Final Project

Bryce Fenlon

5/6/2020

Load packages here

```r
library(dplyr)
```

```
##
## 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
```

```r
library(ggformula)
```

```
## Loading required package: ggplot2

## Loading required package: ggstance

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh

##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")
```

```r
library(tidyr)
```

Load the CSV data exported from python

```
before_tweets <- read.csv('before_r.csv', header = TRUE)
after_tweets <- read.csv('after_r.csv', header = TRUE)
```

Check to see if there are any NA text entries

```
which(is.na(before_tweets$text))
```

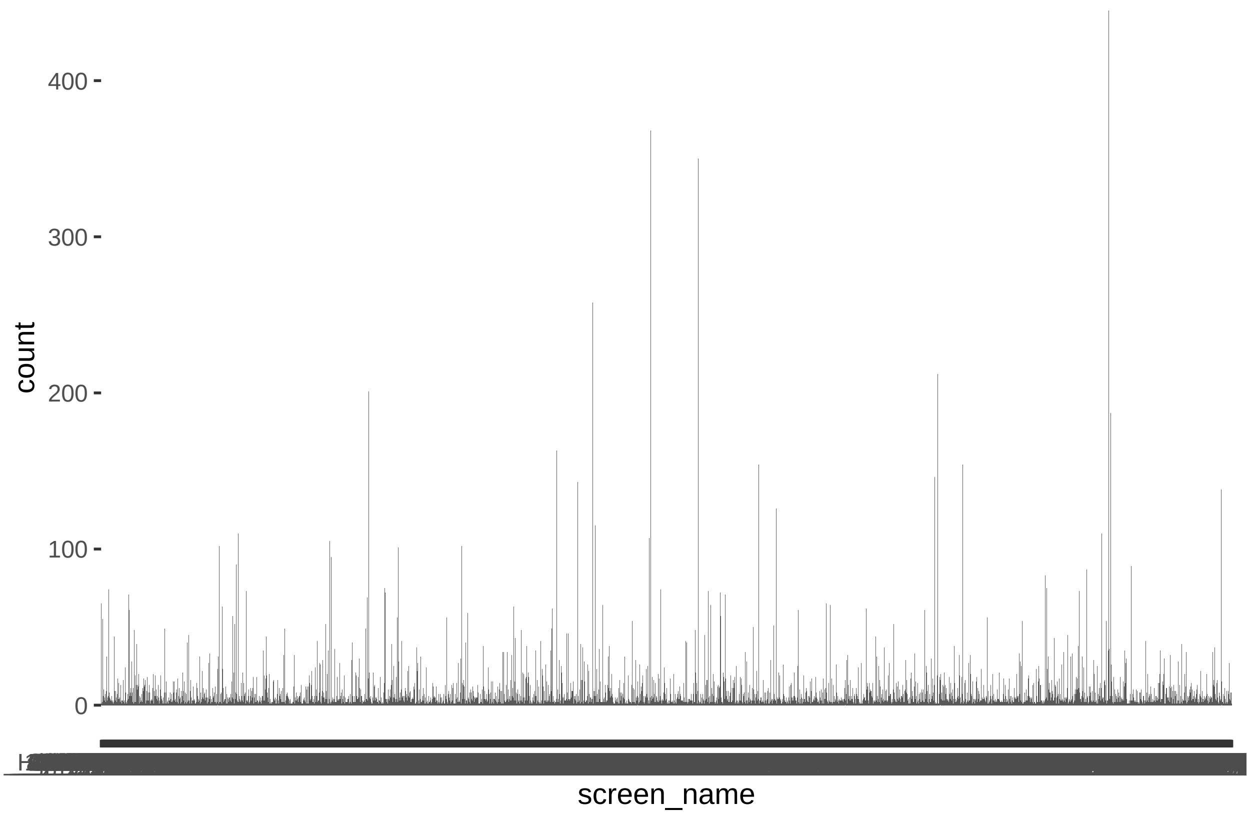
```
## integer(0)
```

```
which(is.na(after_tweets$text))
```

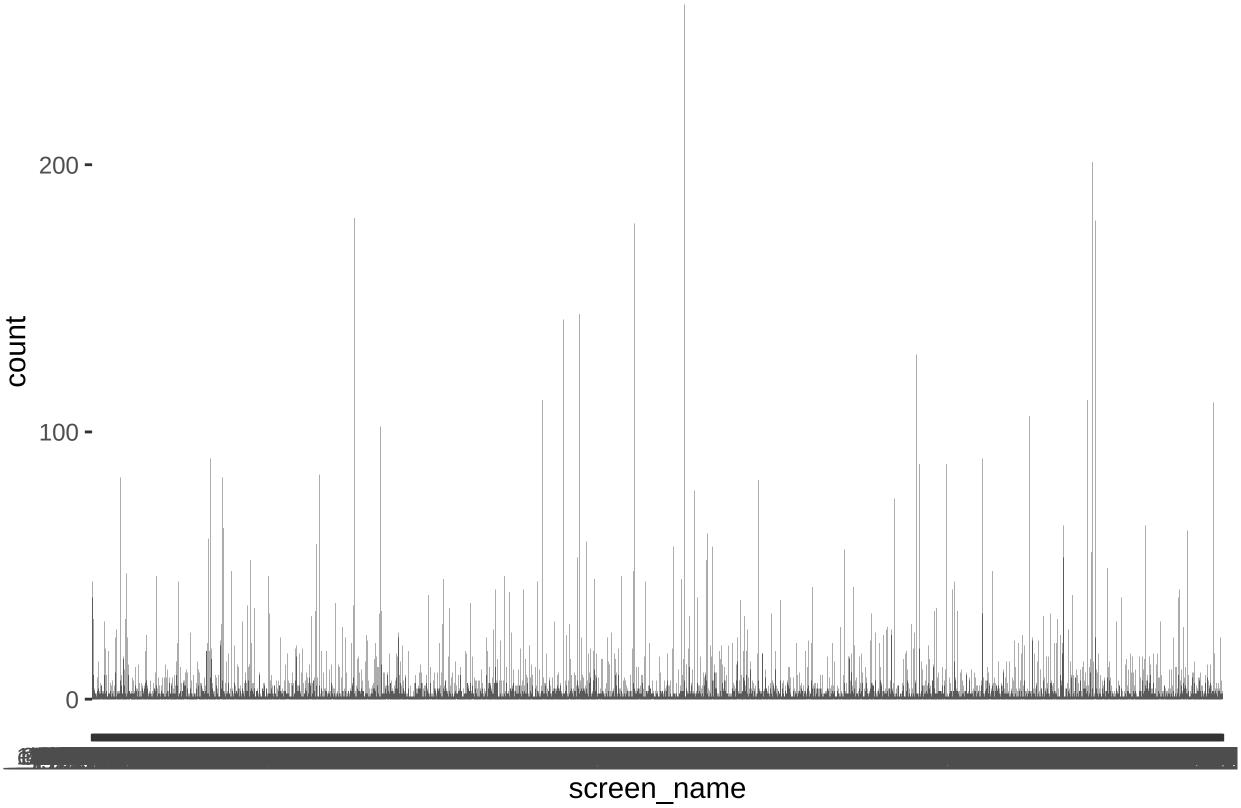
```
## integer(0)
```

Now it's time for a little exploratory analysis. We'll start by making histograms of tweet counts by user, to see if there are any outliers that tweet a lot. This would probably have an impact on the population as a whole, as we want to look at overall sentiment in the population.

```
before_tweets %>% gf_histogram(~screen_name, stat='count', alpha = 1)
```



```
after_tweets %>% gf_histogram(~screen_name, stat='count', alpha = 1)
```



There are a few obvious `screen_name` tweet count outliers in each set of data. Figure out which ones screen names have over 200 tweets, and remove those users and their tweets.

```
to_remove_before <- before_tweets %>% count(screen_name) %>% filter(n > 200)
to_remove_after <- after_tweets %>% count(screen_name) %>% filter(n > 200)
alt_before_tweets <- before_tweets[!(before_tweets$screen_name %in%
                                       to_remove_before$screen_name),]
alt_after_tweets <- after_tweets[!(after_tweets$screen_name %in%
                                     to_remove_after$screen_name),]
```

The data is not the same length because of tweet volume limitations after our desired date and the removal of our 'super tweeters'. Sample the data from `alt_before_tweets` to have a maximum amount of data equal to the length of `alt_after_tweets`.

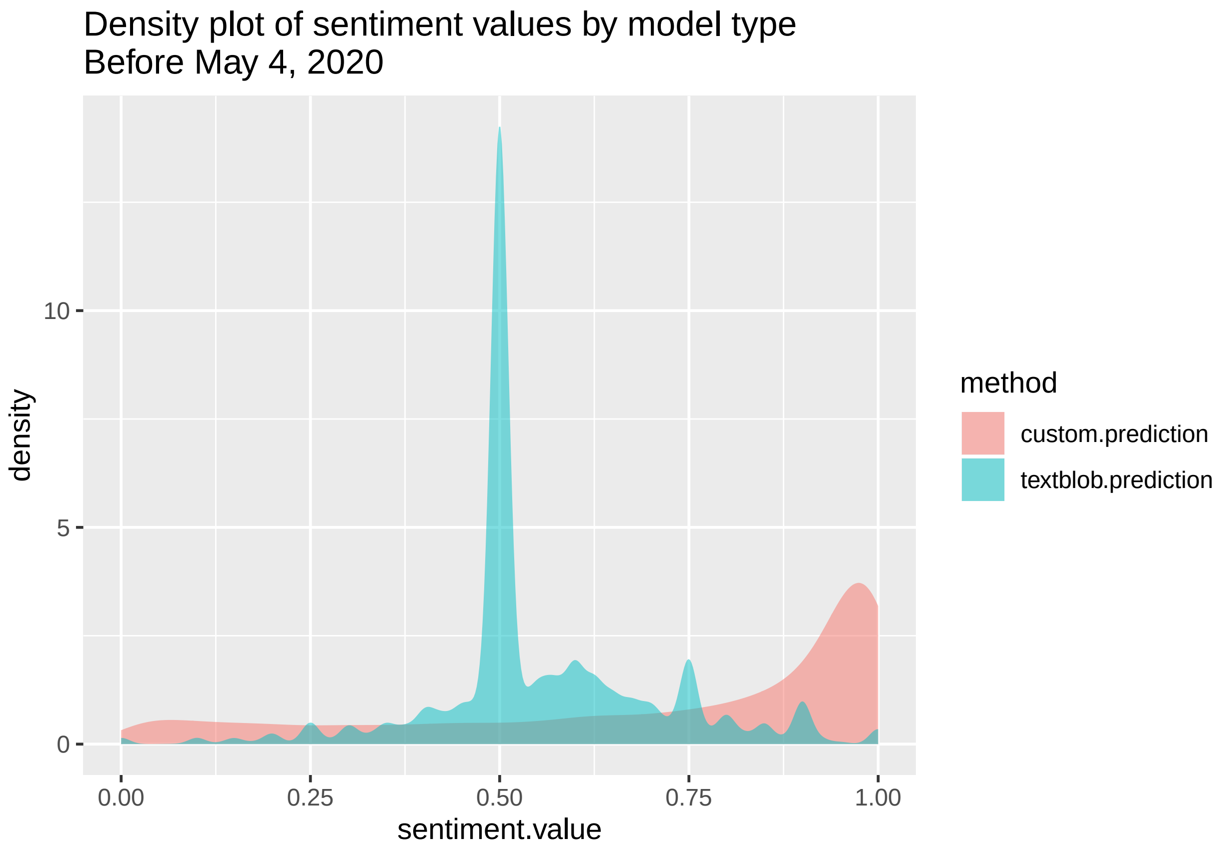
```
max_length <- nrow(alt_after_tweets)
alt_before_tweets <- alt_before_tweets[sample(nrow(alt_before_tweets), max_length),]
```

Okay. Our data is of equal length, and while there are some people that tweeted quite a lot in our two time spans we have removed the few outliers that tweeted the most. Let's alter the sentiment columns so that the scales match, by squeezing the textblob column with the formula [(x - x_min)/x_range]*n, where x is the original value, x_min is the minimum of the textblob column (-1, technically), x_range is the range of the textblob column (2, -1 to 1), and n is the upper bound of the rescaled variable (1).

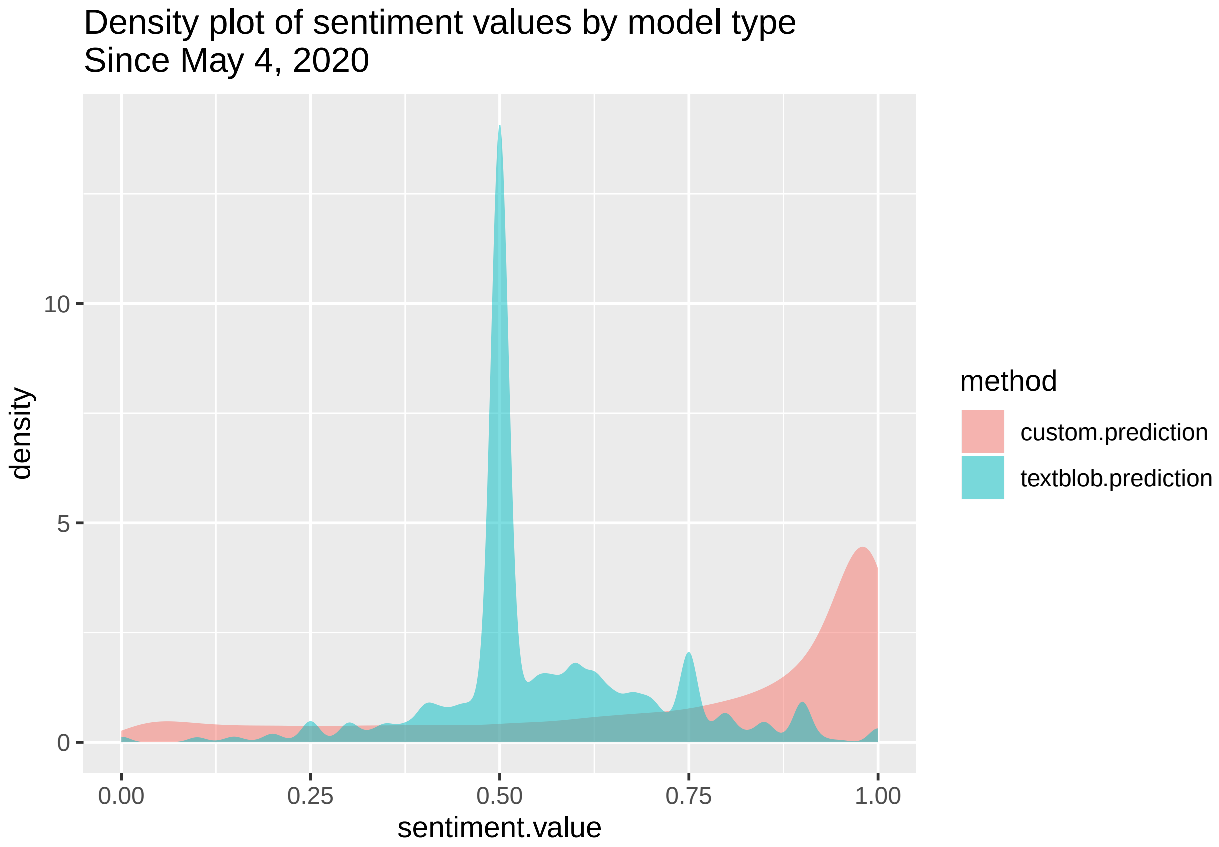
```
# Define function to rescale variable
reScale <- function(x){
  to_return = (x + 1)/2
}

# Rescale textblob column in both sets of data
alt_before_tweets$textblob.prediction <- reScale(alt_before_tweets$textblob.prediction)
alt_after_tweets$textblob.prediction <- reScale(alt_after_tweets$textblob.prediction)
```

Let’s get crackin’ on some analysis.

```
# Create density plots of sentiment values
long_before_sent <- alt_before_tweets %>%
  pivot_longer(c(custom.prediction, textblob.prediction),
               names_to = 'method', values_to = 'sentiment.value')
long_before_sent %>% gf_density(~sentiment.value, fill=~method) %>%
  gf_labs(title = 'Density plot of sentiment values by model type \nBefore May 4, 2020')
```



```
long_after_sent <- alt_after_tweets %>%
  pivot_longer(c(custom.prediction, textblob.prediction),
               names_to = 'method', values_to = 'sentiment.value')
long_after_sent %>% gf_density(~sentiment.value, fill=~method) %>%
  gf_labs(title = 'Density plot of sentiment values by model type \nSince May 4, 2020')
```



Now we're getting somewhere. By the looks of it, there is a measureable increase toward positive sentiment (values closer to 1) in the custom model between the two periods. Let's turn away from cool visualization to some solid statistics.

```r
# Run paired t-tests on both models between time periods
# (paired because these are two observations of the same area from two different time periods)
t.test(alt_before_tweets$custom.prediction, alt_after_tweets$custom.prediction,
       paired = TRUE, alternative = 'two.sided')
```

```
##
##  Paired t-test
##
## data:  alt_before_tweets$custom.prediction and alt_after_tweets$custom.prediction
## t = -13.015, df = 21790, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.04327279 -0.03194488
## sample estimates:
## mean of the differences
##             -0.03760883
```

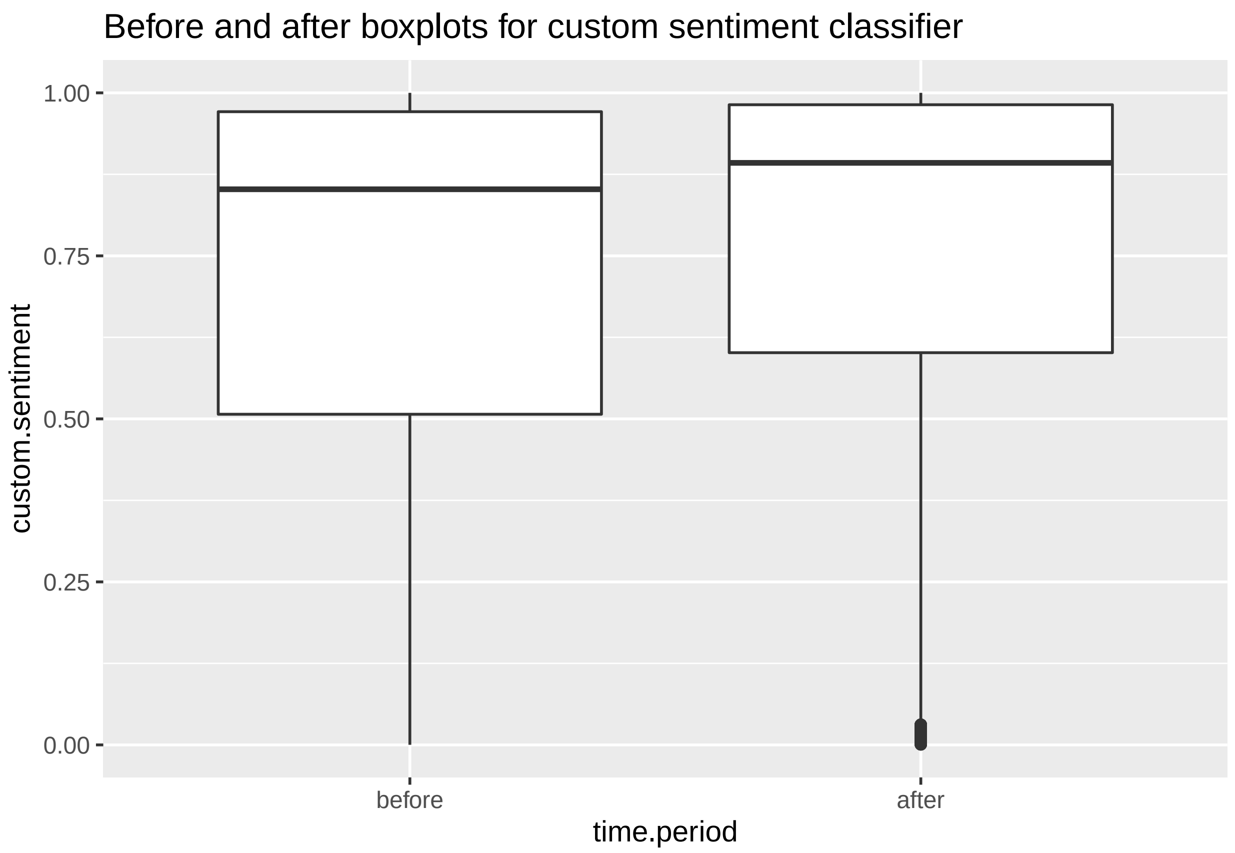
```r
t.test(alt_before_tweets$textblob.prediction, alt_after_tweets$textblob.prediction,
       paired = TRUE, alternative = 'two.sided')
```

```
##
```

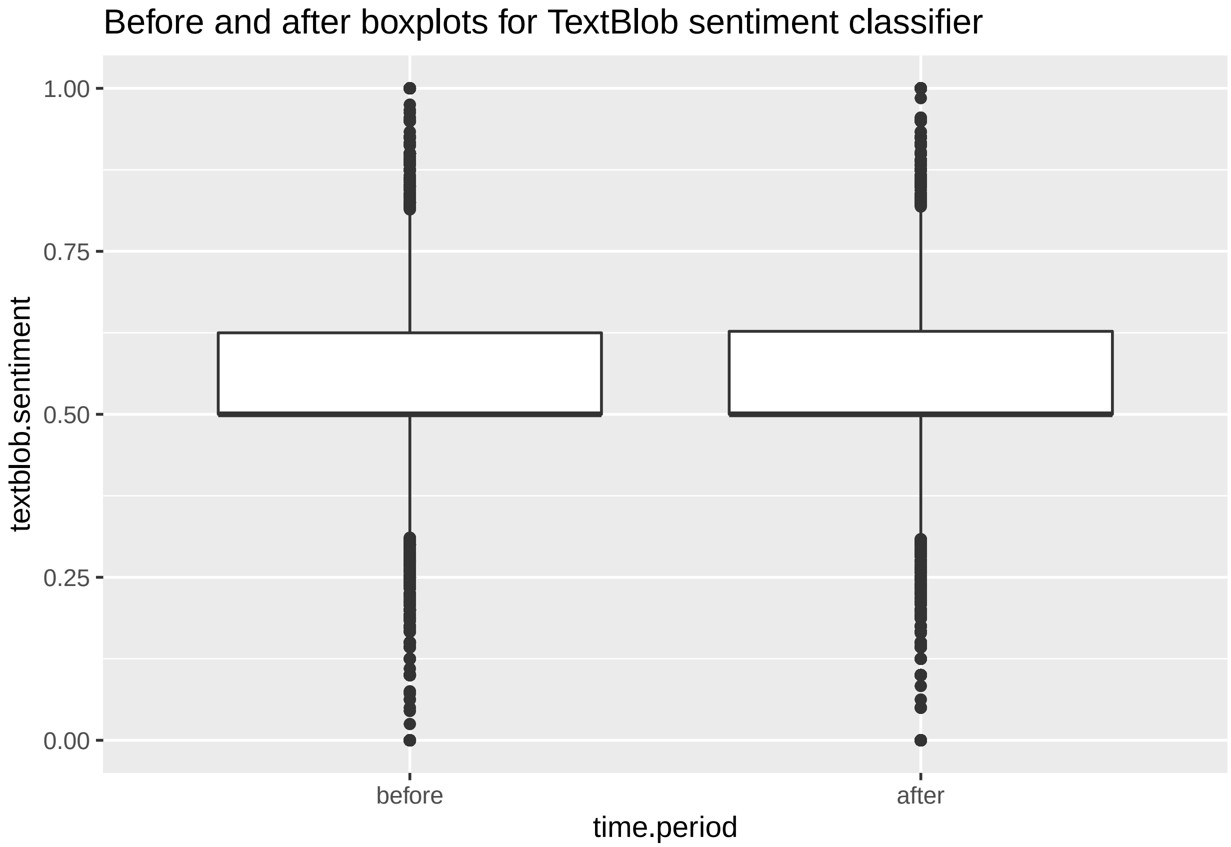
```
##  Paired t-test
##
## data:  alt_before_tweets$textblob.prediction and alt_after_tweets$textblob.prediction
## t = -0.86277, df = 21790, p-value = 0.3883
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.004067400  0.001581082
## sample estimates:
## mean of the differences
##            -0.001243159
```

As a final step, two boxplot graphs will be created. One will be for the custom classifier, and another for the TextBlob classifier.

```
# 'Melt' a data.frame of the custom sentiment values for before and
# after into one long data.frame with names and values
before_after_custom <- data.frame(alt_before_tweets$custom.prediction,
                                  alt_after_tweets$custom.prediction)
colnames(before_after_custom) <- c('before', 'after')
long_before_after_custom <- before_after_custom %>%
  pivot_longer(c(before, after), names_to = 'time.period', values_to = 'custom.sentiment')

# Re-order factors to give plot a logical flow, before --> after
long_before_after_custom$time.period <- factor(long_before_after_custom$time.period,
                                               levels = c('before', 'after'))

# Plot
long_before_after_custom %>% gf_boxplot(custom.sentiment ~ time.period) %>%
  gf_labs(title = 'Before and after boxplots for custom sentiment classifier')
```



```
# Rinse and repeat for TextBlob values
before_after_blob <- data.frame(alt_before_tweets$textblob.prediction,
                                alt_after_tweets$textblob.prediction)
colnames(before_after_blob) <- c('before', 'after')
long_before_after_blob <- before_after_blob %>%
  pivot_longer(c(before, after), names_to = 'time.period', values_to = 'textblob.sentiment')

# Re-order factors to give plot a logical flow, before --> after
long_before_after_blob$time.period <- factor(long_before_after_blob$time.period,
                                             levels = c('before', 'after'))

# Plot
long_before_after_blob %>% gf_boxplot(textblob.sentiment ~ time.period) %>%
  gf_labs(title = 'Before and after boxplots for TextBlob sentiment classifier')
```


Before and after boxplots for TextBlob sentiment classifier
textblob.sentiment
1.00
0.75
0.50
0.25
0.00
before
after
time.period