
## 13 Loan modification,collection,foreclosure Credit decision / Und~   0.563
## 14 Credit decision / Underwriting           Loan modification,col~   0.563
## 15 Loan servicing, payments, escrow account Credit decision / Und~   0.538
## 16 Credit decision / Underwriting           Loan servicing, payme~   0.538
## 17 Loan servicing, payments, escrow account Application, originat~   0.521
## 18 Application, originator, mortgage broker Loan servicing, payme~   0.521
## 19 Loan modification,collection,foreclosure Application, originat~   0.508
## 20 Application, originator, mortgage broker Loan modification,col~   0.508
```

```r
set.seed(2017)

Mortgage_cor_graph <- cors %>%
  filter(correlation > .5) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha = correlation, width = correlation)) +
  geom_node_point(size = 6, color = "lightblue") +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()

Mortgage_cor_graph
```



Credit decision underwriting and Application originator mortgage broker issues are higly correlated.

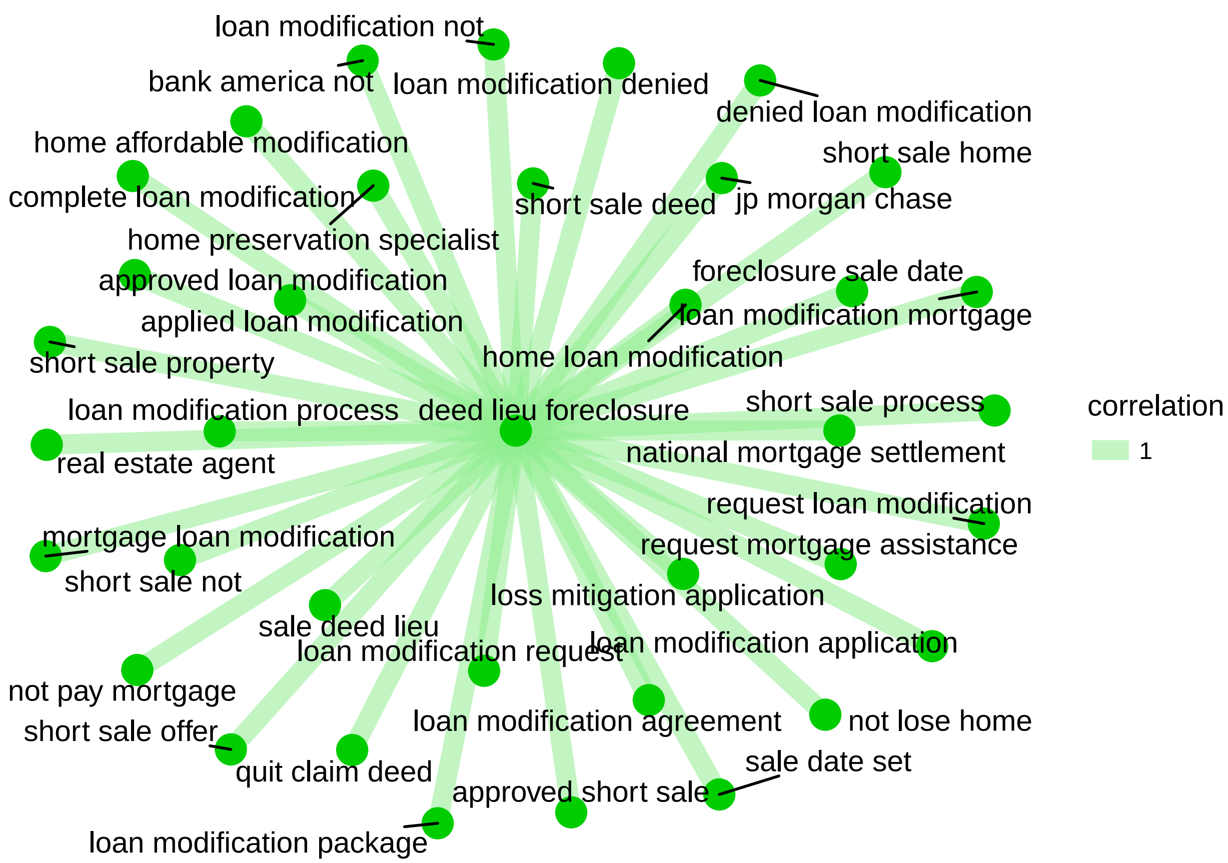
Correlation between trigrams

```
trigram_count <- Mortgage_comments %>%
  group_by(issue)%>%
  unnest_tokens(trigram,consumer_complaint_narrative,token="ngrams",n=3) %>%
  count(issue,trigram)%>%
  filter(n >50)


trigram_count$issue <- gsub("Application, originator, mortgage broker","1",trigram_count$issue)


trigram_count$issue <- gsub("Loan modification,collection,foreclosure","2",trigram_count$issue)

trigram_count$issue <- gsub("Loan servicing, payments, escrow account","3",trigram_count$issue)
trigram_count$issue <- gsub("Settlement process and costs","4",trigram_count$issue)

trigram_count$issue <- as.numeric(as.character(trigram_count$issue))



tri_cor <- trigram_count %>%
  pairwise_cor(trigram,issue,sort=TRUE)

tri_cor
```

```
## # A tibble: 6,162 x 3
##    item1                        item2                    correlation
##    <chr>                        <chr>                          <dbl>
##  1 approved loan modification   applied loan modification       1.000
##  2 approved short sale          applied loan modification       1.000
##  3 bank america not             applied loan modification       1.000
##  4 complete loan modification   applied loan modification       1.000
##  5 deed lieu foreclosure        applied loan modification       1.000
##  6 denied loan modification     applied loan modification       1.000
##  7 foreclosure sale date        applied loan modification       1.000
##  8 home affordable modification applied loan modification       1.000
##  9 home loan modification       applied loan modification       1.000
## 10 home preservation specialist applied loan modification       1.000
## # ... with 6,152 more rows
```

Further drilling down to each word,the correlation with other trigrams can also be found as below

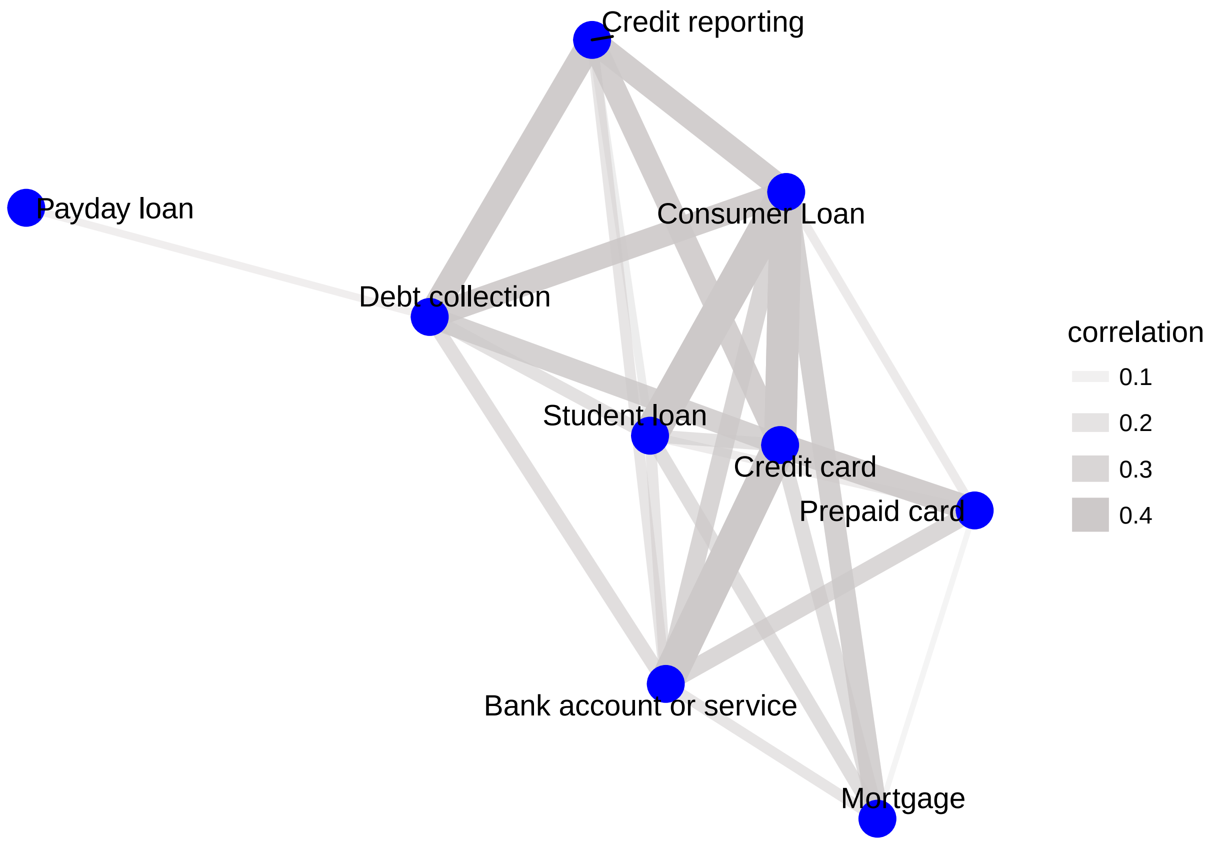
```
tri_cor%>%
  filter(item2 == "deed lieu foreclosure")%>%
 filter(correlation > 0.99)%>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha=correlation, width = correlation),colour = "lightgreen") +
  geom_node_point(size = 5, color = "green3") +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```



Correlations among Products and Issues

```
product_comments <- data%>%
  select(product,issue,consumer_complaint_narrative)

product_comments$consumer_complaint_narrative <- text.clean(product_comments$consumer_complaint_narrati

product_comments$consumer_complaint_narrative <- removeWords(product_comments$consumer_complaint_narrat

bigram_comment <- product_comments %>%
  #group_by(product,issue)%>%
  unnest_tokens(bigram,consumer_complaint_narrative,token="ngrams",n=2) %>%
  count(product,issue,bigram)%>%
  filter(n>75)
 # group_by(bigram)



prod_cor <- pairwise_cor(bigram_comment,product,bigram,sort = TRUE)

prod_cor %>%
  filter(correlation > 0) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
```

```r
  geom_edge_link(aes(alpha = correlation, width = correlation),colour="snow3") +
  geom_node_point(size = 6, color = "blue") +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```



Considering all the issues in the dataset,based on the bigram correlations can be identified.

```r
issue_cor <- pairwise_cor(bigram_comment,issue,bigram,sort = TRUE)

issue_cor %>%
  filter(correlation > .5) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(alpha=correlation, width = correlation),colour = "lightgreen") +
  geom_node_point(size = 5, color = "green3") +
  geom_node_text(aes(label = name), repel = TRUE) +
  theme_void()
```

Collocations - text2vec

Text2vec is another great package for NLP problems. By bigrams,two words occuring frequently similarly it is easy to extract collocations that can be used for Topic Modelling. The Vocabulary can be pruned as one lexicon or two lexicons,based on the dataset.Below study is based on the text2vec package [http://text2vec.org/api.html]

One Lexicon - Comments Analysis

```
tok_fun = word_tokenizer  # using word & not space tokenizers

itok = itoken( product_comments$consumer_complaint_narrative,
               tokenizer = tok_fun,
               progressbar = F)

onelex_vocab = create_vocabulary(itok,     #  func collects unique terms & corresponding statistics
                          ngram = c(1L, 1L))

onelex_pruned_vocab = prune_vocabulary(onelex_vocab,  # filters input vocab & throws out v frequent & v
                               term_count_min = 10)
onelex_model = Collocations$new(vocabulary=onelex_pruned_vocab,collocation_count_min = 50)

onelex_tok <-itoken(product_comments$consumer_complaint_narrative)
onelex_model$fit(onelex_tok, n_iter = 3)
```

```
## INFO [2018-01-23 12:57:40] iteration 1 - found 868 collocations
## INFO [2018-01-23 12:57:58] iteration 2 - found 1009 collocations
## INFO [2018-01-23 12:58:17] iteration 3 - found 1025 collocations
```

```r
onelex_stat <- onelex_model$collocation_stat

onelex_stat
```

```
##        prefix   suffix   n_i   n_j n_ij       pmi      lfmd       gensim
##    1: merrill    lynch    62    58   55 15.999037 -16.49795  5953.843159
##    2:     eos      cca   115   101   97 15.126066 -15.73382 17327.093241
##    3:    mini  miranda   127    94   89 14.962315 -16.14593 13988.788407
##    4:  hunter warfield   170   165  161 14.585058 -14.81282 16944.828663
##    5:    dodd    frank   144   160  132 14.582396 -15.38853 15239.771181
##   ---
## 1021:      st mortgage   354 31568   84  5.008391 -26.26668    13.027937
## 1022:  annual   escrow  1492  5978   67  5.007465 -26.92007     8.161524
## 1023:    week      ago  4166  4097  134  5.005799 -24.79104    20.140494
## 1024:  payoff   amount  2094 18916  297  5.004809 -22.62626    26.701646
## 1025:      no recourse 36978   362  105  5.004529 -25.49600    16.814609
##             llr
##    1: 1281.5568
##    2: 2135.4135
##    3: 1903.0113
##    4: 3494.6311
##    5: 2775.1971
##   ---
## 1021:  441.1557
## 1022:  338.8471
## 1023:  678.4954
## 1024: 1531.2308
## 1025:  557.3320
```

Two Lexicon - Comments Analysis

```r
twolex_vocab = create_vocabulary(itok,    #  func collects unique terms & corresponding statistics
                                ngram = c(2L, 2L))

twolex_pruned_vocab = prune_vocabulary(twolex_vocab,  # filters input vocab & throws out v frequent & v
                                      term_count_min = 10)
twolex_model = Collocations$new(vocabulary=twolex_pruned_vocab,collocation_count_min = 50)

twolex_tok <-itoken(product_comments$consumer_complaint_narrative)
twolex_model$fit(twolex_tok, n_iter = 3)
```

```
## INFO [2018-01-23 12:58:38] iteration 1 - found 262 collocations
## INFO [2018-01-23 12:59:01] iteration 2 - converged
```

```r
twolex_stat <- twolex_model$collocation_stat

twolex_stat
```

```
##                    prefix                 suffix  n_i  n_j n_ij
##   1:     estate_settlement settlement_procedures   61   59   59
##   2: settlement_procedures        procedures_act   59   62   59
##   3:           planet_home          home_lending   56   60   51
```

```
##   4:     central_financial     financial_control    69    64   60
##   5:   national_collegiate      collegiate_trust    75    61   59
##  ---
## 258:       debt_collection   collection_agencies  2448   626   65
## 259:         annual_credit         credit_report   287 19940  240
## 260:       debt_collection     collection_agency  2448  5058  503
## 261:        never_received       received_notice  3350   369   50
## 262:         credit_report            report_due 19940   101   78
##            pmi      lfmd      gensim       llr
##   1: 13.726119 -13.82231 2067.111698 1227.0029
##   2: 13.702660 -13.84577 2033.771186 1220.5839
##   3: 13.615036 -14.35383  246.016369 1006.2865
##   4: 13.455222 -14.04471 1871.863678 1180.3251
##   5: 13.379943 -14.16848 1626.127869 1149.2520
##  ---
## 258:  5.131817 -22.13716    8.091123  344.3962
## 259:  5.115462 -18.38447   27.444153 1536.9899
## 260:  5.069531 -16.29535   30.242051 2713.2732
## 261:  5.063290 -22.96271    0.000000  261.3639
## 262:  5.000669 -21.74224   11.492507  474.0814
```

Collocation - Mortgage

```r
itok_mortgage = itoken( Mortgage_comments$consumer_complaint_narrative,
               tokenizer = tok_fun,
               progressbar = F)
twolex_vocab_mt = create_vocabulary(itok_mortgage,    #  func collects unique terms & corresponding sta
                                ngram = c(2L, 2L))

twolex_pruned_vocab_mt = prune_vocabulary(twolex_vocab_mt,  # filters input vocab & throws out v freque
                                      term_count_min = 10)
twolex_model_mt = Collocations$new(vocabulary=twolex_pruned_vocab_mt,collocation_count_min = 50)

twolex_tok_mt <-itoken(Mortgage_comments$consumer_complaint_narrative)
twolex_model_mt$fit(twolex_tok_mt, n_iter = 3)
```

```
## INFO [2018-01-23 12:59:08] iteration 1 - found 60 collocations
## INFO [2018-01-23 12:59:16] iteration 2 - converged
```

```r
twolex_stat_mt <- twolex_model_mt$collocation_stat
twolex_stat_mt
```

```
##                    prefix                  suffix  n_i  n_j n_ij       pmi
##   1: settlement_procedures          procedures_act   59   61   59 11.769377
##   2:     estate_settlement   settlement_procedures   61   59   59 11.769377
##   3:           planet_home            home_lending   56   58   51 11.707204
##   4:    residential_credit       credit_solutions   93   80   78 11.124430
##   5:              makes_no                no_sense   69   81   52 10.952180
##   6:     national_mortgage     mortgage_settlement   97   93   83 10.936082
##   7:            quit_claim              claim_deed  124  115   98 10.515138
##   8:     home_preservation preservation_specialist  127   99   78 10.367475
##   9:             fair_debt         debt_collection   72  139   58 10.269229
##  10:           home_owners        owners_insurance  188   80   80 10.145526
##  11:              jp_morgan            morgan_chase  174  185  153  9.983177
```

```
## 12:     qualified_written         written_request  154  183  120  9.824519
## 13:           nation_star           star_mortgage  245   72   69  9.702076
## 14:    consumer_financial    financial_protection  250  238  228  9.672403
## 15:  financial_protection       protection_bureau  238  240  206  9.584907
## 16:        home_affordable affordable_modification  250  106   92  9.529972
## 17:       private_mortgage      mortgage_insurance   66  278   62  9.490976
## 18:          bsi_financial      financial_services  163  107   58  9.467900
## 19:           pre_approval         approval_letter  124  144   52  9.276433
## 20:           trial_period             period_plan  361   55   52  9.123340
## 21:       select_portfolio    portfolio_servicing  377  263  252  9.080055
## 22:       credit_reporting           reporting_act  382   58   56  9.072060
## 23:       credit_reporting      reporting_agencies  382  161  146  8.981593
## 24:    carrington_mortgage       mortgage_services  273  185  118  8.958619
## 25:            fair_credit        credit_reporting   63  382   56  8.952761
## 26:            caliber_home              home_loans  297  444  279  8.815501
## 27:             fixed_rate           rate_mortgage  279  110   58  8.652615
## 28:        loss_mitigation      mitigation_options  598   58   56  8.425487
## 29:            home_equity             equity_loan  470  246  184  8.404645
## 30:            real_estate       estate_settlement  653   61   61  8.349175
## 31:            real_estate            estate_taxes  653   65   65  8.349175
## 32:            real_estate            estate_agent  653  140  138  8.328417
## 33:        loss_mitigation  mitigation_application  598   94   84  8.313841
## 34:            home_equity             equity_line  470  289  195  8.256002
## 35:        loss_mitigation   mitigation_department  598   71   60  8.233256
## 36:              bank_home           home_mortgage   76  825   70  7.893220
## 37:             fargo_home           home_mortgage  418  825  380  7.874361
## 38:              days_past                past_due  146  811  129  7.857959
## 39:        numerous_phone             phone_calls   54  878   51  7.839575
## 40:               past_due              due_amount  811  107   90  7.786943
## 41:    nationstar_mortgage            mortgage_llc  861   96   73  7.555107
## 42:           bayview_loan         loan_servicing  149 1108  134  7.433293
## 43:             ocwen_loan         loan_servicing  605 1108  532  7.400863
## 44:         tree_servicing           servicing_llc  276  449   96  7.365981
## 45:           morgan_chase              chase_bank  185  437   59  7.279887
## 46:         escrow_account      account_disclosure 1452   55   53  7.142850
## 47:       customer_service      service_department 1478   75   72  7.111790
## 48:             green_tree          tree_servicing 1541  276  275  7.105227
## 49:       customer_service             service_rep 1478  134  127  7.093279
## 50:       specialized_loan          loan_servicing  170 1108  120  7.083872
## 51:       customer_service service_representative 1478  122  114  7.072837
## 52:         loan_servicing           servicing_llc 1108  449  313  7.065820
## 53:            received_no             no_response  211  404   50  6.964660
## 54:             short_sale           sale_approval 2397   50   50  6.473100
## 55:             short_sale            sale_process 2397  110  107  6.433208
## 56:       foreclosure_sale               sale_date  542  868  161  6.187331
## 57:       monthly_mortgage        mortgage_payment  513 1851  219  5.617997
## 58:         permanent_loan       loan_modification   71 4218   54  5.262912
## 59:       monthly_mortgage       mortgage_payments  513 1102  100  5.235247
## 60:                 ca_not              not_afford 1023  442   78  5.199024
##                     prefix                  suffix  n_i  n_j n_ij       pmi
##          lfmd     gensim       llr
##   1: -11.865566 532.507919 1066.9462
##   2: -11.865566 532.507919 1066.9462
##   3: -12.348174  65.561576  875.9367
```

```
##  4: -11.704995 801.402151 1289.3750
##  5: -13.047170  76.201109  786.2998
##  6: -11.714068 778.977054 1325.5026
##  7: -11.655671 716.782048 1478.2805
##  8: -12.461949 474.225086 1118.5691
##  9: -13.415038 170.219025  808.3559
## 10: -12.610847 424.755319 1165.4136
## 11: -10.902276 681.367878 2222.2526
## 12: -11.761929 528.922007 1637.2330
## 13: -13.481104 229.361451  930.5272
## 14: -10.062047 637.042555 3342.6315
## 15: -10.442321 581.569748 2887.7272
## 16: -12.823133 337.496151 1197.7406
## 17: -14.000861 139.270111  808.3029
## 18: -14.216367  97.675133  708.4051
## 19: -14.722917  23.851254  612.0454
## 20: -14.876010  21.449912  648.0990
## 21: -10.365614 433.830098 3329.9805
## 22: -14.713460  57.666727  699.4516
## 23: -12.038987 332.389581 1811.2879
## 24: -12.676324 286.708088 1389.9343
## 25: -14.832759  53.090002  672.9135
## 26: -10.336486 369.795371 3539.7064
## 27: -15.031653  55.508374  630.8780
## 28: -15.360032  36.837274  646.0337
## 29: -11.948460 246.795503 2056.8170
## 30: -15.189579  58.805111  711.9064
## 31: -15.006318  75.254094  759.0133
## 32: -12.854763 204.977817 1607.6917
## 33: -14.301753 128.799829  935.7315
## 34: -11.929566 227.320032 2115.9605
## 35: -15.353192  50.154035  650.0687
## 36: -15.248443  67.924721  741.6296
## 37: -10.386157 203.774163 4175.9847
## 38: -13.519815 142.075368 1353.8748
## 39: -16.215803   4.491352  539.9563
## 40: -14.629580  98.156885  919.4590
## 41: -15.465473  59.254162  705.3633
## 42: -13.834758 108.347443 1328.9122
## 43:  -9.888801 153.114849 5458.8673
## 44: -14.864323  79.043801  849.1391
## 45: -16.355056  23.705807  507.4274
## 46: -16.801539   7.999399  513.5259
## 47: -15.948589  42.262228  694.1185
## 48: -12.088427 112.651299 2749.4349
## 49: -14.329580  82.789813 1218.7643
## 50: -14.502576  79.136122 1069.9528
## 51: -14.661612  75.580736 1083.2506
## 52: -11.754372 112.573211 2839.7896
## 53: -17.147857   0.000000  402.6521
## 54: -17.639416   0.000000  449.7197
## 55: -15.484088  46.034088  937.5092
## 56: -14.551064  50.242284 1146.7107
## 57: -14.232658  37.899050 1409.5822
```

```
## 58: -18.627543   2.844202  346.7314
## 59: -16.877270  18.833735  559.8759
## 60: -17.630401  13.186378  430.1476
##           lfmd     gensim       llr
```

Topic Modelling

text2vec pacakge is used to incorporate collocations in topic models. But that is mostly used for unstructured data,this dataset already grouped based on product,issues and sub_issues this can be a trail to apply topic modelling on collocations.

```
topics = 10
vectorizer = vocab_vectorizer(twolex_pruned_vocab)
dtm = create_dtm(twolex_tok, vectorizer)
lda = LDA$new(topics)
doc_topic = lda$fit_transform(dtm)
```

```
## INFO [2018-01-23 12:59:27] iter 10 loglikelihood = -6490575.339
## INFO [2018-01-23 12:59:29] iter 20 loglikelihood = -6218055.542
## INFO [2018-01-23 12:59:30] iter 30 loglikelihood = -5990441.065
## INFO [2018-01-23 12:59:32] iter 40 loglikelihood = -5849519.516
## INFO [2018-01-23 12:59:34] iter 50 loglikelihood = -5764571.097
## INFO [2018-01-23 12:59:35] iter 60 loglikelihood = -5710970.334
## INFO [2018-01-23 12:59:37] iter 70 loglikelihood = -5675291.073
## INFO [2018-01-23 12:59:38] iter 80 loglikelihood = -5647759.373
## INFO [2018-01-23 12:59:40] iter 90 loglikelihood = -5627320.540
## INFO [2018-01-23 12:59:42] iter 100 loglikelihood = -5612561.780
## INFO [2018-01-23 12:59:43] iter 110 loglikelihood = -5601936.653
## INFO [2018-01-23 12:59:45] iter 120 loglikelihood = -5590880.798
## INFO [2018-01-23 12:59:46] iter 130 loglikelihood = -5586254.698
## INFO [2018-01-23 12:59:46] early stopping at 130 iteration
```

```
lda$get_top_words(n = 10, topic_number = c(1L, 5L, 10L), lambda = 0.2)
```

```
##       [,1]               [,2]                  [,3]
##  [1,] "mortgage_payment" "loan_modification"   "past_due"
##  [2,] "monthly_payment"  "short_sale"          "late_fees"
##  [3,] "escrow_account"   "home_mortgage"       "late_fee"
##  [4,] "loan_servicing"   "sale_date"           "due_date"
##  [5,] "monthly_payments" "nationstar_mortgage" "chase_bank"
##  [6,] "property_taxes"   "home_loan"           "late_payment"
##  [7,] "ocwen_loan"       "real_estate"         "late_payments"
##  [8,] "monthly_mortgage" "fargo_bank"          "days_late"
##  [9,] "servicing_llc"    "not_qualify"         "payment_due"
## [10,] "loan_amount"      "loss_mitigation"     "credit_union"
```

```
lda$plot()

serVis(lda$plot())
```

Reference

1.[https://www.tidytextmining.com/] 2.[http://text2vec.org/api.html]