Project

Jacob Aronoff

11/12/2017

Make the queries

Getting the data

I decided to get started with my project by only looking about posts relating to a movie, later in the project I want to get into comments and sentiment analysis.

In constructing the query, I ran into a couple of problems. At first the query I was trying to run was returning all NULL values, I then changed the query from doing it all at once, to doing the queries individually. Making this change also allowed me to make it so that each query would only get data up until the movie's release date.

```
if(exists("movieData", inherits = T)) {
    # Pass
  } else {
    movieData = getMovieData(movies)
  }
```

```
## Warning in strptime(x, "%Y/%m/%d"): unknown timezone 'zone/tz/2017c.1.0/
## zoneinfo/America/New_York'
```

```
## [1] "Getting data for Allied"
## [1] 2
## [1] "Getting data for Ben-Hur"
## [1] 4
## [1] "Getting data for The BFG"
## [1] 6
## [1] "Getting data for Deepwater Horizon"
## [1] 8
## [1] "Getting data for The Finest Hours"
## [1] 10
## [1] "Getting data for Ghostbusters"
## [1] 12
## [1] "Getting data for Gods of Egypt"
## [1] 14
## [1] "Getting data for The Great Wall"
## [1] 16
## [1] "Getting data for The Huntsman: Winter's War"
## [1] 18
## [1] "Getting data for Live by Night"
## [1] 20
## [1] "Getting data for Monster Trucks"
## [1] 2
## [1] "Getting data for Captain America: Civil War"
## [1] 4
## [1] "Getting data for Rogue One: A Star Wars Story"
## [1] 6
```

```
## Warning: NAs introduced by coercion
```

```
## [1] "Getting data for Finding Dory"
## [1] 8
## [1] "Getting data for Zootopia"
## [1] 10
## [1] "Getting data for The Jungle Book"
## [1] 12
## [1] "Getting data for The Secret Life of Pets"
## [1] 14
## [1] "Getting data for Batman v Superman: Dawn of Justice"
## [1] 16
## [1] "Getting data for Fantastic Beasts and Where to Find Them"
## [1] 18
## [1] "Getting data for Suicide Squad"
## [1] 20
```

```
movieQueries = list()
for(i in 1:nrow(movieData))
{
  movieQueries <- append(movieQueries, moviePostQuery(movieData[i,]))
}

if(exists("bigQueryData", inherits = T)) {
    # Pass
} else if(file.exists("bigQueryData.csv")) {
    bigQueryData <- read.csv("bigQueryData.csv", header = TRUE)
    class(bigQueryData$created_utc) <- class(Sys.time())
  } else {
    bigQueryData <- data.frame(created_utc=numeric(0),
                               subreddit=character(0),
                               author=character(0),
                               domain=character(0),
                               num_comments=numeric(0),
                               score=numeric(0),
                               ups=numeric(0),
                               downs=numeric(0),
                               title=character(0),
                               selftext=character(0),
                               id=character(0),
                               gilded=numeric(0),
                               movie=character(0),
                               budget=numeric(0),
                               revenue=numeric(0),
                               margin=numeric(0),
                               stringsAsFactors=FALSE)
    for(i in 1:length(movieQueries))
    {
      post.data <- query_exec(movieQueries[[i]][1], project = project, useLegacySql = FALSE, max_pages =
      post.data$movie = movieData[i,]$movie
      post.data$budget = movieData[i,]$budget
      post.data$revenue = movieData[i,]$revenue
      post.data$margin = 100 * (movieData[i,]$revenue - movieData[i,]$budget) / movieData[i,]$revenue
      print(paste("The response has",nrow(post.data), "rows"))
      for(x in 1:nrow(post.data))
      {
```

```
        bigQueryData[nrow(bigQueryData)+1,] = post.data[x,]
      }
    }

    write.csv(bigQueryData, file = "bigQueryData.csv", na="NA")
  }
```

Creating an Analytics Base Table

```
checkDataQuality(data= bigQueryData, out.file.num="dq_num.csv", out.file.cat= "dq_cat.csv")
```

```
## Check for numeric variables completed // Results saved to disk // Time difference of 0.2464502 secs
## Check for categorical variables completed // Results saved to disk // Time difference of 1.154331 se
```

```
numericalQuality <- read.csv("dq_num.csv", header = TRUE)
categoricalQuality <- read.csv("dq_cat.csv", header = TRUE)

print(numericalQuality)
```

```
##              X non.missing missing missing.percent unique         mean
## 1            X       35902       0            0.00  35902     17951.50
## 2 num_comments       35902       0            0.00    457        14.41
## 3        score       35902       0            0.00    751        34.69
## 4          ups       35449     453            1.26    746        33.99
## 5        downs       35449     453            1.26      2         0.00
## 6       gilded       35902       0            0.00      3         0.00
## 7       budget       35902       0            0.00     15 140157066.46
## 8      revenue       35902       0            0.00     20 624115196.90
## 9       margin       35902       0            0.00     20        68.61
##           min          p1          p5          p10          p25
## 1        1.00      360.01     1796.05     3591.10      8976.25
## 2        0.00        0.00        0.00        0.00         0.00
## 3        0.00        0.00        0.00        0.00         1.00
## 4        0.00        0.00        0.00        0.00         1.00
## 5        0.00        0.00        0.00        0.00         0.00
## 6        0.00        0.00        0.00        0.00         0.00
## 7 30000000.00 30000000.00 30000000.00 30000000.00 75000000.00
## 8 22678555.00 64493915.00 119520023.00 119520023.00 295212467.00
## 9     -376.22      -93.82        9.68       28.88        76.53
##            p50          p75          p90          p95          p99
## 1     17951.50     26926.75     32311.90     34106.95     35542.99
## 2         1.00         4.00        18.00        38.00       177.97
## 3         1.00         5.00        32.00        87.00       503.99
## 4         1.00         5.00        32.00        88.00       501.56
## 5         0.00         0.00         0.00         0.00         0.00
## 6         0.00         0.00         0.00         0.00         0.00
## 7 175000000.00 180000000.00 250000000.00 250000000.00 250000000.00
## 8 745600054.00 966550600.00 1153304495.00 1153304495.00 1153304495.00
## 9        78.32        89.84        89.84        89.84        89.84
##             max
## 1      35902.00
## 2      10389.00
## 3      13129.00
```

```
## 4       9424.00
## 5          0.00
## 6          2.00
## 7  250000000.00
## 8 1153304495.00
## 9         91.43
```

```
print(categoricalQuality)
```

```
##           X n.non.miss n.miss n.miss.percent n.unique
## 1 subreddit      35871     31           0.09     3825
## 2    author      35902      0           0.00    13300
## 3    domain      35902      0           0.00     4611
## 4     title      35902      0           0.00    29359
## 5  selftext      11425  24477          68.18     3864
## 6        id      35902      0           0.00    35856
## 7     movie      35902      0           0.00       20
##                                         cat_1 freq_1         cat_2
## 1                                      movies   5527  DC_Cinematic
## 2                                   [deleted]   6537  ell_computer
## 3                                 youtube.com   6682     imgur.com
## 4 Rogue One: A Star Wars Story Trailer (Official)     61 Suicide Squad
## 5                                   [deleted]   5172     [removed]
## 6                                      3zfoiz      3        3zfp94
## 7                                Ghostbusters   8937 Suicide Squad
##   freq_2
## 1   1974
## 2    635
## 3   1780
## 4     50
## 5   2334
## 6      3
## 7   8773
##
## 1
## 2
## 3
## 4
## 5 Watch... Batman v Superman: Dawn of Justice... Full... Movie... Free... Streaming... Online... wit
## 6
## 7
##   freq_3
## 1   1150
## 2    382
## 3   1701
## 4     49
## 5     12
## 6      2
## 7   3961
##
## 1
## 2
## 3
## 4
## 5 **Goals: FUN, Community, and Dank Memes**\n\n**Information:**\nTired of all the mil-sim bullshit? T
```

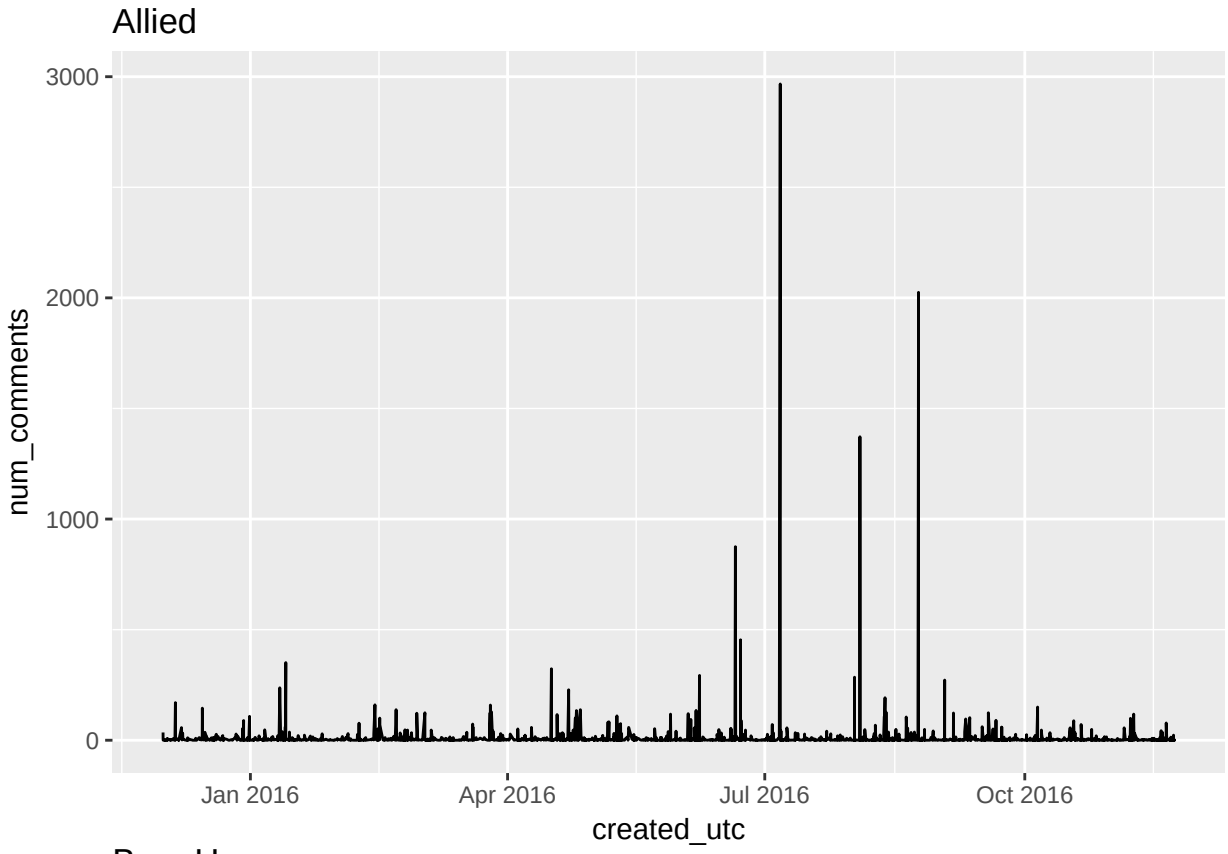
```
## 6
## 7
##   freq_4
## 1    988
## 2    381
## 3   1646
## 4     44
## 5      6
## 6      2
## 7   2549
##
## 1
## 2
## 3
## 4
## 5 **Goals: FUN, Community, and Dank Memes**\n\n**Information:**\nTired of all the mil-sim bullshit? T
## 6
## 7
##   freq_5
## 1    756
## 2    369
## 3    905
## 4     41
## 5      5
## 6      2
## 7   2340
##
## 1
## 2
## 3
## 4
## 5 **Goals: FUN, Community, and Dank Memes**\n\n**Information:**\nTired of all the mil-sim bullshit? T
## 6
## 7
##   freq_6                                                cat_7 freq_7
## 1    652                                            DCcomics    534
## 2    365                                             ImaBlue    329
## 3    727                                           reddit.com    638
## 4     40 Rogue One: A Star Wars Story Trailer #2 (Official)     36
## 5      4                                      Maria Williams      3
## 6      2                                              423apy      2
## 7   1499                                      The Great Wall   1333
##
## 1
## 2
## 3
## 4
## 5 \n\nPutlocker. Watch. Gods of Egypt Online. Free. Movie. Streaming. STREAM..FREE. 1080p Watch... Go
## 6
## 7
##   freq_8
## 1    418
## 2    313
## 3    440
```

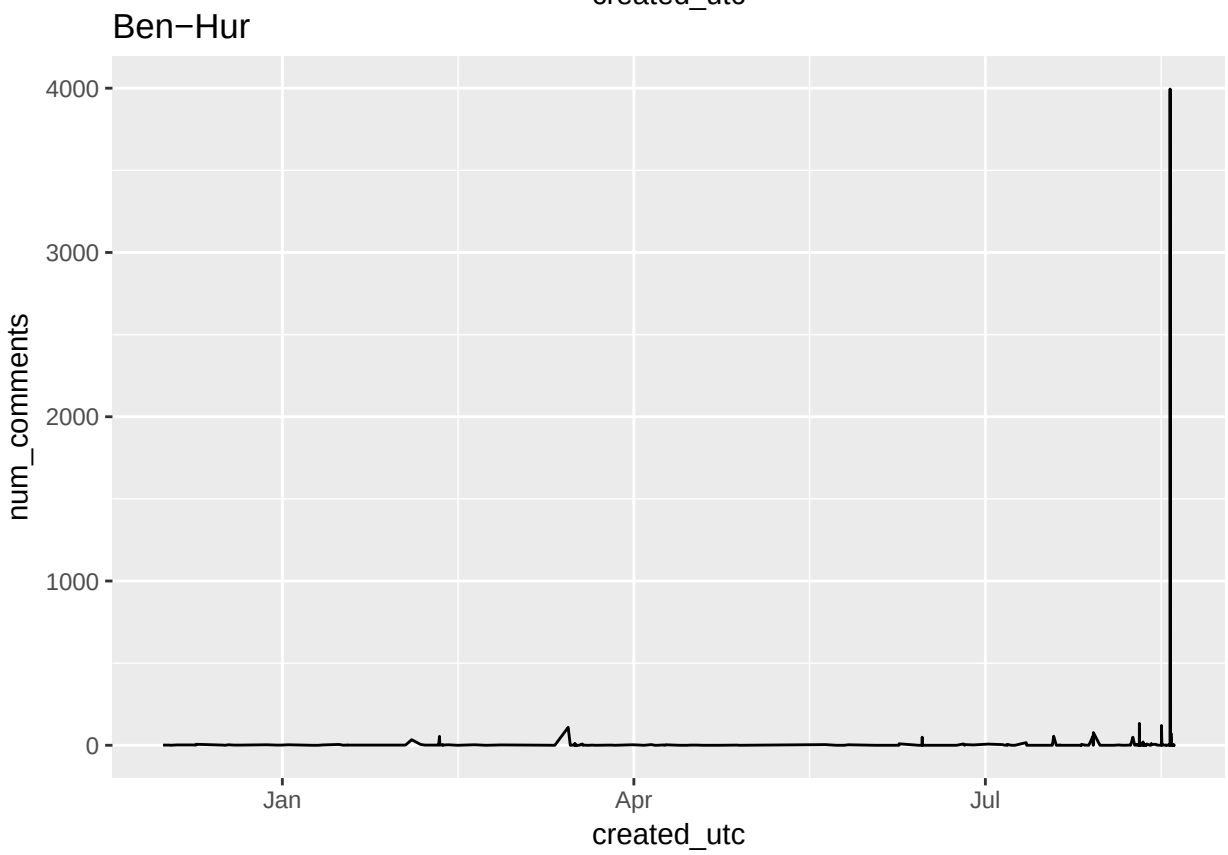
```
## 4     32
## 5      2
## 6      2
## 7   1188
##
## 1
## 2
## 3
## 4
## 5 [Amazon] (https://www.amazon.com/b/ref=as_li_ss_tl?ie=UTF8&amp;node=14102689011&amp;linkCode=sl2&a
## 6
## 7
##   freq_9
## 1    404
## 2    264
## 3    406
## 4     32
## 5      2
## 6      2
## 7   1023
##
## 1
## 2
## 3
## 4
## 5 [Dat Trailer Though](http://m.youtube.com/watch?v=CmRih_VtVAs)\n\nI think the thing I've always fo
## 6
## 7
##   freq_10
## 1     382
## 2     235
## 3     385
## 4      30
## 5       2
## 6       2
## 7     814
```

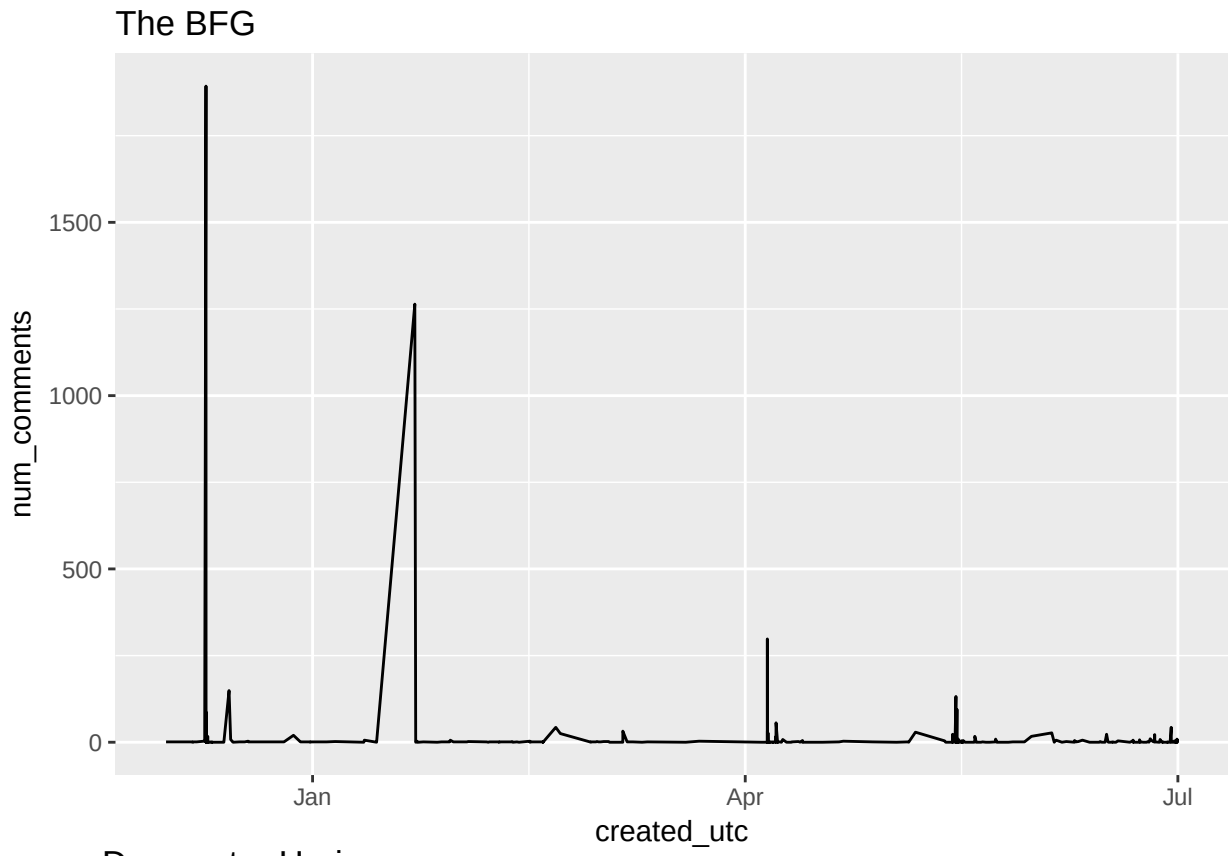
Exploring Data

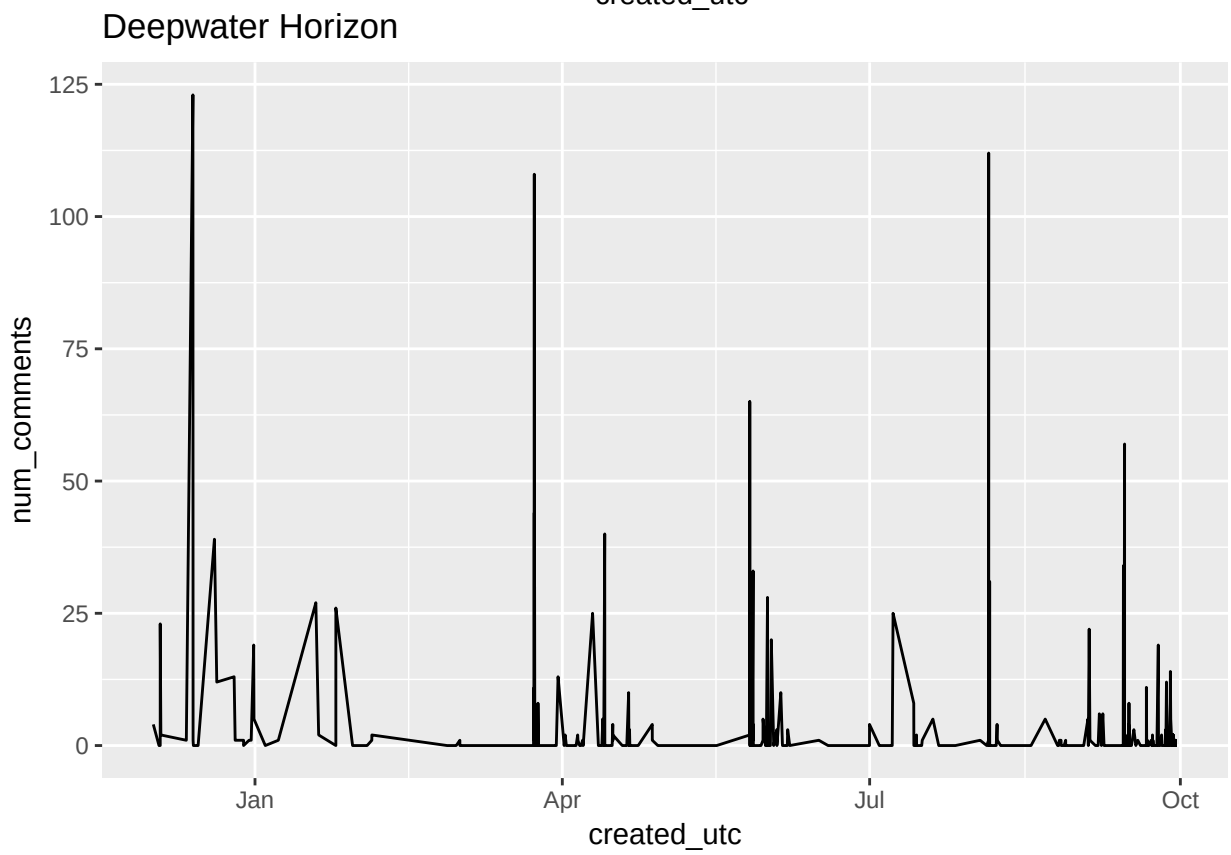
In exploring my data I wanted to just look at basic patterns in the data, and it looks like there are some general trends in a few of the fields. I'll be able to do some better analysis later, when I implement Plotly so I can easily change around the data.

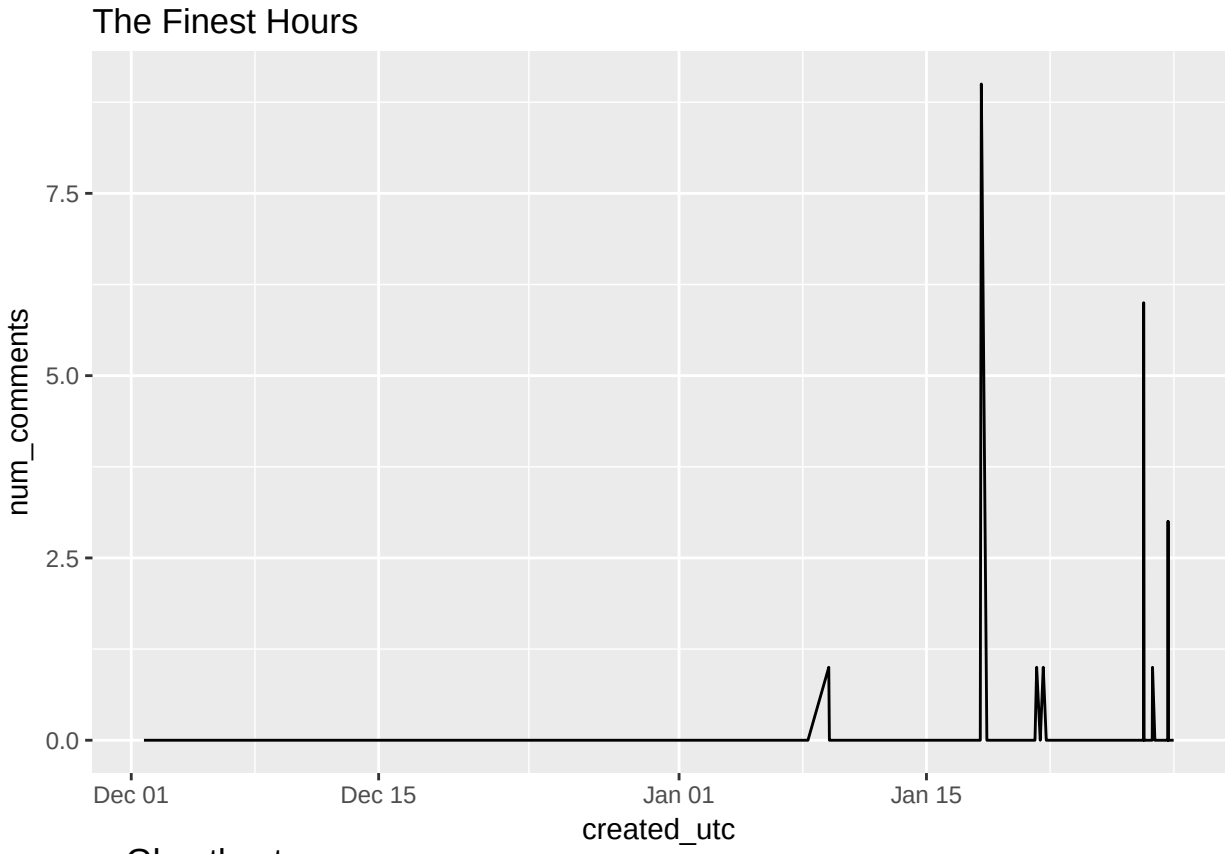
```
for(movie in movies)
{
  p <- ggplot(bigQueryData[bigQueryData$movie == movie,], aes(x = created_utc, y = num_comments)) + geo
  print(p)
}
```

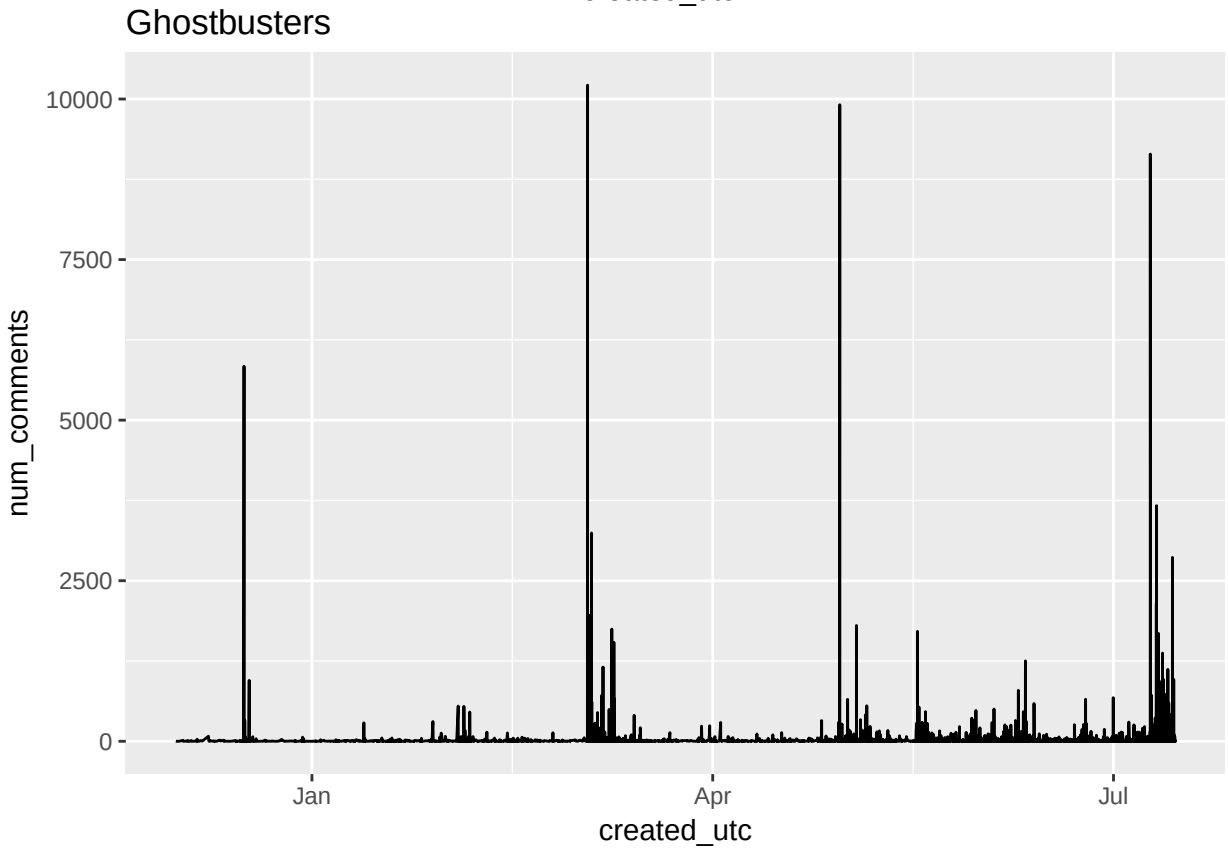

Allied
3000
2000
1000
0
num_comments
Jan 2016
Apr 2016
Jul 2016
Oct 2016
created_utc

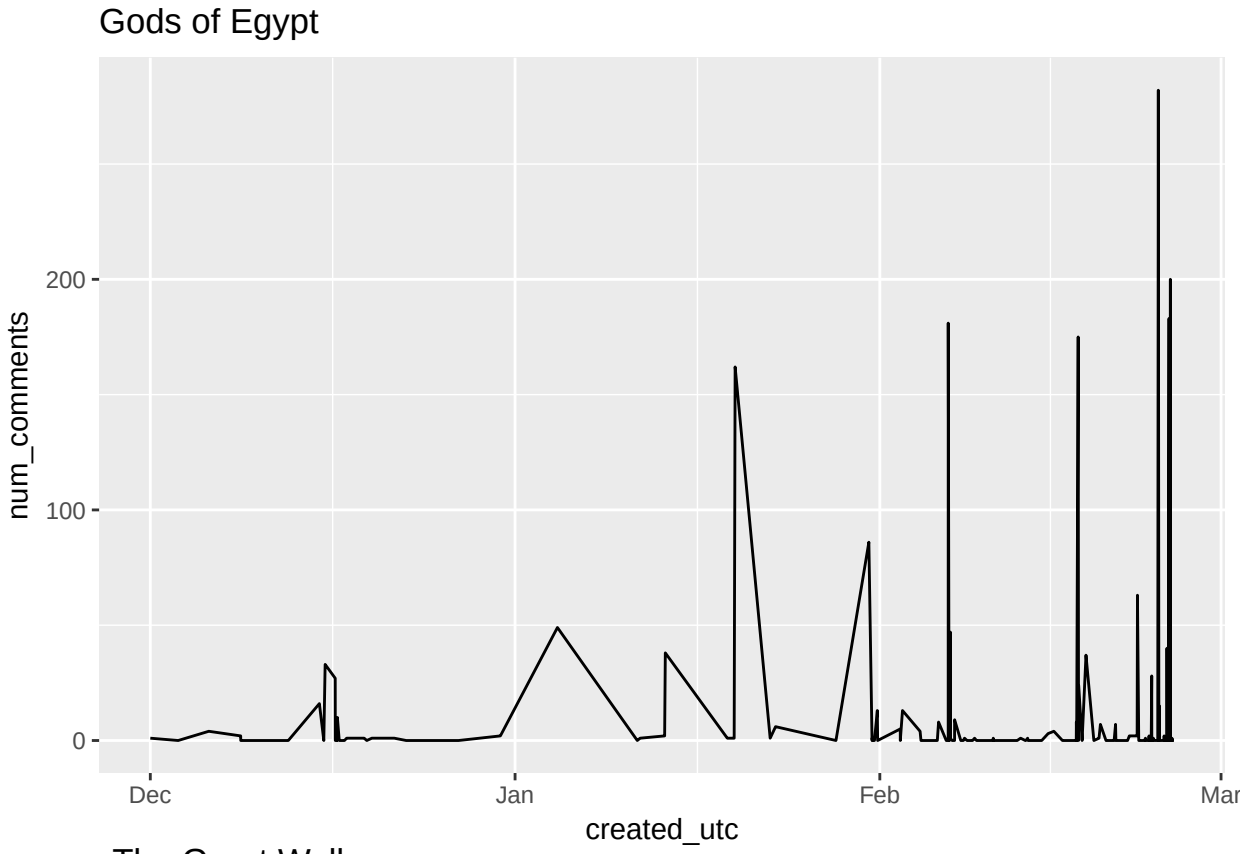

Ben−Hur
4000
3000
2000
1000
0
num_comments
Jan
Apr
Jul
created_utc

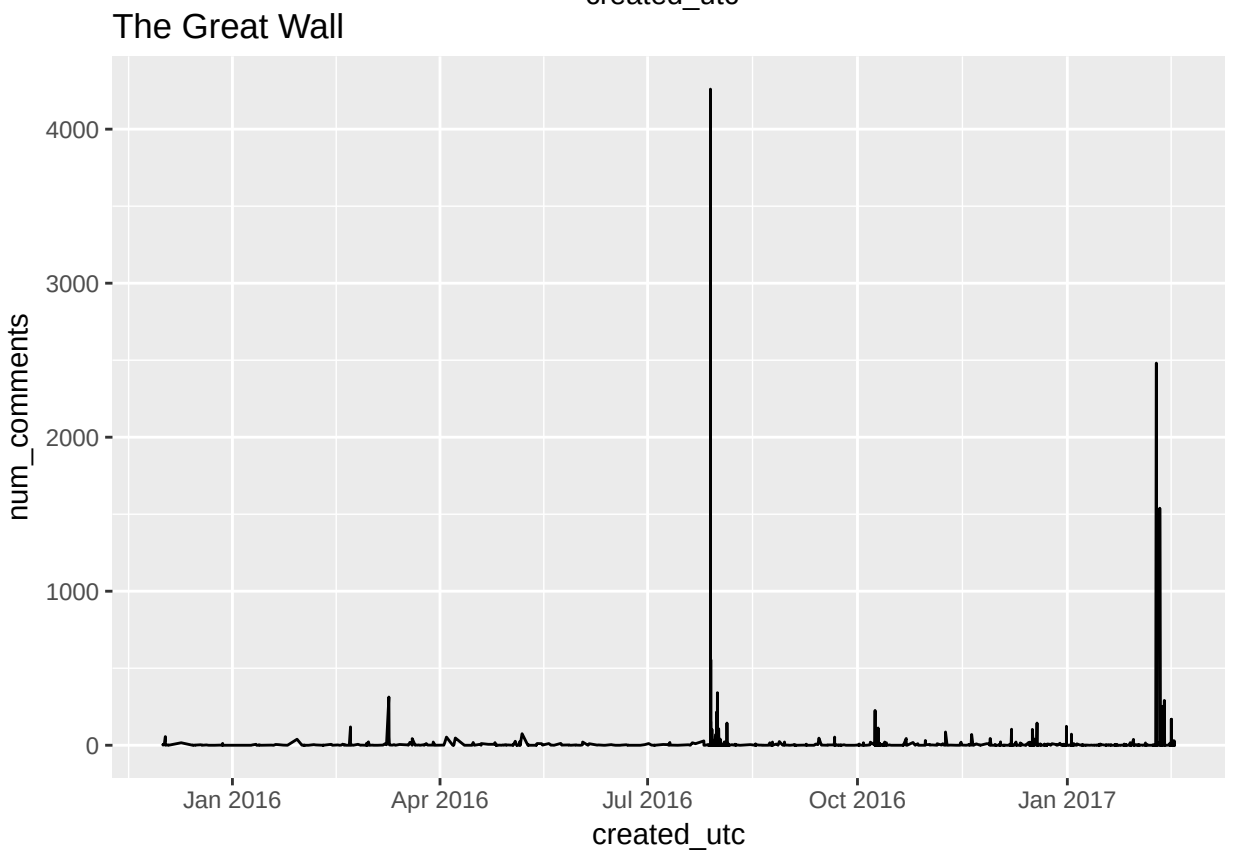

The BFG
num_comments
1500
1000
500
0
Jan
Apr
Jul
created_utc

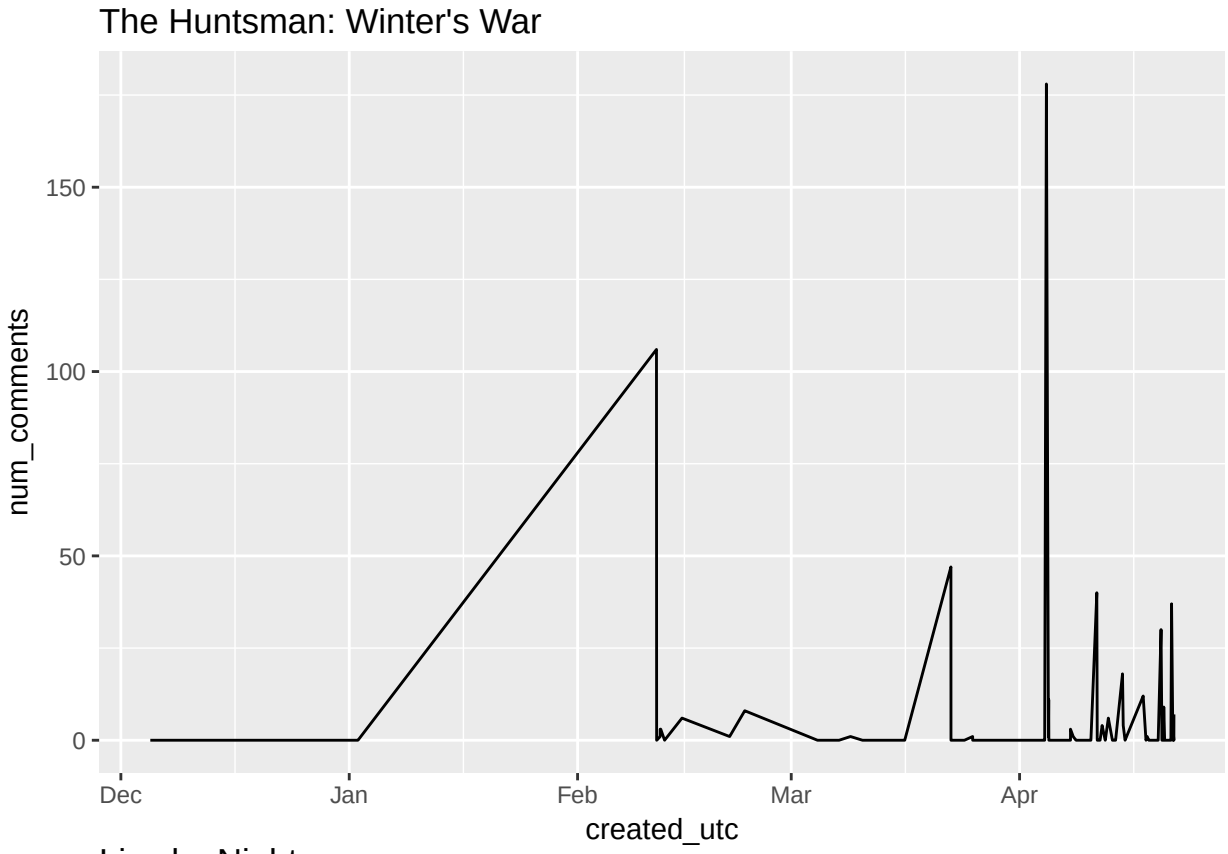

Deepwater Horizon
num_comments
125
100
75
50
25
0
Jan
Apr
Jul
Oct
created_utc

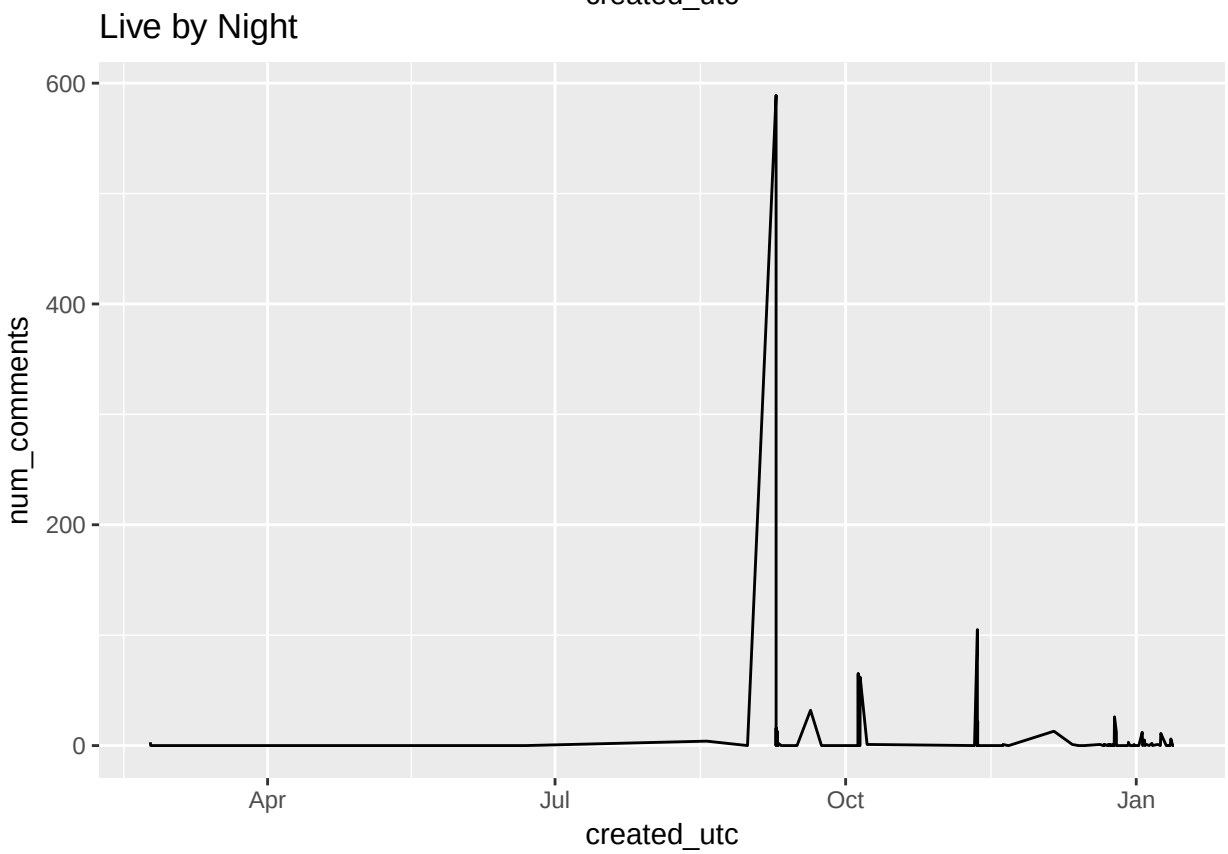

The Finest Hours
num_comments
7.5
5.0
2.5
0.0
Dec 01
Dec 15
Jan 01
Jan 15
created_utc

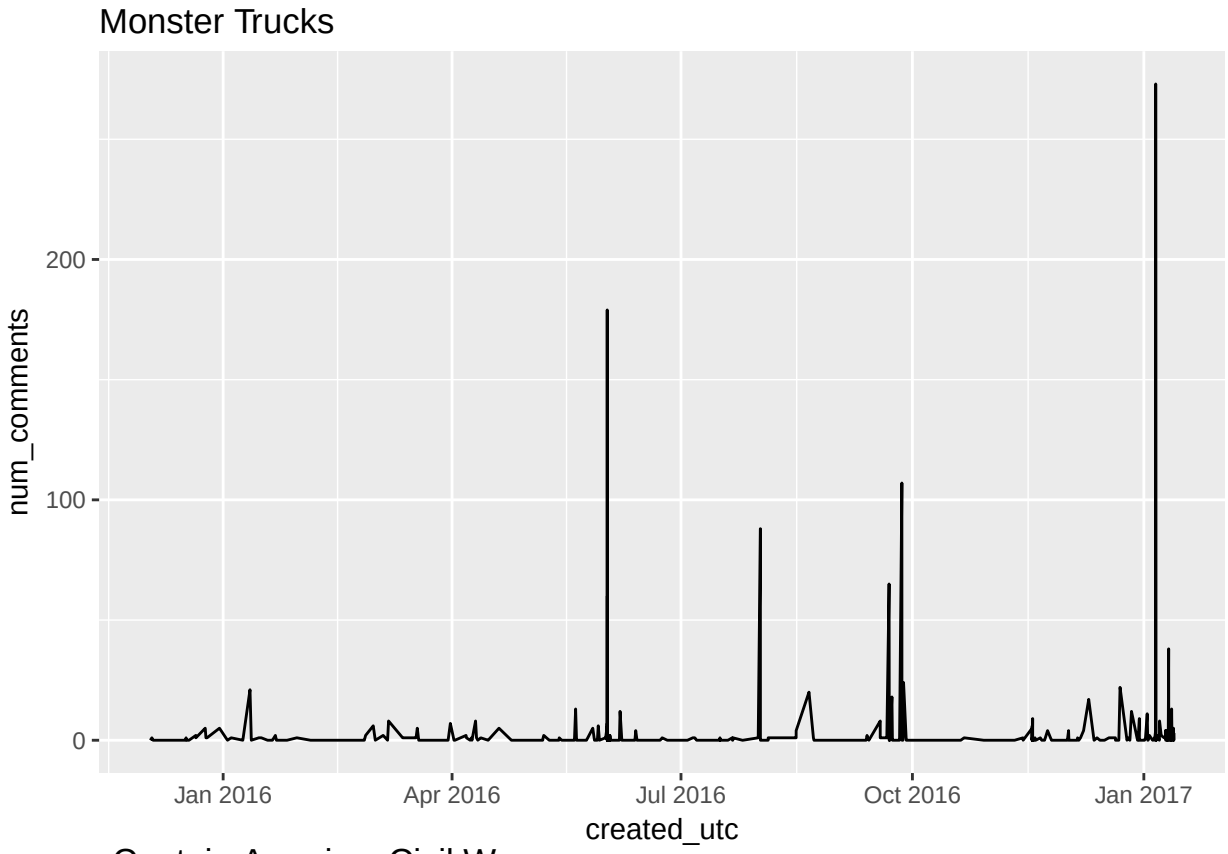

Ghostbusters
num_comments
10000
7500
5000
2500
0
Jan
Apr
Jul
created_utc

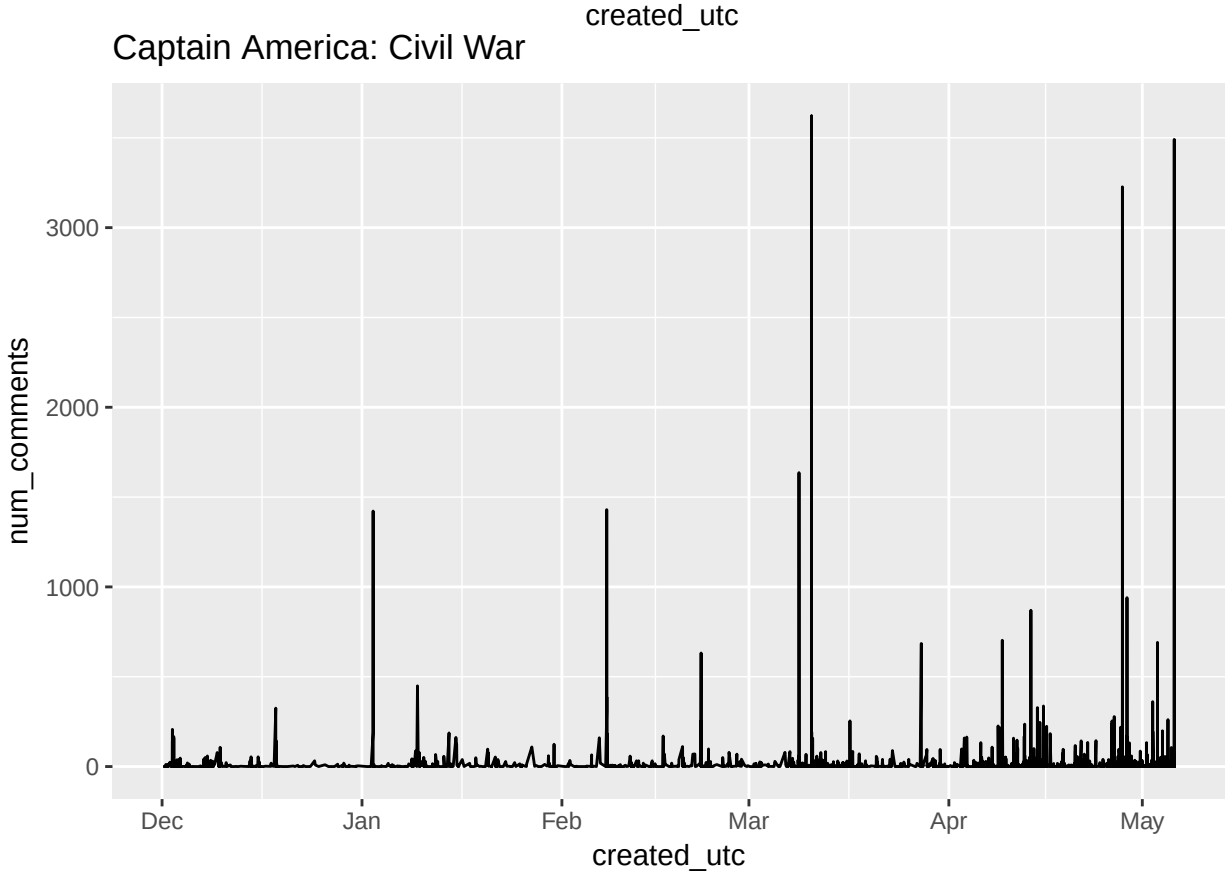

Gods of Egypt
num_comments
200
100
0
Dec
Jan
Feb
Mar
created_utc

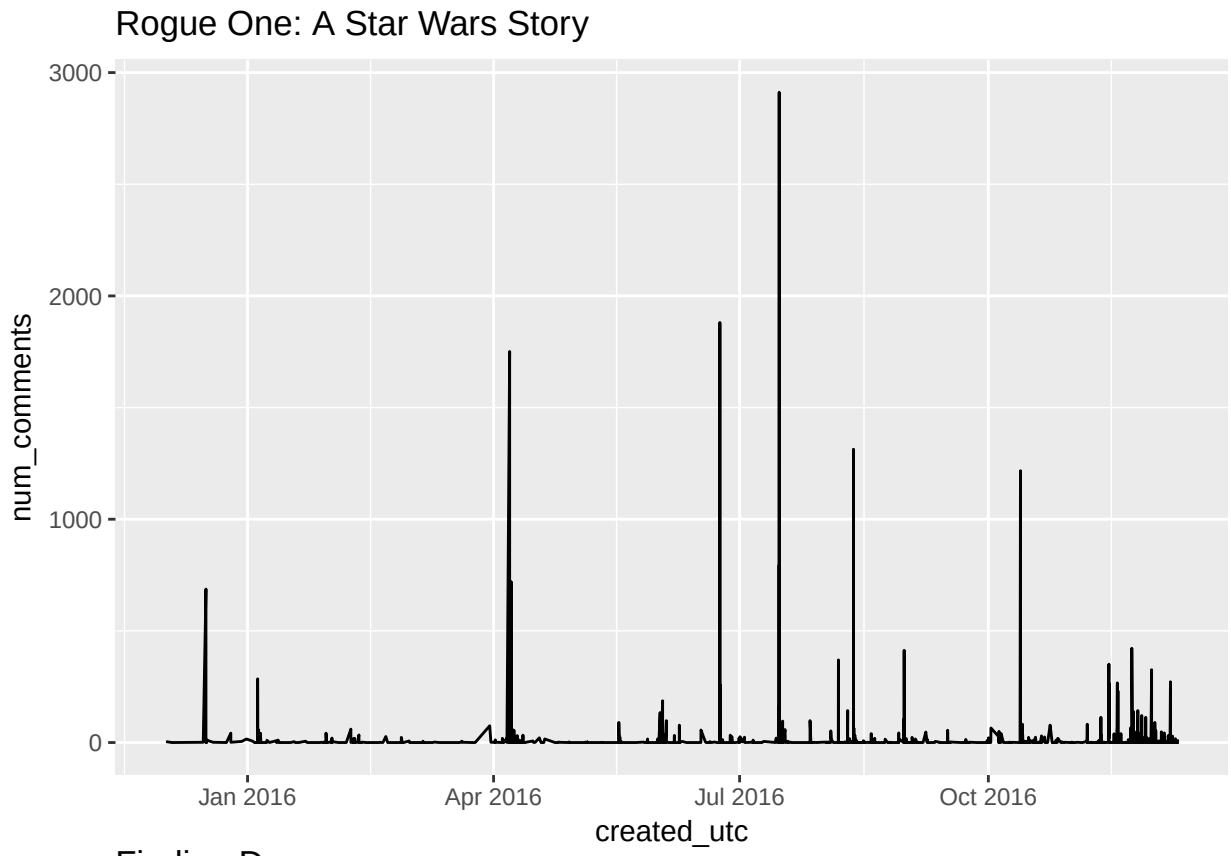

The Great Wall
num_comments
4000
3000
2000
1000
0
Jan 2016
Apr 2016
Jul 2016
Oct 2016
Jan 2017
created_utc

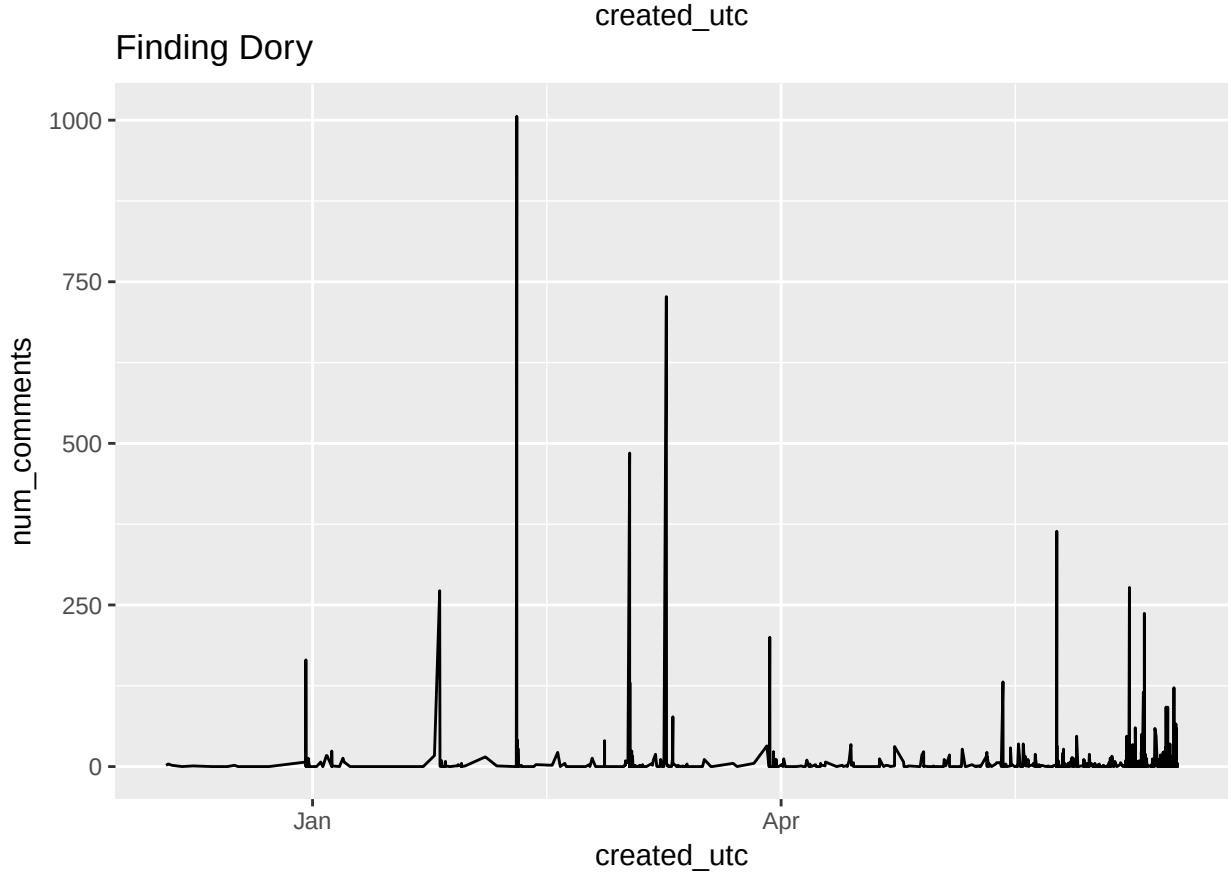

The Huntsman: Winter's War
num_comments
150
100
50
0
Dec
Jan
Feb
Mar
Apr
created_utc

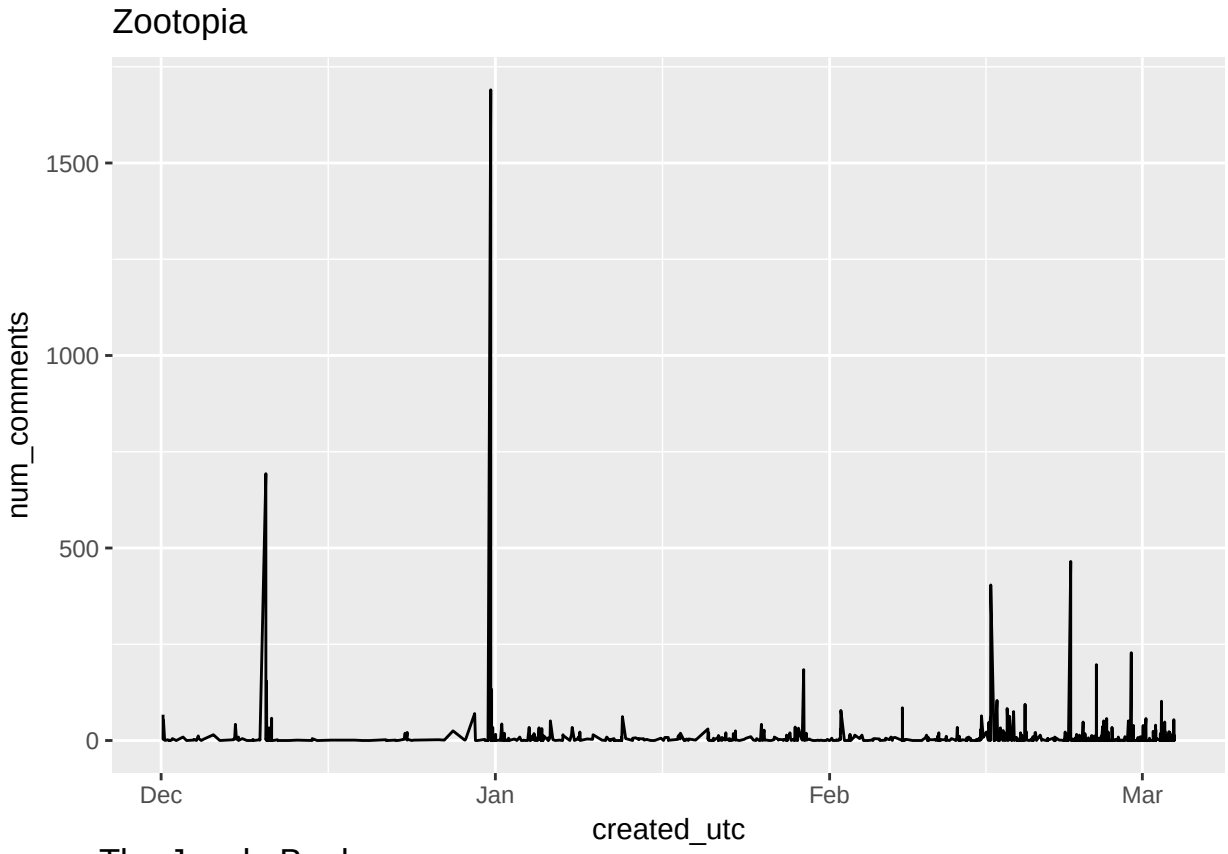

Live by Night
num_comments
600
400
200
0
Apr
Jul
Oct
Jan
created_utc

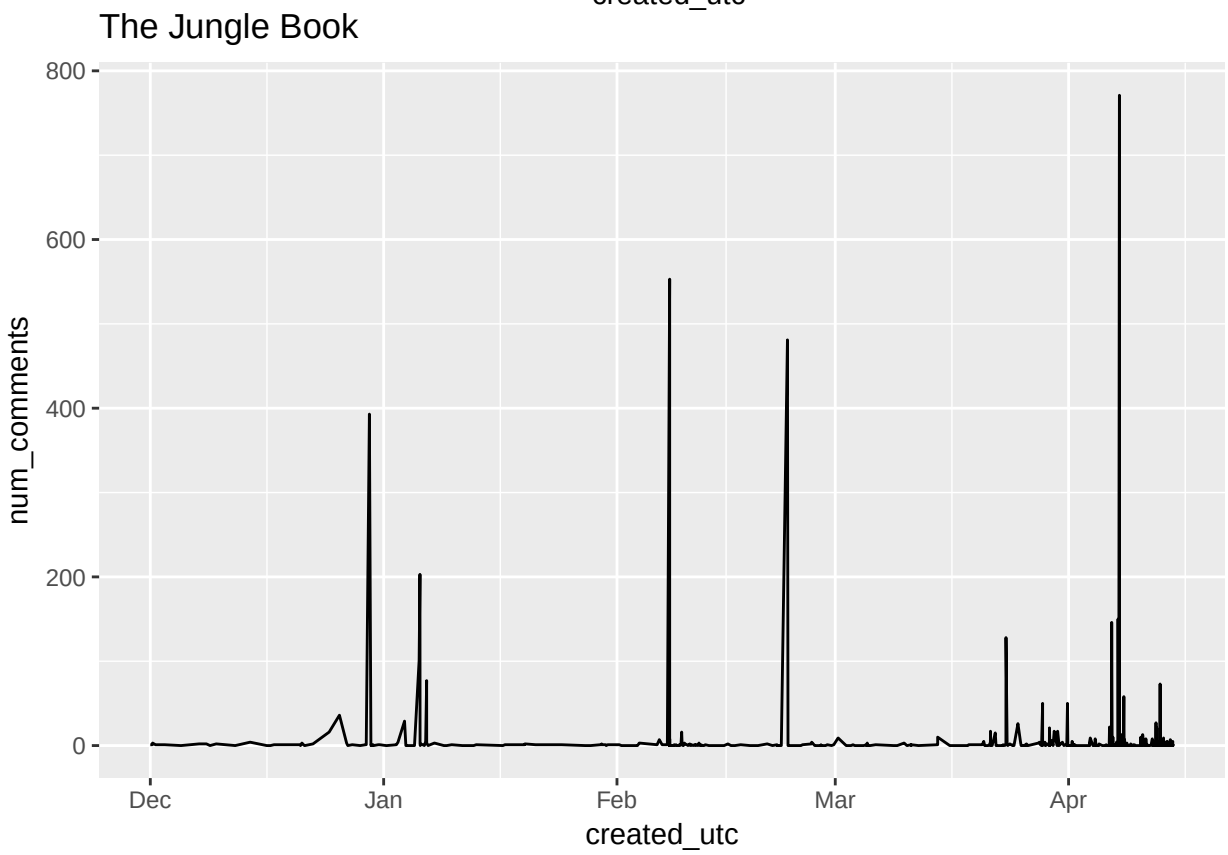

Monster Trucks
num_comments
200
100
0
Jan 2016
Apr 2016
Jul 2016
Oct 2016
Jan 2017
created_utc

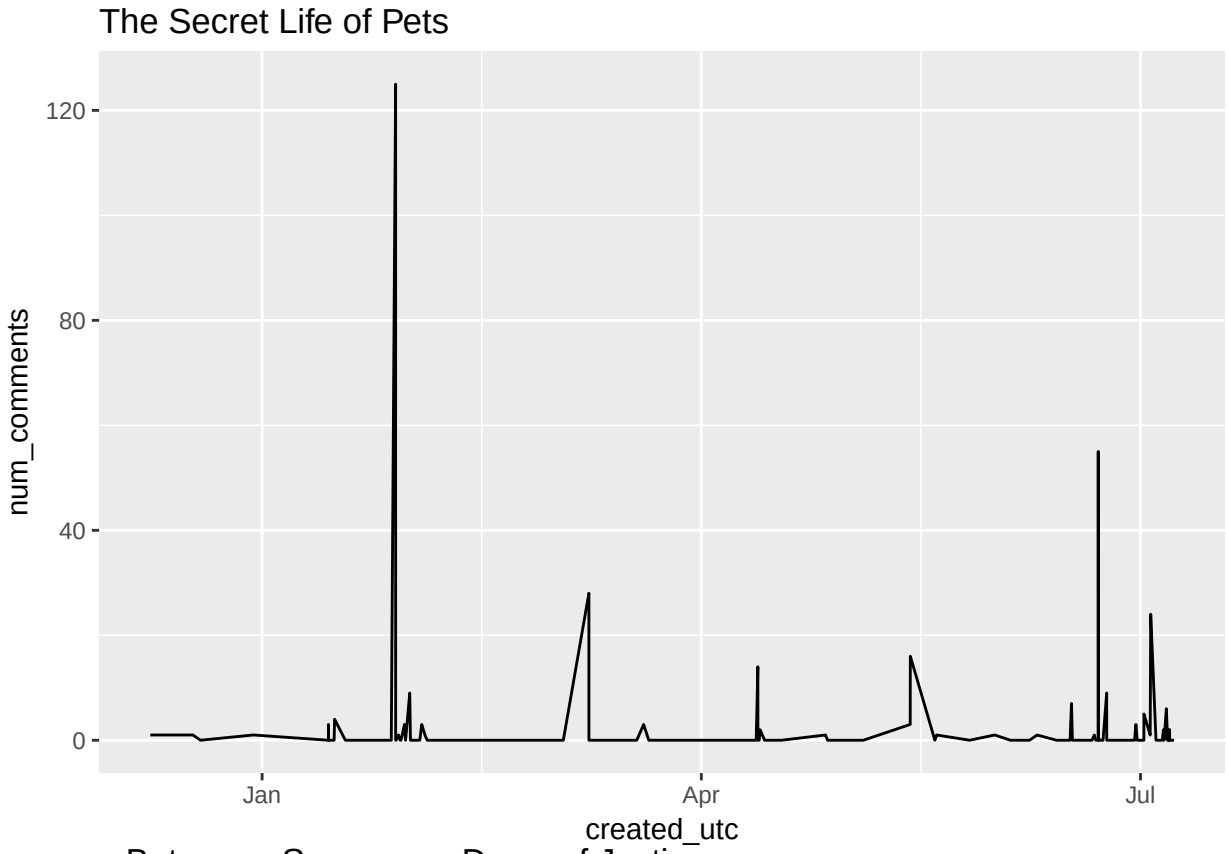

Captain America: Civil War
num_comments
3000
2000
1000
0
Dec
Jan
Feb
Mar
Apr
May
created_utc

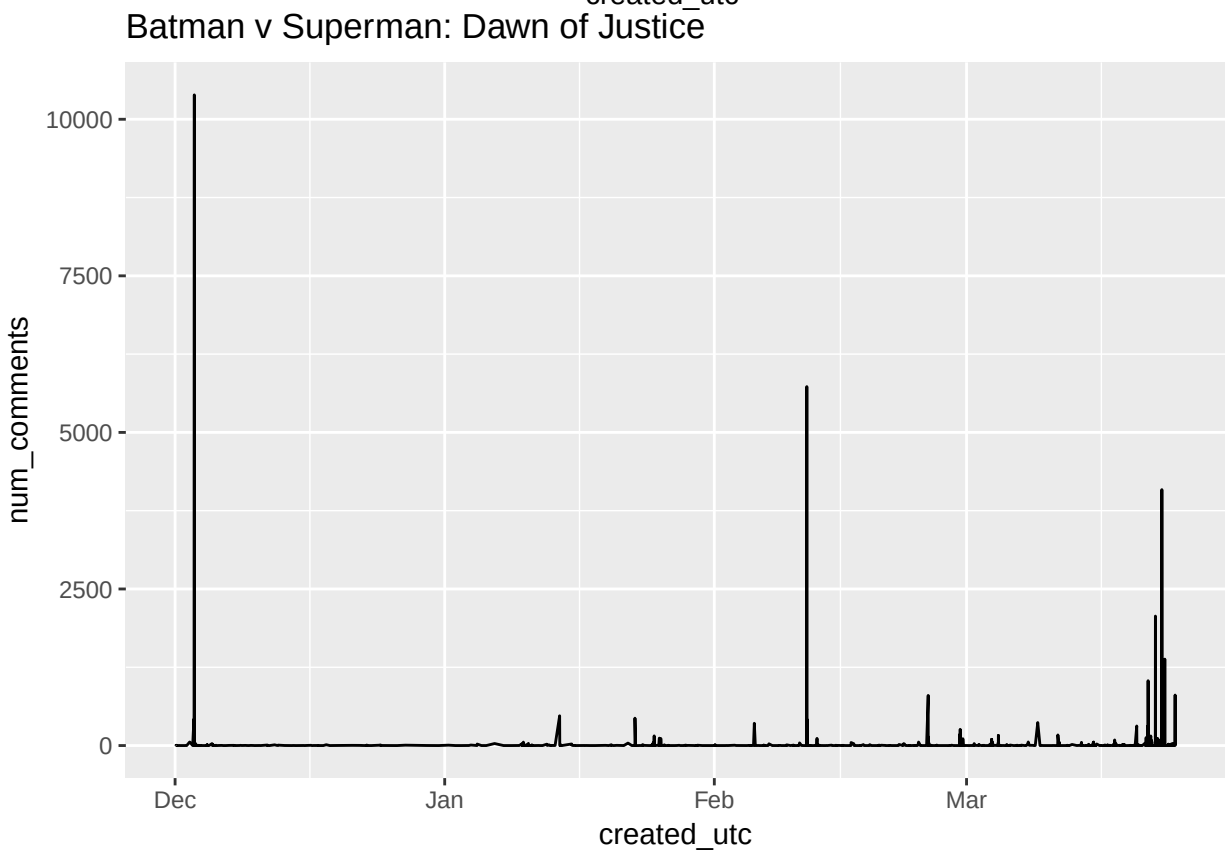

Rogue One: A Star Wars Story
3000
2000
1000
0
num_comments
Jan 2016
Apr 2016
Jul 2016
Oct 2016
created_utc

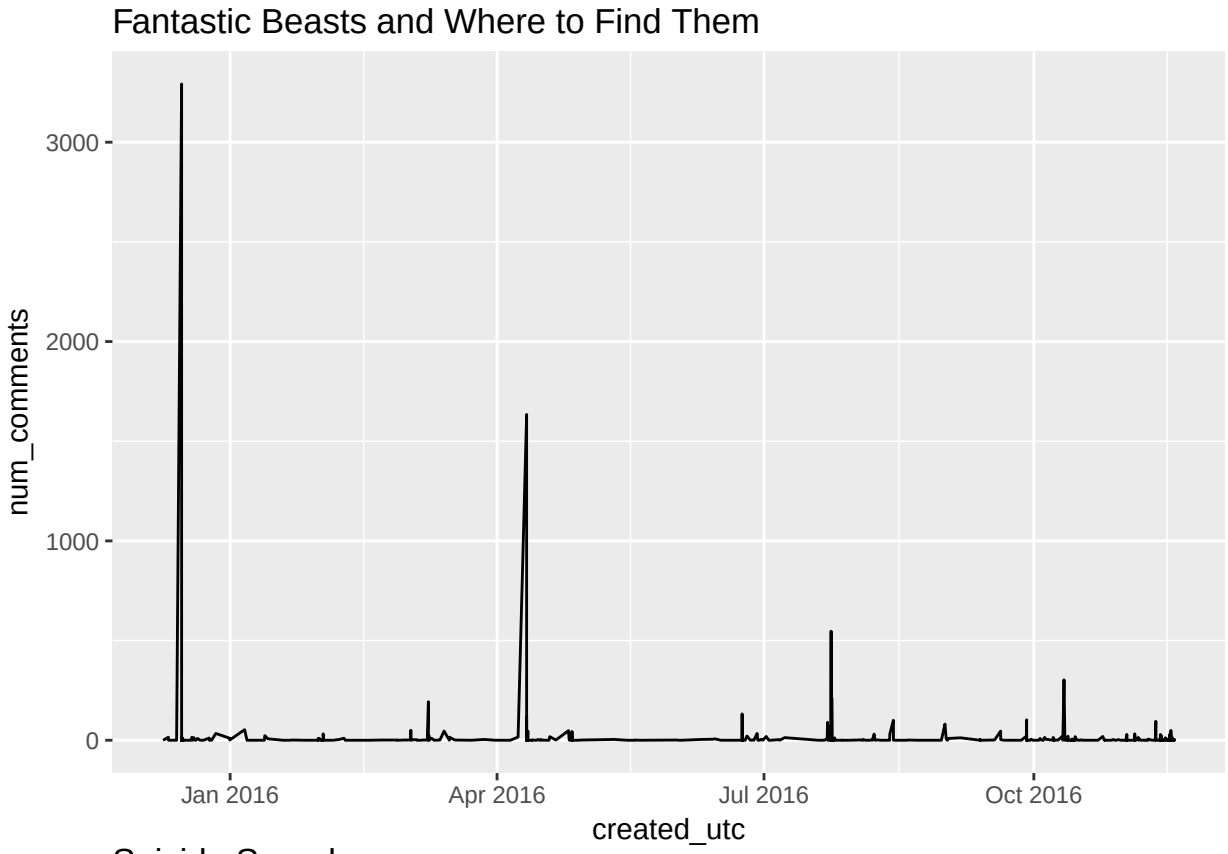

Finding Dory
1000
750
500
250
0
num_comments
Jan
Apr
created_utc

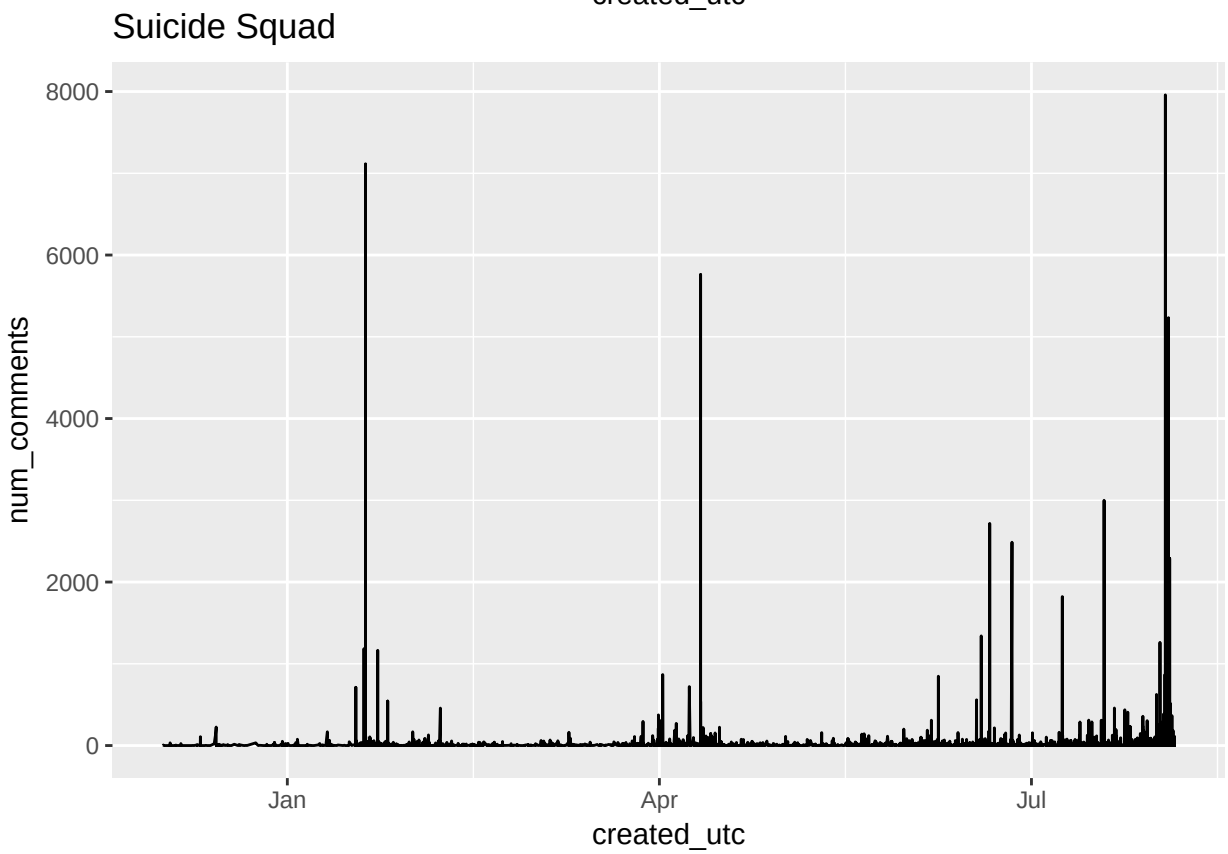

Zootopia
num_comments
1500
1000
500
0
Dec
Jan
Feb
Mar
created_utc


The Jungle Book
num_comments
800
600
400
200
0
Dec
Jan
Feb
Mar
Apr
created_utc


The Secret Life of Pets
num_comments
120
80
40
0
Jan
Apr
Jul
created_utc


Batman v Superman: Dawn of Justice
num_comments
10000
7500
5000
2500
0
Dec
Jan
Feb
Mar
created_utc


Fantastic Beasts and Where to Find Them
num_comments
3000
2000
1000
0
Jan 2016
Apr 2016
Jul 2016
Oct 2016
created_utc


Suicide Squad
num_comments
8000
6000
4000
2000
0
Jan
Apr
Jul
created_utc

Techniques to be used in predictions

I believe the two best techniques to be used for my predictions is going to be either a random forest or using a naive bayesian model. It also may be useful to use a classification algorithm to simplify my problem; rather than trying to predict an exact box office outcome, I could also try and predict whether the movie is a flop, breakeven, or hit. Breaking it up into a categorical variable would allow me to use a support vector machine.

Get the test data

I wanted to test my model with the following five movies

Arrival Moonlight *La La Land* Bad Santa 2 *Allegiant

The first three movies are "hits" The last two movies are "flops"

```
if(exists("movieTestData", inherits = T)) {
    # Pass
  } else {
    movieTestData = getMovieData(testMovies)
  }
```

```
## [1] "Getting data for Arrival"
## [1] 2
## [1] "Getting data for Moonlight"
## [1] 4
## [1] "Getting data for La La Land"
## [1] 6
## [1] "Getting data for Bad Santa 2"
## [1] 8
## [1] "Getting data for Allegiant"
## [1] 10
```

```
movieTestQueries = list()
for(i in 1:nrow(movieTestData))
{
  movieTestQueries <- append(movieTestQueries, moviePostQuery(movieTestData[i,]))
}

if(exists("bigQueryTestData", inherits = T)) {
    # Pass
} else if(file.exists("bigQueryTestData.csv")) {
    bigQueryTestData <- read.csv("bigQueryTestData.csv", header = TRUE)
    class(bigQueryTestData$created_utc) <- class(Sys.time())
  } else {
    bigQueryTestData <- data.frame(created_utc=numeric(0),
                                   subreddit=character(0),
                                   author=character(0),
                                   domain=character(0),
                                   num_comments=numeric(0),
                                   score=numeric(0),
                                   ups=numeric(0),
                                   downs=numeric(0),
                                   title=character(0),
                                   selftext=character(0),
                                   id=character(0),
```

```
                                gilded=numeric(0),
                                movie=character(0),
                                budget=numeric(0),
                                revenue=numeric(0),
                                margin=numeric(0),
                                stringsAsFactors=FALSE)
    for(i in 1:length(movieTestQueries))
    {
      post.data <- query_exec(movieTestQueries[[i]][1], project = project, useLegacySql = FALSE, max_pag
      post.data$movie = movieTestData[i,]$movie
      post.data$budget = movieTestData[i,]$budget
      post.data$revenue = movieTestData[i,]$revenue
      post.data$margin = 100 * (movieTestData[i,]$revenue - movieTestData[i,]$budget) / movieTestData[i
      print(paste("The response has",nrow(post.data), "rows"))
      for(x in 1:nrow(post.data))
      {
        bigQueryTestData[nrow(bigQueryTestData)+1,] = post.data[x,]
      }
    }

    write.csv(bigQueryTestData, file = "bigQueryTestData.csv", na="NA")
  }
bigQueryTestData = bigQueryTestData[complete.cases(bigQueryTestData), ]
```

Predictions

I wanted to start by trying to predict using an SVM classifier. I'm going to try using two different kernels, radial and linear.

```
# levels(bigQueryData$subreddit) = subs
# bigQueryTestData$subreddit <- as.factor(bigQueryTestData$subreddit)
# levels(bigQueryTestData$subreddit) = subs
usefulData <- bigQueryData[,c("subreddit", "num_comments", "score", "gilded", "margin")]
usefulData$IsFlop <- as.factor(ifelse(usefulData$margin < 65, 1, 0))
testData <- bigQueryTestData[,c("subreddit", "num_comments", "score", "gilded", "margin", "movie")]
testData$IsFlop <- as.factor(ifelse(testData$margin < 65, 1, 0))
subs <- union(unique(bigQueryData$subreddit), unique(bigQueryTestData$subreddit))
levels(testData$subreddit) = subs
levels(usefulData$subreddit) = subs

if(exists("svm_model", inherits = T)) {
    # Pass
  } else {
    svm_model <- svm(IsFlop ~ . - margin, data=usefulData, kernel = 'linear')
  }
# for(movie in unique(movieTestData$movie))
# {
#   p <- predict(svm_model, testData[testData$movie==movie,1:5])
#   pTable <- table(p)
#   percentChanceFlop <- pTable["1"] / (pTable["1"]+pTable["0"])
#   print(paste("The percent chance", movie, "is a flop:",percentChanceFlop))
#   isFlopText <- ifelse(bigQueryTestData[bigQueryTestData$movie==movie,]$margin[1] < 65, "is", "is not
#   print(paste("The movie", movie, isFlopText,"a flop"))
```

```
# }

prediction <- predict(svm_model, testData[,1:5])
power <- nrow(testData[prediction == testData$IsFlop,]) / nrow(testData)
```

```
## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```
## Warning in `==.default`(prediction, testData$IsFlop): longer object length
## is not a multiple of shorter object length
```

```
print(power)
```

```
## [1] 0.7395427
```

```
rm(usefulData)
rm(testData)
```

I found that SVM is really not useful in this case, for most cases the prediction was just incredibly incorrect. My next mode of predicting is going to be through a random forest combined with sentiment analysis.

Sentiment Analysis

```
if(exists("usefulData", inherits = T)) {
    # Pass
  } else {
    usefulData <- bigQueryData[,c("subreddit", "num_comments", "score", "gilded", "margin", "title", "m
    usefulData$IsFlop <- as.factor(ifelse(usefulData$margin < 65, 1, 0))
    usefulData$titleSentiment <- usefulData$title %>% as.character() %>% get_sentences() %>% sentiment_
  }
if(exists("testData", inherits = T)) {
    # Pass
  } else {
    testData <- bigQueryTestData[,c("subreddit", "num_comments", "score", "gilded", "margin", "movie",
    testData$IsFlop <- as.factor(ifelse(testData$margin < 65, 1, 0))
    testData$titleSentiment <- testData$title %>% as.character() %>% get_sentences() %>% sentiment_by()
  }
if(exists("forestClassPred", inherits = T)) {
    # Pass
  } else {
    forestClassPred <- randomForest::randomForest(IsFlop ~ . - margin - title - subreddit - movie, data=
  }

prediction <- predict(forestClassPred, testData)
power <- nrow(testData[prediction == testData$IsFlop,]) / nrow(testData)
print(power)
```

```
## [1] 0.9417576
```

```
rm(usefulData)
rm(testData)
```

Using Naive Bayes

```
if(exists("usefulData", inherits = T)) {
    # Pass
  } else {
    usefulData <- bigQueryData[,c("subreddit", "num_comments", "score", "gilded", "margin", "title", "m
    usefulData$IsFlop <- as.factor(ifelse(usefulData$margin < 65, 1, 0))
    usefulData$titleSentiment <- usefulData$title %>% as.character() %>% get_sentences() %>% sentiment_
  }
if(exists("testData", inherits = T)) {
    # Pass
  } else {
    testData <- bigQueryTestData[,c("subreddit", "num_comments", "score", "gilded", "margin", "movie",
    testData$IsFlop <- as.factor(ifelse(testData$margin < 65, 1, 0))
    testData$titleSentiment <- testData$title %>% as.character() %>% get_sentences() %>% sentiment_by()
  }
if(exists("naivePred", inherits = T)) {
    # Pass
  } else {
    naivePred <- naiveBayes(IsFlop ~ titleSentiment + score, data=usefulData, na.action=na.exclude)
  }

prediction <- predict(naivePred, testData)
power <- nrow(testData[prediction == testData$IsFlop,]) / nrow(testData)
print(power)
```

```
## [1] 0.9486097
```

```
rm(usefulData)
rm(testData)
```

SVM with Sentiment Analysis

```
if(exists("usefulData", inherits = T)) {
    # Pass
  } else {
    usefulData <- bigQueryData[,c("subreddit", "num_comments", "score", "gilded", "margin", "title", "m
    usefulData$IsFlop <- as.factor(ifelse(usefulData$margin < 65, 1, 0))
    usefulData$titleSentiment <- usefulData$title %>% as.character() %>% get_sentences() %>% sentiment_
  }
if(exists("testData", inherits = T)) {
    # Pass
  } else {
    testData <- bigQueryTestData[,c("subreddit", "num_comments", "score", "gilded", "margin", "movie",
    testData$IsFlop <- as.factor(ifelse(testData$margin < 65, 1, 0))
    testData$titleSentiment <- testData$title %>% as.character() %>% get_sentences() %>% sentiment_by()
  }
if(exists("svm_sentiment_model", inherits = T)) {
    # Pass
  } else {
    svm_sentiment_model <- svm(IsFlop ~ titleSentiment + score + num_comments, data=usefulData, kernel =
    # svm_sentiment_model.1 <- svm(IsFlop ~ titleSentiment + score + num_comments, data=usefulData, ker
    # svm_sentiment_model.2 <- svm(IsFlop ~ titleSentiment + score + num_comments, data=usefulData, ker
```

```
  }

prediction <- predict(svm_sentiment_model, testData)
power <- nrow(testData[prediction == testData$IsFlop,]) / nrow(testData)
print(power)
```

```
## [1] 0.9513186
```

```
rm(usefulData)
rm(testData)
```

Linear Regression

```
goodModel <- function(predColumn, df) {
    initialColumnNames = ""
    for (name in names(df[,!(names(df) %in% c(predColumn))])) {
        initialColumnNames=paste(initialColumnNames,name,"+")
    }
    initialColumnNames = substr(initialColumnNames, 0, nchar(initialColumnNames)-2)
    newFormula = paste(predColumn,"~",initialColumnNames)
    pred <- lm(newFormula, data=df)
    maxValue = max(summary(pred)$coefficients[,4] )
    while(maxValue>0.05 ){
        newFormula = paste(predColumn,"~")
        for (name in names(summary(pred)$coefficients[,4])) {
            if(summary(pred)$coefficients[,4][[name]]!=maxValue & name!="(Intercept)"){
                newFormula = paste(newFormula,name,"+")
            } else if (name!="(Intercept)") {
                print(paste("The",name,"column was dropped, p value:",maxValue))
            }
        }
        newFormula = substr(newFormula, 0, nchar(newFormula)-2)
        pred <- lm(newFormula, data=df)
        maxValue = max(summary(pred)$coefficients[,4] )

        if(summary(pred)$coefficients[,4][["(Intercept)"]]==maxValue){
            break
        }
    }
    return(pred)
}

if(exists("usefulData", inherits = T)) {
    # Pass
  } else {
    usefulData <- bigQueryData[,c("subreddit", "num_comments", "score", "gilded", "margin", "title", "m
    usefulData$titleSentiment <- usefulData$title %>% as.character() %>% get_sentences() %>% sentiment_
    usefulData$IsFlop <- ifelse(usefulData$margin < 65, 1, 0)
  }
if(exists("testData", inherits = T)) {
    # Pass
  } else {
    testData <- bigQueryTestData[,c("subreddit", "num_comments", "score", "gilded", "margin", "movie",
```

```
    testData$titleSentiment <- testData$title %>% as.character() %>% get_sentences() %>% sentiment_by()
    testData$IsFlop <- as.factor(ifelse(testData$margin < 65, 1, 0))
  }
s = c(2, 3, 4, 8, 9)
model <- goodModel("IsFlop", usefulData[,s])
```

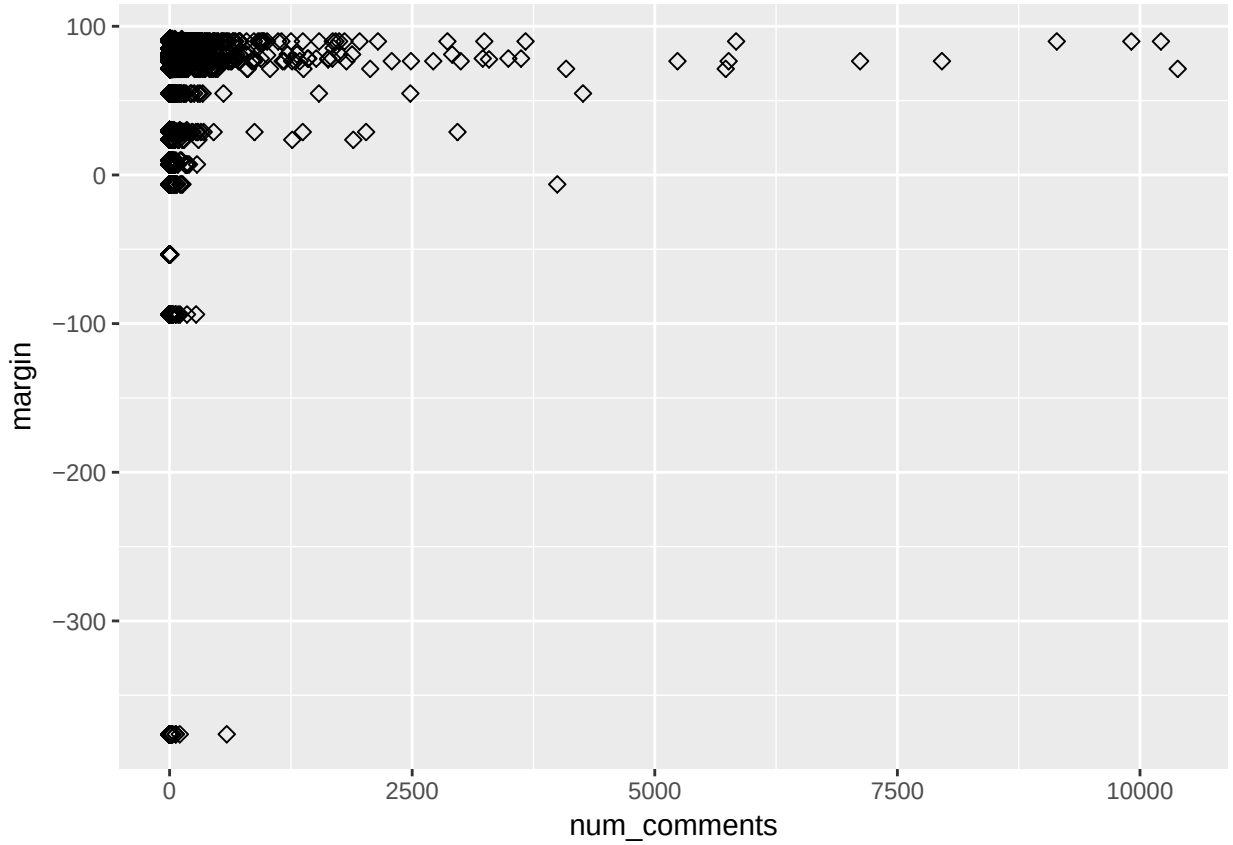
```
## [1] "The gilded column was dropped, p value: 0.243661327540594"
```

```
prediction <- predict(model, testData)
prediction <- as.factor(ifelse(prediction < 0.5, 1, 0))
power <- nrow(testData[prediction == testData$IsFlop,]) / nrow(testData)
print(power)
```

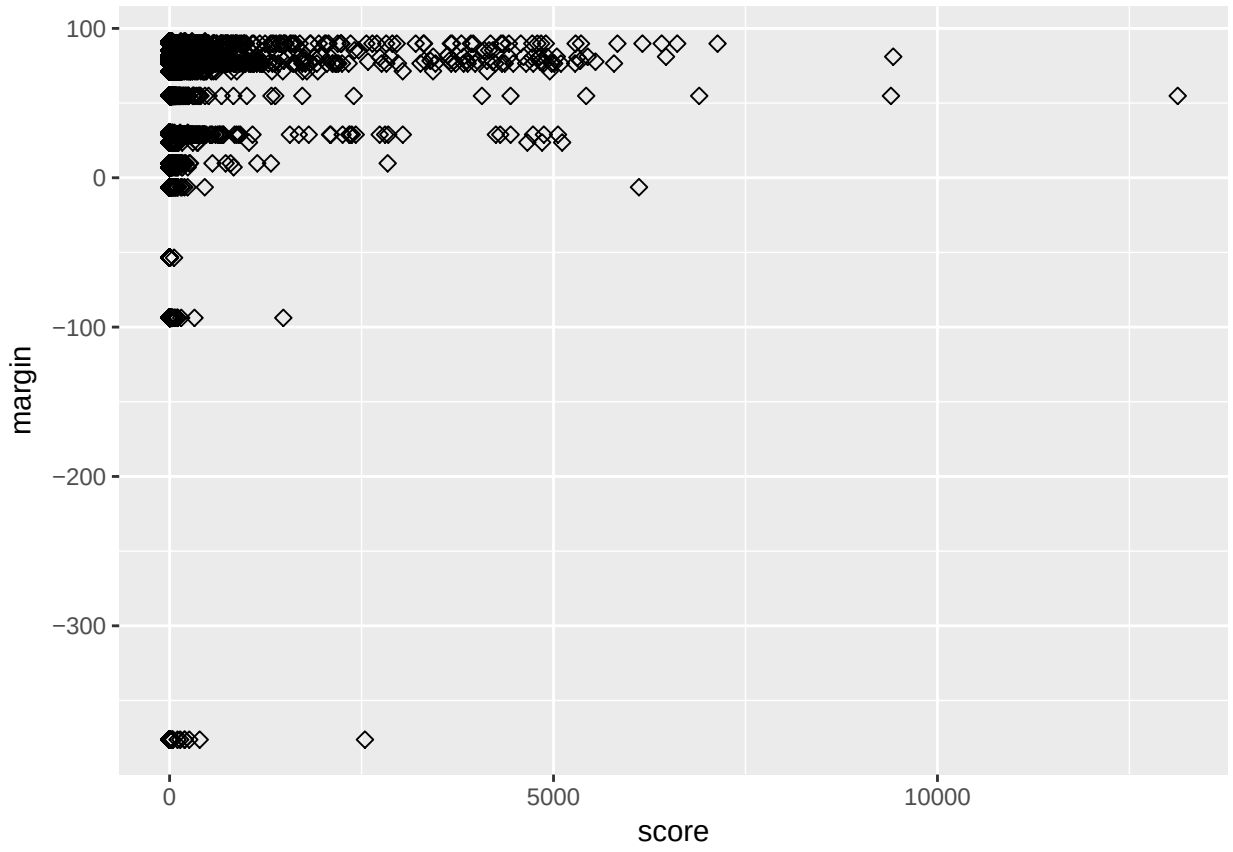
```
## [1] 0.0489204
```

```
rm(usefulData)
rm(testData)
```

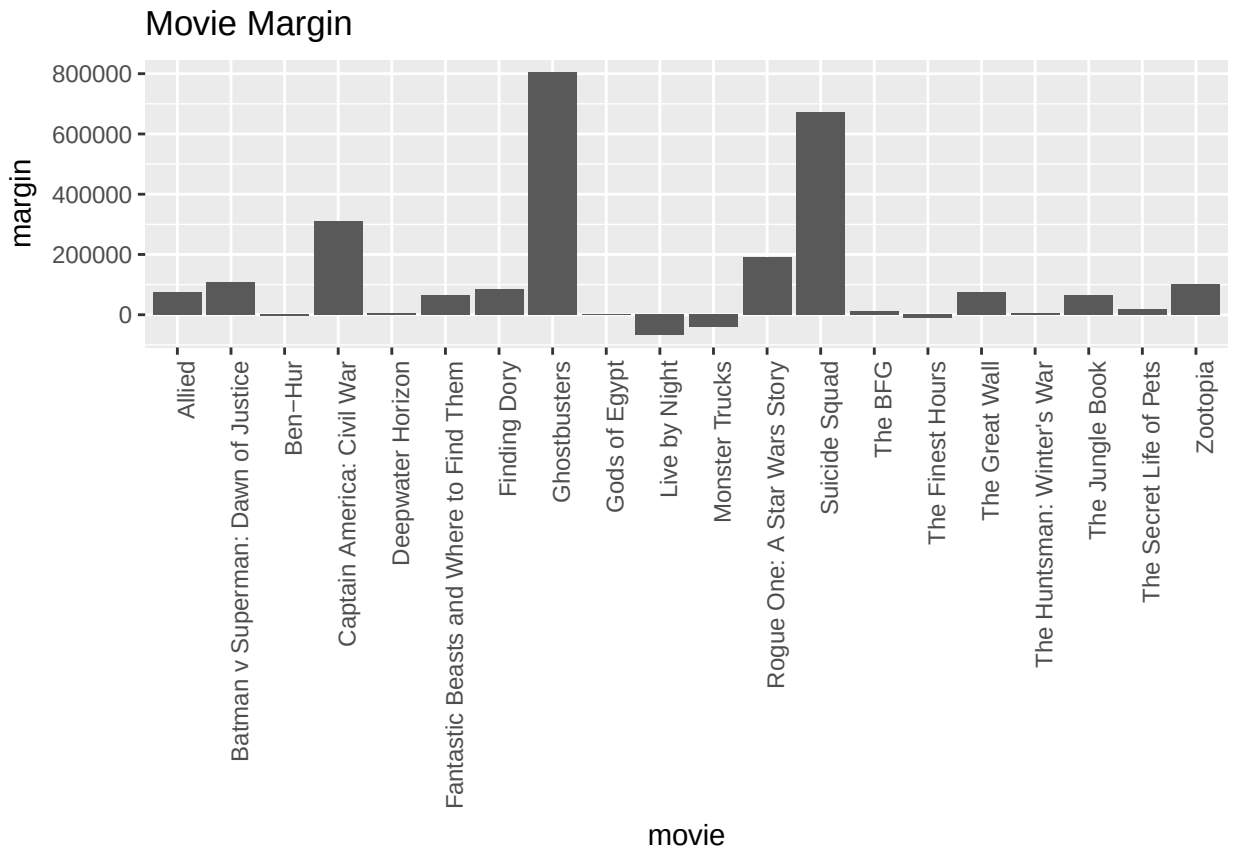
```
margin.vs.num_comments <- ggplot(bigQueryData, aes(x=num_comments, y=margin)) + geom_point(size=2, shape
margin.vs.score <- ggplot(bigQueryData, aes(x=score, y=margin)) + geom_point(size=2, shape=23)
print(margin.vs.num_comments)
```



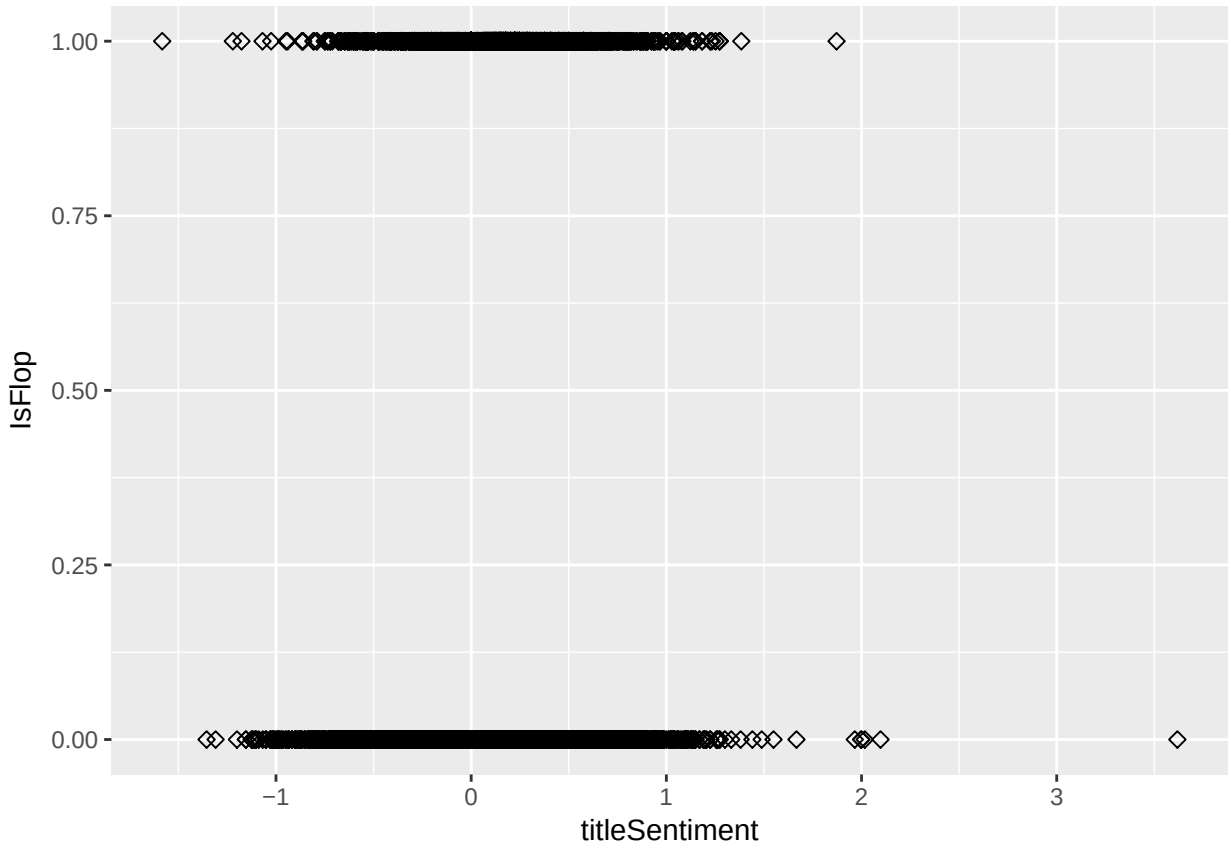

```
print(margin.vs.score)
```



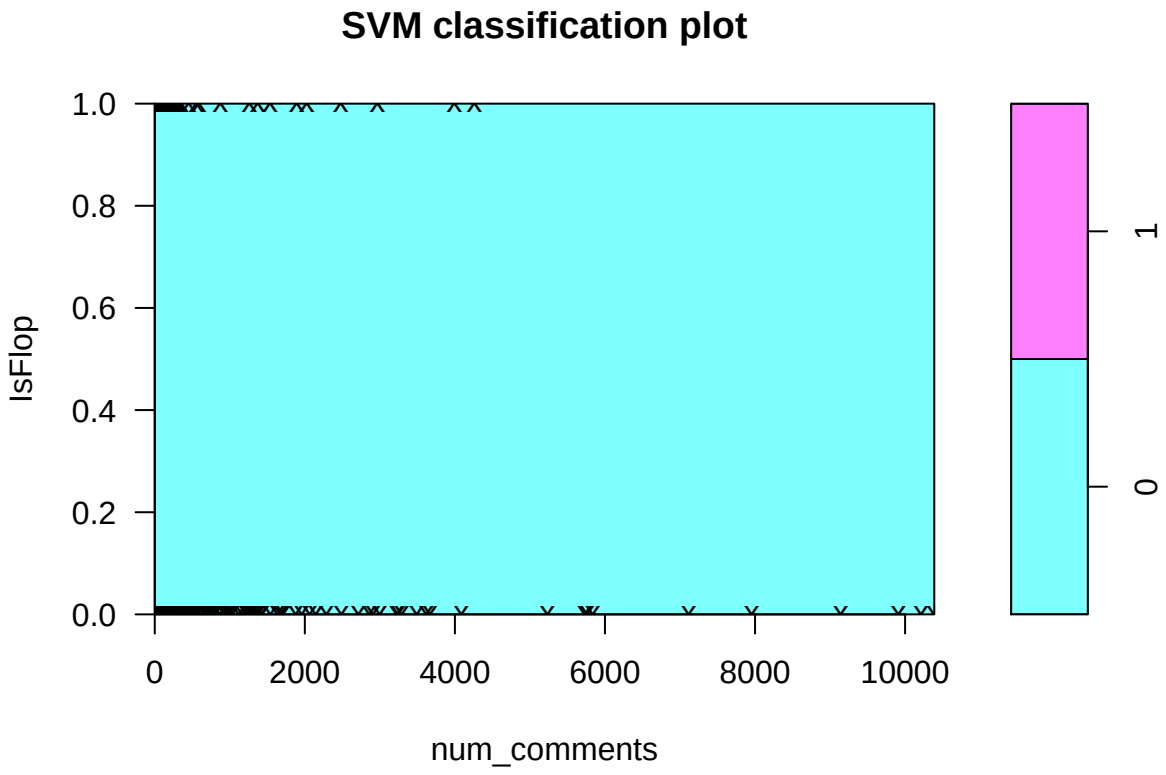

```
margin.for.movie <- ggplot(bigQueryData, aes(x=movie, y=margin)) + geom_bar(stat = "identity") + theme(a
print(margin.for.movie)
```




```
usefulData <- bigQueryData[,c("subreddit", "num_comments", "score", "gilded", "margin", "title", "movie
usefulData$titleSentiment <- usefulData$title %>% as.character() %>% get_sentences() %>% sentiment_by()
usefulData$IsFlop <- ifelse(usefulData$margin < 65, 1, 0)
flop.vs.sentiment <- ggplot(usefulData, aes(x=titleSentiment, y=IsFlop)) + geom_point(size=2, shape=23)
print(flop.vs.sentiment)
```




```
svm.plot <- plot(svm_sentiment_model, usefulData, IsFlop ~ num_comments)
```



```r
print(svm.plot)
```

```
## NULL
```