HappyDB_analysis

Siyu Zhu, sz2716

9/17/2018

Introduction

In this notebook, I'm trying to do some preliminary analysis on the HappyDB dataset, in the meanwhile introducing some basic and useful analytical tools/libraries.
This notebook consists of four parts:
1. Data preparation and basic analyses. The main tasks are loading the data, and drawing a word cloud.
2. The distribution of happiness on "achievement", "affection" and "self-origin", based on overall population and subcategory population.
3. Marital and gender's influence on people's perchasing tendency.
4. Classification of marital. It's a binary classification problem, using text2vec package.
Let's get started!
##1. Data preparation and basic analysis
1.1 Data preparation
Load data

```
library(DT)
library(tidyverse)
library(tm)
library(glmnet)
hm_data <- read.csv(file="/Users/siyuzhu/Documents/Github/ADS/Fall2018-Proj1-Zhusy5/data/cleaned_hm.csv
```

Load packages

```
import pandas as pd
import numpy as np
import sys
import tensorflow as tf
import matplotlib.pyplot as plt
# matplotlib.use('TkAgg')
print('Python version ' + sys.version)
```

```
## Python version 3.6.5 |Anaconda, Inc.| (default, Apr 26 2018, 08:44:39)
## [GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
```

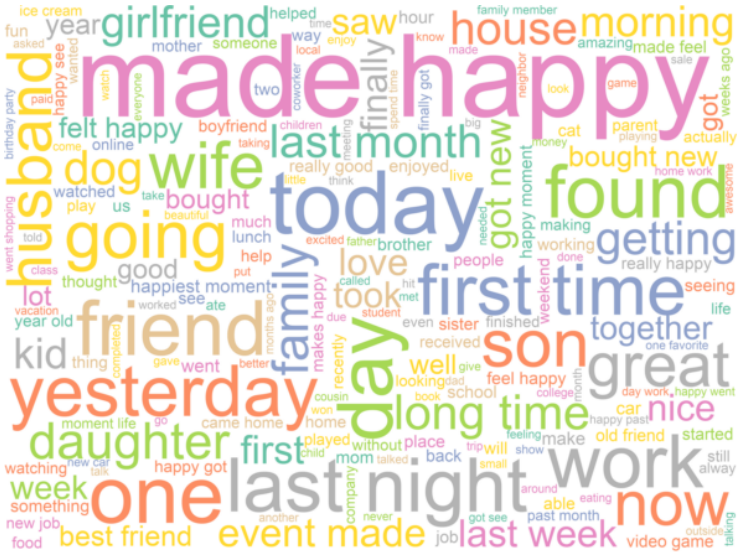
```
print('Pandas version ' + pd.__version__)
```

```
## Pandas version 0.23.0
```

1.2 Word frequencey
We want to find out what words people mention most in their happy moments. So we first to a word cloud.

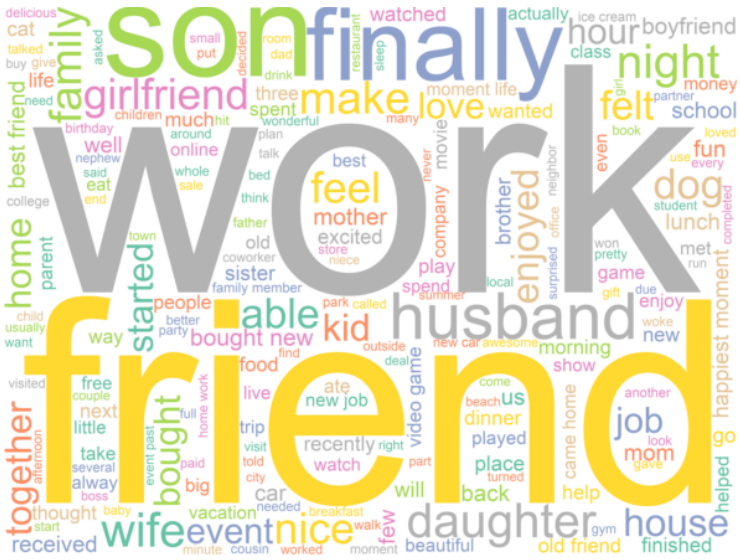
```
from wordcloud import WordCloud, ImageColorGenerator
hm_data = pd.read_csv("../data/cleaned_hm.csv")
df_hm = hm_data[hm_data['cleaned_hm'].notnull()]
text = ' '.join(df_hm['cleaned_hm'].tolist())
text = text.lower()
wordcloud = WordCloud(background_color="white", font_path='../../../Library/Fonts/Arial.ttf', \
                          height=2700, width=3600).generate(text)
```

```
plt.figure( figsize=(14,8) )
plt.imshow(wordcloud.recolor(colormap=plt.get_cmap('Set2')), interpolation='bilinear')
plt.axis("off")
plt.show()
```



Easy to see that some words appear more frequently, such as “work”, “family” and “husband”, which make sense. However, there are also some noisy words that are not very informative, such as “yesterday” and “today”.
Thus, let’s clean the word cloud by removing these noises.

```
LIMIT_WORDS = ['happy', 'day', 'got', 'went', 'today', 'made', 'one', 'two', 'time', 'last', 'first', 'g
text = ' '.join(df_hm['cleaned_hm'].tolist())
text = text.lower()
for w in LIMIT_WORDS:
    text = text.replace(' ' + w, '')
    text = text.replace(w + ' ', '')
wordcloud = WordCloud(background_color="white", font_path='../../../Library/Fonts/Arial.ttf', \
                      height=2700, width=3600).generate(text)
plt.figure( figsize=(14,8) )
plt.imshow(wordcloud.recolor(colormap=plt.get_cmap('Set2')), interpolation='bilinear')
plt.axis("off")
plt.show()
```



2. The distribution of happiness

2.1 Overall happy distribution

We noticed that there is a "predicted_category" column in hm_data, therefore, we want to find out the happiness distribution based on these categories.

```
sum(is.null(hm_data$predicted_category))
```

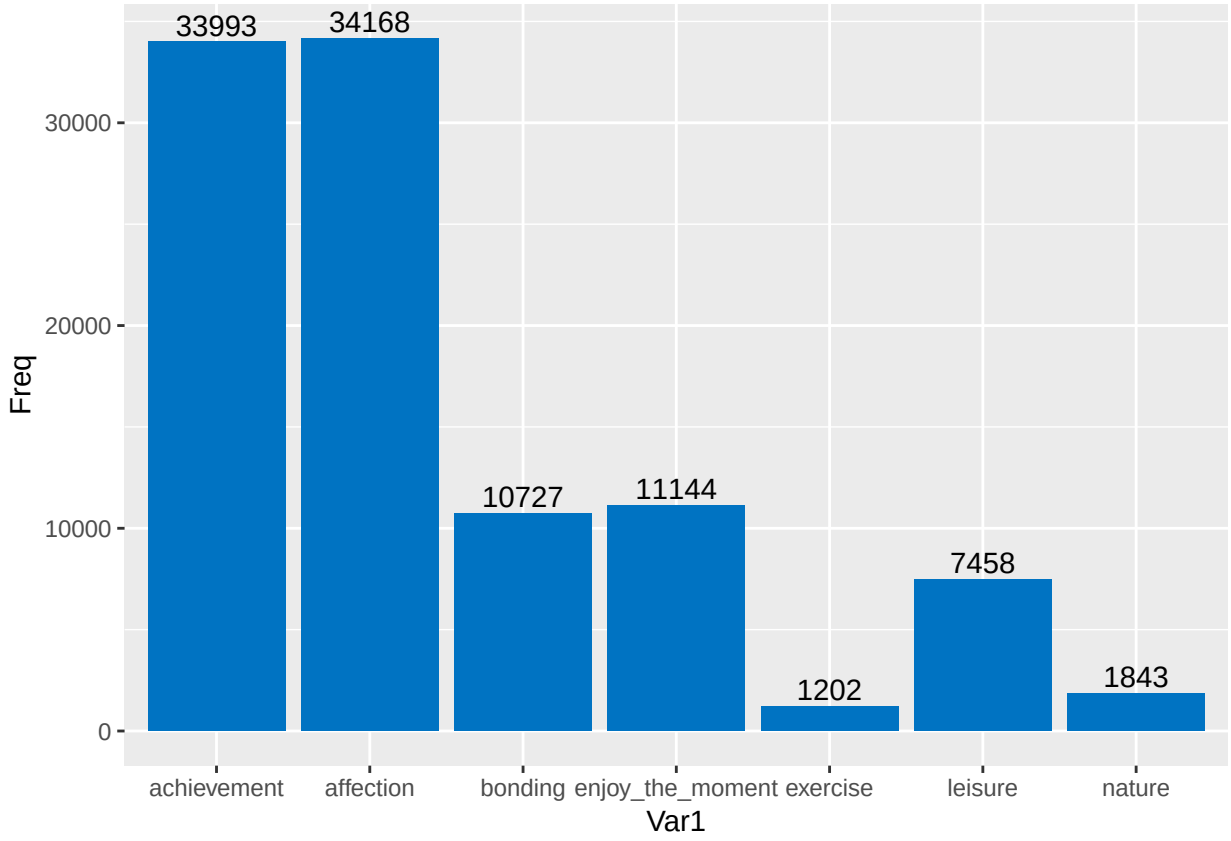
```
## [1] 0
```

```
category_sum1 <- as.data.frame(table(hm_data$predicted_category))
category_sum1
```

```
##                Var1  Freq
## 1       achievement 33993
## 2         affection 34168
## 3           bonding 10727
## 4 enjoy_the_moment 11144
## 5          exercise  1202
## 6           leisure  7458
## 7            nature  1843
```

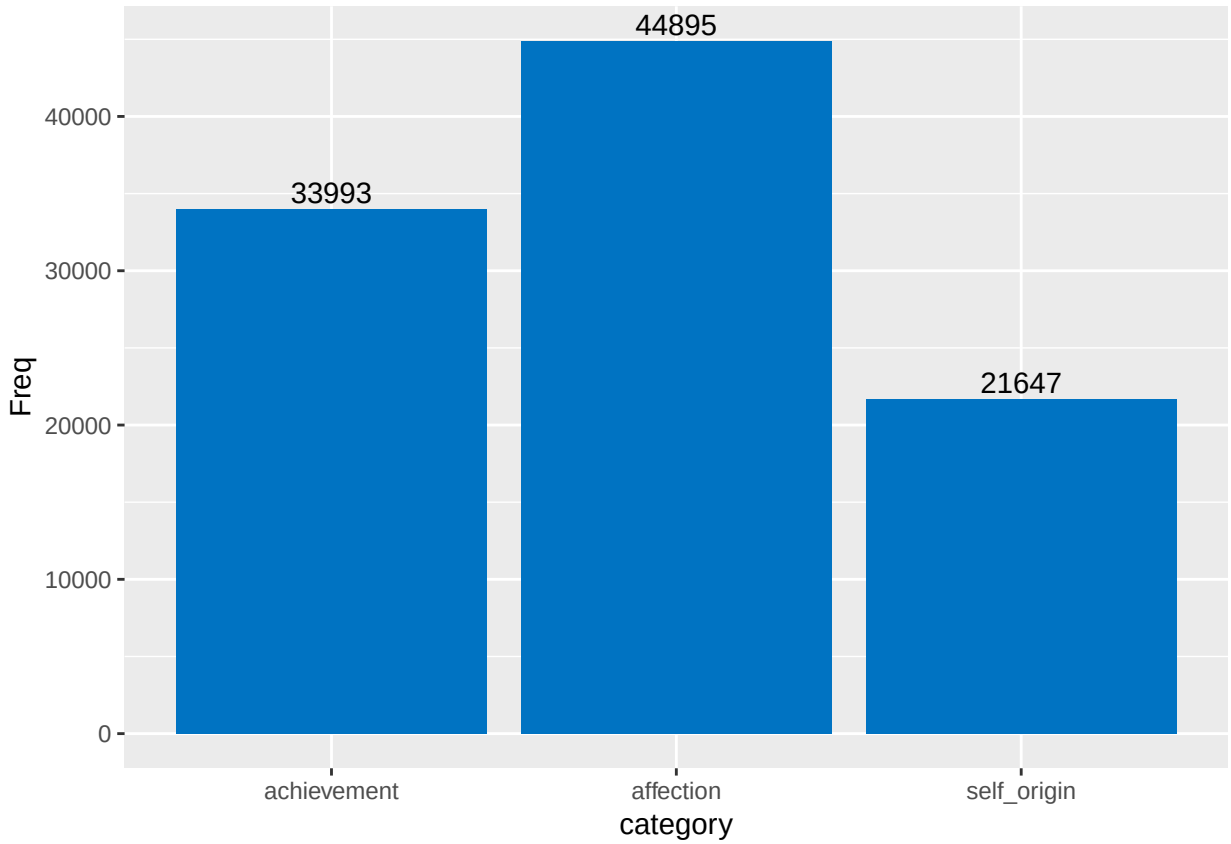
```
category_sum1 <- mutate(category_sum1, Prob = Freq / sum(category_sum1$Freq))
prob <- category_sum1$Prob
names(prob) <- c("achievement", "affection", "bonding", "enjoy_the_moment", "exercise", "leisure", "natu
prob <- as.data.frame(prob)
names(prob) <- "whole_prob"

library(ggplot2)
ggplot(category_sum1, aes(x = Var1, y = Freq)) +
```

```
  geom_bar(fill = "#0073C2FF", stat = "identity") +
  geom_text(aes(label = Freq), vjust = -0.3)
```



We can tell from the above figure that among all the happiness-reasons, people have been more frequently made happy due to the achievement they made and affection they have (the latter one is slightly more influential than the former one), which really makes sense.
Moreover, these two factors are much more influential than all the other factors.
Now, if we look more closely to the factors in the data, we will find that all these factors can be merged and divided into only three bigger factors: "achievment", "affection", and "self-made"(which means self/individual activites, such as exercise, enjoy the moment, relax). We shall do another happiness distribution based on this divide.

```
category_sum2 <- data.frame("category" = c("achievement", "affection", "self_origin"), "Freq" = c(categ

library(ggplot2)
ggplot(category_sum2, aes(x = category, y = Freq)) +
  geom_bar(fill = "#0073C2FF", stat = "identity") +
  geom_text(aes(label = Freq), vjust = -0.3)
```



We can see that affection is still the number one influential factor for happiness, which matches the result of the 75-year Havard study: Good relationships keep us happier and healthier!

2.2 Subcategory-based happy distribution

Now combined with the given demographic data, we can find out the happiness distributions for subcategories(male, female, single, married, nonparent, parent), their .
load demographic data and do inner join

```
library(dplyr)
demo_data <- read.csv(file="/Users/siyuzhu/Documents/Github/ADS/Fall2018-Proj1-Zhusy5/data/demographic.

hmdemo_data <- hm_data %>%
  inner_join(demo_data, by = "wid") %>%
  select(wid,
         hmid,
         cleaned_hm,
         predicted_category,
         gender,
         marital,
         parenthood,
         age,
         country,
         reflection_period)
# datatable(head(hmdemo_data))
```

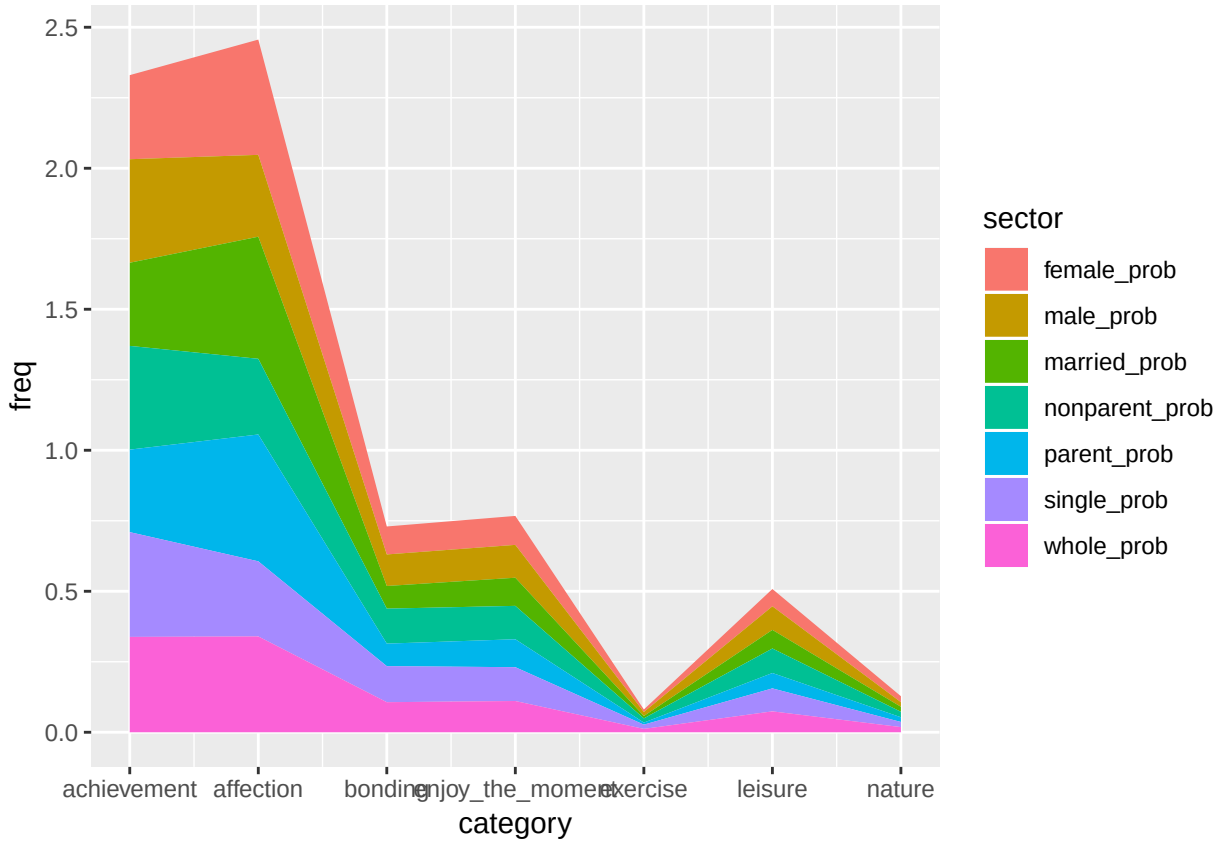
```
# for male
male_cate <- hmdemo_data[hmdemo_data$gender == "m", 4]
male_sum <- as.data.frame(table(male_cate))
male_sum <- mutate(male_sum, Prob = Freq / sum(male_sum$Freq))
```

```r
prob$male_prob <- male_sum$Prob

# for female
female_cate <- hmdemo_data[hmdemo_data$gender == "f", 4]
female_sum <- as.data.frame(table(female_cate))
female_sum <- mutate(female_sum, Prob = Freq / sum(female_sum$Freq))
prob$female_prob <- female_sum$Prob

# for single
single_cate <- hmdemo_data[hmdemo_data$marital == "single", 4]
single_sum <- as.data.frame(table(single_cate))
single_sum <- mutate(single_sum, Prob = Freq / sum(single_sum$Freq))
prob$single_prob <- single_sum$Prob

# for marital
married_cate <- hmdemo_data[hmdemo_data$marital == "married", 4]
married_sum <- as.data.frame(table(married_cate))
married_sum <- mutate(married_sum, Prob = Freq / sum(married_sum$Freq))
prob$married_prob <- married_sum$Prob

# for nonparent
nonparent_cate <- hmdemo_data[hmdemo_data$parenthood == "n", 4]
nonparent_sum <- as.data.frame(table(nonparent_cate))
nonparent_sum <- mutate(nonparent_sum, Prob = Freq / sum(nonparent_sum$Freq))
prob$nonparent_prob <- nonparent_sum$Prob

# for parent
parent_cate <- hmdemo_data[hmdemo_data$parenthood == "y", 4]
parent_sum <- as.data.frame(table(parent_cate))
parent_sum <- mutate(parent_sum, Prob = Freq / sum(parent_sum$Freq))
prob$parent_prob <- parent_sum$Prob

prob <- t(prob)
prob
```

```
##                achievement affection    bonding enjoy_the_moment
## whole_prob       0.3381211 0.3398617 0.10669916       0.11084697
## male_prob        0.3670827 0.2903103 0.11189114       0.11646733
## female_prob      0.2982006 0.4086857 0.09921795       0.10275975
## single_prob      0.3716039 0.2662366 0.12769379       0.11989428
## married_prob     0.2948262 0.4329149 0.08020705       0.09977505
## nonparent_prob   0.3678553 0.2683755 0.12406256       0.11840097
## parent_prob      0.2925101 0.4498482 0.07998482       0.09906377
##                   exercise    leisure     nature
## whole_prob     0.011956035 0.07418312 0.01833192
## male_prob      0.014283238 0.08363668 0.01632865
## female_prob    0.008723763 0.06128028 0.02113195
## single_prob    0.015081507 0.08134033 0.01814956
## married_prob   0.008393198 0.06605713 0.01782648
## nonparent_prob 0.014999097 0.08690943 0.01939708
## parent_prob    0.007287449 0.05457996 0.01672571
```

```r
graph_data <- data.frame("sector" = rep(c("whole_prob", "male_prob", "female_prob", "single_prob", "mar
```

```
graph_data2 <- data.frame(category = as.numeric(as.factor(graph_data$category)),
                          freq = graph_data$freq,
                          sector = graph_data$sector)

ggplot(graph_data2, aes(x=category, y=freq, fill=sector)) +
    geom_area() +
  scale_x_continuous(breaks=seq(1, 7, 1),
        labels=names(table(graph_data$category)))
```



Easy to tell that the happiness distribution patten for overall population still works for subcategory's population.
Moreover, if we look closer, we shall notice that: 1. compared with male, female attaches more importance to affection, whereas male pays more attention to achievement; 2. compared with nonparent people, parent people attache more importance to affection, whereas nonparent people pay more attention to achievement; 3. compared with single people, married people attache more importance to affection, whereas single ones pay more attention to achievement. All of these make sense in real world.

3. Perchase tendency

From part2, gende and marriage make people attach different level of imprtance to affection and achievment. Furthermore, I m also interested in what these two factors' influence on people's daily purchase tendency.

```
count_purchase_hm <- function(data_subset){
  pattern = "(buy) | (purchase) | (bought) | (perchased) | (shopping)"
  corpus_purchase_selection <- sapply(data_subset$cleaned_hm, grep, pattern = pattern)
  v = c()
```

```
  for (i in 1:length(corpus_purchase_selection)){
    test = (as.numeric(corpus_purchase_selection[i] == 1) != 1)
    if(is.na(test)){

    }
    else{
      v = c(v, i)
    }
  }
  return(length(v))
}

m_shop <- count_purchase_hm(subset(hmdemo_data, marital == "married"))
s_shop <- count_purchase_hm(subset(hmdemo_data, marital == "single"))
m_f_shop <- count_purchase_hm(subset(hmdemo_data, marital == "married" & gender == "f"))
m_m_shop <- count_purchase_hm(subset(hmdemo_data, marital == "married" & gender == "m"))
s_f_shop <- count_purchase_hm(subset(hmdemo_data, marital == "single" & gender == "f"))
s_m_shop <- count_purchase_hm(subset(hmdemo_data, marital == "single" & gender == "m"))

data.frame(class = c("married", "single", "married_female", "married_male", "single_female", "single_ma
```

```
##            class number
## 1        married   1748
## 2         single   2246
## 3 married_female    884
## 4   married_male    845
## 5  single_female    782
## 6    single_male   1454
```

From the above dataframe, we notice that single poeple have more purchase tendency than married people (maybe because married people have higher financial pressure and thus buy less to save more).
And for married female and married male, there is no big diffenrence in purchasing, which also makes sense, since couple may share family expenditure together.
Single male, however, buy much more stuff than single female (double actually!), which is a quite interesting finding.

4. A logistic regression classifier for marital

Here, we want to classify people into "single" or "married" categories based on their happy moment text. First of all let's split out dataset into two parts - train and test. We will show how to perform data manipulations on train set and then apply exactly the same manipulations on the test set

```
library(text2vec)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

## The following object is masked from 'package:purrr':
##
```

```
##     transpose
```

```
library(magrittr)
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
hmdemo_data <- hmdemo_data[hmdemo_data$marital %in% c("single", "married"), ]
hm_marry <- hmdemo_data[, c("hmid", "cleaned_hm", "marital")]
hm_marry$marital <- as.numeric(as.factor(hm_marry$marital)) - 1
setDT(hm_marry)
setkey(hm_marry, hmid)
set.seed(2017L)
all_ids = hm_marry$hmid
train_ids = sample(all_ids, 76359)
test_ids = setdiff(all_ids, train_ids)
train = hm_marry[J(train_ids)]
test = hm_marry[J(test_ids)]
```

Let's then create a vocabulary-based DTM. Here we collect unique terms from all documents and mark each of them with a unique ID using the create_vocabulary() function. We use an iterator to create the vocabulary.

```
prep_fun = tolower
tok_fun = word_tokenizer
#hmdemo_data <- read.csv("hm_data_withdemo.csv", stringsAsFactors = F)
it_train = itoken(train$cleaned_hm,
             preprocessor = prep_fun,
             tokenizer = tok_fun,
             ids = train$hmid,
             progressbar = TRUE)
vocab = create_vocabulary(it_train)
```

```
##
  |
  |=======                                                |  10%
  |
  |=============                                          |  20%
  |
  |====================                                   |  30%
  |
  |==========================                             |  40%
  |
  |================================                       |  50%
  |
  |=======================================                |  60%
  |
  |==============================================         |  70%
  |
```

```
  |=================================================             |  80%
  |
  |=============================================================       |  90%
  |
  |=================================================================| 100%
```

```
vocab
```

```
## Number of docs: 76359
## 0 stopwords:  ...
## ngram_min = 1; ngram_max = 1
## Vocabulary:
##              term term_count doc_count
##     1:      liken          1         1
##     2:   citizens          1         1
##     3:    today.a          1         1
##     4: injections          1         1
##     5:  crashersa          1         1
##    ---
## 24551:        and      42784     28610
## 24552:         to      43415     30190
## 24553:          a      54219     39279
## 24554:         my      56341     43345
## 24555:          i      78973     52863
```

Now that we have a vocabulary, we can construct a document-term matrix.

```
vectorizer = vocab_vectorizer(vocab)
t1 = Sys.time()
dtm_train = create_dtm(it_train, vectorizer)
```

```
##
  |
  |=======                                                         |  10%
  |
  |=============                                                   |  20%
  |
  |====================                                            |  30%
  |
  |==========================                                      |  40%
  |
  |================================                                |  50%
  |
  |======================================                          |  60%
  |
  |============================================                    |  70%
  |
  |==================================================              |  80%
  |
  |============================================================    |  90%
  |
  |================================================================| 100%
```

```
print(difftime(Sys.time(), t1, units = 'sec'))
```

```
## Time difference of 1.764904 secs
```

Now we have a DTM and can check its dimensions.

```
dim(dtm_train)
```

```
## [1] 76359 24555
```

As you can see, the DTM has 76359 rows, equal to the number of documents, and 24555 columns, equal to the number of unique terms.
Now we are ready to fit our first model. Here we will use the glmnet package to fit a logistic regression model with an L1 penalty and 4 fold cross-validation.

```
library(glmnet)
NFOLDS = 4
t1 = Sys.time()
glmnet_classifier = cv.glmnet(x = dtm_train, y = train$marital,
                              family = 'binomial',
                              # L1 penalty
                              alpha = 1,
                              # interested in the area under ROC curve
                              type.measure = "auc",
                              # 5-fold cross-validation
                              nfolds = NFOLDS,
                              # high value is less accurate, but has faster training
                              thresh = 1e-3,
                              # again lower number of iterations for faster training
                              maxit = 1e3)
```

```
## Warning: from glmnet Fortran code (error code -37); Convergence for 37th
## lambda value not reached after maxit=1000 iterations; solutions for larger
## lambdas returned
```

```
## Warning: from glmnet Fortran code (error code -41); Convergence for 41th
## lambda value not reached after maxit=1000 iterations; solutions for larger
## lambdas returned
```

```
## Warning: from glmnet Fortran code (error code -35); Convergence for 35th
## lambda value not reached after maxit=1000 iterations; solutions for larger
## lambdas returned
```

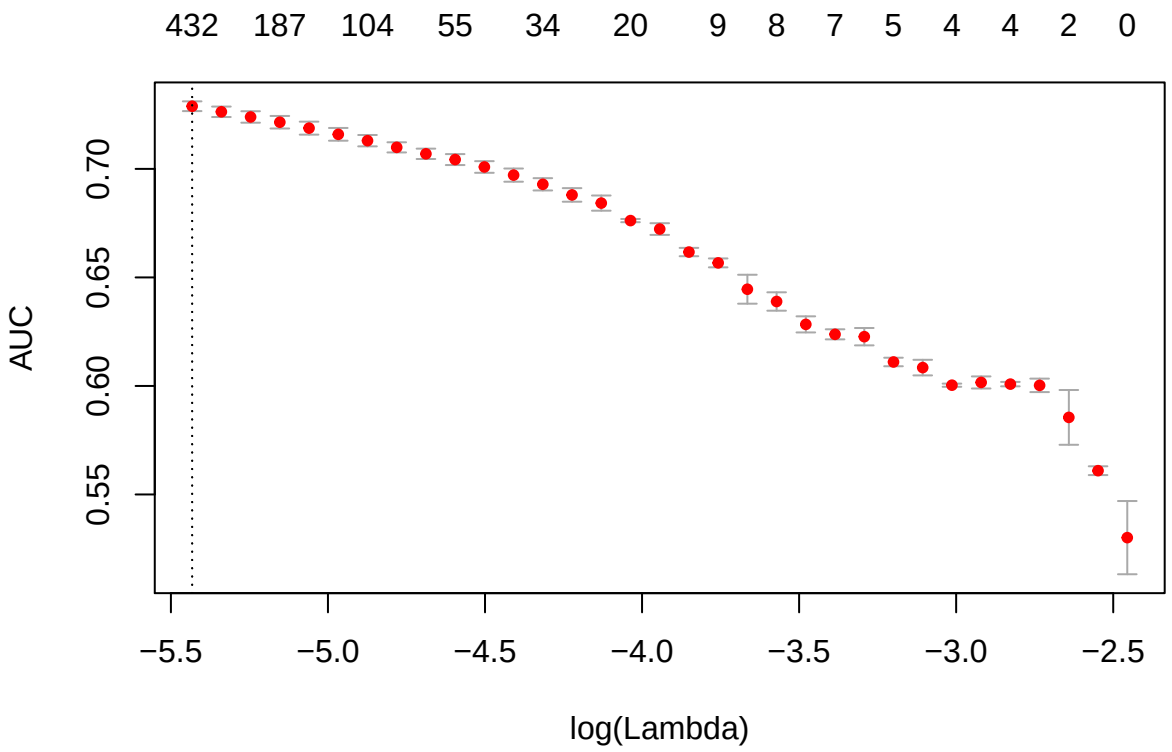
```
## Warning: from glmnet Fortran code (error code -36); Convergence for 36th
## lambda value not reached after maxit=1000 iterations; solutions for larger
## lambdas returned
```

```
## Warning: from glmnet Fortran code (error code -35); Convergence for 35th
## lambda value not reached after maxit=1000 iterations; solutions for larger
## lambdas returned
```

```
print(difftime(Sys.time(), t1, units = 'sec'))
```

```
## Time difference of 20.31485 secs
```

```
plot(glmnet_classifier)
```




```
print(paste("max AUC =", round(max(glmnet_classifier$cvm), 4)))
```

```
## [1] "max AUC = 0.7289"
```

We have successfully fit a model to our DTM. Now we can check the model's performance on test data. Note that we use exactly the same functions from prepossessing and tokenization. Also we reuse/use the same vectorizer - function which maps terms to indices.

```
# Note that most text2vec functions are pipe friendly!
it_test = test$cleaned_hm %>%
  prep_fun %>% tok_fun %>%
  # turn off progressbar because it won't look nice in rmd
  itoken(ids = test$hmid, progressbar = FALSE)


dtm_test = create_dtm(it_test, vectorizer)

preds = predict(glmnet_classifier, dtm_test, type = 'response')[,1]
glmnet:::auc(test$marital, preds)
```

```
## [1] 0.7255306
```

As we can see, prediction performance (72.55% precision) on the test data is roughly the same as we expect from cross-validation (74.88% precision), which is not bad.