ProblemSet2

Yiming Chen

Problem 1 - Modified Random walk

a

```
random_walk1 <- function(n, seed = NULL) {
  set.seed(seed)
  directions <- sample(c(-1, 1), n, replace = TRUE)
  probs <- runif(n)

  steps <- numeric(n)
  for (i in 1:n) {
    if (directions[i] == 1) {
      steps[i] <- ifelse(probs[i] < 0.05, 10, 1)
    } else {
      steps[i] <- ifelse(probs[i] < 0.20, -3, -1)
    }
  }
  sum(steps)
}
```

```
random_walk2 <- function(n, seed = NULL) {
  set.seed(seed)
  directions <- sample(c(-1, 1), n, replace = TRUE)
  probs <- runif(n)

  steps <- ifelse(
    directions == 1,
    ifelse(probs < 0.05, 10, 1),
    ifelse(probs < 0.20, -3, -1)
  )
```

```r
  sum(steps)
}
```

```r
random_walk3 <- function(n, seed = NULL) {
  set.seed(seed)
  directions <- sample(c(-1, 1), n, replace = TRUE)
  probs <- runif(n)

  steps <- sapply(1:n, function(i) {
    if (directions[i] == 1) {
      ifelse(probs[i] < 0.05, 10, 1)
    } else {
      ifelse(probs[i] < 0.20, -3, -1)
    }
  })
  sum(steps)
}
```

```r
random_walk1(10)
```

```
[1] 0
```

```r
random_walk2(10)
```

```
[1] -6
```

```r
random_walk3(10)
```

```
[1] -10
```

```r
random_walk1(1000)
```

```
[1] 42
```

```r
random_walk2(1000)
```

```
[1] 158
```

```r
random_walk3(1000)
```

```
[1] -109
```

b

```r
random_walk1(10, seed = 42)
```

```
[1] 0
```

```r
random_walk2(10, seed = 42)
```

```
[1] 0
```

```r
random_walk3(10, seed = 42)
```

```
[1] 0
```

```r
random_walk1(1000, seed = 42)
```

```
[1] 38
```

```r
random_walk2(1000, seed = 42)
```

```
[1] 38
```

```r
random_walk3(1000, seed = 42)
```

```
[1] 38
```

c

```r
library(microbenchmark)
```

```
Warning: package 'microbenchmark' was built under R version 4.4.2
```

```r
n <- 1000
microbenchmark(
  loop = random_walk1(n, seed = 42),
  vectorized = random_walk2(n, seed = 42),
  applyfun = random_walk3(n, seed = 42),
  times = 50
)
```

```
Unit: microseconds
       expr      min       lq     mean    median       uq      max neval
       loop  619.301  649.201  732.267  674.2515  737.102 1726.601    50
 vectorized   72.100   84.101   98.853   94.2510  113.400  151.401    50
   applyfun 1025.700 1073.100 1179.849 1138.0010 1234.202 1925.001    50
```

```r
n <- 100000
microbenchmark(
  loop = random_walk1(n, seed = 42),
  vectorized = random_walk2(n, seed = 42),
  applyfun = random_walk3(n, seed = 42),
  times = 20
)
```

```
Unit: milliseconds
       expr      min         lq       mean     median         uq        max
       loop  77.9715  83.773051  95.513671  96.910950 105.501801 118.403802
 vectorized   6.6309   6.950001   7.577346   7.538201   8.175051   8.744201
   applyfun 125.3067 132.268851 155.795721 157.373101 168.694101 211.629401
 neval
    20
    20
    20
```

The vectorized implementation (`random_walk2`) consistently outperforms both alternatives, demonstrating the efficiency of R's vectorized operations. In contrast, the loop-based approach (`random_walk1`) scales poorly but still surpasses the apply-based version. The apply implementation (`random_walk3`) proves to be the slowest for both small and large inputs.

d

```r
estimate_prob1 <- function(n, trials = 100000) {
  results <- replicate(trials, random_walk1(n))
  mean(results == 0)
}
estimate_prob2 <- function(n, trials = 100000) {
  results <- replicate(trials, random_walk2(n))
  mean(results == 0)
}
estimate_prob3 <- function(n, trials = 100000) {
  results <- replicate(trials, random_walk3(n))
  mean(results == 0)
}
set.seed(42)

prob2_10   <- estimate_prob2(10, trials = 10000)
prob2_100  <- estimate_prob2(100, trials = 10000)
prob2_1000 <- estimate_prob2(1000, trials = 10000)
cat("The probability that the random walk ends at 0 with 10 steps is:", prob2_10, '\n')
```

```
The probability that the random walk ends at 0 with 10 steps is: 0.1326
```

```r
cat("The probability that the random walk ends at 0 with 100 steps is:", prob2_100, '\n')
```

```
The probability that the random walk ends at 0 with 100 steps is: 0.0211
```

```r
cat("The probability that the random walk ends at 0 with 1000 steps is:", prob2_1000)
```

```
The probability that the random walk ends at 0 with 1000 steps is: 0.0055
```

Problem 2 - Mean of Mixture of Distributions

```r
estimate_daily_mean <- function(trials = 100000, seed = 42) {
  set.seed(seed)
  counts <- rpois(trials, 8) + 2 * rnorm(trials, mean = 60, sd = sqrt(12)) + rpois(trials, 6
  return(counts)
```

```r
}


n <- estimate_daily_mean(trials = 200000, seed = 42)
cat('The estimation of the average number of cars \n that pass an intersection per day under
```

```
The estimation of the average number of cars
 that pass an intersection per day under assumptions is: 264
```

Problem 3 - Linear Regression

a

```r
youtube <- read.csv('https://raw.githubusercontent.com/rfordatascience/tidytuesday/master/da
colnames(youtube)
```

```
 [1] "year"                      "brand"
 [3] "superbowl_ads_dot_com_url" "youtube_url"
 [5] "funny"                     "show_product_quickly"
 [7] "patriotic"                 "celebrity"
 [9] "danger"                    "animals"
[11] "use_sex"                   "id"
[13] "kind"                      "etag"
[15] "view_count"                "like_count"
[17] "dislike_count"             "favorite_count"
[19] "comment_count"             "published_at"
[21] "title"                     "description"
[23] "thumbnail"                 "channel_title"
[25] "category_id"
```

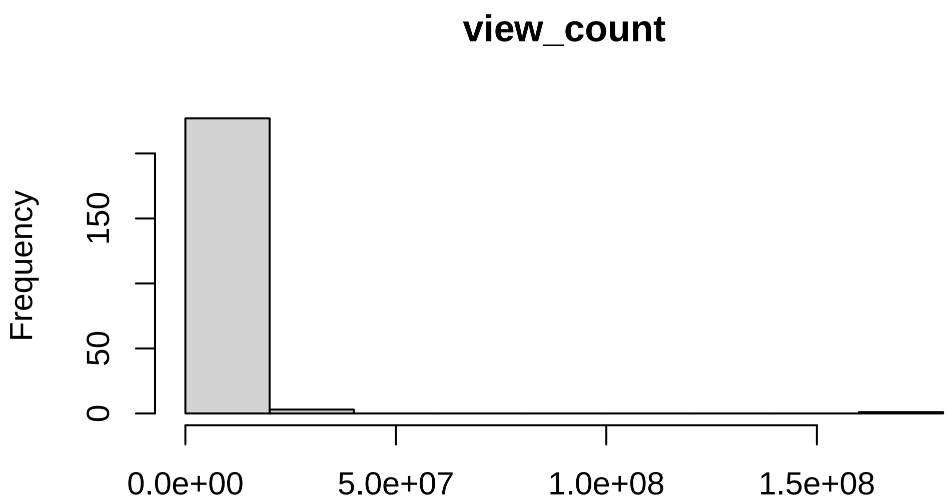
```r
youtube_deid <- youtube[, c("year","funny","show_product_quickly","patriotic",
                            "celebrity","danger","animals","use_sex",
                            "view_count","like_count","dislike_count",
                            "favorite_count","comment_count","category_id")]
dim(youtube_deid)
```

```
[1] 247  14
```

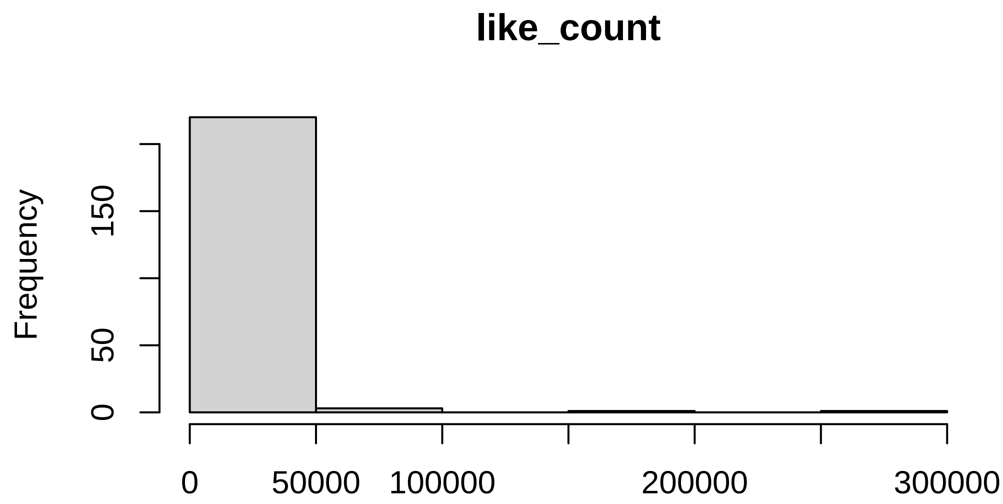
The dimensions of the data after removing these columns are 247 * 14 (247 rows * 14 columns).

b

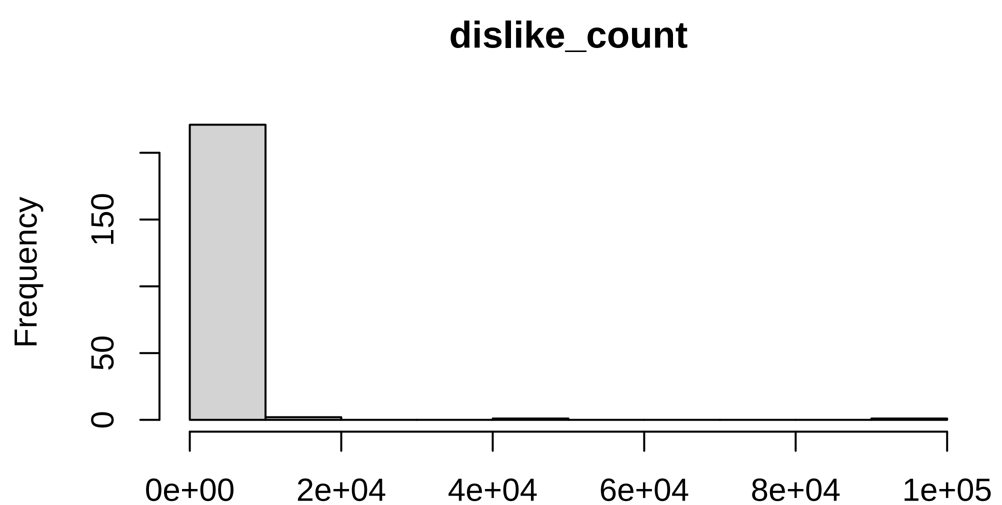
```r
hist(youtube_deid$view_count, main="view_count", xlab="")
```



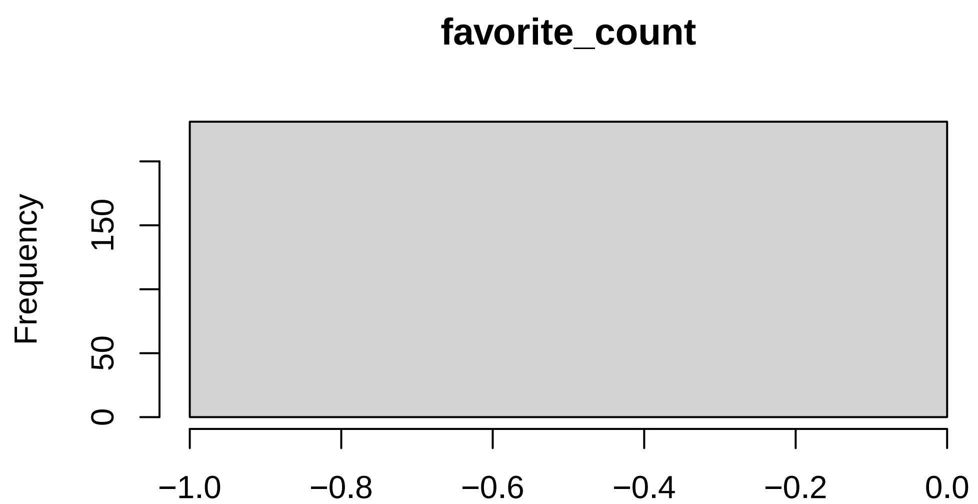

```r
hist(youtube_deid$like_count, main="like_count", xlab="")
```



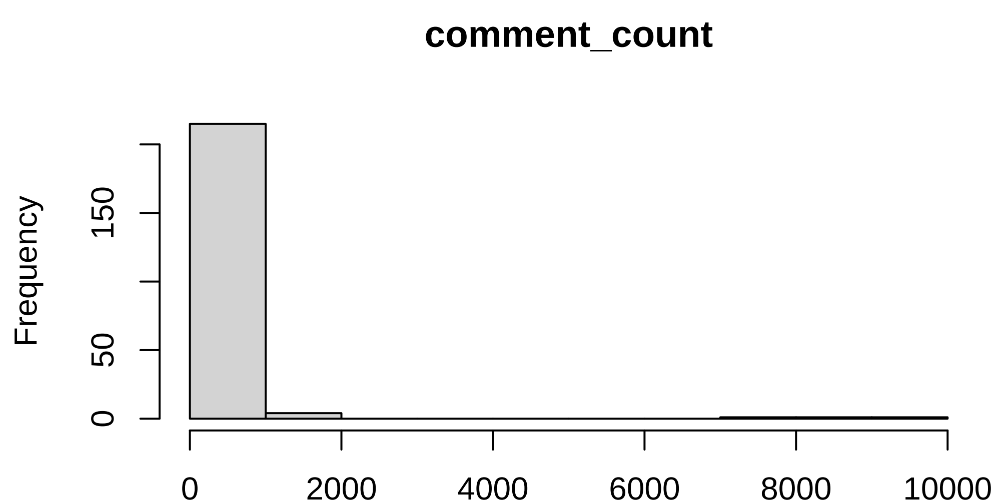

```
hist(youtube_deid$dislike_count, main="dislike_count", xlab="")
```



```r
hist(youtube_deid$favorite_count, main="favorite_count", xlab="")
```

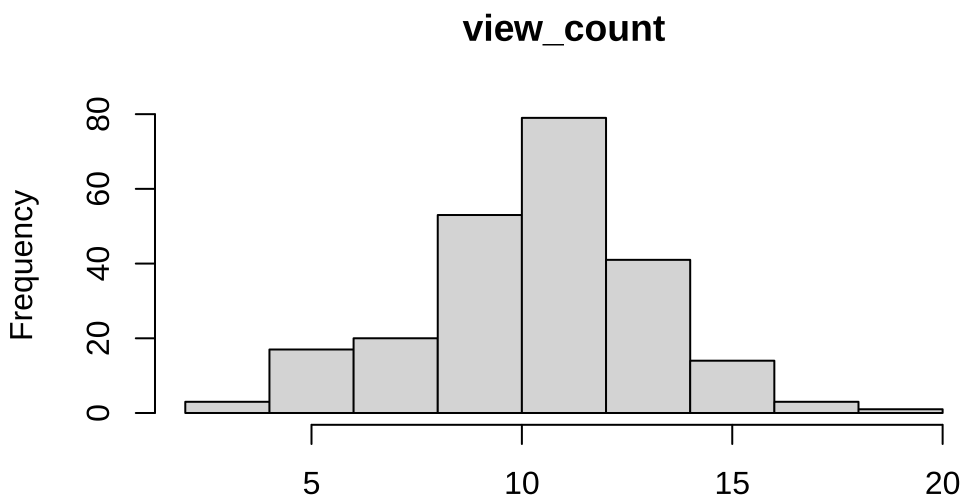



```r
hist(youtube_deid$comment_count, main="comment_count", xlab="")
```

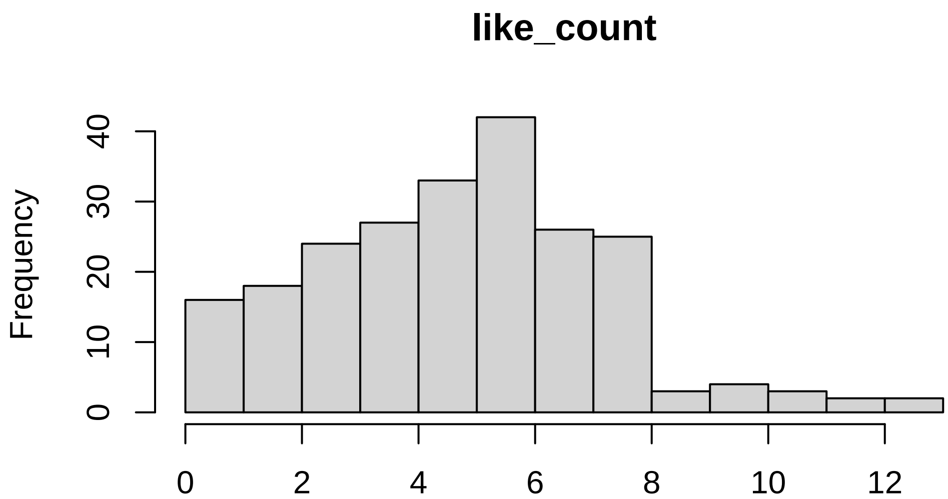


For these five variables, favorite_count only has 0 and null values, so it would not be appropriate to use as the outcome in a linear regression model. All the rest variables are right-skewed,so we can carry out the log(1+x) transformation to make it prior to being used as the outcome in a linear regression model. The histograms after transformation are shown below.

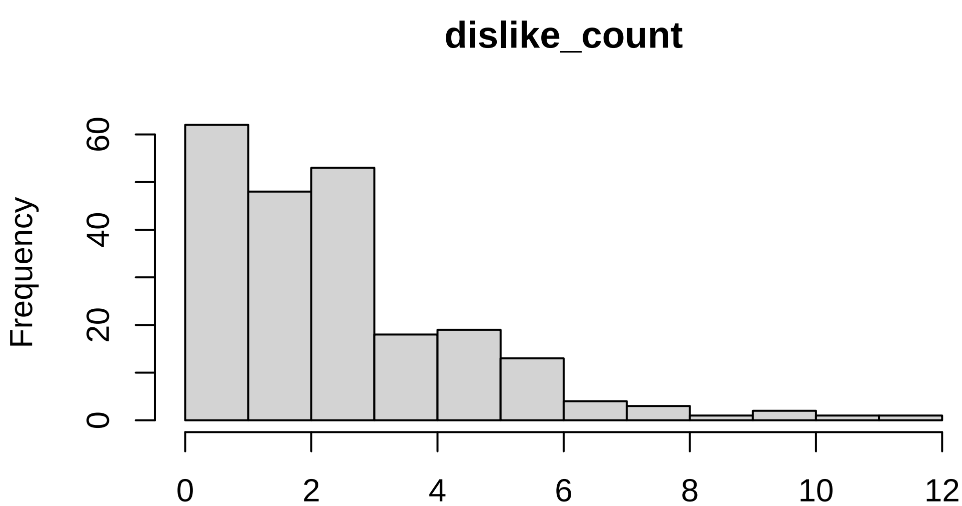
```
hist(log1p(youtube_deid$view_count), main="view_count", xlab="")
```



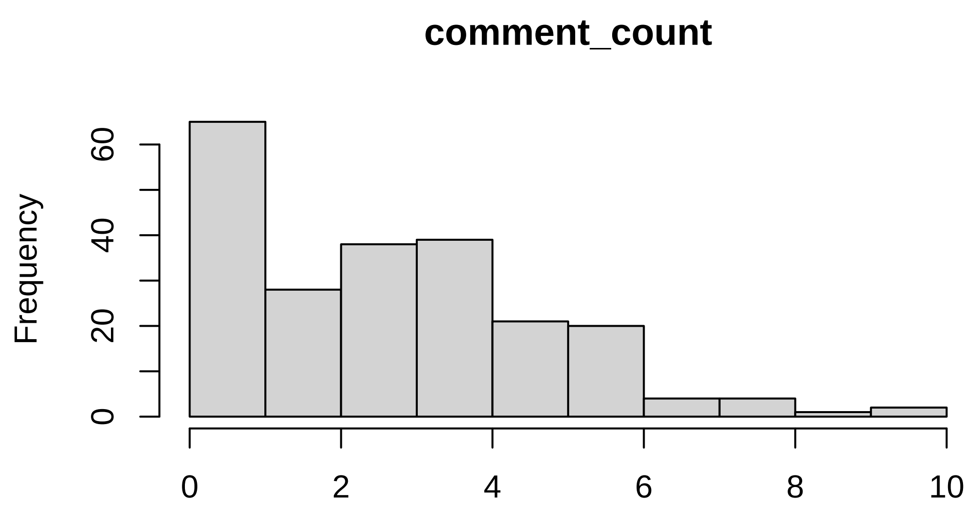

```
hist(log1p(youtube_deid$like_count), main="like_count", xlab="")
```




```r
hist(log1p(youtube_deid$dislike_count), main="dislike_count", xlab="")
```



```r
hist(log1p(youtube_deid$comment_count), main="comment_count", xlab="")
```



c

```r
Sys.setenv(LANG = "en")
Sys.setlocale("LC_ALL", "C")
```

```
[1] "C"
```

```r
youtube_deid$log_views <- log1p(youtube$view_count)
youtube_deid$log_likes <- log1p(youtube$like_count)
youtube_deid$log_dislikes <- log1p(youtube$dislike_count)
youtube_deid$log_comments <- log1p(youtube$comment_count)
model_views <- lm(log_views ~ funny + show_product_quickly + patriotic +
                    celebrity + danger + animals + use_sex + year,
                  data = youtube_deid)

model_likes <- lm(log_likes ~ funny + show_product_quickly + patriotic +
                    celebrity + danger + animals + use_sex + year,
```

```r
                  data = youtube_deid)

model_dislikes <- lm(log_dislikes ~ funny + show_product_quickly + patriotic +
                       celebrity + danger + animals + use_sex + year,
                     data = youtube_deid)

model_comments <- lm(log_comments ~ funny + show_product_quickly + patriotic +
                       celebrity + danger + animals + use_sex + year,
                     data = youtube_deid)
```

```r
summary(model_views)
```

```
Call:
lm(formula = log_views ~ funny + show_product_quickly + patriotic +
    celebrity + danger + animals + use_sex + year, data = youtube_deid)

Residuals:
    Min      1Q  Median      3Q     Max
-7.7742 -1.6152  0.1311  1.7036  8.4481

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)
(Intercept)              -31.55016   71.00538  -0.444    0.657
funnyTRUE                  0.56492    0.46702   1.210    0.228
show_product_quicklyTRUE   0.21089    0.40530   0.520    0.603
patrioticTRUE              0.50699    0.53811   0.942    0.347
celebrityTRUE              0.03548    0.42228   0.084    0.933
dangerTRUE                 0.63131    0.41812   1.510    0.132
animalsTRUE               -0.31002    0.39348  -0.788    0.432
use_sexTRUE               -0.38671    0.44782  -0.864    0.389
year                       0.02053    0.03531   0.582    0.561

Residual standard error: 2.787 on 222 degrees of freedom
  (16 observations deleted due to missingness)
Multiple R-squared:  0.02694,   Adjusted R-squared:  -0.008122
F-statistic: 0.7684 on 8 and 222 DF,  p-value: 0.631
```

For the model of view_count, none of the attributes shows statistically significant with the view counts.

```r
summary(model_likes)
```

```
Call:
lm(formula = log_likes ~ funny + show_product_quickly + patriotic +
    celebrity + danger + animals + use_sex + year, data = youtube_deid)

Residuals:
    Min      1Q  Median      3Q     Max
-5.2860 -1.6333  0.0868  1.4911  7.7431

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)
(Intercept)              -150.51357   63.45723  -2.372   0.0186 *
funnyTRUE                   0.47476    0.41816   1.135   0.2575
show_product_quicklyTRUE    0.20017    0.36391   0.550   0.5828
patrioticTRUE               0.80689    0.49791   1.621   0.1066
celebrityTRUE               0.41283    0.38212   1.080   0.2812
dangerTRUE                  0.63895    0.37350   1.711   0.0886 .
animalsTRUE                -0.05944    0.35298  -0.168   0.8664
use_sexTRUE                -0.42952    0.40064  -1.072   0.2849
year                        0.07685    0.03155   2.436   0.0157 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.467 on 216 degrees of freedom
  (22 observations deleted due to missingness)
Multiple R-squared:  0.07313,   Adjusted R-squared:  0.03881
F-statistic:  2.13 on 8 and 216 DF,  p-value: 0.0342
```

For the model of like_count, only the attribute—'year' shows statistically significant association which is also positive. And the 'danger' attribute tends to have a positive and statistically significant association.

```r
summary(model_dislikes)
```

```
Call:
lm(formula = log_dislikes ~ funny + show_product_quickly + patriotic +
    celebrity + danger + animals + use_sex + year, data = youtube_deid)
```

```
Residuals:
    Min      1Q  Median      3Q     Max 
-4.0292 -1.3299 -0.3192  0.8986  8.7219 

Coefficients:
                           Estimate Std. Error t value Pr(>|t|)    
(Intercept)              -183.06813   53.34768  -3.432 0.000719 ***
funnyTRUE                   0.25949    0.35154   0.738 0.461224    
show_product_quicklyTRUE    0.27511    0.30593   0.899 0.369515    
patrioticTRUE               0.81407    0.41859   1.945 0.053095 .  
celebrityTRUE              -0.20214    0.32125  -0.629 0.529852    
dangerTRUE                  0.22184    0.31400   0.707 0.480630    
animalsTRUE                -0.21192    0.29675  -0.714 0.475911    
use_sexTRUE                -0.32980    0.33681  -0.979 0.328583    
year                        0.09207    0.02653   3.471 0.000626 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.074 on 216 degrees of freedom
  (22 observations deleted due to missingness)
Multiple R-squared:  0.09753,	Adjusted R-squared:  0.06411 
F-statistic: 2.918 on 8 and 216 DF,  p-value: 0.004115
```

For the model of dislike_count, same as like_count, only 'year' shows a positive statistically significant association and the patriotic attribute tends to have a positive statistically significant association with dislike counts.

```r
summary(model_comments)
```

```
Call:
lm(formula = log_comments ~ funny + show_product_quickly + patriotic + 
    celebrity + danger + animals + use_sex + year, data = youtube_deid)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.1372 -1.4665 -0.1427  1.4061  5.8468 

Coefficients:
                          Estimate Std. Error t value Pr(>|t|)  
(Intercept)              -99.09835   52.92351  -1.872   0.0625 .
funnyTRUE                  0.21954    0.34528   0.636   0.5256  
```

```
show_product_quicklyTRUE   0.40939    0.30229   1.354   0.1771
patrioticTRUE              0.66698    0.39902   1.672   0.0961 .
celebrityTRUE              0.29767    0.31541   0.944   0.3464
dangerTRUE                 0.17784    0.31069   0.572   0.5677
animalsTRUE               -0.26802    0.29347  -0.913   0.3621
use_sexTRUE               -0.39323    0.33163  -1.186   0.2370
year                       0.05034    0.02632   1.913   0.0571 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.039 on 213 degrees of freedom
  (25 observations deleted due to missingness)
Multiple R-squared:  0.06535,   Adjusted R-squared:  0.03025
F-statistic: 1.862 on 8 and 213 DF,  p-value: 0.06748
```

For comment_count, only 'year' and 'patriotic' show a tendency of positive significant association and other attributes show no statistically significant association.

d

```r
dat <- na.omit(youtube_deid[, c("log_views","funny","show_product_quickly",
                                "patriotic","celebrity","danger","animals","use_sex","year")]

y <- dat$log_views
X <- model.matrix(~ funny + show_product_quickly + patriotic +
                    celebrity + danger + animals + use_sex + year,
                  data = dat)

beta_hat <- solve(t(X) %*% X) %*% (t(X) %*% y)

coef(model_views)
```

```
             (Intercept)                funnyTRUE show_product_quicklyTRUE
            -31.55015804               0.56492445               0.21088918
           patrioticTRUE            celebrityTRUE               dangerTRUE
              0.50699051               0.03547862               0.63131085
             animalsTRUE              use_sexTRUE                     year
             -0.31001838              -0.38670726               0.02053399
```

```
beta_hat
```

```
                                [,1]
(Intercept)              -31.55015804
funnyTRUE                  0.56492445
show_product_quicklyTRUE   0.21088918
patrioticTRUE              0.50699051
celebrityTRUE              0.03547862
dangerTRUE                 0.63131085
animalsTRUE               -0.31001838
use_sexTRUE               -0.38670726
year                       0.02053399
```

The two results are the same.