Untitled

2023-09-13

Introduction to the project

From psychology's perspective, happiness leads to individuals' well-being, both physically and mentally; how happy we feel every day is the key to reduce chance of anxiety and psychological disorders. However, a lot of the times "the pursuit of happiness" is confusing: how exactly we should do in life to make our hearts feel happy? Is there a guidebook to help us make decisions?

Looking at people around me, especially considering my cultural background, marriage is a topic that is inevitable starting from age 25. My parents get curious about my future plan on marriage and how my relationship status goes. Their opinion, which I believe also resonates with lots of people, is that everyone will eventually get married and build a family. Just like people need water, air, and food to survive, getting into a marriage seems like a solid fact.

But I doubt it. I wonder if it is a necessity.

They mention a lot about marriage, but one topic is missing– if marriage brings happiness. Is life after getting married better or worse? If choosing to build a connection with partner for the rest of life, is that a choice that necessarily bring happiness?

Those questions and concerns lead me to this project.

Dataset- HappyDB

HappyDB is a corpus of more than 100,000 crowd-sourced happy moments on Amazon Mechanical Turk (MTurk.) For every task, it is asked that the MTurk workers describe 3 happy moments in the past 24 hours (or past 3 months.). The goal of the corpus is to advance the state of the art of understanding the causes of happiness that can be gleaned from text. There are 10,843 distinct users, 38,188 distinct words, and a total of 100,922 happy moments collected.

Cleaning the dataset (from text_processing,rmd)

We clean the text by converting all the letters to the lower case, and removing punctuation, numbers, empty words and extra white space. Stemming reduces a word to its word *stem*. We stem the words here and then convert the "tm" object to a "tidy" object for much faster processing. We also make a dictionary to look up the words corresponding to the stems. After removing stopwords provided by the "tidytext" package and also adding custom stopwords in context of our data, we combine the stems and the dictionary into the same "tidy" object. Lastly, we complete the stems by picking the corresponding word with the highest frequency. We want our processed words to resemble the structure of the original happy moments. So we paste the words together to form happy moments.

The final processed data is stored as hm_data and ready to be used for any kind of analysis.

```
urlfile<-'https://raw.githubusercontent.com/rit-public/HappyDB/master/happydb/data/cleaned_hm.csv'
hm_data <- read_csv(urlfile)
```

```
## Rows: 100535 Columns: 9
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr (5): reflection_period, original_hm, cleaned_hm, ground_truth_category, ...
## dbl (3): hmid, wid, num_sentence
## lgl (1): modified
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
summary(hm_data)
```

```
##       hmid             wid        reflection_period  original_hm
##  Min.   : 27673   Min.   :    1   Length:100535      Length:100535
##  1st Qu.: 52942   1st Qu.:  410   Class :character   Class :character
##  Median : 78204   Median : 1125   Mode  :character   Mode  :character
##  Mean   : 78214   Mean   : 2747
##  3rd Qu.:103490   3rd Qu.: 3507
##  Max.   :128766   Max.   :13839
##   cleaned_hm         modified        num_sentence    ground_truth_category
##  Length:100535      Mode :logical   Min.   : 1.000   Length:100535
##  Class :character   FALSE:2206      1st Qu.: 1.000   Class :character
##  Mode  :character   TRUE :98329     Median : 1.000   Mode  :character
##                                     Mean   : 1.341
##                                     3rd Qu.: 1.000
##                                     Max.   :69.000
##  predicted_category
##  Length:100535
##  Class :character
##  Mode  :character
##
##
##
```

```
urlfile1<-'https://raw.githubusercontent.com/rit-public/HappyDB/master/happydb/data/demographic.csv'
demo_data <- read_csv(urlfile1)
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 10844 Columns: 6
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (4): country, gender, marital, parenthood
## dbl (2): wid, age
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(demo_data)
```

```
## # A tibble: 6 x 6
##     wid   age country gender marital parenthood
##   <dbl> <dbl> <chr>   <chr>  <chr>   <chr>
## 1     1    37 USA     m      married y
## 2     2    29 IND     m      married y
## 3     3    25 IND     m      single  n
```

```
## 4     4    32 USA     m       married y
## 5     5    29 USA     m       married y
## 6     6    35 IND     m       married y
```

```
hm_data <- hm_data %>%
  inner_join(demo_data, by = "wid") %>%
  select(wid,
         original_hm,
         gender,
         marital,
         parenthood,
         reflection_period,
         age,
         country,
         ground_truth_category,
         predicted_category,
         text) %>%
  mutate(count = sapply(hm_data$text, wordcount)) %>%
  filter(marital %in% c("single", "married","divorced","separated","widowed")) %>%
  filter(gender %in% c("m", "f")) %>%
  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"))
head(hm_data)
```

```
## # A tibble: 6 x 12
##     wid original_hm    gender marital parenthood reflection_period   age country
##   <dbl> <chr>          <chr>  <chr>   <chr>      <fct>              <dbl> <chr>
## 1  2053 "I went on a ~ m      single  n          hours_24              35 USA
## 2     2 "I was happy ~ m      married y          hours_24              29 IND
## 3  1936 "I went to th~ f      married y          hours_24              30 USA
## 4   206 "We had a ser~ f      married n          hours_24              28 DNK
## 5  6227 "I went with ~ f      divorc~ y          hours_24              55 USA
## 6    45 "I meditated ~ m      single  n          hours_24              23 IND
## # i 4 more variables: ground_truth_category <chr>, predicted_category <chr>,
## #   text <chr>, count <int>
```

Wordcloud

```
wordcloud(words = hm_data$text, min.freq = 1, max.words=200, random.order=FALSE, rot.per=0.35, colors=bl
```

```
## Warning in tm_map.SimpleCorpus(corpus, tm::removePunctuation): transformation
## drops documents
```

```
## Warning in tm_map.SimpleCorpus(corpus, function(x) tm::removeWords(x,
## tm::stopwords())): transformation drops documents
```

```
## Warning in wordcloud(words = hm_data$text, min.freq = 1, max.words = 200, :
## wonderful could not be fit on page. It will not be plotted.
```

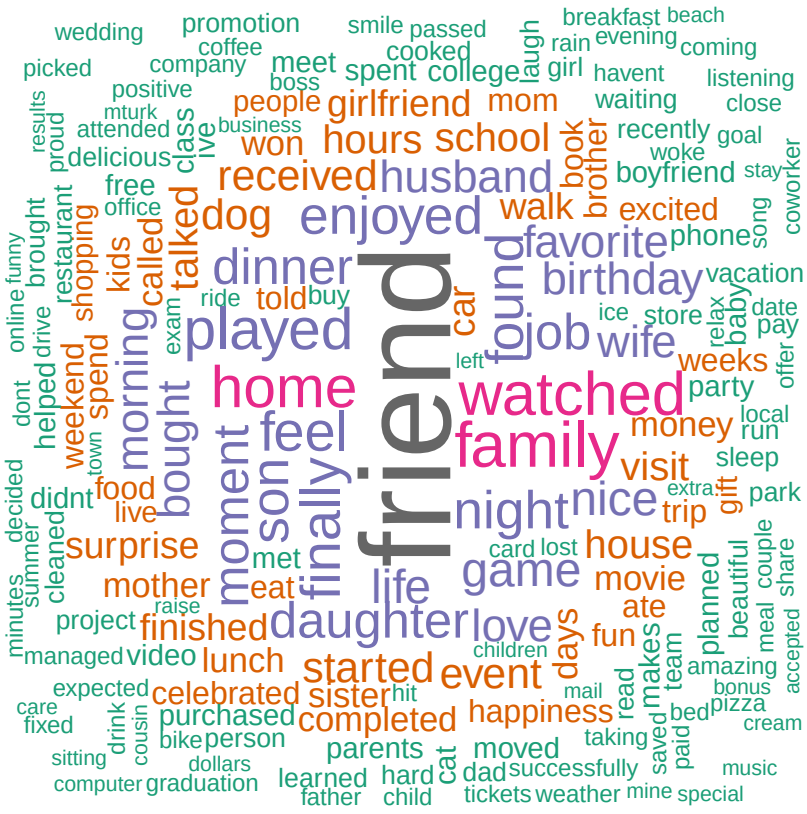
```
## Warning in wordcloud(words = hm_data$text, min.freq = 1, max.words = 200, :
## experience could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = hm_data$text, min.freq = 1, max.words = 200, :
## afternoon could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = hm_data$text, min.freq = 1, max.words = 200, :
## pretty could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = hm_data$text, min.freq = 1, max.words = 200, :
## world could not be fit on page. It will not be plotted.
```

```
## Warning in wordcloud(words = hm_data$text, min.freq = 1, max.words = 200, :
## students could not be fit on page. It will not be plotted.
```



Question 1

What are the most happy moments for single and married people? I start with dividing them into two groups based on if they are in a marriage at the moment. In this case, divorced and widowed are considered single because is currently not in a marriage; separated in still married because in an active marital status.

```
table(hm_data$marital)
```

```
##
##  divorced   married separated    single   widowed
##      3775     40959       630     53554       476
```

```
hm_data_married <- hm_data[hm_data$marital== c("married", "separated"),]
hm_data_single <- hm_data[hm_data$marital== c("divorced", "single", "widowed"),]
```

```
## Warning in hm_data$marital == c("divorced", "single", "widowed"): longer object
## length is not a multiple of shorter object length
```

```
bag_of_words_single <-  hm_data_single %>%
  unnest_tokens(word, text)
```

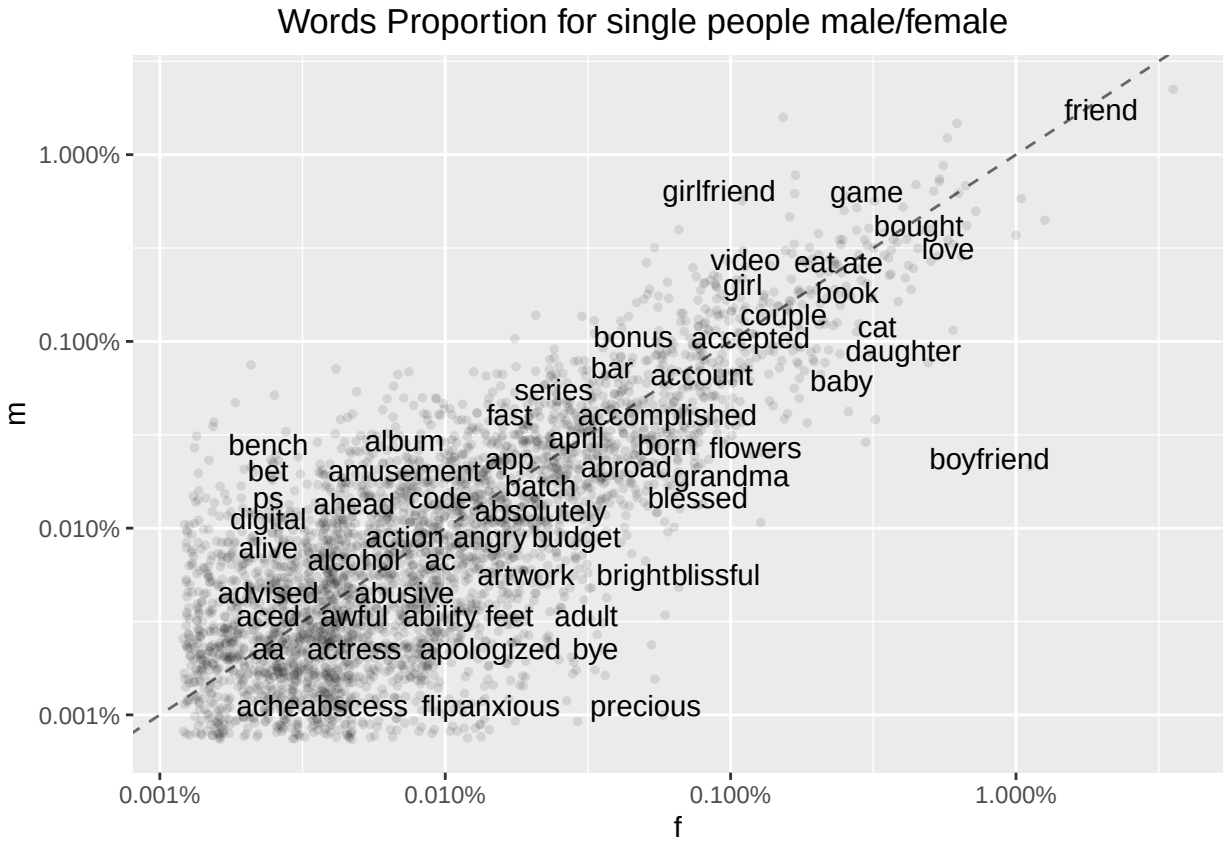
```r
word_count_single <- bag_of_words_single %>%
  count(word, sort = TRUE)

temp <- bag_of_words_single %>%
      count(gender, word) %>%
      group_by(gender) %>%
      mutate(proportion = n / sum(n)) %>%
      select(-n) %>%
      spread(gender, proportion)
ggplot(temp,
             aes_string(x = colnames(temp)[2], y = colnames(temp)[3]),
             color = abs(colnames(temp)[3] - colnames(temp)[2])) +
      geom_abline(color = "gray40", lty = 2) +
      geom_jitter(alpha = 0.1, size = 1, width = 0.3, height = 0.3) +
  labs(title="               Words Proportion for single people male/female")+
      geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
      scale_x_log10(labels = percent_format()) +
      scale_y_log10(labels = percent_format()) +
      scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75") +
      theme(legend.position="none")
```

```
## Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
## i Please use tidy evaluation idioms with `aes()`.
## i See also `vignette("ggplot2-in-packages")` for more information.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Removed 5118 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 5118 rows containing missing values (`geom_text()`).
```



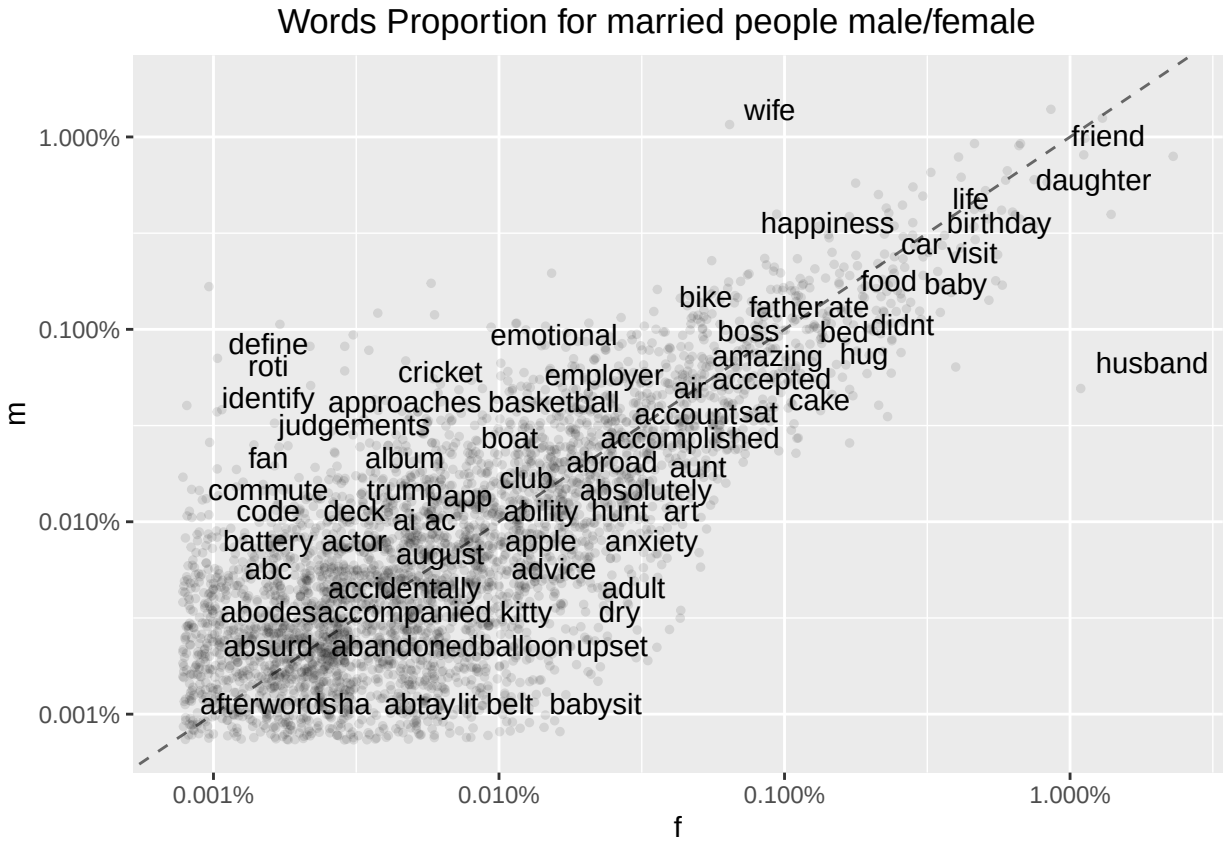

```
bag_of_words_married <-  hm_data_married %>%
  unnest_tokens(word, text)

word_count_married <- bag_of_words_married %>%
  count(word, sort = TRUE)

temp <- bag_of_words_married %>%
      count(gender, word) %>%
      group_by(gender) %>%
      mutate(proportion = n / sum(n)) %>%
      select(-n) %>%
      spread(gender, proportion)
ggplot(temp,
             aes_string(x = colnames(temp)[2], y = colnames(temp)[3]),
             color = abs(colnames(temp)[3] - colnames(temp)[2])) +
      geom_abline(color = "gray40", lty = 2) +
      geom_jitter(alpha = 0.1, size = 1, width = 0.3, height = 0.3) +
  labs(title="               Words Proportion for married people male/female")+
      geom_text(aes(label = word), check_overlap = TRUE, vjust = 1.5) +
      scale_x_log10(labels = percent_format()) +
      scale_y_log10(labels = percent_format()) +
      scale_color_gradient(limits = c(0, 0.001), low = "darkslategray4", high = "gray75") +
      theme(legend.position="none")
```

```
## Warning: Removed 5528 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 5528 rows containing missing values (`geom_text()`).
```




```
bag_of_words_single <-  hm_data_single %>%
  unnest_tokens(word, text)

word_count_single <- bag_of_words_single %>%
  count(word, sort = TRUE)

wordcloud(word_count_single$word,word_count_single$n ,
          scale=c(3,0.1),
          max.words=100,
          min.freq=1,
          random.order=FALSE,
          rot.per=0.3,
          use.r.layout=T,
          random.color=FALSE,
          colors=brewer.pal(9,"Oranges"))
```


```
bag_of_words_married <-  hm_data_married %>%
  unnest_tokens(word, text)

word_count_married <- bag_of_words_married %>%
  count(word, sort = TRUE)

wordcloud(word_count_married$word,word_count_married$n ,
          scale=c(3,0.1),
          max.words=100,
          min.freq=1,
          random.order=FALSE,
          rot.per=0.3,
          use.r.layout=T,
          random.color=FALSE,
          colors=brewer.pal(9,"Oranges"))
```