453 Final Project

Nils Dosaj Mikkelsen (V00901004)

01/04/2022

Introduction

Comment sections on the Internet have a reputation for being infamously negative places [1]. Anyone who has spent any time on YouTube, public forums or has had the misfortune of scrolling too far down on a news article has invariably witnessed the bickering between two, often anonymous, parties quickly devolving into a virtual tennis match of often poorly crafted and misspelled insults. The Internet can also be a beautiful place. A place that fosters social connection, the sharing of ideas and imparting of knowledge, or even a place to fall in love. But at times it appears as though seemingly innocuous forums or meeting grounds can play host to fostering a toxic rapport between two people that is completely disjoint from the original forum's objective itself. Online video games, a leisure activity, is a notorious breeding ground for racial epithets and hateful verbal exchanges. It can seem as though the virtual backdrop in which the conversation is happening does not have any impact on how negative the sentiment of a conversation can become, or does it?

Reddit is a particularly interesting platform. It is a widely popular social networking site that allows users to discuss and aggregate different topics and ideas in a forum style chat. Currently ranked the 20th most popular website in the world [2], people from all over the globe can post images, videos, GIFs, articles or blocks of text each with its own corresponding and separate comment section. The forums of Reddit are divided by topic, these forums are referred to as subreddits. There are subreddits that deal only with popular science, pictures of cute kittens, food, statistics and everything in between; if you can think of a topic, there probably exists a subreddit for it. Each subreddit has a team of moderators responsible for enforcing the rules and removing irrelevant content. Part of what initially made Reddit unique, was the fact that both posts and comments have an inherent voting system. If you like a post or a comment, you can "upvote" it. Alternatively, if you disagree with a post or comment you may "downvote" it. Posts and comments are then ordered according to the amount of upvotes or downvotes that they receive. In order to keep the top posts of each subreddit from becoming too stagnant, a ranking algorithm is applied that down weights a post the longer and closer it remains in proximity to the top of its respective subreddit [3]. As a result, it is not uncommon to see the top 5-10 posts on a given subreddit change daily. The comments however, do not change in their order except when the amount of votes between two comments changes the order.

Some subreddits are more active than others and there exists a list [4] of the top 10 most active subreddits. The top 10 most active subreddits are:

1. Funny
2. AskReddit
3. Gaming
4. Aww
5. Music
6. Pics
7. Worldnews
8. Movies
9. Science
10. Todayilearned

As we can see from our list, the topics are quite varied, and with the popularity and amount of comments and conversations that take place on Reddit, we have an excellent environment to study and analyse our research problem.

Research Problem

Does the subreddit in which anonymous conversations take place have an impact on the sentiment of the comments themselves?

In order to attempt to answer this question we need two things:

1. A way to collect the immense amounts of data that is generated by Reddit every day.
2. A way to quantify and analyse the sentiment of each comment.

Sentiment Analysis

Sentiment analysis is a type of text analysis/natural language processing that is used to systematically identify, extract, quantify and study effective states and subjective information [5]. In the context of this project, it allows us to assign a numerical value based on the number of positive versus negative words that a comment contains. Since the sentiment score of a given comment does not naturally fall on a bounded scale, we normalize the sentiment score for each comment to aid in comparison by employing the following formula [6]:

$$\frac{(tpw - tnw)}{tw}$$

Where:

1. tpw: Total positive words
2. tnw: Total negative words
3. tw: Total words

This has the effect of bounding sentiment on the scale [-1,1]. We define the following discrete sections of the scale as:

$$\begin{aligned}
-1 \leq sentiment < -0.5 &\quad \rightarrow Negative \\
-0.5 \leq sentiment < -0.1 &\quad \rightarrow Negative - Neutral \\
-0.1 \leq sentiment \leq 0.1 &\quad \rightarrow Neutral \\
0.1 < sentiment < 0.5 &\quad \rightarrow Positive - Neutral \\
0.5 \leq sentiment \leq 1 &\quad \rightarrow Positive
\end{aligned}$$

The Syuzhet Package

In order to facilitate our sentiment analysis, we use the `Syuzhet` package developed by the NLP group at Stanford [7]. The package provides a variety of useful functions, but we only need the `get_nrc_sentiment()` function which returns a table of words categorized into eight different emotions along with a total tabulation of negative and positive word sentiment counts. For our experiment, we only need the columns that relate to the total counts of negative and positive words.

Experimental Design

For the period of one day, data was collected from the current top five posts from each subreddit every 8 hours. To address the three principles of experimental design we did the following:

1. **Randomization:** Randomly sample 30 comments from each subreddit for each day
2. **Replication:** Repeat our experiment 7 times, once for each day of the week.
3. **Blocking:** Detect if power users (users who's personal comment count in our data set defined them as outliers) had an impact on our model, and if so, include this information in our model.

While accounting for power users may be controllable. There are certain other factors that are not. Due to the timeline of the project, only one week's worth of data was able to be collected. As a result, it was not possible to tell if the day of the week/month played an impact in the average sentiment for each subreddit. Another nuisance factor is the fact that the conversation on certain subreddits is based off of world events, particularly `worldnews`. Given the recent events going on with Ukraine, it is unknown how this would've impacted the average sentiment score for the week's worth of collected data versus had this data been collected during a different week where world politics was less volatile. As a result, world news and world events are in effect, a nuisance factor for this model. In general, we have no control over the content posted in any of the subreddits we examined, this would be another nuisance factor.

Statistical Model

We use a single factor 10 level experiment. The single factor is the subreddit, of which we consider 10 possible choices (listed above). The response variable is the average sentiment of all 7 days with the range $[-1, 1]$. Given the fact that the runs of the experiment were based on time, we were not able to randomize their ordering. As a result, along with the fact that power users were deemed insignificant (see `anaylsis` section), the model that we used was the **means** model:

$$Y_{ij} = \mu + \tau_i + \epsilon_{ij}$$

Where $i = 1, 2, \ldots, 10$ denoting the top 10 subreddits and $j = 1, 2, \ldots, 7$ denoting the days of the week or replicates.

Data Collection

The data was collected by leveraging the following two tools:

PRAW

Data was collected via a Python script (see `Appendix Part A`) that leveraged the PRAW (Python Reddit API Wrapper) library [8]. This library, which should be obvious given its name, act as a wrapper for the Reddit API and facilitates the pulling of data through several well-defined classes and objects. There exists object types for posts, users and comments. The relevant data for each was extracted and compiled into a data frame.

Windows Task Scheduler

In order to facilitate the execution of our Python script, we used the Windows Task Scheduler to automatically run the script every eight hours at midnight, 8 AM and 4 PM respectively. The script would take around seven minutes to run, and would consist of approximately 13,000 to 17,000 rows per iteration.

Data Dimensions

Once all the data was collected, there was approximately 325,000 total observations. Due to the spacing of the Windows Task Scheduler's execution, there did exist some duplicate entries when all data pulls were fully compiled. The `distinct()` function from the `dplyr` library removed a substantial amount leaving approximately 220,000 remaining unique observations. For each observation, the relevant metadata that was collected for this experiment was:

1. Subreddit
2. Post name
3. Comment
4. Comment Author

There were several other rows that were also collected, such as Karma (the number of upvotes), Karma Ratio (the ratio of upvotes to downvotes), comment time, number of comments etc. but these were ultimately not necessary for our model, but could prove fruitful for later experiments and more in-depth future analysis.

Analysis

The analysis portion of this project was substantial and took an immense amount of time and effort. Approximately 750 lines of code are included in the `Appendix Part B` but ultimately there were closer to 2000-3000 lines written for this project. The lines that were omitted were not relevant to the final outcome and were the result of several weeks of experimentation to determine the best most efficient practices for cleaning and analysing the data along with the creation of several functions, the majority of which proved fruitless or were improved upon in the final iteration of the analysis process.

As a result of the complexity required to clean and format the data, we will restrict this section to only the most relevant pieces. A full breakdown of the code that was run and its corresponding outputs, exists in `Appendix Part B`.

Are Power Users Significant?

We note that within our data we have approximately 225,000 distinct comments made by approximately 137,000 distinct users. Within these 137,000 distinct users, 6342 of them are deemed *power users* by our outlier analysis function. This proportion of *power users* to *non–power users* comprises approximately 4.63% of the total data set. However, once we have randomly sampled the observations necessary for our experiment, we find that 460 of the 2048 distinct commenters within our 2100 observations are deemed *power users*; approximately 22.5%. While this is expected given the disproportionate amount of comments that these *power users* contribute, we wish to check first if their average sentiment is significantly different from those of *non–power users*, and thus, a nuisance factor that will have to be taken into account. `Fig 1` shows the distribution of sentiment between the two groups.

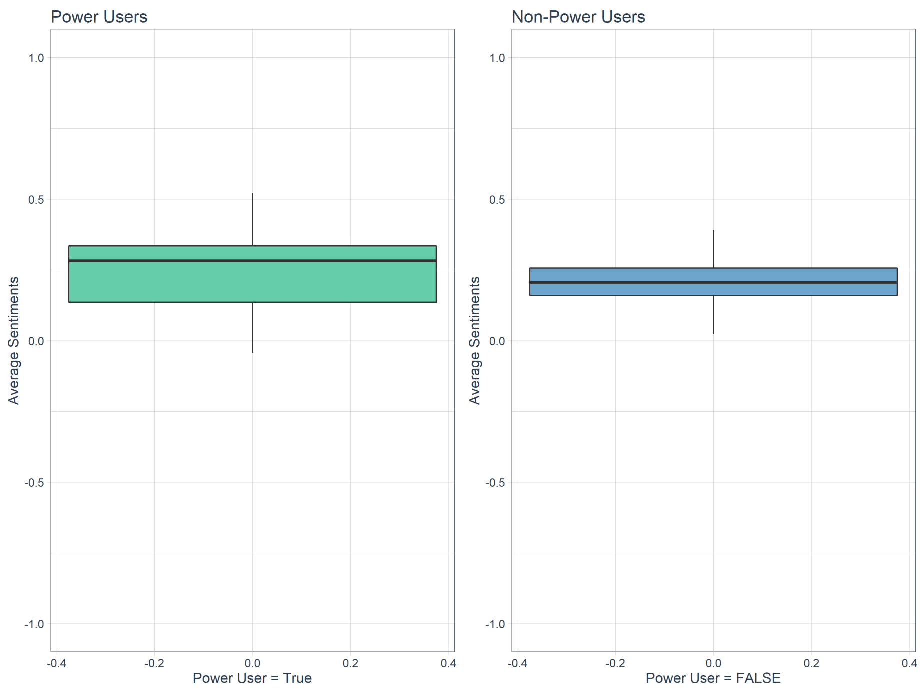


Fig 1: Power users/non-powers users boxplots

We notice that they do not appear to be noticeably different, and after conducting a two-sided `t.test()`, we find that we do not reject the null hypothesis that there difference is significant. Prior to the `t.test()` we verify that the group's respective variances are equal and both data sets are normally distributed. Given these results, we do not include the Boolean value `is_power_user` in our model. See `Appendix Part B` for full results and computation.

A Days Worth of Data

To give some sense of what an average day's worth of data may look like, we show the output for `Day 4` in `Fig 2`. This is our first glimpse of some actual results from our data. We note the high variability and sentiment across some of this subreddits, the relatively high mean value, and the slight clustering at the extreme edges of the range of our sentiment (note that the smaller more transparent dots represent sentiment scores for specific observations). While this is only one day's worth of data, the results are illuminating and this is potentially a foreshadowing of what is to come.

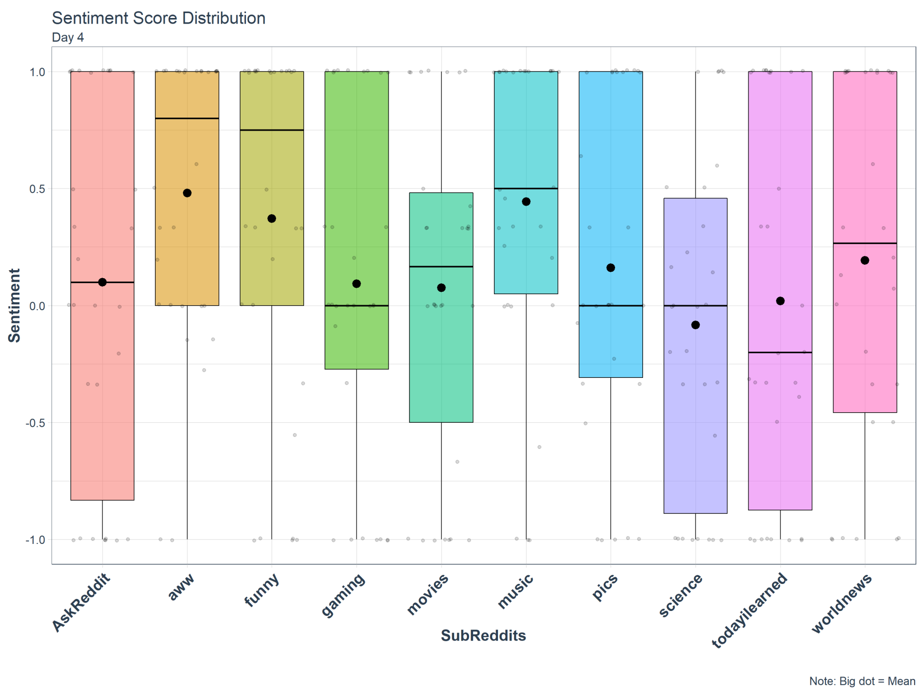


Fig 2: A single day's worth of data, Day 4 = Saturday

All the Data

`Fig 3` shows all sample data from all days simultaneously grouped by subreddit. We now see a greater trend start to emerge, the high variability in subreddits such as `funny` and `gaming`, the relatively high average sentiment score of `aww, music` and `pics` respectively, and once again, the noticeable clustering of observations often times at the extreme bounds of our response variable's range. We also notice that despite the noticeable amount of negative comments in each subreddit's respective sample, that no subreddit has an average sentiment score that is below zero.

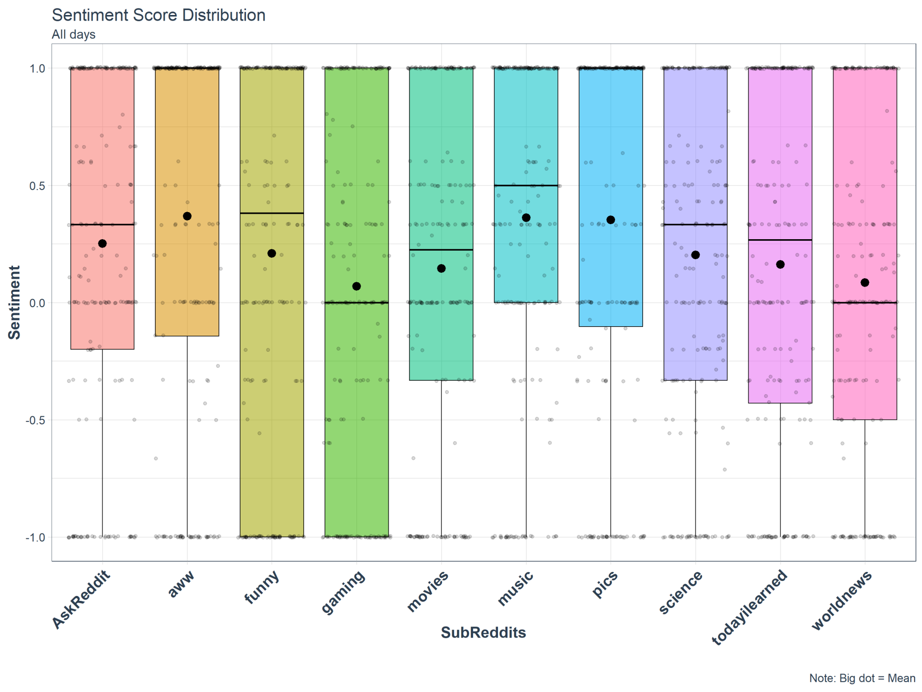


Fig 3: All sample data from all days

Our Final Model

In `Fig 4`, we can finally see the culmination of each day's average sentiment on one plot. We once again notice that the average and median sentiment of every subreddit, even the ones that might be considered the most negative, is still above zero. `Fig 4` presents essentially the same data as in `Fig 3` but in a much cleaner way. We noticed that a few of the subreddits (`gaming`, `science`, `todayilearned`, `worldnews`) do have days were the average sentiment is negative, but the overall trend of all the data is much more in the positive range of our sentiment scale.

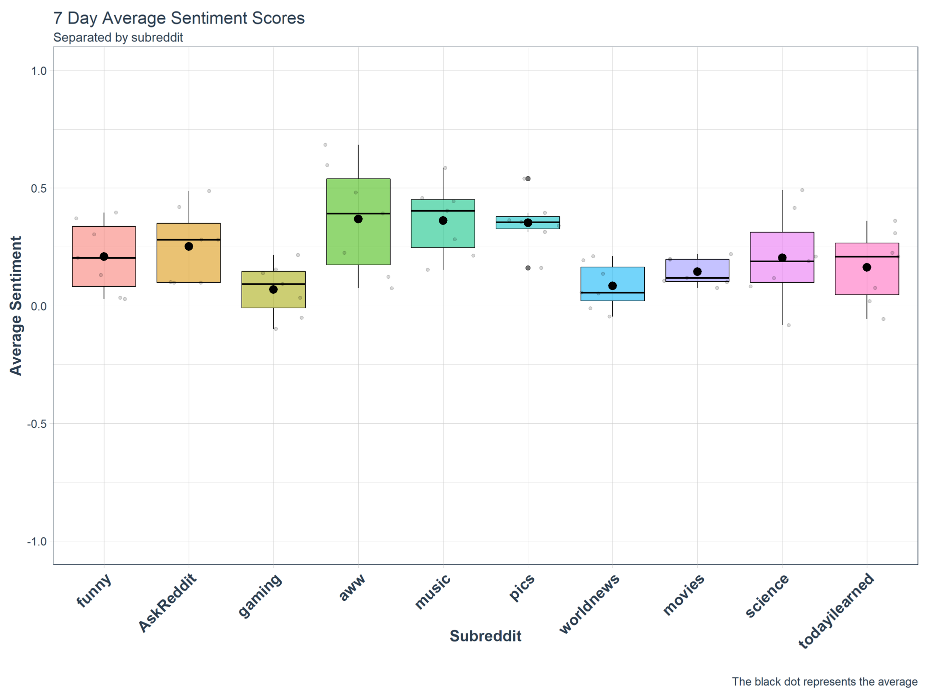


Fig 4: Average sentiment from each day

Performing ANOVA on this final data with the model:

$$SentimentScore \sim Factor(subreddit)$$

returns a p-value of 0.00081 (See `Appendix Part B`) indicating very strong significance against our null-hypothesis that a comment made by a user is unaffected by the subreddit in which the comment is placed. We therefore reject our null-hypothesis and conclude that the subreddit in which a comment is made is at the very least correlated with a change in sentiment.

```
# Basic
means_aov <- aov(average_sentiment ~ factor(subreddit), data = melt_means)
summary(means_aov)
```

```
##                   Df Sum Sq Mean Sq F value  Pr(>F)    
## factor(subreddit)  9 0.7784 0.08649   3.777 0.00081 ***
## Residuals         60 1.3739 0.02290                    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Model Adequacy

Checking the adequacy of our model in `Fig 5`, we notice that the residuals appear to be normally distributed and independent. There appears to be a slight tendency towards heteroscedasticity within the variance of our residuals, but this not always apparent in other random samplings, and could be due to statistical noise given it's very slight increase in range across the fitted values.

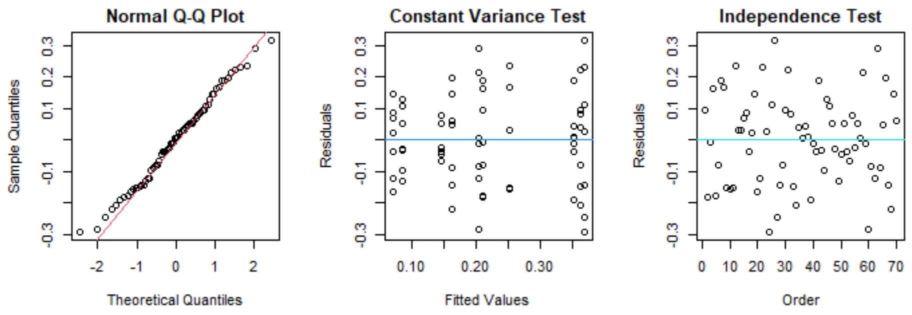
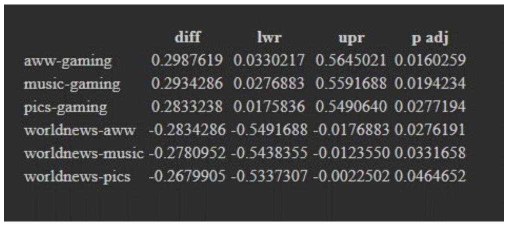


Fig 5: Model adequacy checks

Tukey Test



```
                      diff        lwr        upr     p adj
aww-gaming        0.2987619  0.0330217  0.5645021 0.0160259
music-gaming      0.2934286  0.0276883  0.5591688 0.0194234
pics-gaming       0.2833238  0.0175836  0.5490640 0.0277194
worldnews-aww    -0.2834286 -0.5491688 -0.0176883 0.0276191
worldnews-music  -0.2780952 -0.5438355 -0.0123550 0.0331658
worldnews-pics   -0.2679905 -0.5337307 -0.0022502 0.0464652
```

Fig 6: Significant results of the Tukey test

After deeming that our model was significant and valid, we conduct a `Tukey Test`, to determine where the significant ($p < 0.05$) difference in means specifically exists. We notice the common trend that the average sentiment between `aww, music` and `pics` is significantly different from the average sentiment both `gaming` and `worldnews` respectively.

Conclusion

This proved to be an interesting social experiment. Personally, I had an inkling that the difference in average sentiment would be significant across at least some of the top 10 subreddits. What I was not expecting, was that the average sentiment would be so high in most cases, or the polarization of sampled comments. Not only were there an extreme amount of negative comments sampled, but there was also a very high amount of positive comments sampled. However, I feel that even anecdotally, most people (myself included) feel that the majority of conversations that take place online are disproportionately negative. The data would seem to suggest otherwise. While there is a substantial amount of what can be considered negative sentiment online, there is overall a higher amount of positive sentiment in online conversations occurring on Reddit. For some reason, the negative sentiment seems to be more memorable. This leads to the interesting question: Why might this be the case? Why is the reputation of online comment sections so negative, when the data would seem to suggest otherwise? This could have something to do with the notion in Behavioural Economics that symmetric positive and negative rewards are often not viewed as equal by humans [9][10], but that is outside the scope of this project. Still, I found the results to be equal parts surprising, illuminating and thought-provoking.

Further Steps

This experiment could be improved in a number of ways. First, by including more variables in the model, such as comment time, karma ratio, post author etc.. Some more complex models were attempted, but when checking their model adequacy, their residuals were neither normally distributed nor the variance constant (See the `Additional Models` section at the end of `Appendix Part B`). It could also be beneficial to try increasing the size of the data and allow the data compilation script to run for more than one week. Another one of the hurdles of this experiment was the fact a comment's sentiment was not always able to be analysed and would sometimes return an `NA` value instead of a numerical score. This was due to the fact that a lot of comments in online settings use slang terminology, symbols and emoji's to convey meaning that our sentiment analysis package was not able to analyse. While these comments definitely convey sentiment, we were not able to extract the value with simple regular expressions. A more refined pre-processing stage for comment analysis prior to sentiment analysis would allow more meaning to be extracted from the data. We could also view the other eight emotions that the `syuzhet` package returns instead of just positive/negative sentiment. Overall, more advanced techniques in the NLP world could also be employed to detect non–obvious correlations and patterns within the data.

References

1. https://medium.com/global-editors-network/why-news-websites-are-closing-their-comments-sections-ea31139c469d
2. https://www.similarweb.com/top-websites/
3. https://medium.com/hacking-and-gonzo/how-reddit-ranking-algorithms-work-ef111e33d0d9
4. https://frontpagemetrics.com/top
5. https://en.wikipedia.org/wiki/Sentiment_ analysis
6. https://towardsdatascience.com/design-your-own-sentiment-score-e524308cf787
7. https://cran.r-project.org/web/packages/syuzhet/vignettes/syuzhet-vignette.html
8. https://praw.readthedocs.io/en/stable/
9. https://en.wikipedia.org/wiki/Daniel_ Kahneman
10. https://en.wikipedia.org/wiki/Behavioral_ economics

Appendix

Part A: Python

```
#
# Scrapes reddit for comment metadata and compiles into a data frame
#

import praw
import cred
import pandas as pd
import time
test_df = None


def build_reddit_data_frame():
    reddit = praw.Reddit(client_id=cred.c[0],
                         client_secret=cred.c[1],
                         user_agent=cred.c[2])

    # subreddits to scrape
    subreddits = ["funny",          # 1
                  "AskReddit",      # 2
                  "gaming",         # 3
                  "aww",            # 4
                  "music",          # 5
                  "pics",           # 6
                  "worldnews",      # 7
                  "movies",         # 8
                  "science",        # 9
                  "todayilearned"   # 10
                  ]

    # attributes from post, author, comments
    column_names = ["subreddit",
                    "post_name",
                    "author_name",
                    "time_created",
                    "num_comments",
                    "karma",
                    "karma_ratio",
                    "comment_author",
                    "comment",
                    "comment_time",
                    "comment_karma"]

    df = pd.DataFrame(columns=column_names)

    for s in subreddits:
        posts = reddit.subreddit(s).hot(limit=7) # grab top n posts
        for p in posts:

            # Skip over stickied posts
```

```python
            if p.stickied == False:
                submission = reddit.submission(id=p.id)  # get comments

                # removes all more comments/Continue buttons
                submission.comments.replace_more(limit=0)

                # for every comment in post
                for c in submission.comments.list():
                    try: # Exception for deleted comments
                        entry = {"subreddit": s,  # data frame row
                                 "post_name": submission.title,
                                 "post_author": submission.author.name,
                                 "time_created": submission.created_utc,
                                 "num_comments": submission.num_comments,
                                 "karma": submission.score,
                                 "karma_ratio": submission.upvote_ratio,
                                 "comment_author": c.author,
                                 "comment": c.body,
                                 "comment_time": c.created_utc,
                                 "comment_karma": c.score}
                        # df = df.concat(entry, ignore_index=True)
                        df.loc[len(df)]  = entry
                    except AttributeError:
                        continue

    # Output
    print("df shape: ", df.shape)
    today_date = time.strftime("%Y-%m-%d-%H-%M")
    base_path = r'''C:/Users/nilsm/Dropbox/UvicNotes/Semester13/
                    STAT453/FinalProject/reddit_comment_analyzer/data_dumps/'''

    extension = ".csv"
    full_path = base_path + "data_" + today_date + extension
    df.to_csv(full_path)
    global test_df
    test_df = df
    return df
```

```python
import reddit_tractor_beam as rtb


def main():
    reddit_data = rtb.build_reddit_data_frame() # Scrape reddit
    print("okay")


if __name__ == "__main__":
    main()
```

Part B: R

Libraries

```r
library(tidyr)
library(tibble)
library(stringr)
library(purrr)
library(dplyr)
library(ggplot2)
library(gridExtra)
library(ggridges)
library(tidyquant)
library(syuzhet)
```

Functions

```r
# Sentiment score of a given comment
calculate_sentiment_score <- function(comment){
    comment <- gsub("[\r\n]"," ", comment)
    comment <- gsub("[\r\n\n]","", comment)
    comment <- str_replace_all(comment, "[^[:alnum:]]", " ")
    sd <- get_nrc_sentiment(comment)
    num_score <- (sd$positive - sd$negative) / (sd$positive + sd$negative)
    return(round(num_score, 3))
}


# Bins a given sentiment score into 5 discrete bins
sentiment_bin <- function(s){
    if(-1 <= s & s <= -.5){
        return("Negative")
    }

    if(-.5 < s & s < -.1){
        return("Neg-Neutral")
    }

    if(-.1 <= s & s <= .1){
        return("Neutral")
    }

    if(.1 < s & s < .5){
        return("Pos-Neutral")
    }

    if(.5 <= s & s <= 1){
        return("Positive")
    }
```

```r
}

# Sample comment from given data frame and returns all attributes as a vector
sample_comments_and_sentiment_analysis <- function(df){
    blank_df <- data.frame(subreddit = character(),
                                post_name = character(),
                                karma = integer(),
                                karma_ratio = numeric(),
                                comment_author = character(),
                                comment = character(),
                                comment_karma = numeric(),
                                comment_score = numeric(),
                                id_tag = character())
    blank_df <- data.frame()
    for(i in 1:30){
        comment_score <- NaN
        while(is.nan(comment_score)){
            sample_row <- df[sample(1:nrow(df), 1, replace = FALSE), ]
            comment = sample_row[,6]
            comment_score <- calculate_sentiment_score(comment)
        }
        result = c(sample_row$subreddit,
                   sample_row$post_name,
                   sample_row$karma,
                   sample_row$karma_ratio,
                   sample_row$comment_author,
                   comment,
                   sample_row$comment_karma,
                   comment_score,
                   sample_row$id_tag)
        names(result) <- names(blank_df)
        blank_df = rbind(blank_df, result)
    }
    return(blank_df)
}

# Calculates the number of comments in a given post
calculate_number_of_comments <- function(df){
    comment_counter <- df %>%
        select(subreddit, post_name, comment) %>%
        group_by(subreddit, post_name) %>%
        mutate(comment_count = n()) %>%
        ungroup() %>%
        select(subreddit, post_name, comment_count) %>%
        distinct()
    return(comment_counter)
}

# Separate unique post list into vectorized list separated by subreddit
vectorize_posts <- function(df, sub){
    post_names_df <- df %>%
    filter(subreddit == sub) %>%
    arrange(desc(comment_count)) %>%
    select(post_name) %>%
    head(21)
    result_vector <- as.vector(post_names_df)
    return(result_vector)

}

# Function for binding data frames by day
data_frame_bind <- function(blank_df, master_df, n){
    for(i in n:(n + 2)){
        blank_df <- rbind(blank_df, master_df[[i]])
    }
    return(blank_df)
}


# Count unique posts
count_unique_posts <- function(df){
    df <- df %>%
    group_by(subreddit) %>%
    mutate(post_count = n()) %>%
    ungroup() %>%
    select(subreddit, post_count) %>%
    distinct() %>%
    arrange(desc(post_count))
    return(df)
}

# Outlier detector function
# Source: https://stackoverflow.com/questions/33524669/labeling-outliers-of-boxplots-in-r
is_outlier <- function(x) {
  return(x < quantile(x, 0.25) - 1.5 * IQR(x) | x > quantile(x, 0.75) + 1.5 * IQR(x))
}


# Build daily dataframe
sample_daily_dfs <- function(df, df_sampled_comments){
    for(i in 1:10){
        current_df <-
        df %>%
        filter(subreddit == subreddits[i]) %>%
        group_by(subreddit, post_name) %>%
        mutate(comment_count = n()) %>%
        ungroup() %>%
        filter(comment_count >= 30) %>%
        select(subreddit, post_name, karma,
               karma_ratio, comment_author,
               comment, comment_karma, id_tag)

        sampled_comments  <- sample_comments_and_sentiment_analysis(current_df)
        df_sampled_comments <- rbind(df_sampled_comments, sampled_comments)
    }
    return(df_sampled_comments)
```

```
}


# Feature engineering for sentiment binning and power user boolean
feature_engineering <- function(df){
    df <- df %>%
    mutate(is_power_user =
                ifelse(comment_author %in% power_users_vector, TRUE, FALSE),
           sentiment_bin = map(comment_score, sentiment_bin)) %>%
    rename(sentiment_score = comment_score)
    return(df)
}


# Plotting function
boxplot_plotting_with_dots <- function(df, sub = "All days"){
    result <- df %>%
    ggplot(aes(x = subreddit, y = sentiment_score, group = subreddit)) +
    geom_boxplot(data = df, aes(fill = subreddit, alpha = 0.4),
    notch = FALSE, show.legend = FALSE, size = .3,
    colour = "black") +
    stat_summary(fun = "mean") +
    geom_jitter(color = "black", fill = "white", size = 1, alpha = 0.15) +
    labs(
        title    = "Sentiment Score Distribution",
        subtitle = sub,
        caption  = "Note: Big dot = Mean" ,
        x        = "SubReddits",
        y        = "Sentiment"
    ) + theme_tq() +
    theme(axis.text.x   = element_text(angle = 45, vjust = 1,
                                      hjust=1, size = 12, face = "bold"),
          axis.title.x  = element_text(size = 12, face = "bold", vjust = 10),
          axis.title.y  = element_text(size = 12, face = "bold"))
    return(result)
}

# Conduct model adequacy checks
model_tests <- function(df, n){
    par(mfrow =c(1,3), pty = "s")
    res <- df$residuals
    fitted <- df$fitted.values

    # Normality
    qqnorm(res)
    qqline(res, col = 2)
    print(shapiro.test(res))

    # Constant Variance
    plot(fitted, res, ylab = "Residuals",xlab = "Fitted Values",
         main = "Constant Variance Test")
    abline(h = 0, col = 4)
```

```r
    # Independence
    plot(1:n,res, ylab = "Residuals",xlab = "Order", main = "Independence Test")
    abline(h = 0, col = 5)
}
```

Data Import

```r
# Day 1 (Wednesday)
data_day_01_1 <- read.csv(file = "../data_dumps/data_2022-03-16-00-07.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_01_2 <- read.csv(file = "../data_dumps/data_2022-03-16-08-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_01_3 <- read.csv(file = "../data_dumps/data_2022-03-16-16-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")

# Day 2 (Thursday)
data_day_02_1 <- read.csv(file = "../data_dumps/data_2022-03-17-00-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_02_2 <- read.csv(file = "../data_dumps/data_2022-03-17-08-07.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_02_3 <- read.csv(file = "../data_dumps/data_2022-03-17-16-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")

# Day 3 (Friday)
data_day_03_1 <- read.csv(file = "../data_dumps/data_2022-03-18-00-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_03_2 <- read.csv(file = "../data_dumps/data_2022-03-18-08-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_03_3 <- read.csv(file = "../data_dumps/data_2022-03-18-16-07.csv",
                          header = T, sep = ",", encoding = "UTF-8")

# Day 4 (Saturday)
data_day_04_1 <- read.csv(file = "../data_dumps/data_2022-03-19-00-07.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_04_2 <- read.csv(file = "../data_dumps/data_2022-03-19-08-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_04_3 <- read.csv(file = "../data_dumps/data_2022-03-19-16-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")

# Day 5 (Sunday)
data_day_05_1 <- read.csv(file = "../data_dumps/data_2022-03-20-00-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_05_2 <- read.csv(file = "../data_dumps/data_2022-03-20-08-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_05_3 <- read.csv(file = "../data_dumps/data_2022-03-20-16-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")

# Day 6 (Monday)
data_day_06_1 <- read.csv(file = "../data_dumps/data_2022-03-21-00-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_06_2 <- read.csv(file = "../data_dumps/data_2022-03-21-08-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_06_3 <- read.csv(file = "../data_dumps/data_2022-03-21-16-07.csv",
                          header = T, sep = ",", encoding = "UTF-8")

# Day 7 (Tuesday)
data_day_07_1 <- read.csv(file = "../data_dumps/data_2022-03-22-00-07.csv",
                          header = T, sep = ",", encoding = "UTF-8")
```

```
data_day_07_2 <- read.csv(file = "../data_dumps/data_2022-03-22-08-06.csv",
                          header = T, sep = ",", encoding = "UTF-8")
data_day_07_3 <- read.csv(file = "../data_dumps/data_2022-03-22-16-07.csv",
                          header = T, sep = ",", encoding = "UTF-8")

data_pulls <- list(data_day_01_1, data_day_01_2, data_day_01_3,
                   data_day_02_1, data_day_02_2, data_day_02_3,
                   data_day_03_1, data_day_03_2, data_day_03_3,
                   data_day_04_1, data_day_04_2, data_day_04_3,
                   data_day_05_1, data_day_05_2, data_day_05_3,
                   data_day_06_1, data_day_06_2, data_day_06_3,
                   data_day_07_1, data_day_07_2, data_day_07_3)


subreddits <- c(  "funny",          # 1
                  "AskReddit",      # 2
                  "gaming",         # 3
                  "aww",            # 4
                  "music",          # 5
                  "pics",           # 6
                  "worldnews",      # 7
                  "movies",         # 8
                  "science",        # 9
                  "todayilearned")  # 10



# Create identifier tags as new column Day #1 Pull #2
days <- c(rep(1,3), rep(2,3), rep(3,3), rep(4,3), rep(5,3), rep(6,3), rep(7,3))
times <- c(rep(c(1,2,3),7))
for(i in 1:21){
    data_pulls[[i]] <- data_pulls[[i]] %>%
        mutate(id_tag = paste0("D", days[i], "P", times[i]))
}

# Data frames binded by day
df_day_1 <- data_frame_bind(data.frame(), data_pulls, 1)
df_day_2 <- data_frame_bind(data.frame(), data_pulls, 4)
df_day_3 <- data_frame_bind(data.frame(), data_pulls, 7)
df_day_4 <- data_frame_bind(data.frame(), data_pulls, 10)
df_day_5 <- data_frame_bind(data.frame(), data_pulls, 13)
df_day_6 <- data_frame_bind(data.frame(), data_pulls, 16)
df_day_7 <- data_frame_bind(data.frame(), data_pulls, 19)


# Filter outpost with less than 30 comments
# Each data frame now has between 113-124 unique posts remaining
df_day_1_filtered <-
    df_day_1 %>%
    calculate_number_of_comments() %>%
    filter(comment_count >= 30)

df_day_2_filtered <-
```

```
    df_day_2 %>%
    calculate_number_of_comments() %>%
    filter(comment_count >= 30)

df_day_3_filtered <-
    df_day_3 %>%
    calculate_number_of_comments() %>%
    filter(comment_count >= 30)

df_day_4_filtered <-
    df_day_4 %>%
    calculate_number_of_comments() %>%
    filter(comment_count >= 30)

df_day_5_filtered <-
    df_day_5 %>%
    calculate_number_of_comments() %>%
    filter(comment_count >= 30)

df_day_6_filtered <-
    df_day_6 %>%
    calculate_number_of_comments() %>%
    filter(comment_count >= 30)

df_day_7_filtered <-
    df_day_7 %>%
    calculate_number_of_comments() %>%
    filter(comment_count >= 30)

# Number of posts remaining
# dim(df_day_1_filtered) # 124
# dim(df_day_2_filtered) # 118
# dim(df_day_3_filtered) # 113
# dim(df_day_4_filtered) # 117
# dim(df_day_5_filtered) # 117
# dim(df_day_6_filtered) # 118
# dim(df_day_7_filtered) # 119

# Minimum number of unique posts is 4 (music, Sunday)
unique_post_count_1 <- count_unique_posts(df_day_1_filtered)
unique_post_count_2 <- count_unique_posts(df_day_2_filtered)
unique_post_count_3 <- count_unique_posts(df_day_3_filtered)
unique_post_count_4 <- count_unique_posts(df_day_4_filtered)
unique_post_count_5 <- count_unique_posts(df_day_5_filtered)
unique_post_count_6 <- count_unique_posts(df_day_6_filtered)
unique_post_count_7 <- count_unique_posts(df_day_7_filtered)


# Master data frame of all data pulls
master_data <- data.frame()
for(i in 1:21){
    master_data <- rbind(master_data, data_pulls[[i]])
}
```

```r
# Count the number of unique comment authors
comment_author_count_df <-
    master_data %>%
    group_by(comment_author) %>%
    mutate(comment_author_count = n()) %>%
    ungroup() %>%
    select(comment_author, comment_author_count) %>%
    distinct() %>%
    mutate(power_user = ifelse(is_outlier(comment_author_count), TRUE, FALSE))  %>%
    arrange(desc(comment_author_count))

# Remove extraneous first row
comment_author_count_df <- comment_author_count_df[-1,]


# Top 10 most frequent commenters over the entire week
comment_author_count_df %>% head(10)
```

```
## # A tibble: 10 x 3
##    comment_author   comment_author_count power_user
##    <chr>                           <int> <lgl>
##  1 AutoModerator                     288 TRUE
##  2 FigsFanPhotos                     251 TRUE
##  3 geek2785                          187 TRUE
##  4 dsherwo                           120 TRUE
##  5 FjotraTheGodless                  112 TRUE
##  6 kim_bubbless                      102 TRUE
##  7 DukeVerde                         102 TRUE
##  8 bunkerbash                         97 TRUE
##  9 blackmachine312                    96 TRUE
## 10 MMMearsArt                         94 TRUE
```

```r
# We see that we have 137,074 distinct users
# comment_author_count_df %>% distinct()

# Isolate power users (6,342)
power_users <-
    comment_author_count_df %>%
    filter(power_user == TRUE) %>%
    select(comment_author)

total_author_count <- 137074
power_user_count <- 6342

# power_user_proportion (Approximately 4.63 % of commenters are power users in total data set)
power_user_count / total_author_count
```

```
## [1] 0.04626698
```

Sampling

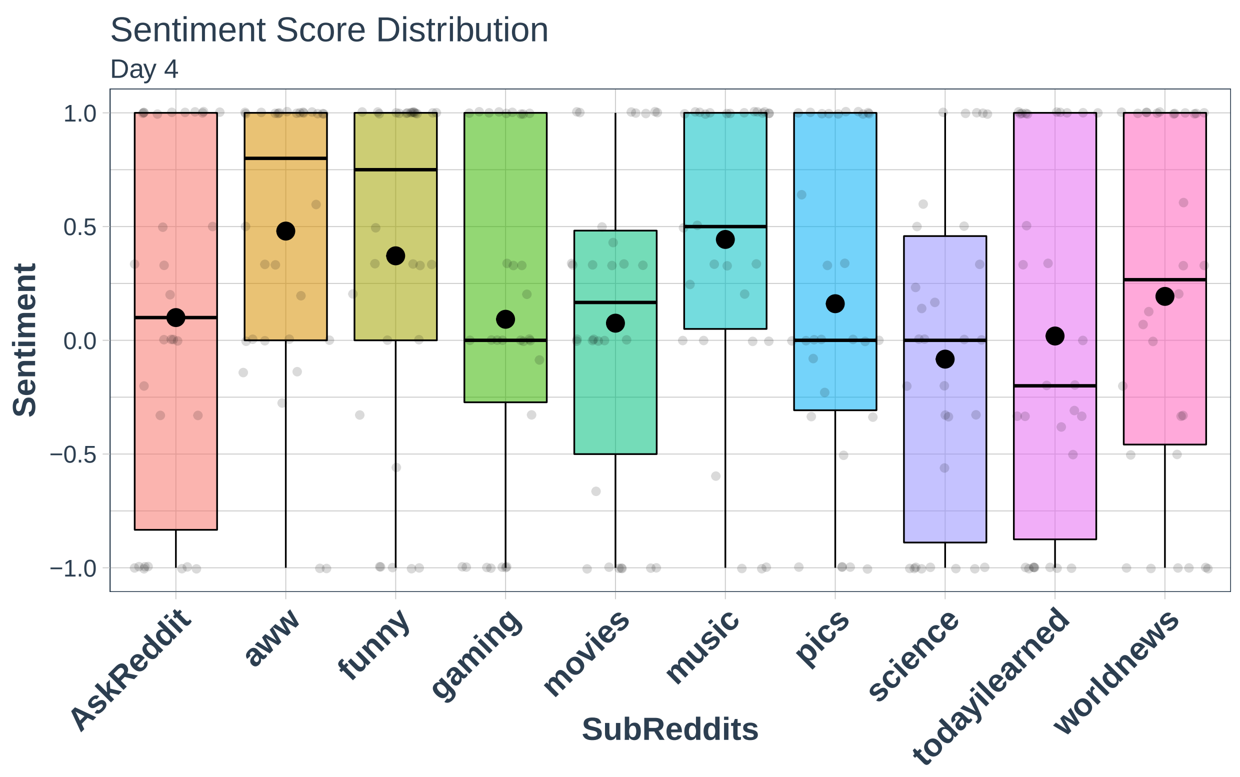
```r
set.seed(54321)
# Randomly select 30 comments from each subreddit:
df_sampled_comments <- data.frame(subreddit = character(),
                                post_name = character(),
                                karma = integer(),
                                karma_ratio = numeric(),
                                comment_author = character(),
                                comment = character(),
                                comment_karma = numeric(),
                                comment_score = numeric(),
                                id_tag = character())

# Randomly sample 30 comments from each subReddit
df_day_1_comments <- sample_daily_dfs(df_day_1, df_sampled_comments)
df_day_2_comments <- sample_daily_dfs(df_day_2, df_sampled_comments)
df_day_3_comments <- sample_daily_dfs(df_day_3, df_sampled_comments)
df_day_4_comments <- sample_daily_dfs(df_day_4, df_sampled_comments)
df_day_5_comments <- sample_daily_dfs(df_day_5, df_sampled_comments)
df_day_6_comments <- sample_daily_dfs(df_day_6, df_sampled_comments)
df_day_7_comments <- sample_daily_dfs(df_day_7, df_sampled_comments)

# Rename columns
names(df_day_1_comments) <- names(df_sampled_comments)
names(df_day_2_comments) <- names(df_sampled_comments)
names(df_day_3_comments) <- names(df_sampled_comments)
names(df_day_4_comments) <- names(df_sampled_comments)
names(df_day_5_comments) <- names(df_sampled_comments)
names(df_day_6_comments) <- names(df_sampled_comments)
names(df_day_7_comments) <- names(df_sampled_comments)
```

Feature Engineering

```r
# Vectorize list of power users
power_users_vector <- power_users[[1]] %>% as.vector()

# Sentiment bins and power user Boolean columns
df_day_1_comments <- df_day_1_comments %>% feature_engineering()
df_day_2_comments <- df_day_2_comments %>% feature_engineering()
df_day_3_comments <- df_day_3_comments %>% feature_engineering()
df_day_4_comments <- df_day_4_comments %>% feature_engineering()
df_day_5_comments <- df_day_5_comments %>% feature_engineering()
df_day_6_comments <- df_day_6_comments %>% feature_engineering()
df_day_7_comments <- df_day_7_comments %>% feature_engineering()
```

Plotting by day

```r
p1 <- boxplot_plotting_with_dots(df_day_1_comments, "Day 1")
p2 <- boxplot_plotting_with_dots(df_day_2_comments, "Day 2")
p3 <- boxplot_plotting_with_dots(df_day_3_comments, "Day 3")
p4 <- boxplot_plotting_with_dots(df_day_4_comments, "Day 4")
p5 <- boxplot_plotting_with_dots(df_day_5_comments, "Day 5")
p6 <- boxplot_plotting_with_dots(df_day_6_comments, "Day 6")
p7 <- boxplot_plotting_with_dots(df_day_7_comments, "Day 7")

# p1
# p2
# p3
# p5
# p6
# p7
p4
```



Note: Big dot = Mean

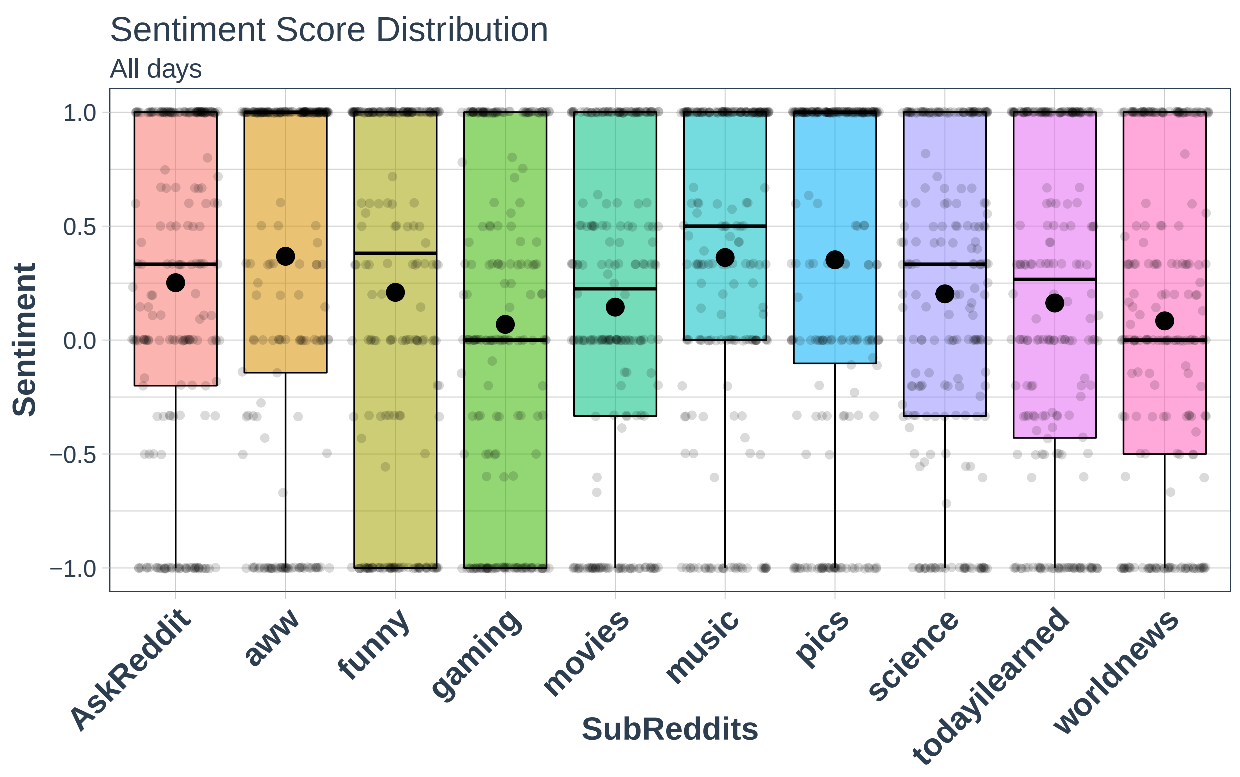
Compile replications into data frame

```
df_comments_all_days <- list(df_day_1_comments, df_day_2_comments,
                             df_day_3_comments, df_day_4_comments,
                             df_day_5_comments, df_day_6_comments,
                             df_day_7_comments)
```

All observations and one data from

```
all_days_df <- data.frame()
for(i in 1:7){
  all_days_df <- rbind(all_days_df, df_comments_all_days[[i]])
}

all_data_plot <- all_days_df %>% boxplot_plotting_with_dots()
all_data_plot
```

```
## Warning: Removed 10 rows containing missing values (geom_segment).
```



Note: Big dot = Mean

Check if the power users are significantly different in sentiment from normal users

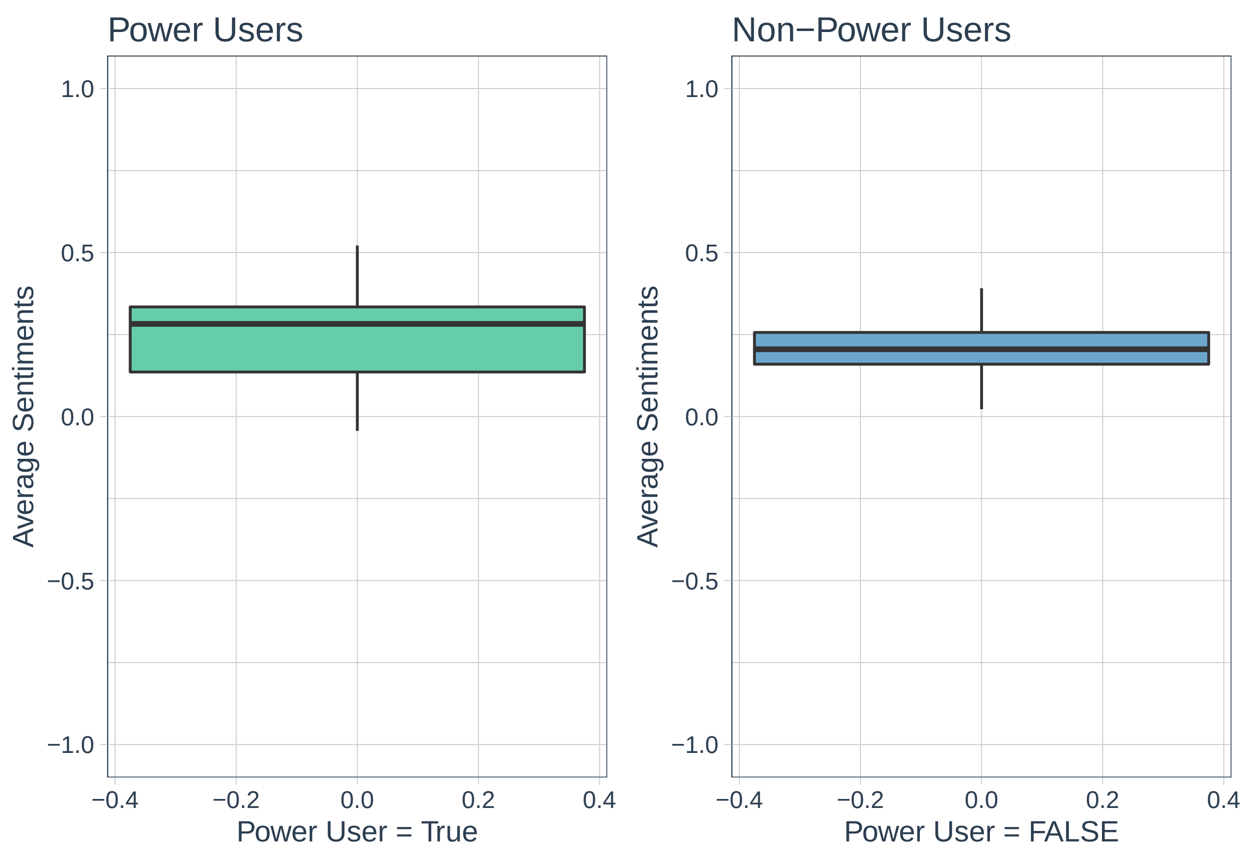
```
# Calculate the number of power users in the sample data set
# all_days_df # 2100 obvs

# 2,048 Distinct commenters
# 460 comments are from power users
# all_days_df %>%
#   select(comment_author, is_power_user) %>%
#   distinct() %>%
#   filter(is_power_user == TRUE)

460/2048 # Approximately 22.5% of comments in the sample data set are from power users
```

```
## [1] 0.2246094
```

```
# Determine if there is a noticeable difference between power users and nonpower users in terms of sent


all_days_df_power_users_true <-
all_days_df %>%
  filter(is_power_user == TRUE)

all_days_df_power_users_false <-
all_days_df %>%
  filter(is_power_user == FALSE)


average_sentiments_true <-
  all_days_df_power_users_true %>%
  select(subreddit, sentiment_score) %>%
  group_by(subreddit) %>%
  mutate(average_sentiment = mean(sentiment_score)) %>%
  ungroup() %>%
  select(subreddit, average_sentiment) %>%
  distinct()

average_sentiments_false <-
  all_days_df_power_users_false %>%
  select(subreddit, sentiment_score) %>%
  group_by(subreddit) %>%
  mutate(average_sentiment = mean(sentiment_score)) %>%
  ungroup() %>%
  select(subreddit, average_sentiment) %>%
  distinct()
```

Graph power users and nonpower user box plots

```
pu_true_plot <-
  average_sentiments_true %>%
  ggplot(aes(y = average_sentiment)) +
  geom_boxplot(fill = "Aquamarine3") +
  labs(
    title    = "Power Users",
    x        = "Power User = True",
    y        = "Average Sentiments"
  ) + theme_tq() + ylim(-1,1)

pu_false_plot <-
average_sentiments_false %>%
  ggplot(aes(y = average_sentiment)) +
  geom_boxplot(fill = "Skyblue3") +
  labs(
    title    = "Non-Power Users",
    x        = "Power User = FALSE",
    y        = "Average Sentiments",
    ) + theme_tq() + ylim(-1,1)

power_user_double_plot <- grid.arrange(pu_true_plot, pu_false_plot, ncol = 2)
```



Power users t-test

```
# Vectorize average sentiment scores for each subReddit
true_sentiments <- average_sentiments_true[[2]]
false_sentiments <- average_sentiments_false[[2]]


# Variance is equal
var.test(true_sentiments, false_sentiments)
```

```
##
##  F test to compare two variances
##
## data:  true_sentiments and false_sentiments
## F = 2.5979, num df = 9, denom df = 9, p-value = 0.1712
## alternative hypothesis: true ratio of variances is not equal to 1
## 95 percent confidence interval:
##   0.6452789 10.4590867
## sample estimates:
## ratio of variances
##           2.597889
```

```r
# Test normality ( Both are normal )
shapiro.test(true_sentiments)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  true_sentiments
## W = 0.9662, p-value = 0.8536
```

```r
shapiro.test(false_sentiments)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  false_sentiments
## W = 0.9604, p-value = 0.7904
```

```r
# We do not reject the null hypothesis, power users do not matter
t.test(true_sentiments, false_sentiments, var.equal = TRUE)
```

```
##
##  Two Sample t-test
##
## data:  true_sentiments and false_sentiments
## t = 0.58087, df = 18, p-value = 0.5685
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1018259  0.1796492
## sample estimates:
## mean of x mean of y
## 0.2522518 0.2133402
```

Creating seven-day average

```r
funny_means <- c()
AskReddit_means <- c()
gaming_means <- c()
aww_means <- c()
music_means <- c()
pics_means <- c()
worldnews_means <- c()
movies_means <- c()
science_means <- c()
todayilearned_means <- c()

# List of means sentiment for each sub Reddit over
means_list <- list(funny_means, AskReddit_means,
                   gaming_means, aww_means, music_means,
                   pics_means, worldnews_means, movies_means,
                   science_means, todayilearned_means)


# Isolate each sub Reddit in each data frame
# Calculate the average sentiment
# and store in the corresponding means_list
for(i in 1:7){
  curr_df <- df_comments_all_days[[i]]
  for(j in 1:10){
    curr_df_2 <- curr_df %>%
      filter(subreddit == subreddits[j]) %>%
      mutate(average_sentiment = mean(sentiment_score)) %>%
      select(average_sentiment) %>%
      distinct()
      means_list[[j]][i] <- curr_df_2[[1]]
  }
}

# Name columns
names(means_list) <- subreddits

# Converted data frame
means_df <- means_list %>% as.data.frame()
```

Normality check

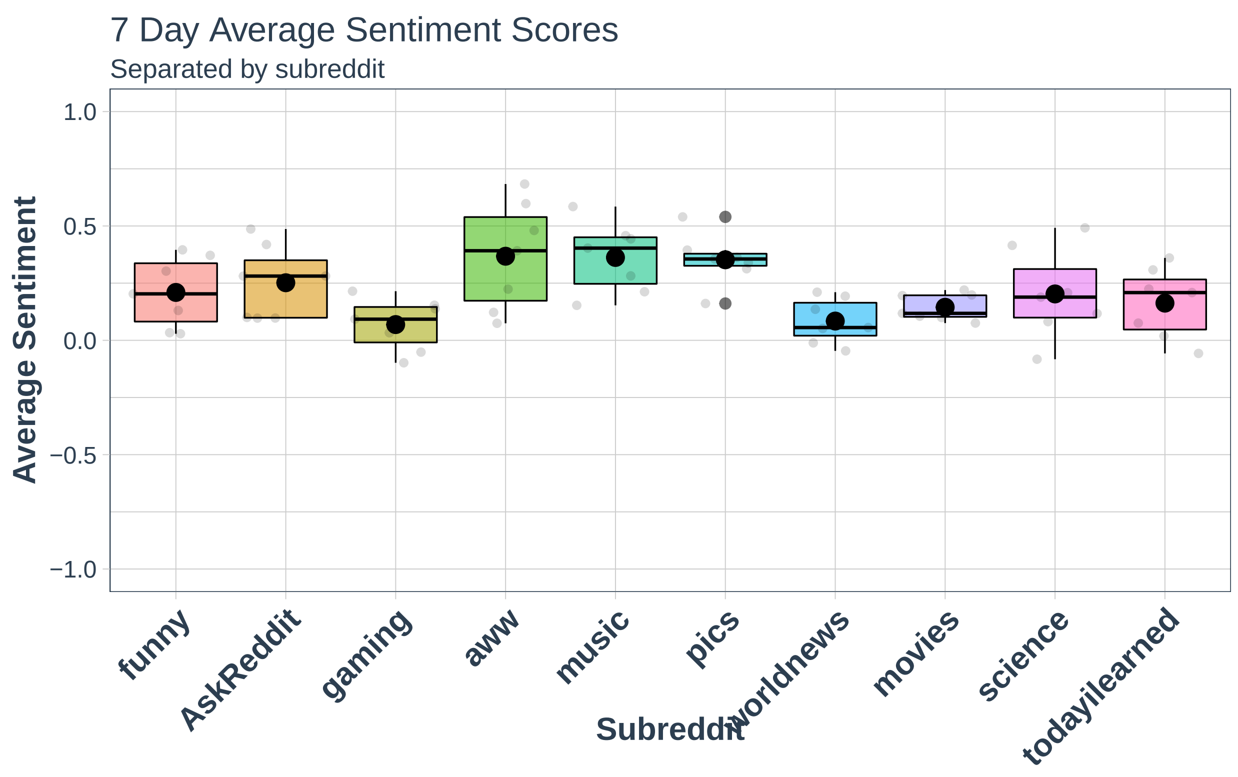
```
# Test every column for normality (all columns pass)
for(i in 1:7){
  s <- shapiro.test(means_df[,i])
  print(s)
}
```

```
##
##  Shapiro-Wilk normality test
##
## data:  means_df[, i]
## W = 0.90492, p-value = 0.3618
##
##
##  Shapiro-Wilk normality test
##
## data:  means_df[, i]
## W = 0.85975, p-value = 0.1506
##
##
##  Shapiro-Wilk normality test
##
## data:  means_df[, i]
## W = 0.9545, p-value = 0.7704
##
```

```
##
##  Shapiro-Wilk normality test
##
## data:  means_df[, i]
## W = 0.9414, p-value = 0.6513
##
##
##  Shapiro-Wilk normality test
##
## data:  means_df[, i]
## W = 0.96033, p-value = 0.8216
##
##
##  Shapiro-Wilk normality test
##
## data:  means_df[, i]
## W = 0.92562, p-value = 0.5143
##
##
##  Shapiro-Wilk normality test
##
## data:  means_df[, i]
## W = 0.93488, p-value = 0.5931
```

Master Visualization

```
melt_means <- means_df %>%
  data.table::melt(quietly = TRUE) %>%
  rename(subreddit = variable, average_sentiment = value)

final_plot <- melt_means %>%
  ggplot(aes(x = subreddit, y = average_sentiment, group = subreddit)) +
  geom_boxplot(aes(fill = subreddit, alpha = 0.4), show.legend = FALSE,
               size = .3, colour = "black") +
  stat_summary(fun = "mean") +
  geom_jitter(color = "black", fill = "white", size = 1, alpha = 0.15) +
  labs(
    title    = "7 Day Average Sentiment Scores",
    subtitle = "Separated by subreddit",
    caption  = "The black dot represents the average",
    x        = "Subreddit",
    y        = "Average Sentiment",
  ) +
  theme_tq() +
  theme(axis.text.x   = element_text(angle = 45, vjust = 1, hjust=1, size = 12, face = "bold"),
        axis.title.x  = element_text(size = 12, face = "bold", vjust = 10),
        axis.title.y  = element_text(size = 12, face = "bold")) +
  ylim(-1,1)
final_plot
```



The black dot represents the average

ANOVA

```
# Basic
means_aov <- aov(average_sentiment ~ factor(subreddit), data = melt_means)
summary(means_aov)
```

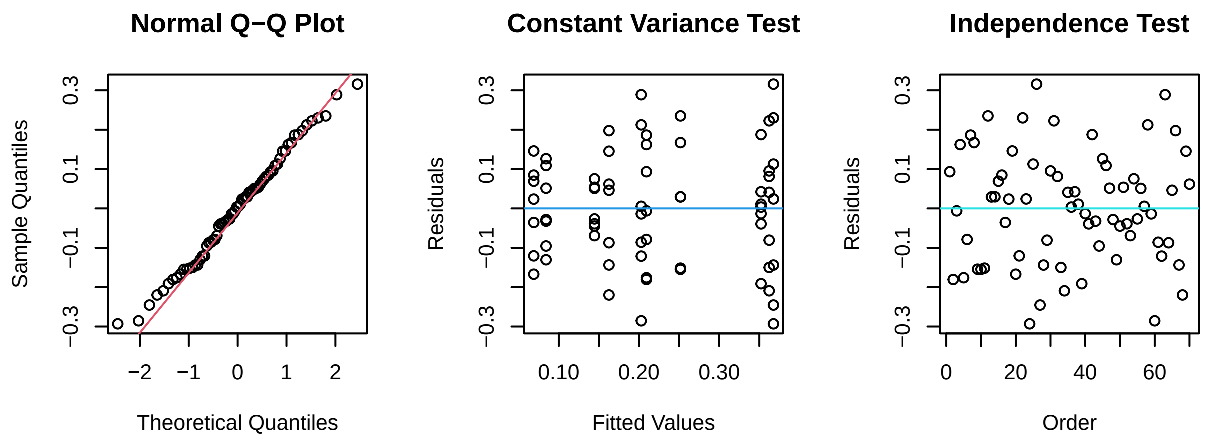
```
##                   Df Sum Sq Mean Sq F value  Pr(>F)
## factor(subreddit)  9 0.7784 0.08649   3.777 0.00081 ***
## Residuals         60 1.3739 0.02290
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
shapiro.test(means_aov$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  means_aov$residuals
## W = 0.9891, p-value = 0.8075
```

```
model_adequacy <- model_tests(means_aov, 70)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.9891, p-value = 0.8075
```



Tukey Test

```
tukey_test <- TukeyHSD(means_aov)
tukey_df <- tukey_test$`factor(subreddit)` %>%
  as.data.frame()

tukey_df_filter <-
  tukey_df %>%
  filter(`p adj` <= 0.05)

final_tukey_table <- tukey_df_filter %>% knitr::kable(align = c("c", "c", "c", "c"))
final_tukey_table
```

	diff	lwr	upr	p adj
aww-gaming	0.2987619	0.0330217	0.5645021	0.0160259
music-gaming	0.2934286	0.0276883	0.5591688	0.0194234
pics-gaming	0.2833238	0.0175836	0.5490640	0.0277194
worldnews-aww	-0.2834286	-0.5491688	-0.0176883	0.0276191
worldnews-music	-0.2780952	-0.5438355	-0.0123550	0.0331658
worldnews-pics	-0.2679905	-0.5337307	-0.0022502	0.0464652

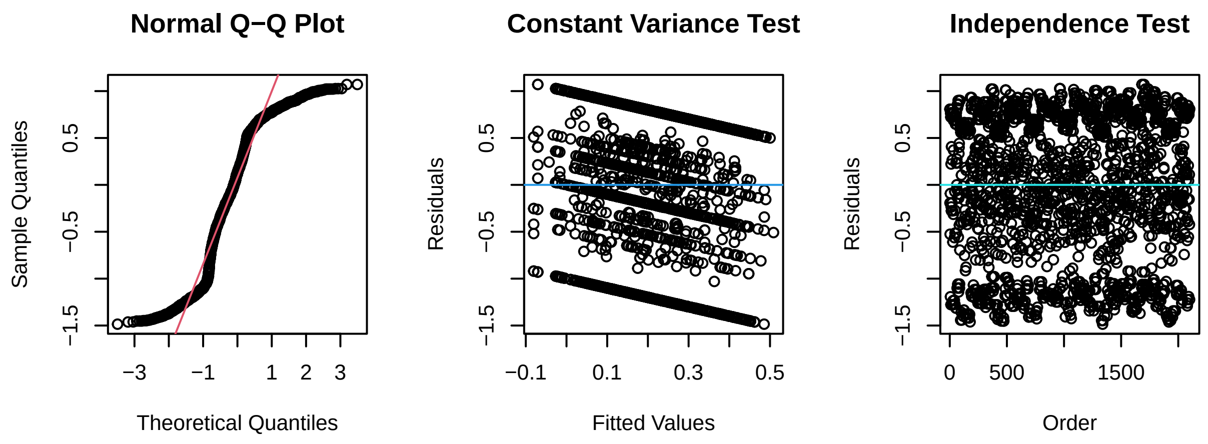
Additional models

```
# Blocking
all_days_df_aov <- aov(sentiment_score ~
                       factor(subreddit) +
                       factor(id_tag) +
                       factor(is_power_user),
                       data = all_days_df)
summary(all_days_df_aov)
```

```
##                          Df Sum Sq Mean Sq F value   Pr(>F)
## factor(subreddit)         9   23.4  2.5947   4.497 7.15e-06 ***
## factor(id_tag)           20    9.3  0.4672   0.810    0.704
## factor(is_power_user)     1    0.5  0.5293   0.917    0.338
## Residuals              2069 1193.9  0.5770
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Model not valid
model_tests(all_days_df_aov, 2100)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.90327, p-value < 2.2e-16
```

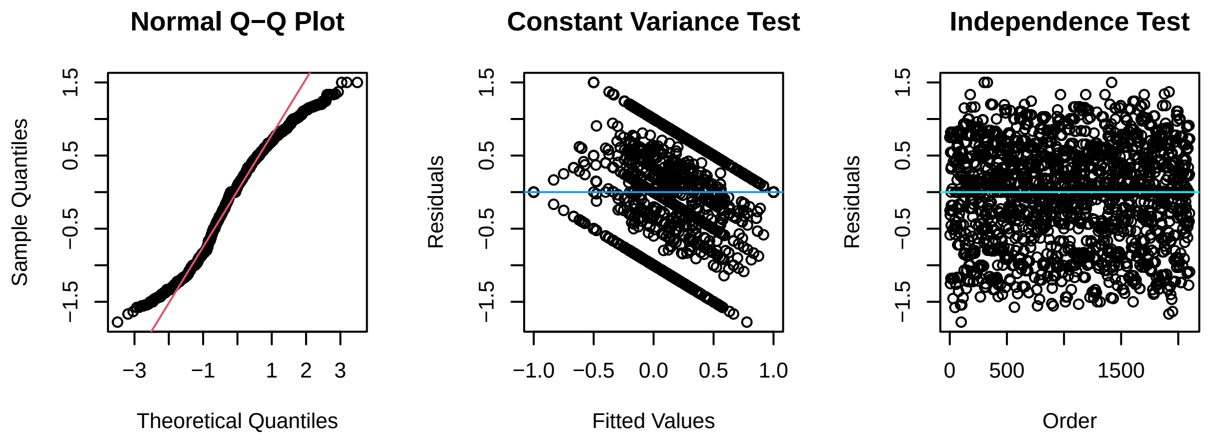


```r
# Blocking And interactions
all_days_df_aov_interactions <-
                        aov(sentiment_score ~
                        factor(subreddit) *
                        factor(id_tag) *
                        factor(is_power_user),
                        data = all_days_df)
summary(all_days_df_aov_interactions)
```

```
##                                                      Df Sum Sq Mean Sq
## factor(subreddit)                                     9   23.4  2.5947
## factor(id_tag)                                       20    9.3  0.4672
## factor(is_power_user)                                 1    0.5  0.5293
## factor(subreddit):factor(id_tag)                    179  132.1  0.7380
## factor(subreddit):factor(is_power_user)               9    9.0  1.0031
## factor(id_tag):factor(is_power_user)                 20    8.5  0.4245
## factor(subreddit):factor(id_tag):factor(is_power_user) 152   73.1  0.4811
## Residuals                                          1709  971.1  0.5682
##                                                    F value   Pr(>F)
## factor(subreddit)                                    4.566 5.69e-06 ***
## factor(id_tag)                                       0.822  0.68818
## factor(is_power_user)                                0.931  0.33463
## factor(subreddit):factor(id_tag)                     1.299  0.00681 **
## factor(subreddit):factor(is_power_user)              1.765  0.07015 .
## factor(id_tag):factor(is_power_user)                 0.747  0.77893
## factor(subreddit):factor(id_tag):factor(is_power_user)   0.847  0.90745
## Residuals
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Model not valid
model_tests(all_days_df_aov_interactions, 2100)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  res
## W = 0.9707, p-value < 2.2e-16
```


Normal Q−Q Plot
Sample Quantiles
Theoretical Quantiles
Constant Variance Test
Residuals
Fitted Values
Independence Test
Residuals
Order