Project1

Nannan Wang

## 1.Data Preparation

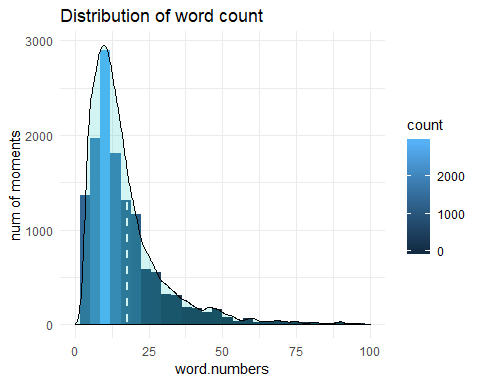
urlfile1 <- 'https://raw.githubusercontent.com/rit-public/HappyDB/master/happydb/data/cleaned\_hm.csv'  
hm\_data <- read\_csv(urlfile1)  
urlfile2 <- 'https://raw.githubusercontent.com/rit-public/HappyDB/master/happydb/data/demographic.csv'  
demographic <- read\_csv(urlfile2)  
fulldata <- full\_join(hm\_data, demographic, c("wid" = "wid"))  
new <- fulldata[,-1]  
newdata <- new %>%  
 filter(gender %in% c("m", "f")) %>%  
 filter(marital %in% c("single", "married")) %>%  
 filter(parenthood %in% c("n", "y")) %>%  
 filter(reflection\_period %in% c("24h", "3m")) %>%  
 mutate(reflection\_period = fct\_recode(reflection\_period,   
 months\_3 = "3m", hours\_24 = "24h"))  
newdata <- na.omit(newdata)  
# head(newdata)

## 2.Data Presentation

In this part, we use different forms to display the happy moments' text and explore some interesting details.

### 2.1 Word Count

word.count <- vector(length = length(newdata$cleaned\_hm))  
for (i in 1:length(newdata$cleaned\_hm)) {  
 word.count[i] <- wordcount(newdata$cleaned\_hm[i], sep = " ", count.function = sum)  
}  
newnewdata <- cbind(newdata,word.count)  
ggplot(newnewdata, aes(x=word.count)) +  
 xlim(0,100) +  
 geom\_histogram(bins=30, aes(fill = ..count..)) +   
 geom\_vline(aes(xintercept=mean(word.count)),  
 color="#FFFFFF", linetype="dashed", size=1) +  
 geom\_density(aes(y=4 \* ..count..),alpha=.2, fill="#1CCCC6") +  
 ylab("num of moments") + xlab ("word.numbers") +  
 ggtitle("Distribution of word count") +  
 theme\_minimal()

 In this Part, I count the word number for each happy moment discribtion, and most people can express their happiness with less than 15 words. It perhaps shows that happiness do not need too much words to speak out.

### 2.2 Word Frequency

happy\_text <- newdata$cleaned\_hm  
docs <- Corpus(VectorSource(happy\_text))  
# Converting the text to lower case  
docs <- tm\_map(docs, content\_transformer(tolower))  
# Removing english common stopwords  
docs <- tm\_map(docs, removeWords, stopwords("english"))  
docs <- tm\_map(docs, removeWords, c("happy", "got", "went", "made", "day", "time", "just", "last", "great", "get"))  
# creating term document matrix  
tdm <- TermDocumentMatrix(docs)  
# defining tdm as matrix  
m <- as.matrix(tdm)  
# getting word counts in decreasing order  
word\_freqs = sort(rowSums(m), decreasing=TRUE)  
# creating a data frame with words and their frequencies  
text\_wc\_df <- data.frame(word=names(word\_freqs), freq=word\_freqs)  
text\_wc\_df <- text\_wc\_df[1:500,]  
# plotting wordcloud  
set.seed(1234)  
wordcloud(words = text\_wc\_df$word, freq = text\_wc\_df$freq,  
min.freq = 0,scale=c(5,.5),  
max.words=300, random.order=FALSE, rot.per=0.15,  
colors=brewer.pal(10, "Dark2"))



In this part, We find that some words appear most frenquently in people's happy moments, such like: "work", "friend", "new", "family", "son","game", "birthday" etc.

### 2.3 Bigrams

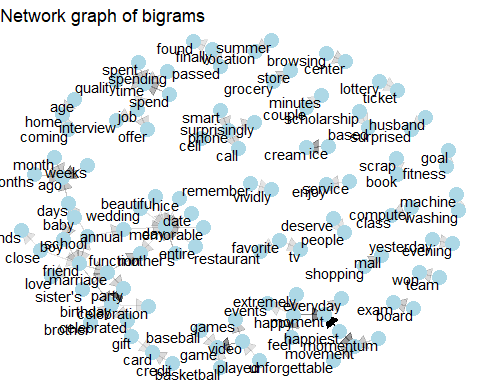
count\_bigrams <- function(dataset) {  
dataset %>%  
unnest\_tokens(bigram, cleaned\_hm , token = "ngrams", n = 2) %>%  
separate(bigram, c("word1", "word2"), sep = " ") %>%  
filter(!word1 %in% stop\_words$word,  
!word2 %in% stop\_words$word) %>%  
count(word1, word2, sort = TRUE)  
}  
   
text\_bigrams <- newdata %>%  
count\_bigrams()  
   
head(text\_bigrams, 10)

## # A tibble: 10 x 3  
## word1 word2 n  
## <chr> <chr> <int>  
## 1 happiest moment 191  
## 2 birthday party 105  
## 3 happy moment 86  
## 4 video game 79  
## 5 happiest movement 73  
## 6 weeks ago 73  
## 7 3 months 66  
## 8 ice cream 63  
## 9 24 hours 55  
## 10 feel happy 53

In this part, we focus on the bigrams which are phrases we used in the daily life. In terms of top 10 bigrams, we find top three meaningful phrases which play very important roles in people's happy moments:1)birthday party 2)video game 3)ice cream, that is amazing!

### 2.4 Bigrams Visualization

visualize\_bigrams <- function(bigrams) {  
set.seed(2018)  
a <- grid::arrow(type = "closed", length = unit(.15, "inches"))  
   
bigrams %>%  
graph\_from\_data\_frame() %>%  
ggraph(layout = "fr") +  
geom\_edge\_link(aes(edge\_alpha = n), show.legend = FALSE, arrow = a) +  
geom\_node\_point(color = "lightblue", size = 5) +  
geom\_node\_text(aes(label = name), vjust = 1, hjust = 1) +  
ggtitle("Network graph of bigrams") +  
theme\_void()  
}  
   
text\_bigrams %>%  
 filter(n > 14,  
 !str\_detect(word1, "\\d"),  
 !str\_detect(word2, "\\d")) %>%  
visualize\_bigrams()

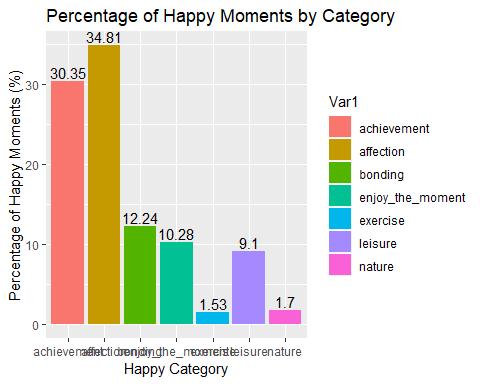


In this part, based on the bigrams we focused, we visualize the bigrams using the network graph, and we can find the most popular happy moments and their connections! For instance, the wedding/date/marriage can be connected together.

## 3.Exploration of Happy Moments

In this part, we wan to explore different Categories of Happy Moments, and we also explore the moments by different age groups/ gender groups/ marital groups. ###3.1 Percentage of Happy Moments by Category

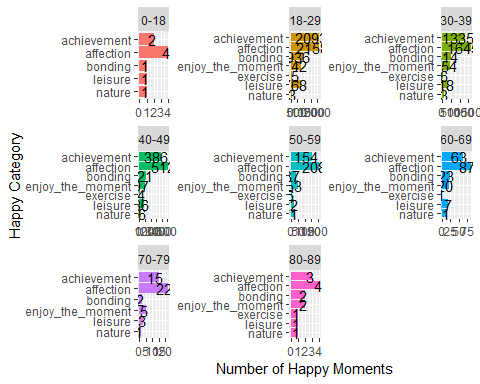
ggplot(data.frame(prop.table(table(newdata$predicted\_category))), aes(x=Var1, y = Freq\*100, fill = Var1)) +   
 geom\_bar(stat = 'identity') +   
 xlab('Happy Category') +   
 ylab('Percentage of Happy Moments (%)') +   
 geom\_text(aes(label=round(Freq\*100,2)), vjust=-0.25) +   
 ggtitle('Percentage of Happy Moments by Category')



In this part, we find that based on all the happy moments, people get their happiness from affenction most, and then the achivement, which is related to the wordcloud we generated above.

### 3.2 Happy Category by Age

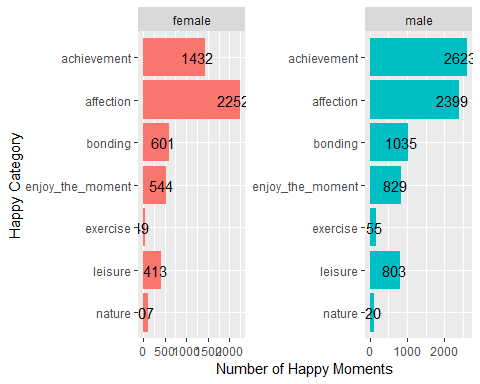
newdata$age <- as.numeric(as.character(newdata$age))  
ages <- newdata %>% select(wid, age, predicted\_category) %>% mutate(Age\_group = ifelse(age < 18, '0-18',ifelse(age < 30, '18-29', ifelse(age < 40, '30-39', ifelse(age < 50, '40-49', ifelse(age < 60, '50-59', ifelse(age < 70, '60-69', ifelse(age < 80, '70-79', ifelse(age < 90, '80-89', '90-99')))))))))  
  
ages <- ages %>% mutate(Age\_group = factor(Age\_group), predicted\_category = factor(predicted\_category, levels = rev(c( 'achievement', 'affection', 'bonding', 'enjoy\_the\_moment','exercise', 'leisure', 'nature'))))  
  
ages %>% filter(age < 100) %>% group\_by(Age\_group) %>% count(predicted\_category) %>% ggplot(aes(predicted\_category, n, fill = Age\_group)) + geom\_bar(stat='identity', show.legend = FALSE) + facet\_wrap(~Age\_group, scales = 'free') + xlab('Happy Category') + ylab('Number of Happy Moments') + geom\_text(aes(label = n), hjust = .73) + coord\_flip()



In this part, based on different age group, we can not explore significant difference bewteen different groups, they all get happiness from affection most, and second is the achievement.

### 3.3 Happy Category by Gender

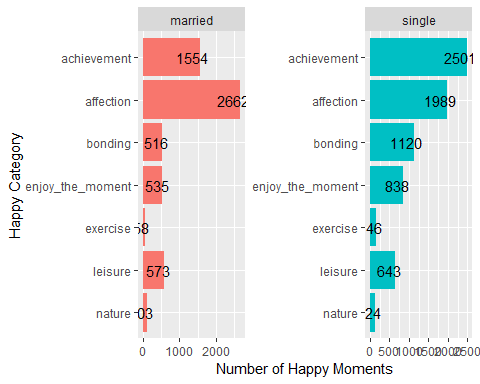
genders <- newdata %>% select(wid, gender, predicted\_category) %>% mutate(Gender\_group = ifelse(gender == "f" , 'female',ifelse(gender == "m", 'male',NA)))  
  
genders <- genders %>% mutate(Gender\_group = factor(Gender\_group), predicted\_category = factor(predicted\_category, levels = rev(c( 'achievement', 'affection', 'bonding', 'enjoy\_the\_moment','exercise', 'leisure', 'nature'))))  
  
genders %>% group\_by(Gender\_group) %>% count(predicted\_category) %>% ggplot(aes(predicted\_category, n, fill = Gender\_group)) + geom\_bar(stat='identity', show.legend = FALSE) + facet\_wrap(~Gender\_group, scales = 'free') + xlab('Happy Category') + ylab('Number of Happy Moments') + geom\_text(aes(label = n), hjust = .73) + coord\_flip()



In this part, we find that male get happiness from achievement most which is totally different from the results above, and also exercise accounts for a lot by male than female. That makes sense!

### 3.4 Happy Category by Marital Status

Marital <- newdata %>% select(wid, marital, predicted\_category) %>% mutate(Marital\_group = ifelse(marital == "married" , 'married',ifelse(marital == "single", 'single',NA)))  
  
Marital <- Marital %>% mutate(Marital\_group = factor(Marital\_group), predicted\_category = factor(predicted\_category, levels = rev(c( 'achievement', 'affection', 'bonding', 'enjoy\_the\_moment','exercise', 'leisure', 'nature'))))  
  
Marital %>% group\_by(Marital\_group) %>% count(predicted\_category) %>% ggplot(aes(predicted\_category, n, fill = Marital\_group)) + geom\_bar(stat='identity', show.legend = FALSE) + facet\_wrap(~Marital\_group, scales = 'free') + xlab('Happy Category') + ylab('Number of Happy Moments') + geom\_text(aes(label = n), hjust = .73) + coord\_flip()



In this part, we find that single people get happiness from achievement most and the affection is the second, which perhaphs means that single people have less happiness from affection whithout their own kids and husband(wife).

## LDA

#Find the sum of words in each Document  
#rowTotals <- apply(tdm , 1, sum)  
#tdm <- tdm[rowTotals > 0, ]  
#topic <- LDA(text\_wc\_df, k=5, method = "Gibbs",   
 #control = list(seed = 2018,  
 # burnin = 1000,  
 # thin = 100,  
 # iter = 1000,  
 # alpha = 0.5))  
#terms(topic,5)

## 4. Logistic Regrssion

In this part, we want to explore deeper in people's happy moments, we can apply logistic regression to build some classifiers to recognize people's gender/marital status/parenthood status according to their happy moment descriptions. ###4.1 Classifier for Gender

genderdata <- cbind(newdata[,4], newdata[,11])  
genderdata$dum <- ifelse(genderdata$gender == "f", 1, 0)  
head(genderdata$cleaned\_hm)

## [1] "We had a serious talk with some friends of ours who have been flaky lately. They understood and we had a good evening hanging out."  
## [2] "I meditated last night."   
## [3] "My grandmother start to walk from the bed after a long time."   
## [4] "when i received flowers from my best friend"   
## [5] "I went shopping"   
## [6] "The phone that I have ordered in a local online store was delivered this morning. I like the phone so much!"

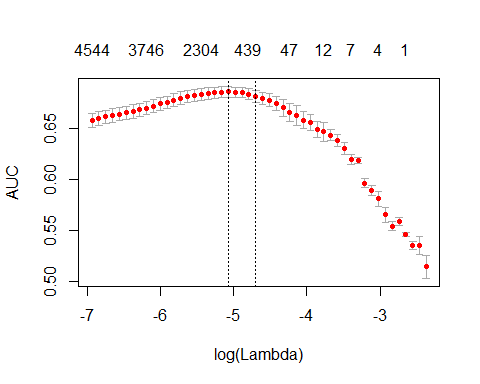
trainnum <- round(0.7\*nrow(genderdata))  
set.seed(2018)  
traingender <- genderdata[sample(c(1:nrow(genderdata)), size = trainnum),]  
testgender <- genderdata[-sample(c(1:nrow(genderdata)), size = trainnum),]  
## Vocabulary-based vectorization  
prep\_fun <- tolower  
tok\_fun <- word\_tokenizer  
  
it\_train <- itoken(traingender$cleaned\_hm,   
 preprocessor = prep\_fun,   
 tokenizer = tok\_fun,   
 genders = traingender$dum,   
 progressbar = FALSE)  
vocab <- create\_vocabulary(it\_train)  
train\_tokens <- traingender$cleaned\_hm %>%   
 prep\_fun %>%   
 tok\_fun  
it\_train <- itoken(train\_tokens,   
 genders = traingender$dum,  
 # turn off progressbar because it won't look nice in rmd  
 progressbar = FALSE)  
  
vocab <- create\_vocabulary(it\_train)  
vectorizer <- vocab\_vectorizer(vocab)  
t1 = Sys.time()  
dtm\_train = create\_dtm(it\_train, vectorizer)  
print(difftime(Sys.time(), t1, units = 'sec'))

## Time difference of 0.205008 secs

# Logistic Regression  
NFOLDS = 4  
t1 = Sys.time()  
glmnet\_classifier = cv.glmnet(x = dtm\_train, y = traingender$dum,   
 family = 'binomial',   
 alpha = 1, # L1 penalty  
 type.measure = "auc",# the area under ROC curve  
 nfolds = NFOLDS,# 5-fold cross-validation  
 thresh = 1e-3, # high value is less accurate, but has faster training  
 maxit = 1e3)# lower number of iterations for faster training  
print(difftime(Sys.time(), t1, units = 'sec'))

## Time difference of 2.300145 secs

plot(glmnet\_classifier)



print(paste("max AUC =", round(max(glmnet\_classifier$cvm), 4)))

## [1] "max AUC = 0.6856"

# Test the Classifier  
it\_test = testgender$cleaned\_hm %>%   
 prep\_fun %>% tok\_fun %>%   
 # turn off progressbar because it won't look nice in rmd  
 itoken(genders = testgender$dum, progressbar = FALSE)  
   
  
dtm\_test = create\_dtm(it\_test, vectorizer)  
  
preds = predict(glmnet\_classifier, dtm\_test, type = 'response')[,1]  
glmnet:::auc(testgender$dum, preds)

## [1] 0.7297626

In this part, we make a classifier to classify people's gender. First, split 70% data randomly as train data and the rest 30% are test data. And then apply the logistic regression for train data. At last test the calssifier on the test data. In this case, we find the the classifier accurate is about 0.7297626 tested on the test data, the classifier works pretty well.

### 4.2 Classifier for Marital Status

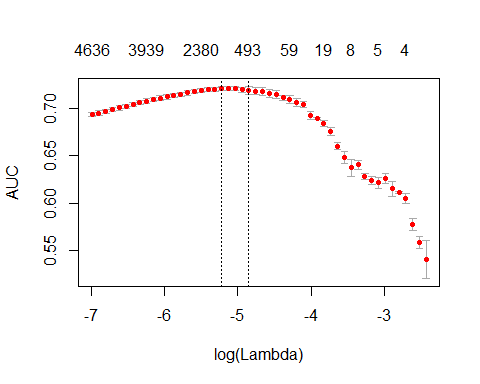
maritaldata <- cbind(newdata[,4], newdata[,12])  
maritaldata$dum <- ifelse(maritaldata$marital== "married", 1, 0)  
trainnum <- round(0.7\*nrow(maritaldata))  
set.seed(2018)  
trainmarital <- maritaldata[sample(c(1:nrow(maritaldata)), size = trainnum),]  
testmarital <- maritaldata[-sample(c(1:nrow(maritaldata)), size = trainnum),]  
prep\_funm <- tolower  
tok\_funm <- word\_tokenizer  
  
it\_trainm <- itoken(trainmarital$cleaned\_hm,   
 preprocessor = prep\_funm,   
 tokenizer = tok\_funm,   
 maritals = trainmarital$dum,   
 progressbar = FALSE)  
vocabm <- create\_vocabulary(it\_trainm)  
train\_tokensm<- trainmarital$cleaned\_hm %>%   
 prep\_funm %>%   
 tok\_funm  
it\_trainm <- itoken(train\_tokensm,   
 maritals = trainmarital$dum,  
 progressbar = FALSE)  
  
vocabm <- create\_vocabulary(it\_trainm)  
vectorizerm <- vocab\_vectorizer(vocabm)  
t1m = Sys.time()  
dtm\_trainm = create\_dtm(it\_trainm, vectorizerm)  
print(difftime(Sys.time(), t1m, units = 'sec'))

## Time difference of 0.1198089 secs

# Logistic Regression  
NFOLDS = 4  
t1m = Sys.time()  
glmnet\_classifierm = cv.glmnet(x = dtm\_trainm, y = trainmarital$dum,   
 family = 'binomial',   
 alpha = 1, # L1 penalty  
 type.measure = "auc",# the area under ROC curve  
 nfolds = NFOLDS,# 5-fold cross-validation  
 thresh = 1e-3, # high value is less accurate, but has faster training  
 maxit = 1e3)# lower number of iterations for faster training  
print(difftime(Sys.time(), t1m, units = 'sec'))

## Time difference of 2.05114 secs

plot(glmnet\_classifierm)



print(paste("max AUC =", round(max(glmnet\_classifierm$cvm), 4)))

## [1] "max AUC = 0.7209"

# Test the Classifier  
it\_testm = testmarital$cleaned\_hm %>%   
 prep\_funm %>% tok\_funm %>%   
 itoken(maritals = testmarital$dum, progressbar = FALSE)  
   
dtm\_testm = create\_dtm(it\_testm, vectorizerm)  
  
predsm = predict(glmnet\_classifierm, dtm\_testm, type = 'response')[,1]  
glmnet:::auc(testmarital$dum, predsm)

## [1] 0.7783142

The same method as 4.1, and the accurate of marital classifier is 0.7783142, it also works well. The graph shows the max AUC is 0.7209.

### 4.3 Classifier for Parenthood

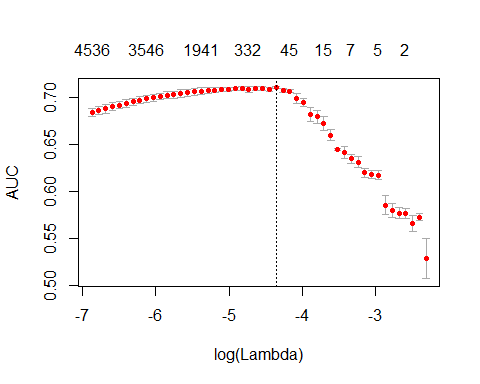
parentdata <- cbind(newdata[,4], newdata[,13])  
parentdata$dum <- ifelse(parentdata$parenthood== "y", 1, 0)  
trainnum <- round(0.7\*nrow(parentdata))  
set.seed(2018)  
trainparent <- parentdata[sample(c(1:nrow(parentdata)), size = trainnum),]  
testparent <- parentdata[-sample(c(1:nrow(parentdata)), size = trainnum),]  
prep\_funp <- tolower  
tok\_funp <- word\_tokenizer  
  
it\_trainp <- itoken(trainparent$cleaned\_hm,   
 preprocessor = prep\_funp,   
 tokenizer = tok\_funp,   
 parents = trainparent$dum,   
 progressbar = FALSE)  
vocabp <- create\_vocabulary(it\_trainp)  
train\_tokensp <- trainparent$cleaned\_hm %>%   
 prep\_funp %>%   
 tok\_funp  
it\_trainp <- itoken(train\_tokensp,   
 parents = trainparent$dum,  
 progressbar = FALSE)  
  
vocabp <- create\_vocabulary(it\_trainp)  
vectorizerp <- vocab\_vectorizer(vocabp)  
t1p = Sys.time()  
dtm\_trainp = create\_dtm(it\_trainp, vectorizerp)  
print(difftime(Sys.time(), t1p, units = 'sec'))

## Time difference of 0.141577 secs

# Logistic Regression  
NFOLDS = 4  
t1p = Sys.time()  
glmnet\_classifierp = cv.glmnet(x = dtm\_trainp, y = trainparent$dum,   
 family = 'binomial',   
 alpha = 1, # L1 penalty  
 type.measure = "auc",# the area under ROC curve  
 nfolds = NFOLDS,# 5-fold cross-validation  
 thresh = 1e-3, # high value is less accurate, but has faster training  
 maxit = 1e3)# lower number of iterations for faster training  
print(difftime(Sys.time(), t1p, units = 'sec'))

## Time difference of 1.737568 secs

plot(glmnet\_classifierp)



print(paste("max AUC =", round(max(glmnet\_classifierp$cvm), 4)))

## [1] "max AUC = 0.7101"

# Test the Classifier  
it\_testp = testparent$cleaned\_hm %>%   
 prep\_funp %>% tok\_funp %>%   
 itoken(parents = testparent$dum, progressbar = FALSE)  
   
dtm\_testp = create\_dtm(it\_testp, vectorizerp)  
  
predsp = predict(glmnet\_classifierp, dtm\_testp, type = 'response')[,1]  
glmnet:::auc(testparent$dum, predsp)

## [1] 0.7241155

The same method as 4.1, and the accurate of parent classifier is 0.7241155, it also works well.