Sentiment Analysis for Mental Health - Data Preprocessing, Exploration, Visualization

2024-11-26

1.1

Loading the orginal dataset

```
# Set the file path for the CSV file
file_path2 <- "C:/Users/jivko/Documents/Data Analytics, Big Data, and Predictive Analytics/Pe
rsonal Project/Sentiment Analysis for Mental Health/Combined Data.csv"


# Read the CSV file into a dataframe
sentiment_analysis <- read.csv(file_path2, header = TRUE)
```

Printing the first few rows of the dataframe

```
# Print the first few rows of the dataframe
print(head(sentiment_analysis))
```

```
##   X
## 1 0
## 2 1
## 3 2
## 4 3
## 5 4
## 6 5
##                                                                        statement
## 1                                                                     oh my gosh
## 2               trouble sleeping, confused mind, restless heart. All out of tune
## 3 All wrong, back off dear, forward doubt. Stay in a restless and restless place
## 4                  I've shifted my focus to something else but I'm still worried
## 5       I'm restless and restless, it's been a month now, boy. What do you mean?
## 6   every break, you must be nervous, like something is wrong, but what the heck
##    status
## 1 Anxiety
## 2 Anxiety
## 3 Anxiety
## 4 Anxiety
## 5 Anxiety
## 6 Anxiety
```

Removing redundant X column

```
sentiment_analysis_use <- sentiment_analysis[, !names(sentiment_analysis) %in% c("X")]
```

```
print(head(sentiment_analysis_use))
```

```
##                                                                        statement
## 1                                                                     oh my gosh
## 2               trouble sleeping, confused mind, restless heart. All out of tune
## 3 All wrong, back off dear, forward doubt. Stay in a restless and restless place
## 4                  I've shifted my focus to something else but I'm still worried
## 5       I'm restless and restless, it's been a month now, boy. What do you mean?
## 6   every break, you must be nervous, like something is wrong, but what the heck
##    status
## 1 Anxiety
## 2 Anxiety
## 3 Anxiety
## 4 Anxiety
## 5 Anxiety
## 6 Anxiety
```

Structure of dataset

```
str(sentiment_analysis_use)
```

```
## 'data.frame':	53043 obs. of  2 variables:
##  $ statement: chr  "oh my gosh" "trouble sleeping, confused mind, restless heart. All out of tune" "All wrong, back off dear, forward doubt. Stay in a restless and restless place" "I've shifted my focus to something else but I'm still worried" ...
##  $ status   : chr  "Anxiety" "Anxiety" "Anxiety" "Anxiety" ...
```

Check for missing values

```
# Check for missing values in each column
colSums(is.na(sentiment_analysis_use))
```

```
## statement    status 
##         0         0
```

1.2

Distribution of mental health statuses

```
status_counts <- table(sentiment_analysis_use$status)
print(status_counts)
```

```
## 
##              Anxiety              Bipolar           Depression 
##                 3888                 2877                15404 
##               Normal Personality disorder               Stress 
##                16351                 1201                 2669 
##             Suicidal 
##                10653
```

Matching each status count to median (3888)

```
# Load the libraries
library(dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.4.2
```

```
## 
## Attaching package: 'dplyr'
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.4.2
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 4.4.2
```

```
## Loading required package: lattice
```

```r
# Assuming your dataset is called sentiment_analysis
# Get the distribution of the classes in the sentiment_analysis dataset
class_counts <- table(sentiment_analysis_use$status)

# Initialize an empty list to hold the balanced dataset
balanced_data <- list()

# Loop through each class
for (class in names(class_counts)) {
  # Subset the data for the current class
  class_data <- sentiment_analysis_use %>% filter(status == class)

  # If the class has fewer than 3888 samples, oversample
  if (nrow(class_data) < 3888) {
    # Oversample with replacement
    class_data <- class_data[sample(1:nrow(class_data), 3888, replace = TRUE), ]
  }

  # If the class has more than 3888 samples, undersample
  else if (nrow(class_data) > 3888) {
    # Undersample to 3888 samples
    class_data <- class_data[sample(1:nrow(class_data), 3888), ]
  }

  # Add the balanced class data to the list
  balanced_data[[class]] <- class_data
}

# Combine the balanced data
balanced_data <- do.call(rbind, balanced_data)

# Check the distribution of the balanced data
balanced_class_counts <- table(balanced_data$status)
print(balanced_class_counts)
```

```
##
##              Anxiety              Bipolar           Depression
##                 3888                 3888                 3888
##               Normal Personality disorder               Stress
##                 3888                 3888                 3888
##             Suicidal
##                 3888
```

Structure of balanced dataset

```r
str(balanced_data)
```

```
## 'data.frame':    27216 obs. of  2 variables:
##  $ statement: chr  "oh my gosh" "trouble sleeping, confused mind, restless heart. All out
of tune" "All wrong, back off dear, forward doubt. Stay in a restless and restless place" "I'
ve shifted my focus to something else but I'm still worried" ...
##  $ status   : chr  "Anxiety" "Anxiety" "Anxiety" "Anxiety" ...
```

1.3

Setting a fixed sample size per class (15% of 3888)

```
# Set a fixed sample size per class (e.g., 15% of 3888)
fixed_sample_size <- 583  # Round down to ensure consistency across classes

# Perform stratified sampling
sampled_balanced_data <- do.call(rbind, lapply(split(balanced_data, balanced_data$status), fu
nction(class_data) {
  class_data[sample(1:nrow(class_data), fixed_sample_size), ]
}))

# Check the new distribution
table(sampled_balanced_data$status)
```

```
## 
##              Anxiety              Bipolar           Depression 
##                  583                  583                  583 
##               Normal Personality disorder               Stress 
##                  583                  583                  583 
##             Suicidal 
##                  583 
```

Structure of sampled balanced dataset

```
str(sampled_balanced_data)
```

```
## 'data.frame':    4081 obs. of  2 variables:
##  $ statement: chr  "Why are you so nervous ²" "Lack of appetite for months..anxiety? Been
like this for a while now and dont know of it is anxiety or depressi"| __truncated__ "I’m dea
thly afraid of getting a brain aneurysm My mother died from one when I was a baby, and I neve
r really re"| __truncated__ "I'm confused when I've finished something, what's next? I feel l
ike everything is already there, but what's mis"| __truncated__ ...
##  $ status   : chr  "Anxiety" "Anxiety" "Anxiety" "Anxiety" ...
```

1.4

Preprocessing text data

```r
# Load required libraries
library(textstem)
```

```
## Warning: package 'textstem' was built under R version 4.4.2
```

```
## Loading required package: koRpus.lang.en
```

```
## Warning: package 'koRpus.lang.en' was built under R version 4.4.2
```

```
## Loading required package: koRpus
```

```
## Warning: package 'koRpus' was built under R version 4.4.2
```

```
## Loading required package: sylly
```

```
## Warning: package 'sylly' was built under R version 4.4.2
```

```
## For information on available language packages for 'koRpus', run
##
##   available.koRpus.lang()
##
## and see ?install.koRpus.lang()
```

```r
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.4.2
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
##
## Attaching package: 'tm'
## The following object is masked from 'package:koRpus':
##
##     readTagged
```

```
library(textclean)  # Ensure this library is loaded for replace_contraction()
```

```
## Warning: package 'textclean' was built under R version 4.4.2
```

```
library(quanteda)   # For tokenization and n-grams
```

```
## Warning: package 'quanteda' was built under R version 4.4.2
```

```
## Package version: 4.1.0
## Unicode version: 15.1
## ICU version: 74.1
```

```
## Parallel computing: 8 of 8 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:tm':
##
##     stopwords
```

```
## The following objects are masked from 'package:NLP':
##
##     meta, meta<-
```

```
## The following objects are masked from 'package:koRpus':
##
##     tokens, types
```

```
# Replace stemming with lemmatization
corpus <- Corpus(VectorSource(sampled_balanced_data$statement))

# Apply preprocessing steps
corpus <- tm_map(corpus, content_transformer(tolower))        # Convert to lowercase
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(tolower)):
## transformation drops documents
```

```
corpus <- tm_map(corpus, content_transformer(replace_contraction))  # Correctly use textclean
's function
```

```
## Warning in tm_map.SimpleCorpus(corpus,
## content_transformer(replace_contraction)): transformation drops documents
```

```
corpus <- tm_map(corpus, removePunctuation)                  # Remove punctuation
```

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

```
corpus <- tm_map(corpus, removeNumbers)                      # Remove numbers
```

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

```
corpus <- tm_map(corpus, stripWhitespace)                    # Remove extra whitespaces
```

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

```
# Remove non-alphanumeric characters (optional)
corpus <- tm_map(corpus, content_transformer(function(x) gsub("[^[:alnum:] ]", "", x)))
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(function(x)
## gsub("[^[:alnum:] ]", : transformation drops documents
```

```
# Remove URLs (optional)
corpus <- tm_map(corpus, content_transformer(function(x) gsub("http[s]?://\\S+", "", x)))
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(function(x)
## gsub("http[s]?://\\S+", : transformation drops documents
```

```
# Remove mentions and hashtags (optional, useful for social media data)
corpus <- tm_map(corpus, content_transformer(function(x) gsub("@\\S+|#\\S+", "", x)))
```

```
## Warning in tm_map.SimpleCorpus(corpus, content_transformer(function(x)
## gsub("@\\S+|#\\S+", : transformation drops documents
```

```r
# Remove stopwords
corpus <- tm_map(corpus, removeWords, stopwords("en"))
```

```
## Warning in tm_map.SimpleCorpus(corpus, removeWords, stopwords("en")):
## transformation drops documents
```

```r
# Correct spelling mistakes (optional, if needed)
# Use textclean or hunspell for spell correction if applicable

# Apply lemmatization
cleaned_statements <- data.frame(statement = lemmatize_strings(sapply(corpus, as.character)),
                                 status = sampled_balanced_data$status)

# Remove short texts (optional)
cleaned_statements <- cleaned_statements[nchar(as.character(cleaned_statements$statement)) >
3, ]

# View a sample of the cleaned data
head(cleaned_statements)
```

```
##
statement
## 1
nervous
## 2
lack appetite monthsanxiety like now do know anxiety depression lack appetite lose weight doc
always brush anxiety really bad blood work do multiple time know do show everything wrong red
flag loook just wanna enjoy eat whenever force do problem do get full feel either weird also
problem pooping guess im eat little
## 3
im deathly afraid get brain aneurysm mother die one baby never really realize recently since
ive research learn history family youre likely get one now im just dread randomly get head ac
he just pop im dead sometimes cant even sleep night im scare happen im asleep happen next day
night last wouldnt even know im think go doctor get scan stepmom say get one experience anyal
l symptom loss balance double vision loss consciousness thing im worry may even get point jus
t outright die spot randomly one day
## 4
I confuse I finish something next feel like everything already miss default restless
## 5
need support week hello friend long time reader first time poster I suffer health anxiety yea
r now I currently go one bad bout yet I get colonoscopy late week gastrointestinal issue I s
doctor assure multiple time expect colorectal cancer base symptom however can think can stop
googling symptom find people young diagnose cancer find people little symptom find cancer etc
just look advice get next day send much love everyone live like
## 6 stiff neck head ache worry bacterial meningitis ear infection day ago start get strange
head ache feel sudden short sharp pain leave side head follow extreme warmth panic attack sin
ce I sharp head pain come randomly couple second go usually leave middle side head sometimes
happen right I also bout pain back head good havent need take medicine pain come randomly pai
nful enough need relief ampxb first day happen also stiff shoulder neck assume anxiety headac
he stress muscle strain today neck pain get bad need advil feel stiff course last hour get si
gnificantly bad advil kick yet even though take min ago I really worry bacterial meningitis g
et ear infection couple day initial headache prescribe antibiotic take infection get good cou
ple day advice greatly appreciate
##    status
## 1 Anxiety
## 2 Anxiety
## 3 Anxiety
## 4 Anxiety
## 5 Anxiety
## 6 Anxiety
```

Structure of cleaned sampled balanced dataset

```
str(cleaned_statements)
```

```
## 'data.frame':    3956 obs. of  2 variables:
##  $ statement: chr  "nervous" "lack appetite monthsanxiety like now do know anxiety depress
ion lack appetite lose weight doc always brush anxi"| __truncated__ "im deathly afraid get br
ain aneurysm mother die one baby never really realize recently since ive research learn"| __t
runcated__ "I confuse I finish something next feel like everything already miss default restl
ess" ...
##  $ status   : chr  "Anxiety" "Anxiety" "Anxiety" "Anxiety" ...
```

2.1

TF of dataset

```r
# Load necessary libraries
library(dplyr)
library(tm)

# Assuming cleaned_statements is your data frame with text data and 'status' as the target va
riable

# Create a corpus from the cleaned statements
corpus <- Corpus(VectorSource(cleaned_statements$statement))

# Apply raw term frequency weighting (default behavior of DocumentTermMatrix)
dtm <- DocumentTermMatrix(corpus)

# Convert DTM to a numeric matrix
dtm_matrix <- as.matrix(dtm)

# Convert DTM to a data frame
tf_features <- as.data.frame(dtm_matrix)

# Add the target variable (status) to the features
tf_features$status <- cleaned_statements$status

# Function to get top words based on TF for each status
get_top_tf_words <- function(status_data, num_top_words = 10) {
  # Filter data for the given status
  status_data <- tf_features %>% filter(status == status_data)

  # Remove the 'status' column before calculating word frequencies
  status_data <- status_data[, -ncol(status_data)]

  # Ensure all values are numeric
  status_data <- as.data.frame(lapply(status_data, as.numeric))

  # Calculate the sum of raw term frequencies for each word
  word_freq <- colSums(status_data)

  # Sort the words by their raw term frequencies in decreasing order
  sorted_word_freq <- sort(word_freq, decreasing = TRUE)

  # Get the top N words based on raw term frequency
  top_words <- head(sorted_word_freq, num_top_words)

  return(top_words)
}

# List of unique statuses
statuses <- unique(tf_features$status)

# Get top 10 TF words for each status
top_tf_words_by_status <- lapply(statuses, function(status) get_top_tf_words(status, num_top_
words = 10))
```

```
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
```

```
# Display the top 10 TF words for each status
names(top_tf_words_by_status) <- statuses
top_tf_words_by_status
```

```
## $Anxiety
##    feel     get anxiety    like    just   think    know    time     can     day
##     801     679     570     549     509     360     340     332     316     305
##
## $Bipolar
##    feel    just    like     get    know bipolar    take    good    want     can
##     806     720     685     637     432     426     389     387     380     379
##
## $Depression
##  feel  just  like   get  want  life  know  good   can think
##   992   951   743   676   612   484   470   453   414   401
##
## $Normal
##    can   like   good    get   just    one   want really   time   take
##     56     49     47     43     39     37     33     26     26     26
##
## $`Personality disorder`
##   feel   like   just people    get   know    can   want  think   even
##    829    798    682    599    502    453    425    415    394    389
##
## $Stress
##    get stress   feel   just   like    can   time   know   work   good
##    531    490    484    400    391    372    286    284    244    219
##
## $Suicidal
## anymore    take    just    want    feel    like     get    life    know   think
##    1449    1427     745     692     583     511     493     416     378     359
```

TF - BARCHART

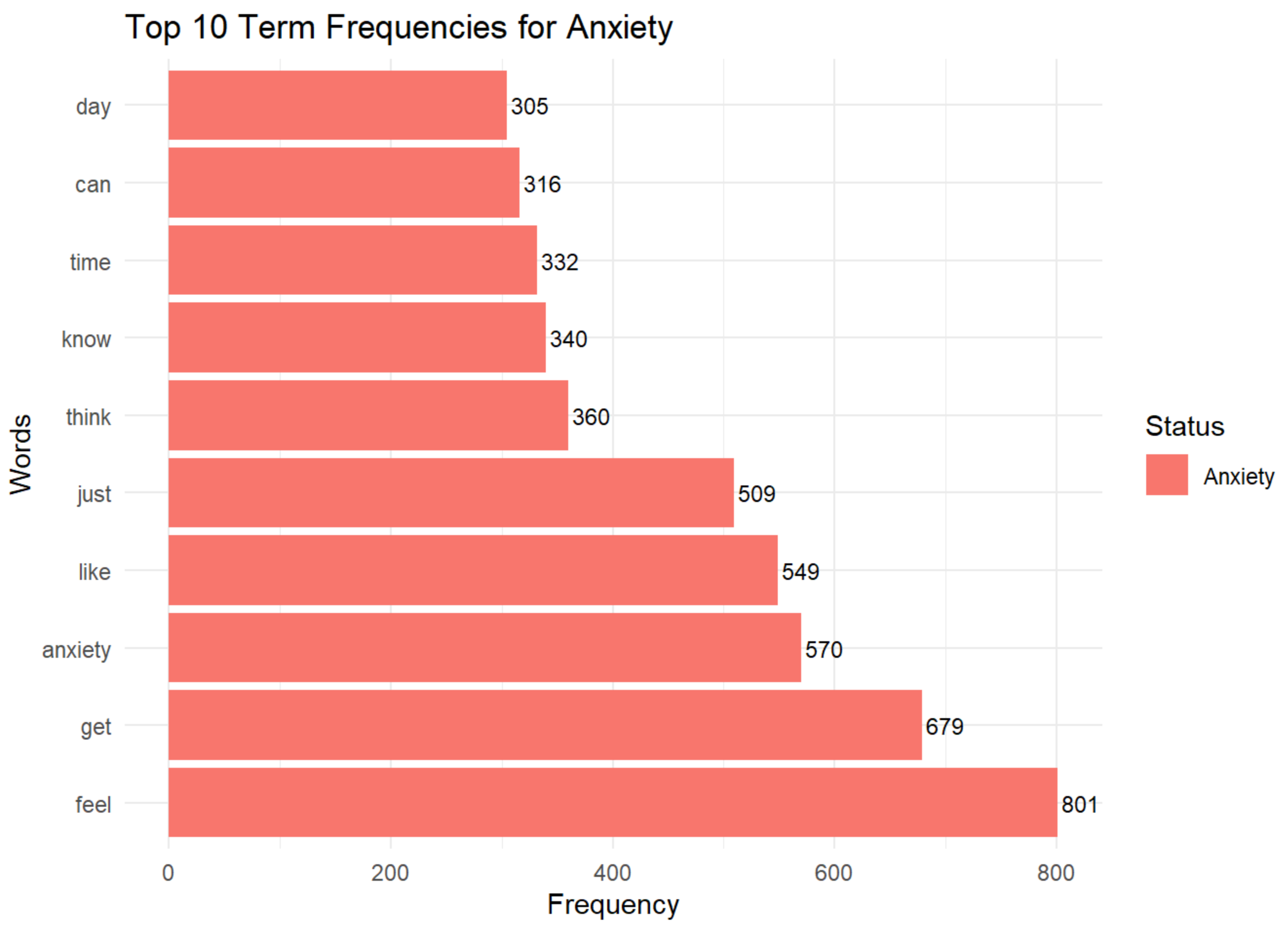
```
# Load necessary libraries
library(ggplot2)
library(purrr)
```

```
## Warning: package 'purrr' was built under R version 4.4.2
```

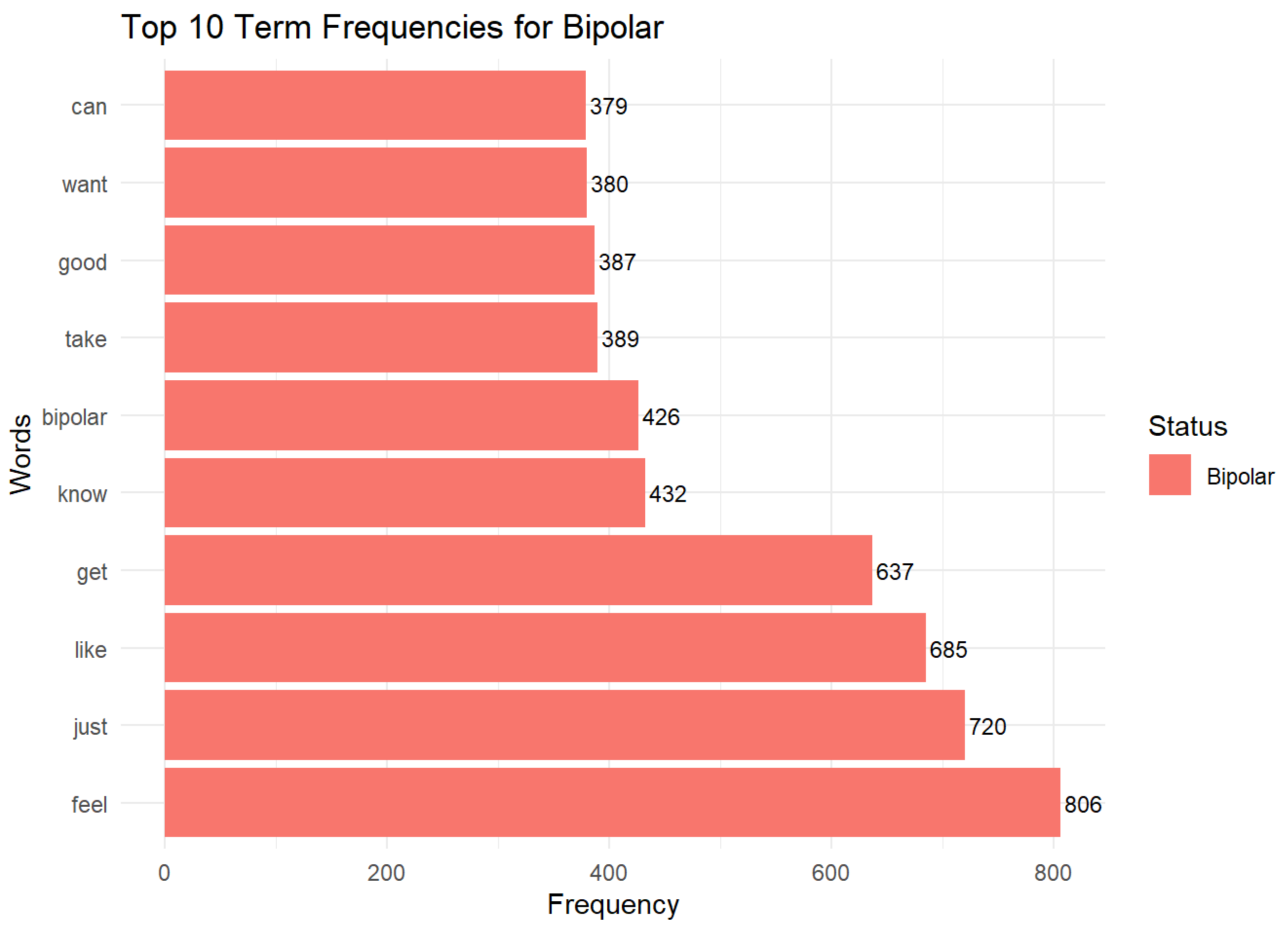
```
##
## Attaching package: 'purrr'
```

```
## The following object is masked from 'package:caret':
##
##     lift
```

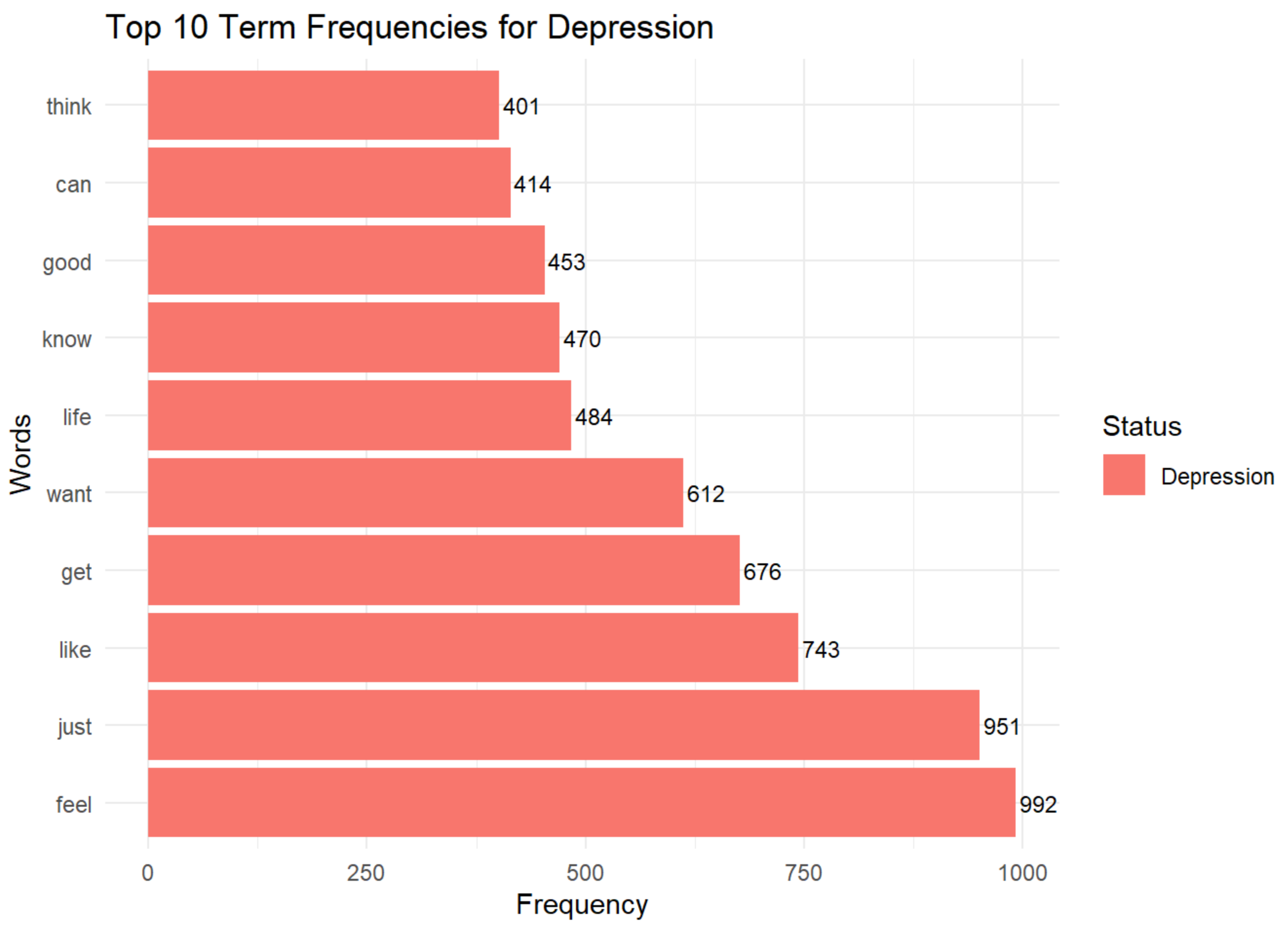
```r
library(dplyr)

# Convert the list to a combined data frame for plotting
plot_data <- lapply(names(top_tf_words_by_status), function(status) {
  data.frame(Status = status,
             Word = names(top_tf_words_by_status[[status]]),
             Frequency = unname(top_tf_words_by_status[[status]]))
}) %>%
  bind_rows()

# Create individual plots for each status with frequencies next to bars
status_plots <- plot_data %>%
  split(.$Status) %>%
  map(~ ggplot(.x, aes(x = reorder(Word, -Frequency), y = Frequency, fill = Status)) +
        geom_bar(stat = "identity") +
        geom_text(aes(label = Frequency), vjust = 0.5, hjust = -0.1, size = 3) +  # Add frequ
encies next to the bars
        coord_flip() +
        labs(title = paste("Top 10 Term Frequencies for", .x$Status[1]),
             x = "Words", y = "Frequency") +
        theme_minimal())

# View individual plots by referencing their names
# Example: Print the plot for "Anxiety"
status_plots$Anxiety
```



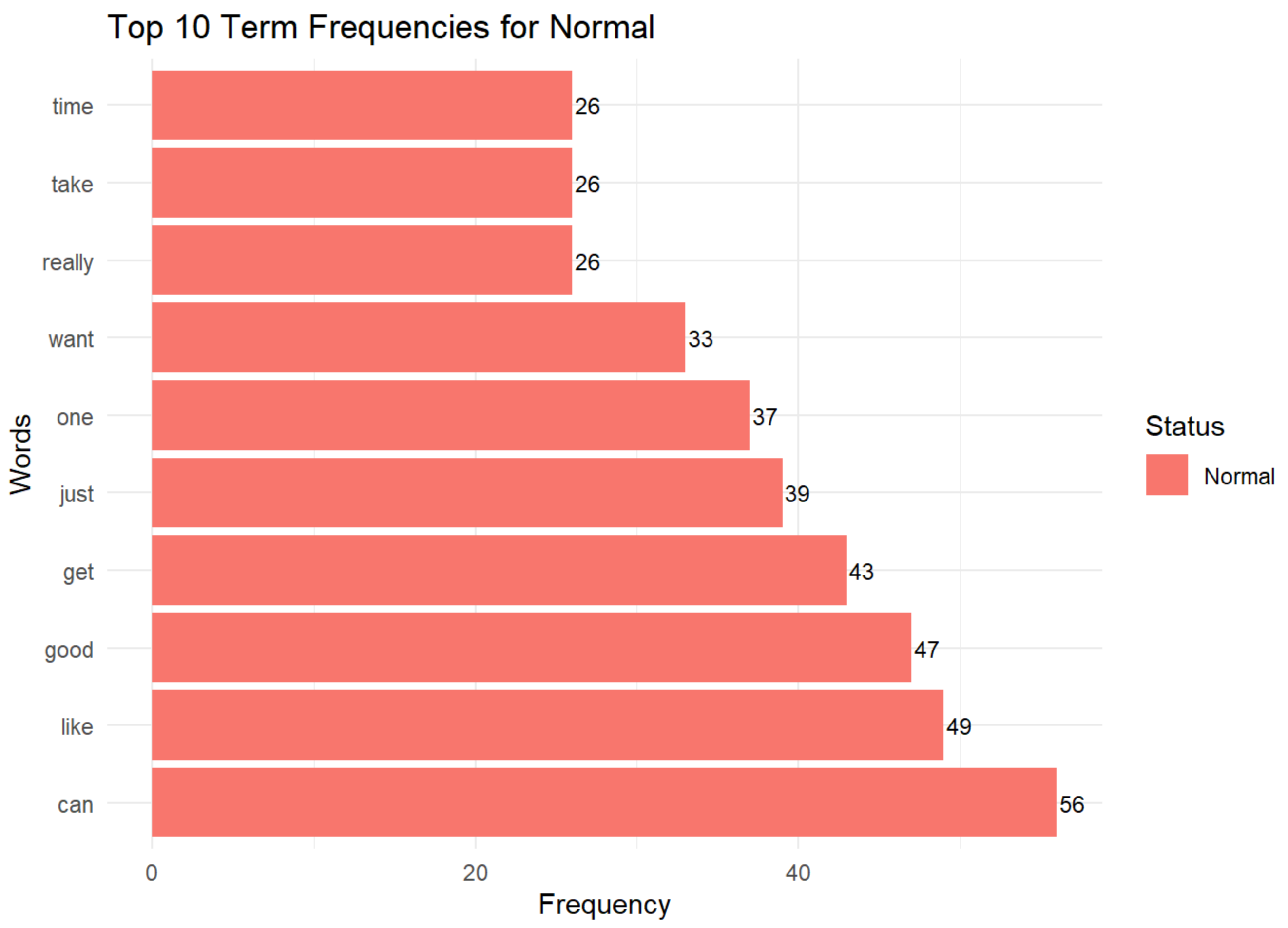
```
# Example: Print the plot for "Bipolar"
status_plots$Bipolar
```



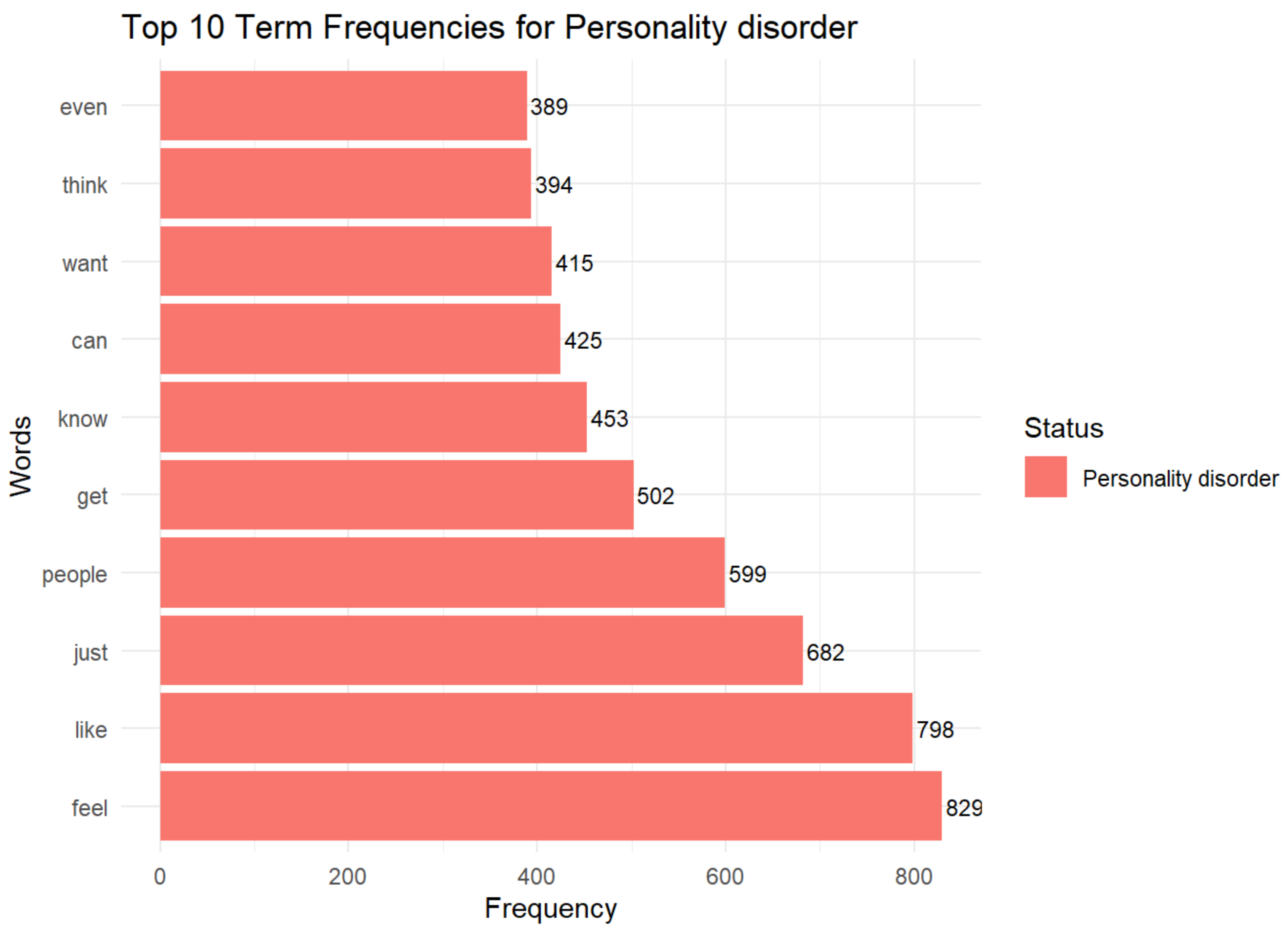
```r
# Example: Print the plot for "Depression"
status_plots$Depression
```



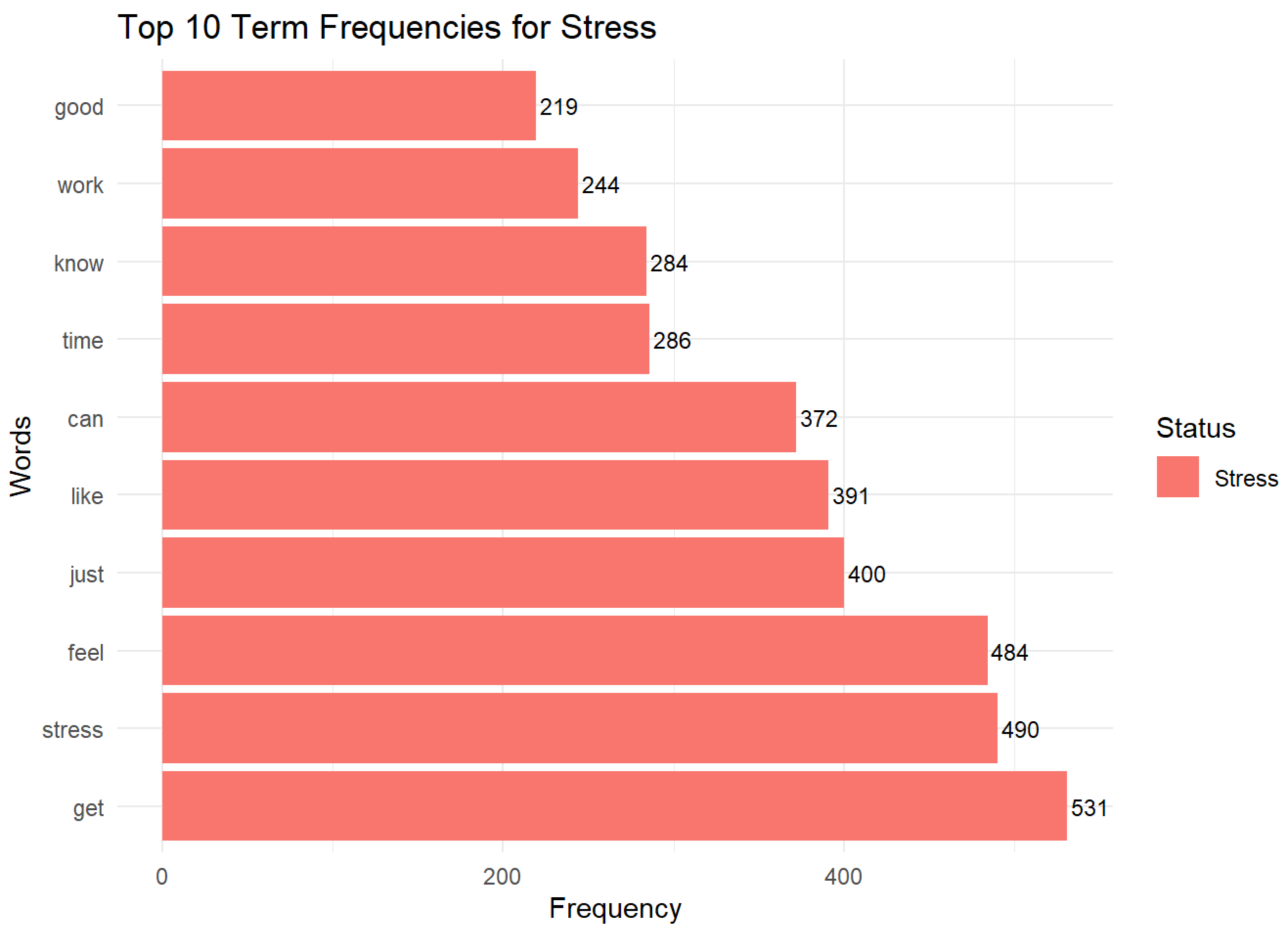
```
# Example: Print the plot for "Normal"
status_plots$Normal
```



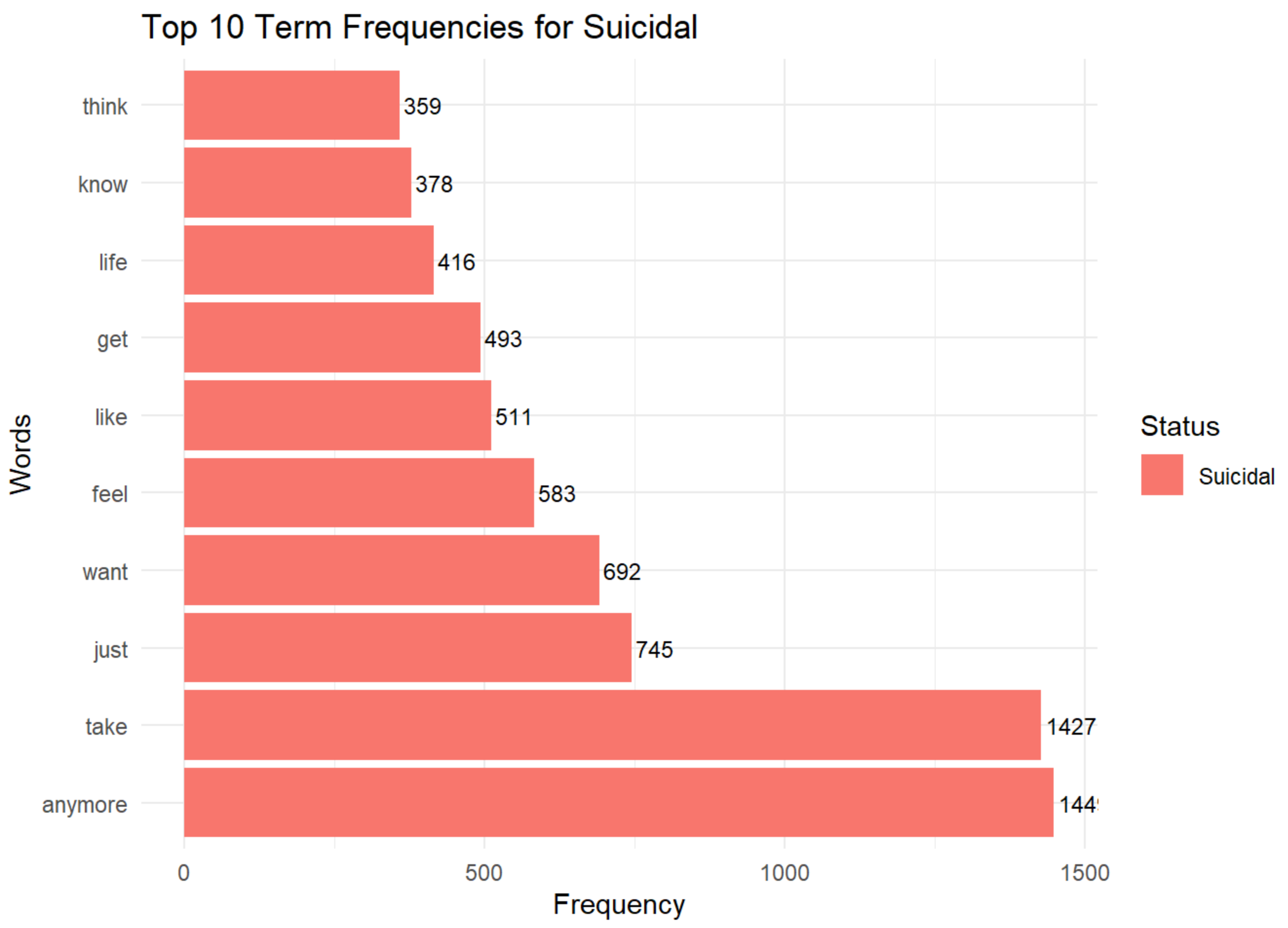
```
# Example: Print the plot for "Personality disorder"
status_plots$`Personality disorder`
```



```
# Example: Print the plot for "Stress"
status_plots$Stress
```



```
# Example: Print the plot for "Suicidal"
status_plots$Suicidal
```



2.2

Top TF-IDF Words by Status

```
# Load necessary libraries
library(dplyr)
library(tm)
library(SnowballC)
library(caret)

# Assuming cleaned_statements is your data frame with text data and 'status' as the target va
riable

# Create a corpus from the cleaned statements
corpus <- Corpus(VectorSource(cleaned_statements$statement))

# Apply TF-IDF weighting
dtm <- DocumentTermMatrix(corpus, control = list(weighting = weightTfIdf))
```

```
## Warning in TermDocumentMatrix.SimpleCorpus(x, control): custom functions are
## ignored
## Warning in weighting(x): empty document(s): 1824 1890
```

```
# Convert DTM to a numeric matrix
dtm_matrix <- as.matrix(dtm)

# Convert DTM to a data frame
tfidf_features <- as.data.frame(dtm_matrix)

# Add the target variable (status) to the features
tfidf_features$status <- cleaned_statements$status

# Function to get top words based on TF-IDF for each status
get_top_tfidf_words <- function(status_data, num_top_words = 10) {
  # Filter data for the given status
  status_data <- tfidf_features %>% filter(status == status_data)

  # Remove the 'status' column before calculating word frequencies
  status_data <- status_data[, -ncol(status_data)]

  # Ensure all values are numeric
  status_data <- as.data.frame(lapply(status_data, as.numeric))

  # Calculate the sum of TF-IDF scores for each word
  word_freq <- colSums(status_data)

  # Sort the words by their TF-IDF scores in decreasing order
  sorted_word_freq <- sort(word_freq, decreasing = TRUE)

  # Get the top N words based on TF-IDF score
  top_words <- head(sorted_word_freq, num_top_words)

  return(top_words)
}

# List of unique statuses
statuses <- unique(tfidf_features$status)

# Get top 10 TF-IDF words for each status
top_words_per_status <- lapply(statuses, function(status) get_top_tfidf_words(status, num_top
_words = 10))
```

```
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
```

```
# Display the top 10 TF-IDF words for each status
names(top_words_per_status) <- statuses
top_words_per_status
```

```
## $Anxiety
## restless    worry  nervous  anxiety  anxious     feel    heart     pain
## 54.62358 34.54000 30.99318 22.77122 14.52823 13.75523 11.86430 11.48100
##    sleep      get
## 11.07568 10.63018
##
## $Bipolar
##    bipolar    episode      manic       take      sleep       feel medication
##  21.707255  16.111318  12.955134  11.217149  11.117063  10.741150  10.677332
##       meds       just    lithium
##  10.259298   9.925008   9.848964
##
## $Depression
## depression       feel       just       want       life        don       like
##  20.433115  16.373187  15.019920  14.496964  13.379186  12.246087  11.384775
##        get       know       good
##  10.244401   9.557587   9.418175
##
## $Normal
##  morning tomorrow     good      yes     cool     miss dreamies     nice
## 21.09334 18.21328 17.53134 16.99105 16.68597 15.01088 13.81982 13.00406
##     quot      bun
## 12.95446 11.94983
##
## $`Personality disorder`
##          avpd        people          view          like          feel
##     28.924951     16.457090     13.462365     12.313882     11.328573
##      disorder        social          make hypochondrium         think
##      9.786664      9.726000      9.129237      8.949827      8.898592
##
## $Stress
##    stress       get      work      feel      help       can      like      know
## 37.187586 11.324293 10.363483  9.957410  9.903049  9.700702  8.826499  8.672272
##      just      time
##  8.436788  8.311070
##
## $Suicidal
##     want     kill     fuck      die  anymore     hate     life     just
## 23.65549 22.79657 20.78448 18.88732 18.14649 15.99886 14.91348 14.53012
##     tire     live
## 12.67888 12.60583
```

TF-IDF - BARCHART

```r
# Load necessary libraries
library(dplyr)
library(tm)
library(SnowballC)
library(caret)
library(ggplot2)
library(purrr)  # Load purrr for the 'map' function

# Assuming cleaned_statements is your data frame with text data and 'status' as the target va
riable

# Create a corpus from the cleaned statements
corpus <- Corpus(VectorSource(cleaned_statements$statement))

# Apply TF-IDF weighting
dtm <- DocumentTermMatrix(corpus, control = list(weighting = weightTfIdf))
```

```
## Warning in TermDocumentMatrix.SimpleCorpus(x, control): custom functions are
## ignored
```

```
## Warning in weighting(x): empty document(s): 1824 1890
```

```
# Convert DTM to a numeric matrix
dtm_matrix <- as.matrix(dtm)

# Convert DTM to a data frame
tfidf_features <- as.data.frame(dtm_matrix)

# Add the target variable (status) to the features
tfidf_features$status <- cleaned_statements$status

# Function to get top words based on TF-IDF for each status
get_top_tfidf_words <- function(status_data, num_top_words = 10) {
  # Filter data for the given status
  status_data <- tfidf_features %>% filter(status == status_data)

  # Remove the 'status' column before calculating word frequencies
  status_data <- status_data[, -ncol(status_data)]

  # Ensure all values are numeric
  status_data <- as.data.frame(lapply(status_data, as.numeric))

  # Calculate the sum of TF-IDF scores for each word
  word_freq <- colSums(status_data)

  # Sort the words by their TF-IDF scores in decreasing order
  sorted_word_freq <- sort(word_freq, decreasing = TRUE)

  # Get the top N words based on TF-IDF score
  top_words <- head(sorted_word_freq, num_top_words)

  return(top_words)
}

# List of unique statuses
statuses <- unique(tfidf_features$status)

# Get top 10 TF-IDF words for each status
top_words_per_status <- lapply(statuses, function(status) get_top_tfidf_words(status, num_top
_words = 10))
```

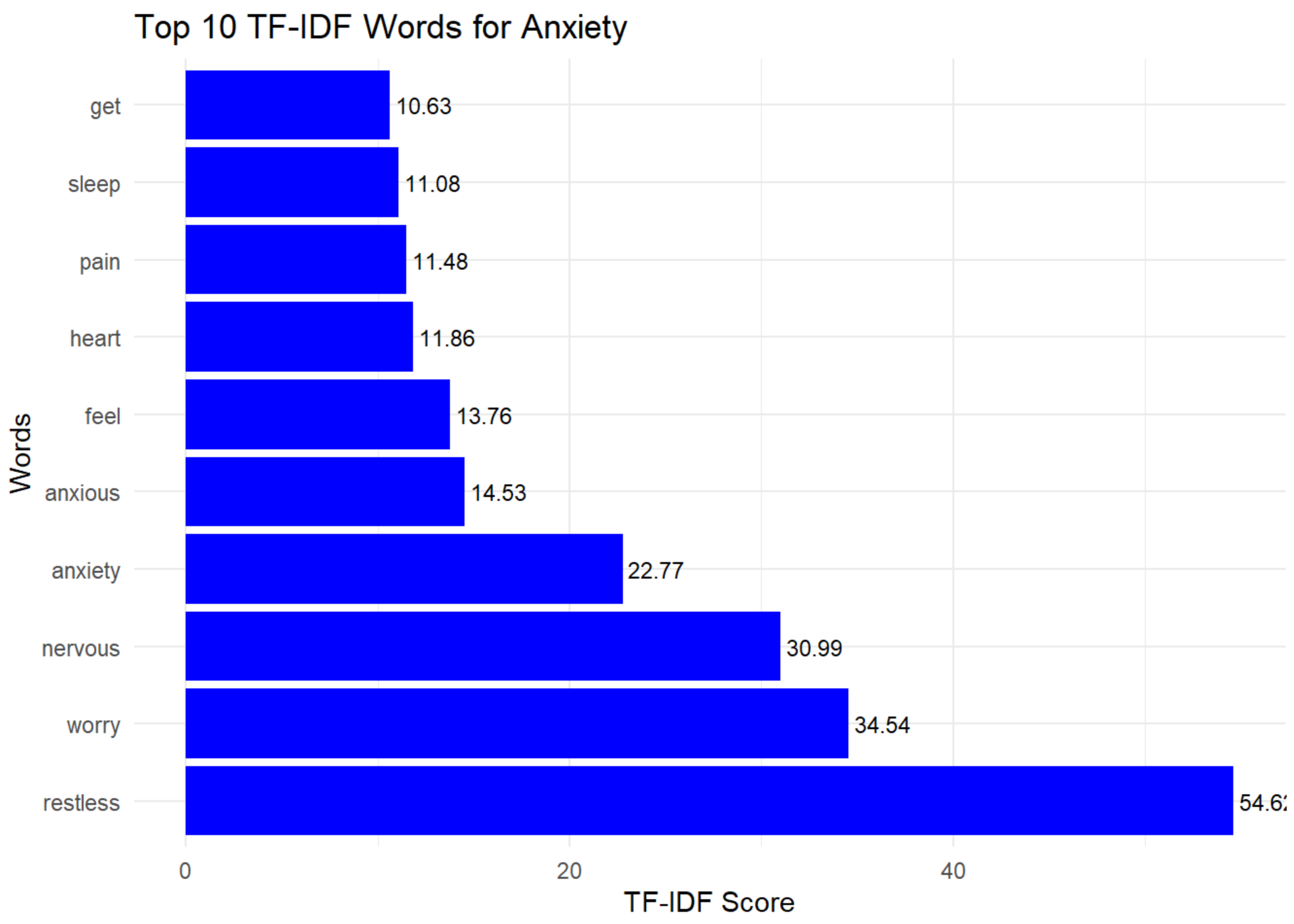
```
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
```

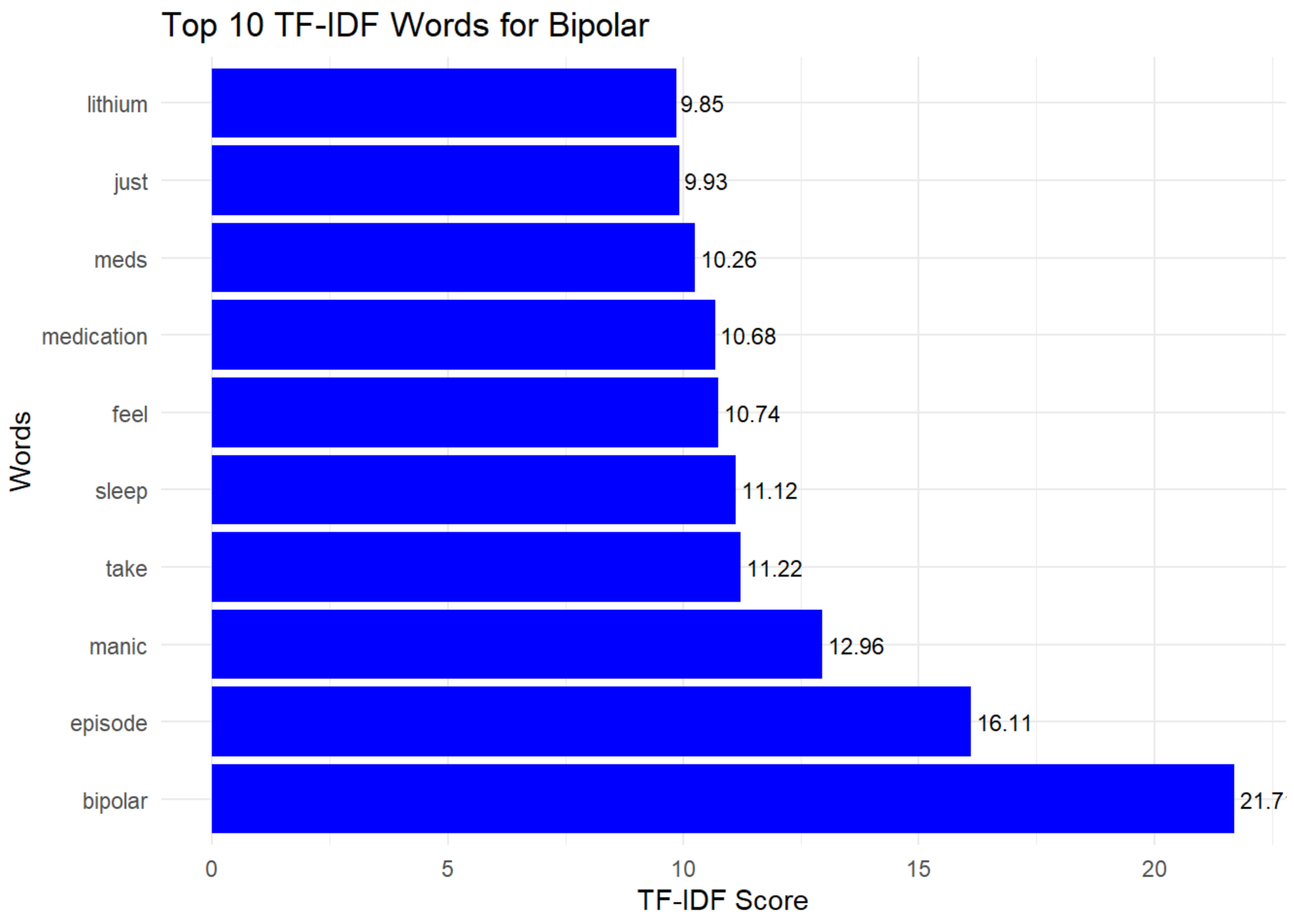
```
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
## Warning in lapply(status_data, as.numeric): NAs introduced by coercion
```

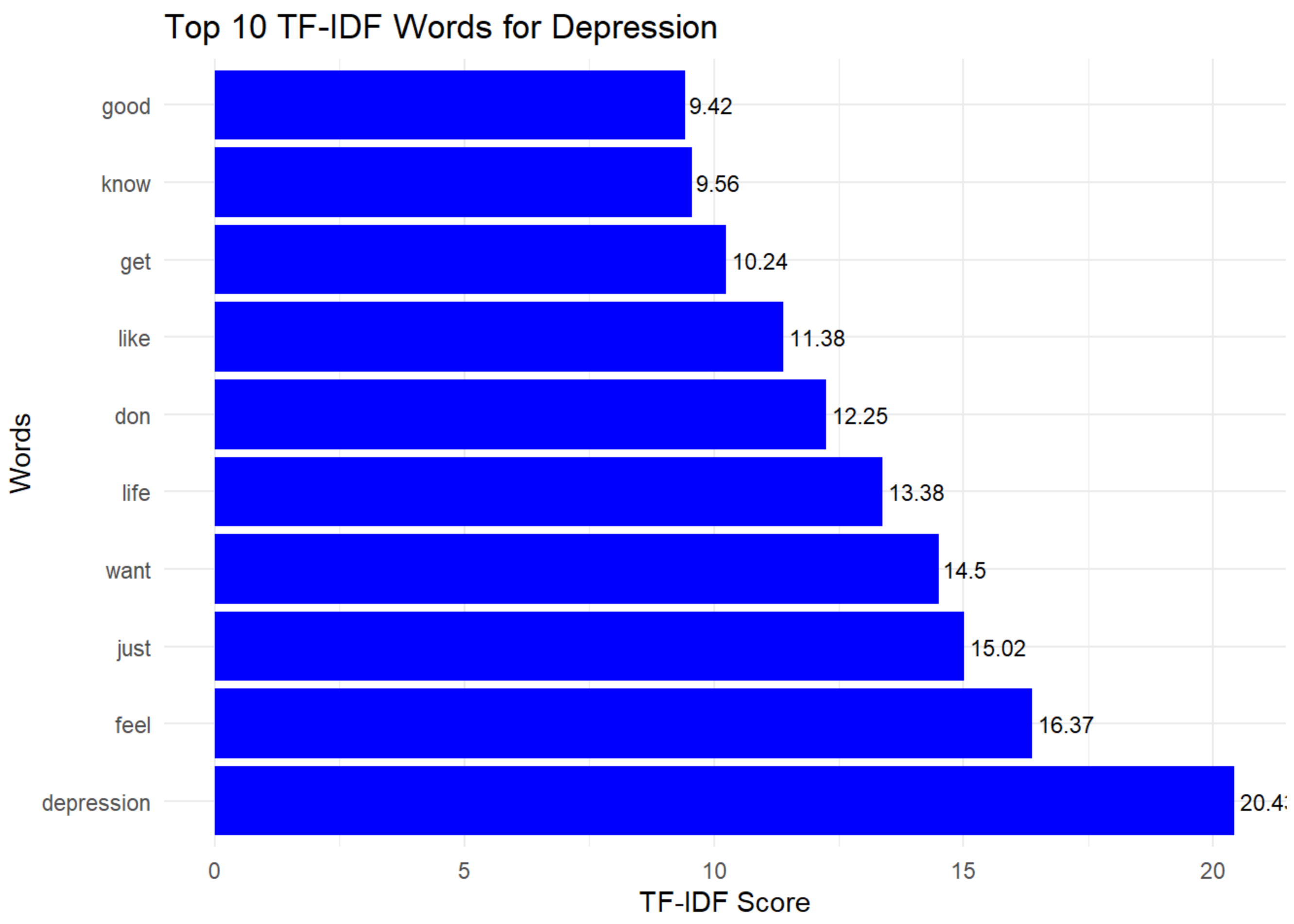
```
# Display the top 10 TF-IDF words for each status
names(top_words_per_status) <- statuses
top_words_per_status
```

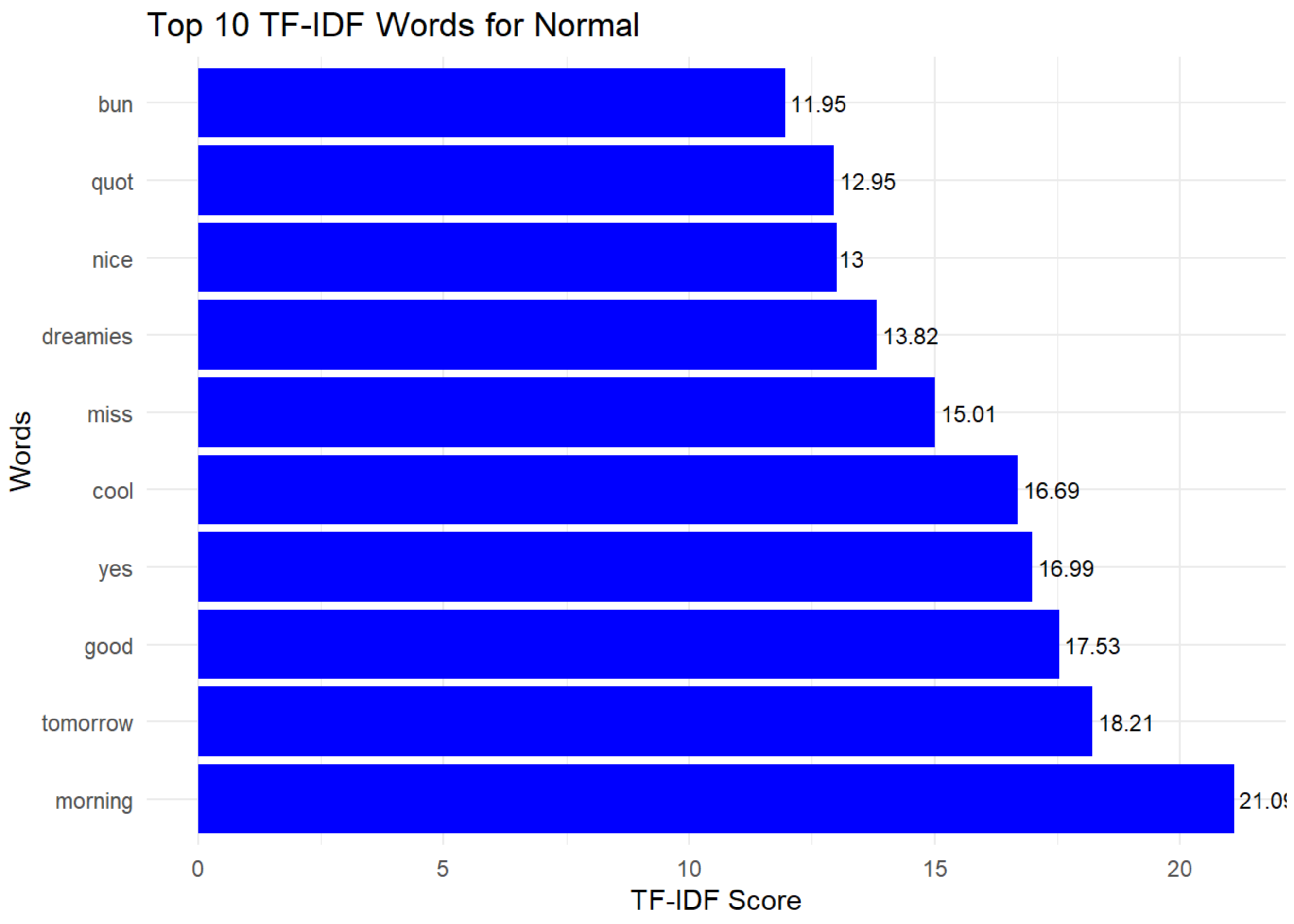
```
## $Anxiety
## restless    worry  nervous  anxiety  anxious     feel    heart     pain
## 54.62358 34.54000 30.99318 22.77122 14.52823 13.75523 11.86430 11.48100
##    sleep      get
## 11.07568 10.63018
##
## $Bipolar
##    bipolar    episode      manic       take      sleep       feel medication
##  21.707255  16.111318  12.955134  11.217149  11.117063  10.741150  10.677332
##       meds       just    lithium
##  10.259298   9.925008   9.848964
##
## $Depression
## depression       feel       just       want       life        don       like
##  20.433115  16.373187  15.019920  14.496964  13.379186  12.246087  11.384775
##        get       know       good
##  10.244401   9.557587   9.418175
##
## $Normal
##  morning tomorrow     good      yes     cool     miss dreamies     nice
## 21.09334 18.21328 17.53134 16.99105 16.68597 15.01088 13.81982 13.00406
##     quot      bun
## 12.95446 11.94983
##
## $`Personality disorder`
##          avpd        people          view          like          feel
##     28.924951     16.457090     13.462365     12.313882     11.328573
##      disorder        social          make hypochondrium         think
##      9.786664      9.726000      9.129237      8.949827      8.898592
##
## $Stress
##    stress       get      work      feel      help       can      like      know
## 37.187586 11.324293 10.363483  9.957410  9.903049  9.700702  8.826499  8.672272
##      just      time
##  8.436788  8.311070
##
## $Suicidal
##     want     kill     fuck      die  anymore     hate     life     just
## 23.65549 22.79657 20.78448 18.88732 18.14649 15.99886 14.91348 14.53012
##     tire     live
## 12.67888 12.60583
```

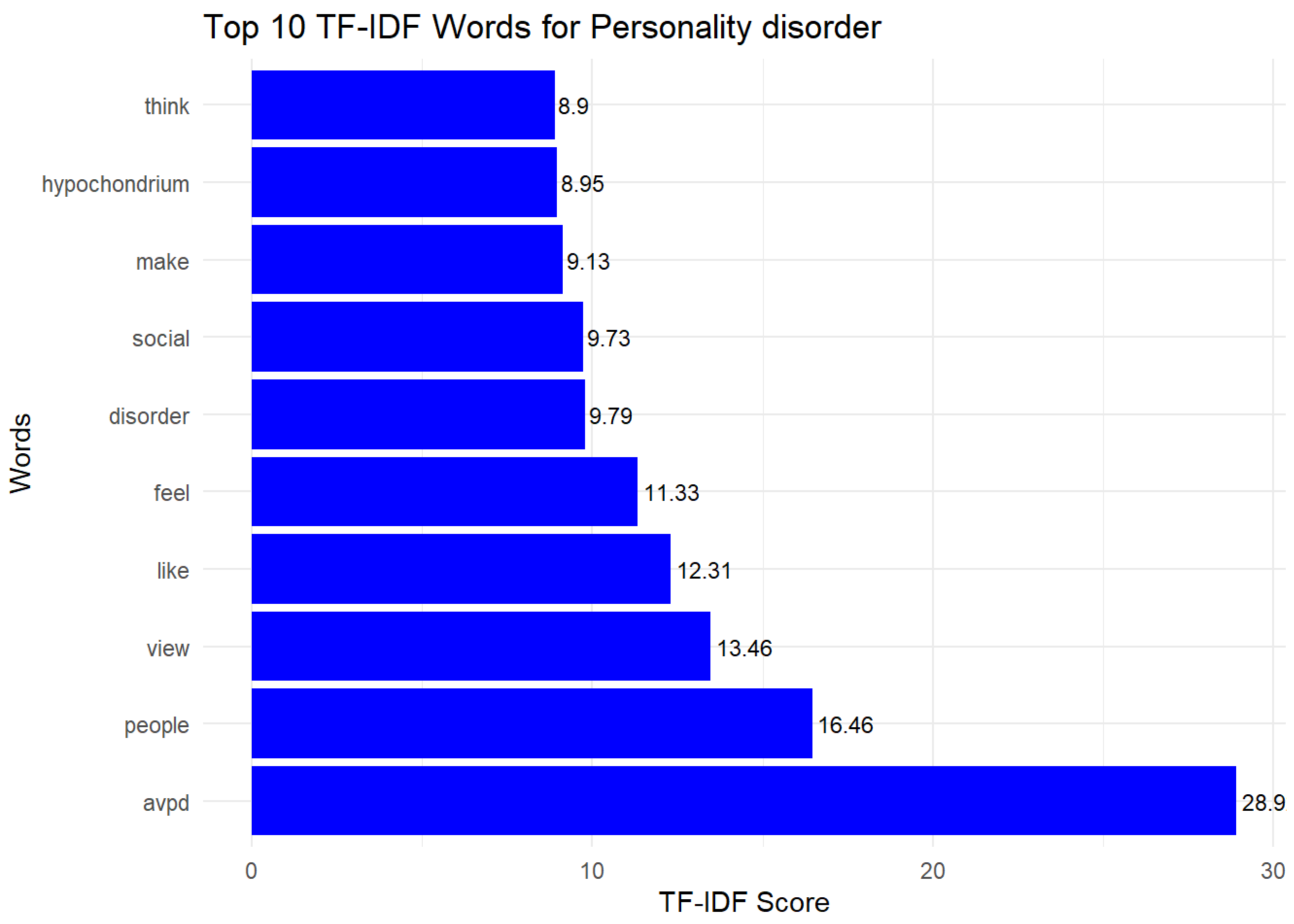
```r
# TF-IDF Bar Chart for each status

# Convert the list to a combined data frame for plotting TF-IDF words
plot_tfidf_data <- lapply(names(top_words_per_status), function(status) {
  data.frame(Status = status,
             Word = names(top_words_per_status[[status]]),
             TF_IDF = unname(top_words_per_status[[status]]))
}) %>%
  bind_rows()

# Create individual plots for each status with frequencies next to bars
status_tfidf_plots <- plot_tfidf_data %>%
  split(.$Status) %>%
  map(~ ggplot(.x, aes(x = reorder(Word, -TF_IDF), y = TF_IDF)) +  # Remove 'fill' aesthetic
        geom_bar(stat = "identity", fill = "blue") +  # Set bars color to blue
        geom_text(aes(label = round(TF_IDF, 2)), vjust = 0.5, hjust = -0.1, size = 3) +  # Ad
d TF-IDF scores next to the bars
        coord_flip() +
        labs(title = paste("Top 10 TF-IDF Words for", .x$Status[1]),
             x = "Words", y = "TF-IDF Score") +
        theme_minimal())

# Loop through each status and print the corresponding plot
for(status in names(status_tfidf_plots)) {
  print(status_tfidf_plots[[status]])
}
```

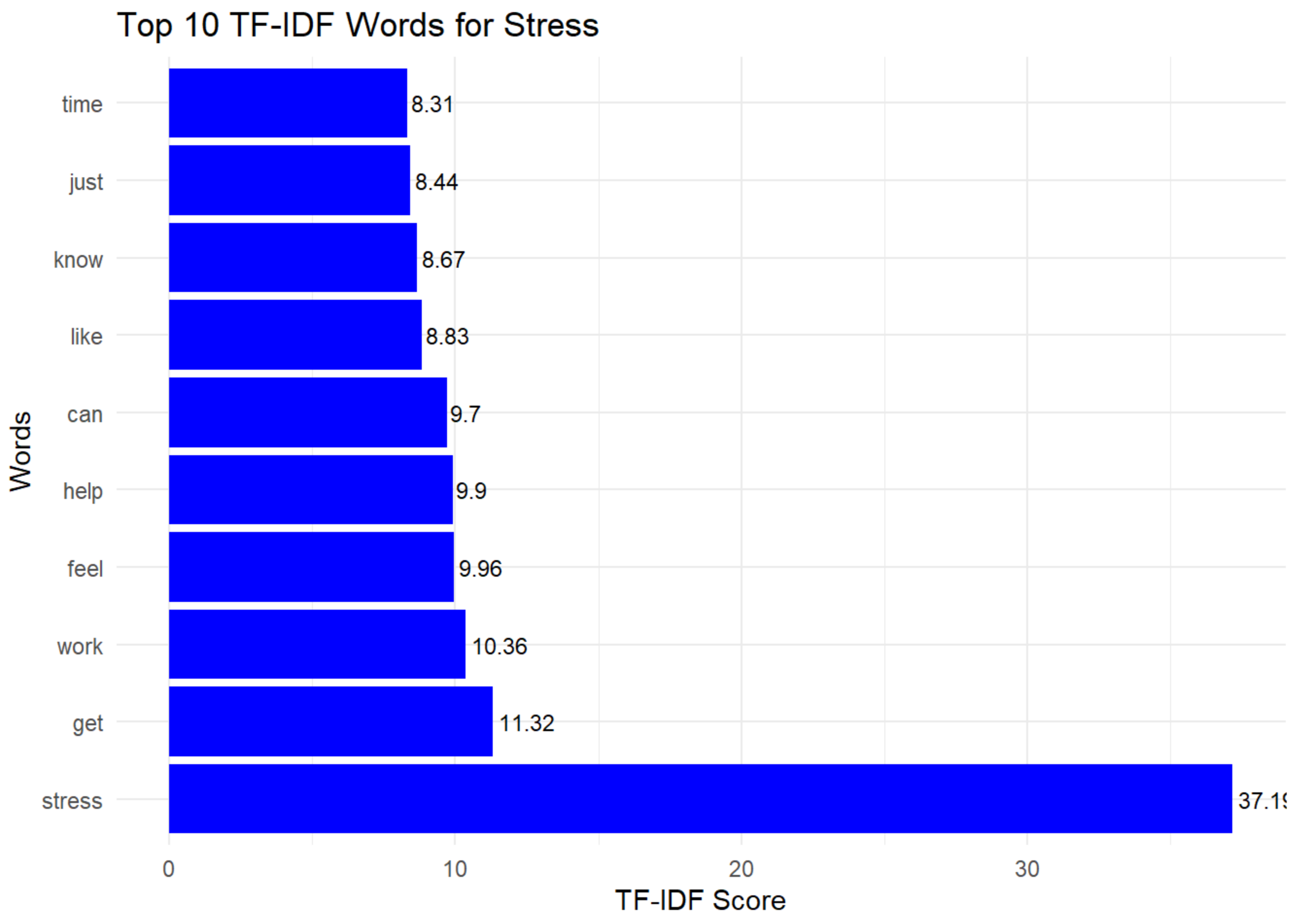

Top 10 TF-IDF Words for Anxiety
Words
get 10.63
sleep 11.08
pain 11.48
heart 11.86
feel 13.76
anxious 14.53
anxiety 22.77
nervous 30.99
worry 34.54
restless 54.62
0
20
40
TF-IDF Score

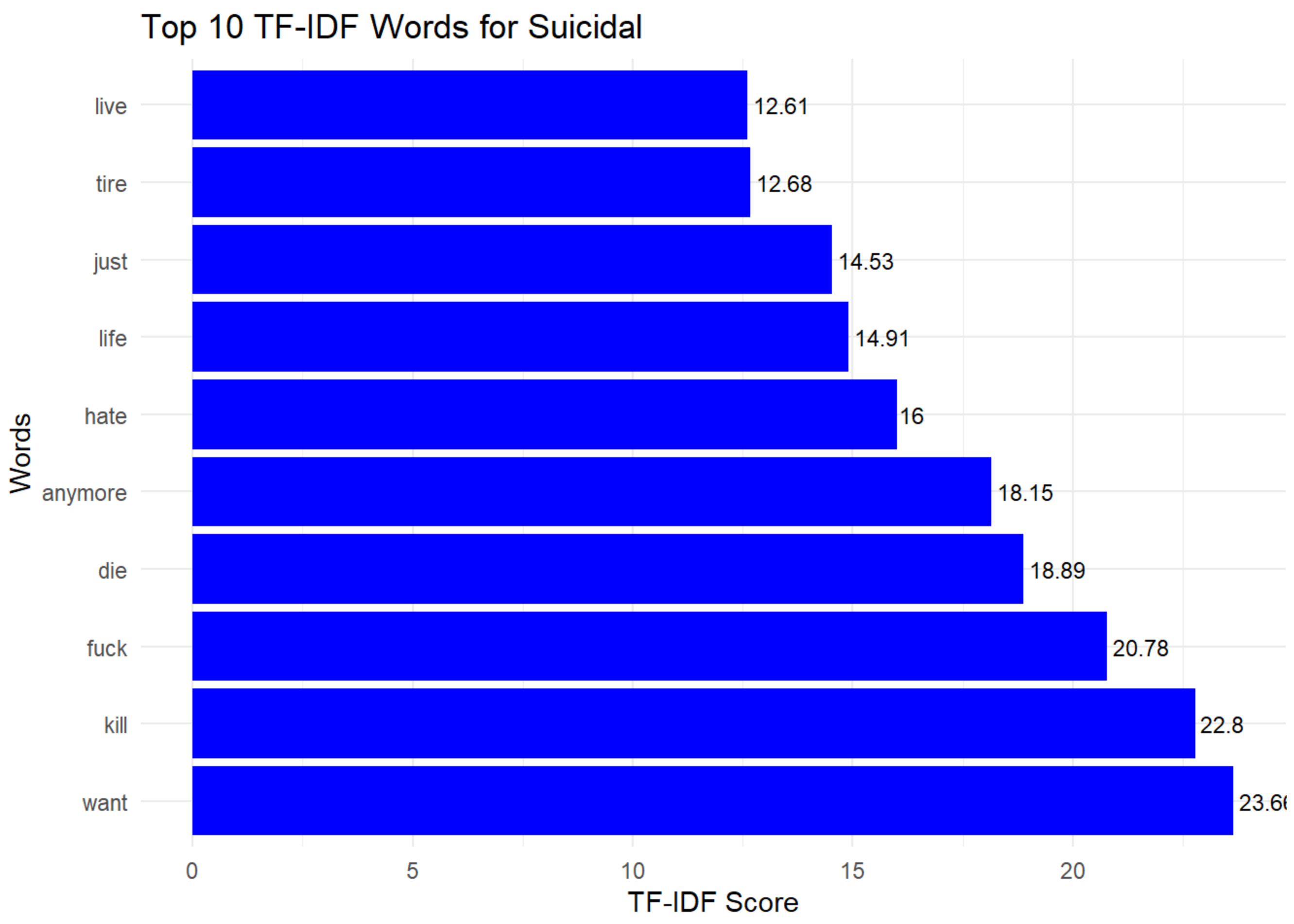

Top 10 TF-IDF Words for Bipolar
Words
lithium 9.85
just 9.93
meds 10.26
medication 10.68
feel 10.74
sleep 11.12
take 11.22
manic 12.96
episode 16.11
bipolar 21.7
0
5
10
15
20
TF-IDF Score


Top 10 TF-IDF Words for Depression
Words
good
know
get
like
don
life
want
just
feel
depression
9.42
9.56
10.24
11.38
12.25
13.38
14.5
15.02
16.37
20.4
0
5
10
15
20
TF-IDF Score


Top 10 TF-IDF Words for Normal
Words
bun
quot
nice
dreamies
miss
cool
yes
good
tomorrow
morning
11.95
12.95
13
13.82
15.01
16.69
16.99
17.53
18.21
21.0
0
5
10
15
20
TF-IDF Score


Top 10 TF-IDF Words for Personality disorder
Words
think 8.9
hypochondrium 8.95
make 9.13
social 9.73
disorder 9.79
feel 11.33
like 12.31
view 13.46
people 16.46
avpd 28.9
0
10
20
30
TF-IDF Score


Top 10 TF-IDF Words for Stress
Words
time 8.31
just 8.44
know 8.67
like 8.83
can 9.7
help 9.9
feel 9.96
work 10.36
get 11.32
stress 37.1
0
10
20
30
TF-IDF Score



2.3

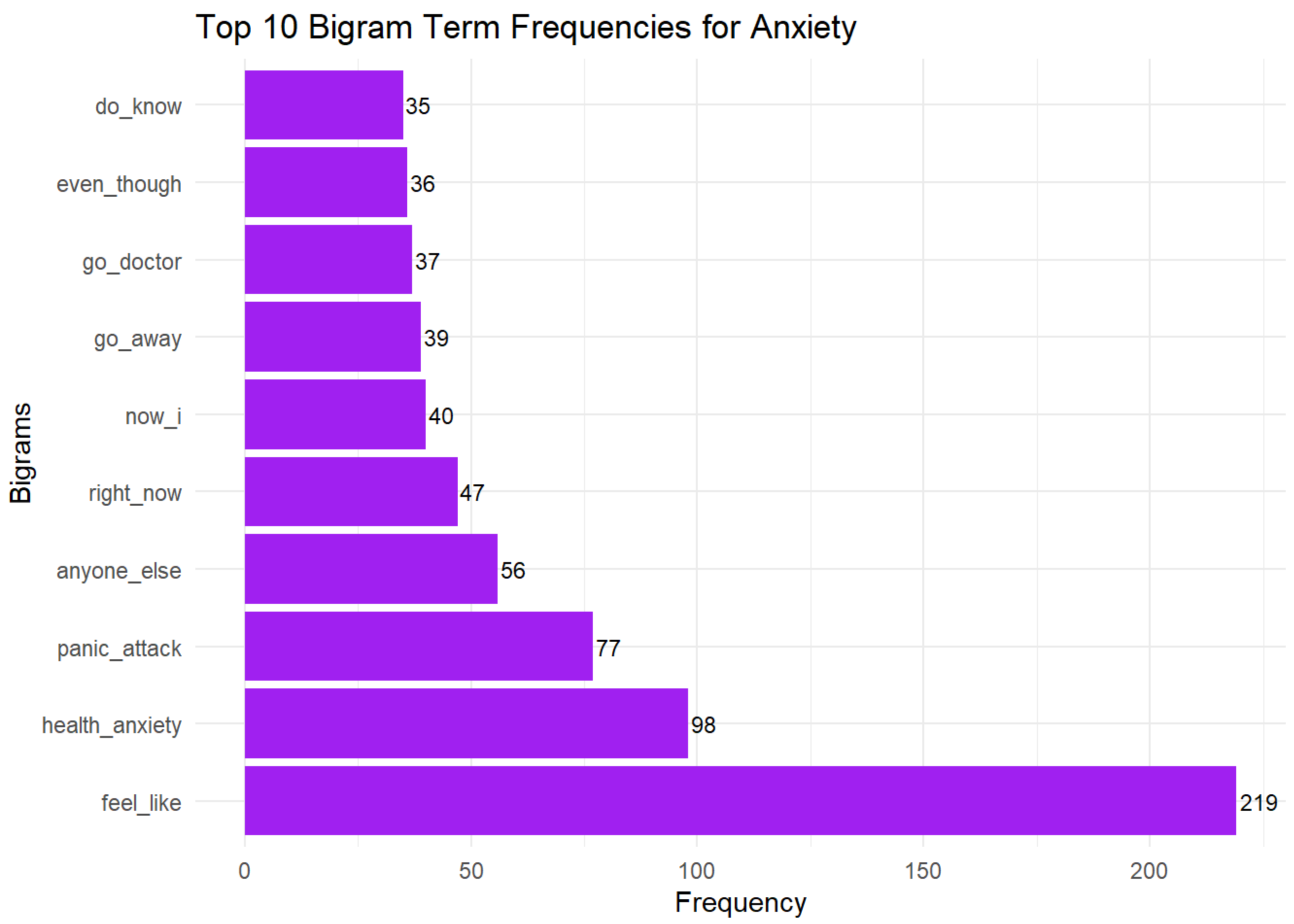
TF of dataset - Bigram

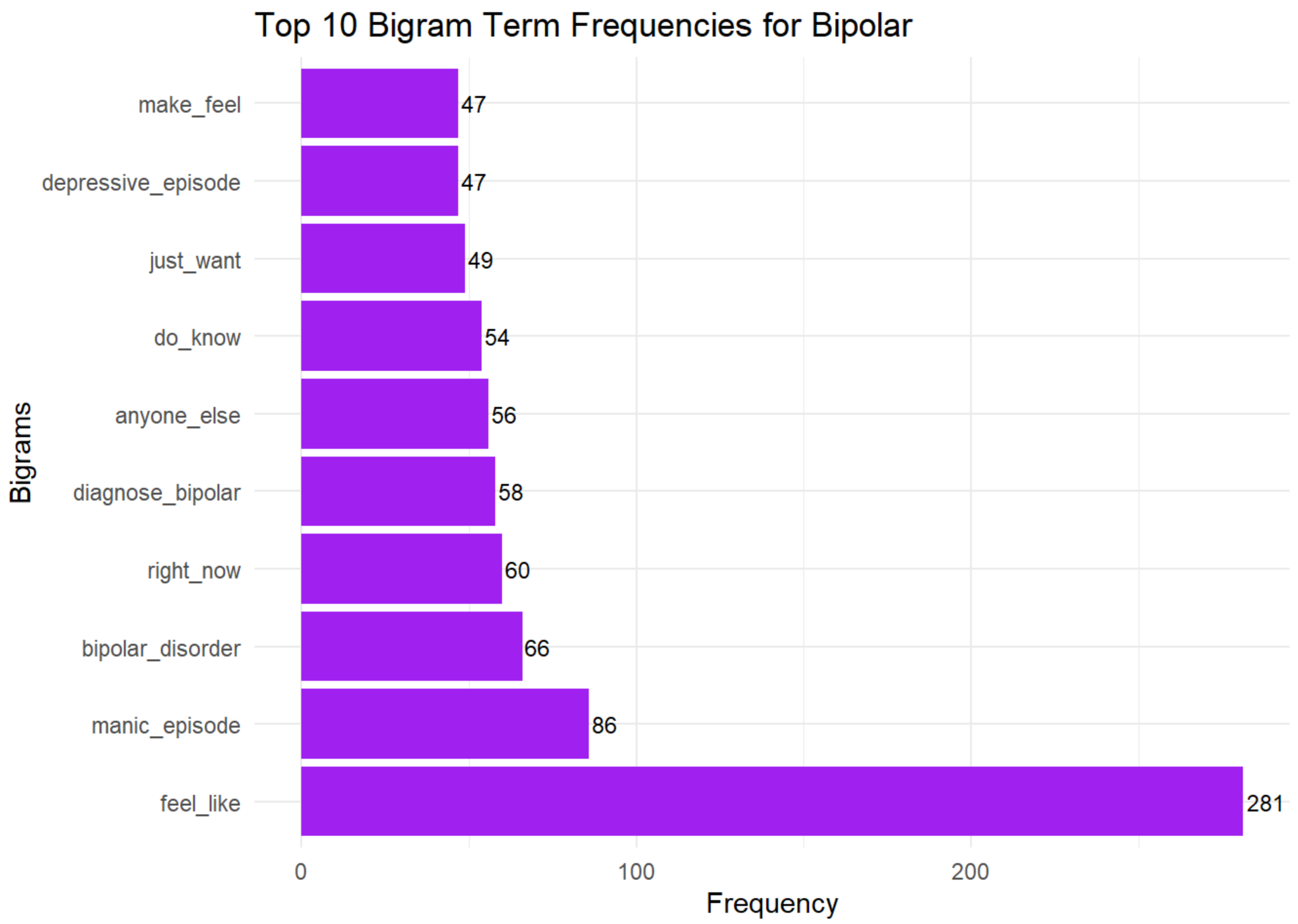
```r
# Load necessary libraries
library(dplyr)
library(quanteda)

# Tokenize the statements into bigrams
tokens <- quanteda::tokens(cleaned_statements$statement, remove_punct = TRUE)
tokens <- tokens_ngrams(tokens, n = 2)

# Create a document-feature matrix (DFM) from the bigrams
dfm <- dfm(tokens)

# Add the target variable (status) to the DFM as a docvar (document variable)
docvars(dfm, "status") <- cleaned_statements$status

# Function to get top bigrams based on term frequencies (TF) for each status
get_top_tf_bigrams <- function(dfm, status, num_top_bigrams = 10) {
  # Filter the DFM for the given status
  status_dfm <- dfm[docvars(dfm, "status") == status, ]

  # Calculate the sum of raw term frequencies for each bigram
  bigram_freq <- colSums(status_dfm)

  # Sort the bigrams by their raw term frequencies in decreasing order
  sorted_bigram_freq <- sort(bigram_freq, decreasing = TRUE)

  # Get the top N bigrams
  top_bigrams <- head(sorted_bigram_freq, num_top_bigrams)

  return(top_bigrams)
}

# List of unique statuses
statuses <- unique(cleaned_statements$status)

# Get top 10 bigrams for each status
top_tf_bigrams_by_status <- lapply(statuses, function(status) {
  get_top_tf_bigrams(dfm, status, num_top_bigrams = 10)
})

# Assign status names to the results
names(top_tf_bigrams_by_status) <- statuses

# Display the top 10 bigrams for each status
top_tf_bigrams_by_status
```

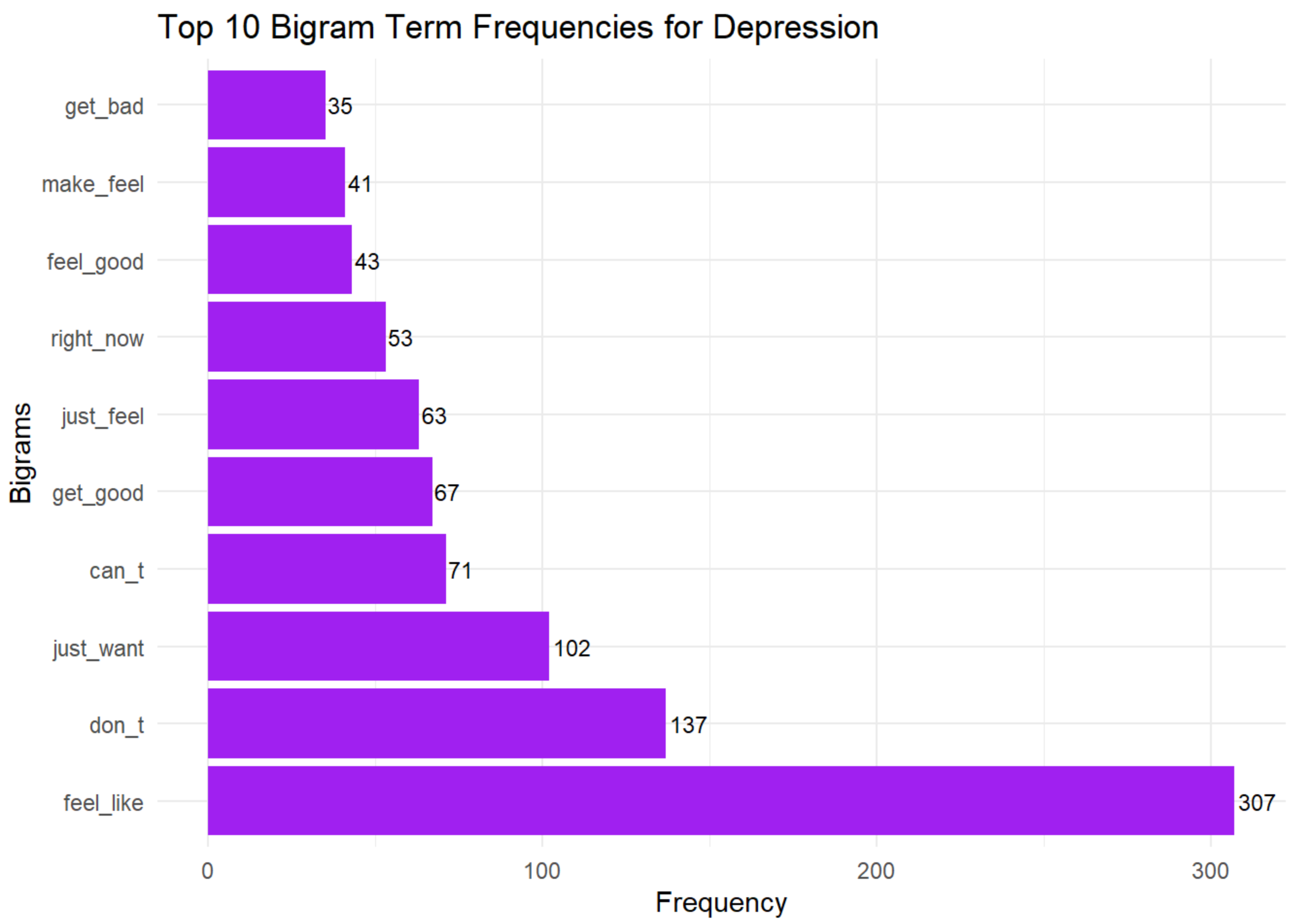
```
## $Anxiety
##       feel_like health_anxiety   panic_attack    anyone_else      right_now
##             219             98             77             56             47
##           now_i        go_away      go_doctor    even_though        do_know
##              40             39             37             36             35
##
## $Bipolar
##           feel_like      manic_episode   bipolar_disorder          right_now
##                 281                 86                 66                 60
##    diagnose_bipolar        anyone_else            do_know          just_want
##                  58                 56                 54                 49
##           make_feel depressive_episode
##                  47                 47
##
## $Depression
## feel_like     don_t just_want     can_t  get_good just_feel right_now feel_good
##       307       137       102        71        67        63        53        43
## make_feel   get_bad
##        41        35
##
## $Normal
##       don_t   feel_like even_though       can_t         â_â high_school
##          11          10           7           7           7           5
##     t_think      let_us   right_now   look_like
##           5           4           4           4
##
## $`Personality disorder`
##       feel_like    anyone_else      make_feel         like_i      just_want
##             306             59             59             56             54
##         do_know social_anxiety      just_feel        like_im        i_just
##              48             43             39             39             39
##
## $Stress
##    feel_like    right_now      do_know     just_get       like_i panic_attack
##          167           43           36           31           31           30
##  even_though        i_try    just_want    make_feel
##           30           27           27           23
##
## $Suicidal
## take_anymore anymore_take    feel_like    just_want     want_die    right_now
##         1273         1259          208           94           91           55
##     get_good  even_though    just_feel    make_feel
##           48           31           31           31
```

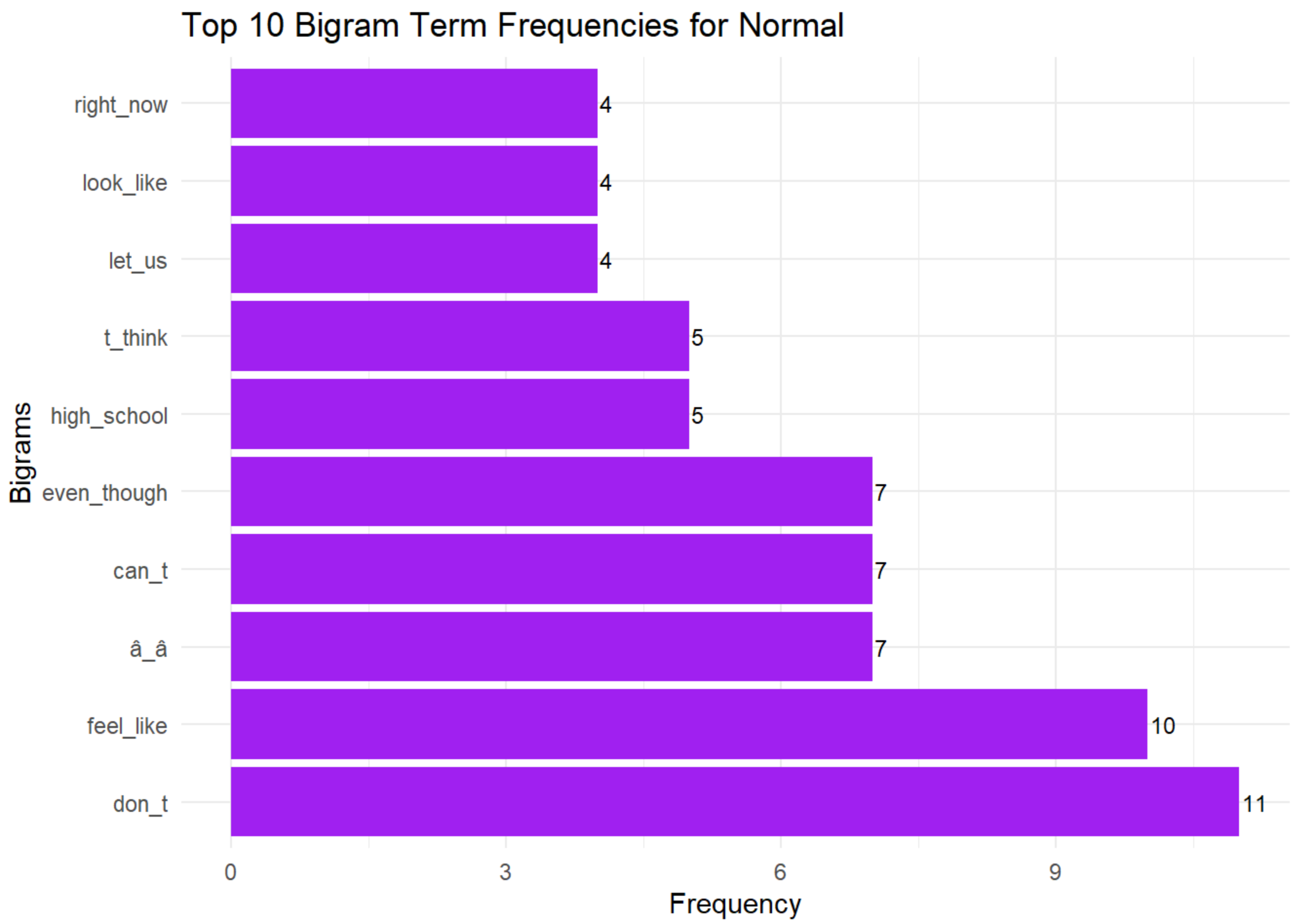
Bigram - BARCHART

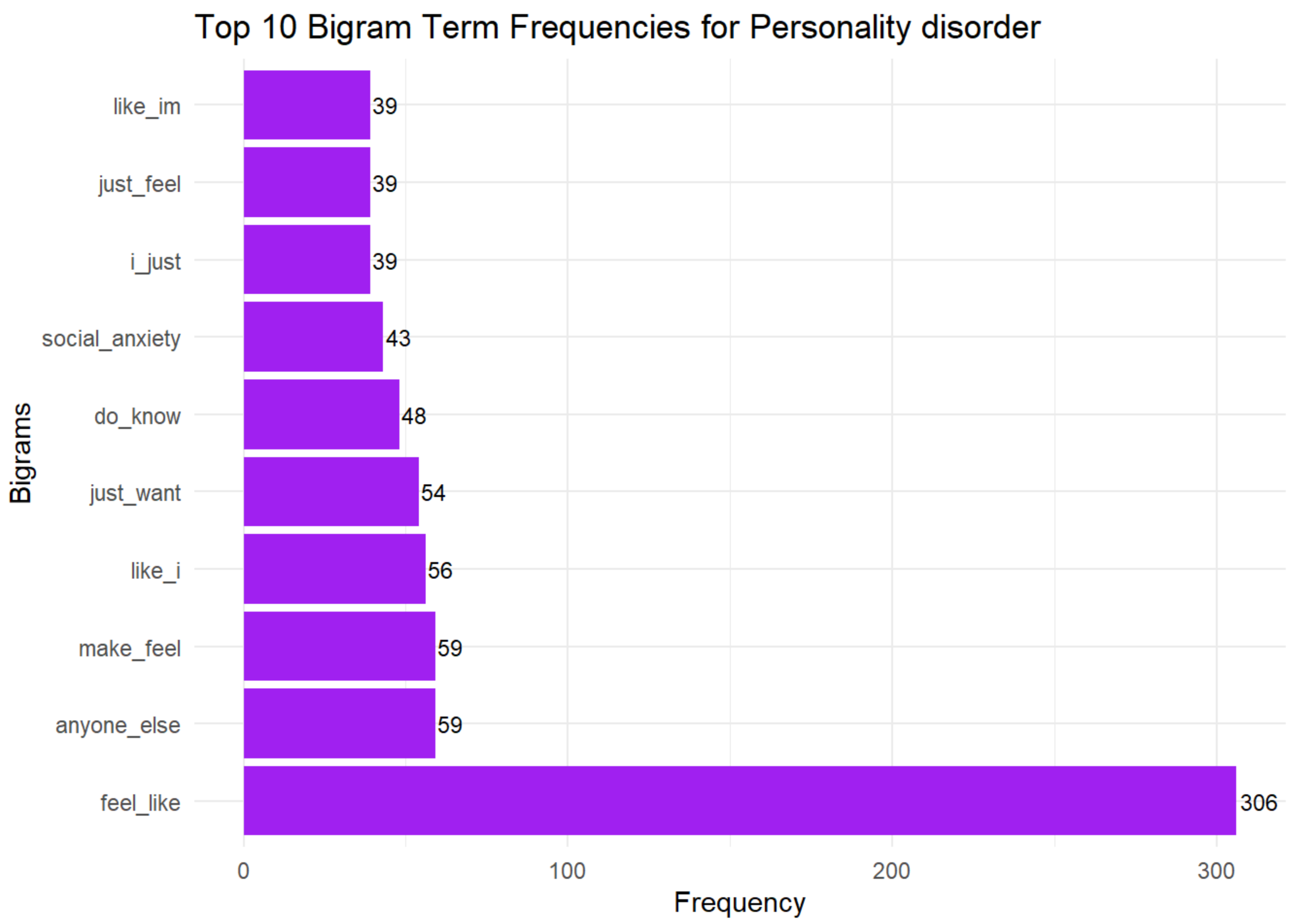
```r
library(ggplot2)
library(dplyr)

# Convert the list to a combined data frame for plotting bigram term frequencies
plot_bigram_data <- lapply(names(top_tf_bigrams_by_status), function(status) {
  data.frame(Status = status,
             Bigram = names(top_tf_bigrams_by_status[[status]]),
             Frequency = unname(top_tf_bigrams_by_status[[status]]))
}) %>%
  bind_rows()

# Create individual plots for each status with frequencies next to bars, and set bar color to
purple
status_bigram_plots <- plot_bigram_data %>%
  split(.$Status) %>%
  map(~ ggplot(.x, aes(x = reorder(Bigram, -Frequency), y = Frequency)) +
        geom_bar(stat = "identity", fill = "purple") +  # Set bars color to purple
        geom_text(aes(label = Frequency), vjust = 0.5, hjust = -0.1, size = 3) +  # Add frequ
encies next to the bars
        coord_flip() +
        labs(title = paste("Top 10 Bigram Term Frequencies for", .x$Status[1]),
             x = "Bigrams", y = "Frequency") +
        theme_minimal())

# Loop through each status and print the corresponding plot
for(status in names(status_bigram_plots)) {
  print(status_bigram_plots[[status]])
}
```

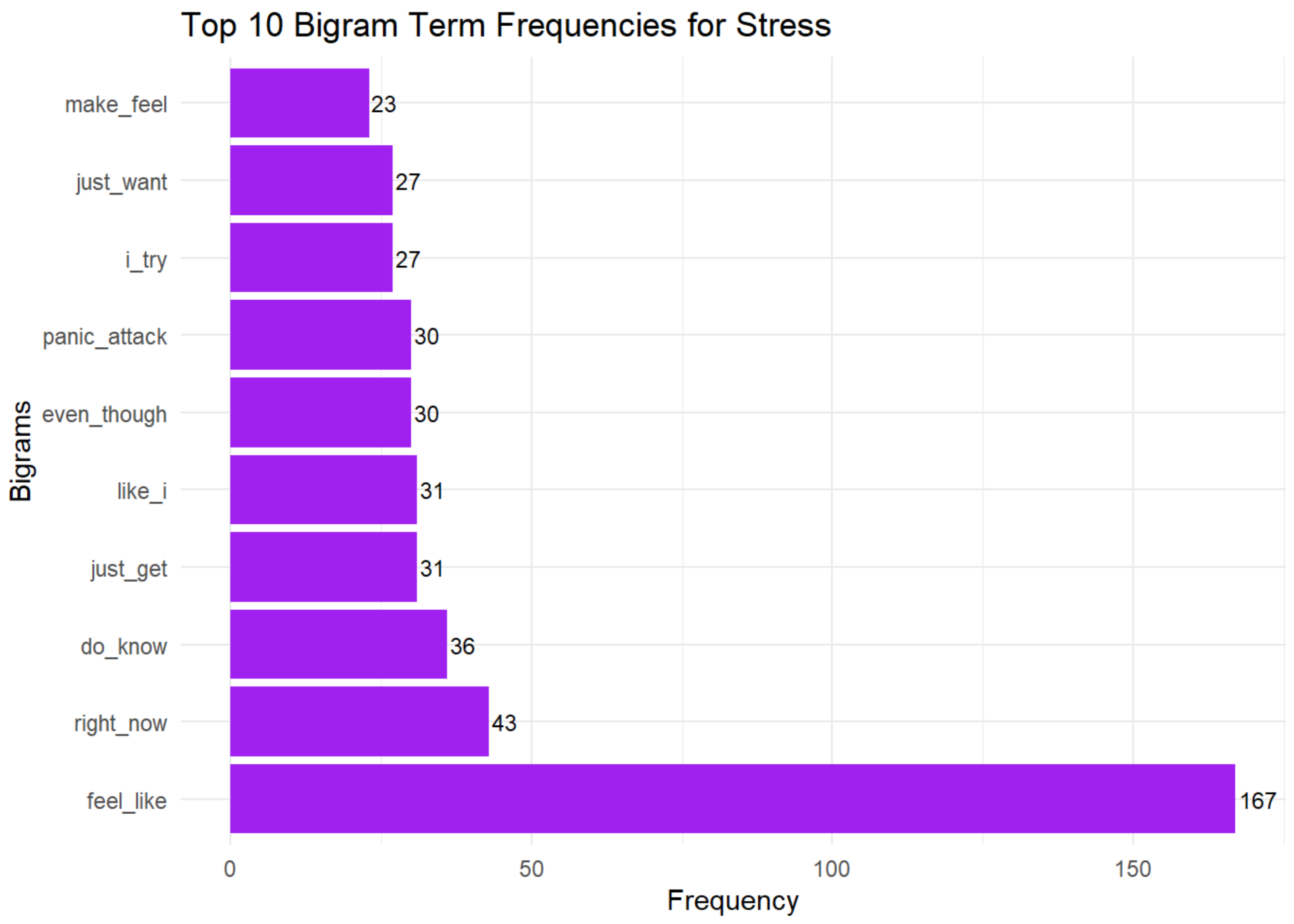

Top 10 Bigram Term Frequencies for Anxiety
Bigrams
do_know 35
even_though 36
go_doctor 37
go_away 39
now_i 40
right_now 47
anyone_else 56
panic_attack 77
health_anxiety 98
feel_like 219
0
50
100
150
200
Frequency

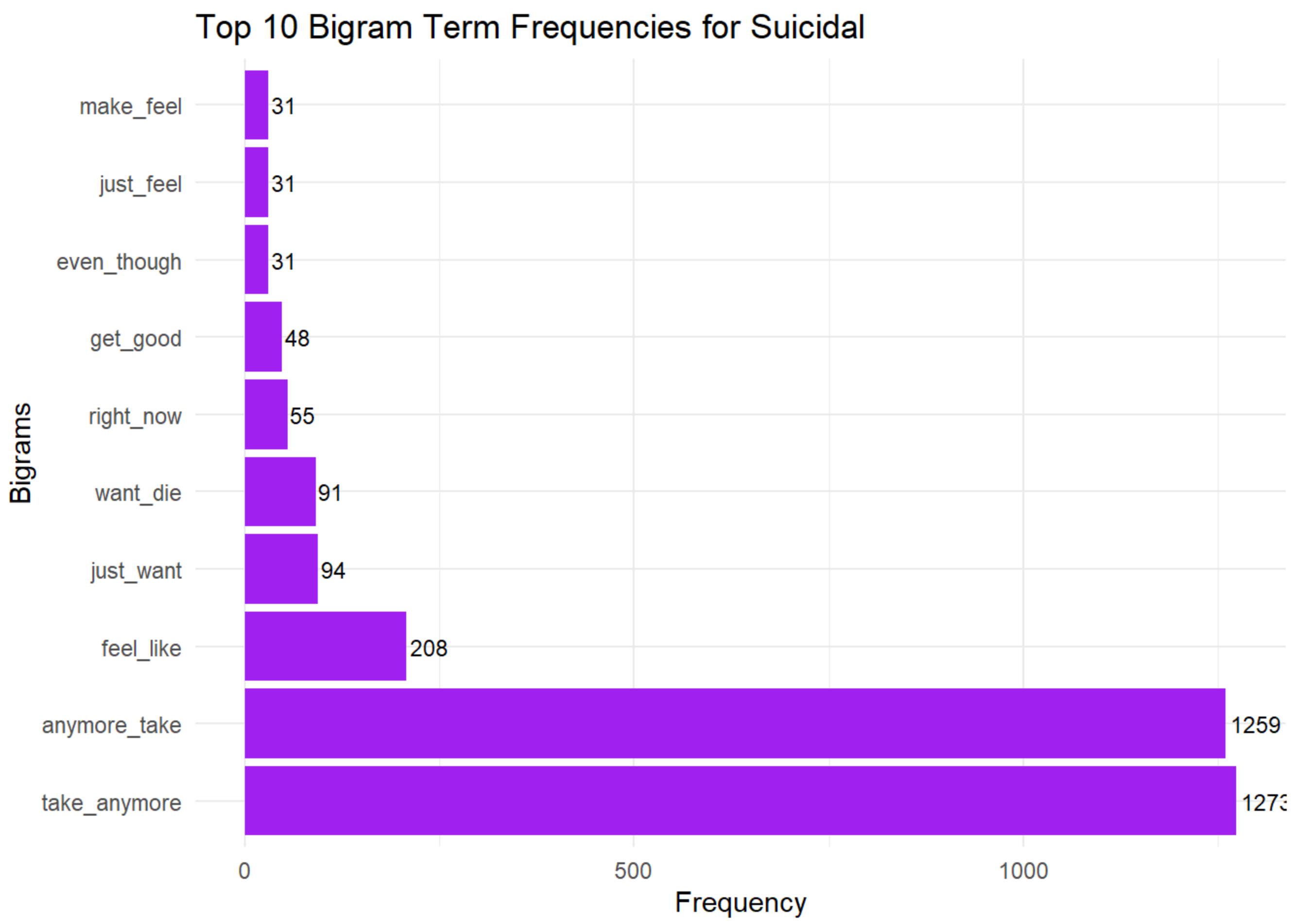

Top 10 Bigram Term Frequencies for Bipolar
Bigrams
make_feel 47
depressive_episode 47
just_want 49
do_know 54
anyone_else 56
diagnose_bipolar 58
right_now 60
bipolar_disorder 66
manic_episode 86
feel_like 281
0
100
200
Frequency


Top 10 Bigram Term Frequencies for Depression
Bigrams
get_bad 35
make_feel 41
feel_good 43
right_now 53
just_feel 63
get_good 67
can_t 71
just_want 102
don_t 137
feel_like 307
0
100
200
300
Frequency


Top 10 Bigram Term Frequencies for Normal
Bigrams
right_now 4
look_like 4
let_us 4
t_think 5
high_school 5
even_though 7
can_t 7
â_â 7
feel_like 10
don_t 11
0
3
6
9
Frequency


Top 10 Bigram Term Frequencies for Personality disorder
Bigrams
like_im 39
just_feel 39
i_just 39
social_anxiety 43
do_know 48
just_want 54
like_i 56
make_feel 59
anyone_else 59
feel_like 306
0
100
200
300
Frequency


Top 10 Bigram Term Frequencies for Stress
Bigrams
make_feel 23
just_want 27
i_try 27
panic_attack 30
even_though 30
like_i 31
just_get 31
do_know 36
right_now 43
feel_like 167
0
50
100
150
Frequency



#

3.1 ## Function to search for a word in a particular status and return the count of occurrences

```r
search_word_in_status_count <- function(word, status_value, cleaned_statements) {

  # Convert the word and status to lowercase for case-insensitive search
  word <- tolower(word)
  status_value <- tolower(status_value)

  # Filter the dataset based on the status
  filtered_data <- cleaned_statements %>% filter(tolower(status) == status_value)

  # Search for the word in the statement (case-insensitive)
  matched_data <- filtered_data %>%
    filter(grepl(word, tolower(statement)))  # 'tolower' ensures case-insensitive matching

  # Return the count of rows where the word appears
  return(nrow(matched_data))
}

# Example usage
word_to_search <- "sad"
status_to_filter <- "Depression"

# Get the count of occurrences of the word "feel" in the "Depression" status category
word_count <- search_word_in_status_count(word_to_search, status_to_filter, cleaned_statement
s)

# View the count
word_count
```

```
## [1] 77
```

3.2

Function to search for a word in a particular status

```r
search_word_in_status <- function(word, status_value, cleaned_statements) {

  # Convert the word and status to lowercase for case-insensitive search
  word <- tolower(word)
  status_value <- tolower(status_value)

  # Filter the dataset based on the status
  filtered_data <- cleaned_statements %>% filter(tolower(status) == status_value)

  # Search for the word in the statement (case-insensitive)
  matched_data <- filtered_data %>%
    filter(grepl(word, tolower(statement)))  # 'tolower' ensures case-insensitive matching

  # Return the rows where the word appears
  return(matched_data)
}

# Example usage
word_to_search <- "change"
status_to_filter <- "Bipolar"

# Search for the word "stress" in the "Stress" status category
results <- search_word_in_status(word_to_search, status_to_filter, cleaned_statements)

# View the results
head(results)
```

```
##
statement
## 1
brutal recent med change zoloft make mix totally insane get lithium poison akathisia abilify
suicidal trileptal suicidal latuda depakote work haldol right much wellbutrin make unstable a
lmost nonstop joyride almost hospitalize time I never big deal legitimately almost kill occas
ion dabble self harm rough year professionally get along old teach team turn kid principal ge
t involve side ugly finish first year teach now 2 year go much good end first year miserable
friend see idea facebook put note jar week one good thing happen week note end year look grea
t thing happen love idea heres hope everyone good can bad sure
## 2 insight two year custom mood track much datum background statistic need understand inter
pretation I basic explanation can make accessible risk misrepresent want get pedantic explain
good comment go ahead explanation footnote mood past two year improve slightly overall stay a
verage mood something day day feel good bad spend day last two year mood low day spend day se
verely depress spend day strong suicidal feeling day actively pursue suicide spend day mood r
emainder either day record spend day euphoric mood spend day top world euphoric moodgt mood s
tandard deviation percentage point percentage point I get stable full time job guess long tri
bulation terminal cancer life happy day last two year low standard deviation suggest much emp
hatic thank redditor behind one custom manic rate scale suggest spend day kind manic day kind
manic get lower stress level increase mood orgasm reduce perceive stress percentage point p v
alue mean stress level difference per orgasm orgasm also raise mood percentage point typical
increase per orgasm point increase stress level increase anxiety average percentage point mea
n anxiety level difference per point effect alcohol mood unclear mood tracker strongly sugges
t plt drink feel good direction causality likely drink feel good rather make feel good datum
nature can tell affect duration alcohol clear sufficient evidence strongly support hypothesis
alcohol make particular depress sleep play highly significant role minimize stress anxiety p
value respectively meaningful correlation coefficient wee improve mood percentage point per l
evel rtrees highness scale p value mean mood improvement per point highness total high increa
se mood high likelihood experience psychosis moderate high may increase mood word wee definit
ely improve mood overall level factor still important worth note percentage increase mood dif
ferent base baseline mood I suicidal depression typical mood around suicidal ever scale point
increase mood actually increase mood actually use wee last time suicidal get feeling help wee
risky make us psychotic can make us depress use caution wee increase sleep average night slee
p last hour increase hour per point rtrees highness scale strong high increase hour sleep hal
f hour probably weak alarm wee impair cognition p value lt wee cause psychosis p value little
typically psychosis severe something I continue help call vice drink bite thank bender around
july learn cope boredom good drink per day vs drink per day I two drink average drink per day
intend drink I give live break time thing teach effect substance use complicate hard measure
deliberately minimize analysis alcohol test demonstrate statistical significance anecdotal ob
servation can tell I learn increase effect stress body can exacerbate either depression mania
help feel connect friend may benefit one way hurt another lot keep track probably mood tracke
r can keep track column variable sleep meds make world difference mood tracker informative da
tum want see substance affect gotta track measurably support network friend can make world di
fference learn observation verify still meds definitely help pretty dumbed statistic p value
basically explain likelihood occur due chance mean mean randomly occur know quite right simpl
ification r square good algorithm explain datum high r square mean regression high explanator
y power perfect fit fit beta strength correlation mood measure point scale one unit sleep cha
nge mood point sleep beta standard deviation descriptor amount variation datum set everything
close average get low standard deviation everything disparate high sd edit think percent perc
entage point confuse somewhere wrong fix twice oops edit want share feel wear set new one guy
uglitterbeast remind post dropbox save file ithttpswwwredditcomrbipolarredditcommentsmmchnmak
```

```
ingamoodtrackerhowididitwhyyoumight update one link description make likely output possible m
ania seem fail correct error get way accurate alcoholrelated depression result change date ed
it big thing may change role sex drive manic rate people keep baseline libido plenty people l
ibido strong indicator mania
## 3
doctor want medication change long history psychiatric illness medication I jump ahead half y
ear ago get extreme ppd hospitalize twice small bite klonopin first stop work everything try
make crazy one doctor put celexa initial anxiety change life see regular psych try different
thing get manic hesitant huge drunken manic episode hospitalize prescribe abilify help lot al
most year ago ampxb okay take sleep pill night stomach meds ativan propanolol need birth cont
rol pmdd recently psych take sleep pill end propanolol get severe anxiety panic attack birth
control make anxiety unbearable one week every month med doc switch something else increase s
tomach meds acid refluxgastritis start anti inflammatory diet tire mess exercise good happy p
sych think hypomania although still depress lot want drop dosage mg remain mg abilify terribl
y med sensitive many thing cause crazy reaction long time big cut basically one month new bir
th control also sensitive sleep meds little anti anxiety meds stomach problem now decrease me
d problem I two day already go crazy can expect ampxb long story short lot med change abilify
mg ativan twice day loe estrin fe drop mg citalopram feel mess
## 4
proud responsible get stuff do I need get oil change past mile headlight less month today fin
ally get oil change headlight replace even change car air filter think I even get tab renew
## 5
try understand exgirlfriend good hey know type question allow subreddit figure good way under
stand someone live bipolar disorder ask people live bipolar disorder meet exgirlfriend young
become good friend save life time try commit suicide always think really deep connection howe
ver start date year know date year relationship beautiful also exhaust time time constant sui
cide thought slight issue also shitty behaviour show time time include insult manipulate blac
kmail also depression social anxiety schizophrenia think borderline hard distinguish behave l
ike however year date good friend friend come back life suddenly lose interest personally thi
nk borderline since always need favourite person life can time really surprise can really exp
lain thing really change lot quickly always use play videogame watch movie go walk park wood
good friend come back life change completely suddenly go party drink alcohol every day also s
eem lose every mental illness sign depression anymore sign social anxiety anymore suddenly li
ke one girl clichee american student movie drink alcohol party just enjoy live without fear r
epercussion break sense guilt anymore cancel date go another guy break whatsapp minute late c
ome another guy pick stuff talk mother mother cry tell daughter just say go way now whole tim
e parent also worry know happen apparently also home point week anymore also use social mediu
m platform anymore play video game anymore basically whole life first think borderline ask pe
ople borderline say even never anything like last hope accept just good human ask guy explain
bipolar disorder like manic episode
## 6
cant sleep sleep much little two week now im sure feel morning inevitably get less couple hou
r rest think time change blow do want work tomorrow uncomfortable bed moment body just hurt b
ody rest pathetic work get computer right next work home do even want far active night always
just do want morning work job get bad bad good back original post just cant sleep pretty anno
y
##    status
## 1 Bipolar
## 2 Bipolar
## 3 Bipolar
## 4 Bipolar
## 5 Bipolar
## 6 Bipolar
```

3.3

Cosine Similarity

```
# Load the textTinyR package
library(textTinyR)
```

```
## Warning: package 'textTinyR' was built under R version 4.4.2
```

```
## Loading required package: Matrix
```

```
## Warning: package 'Matrix' was built under R version 4.4.2
```

```
# Define two example text inputs
text1 <- "I keep waking up with a racing heart and feel like my anxiety is out of control."

text2 <- "Life feels empty, and I can't find joy or hope in anything anymore."

# Use the COS_TEXT function to calculate cosine similarity
cosine_sim <- COS_TEXT(text_vector1 = text1,
                       text_vector2 = text2,
                       threads = 1)

# Print the cosine similarity result
print(cosine_sim)
```

```
## [1] 0.1345346
```