HW week 12

Jean-Luc Jackson, Christian Montecillo, Richard Robbins

April 2, 2022

Regression analysis of YouTube dataset

You want to explain how much the quality of a video affects the number of views it receives on social media. In a world where people can now buy followers and likes, would such an investment increase the number of views that their content receives? **This is a causal question.**

You will use a dataset created by Cheng, Dale and Liu at Simon Fraser University. It includes observations about 9618 videos shared on YouTube. Please see this link for details about how the data was collected.

You will use the following variables:

- `views`: the number of views by YouTube users.
- `average_rating`: This is the average of the ratings that the video received, it is a renamed feature from `rate` that is provided in the original dataset. (Notice that this is different from `count_of_ratings` which is a count of the total number of ratings that a video has received.
- `length`: the duration of the video in seconds.

```
d <- load_and_clean(input = 'videos.txt')
str(d, max.level=1)
```

```
## tibble [9,618 x 10] (S3: tbl_df/tbl/data.frame)
```

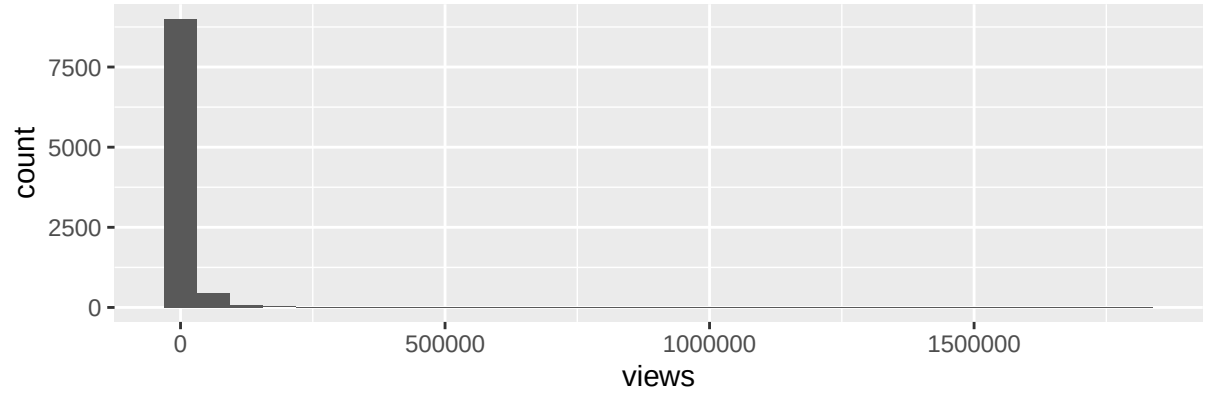
a. Perform a brief exploratory data analysis on the data to discover patterns, outliers, or wrong data entries and summarize your findings.

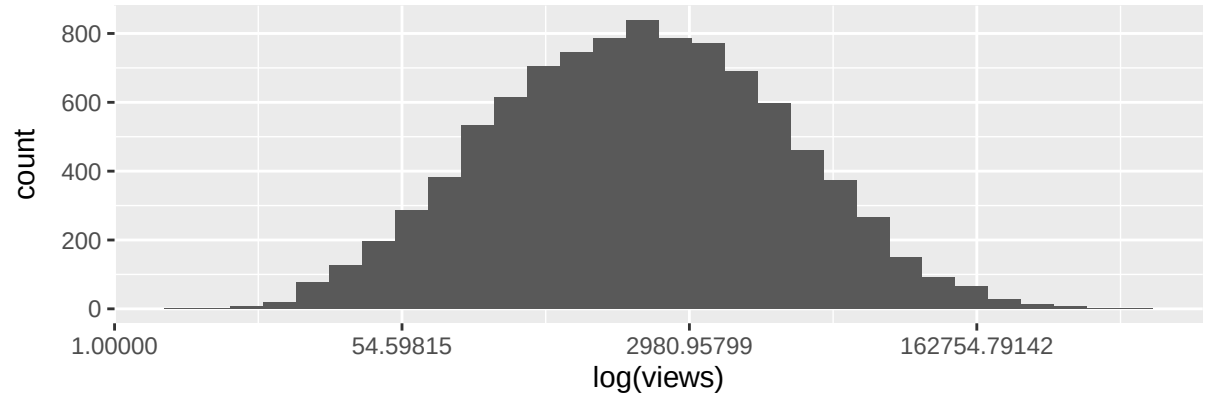
   `views`

   There are 9 rows without a value for `views`. These rows should be discarded. The `views` distribution is especially concentrated with the third quartile at 6,179 and a maximum of 1,807,640. A log tranformation will be appropriate.

```
summary(d$views)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       3     348    1453    9346    6179 1807640       9
```





`average_ratings`

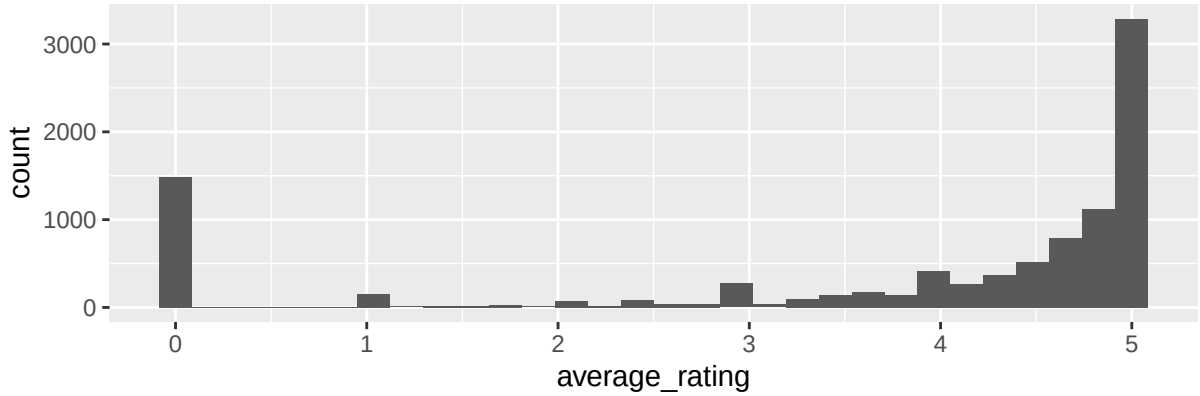
There are 9 rows without a value for `average_ratings`. These rows should be discarded. There are 1,490 rows where the average_rating is 0 and count_of_ratings is also 0. The rating scale was from one to five. A rating of 0 in the database is not meaningful and we should not infer anything from that data. It could simply be a recording error. These rows also should be discarded. After removing these rows, the mean of the distribution is NA, the distribution is heavily skewed left.

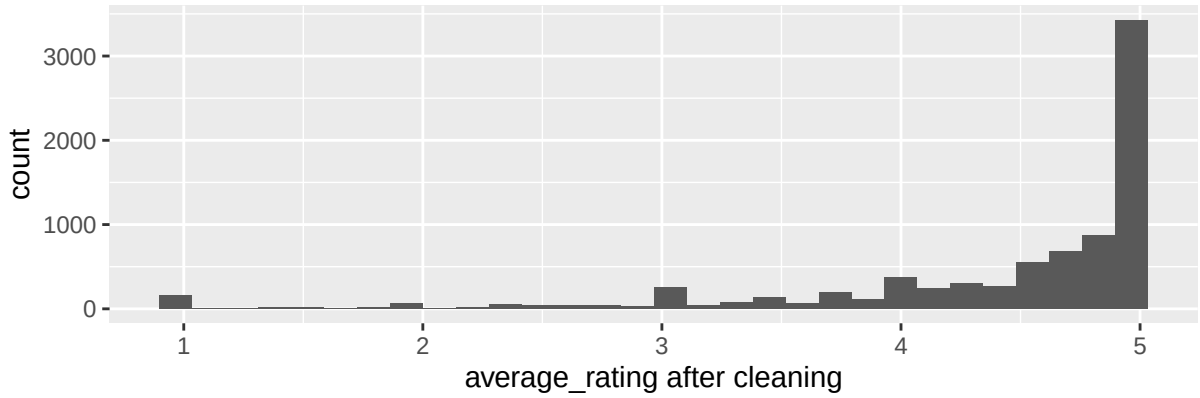
```
nrow(d %>% filter(average_rating == 0 & count_of_ratings == 0))
```

```
## [1] 1490
```

```
summary(filter(d, d$average_rating !=0) %>% select(average_rating))
```

```
##  average_rating
##  Min.   :1.000
##  1st Qu.:4.220
##  Median :4.800
##  Mean   :4.431
##  3rd Qu.:5.000
##  Max.   :5.000
```



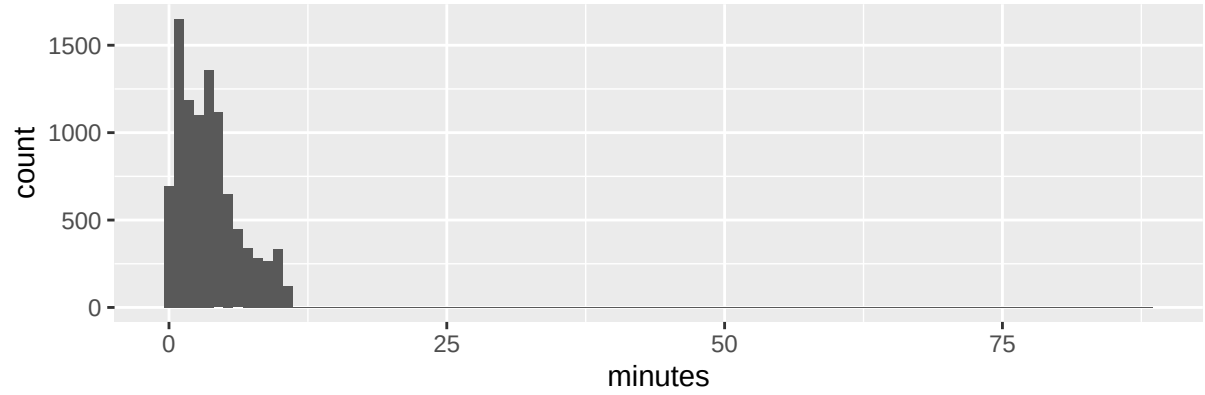


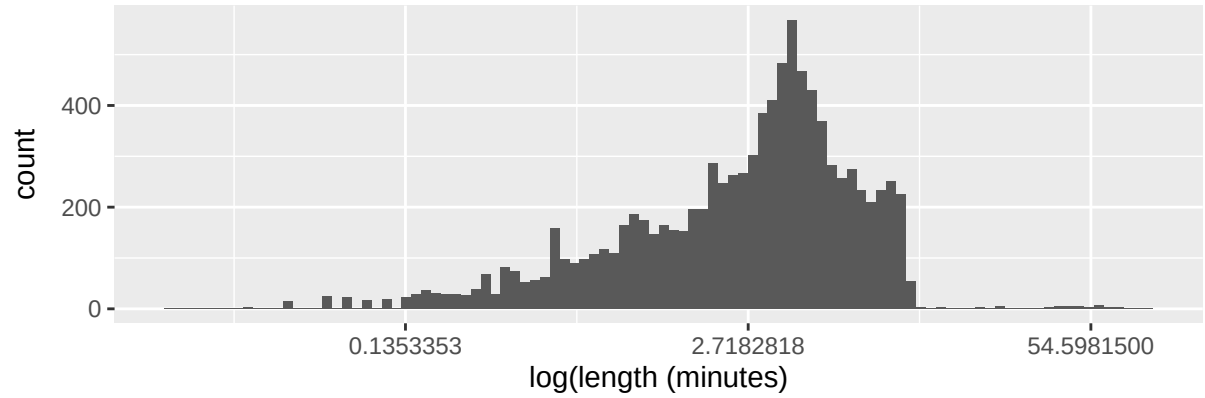
`length`

There are 9 rows without a value for `length`. These rows should be discarded. Most of the videos are relatively brief, the mean is 3.78 minutes and a standard deviation of 3.98 minutes. However, there are small number of comparatively long videos, with the longest being 88.15 minutes.

```
summary(d$length)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##       1      83     193     227     299    5289       9
```





b. Based on your EDA, select an appropriate variable transformation (if any) to apply to each of your three variables. You will fit a model of the type,

$$f(\text{views}) = \beta_0 + \beta_1 g(\text{rate}) + \beta_3 h(\text{length})$$

Where f, g and h are sensible transformations, which might include making *no* transformation.

```
d.filtered <- d %>%
  drop_na(views) %>%
  drop_na(length) %>%
  drop_na(average_rating) %>%
  filter(average_rating > 0)

model <- lm (log(views) ~ average_rating + length, data = d.filtered)

stargazer(
   model,
   type = 'text',
   se = list(get_robust_se(model))
   )
```

```
##
## ================================================
##                         Dependent variable:
##                     ---------------------------
##                              log(views)
## ------------------------------------------------
```

```
## average_rating               0.172***
##                               (0.024)
##
## length                       0.001***
##                              (0.0001)
##
## Constant                     6.775***
##                               (0.105)
##
## ------------------------------------------------
## Observations                    8,119
## R2                              0.013
## Adjusted R2                     0.013
## Residual Std. Error       1.831 (df = 8116)
## F Statistic           53.786*** (df = 2; 8116)
## ================================================
## Note:                *p<0.1; **p<0.05; ***p<0.01
```

c. Using diagnostic plots, background knowledge, and statistical tests, assess all five assumptions of the CLM. When an assumption is violated, state what response you will take. As part of this process, you should decide what transformation (if any) to apply to each variable. Iterate against your model until your satisfied that at least four of the five assumption have been reasonably addressed.

1. **IID Data:**

Independence: Information about the YouTube videos were obtained using a web crawler. The researchers selected initial videos for the web crawler from YouTube's "Recently Featured", "Most Viewed", "Top Rated", and "Most Discussed" video lists. The web crawler only processed distinct videos (skipping those it had already visited), and once information about a video was collected the web crawler selected new videos to process by checking YouTube's provided list of **related** videos. Thus, videos in this dataset are part of a graph network where our initial nodes had a high likelihood of having many `views` and a high `average_rating`. Since initial videos are selected to be popular videos, neighbors to these videos are also likely to be popular. Fortunately, the web crawler continued traversing this graph for a long time, collecting 750,000 videos in five days. Continuing this exploration through related videos for so long helped to collect videos outside of the initial popular cluster and visit those in other neighborhoods. However, the initial popular cluster of videos likely skews this data.

Sampling Distribution: The web crawler was positioned to collect information from the same distribution of videos (starting with popular then traversing related videos) so the data was collected in the same way each time. Also, the videos had identical distributions since each cluster is likely to have similar distributions of `views` – the initial *popular* cluster and the following clusters of neighbors to popular videos.

Thus, **this data is not independent**, though **it is identically distributed**. **We can still conduct this study**, but the results may not extend to the underlying population. We will infer practical implications cautiously with this in mind.

When someone views a video and rates a video they will be aware of how many times the video has been seen and the ratings made by others for that video. That calls independence into question. Moreover, people with common interests and in clustered social groups may influence each other regarding videos. YouTube's algorithms cluster users and emphasize videos commonly viewed by cluster members. These algorithms take into account the creators to whom individuals subscribe. All of these elements of how YouTube operates run against the data being independent.

2. **No Perfect Collinearity:**

Here we investigate whether some combination of features could describe another feature. Our two

features here are `average_rating` and `length`, which could not be used to describe each other. In other words, knowing that a video received a certain average rating would not perfectly indicate the video's length, and vice versa. To check for near-perfect collinearity, we inspect our model's coefficients — `average_rating`: 0.172 and `length`: 0.001. These coefficients do not seem to be "balancing" one another in a shared measurement; their signs are the same and their magnitudes are not similar. Thus, **neither perfect collinearity nor near-perfect collinearity exists.**

No perfect collinearity implies that none of the regressor variables can be written as a linear combination of the others. It is not possible to express `length` or `average rating` as a linear function of the other. There is no perfect collinearity.

3. **Linear Conditional Expectation:**

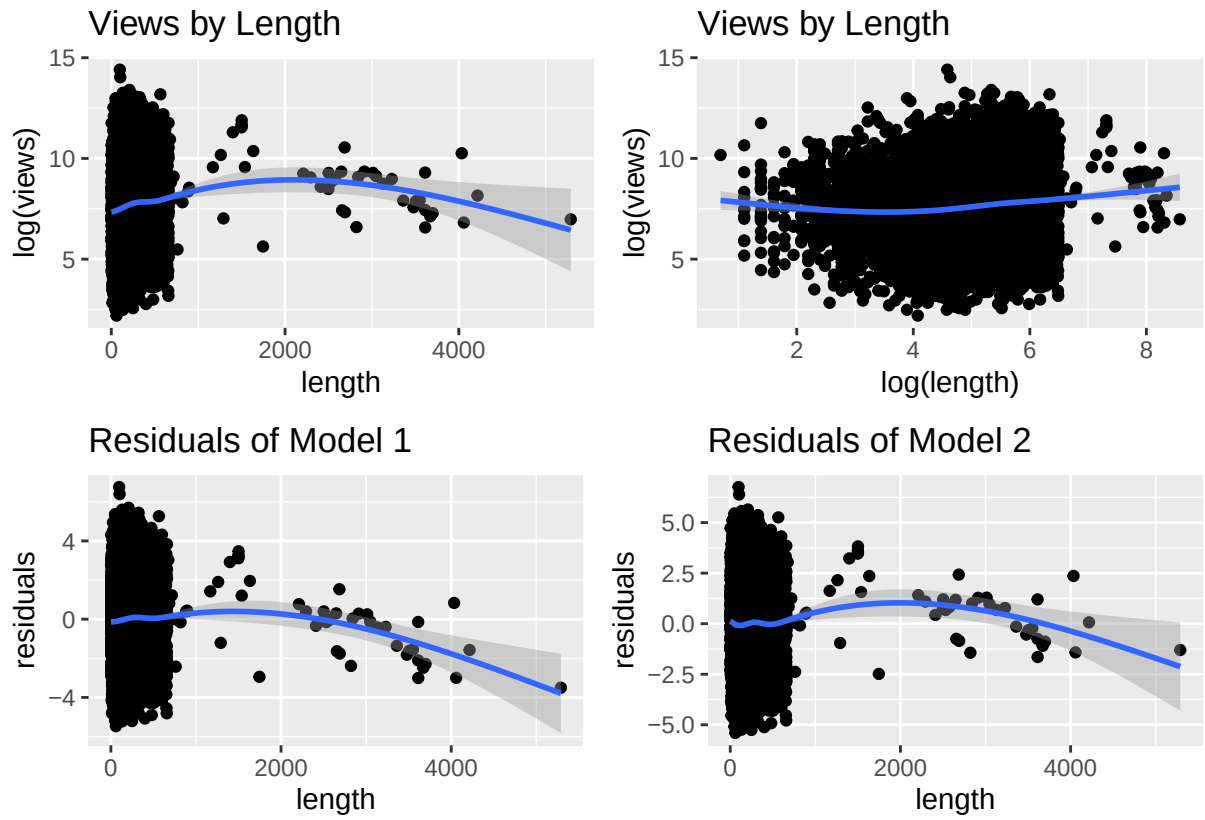
   We first examine the residuals as a function of each variable, and then as a function of the predicted values to investigate if the model is capturing the complexity and form of the data.

```
model_1 <- lm(log(views) ~ average_rating + length,
              data = d.filtered)

model_2 <- lm(log(views) ~ average_rating + log(length),
              data = d.filtered)

df_cond_exp <- d.filtered %>%
  mutate(
    log_views = log(views),
    model_predict_1 = predict(model_1),
    model_resid_1 = resid(model_1),
    model_predict_2 = predict(model_2),
    model_resid_2 = resid(model_2)
  )


# Plotting views by length
length_views_1 <- df_cond_exp %>%
  ggplot() +
  aes(x = length,
      y = log_views) +
  geom_point() +
  geom_smooth() +
  labs(title = "Views by Length",
       x = "length",
       y = "log(views)")

length_views_2 <- df_cond_exp %>%
  ggplot() +
  aes(x = log(length),
      y = log_views) +
  geom_point() +
  geom_smooth() +
  labs(title = "Views by Length",
       x = "log(length)",
       y = "log(views)")

# Plotting views by average rating
rating_views_1 <- df_cond_exp %>%
  ggplot() +
```

```
  aes(x = average_rating,
      y = log_views) +
  geom_point() +
  geom_smooth() +
  labs(title = "Views by Average Rating",
       x = "average_rating",
       y = "log(views)")

rating_views_2 <- df_cond_exp %>%
  ggplot() +
  aes(x = log(average_rating),
      y = log_views) +
  geom_point() +
  geom_smooth() +
  labs(title = "Views by Average Rating",
       x = "log(average_rating)",
       y = "log(views)")


# Plots of residuals as a function of LENGTH

length_plot_1 <- df_cond_exp %>%
  ggplot() +
  aes(x = length,
      y = model_resid_1) +
  geom_point() +
  geom_smooth() +
  labs(title = "Residuals of Model 1",
       x = "length",
       y = "residuals")

length_plot_2 <- df_cond_exp %>%
  ggplot() +
  aes(x = length,
      y = model_resid_2) +
  geom_point() +
  geom_smooth() +
  labs(title = "Residuals of Model 2",
       x = "length",
       y = "residuals")


# Plots of residuals as a function of AVERAGE RATING

rating_plot_1 <- df_cond_exp %>%
  ggplot() +
  aes(x = average_rating,
      y = model_resid_1) +
  geom_point() +
  geom_smooth() +
  labs(title = "Residuals of Model 1",
       x = "average_rating",
       y = "residuals")
```

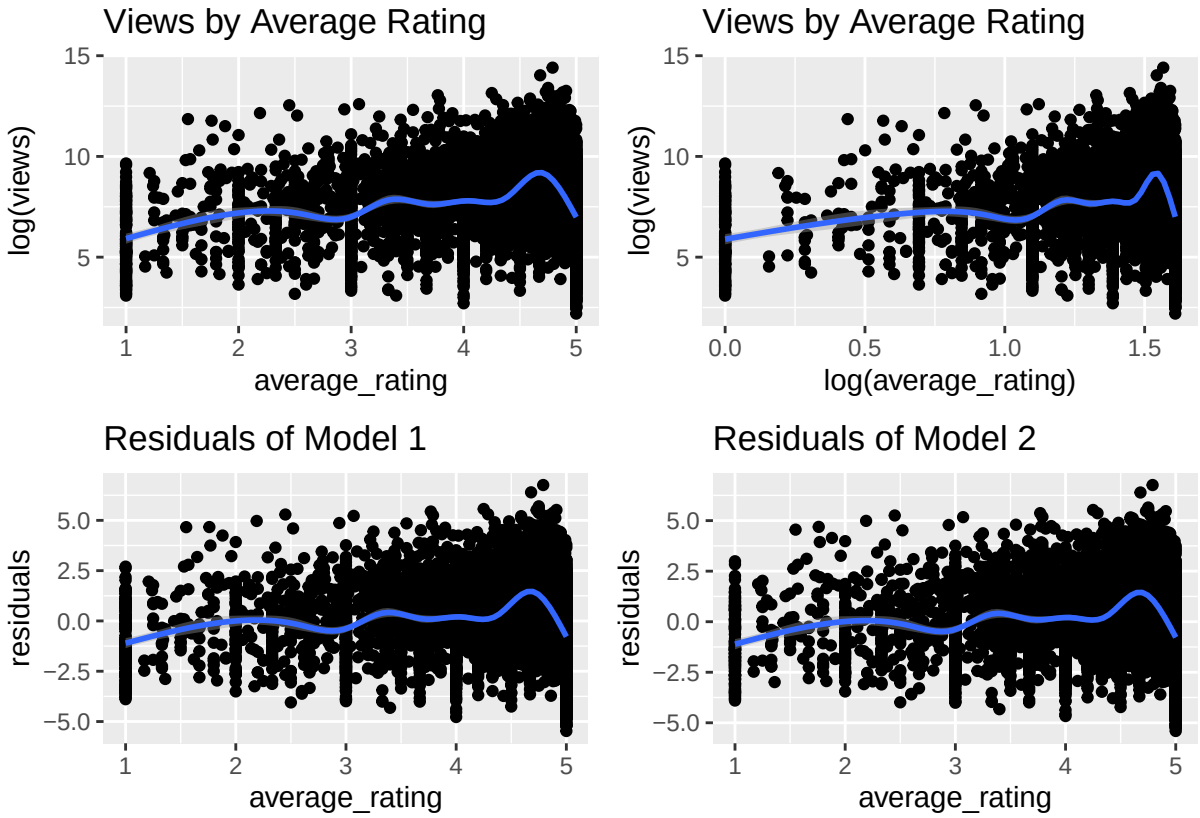
```
rating_plot_2 <- df_cond_exp %>%
  ggplot() +
  aes(x = average_rating,
      y = model_resid_2) +
  geom_point() +
  geom_smooth() +
  labs(title = "Residuals of Model 2",
       x = "average_rating",
       y = "residuals")


(length_views_1 | length_views_2) / (length_plot_1 | length_plot_2)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
(rating_views_1 | rating_views_2) / (rating_plot_1 | rating_plot_2)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



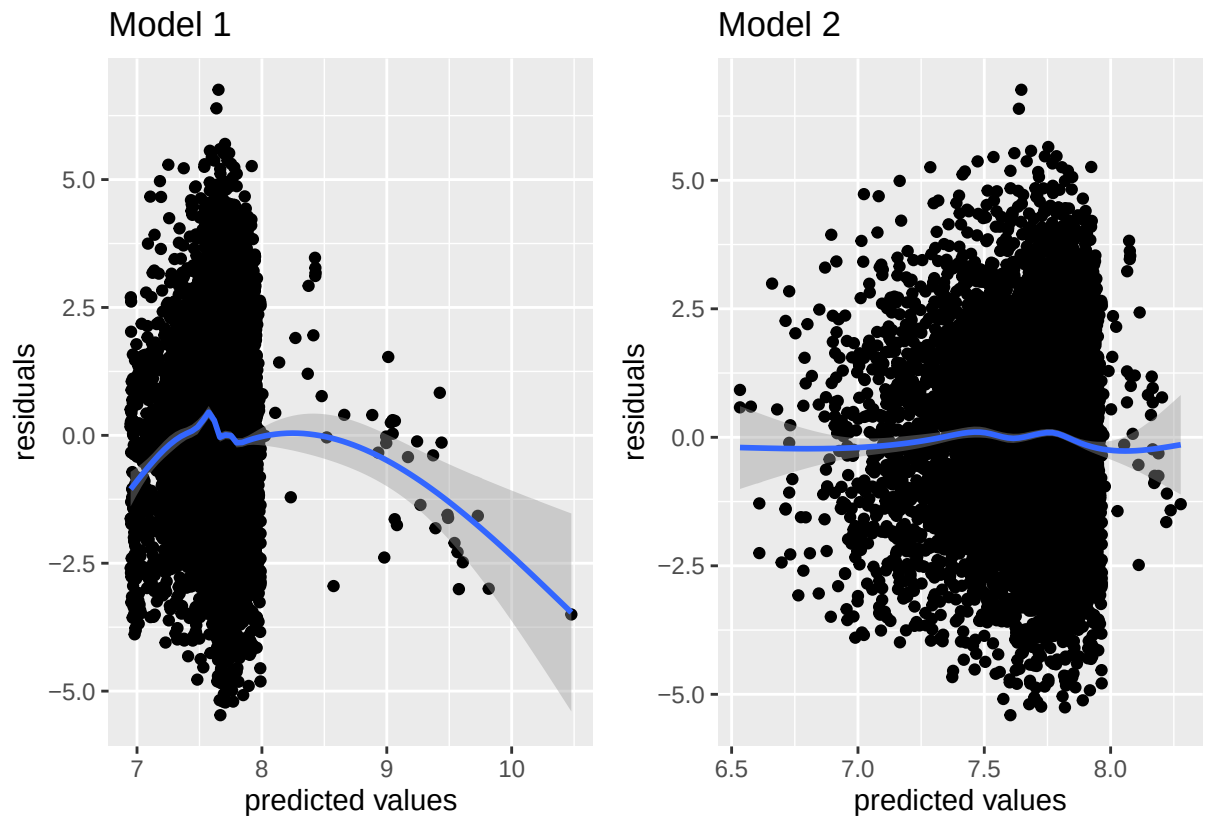
Beginning with the `length` variable, there appears to be nonlinear behavior for higher values of `length`. The second model with `log(length)` is closest to linear and has an acceptable residual level for this application – the largest residual of about 5 equates to 150 views. Both models treat `average_rating` as a linear term and do a decent job of capturing overall behavior, indicated by the horizontal line along zero in the residual plots. There are slight deviations from horizontal but, again, their magnitudes are acceptable for this application.

The two plots below show model residuals as a function of predicted values. The second model demonstrates minimal residuals for the range of predicted values. Combining this with the discussion above of the two variables, the **second model does have a linear conditional expectation**.

```
# Plots of residuals as a function of PREDICTED VALUES

CE_1_plot <- df_cond_exp %>%
  ggplot() +
  aes(x = model_predict_1,
      y = model_resid_1) +
  geom_point() +
  stat_smooth() +
  labs(title = "Model 1",
       x = "predicted values",
       y = "residuals")

CE_2_plot <- df_cond_exp %>%
  ggplot() +
  aes(x = model_predict_2,
```

```
      y = model_resid_2) +
  geom_point() +
  stat_smooth() +
  labs(title = "Model 2",
       x = "predicted values",
       y = "residuals")


(CE_1_plot | CE_2_plot)
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```

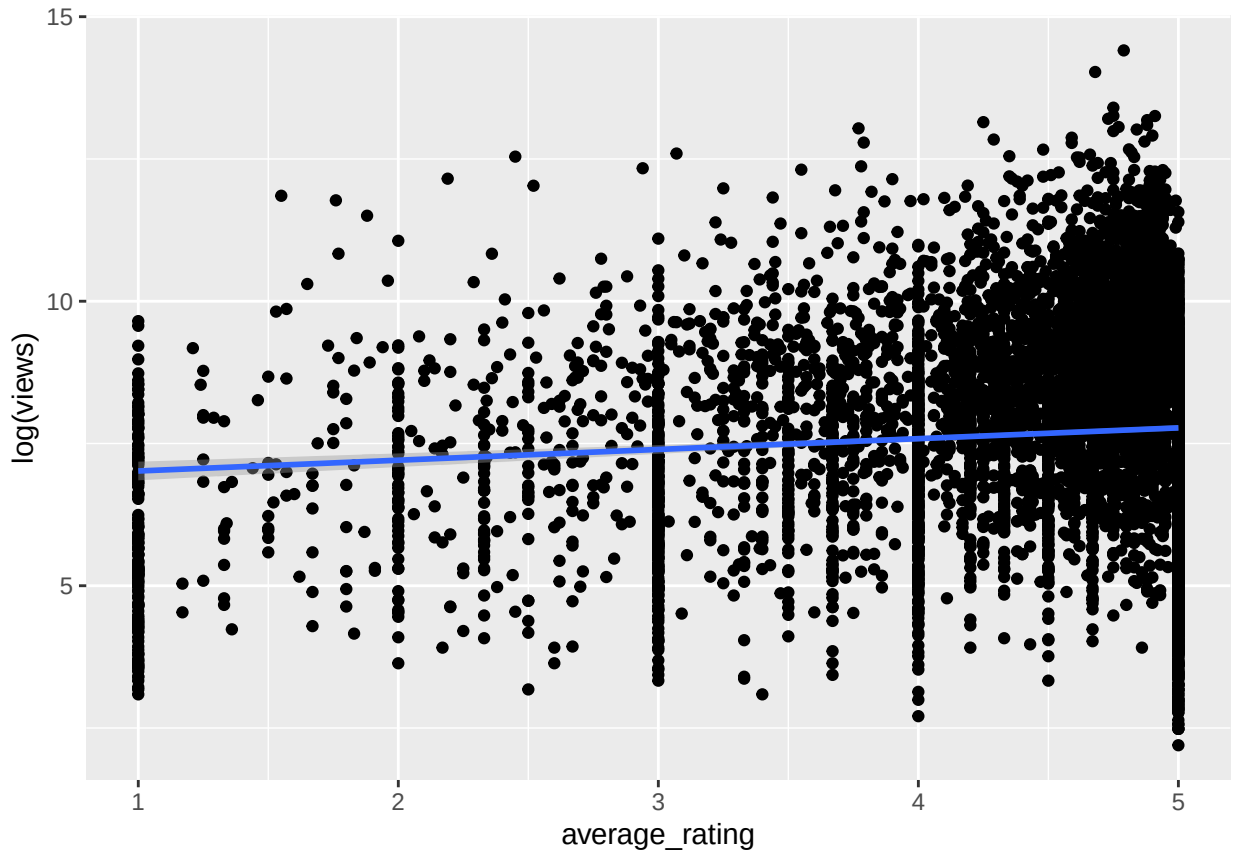


The mean of the residuals is 2.656112e-16 – essentially 0.

```
## remove this commented block and write code that can help you assess whether
## your model satisfied the requirement of a linear conditional expectation.
model_1 <- lm (log(views) ~ average_rating + length, data = d.filtered)

# mutate a new column of residuals of model_1
d.filtered_res <- d.filtered %>%
  mutate(model_1_res = resid(model_1))

plot_1 <- d.filtered_res %>%
  ggplot(aes(x = average_rating, y = log(views))) +
  geom_point() +
  geom_smooth(method = 'lm')
```

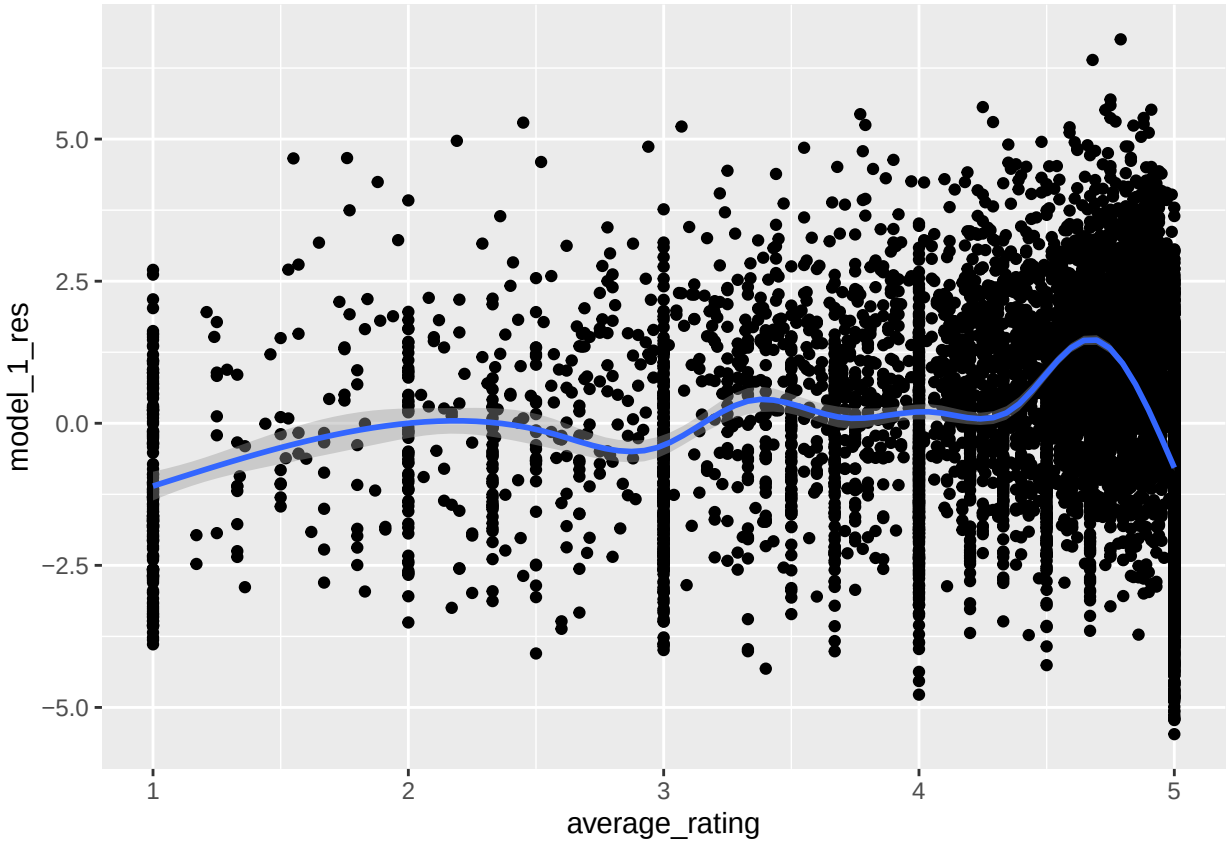
```
plot_2 <- d.filtered_res %>%
  ggplot(aes(x = average_rating, y = model_1_res)) +
  geom_point() +
  stat_smooth(se = TRUE)

plot_1
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
plot_2
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
mean(d.filtered_res$model_1_res) # 2.656112e-16
```

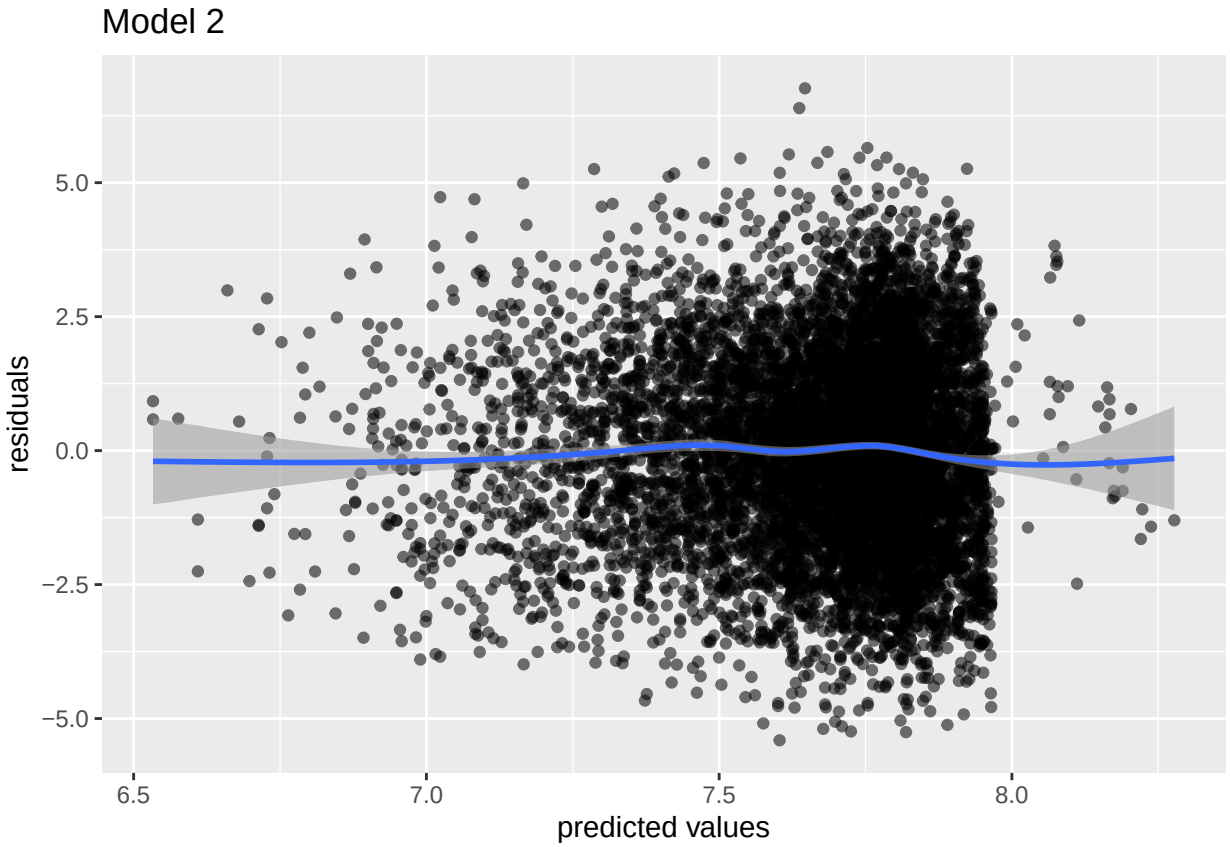
```
## [1] 1.543567e-16
```

4. **Homoskedastic Errors:** 'Replace this quote with what you are looking for when you are assessing homoskedastic errors. Also include what you observe in your plot. In the model that you have chosen to report, do you satisfy the assumption of homoskedastic errors?'

We will now determine if the error variance is constant across all predicting features. We begin by plotting residuals against predicted values.

```
df_cond_exp %>%
  ggplot() +
  aes(x = model_predict_2,
      y = model_resid_2,
      alpha = 0.2) +
  geom_point() +
  stat_smooth() +
  labs(title = "Model 2",
       x = "predicted values",
       y = "residuals") +
  theme(legend.position = 'none')
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



The concern is if residuals vary over the range of the predictions. Considering this plot for Model 2, the residual value does not appear to be a function of the predicted value — the rolling average is horizontal around zero and the range of residuals is nearly constant. We can check this formally by conducting a Breusch-Pagan test:

```
lmtest::bptest(model_2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  model_2
## BP = 10.094, df = 2, p-value = 0.006429
```

The resulting p-value = 0.006, indicating that the null hypothesis of no evidence for heteroskedastic errors should be rejected for the alternative hypothesis that there is evidence of heteroskedasticity. Thus, **we fulfill the requirement for heterskedastic data.**

5. **Normally Distributed Errors:** 'Replace this quote with what you are looking for when you are assessing the normality of your error distribution. Also include what you observe in your plot. In the model that you have chosen to report, do you satisfy the assumption of normally distributed errors?'

Referencing the histogram and quantile-quantile plot below, the residuals are indeed normally distributed. There are slight deviations at the tails – the Q-Q plot shows "thin tails" – indicating that the residuals are more concentrated in the center and less in the tails. In other words, there are fewer extreme values of residuals compared to the theoretical normal distribution.

```
## remove this commented block and write code that can help you assess whether
## your model satisfied the requirement of normally distributed errors
```

```
resid_hist <- df_cond_exp %>%
  ggplot() +
  aes(x = model_resid_2) +
  geom_histogram(bins = 100,
                 fill = 'white',
                 color = 'blue') +
  labs(title = 'Histogram of Residuals',
       x = 'Model 2 Residuals')

resid_qq <- df_cond_exp %>%
  ggplot() +
  aes(sample = model_resid_2) +
  stat_qq() + stat_qq_line() +
  labs(title = 'Q-Q Plot of Residuals',
       x = 'Theoretical Quantiles',
       y = 'Sample Quantiles')

(resid_hist | resid_qq)
```