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()
## )
```

```
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.

```
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)
```

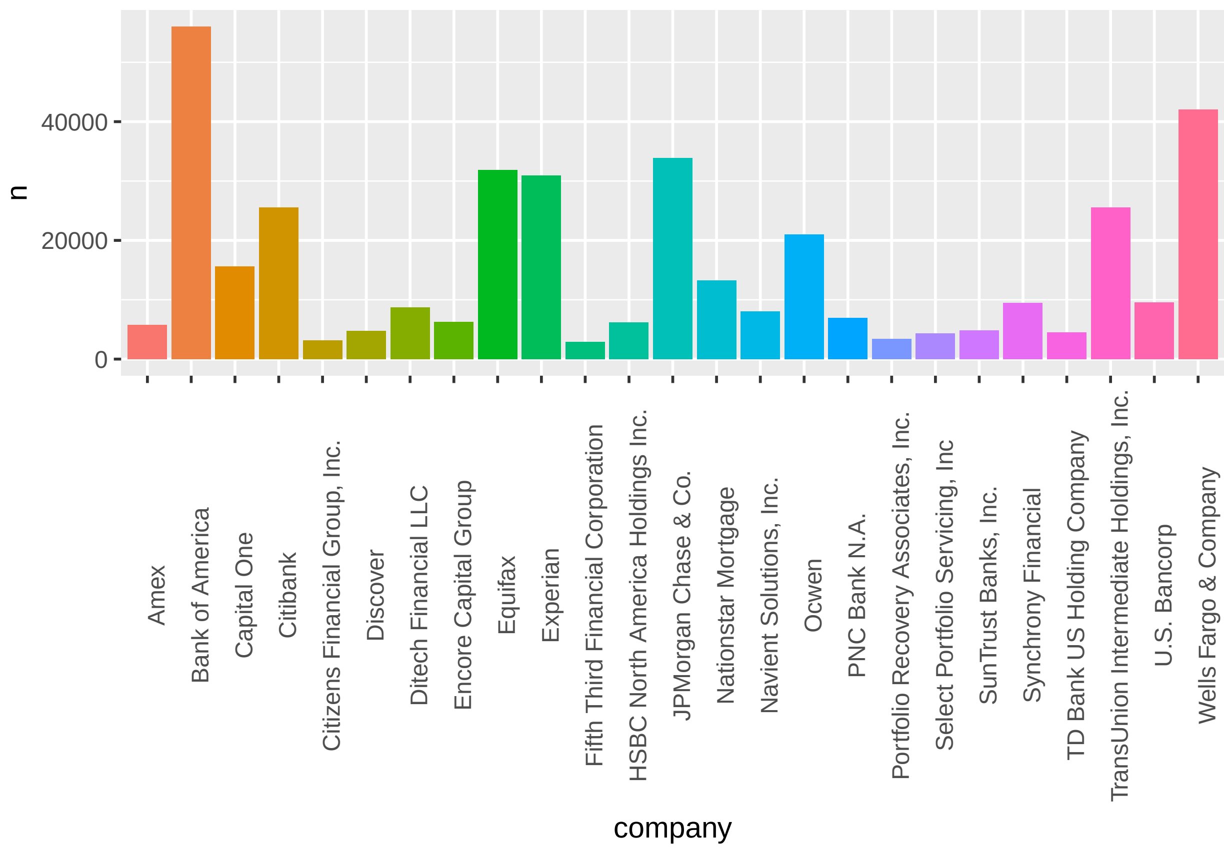
Exploratory Data Analysis

```
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

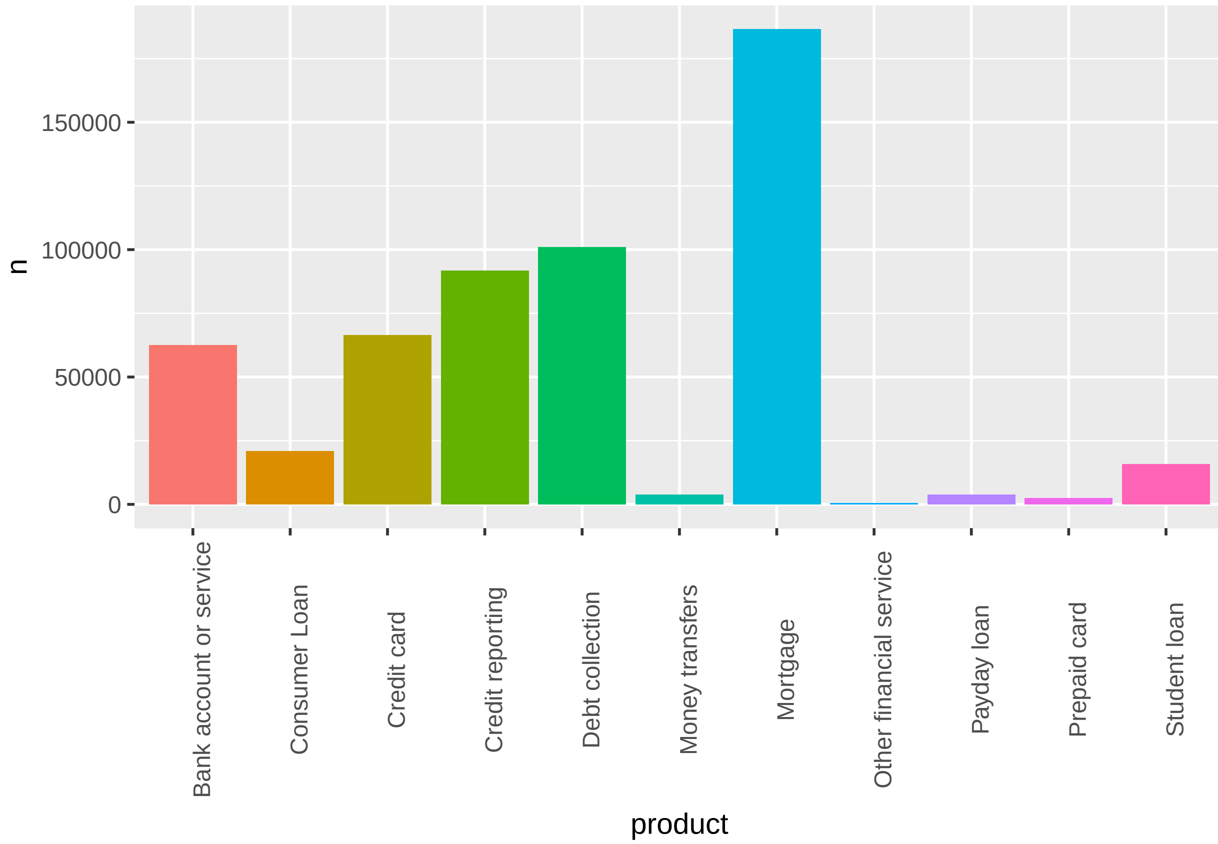
```
complaint2 %>%
  count(company) %>%
  arrange(desc(n))%>%
  top_n(25) %>%
  ggplot(aes(company,n,fill=company))+
  geom_bar(stat="identity") +
  theme(axis.text.x=element_text(angle=90),legend.position = "none")
```

```
## 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")
```

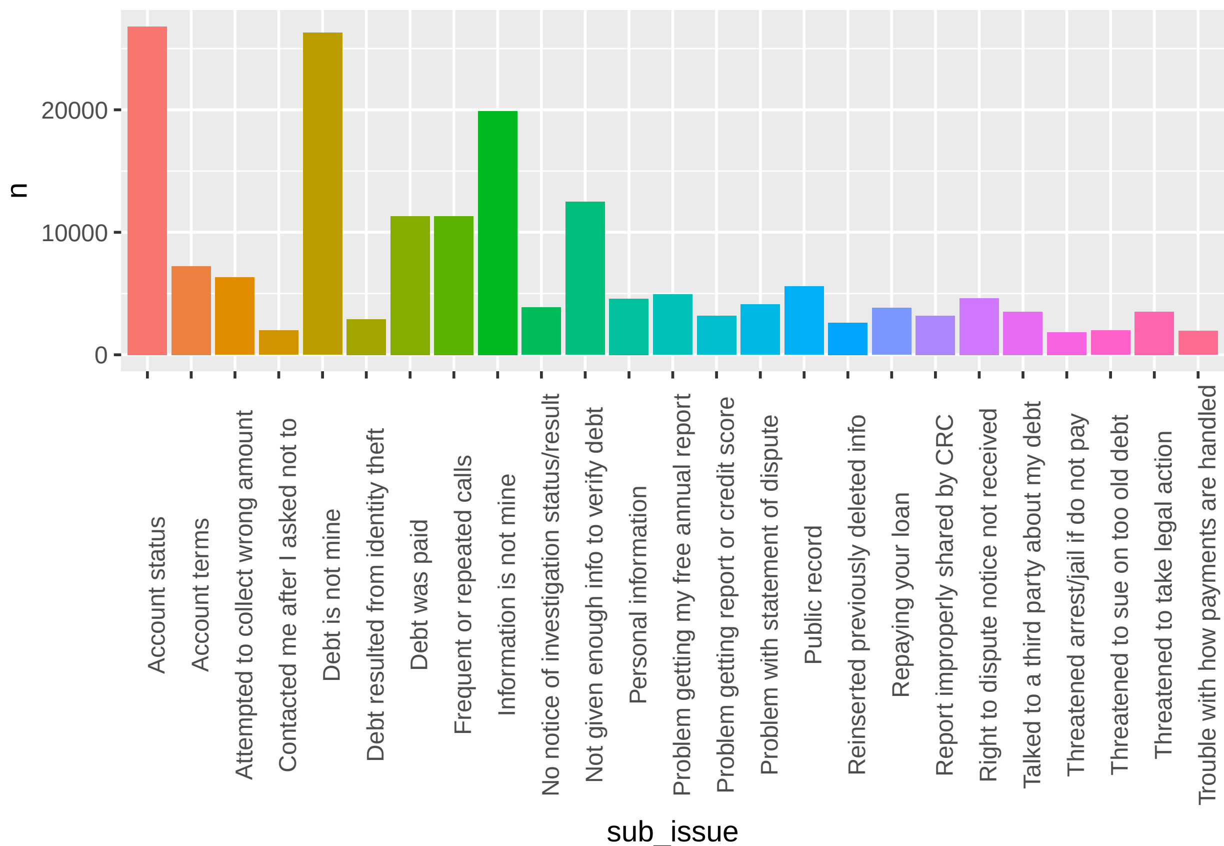



```
sub_issue_data <- complaint2 %>%
select(complaint_id,sub_issue)%>%
na.omit()
```

Complaint numbers based on the Issue categories

```
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")
```

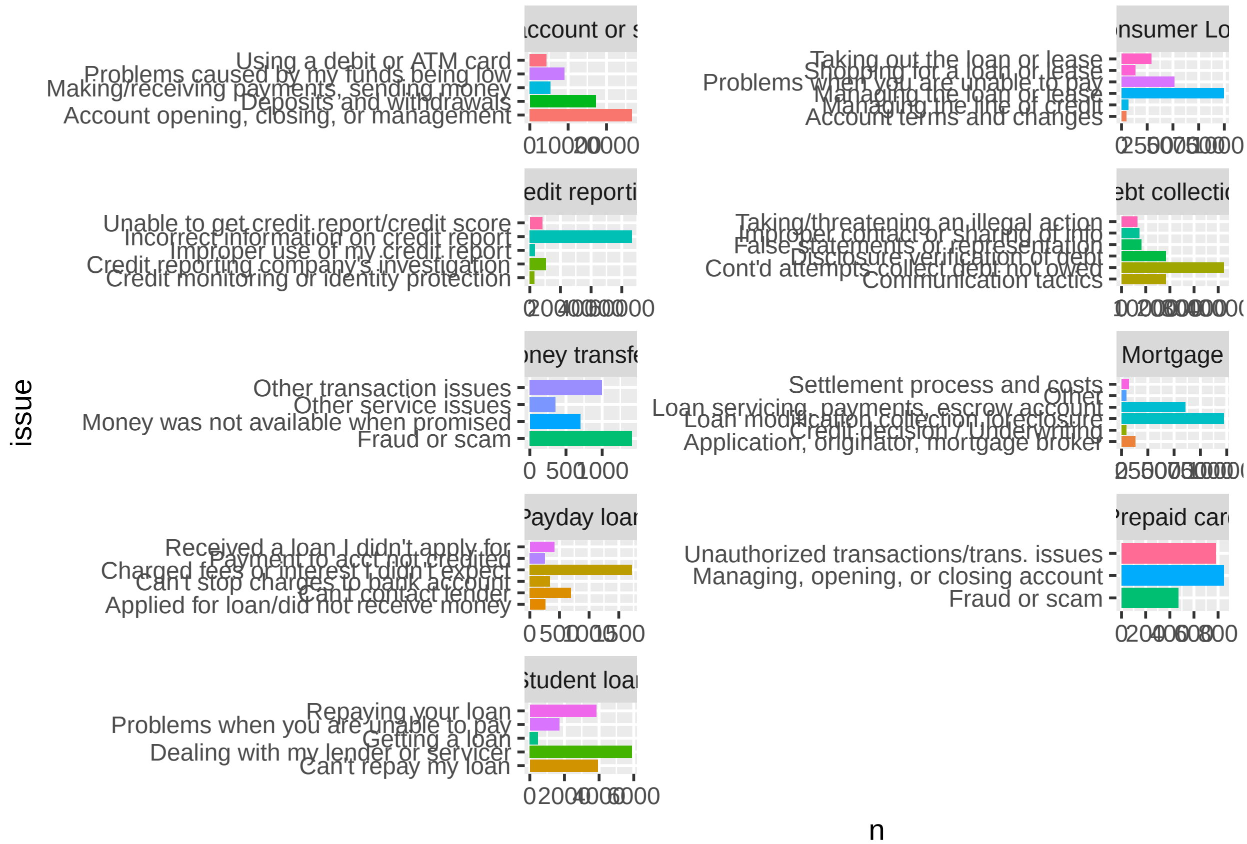
```
## Selecting by n
```




```r
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

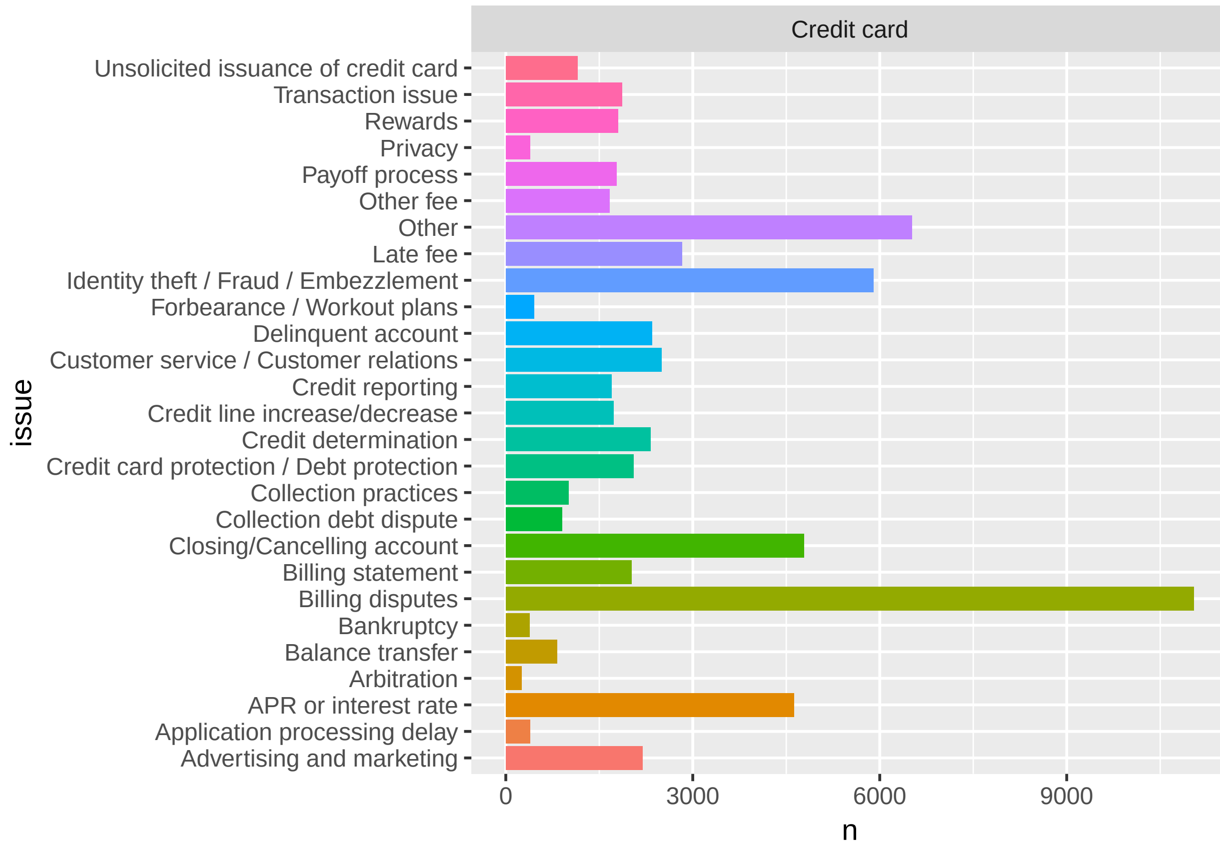
```r
product_issue %>%
  filter(n>250)%>%
  filter(product !="Credit card") %>%
  ggplot(aes(issue,n,fill=issue))+
  geom_bar(stat="identity")+
 theme(legend.position = "none")+
 facet_wrap(~product,scale= "free",nrow = 6)+
  coord_flip()
```



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.

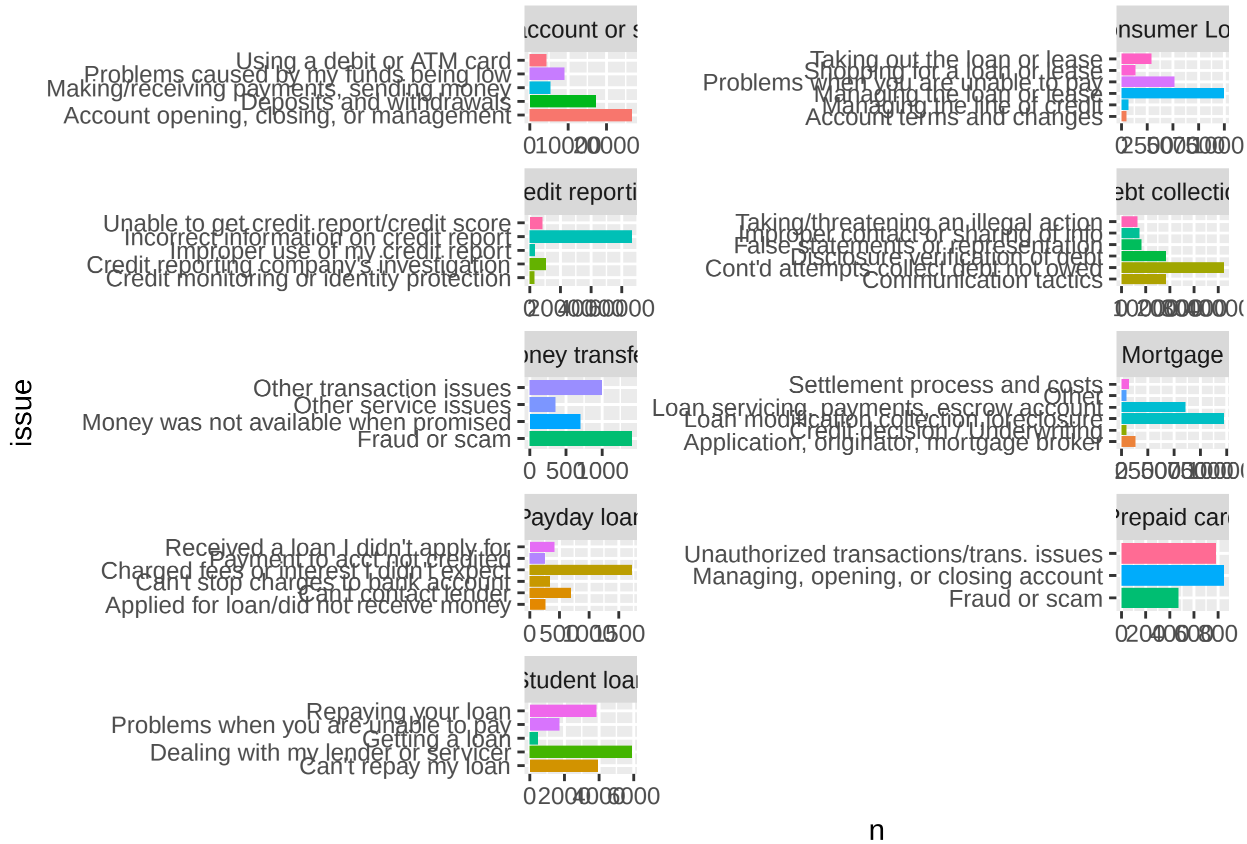
Complaint category under Credit card

```
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()
```



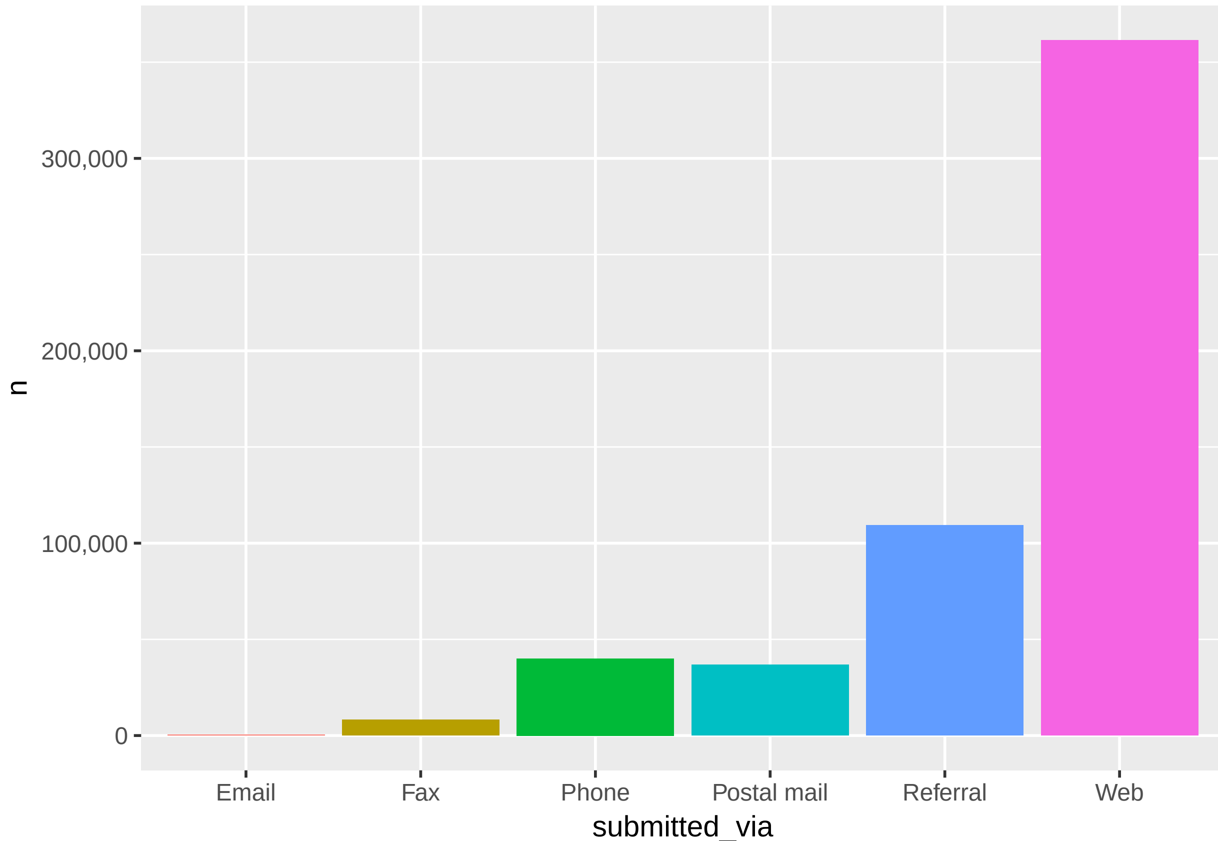
####Complaint category for products other than credit card

```
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")
```

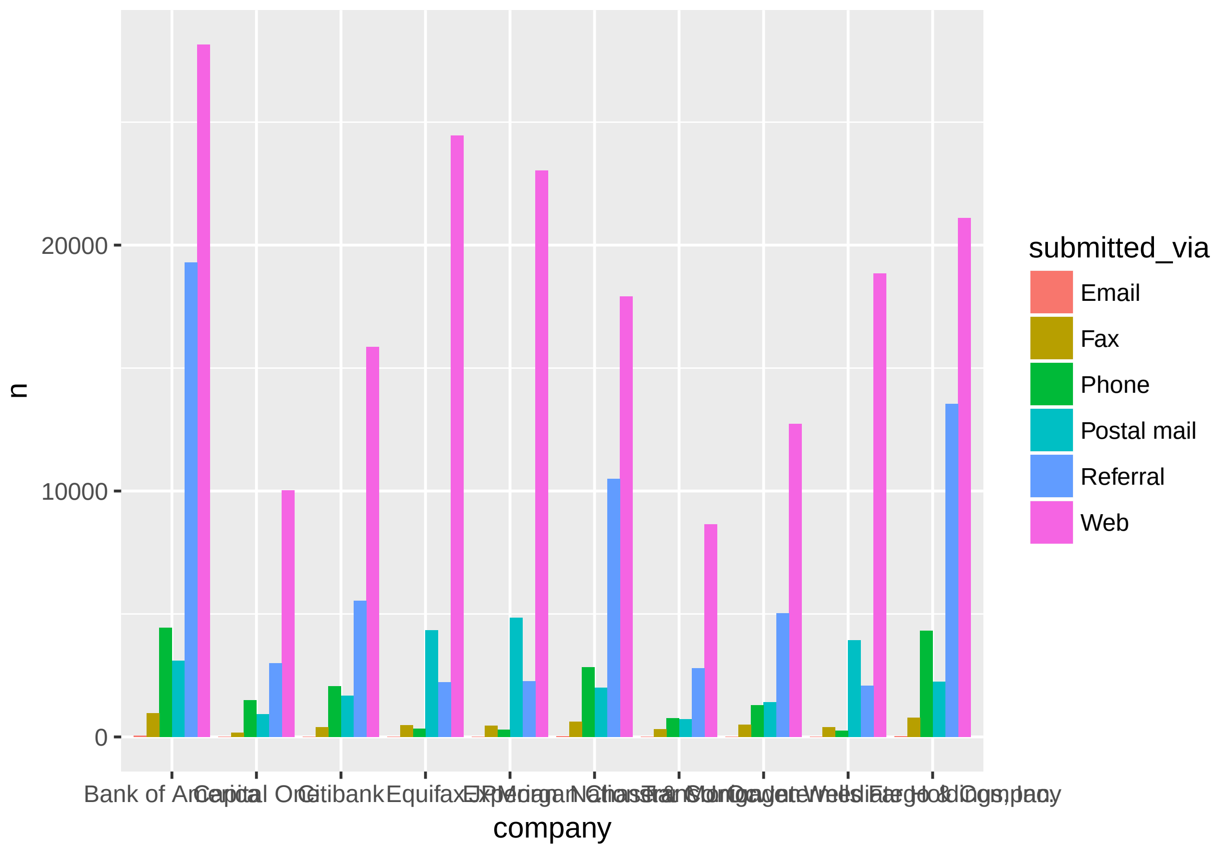



```
top_companies <- complaint2 %>%
  count(company) %>%
  arrange(desc(n))%>%
  top_n(10)
```

```
## 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

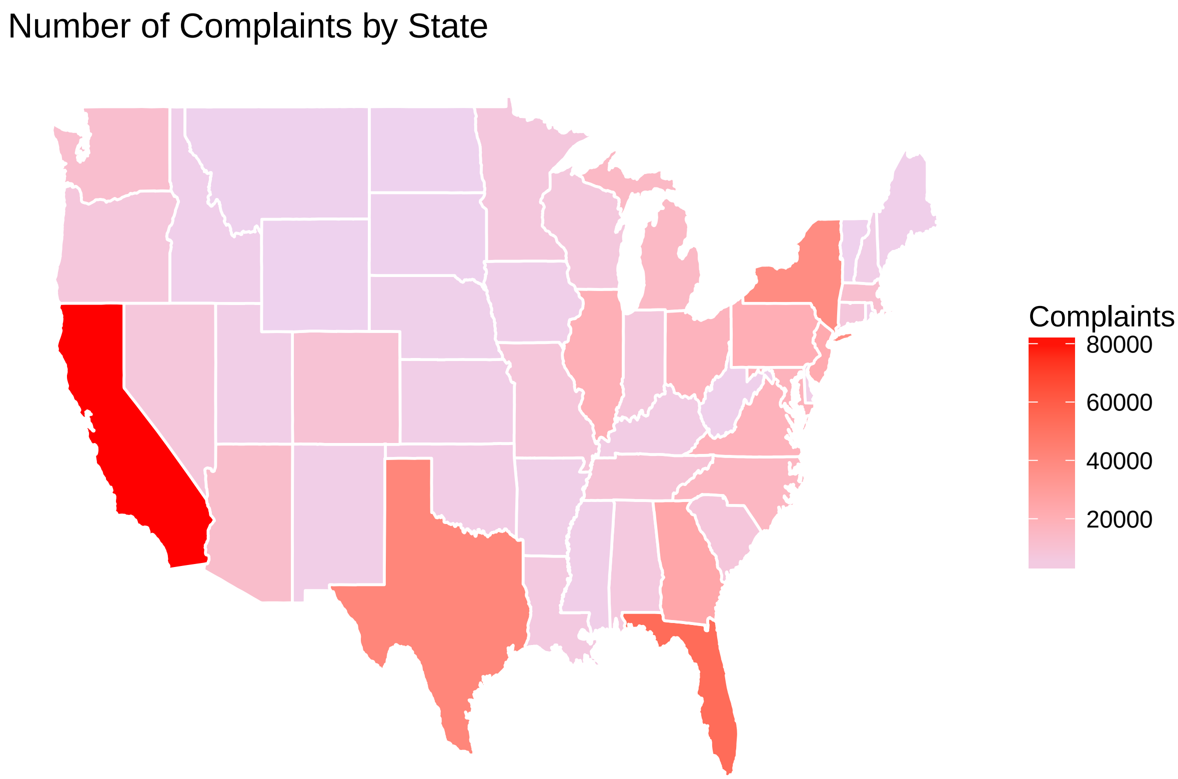
As there is a state information from which the complaints was received, distribution of complaints over United States can be visualized by map packages.

```
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)
```

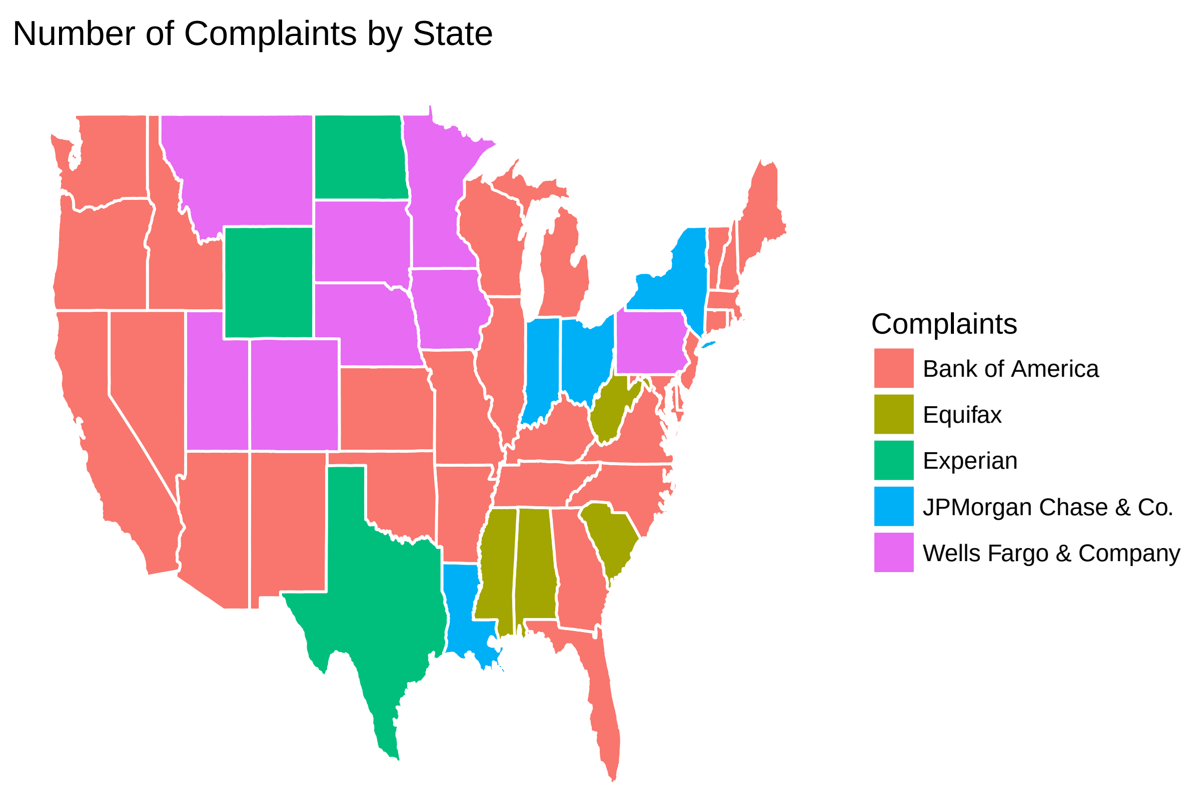
```
map_state <- merge(all_states,map_complaint,by="region")
map_state<- map_state[map_state$region!="district of columbia",]

    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())
```



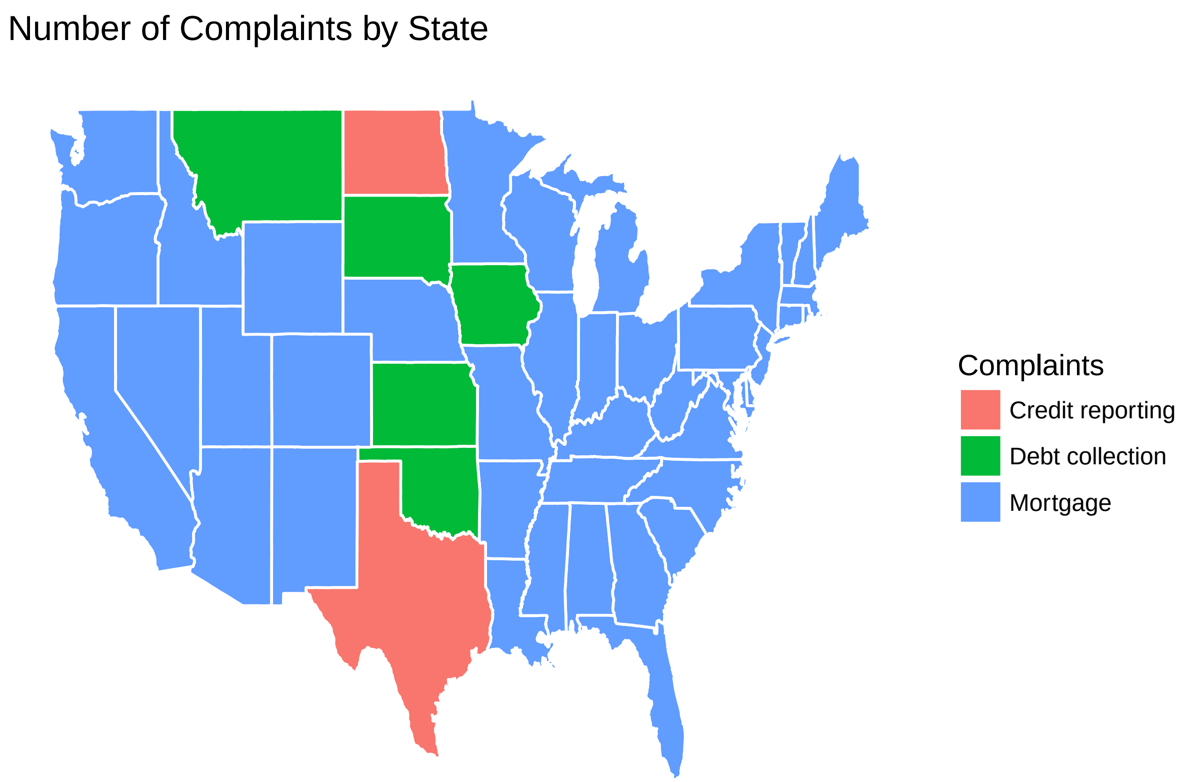
Companies with highest number of complaints distribution based on state

```
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))
```

```r
}
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

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

  return(d4)
   }
```

Companies timely response

```r
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>
```

```
data3 <- data%>%
  na.omit() %>%
  select(company,consumer_complaint_narrative) %>%
  filter(company =='Equifax')

text.clean = function(x)                    # text data
{ require("tm")
  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)
```

```
}



data3$id <- seq.int(nrow(data3))

stp <- tm::stopwords('english')
stp1 <- c("xxxx","xxx","xxxxx","xx","x","company","companies","said","told",
          "however","since","asked","stated","equifax","well","item","items","done",
          "going","n_t")
comn  = unique(c(stp, stp1) )                 # Union of two list
stopwords = unique(gsub("'"," ",comn) )


x= text.clean(data3$consumer_complaint_narrative)
x  =  removeWords(x,stopwords)            # removing stopwords created above
x  =  stripWhitespace(x )                  # removing white spac




tok_fun = word_tokenizer  # using word & not space tokenizers

it_0 = itoken( x,
               #preprocessor = text.clean,
               tokenizer = tok_fun,
               ids = data3$id,
               progressbar = F)

vocab = create_vocabulary(it_0,    #  func collects unique terms & corresponding statistics
                          ngram = c(2L, 2L) #,
                          #stopwords = stopwords
)

pruned_vocab = prune_vocabulary(vocab,  # filters input vocab & throws out v frequent & v infrequent te
                                term_count_min = 10)

length(pruned_vocab);  str(pruned_vocab)
```

```
## [1] 3

## Classes 'text2vec_vocabulary' and 'data.frame':  2299 obs. of  3 variables:
##  $ term      : chr  "years_removed" "hit_credit" "bankruptcy_per" "times_time" ...
##  $ term_count: int  10 10 10 10 10 10 10 10 10 10 ...
##  $ doc_count : int  9 8 9 10 3 10 10 9 10 8 ...
##  - attr(*, "ngram")= Named int  2 2
##   ..- attr(*, "names")= chr  "ngram_min" "ngram_max"
##  - attr(*, "document_count")= int 4187
##  - attr(*, "stopwords")= chr
##  - attr(*, "sep_ngram")= chr "_"
```

```
vectorizer = vocab_vectorizer(pruned_vocab) #  creates a text vectorizer func used in constructing a dt

dtm_0  = create_dtm(it_0, vectorizer) # high-level function for creating a document-term matrix
```

```
# Sort bi-gram with decreasing order of freq
tsum = as.matrix(t(rollup(dtm_0, 1, na.rm=TRUE, FUN = sum))) # find sum of freq for each term
tsum = tsum[order(tsum, decreasing = T),]       # terms in decreasing order of freq
head(tsum)
```

```
##    credit_report              n_t credit_reporting   credit_bureaus
##             3444             1224              832              598
##      credit_file     credit_score
##              575              527
```

```
text2 = x
text2 = paste("",text2,"")

pb <- txtProgressBar(min = 1, max = (length(tsum)), style = 3) ; i = 0

for (term in names(tsum)){
  i = i + 1
  focal.term = gsub("_", " ",term)        # in case dot was word-separator
  replacement.term = term
  text2 = gsub(paste("",focal.term,""),paste("",replacement.term,""), text2)
  # setTxtProgressBar(pb, i)
}


it_m = itoken(text2,     # function creates iterators over input objects to vocabularies, corpora, DTM &
              # preprocessor = text.clean,
              tokenizer = tok_fun,
              ids = data$id,
              progressbar = F)
```

```
## Warning: Unknown or uninitialised column: 'id'.
```

```
vocab = create_vocabulary(it_m     # vocab func collects unique terms and corresponding statistics
                          # ngram = c(2L, 2L),
                          #stopwords = stopwords
)


pruned_vocab = prune_vocabulary(vocab,
                                term_count_min = 1)
vectorizer = vocab_vectorizer(pruned_vocab)

dtm_m  = create_dtm(it_m, vectorizer)
dim(dtm_m)
```

```
## [1]  4187 11753
```

```
dtm = as.DocumentTermMatrix(dtm_m, weighting = weightTf)
a0 = (apply(dtm, 1, sum) > 0)   # build vector to identify non-empty docs
dtm = dtm[a0,]                  # drop empty docs

dtm = dtm[,order(apply(dtm, 2, sum), decreasing = T)]     # sorting dtm's columns in decreasing order of
inspect(dtm[1:5, 1:5])     # inspect() func used to view parts of a DTM object
```

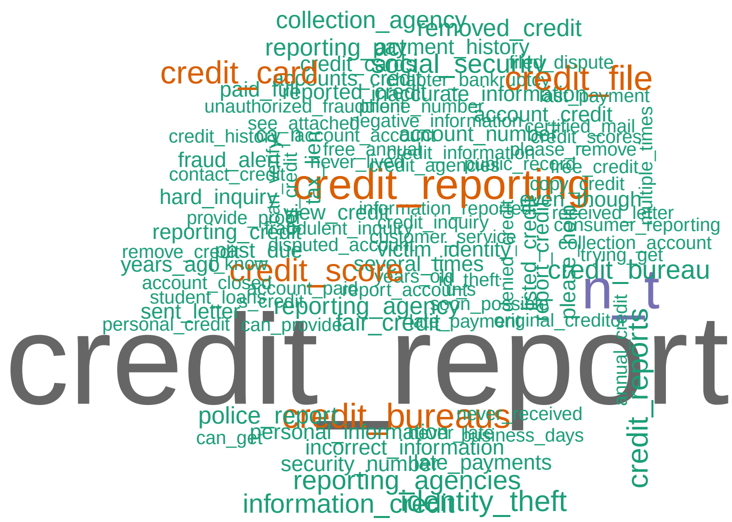
```
## <<DocumentTermMatrix (documents: 5, terms: 5)>>
## Non-/sparse entries: 6/19
```

```
## Sparsity           : 76%
## Maximal term length: 13
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs account credit_report information n_t report
##    1       0             1           0   1      0
##    2       2             0           0   0      0
##    3       1             0           1   0      0
##    4       0             3           0   0      0
##    5       0             0           0   0      0
```

```
#---------------------------------------------------------#
## Step 2a:     # Build word cloud                        #
#---------------------------------------------------------#

#   1- Using Term frequency(tf)

tst = round(ncol(dtm_0)/100)  # divide DTM's cols into 100 manageble parts
a = rep(tst,99)
b = cumsum(a);rm(a)
b = c(0,b,ncol(dtm_0))

ss.col = c(NULL)
for (i in 1:(length(b)-1)) {
  tempdtm = dtm_0[,(b[i]+1):(b[i+1])]
  s = colSums(as.matrix(tempdtm))
  ss.col = c(ss.col,s)
}

tsum = ss.col
tsum = tsum[order(tsum, decreasing = T)]       #terms in decreasing order of freq
head(tsum)
```

```
##    credit_report              n_t credit_reporting   credit_bureaus
##             3444             1224              832              598
##      credit_file     credit_score
##              575              527
```

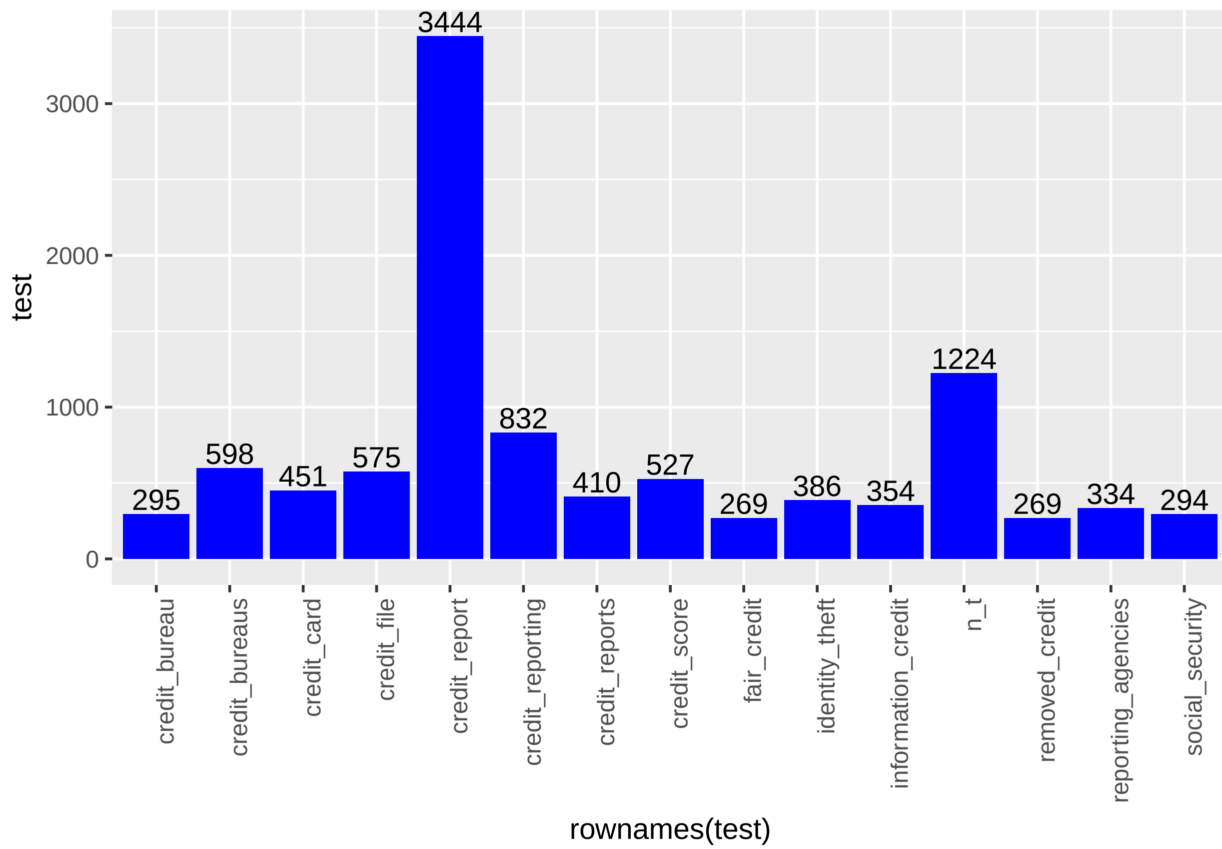
```
windows()  # New plot window
wordcloud(names(tsum), tsum,     # words, their freqs
          scale = c(4, 0.5),     # range of word sizes
          1,                     # min.freq of words to consider
          max.words = 100,       # max #words
          colors = brewer.pal(8, "Dark2"))    # Plot results in a word cloud
title(sub = "Term Frequency - Wordcloud")     # title for the wordcloud display
```



Term Frequency − Wordcloud

```r
# plot barchart for top tokens
test = as.data.frame(round(tsum[1:15],0))

windows()  # New plot window
ggplot(test, aes(x = rownames(test), y = test)) +
  geom_bar(stat = "identity", fill = "Blue") +
  geom_text(aes(label = test), vjust= -0.20) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type data.frame. Defaulting to continuous.
```




```
dtm.tfidf = tfidf(dtm, normalize= FALSE)

tst = round(ncol(dtm.tfidf)/100)
a = rep(tst, 99)
b = cumsum(a);rm(a)
b = c(0,b,ncol(dtm.tfidf))

ss.col = c(NULL)
for (i in 1:(length(b)-1)) {
  tempdtm = dtm.tfidf[,(b[i]+1):(b[i+1])]
  s = colSums(as.matrix(tempdtm))
  ss.col = c(ss.col,s)

}

tsum = ss.col

tsum = tsum[order(tsum, decreasing = T)]        #terms in decreasing order of freq
head(tsum)
```

```
## credit_report           n_t       account   information        report
##      2514.434      2109.606      2027.938      1727.030      1649.457
##      accounts
##      1633.383
```

```
windows()
wordcloud(names(tsum), tsum, scale=c(4,0.5),1, max.words=100,colors=brewer.pal(8, "Dark2")) # Plot resu
```

```
## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : credit_report could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : credit_file could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : name could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : years could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : address could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : dispute could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : request could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : time could not be fit on page. It will not be plotted.

## Warning in wordcloud(names(tsum), tsum, scale = c(4, 0.5), 1, max.words =
## 100, : remove could not be fit on page. It will not be plotted.
```

```r
title(sub = "Term Frequency Inverse Document Frequency - Wordcloud")
```

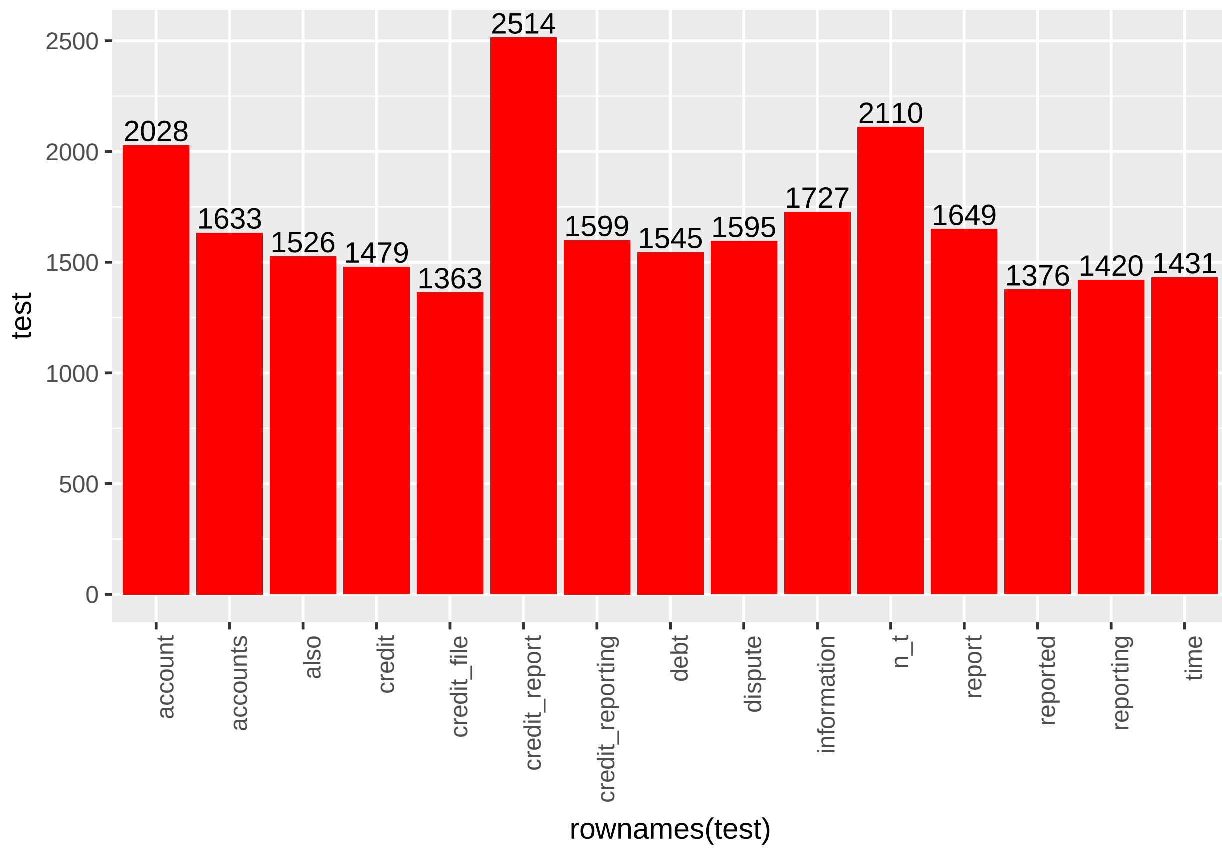
Term Frequency Inverse Document Frequency − Wordcloud

```r
as.matrix(tsum[1:20])      #  to see the top few tokens & their IDF scores
```

```
##                      [,1]
## credit_report    2514.434
## n_t              2109.606
## account          2027.938
## information      1727.030
## report           1649.457
## accounts         1633.383
## credit_reporting 1599.378
## dispute          1595.007
## debt             1544.974
## also             1525.817
## credit           1479.474
## time             1431.452
## reporting        1420.179
## reported         1376.156
## credit_file      1363.344
## credit_bureaus   1350.950
## can              1334.715
## removed          1326.454
## s                1319.873
## name             1288.828
```

```r
# plot barchart for top tokens
test = as.data.frame(round(tsum[1:15],0))
windows()  # New plot window
ggplot(test, aes(x = rownames(test), y = test)) +
  geom_bar(stat = "identity", fill = "red") +
  geom_text(aes(label = test), vjust= -0.20) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type data.frame. Defaulting to continuous.
```




```
vectorizer = vocab_vectorizer(pruned_vocab,
                              grow_dtm = FALSE,
                              skip_grams_window = 5L)
```

```
## Error in vocab_vectorizer(pruned_vocab, grow_dtm = FALSE, skip_grams_window = 5L): unused arguments
```

```
tcm = create_tcm(it_m, vectorizer) # func to build a TCM

tcm.mat = as.matrix(tcm)         # use tcm.mat[1:5, 1:5] to view
adj.mat = tcm.mat + t(tcm.mat)   # since adjacency matrices are symmetric

z = order(colSums(adj.mat), decreasing = T)
adj.mat = adj.mat[z,z]

# Plot Simple Term Co-occurance graph
adj = adj.mat[1:30,1:30]

library(igraph)
```

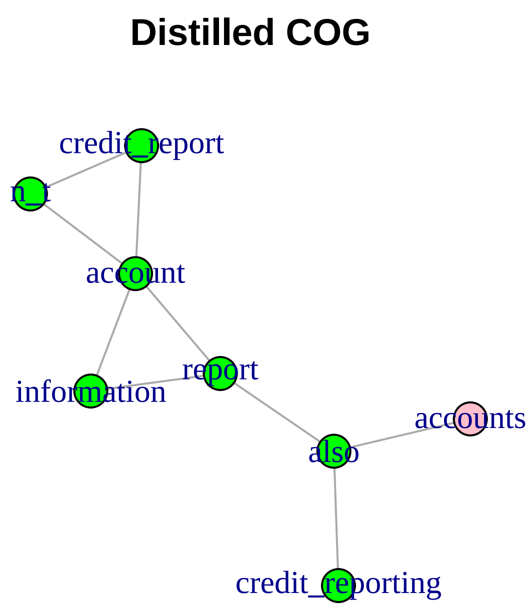
```
##
## Attaching package: 'igraph'

## The following object is masked from 'package:text2vec':
##
##     normalize

## The following objects are masked from 'package:qdap':
##
```

```
##     %>%, diversity
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
## The following object is masked from 'package:tibble':
##
##     as_data_frame
## The following object is masked from 'package:tidyr':
##
##     crossing
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
## The following object is masked from 'package:base':
##
##     union
```

```
distill.cog = function(mat1, # input TCM ADJ MAT
                       title, # title for the graph
                       s,    # no. of central nodes
                       k1){  # max no. of connections
  library(igraph)
  a = colSums(mat1) # collect colsums into a vector obj a
  b = order(-a)     # nice syntax for ordering vector in decr order

  mat2 = mat1[b, b]     # order both rows and columns along vector b

  diag(mat2) =  0

  ## +++ go row by row and find top k adjacencies +++ ##

  wc = NULL

  for (i1 in 1:s){
    thresh1 = mat2[i1,][order(-mat2[i1, ])[k1]]
    mat2[i1, mat2[i1,] < thresh1] = 0
    mat2[i1, mat2[i1,] > 0 ] = 1
    word = names(mat2[i1, mat2[i1,] > 0])
    mat2[(i1+1):nrow(mat2), match(word,colnames(mat2))] = 0
    wc = c(wc,word)
  } # i1 loop ends


  mat3 = mat2[match(wc, colnames(mat2)), match(wc, colnames(mat2))]
  ord = colnames(mat2)[which(!is.na(match(colnames(mat2), colnames(mat3))))]  # removed any NAs from th
```

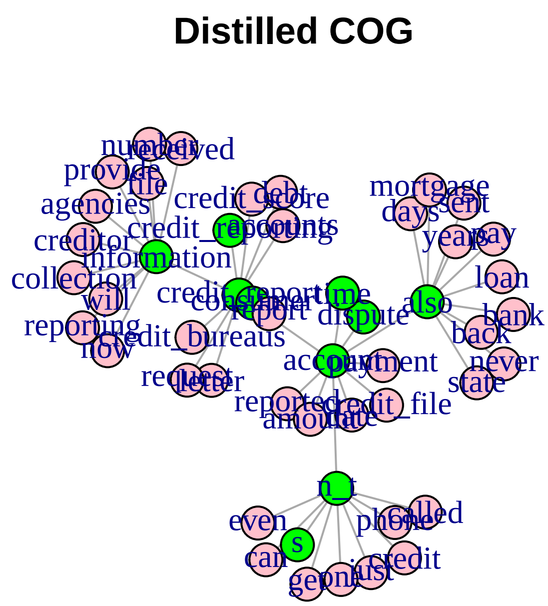
```
  mat4 = mat3[match(ord, colnames(mat3)), match(ord, colnames(mat3))]
  graph <- graph.adjacency(mat4, mode = "undirected", weighted=T)    # Create Network object
  graph = simplify(graph)
  V(graph)$color[1:s] = "green"
  V(graph)$color[(s+1):length(V(graph))] = "pink"

  graph = delete.vertices(graph, V(graph)[ degree(graph) == 0 ])

  plot(graph,
       layout = layout.kamada.kawai,
       main = title)

} # func ends

windows()
distill.cog(tcm.mat, 'Distilled COG',  10,  5)
```

```
## Warning in vattrs[[name]][index] <- value: number of items to replace is
## not a multiple of replacement length

## Warning in vattrs[[name]][index] <- value: number of items to replace is
## not a multiple of replacement length
```




```
## adj.mat and distilled cog for tfidf DTMs ##

adj.mat = t(dtm.tfidf) %*% dtm.tfidf
```

```r
diag(adj.mat) = 0
a1 = order(apply(adj.mat, 2, sum), decreasing = T)
adj.mat = as.matrix(adj.mat[a1[1:50], a1[1:50]])

windows()
distill.cog(adj.mat, 'Distilled COG',  10,  10)
```




```r
#--------------------------------------------------------#
#             Sentiment Analysis                         #
#--------------------------------------------------------#

library(qdap)

x1 = x[a0]     # remove empty docs from corpus

t1 = Sys.time()   # set timer

pol = polarity(x1)         # Calculate the polarity from qdap dictionary
wc = pol$all[,2]                  # Word Count in each doc
val = pol$all[,3]                 # average polarity score
p  = pol$all[,4]                  # Positive words info
n  = pol$all[,5]                  # Negative Words info

dim(pol)
```

```
## NULL
```

```r
head(pol$group)
```

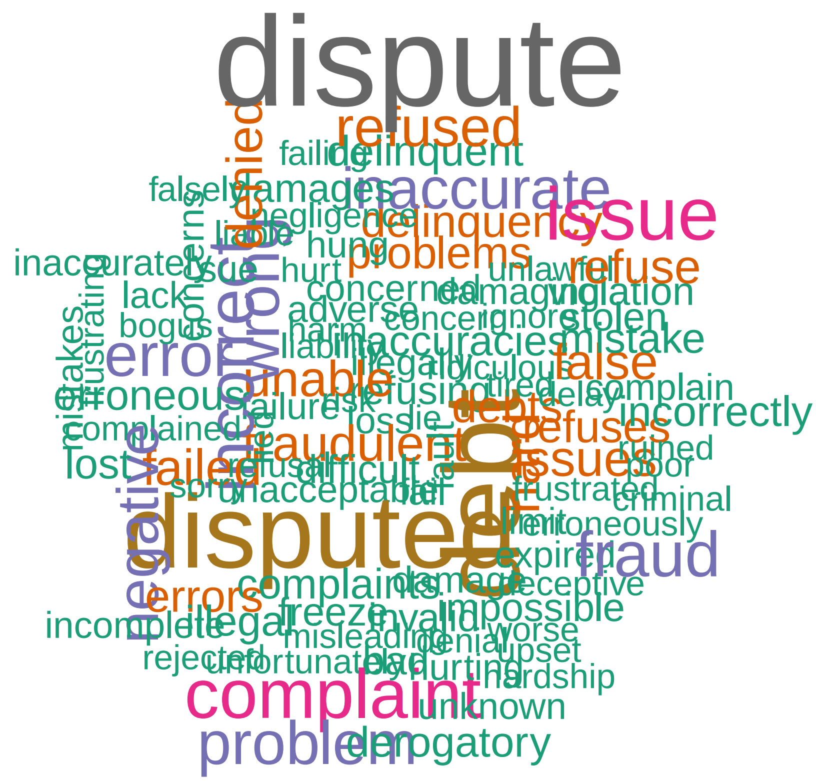
```
##   all total.sentences total.words ave.polarity sd.polarity
## 1 all            4187      240695   -0.2962299   0.4027941
##   stan.mean.polarity
## 1         -0.7354377
```

```r
positive_words = unique(setdiff(unlist(p),"-"))  # Positive words list
negative_words = unique(setdiff(unlist(n),"-"))  # Negative words list

#----------------------------------------------------------#
#   Create Postive Words wordcloud                         #
#----------------------------------------------------------#

pos.tdm = dtm[,which(colnames(dtm) %in% positive_words)]
m = as.matrix(pos.tdm)
v = sort(colSums(m), decreasing = TRUE)
windows() # opens new image window
wordcloud(names(v), v, scale=c(4,1),1, max.words=100,colors=brewer.pal(8, "Dark2"))
title(sub = "Positive Words - Wordcloud")
```

Positive Words − Wordcloud

```r
#----------------------------------------------------------#
#  Create Negative Words wordcloud                         #
#----------------------------------------------------------#

neg.tdm = dtm[,which(colnames(dtm) %in% negative_words) ]
m = as.matrix(neg.tdm)
```

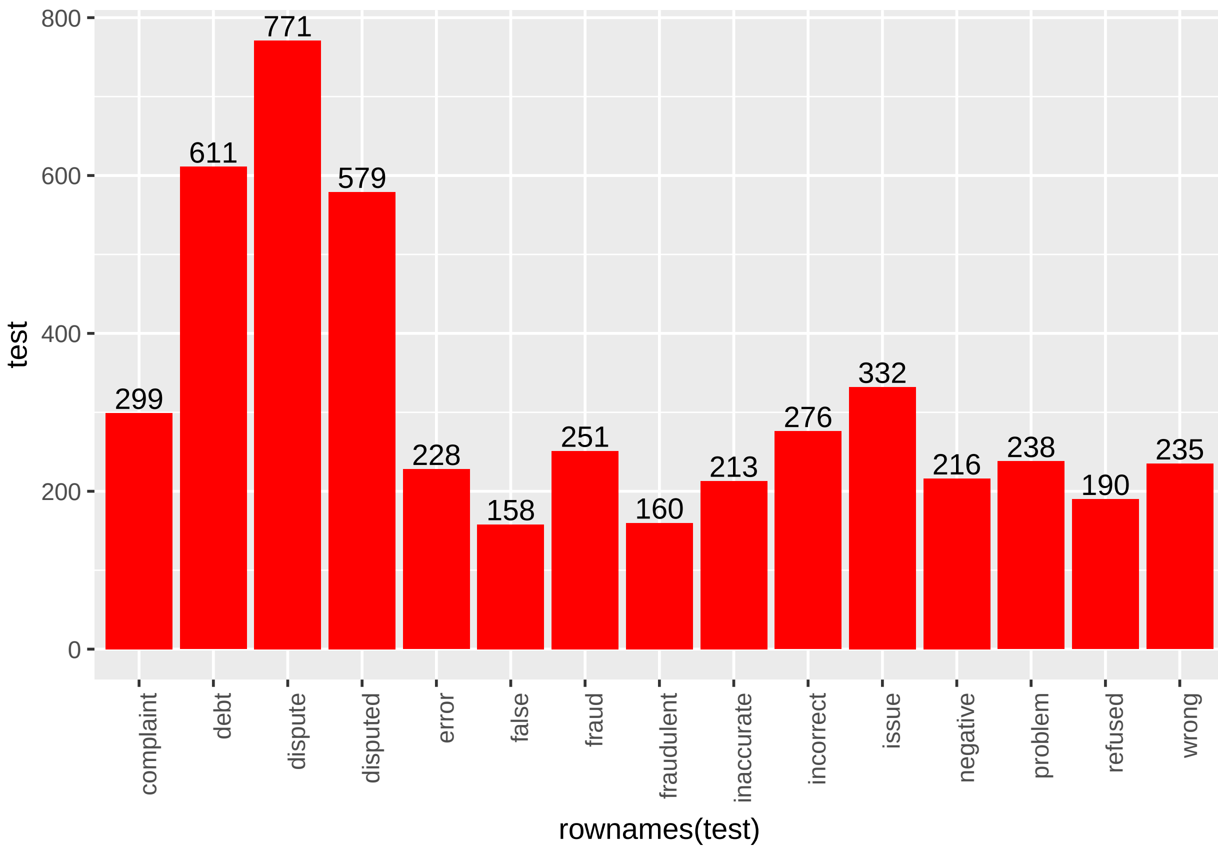
```r
v = sort(colSums(m), decreasing = TRUE)
windows()
wordcloud(names(v), v, scale=c(4,1),1, max.words=100,colors=brewer.pal(8, "Dark2"))
title(sub = "Negative Words - Wordcloud")
```



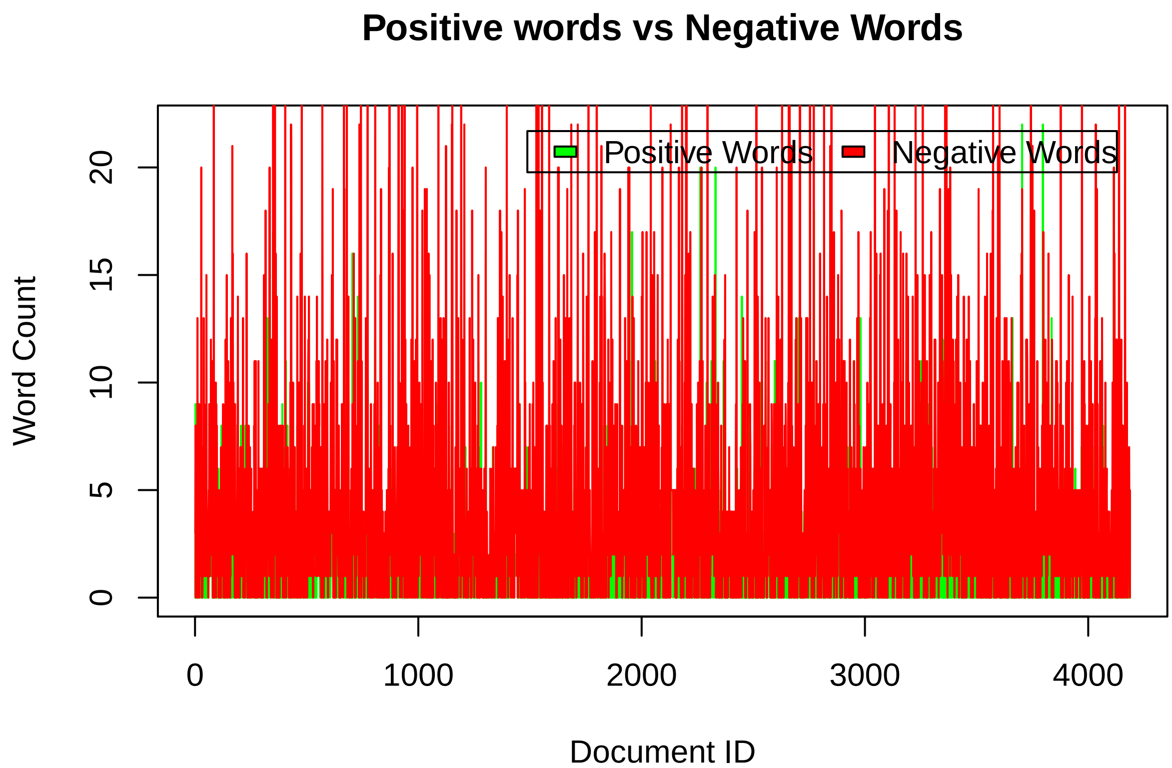
Negative Words − Wordcloud

```r
# plot barchart for top tokens
test = as.data.frame(v[1:15])
windows()
ggplot(test, aes(x = rownames(test), y = test)) +
  geom_bar(stat = "identity", fill = "red") +
  geom_text(aes(label = test), vjust= -0.20) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

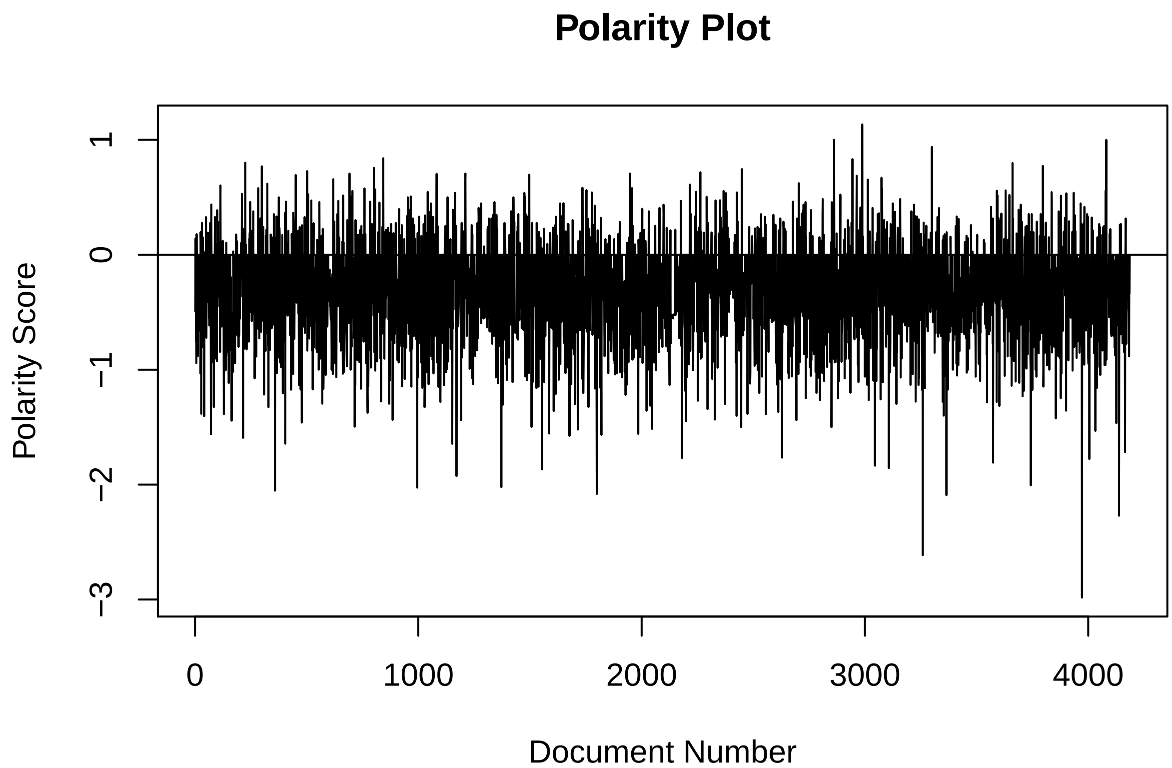
```
## Don't know how to automatically pick scale for object of type data.frame. Defaulting to continuous.
## Don't know how to automatically pick scale for object of type data.frame. Defaulting to continuous.
```




```
#--------------------------------------------------------#
#  Positive words vs Negative Words plot                 #
#--------------------------------------------------------#

len = function(x){
  if ( x == "-" && length(x) == 1)  {return (0)}
  else {return(length(unlist(x)))}
}

pcount = unlist(lapply(p, len))
ncount = unlist(lapply(n, len))
doc_id = seq(1:length(wc))

windows()
plot(doc_id,pcount,type="l",col="green",xlab = "Document ID", ylab= "Word Count")
lines(doc_id,ncount,type= "l", col="red")
title(main = "Positive words vs Negative Words" )
legend("topright", inset=.05, c("Positive Words","Negative Words"), fill=c("green","red"), horiz=TRUE)
```




```r
# Documet Sentiment Running plot
windows()
plot(pol$all$polarity, type = "l", ylab = "Polarity Score",xlab = "Document Number")
abline(h=0)
title(main = "Polarity Plot" )
```



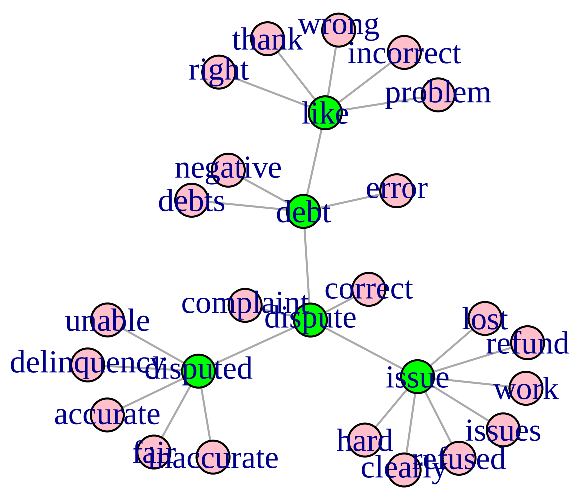

```
### COG for sentiment-laden words ? ###

senti.dtm = cbind(pos.tdm, neg.tdm); dim(senti.dtm)
```

```
## [1] 4187 1336
```

```
senti.adj.mat = as.matrix(t(senti.dtm)) %*% as.matrix(senti.dtm)
diag(senti.adj.mat) = 0

windows()
distill.cog(senti.adj.mat,    # ad mat obj
            'Distilled COG of senti words',        # plot title
            5,        # max #central nodes
            5)         # max #connexns
```

```
## Warning in vattrs[[name]][index] <- value: number of items to replace is
## not a multiple of replacement length
```

Distilled COG of senti words



Collocations

```
y= text.clean(data3$consumer_complaint_narrative)
 y  =  removeWords(y,stopwords)            # removing stopwords created above
 y  =  stripWhitespace(y )                  # removing white spac





 tok_fun_y = word_tokenizer  # using word & not space tokenizers

 it_y = itoken( y,
               #preprocessor = text.clean,
               tokenizer = tok_fun_y,
               ids = data3$id,
               progressbar = F)

 vocab_y = create_vocabulary(it_y,    #  func collects unique terms & corresponding statistics
                          ngram = c(2L, 2L) #,
                          #stopwords = stopwords
 )

 pruned_vocab_y = prune_vocabulary(vocab_y,  # filters input vocab & throws out v frequent & v infreque
                               term_count_min = 10)
```

```r
model = Collocations$new(vocabulary=pruned_vocab_y,collocation_count_min = 50)
it_yy = itoken(y)
model$fit(it_yy, n_iter = 4)
```

```
## INFO [2018-01-07 23:37:59] iteration 1 - found 62 collocations

## Warning in get_tcm(corp): Something goes wrong, tcm has 0 rows...

## INFO [2018-01-07 23:38:01] iteration 2 - found 62 collocations

## Warning in get_tcm(corp): Something goes wrong, tcm has 0 rows...

## INFO [2018-01-07 23:38:02] iteration 3 - found 62 collocations

## Warning in get_tcm(corp): Something goes wrong, tcm has 0 rows...

## INFO [2018-01-07 23:38:03] iteration 4 - found 62 collocations
```

```r
colloc <- model$collocation_stat

colloc %>%
  arrange(desc(n_ij))
```

```
##                    prefix                 suffix  n_i  n_j n_ij
## 1        credit_reporting      reporting_agencies  832  334  253
## 2             fair_credit        credit_reporting  269  832  248
## 3        credit_reporting           reporting_act  832  246  240
## 4                    ca_n                     n_t  164 1224  164
## 5         victim_identity          identity_theft  157  386  155
## 6         social_security         security_number  294  137  137
## 7        credit_reporting        reporting_agency  832  244  133
## 8                     n_t                  t_know 1224  101  101
## 9      consumer_reporting        reporting_agency  101  244   73
## 10        social_security           security_card  294   70   70
## 11                    u_s                     s_c  111   69   68
## 12                    n_t                  t_even 1224   65   65
## 13          reporting_act             act_section  246   64   64
## 14            free_annual          annual_credit   89   98   63
## 15           show_details           details_show   65   61   61
## 16           details_show           show_details   61   65   61
## 17          violated_fair             fair_credit   60  269   60
## 18          provide_proof        proof_authorized  100   56   56
## 19         credit_inquiry            inquiry_made   94   58   54
## 20            can_provide           provide_proof   82  100   54
## 21               driver_s               s_license   54   53   53
## 22         contact_credit          credit_bureaus   75  598   53
## 23         serious_breach          breach_privacy   52   52   52
## 24       fraudulent_credit         credit_inquiry   59   94   52
## 25 unauthorized_fraudulent      fraudulent_credit  110   59   52
## 26        anyone_employed           employed_make   52   51   51
## 27      authorization_form              form_gave   51   52   51
## 28        inquiry_serious          serious_breach   51   52   51
## 29              c_legally        legally_entitled   52   54   51
## 30      fraudulent_inquiry        inquiry_serious   99   51   51
## 31    remove_unauthorized unauthorized_fraudulent   57  110   51
## 32         fraudulent_hard           hard_inquiry   52  132   51
## 33           hard_inquiry    inquiry_immediately  132   52   51
```

```
## 34                    wo_n                   n_t   51 1224   51
## 35        validity_advised           advised_can   50   50   50
## 36              gave_right            right_view   50   50   50
## 37        report_demanding     demanding_contact   51   50   50
## 38           employed_make          make_inquiry   51   50   50
## 39           entitled_make       make_fraudulent   50   51   50
## 40               form_gave            gave_right   52   50   50
## 41             within_five         five_business   53   50   50
## 42          breach_privacy        privacy_rights   52   51   50
## 43        legally_entitled         entitled_make   54   50   50
## 44             copy_signed  signed_authorization   51   53   50
## 45    signed_authorization    authorization_form   53   51   50
## 46         verify_validity      validity_advised   55   50   50
## 47           credit_within           within_five   52   53   50
## 48        proof_authorized       authorized_view   56   50   50
## 49         report_violated         violated_fair   51   60   50
## 50               mail_copy           copy_signed   63   51   50
## 51       demanding_contact        contact_credit   50   75   50
## 52             advised_can          can_provide   50   82   50
## 53           five_business         business_days   50   88   50
## 54           business_days              days_can   88   52   50
## 55         make_fraudulent    fraudulent_inquiry   51   99   50
## 56                days_can            can_verify   52  100   50
## 57              can_verify       verify_validity  100   55   50
## 58 unauthorized_fraudulent       fraudulent_hard  110   52   50
## 59         authorized_view           view_credit   50  165   50
## 60              right_view           view_credit   50  165   50
## 61             view_credit         credit_within  165   52   50
## 62                     n_t                 t_get 1224   50   50
##          pmi       lfmd     gensim rank_pmi rank_lfmd rank_gensim
## 1   5.881331 -10.117644  47.296879       56         5          16
## 2   6.164775  -9.891794  57.279052       55         1          13
## 3   6.246418  -9.904764  60.103727       54         2          11
## 4   5.725093 -11.524765  36.769339       60        43          18
## 5   7.371528 -10.041186 112.178228       52         4           5
## 6   7.782809  -9.986089 139.848428       49         3           4
## 7   5.406587 -12.447811  26.471050       62        57          21
## 8   5.725093 -12.923446  26.709983       61        58          20
## 9   7.583357 -12.001956  60.425864       51        52          10
## 10  7.782809 -11.923587  62.920311       50        50           9
## 11  9.167004 -10.623033 152.162162       36        28           3
## 12  5.725093 -14.195133  12.206825       59        59          29
## 13  8.039967 -11.924996  57.573044       47        51          12
## 14  8.869318 -11.141085  96.501376       42        35           6
## 15  9.960113 -10.143374 179.620429       23         6           1
## 16  9.960113 -10.143374 179.620429       24         7           2
## 17  7.911019 -12.240162  40.114622       48        56          17
## 18  9.338625 -11.011628  69.369643       30        32           7
## 19  9.324799 -11.130389  47.501834       31        34          15
## 20  8.735960 -11.719227  31.582927       43        49          19
## 21 10.227594 -10.281528  67.866876       13         8           8
## 22  6.257581 -14.251540   4.330769       53        60          33
## 23 10.282041 -10.282041  47.888314       10         9          14
## 24  9.245689 -11.318394  23.348359       34        39          25
```

```
## 25  9.018918 -11.545165  19.952234        39         45           27
## 26 10.282041 -10.338070  24.413650         7         10           22
## 27 10.282041 -10.338070  24.413650         8         11           23
## 28 10.282041 -10.338070  24.413650         9         12           24
## 29 10.199579 -10.420532  23.057336        18         19           26
## 30  9.353125 -11.266987  12.823331        29         38           28
## 31  9.040657 -11.579455  10.326156        38         46           30
## 32  8.910073 -11.710039   9.432547        40         47           31
## 33  8.910073 -11.710039   9.432547        41         48           32
## 34  5.725093 -14.895018   1.037181        58         61           34
## 35 10.338625 -10.338625   0.000000         1         13           35
## 36 10.338625 -10.338625   0.000000         2         14           36
## 37 10.310056 -10.367194   0.000000         3         15           37
## 38 10.310056 -10.367194   0.000000         4         16           38
## 39 10.310056 -10.367194   0.000000         5         17           39
## 40 10.282041 -10.395209   0.000000         6         18           40
## 41 10.254561 -10.422689   0.000000        11         20           41
## 42 10.253472 -10.423778   0.000000        12         21           42
## 43 10.227594 -10.449656   0.000000        14         22           43
## 44 10.225992 -10.451258   0.000000        15         23           44
## 45 10.225992 -10.451258   0.000000        16         24           45
## 46 10.201121 -10.476128   0.000000        17         25           46
## 47 10.197977 -10.479273   0.000000        19         26           47
## 48 10.175126 -10.502124   0.000000        20         27           48
## 49 10.047021 -10.630229   0.000000        21         29           49
## 50  9.976632 -10.700618   0.000000        22         30           50
## 51  9.753662 -10.923587   0.000000        25         31           51
## 52  9.624929 -11.052321   0.000000        26         33           52
## 53  9.523050 -11.154200   0.000000        27         36           53
## 54  9.466466 -11.210784   0.000000        28         37           54
## 55  9.324555 -11.352695   0.000000        32         40           55
## 56  9.282041 -11.395209   0.000000        33         41           56
## 57  9.201121 -11.476128   0.000000        35         42           57
## 58  9.144538 -11.532712   0.000000        37         44           58
## 59  8.616159 -12.061091   0.000000        44         53           59
## 60  8.616159 -12.061091   0.000000        45         54           60
## 61  8.559575 -12.117675   0.000000        46         55           61
## 62  5.725093 -14.952157   0.000000        57         62           62
```