

# Group\_project

Chan Hou Long, Guyver

11/18/2021

## Summary

## Introduction

## Loading and Exploring Data

### Libraries required

### Data size and structure

```
df.train <- read_csv("~/Documents/HKU/SOWK3136/project/dreaddit-train.csv")
df.test <- read_csv("~/Documents/HKU/SOWK3136/project/dreaddit-test.csv")

df.full <- bind_rows(df.train, df.test)
```

```
dim(df.full)
```

```
## [1] 3553 116
```

```
str(df.full[,c(1:10)])
```

```
## tibble [3,553 x 10] (S3: tbl_df/tbl/data.frame)
## $ subreddit      : chr [1:3553] "ptsd" "assistance" "ptsd" "relationships" ...
## $ post_id        : chr [1:3553] "8601tu" "8lbrx9" "9ch1zh" "7rorpp" ...
## $ sentence_range  : chr [1:3553] "(15, 20)" "(0, 5)" "(15, 20)" "[5, 10]" ...
## $ text           : chr [1:3553] "He said he had not felt that way before, suggeted I go rest and s
## $ id             : num [1:3553] 33181 2606 38816 239 1421 ...
## $ label          : num [1:3553] 1 0 1 1 1 1 0 1 1 1 ...
## $ confidence      : num [1:3553] 0.8 1 0.8 0.6 0.8 1 0.8 0.8 0.6 1 ...
## $ social_timestamp: num [1:3553] 1.52e+09 1.53e+09 1.54e+09 1.52e+09 1.54e+09 ...
## $ social_karma    : num [1:3553] 5 4 2 0 24 2 6 1 134 20 ...
## $ syntax_ari      : num [1:3553] 1.81 9.43 7.77 2.67 7.55 ...
```

## Missing data, label encoding, and factorizing variables

```
sum(is.na(df.full))
```

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

```
apply(apply(df.full,2,is.na),2,sum) ; nrow(df.full)
```

##	subreddit	post_id	sentence_range
##	0	0	0
##	text	id	label
##	0	0	0
##	confidence	social_timestamp	social_karma
##	0	0	0
##	syntax_ari	lex_liwc_WC	lex_liwc_Analytic
##	0	0	0
##	lex_liwc_Clout	lex_liwc_Authentic	lex_liwc_Tone
##	0	0	0
##	lex_liwc_WPS	lex_liwc_Sixltr	lex_liwc_Dic
##	0	0	0
##	lex_liwc_function	lex_liwc_pronoun	lex_liwc_ppron
##	0	0	0
##	lex_liwc_i	lex_liwc_we	lex_liwc_you
##	0	0	0
##	lex_liwc_shehe	lex_liwc_they	lex_liwc_ipron
##	0	0	0
##	lex_liwc_article	lex_liwc_prep	lex_liwc_auxverb
##	0	0	0
##	lex_liwc_adverb	lex_liwc_conj	lex_liwc_negate
##	0	0	0
##	lex_liwc_verb	lex_liwc_adj	lex_liwc_compare
##	0	0	0
##	lex_liwc_interrog	lex_liwc_number	lex_liwc_quant
##	0	0	0
##	lex_liwc_affect	lex_liwc_posemo	lex_liwc_negemo
##	0	0	0
##	lex_liwc_anx	lex_liwc_anger	lex_liwc_sad
##	0	0	0
##	lex_liwc_social	lex_liwc_family	lex_liwc_friend
##	0	0	0
##	lex_liwc_female	lex_liwc_male	lex_liwc_cogproc
##	0	0	0
##	lex_liwc_insight	lex_liwc_cause	lex_liwc_discrep
##	0	0	0
##	lex_liwc_tentat	lex_liwc_certain	lex_liwc_differ
##	0	0	0
##	lex_liwc_percept	lex_liwc_see	lex_liwc_hear
##	0	0	0
##	lex_liwc_feel	lex_liwc_bio	lex_liwc_body
##	0	0	0
##	lex_liwc_health	lex_liwc_sexual	lex_liwc_ingest
##	0	0	0
##	lex_liwc_drives	lex_liwc_affiliation	lex_liwc_achieve
##	0	0	0

```
##          lex_liwc_power          lex_liwc_reward          lex_liwc_risk
##                0                0                0
##    lex_liwc_focuspast    lex_liwc_focuspresent    lex_liwc_focusfuture
##                0                0                0
##    lex_liwc_relattiv    lex_liwc_motion          lex_liwc_space
##                0                0                0
##    lex_liwc_time        lex_liwc_work          lex_liwc_leisure
##                0                0                0
##    lex_liwc_home        lex_liwc_money          lex_liwc_relig
##                0                0                0
##    lex_liwc_death        lex_liwc_informal        lex_liwc_swear
##                0                0                0
##    lex_liwc_netspeak    lex_liwc_assent          lex_liwc_nonflu
##                0                0                0
##    lex_liwc_filler        lex_liwc_AllPunc        lex_liwc_Period
##                0                0                0
##    lex_liwc_Comma        lex_liwc_Colon          lex_liwc_SemiC
##                0                0                0
##    lex_liwc_QMark        lex_liwc_Exclam        lex_liwc_Dash
##                0                0                0
##    lex_liwc_Quote        lex_liwc_Apostro        lex_liwc_Parenth
##                0                0                0
##    lex_liwc_OtherP    lex_dal_max_pleasantness    lex_dal_max_activation
##                0                0                0
##    lex_dal_max_imagery    lex_dal_min_pleasantness    lex_dal_min_activation
##                0                0                0
##    lex_dal_min_imagery    lex_dal_avg_activation    lex_dal_avg_imagery
##                0                0                0
##    lex_dal_avg_pleasantness    social_upvote_ratio    social_num_comments
##                0                0                0
##    syntax_fk_grade          sentiment
##                0                0
```

```
## [1] 3553
```

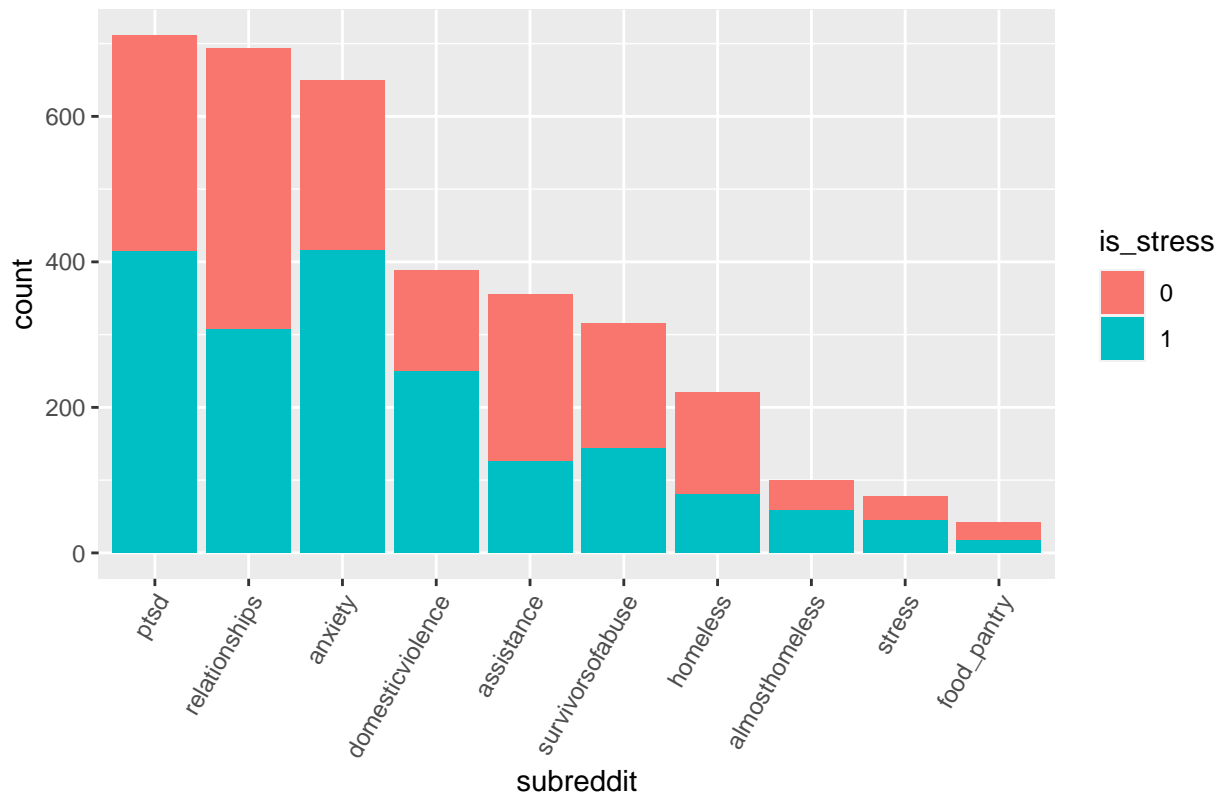
## Exploring some of the most important variables

```
df.full <- df.full[, c("subreddit", "text", "label", "social_karma",
                      "social_upvote_ratio", "social_num_comments")] %>%
  rename(is_stress = label)
```

```
result = summarize(group_by(df.full, subreddit, is_stress),
                   count=n())
result$is_stress = factor(result$is_stress)
```

```
ggplot(result, aes(fill=is_stress, y=count, x=reorder(subreddit, -count))) +
  geom_bar(position="stack", stat="identity") +
  theme(axis.text.x = element_text(angle = 60, hjust = 1)) +
  ggtitle("Number of subreddit with label") +
  xlab("subreddit")
```

Number of subreddit with label



```
summarize(group_by(df.full, subreddit),
           count=n()) %>% arrange(desc(count))
```

```
## # A tibble: 10 x 2
##   subreddit    count
##   <chr>      <int>
## 1 ptsd         711
## 2 relationships 694
## 3 anxiety      650
## 4 domesticviolence 388
## 5 assistance    355
## 6 survivorssofabuse 315
## 7 homeless      220
## 8 almothomeless   99
## 9 stress        78
## 10 food_pantry    43
```

```
summarize(group_by(df.full, is_stress),
           count=n()) %>% arrange(desc(count))
```

```
## # A tibble: 2 x 2
##   is_stress count
##   <dbl> <int>
## 1       1  1857
## 2       0  1696
```

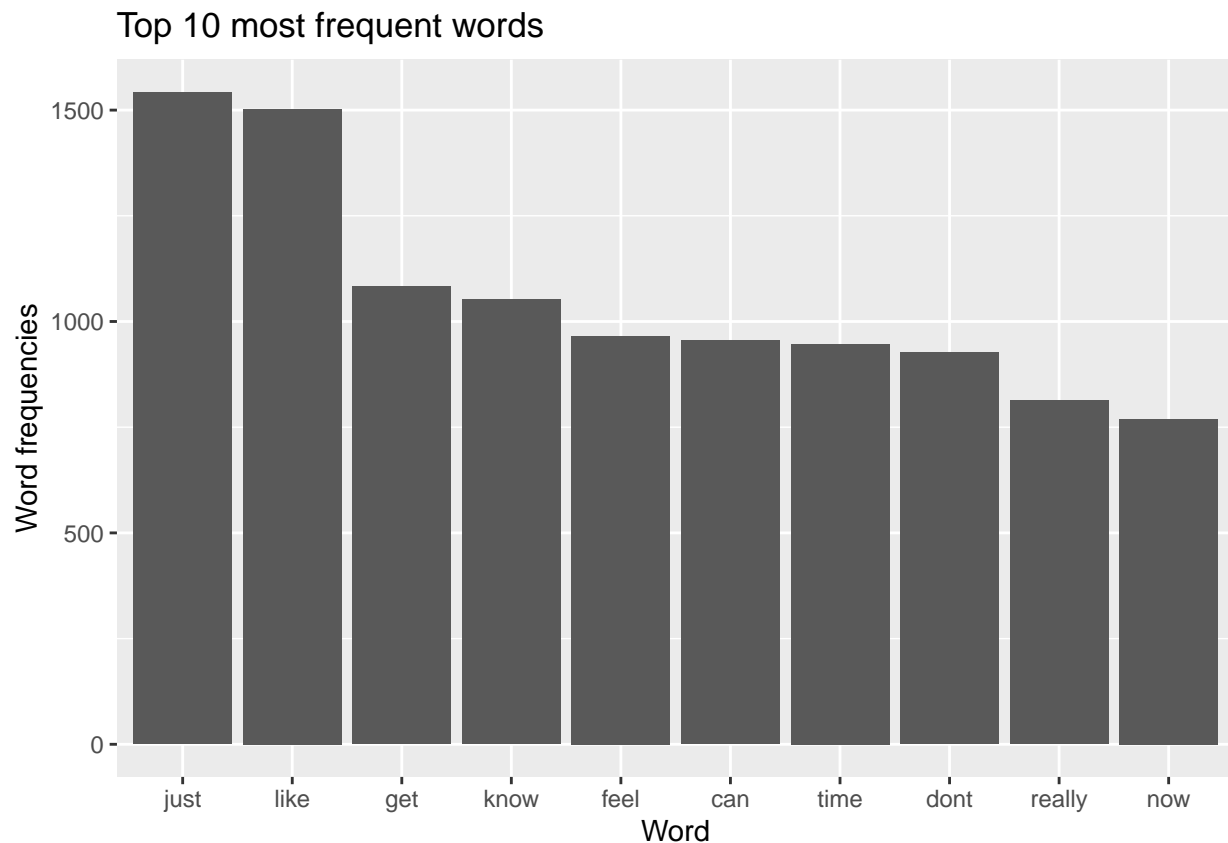
## Sentiment Analysis

```
set.seed(5312)
#Create a vector containing only the text
text <- df.full$text
# Create a corpus
corpus <- Corpus(VectorSource(text))

corpus <- corpus %>%
  tm_map(removeNumbers) %>%
  tm_map(removePunctuation) %>%
  tm_map(stripWhitespace)
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeWords, stopwords("english"))

dtm <- TermDocumentMatrix(corpus)
dtm.M <- as.matrix(dtm)
words <- sort(rowSums(dtm.M),decreasing=TRUE)
dtm_df <- data.frame(word = names(words),freq=words)

ggplot(dtm_df[1:10,], aes(y=freq, x=reorder(word, -freq))) +
  geom_bar(position="stack", stat="identity") +
  ggtitle("Top 10 most frequent words") +
  ylab("Word frequencies") + xlab("Word")
```



```
dtm_df[1:10,]
```

```
##           word freq
## just      just 1542
## like      like 1503
## get       get  1083
## know      know 1052
## feel      feel  966
## can       can  955
## time      time  947
## dont      dont  927
## really    really 813
## now       now  770
```

```
gen_dtm_df <- function(df) {
  #Create a vector containing only the text
  text <- df$text
  # Create a corpus
  corpus <- Corpus(VectorSource(text))

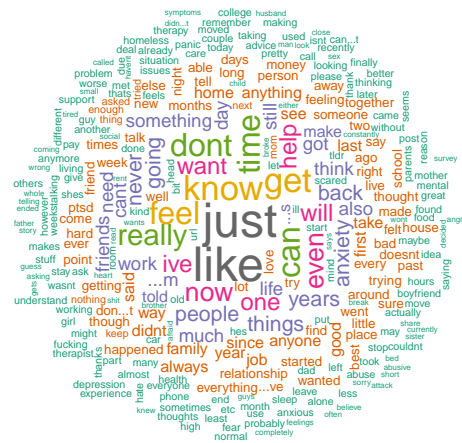
  corpus <- corpus %>%
    tm_map(removeNumbers) %>%
    tm_map(removePunctuation) %>%
    tm_map(stripWhitespace)
  corpus <- tm_map(corpus, content_transformer(tolower))
  corpus <- tm_map(corpus, removeWords, stopwords("english"))

  dtm <- TermDocumentMatrix(corpus)
  dtm.M <- as.matrix(dtm)
  words <- sort(rowSums(dtm.M),decreasing=TRUE)
  dtm_df <- data.frame(word = names(words),freq=words)
  top10_bar <- ggplot(dtm_df[1:10,], aes(y=freq, x=reorder(word, -freq))) +
    geom_bar(position="stack", stat="identity") +
    ggtitle("Top 10 most frequent words") +
    ylab("Word frequencies") + xlab("Word")
  return(top10_bar)
}
```

```
# gen_dtm_df(df.full)
# gen_dtm_df(df.full[df.full$ subreddit == "ptsd", ])
# gen_dtm_df(df.full[df.full$ subreddit == "assistance", ])
# gen_dtm_df(df.full[df.full$ subreddit == "relationships", ])
# gen_dtm_df(df.full[df.full$ subreddit == "survivorsofabuse", ])
# gen_dtm_df(df.full[df.full$ subreddit == "domesticviolence", ])
# gen_dtm_df(df.full[df.full$ subreddit == "anxiety", ])
# gen_dtm_df(df.full[df.full$ subreddit == "homeless", ])
# gen_dtm_df(df.full[df.full$ subreddit == "food_pantry", ])
# gen_dtm_df(df.full[df.full$ subreddit == "almosthomeless", ])
# gen_dtm_df(df.full[df.full$ subreddit == "stress", ])
```

## Word Cloud

```
set.seed(5312)
wordcloud(dtm_df$word, dtm_df$freq, scale=c(1.5, .2), min.freq=3,
          max.words=300, random.order=FALSE, rot.per=.15,
          colors=brewer.pal(8, "Dark2"))
```



```
# findAssocs(dtm, terms = findFreqTerms(dtm, lowfreq = 900), corlimit = 0.25)
```

```
# regular sentiment score using get_sentiment() function and method of your choice
# please note that different methods may have different scales
```

# -1 - +1

```
set.seed(5312)
```

```
syuzhet <- get_sentiment(df.full$text, method="syuzhet")
```

```
df.full$syuzhet <- syuzhet
```

# bing

```
bing <- get_sentiment(df.full$text, method="bing")
```

```
df.full$bing <- bing
```

## #affin

```
afinn <- get_sentiment(df.full$text, method="afinn")
```

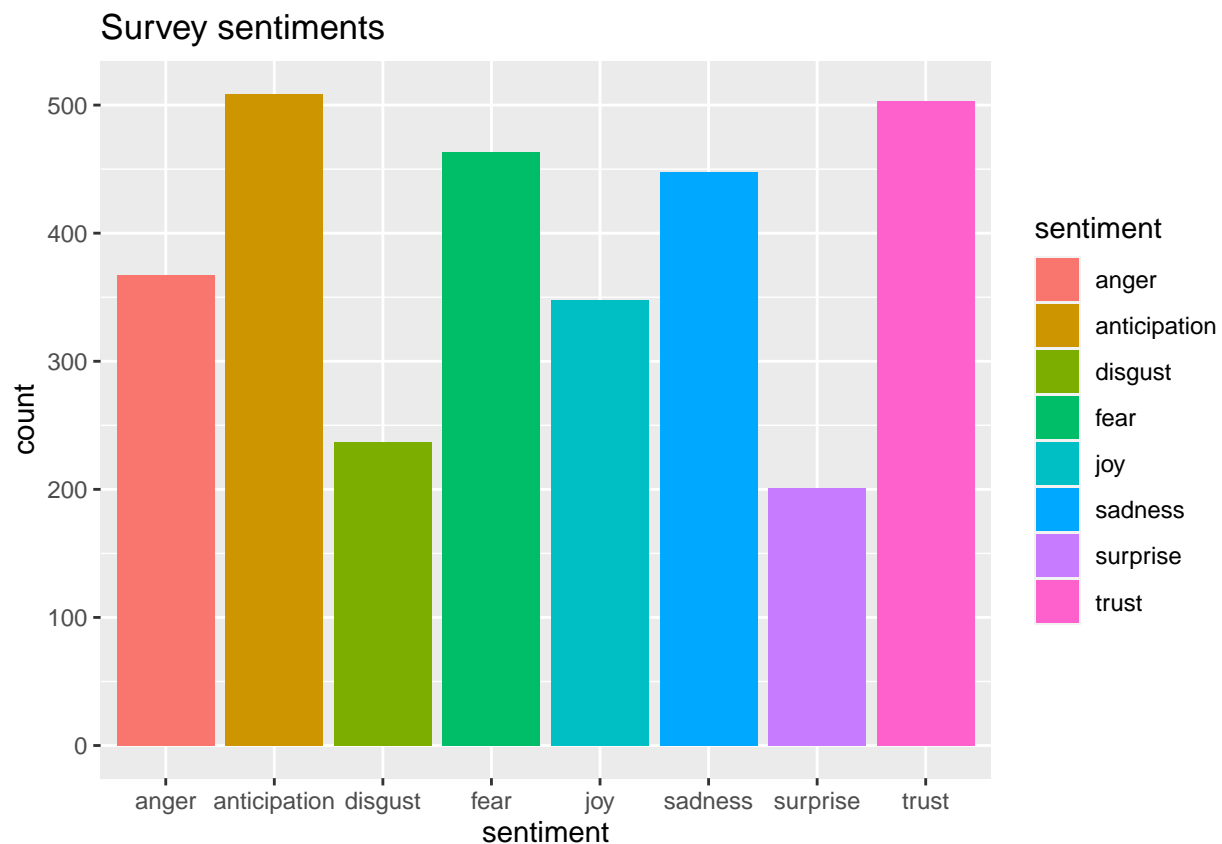
```
df.full$afinn <- afinn
```

```
# run nrc sentiment analysis to return data frame with each row classified as one of the following
# emotions, rather than a score:
# anger, anticipation, disgust, fear, joy, sadness, surprise, trust
# It also counts the number of positive and negative emotions found in each row
nrc<-get_nrc_sentiment(df.full$text)
```

```
df.full <- cbind(df.full, nrc)
```

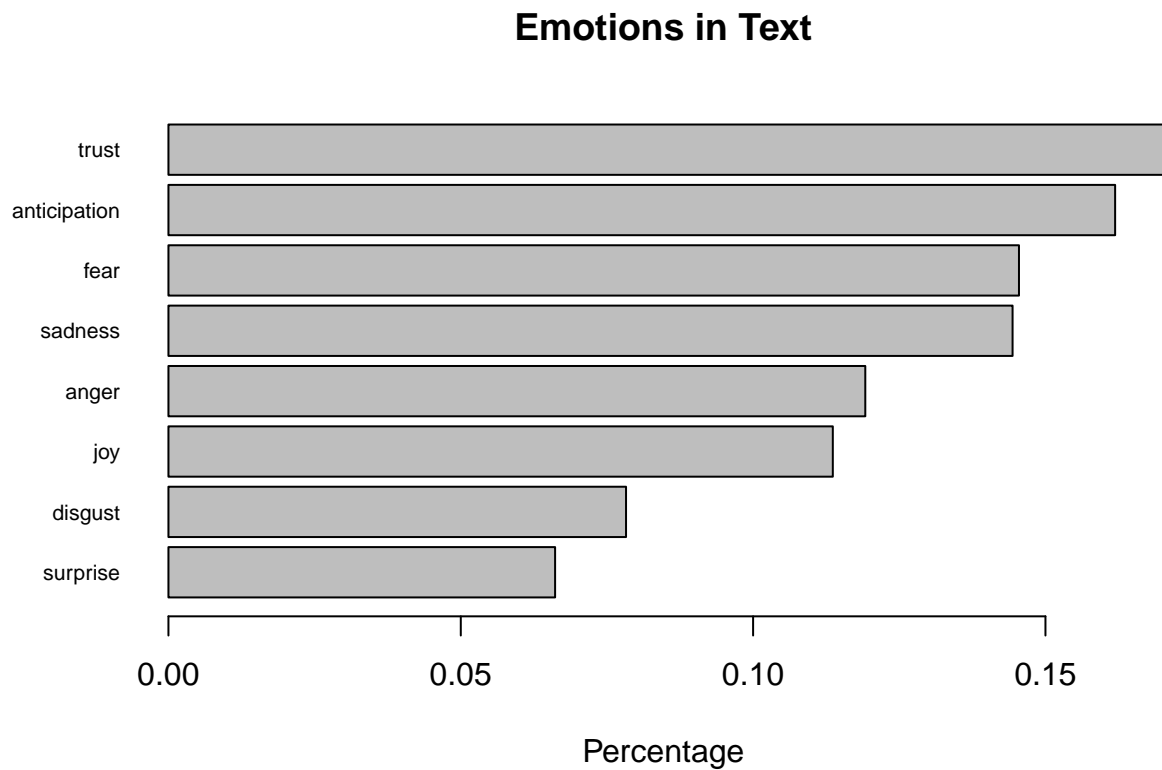
```
# standardize
df.full[4:19] <- scale(df.full[4:19])
```

```
#transpose
td<-data.frame(t(nrc))
#The function rowSums computes column sums across rows for each level of a grouping variable.
td_new <- data.frame(rowSums(td[2:253]))
#Transformation and cleaning
names(td_new)[1] <- "count"
td_new <- cbind("sentiment" = rownames(td_new), td_new)
rownames(td_new) <- NULL
td_new2<-td_new[1:8,]
#Plot One - count of words associated with each sentiment
quickplot(sentiment, data=td_new2, weight=count, geom="bar", fill=sentiment, ylab="count")+ggtitle("Survey sentiments")
```





```
#Plot two - count of words associated with each sentiment, expressed as a percentage
barplot(
  sort(colSums(prop.table(nrc[, 1:8]))),
  horiz = TRUE,
  cex.names = 0.7,
  las = 1,
  main = "Emotions in Text", xlab="Percentage"
)
```



## Prediction

```
set.seed(5312)
train_size <- floor(0.8 * nrow(df.full))

in_rows <- sample(c(1:nrow(df.full)), size = train_size, replace = FALSE)

df.train <- df.full[in_rows, ]
df.test <- df.full[-in_rows, ]

df.train = df.train[-(1:2)]
df.train$is_stress = as.factor(df.train$is_stress)
df.test$is_stress = as.factor(df.test$is_stress)
```

```
# Run algorithms using 10-fold cross validation
control <- trainControl(method="cv", number=10)
metric <- "Accuracy"
```

## Logistic regression

```
# df.train.stress = df.train[df.train$is_stress == 1, ]
set.seed(5312)
lr.fit <- glm(formula = is_stress ~ ., data = df.train, family=binomial)
summary(lr.fit)
```

```
##
## Call:
## glm(formula = is_stress ~ ., family = binomial, data = df.train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -3.2210  -0.9563   0.3323   0.9603   2.4572
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.187569   0.043551   4.307 1.66e-05 ***
## social_karma   -0.290456   0.086242  -3.368 0.000757 ***
## social_upvote_ratio 0.104413   0.043711   2.389 0.016908 *
## social_num_comments 0.121356   0.070339   1.725 0.084475 .
## syuzhet        -0.634028   0.121587  -5.215 1.84e-07 ***
## bing            0.018100   0.090128   0.201 0.840838
## afinn          -0.453998   0.095298  -4.764 1.90e-06 ***
## anger           0.073398   0.080902   0.907 0.364279
## anticipation    -0.037069   0.057635  -0.643 0.520122
## disgust         -0.067970   0.065484  -1.038 0.299291
## fear            0.008741   0.076870   0.114 0.909466
## joy            -0.018343   0.073268  -0.250 0.802315
## sadness         0.152041   0.079155   1.921 0.054757 .
## surprise        0.132108   0.053142   2.486 0.012921 *
## trust          -0.002089   0.064824  -0.032 0.974292
## negative        0.156284   0.100792   1.551 0.121009
## positive        0.064467   0.076763   0.840 0.401011
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 3932.8  on 2841  degrees of freedom
## Residual deviance: 3220.7  on 2825  degrees of freedom
## AIC: 3254.7
##
## Number of Fisher Scoring iterations: 4
```

```
set.seed(5312)
summary(lr.fit2)
```

```
##
## Call:
## glm(formula = is_stress ~ social_karma + social_upvote_ratio +
##      social_num_comments + syuzhet + afinn + sadness + surprise +
##      negative, family = binomial, data = df.train)
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
## -3.2097  -0.9555   0.3392   0.9605   2.4173
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)      0.18590    0.04346   4.277 1.89e-05 ***
## social_karma     -0.28379    0.08554  -3.318 0.000908 ***
## social_upvote_ratio 0.10601    0.04351   2.437 0.014827 *
## social_num_comments 0.11500    0.06982   1.647 0.099523 .
## syuzhet          -0.59020    0.09027  -6.538 6.22e-11 ***
## afinn            -0.46471    0.08899  -5.222 1.77e-07 ***
## sadness           0.14912    0.07459   1.999 0.045593 *
## surprise          0.12525    0.04744   2.640 0.008284 **
## negative          0.17835    0.08484   2.102 0.035540 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##      Null deviance: 3932.8  on 2841  degrees of freedom
## Residual deviance: 3223.4  on 2833  degrees of freedom
## AIC: 3241.4
##
## Number of Fisher Scoring iterations: 4
```

*# The Variance Inflation Factor(VIF) is used to measure the multicollinearity between predictor variabl*

```
# VIF(lr.fit2)
```

```
set.seed(5312)
lr.fit3 <- train(is_stress ~ social_karma + social_upvote_ratio +
                 social_num_comments + syuzhet + afinn +
                 sadness + surprise + negative, data=df.train,
                 method="glm", metric=metric, trControl=control)
summary(lr.fit3)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min       1Q   Median       3Q      Max
```

```
## -3.2097 -0.9555 0.3392 0.9605 2.4173
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)    0.18590    0.04346   4.277 1.89e-05 ***
## social_karma   -0.28379    0.08554  -3.318 0.000908 ***
## social_upvote_ratio 0.10601    0.04351   2.437 0.014827 *
## social_num_comments 0.11500    0.06982   1.647 0.099523 .
## syuzhet        -0.59020    0.09027  -6.538 6.22e-11 ***
## afinn          -0.46471    0.08899  -5.222 1.77e-07 ***
## sadness         0.14912    0.07459   1.999 0.045593 *
## surprise        0.12525    0.04744   2.640 0.008284 **
## negative        0.17835    0.08484   2.102 0.035540 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##    Null deviance: 3932.8  on 2841  degrees of freedom
## Residual deviance: 3223.4  on 2833  degrees of freedom
## AIC: 3241.4
##
## Number of Fisher Scoring iterations: 4
```

## Decision tree

```
set.seed(5312)
dt.fit <- train(is_stress ~ ., data=df.train, method="rpart", metric=metric, trControl=control)
```

## Random forest

```
set.seed(5312)
rf.fit <- train(is_stress ~ ., data=df.train, method="rf", metric=metric, trControl=control)
```

## SVM

```
set.seed(5312)
svm.fit <- train(is_stress ~ ., data=df.train, method="svmRadial", metric=metric, trControl=control)
```

## GBM

```
set.seed(5312)
gbm.fit <- train(is_stress ~ ., data=df.train, method="gbm",
                 metric=metric, trControl=control, verbose=FALSE)
```

## Conclusion

```
set.seed(5312)
AccCalc <- function(TestFit, name) {
  # prediction
  predictedval <- predict(TestFit, newdata=df.test)

  # summarize results with confusion matrix
  cm <- confusionMatrix(predictedval, df.test$is_stress)

  # calculate accuracy of the model
  Accuracy<-round(cm$overall[1],4)
  Sensitivity <- round(cm$byClass[1], 4)
  Specificity <- round(cm$byClass[2], 4)
  acc <- data.frame(Accuracy, Sensitivity, Specificity)

  roc_obj <- roc(df.test$is_stress, as.numeric(predictedval))
  acc$Auc <- auc(roc_obj)

  acc$FitName <- name
  return(acc)
}

accAll <- AccCalc(lr.fit3, "lr")
accAll <- rbind(accAll, AccCalc(dt.fit, "dt"))
accAll <- rbind(accAll, AccCalc(rf.fit, "rf"))
accAll <- rbind(accAll, AccCalc(svm.fit, "svm"))
accAll <- rbind(accAll, AccCalc(gbm.fit, "gbm"))
rownames(accAll) <- c()
arrange(accAll,desc(Accuracy))
```

##	Accuracy	Sensitivity	Specificity	Auc	FitName
## 1	0.7201	0.6647	0.7726	0.7186713	lr
## 2	0.7117	0.6445	0.7753	0.7099256	gbm
## 3	0.7089	0.6329	0.7808	0.7068849	rf
## 4	0.7032	0.5983	0.8027	0.7005028	svm
## 5	0.6906	0.6387	0.7397	0.6892272	dt

```
set.seed(5312)
pred.lr <- predict(lr.fit3, newdata=df.test)
pred.dt <- predict(dt.fit, newdata=df.test)
pred.rf <- predict(rf.fit, newdata=df.test)
pred.svm <- predict(svm.fit, newdata=df.test)
pred.gbm <- predict(gbm.fit, newdata=df.test)

lr.roc <- roc(response = df.test$is_stress, predictor = as.numeric(pred.lr))
dt.roc <- roc(response = df.test$is_stress, predictor = as.numeric(pred.dt))
rf.roc <- roc(response = df.test$is_stress, predictor = as.numeric(pred.rf))
svm.roc <- roc(response = df.test$is_stress, predictor = as.numeric(pred.svm))
gbm.roc <- roc(response = df.test$is_stress, predictor = as.numeric(pred.gbm))

plot(lr.roc, legacy.axes = TRUE, print.auc.y = 0.95, print.auc = TRUE, print.auc.x = 0)
```

```

plot(dt.roc, col = "blue", add = TRUE, print.auc.y = 0.55, print.auc = TRUE, print.auc.x = 0)
plot(rf.roc, col = "red", add = TRUE, print.auc.y = 0.75, print.auc = TRUE, print.auc.x = 0)
plot(svm.roc, col = "darkgreen", add = TRUE, print.auc.y = 0.65, print.auc = TRUE, print.auc.x = 0)
plot(gbm.roc, col = "orange", add = TRUE, print.auc.y = 0.85, print.auc = TRUE, print.auc.x = 0)
legend("bottomright", c("Logistic", "Decision Tree", "Random Forest", "SVM", "GBM"),
      lty = c(1,1), lwd = c(2, 2), col = c("black", "blue", "red", "darkgreen", "orange"), cex = 0.75)

```

