# Problem Set 5

## Regression

Alex Tomack

Due Date: 2023-02-24

# Getting Set Up

Open `RStudio` and create a new RMarkDown file ( `.Rmd` ) by going to `File -> New File -> R Markdown...` . Accept defaults and save this file as `[LAST NAME]_ps5.Rmd` to your `code` folder.

Copy and paste the contents of this file into your `[LAST NAME]_ps5.Rmd` file. Then change the `author: [YOUR NAME]` (line 4) to your name.

All of the following questions should be answered in this `.Rmd` file. There are code chunks with incomplete code that need to be filled in.

This problem set is worth 10 total points, plus five extra credit points. The point values for each question are indicated in brackets below. To receive full credit, you must both have the correct code **and include a comment describing what each line does**. In addition, some questions ask you to provide a written response in addition to the code. Unlike the first two problem sets, some of the code chunks are totally empty, requiring you to try writing the code from scratch. Make sure to comment each line, explaining what it is doing!

You are free to rely on whatever resources you need to complete this problem set, including lecture notes, lecture presentations, Google, your classmates…you name it. However, the final submission must be complete by you. There are no group assignments. To submit, compiled the completed problem set and upload the PDF file to Brightspace by midnight on 2023/02/24.

**Good luck!**

# Question 0

Require `tidyverse` and load the `mv.Rds` (https://github.com/jbisbee1/DS1000_S2023/blob/main/Lectures/5_Regression/data/mv.Rds?raw=true') data to an object called `movies` . (Tip: use the `read_rds()` function with the link to the raw data.)

```
require(tidyverse)
```

```
## Loading required package: tidyverse
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✓ ggplot2 3.4.0      ✓ purrr   1.0.0
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.5.0
## ✓ readr   2.1.3      ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
require(plotly)
```

```
## Loading required package: plotly
##
## Attaching package: 'plotly'
##
## The following object is masked from 'package:ggplot2':
##
##     last_plot
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following object is masked from 'package:graphics':
##
##     layout
```

```
require(scales)
```

```
## Loading required package: scales
##
## Attaching package: 'scales'
##
## The following object is masked from 'package:purrr':
##
##     discard
##
## The following object is masked from 'package:readr':
##
##     col_factor
```

```
movies<-read_rds("https://github.com/jbisbee1/DS1000_S2023/blob/main/Lectures/5_Regression/data/mv.Rds?raw=true")
```

# Question 1 [1 point]

In this problem set, we will answer the following research question: do movies that score higher among audiences ( `score` ) make more money ( `gross` ). First, write out a **theory** that answers this question and transform it into a **hypothesis**.

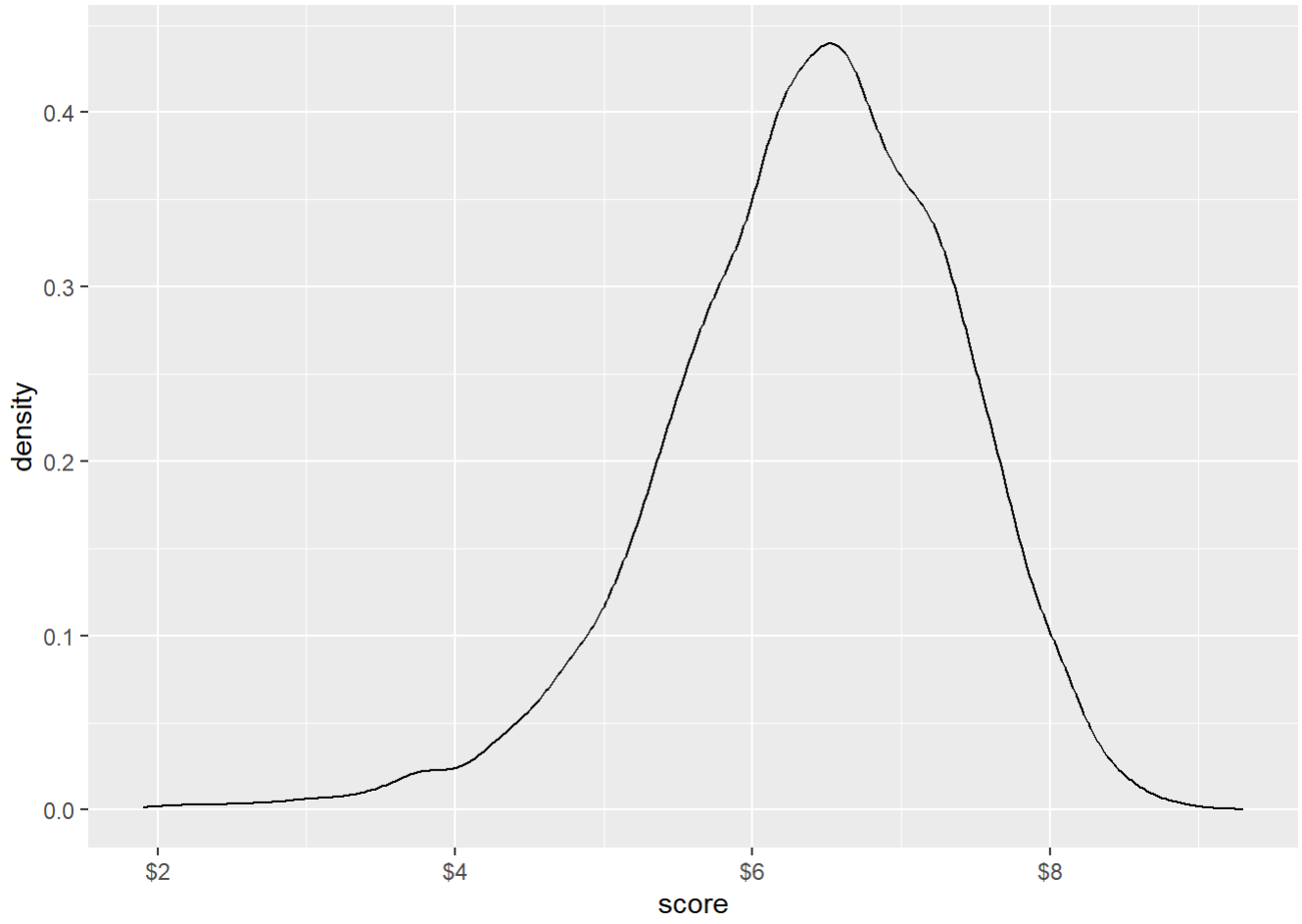
> Question: Do movies that score higher among audiences make more money? Theory: Higher scoring movies make more money. Hypothesis: There is a positive relationship between how score and money made.

# Question 2 [1 point]

Based on your theory, which variable is the $X$ variable (i.e., the independent variable or the predictor)? Which variable is the $Y$ variable (i.e., the dependent variable or the outcome)? Use **univariate** visualization to create two plots, one for each variable. Do you need to apply a log-transformation to either of these variables? Why?

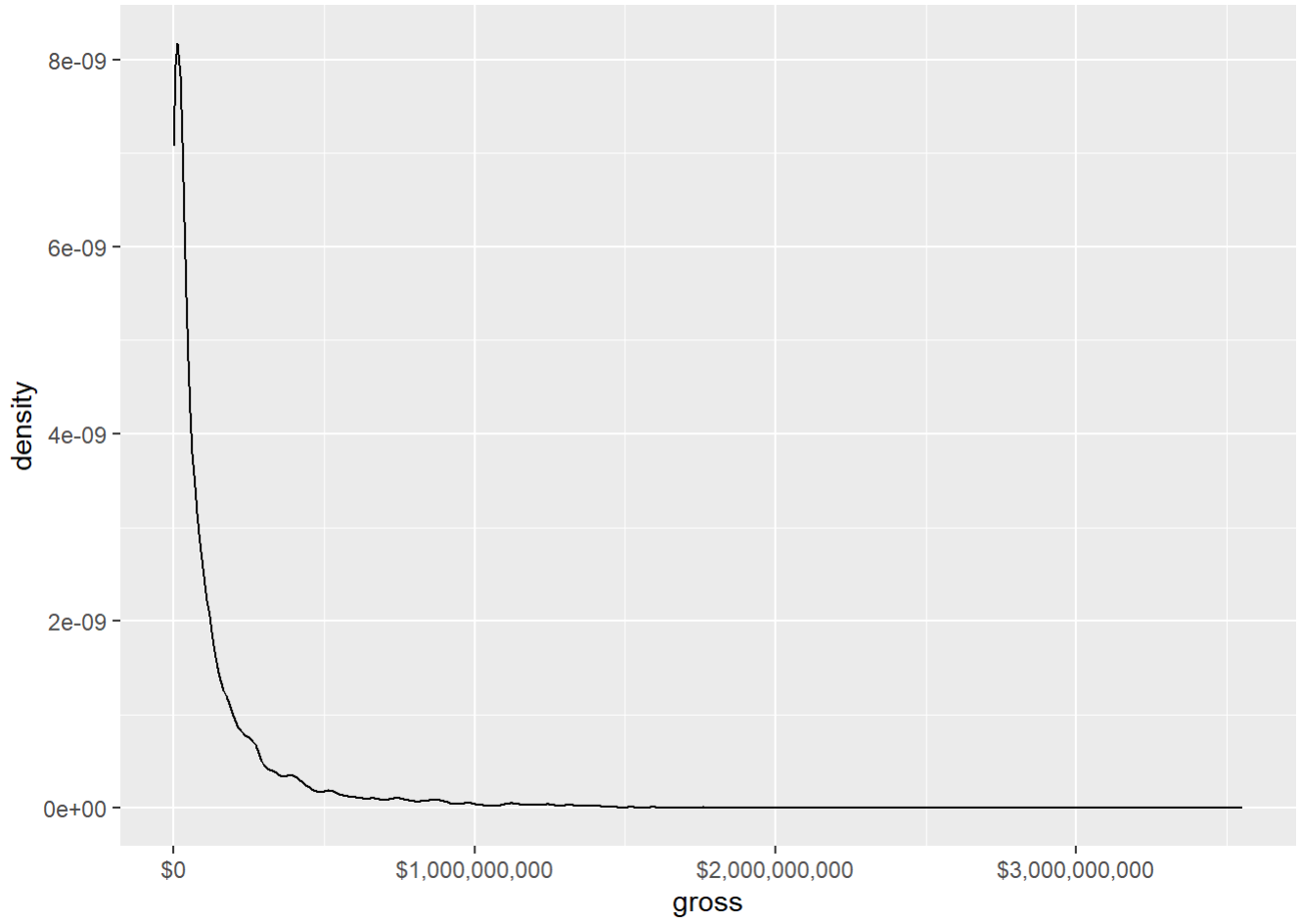
```
#If we want to see visually whether we'll have to apply a log function, lets take a look at the
univariate analysis for each.
# score-- we can see there's no necessary manipulation by a log function for this variable.
movies %>%
  ggplot(aes(score))+
  geom_density()+
  scale_x_continuous(labels=dollar_format())
```

```
## Warning: Removed 3 rows containing non-finite values (`stat_density()`).
```



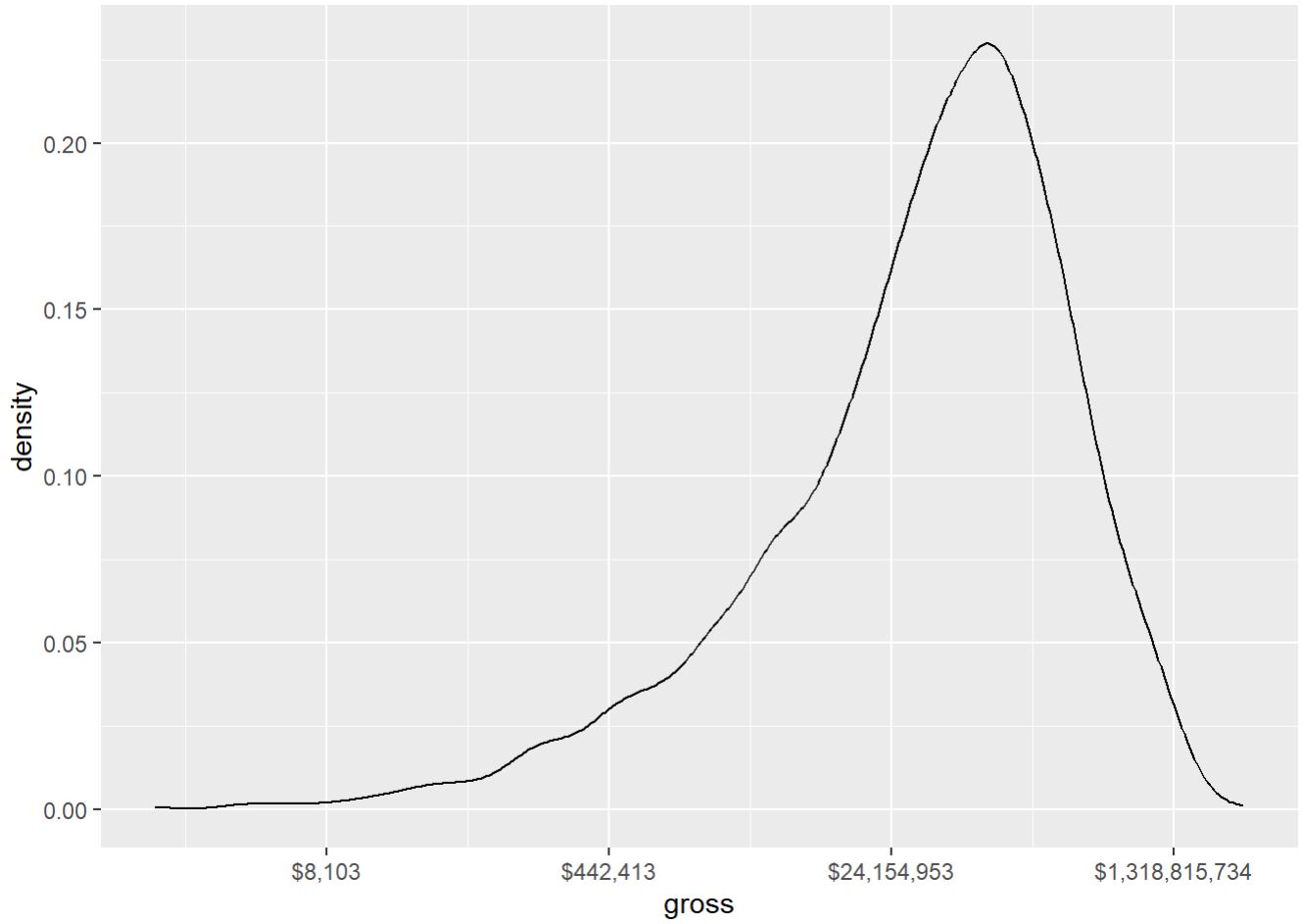

```
# gross-- we have a clear natural log function, so for this variable we need to log linearize.
movies %>%
  ggplot(aes(gross))+
  geom_density()+
  scale_x_continuous(labels=dollar_format())
```

```
## Warning: Removed 3668 rows containing non-finite values (`stat_density()`).
```




```
#gross log-scaled, we can see its less right skewed and a bit closer to normal.
movies %>%
  ggplot(aes(gross))+
  geom_density()+
  scale_x_continuous(trans="log", labels=dollar_format())
```

```
## Warning: Removed 3668 rows containing non-finite values (`stat_density()`).
```

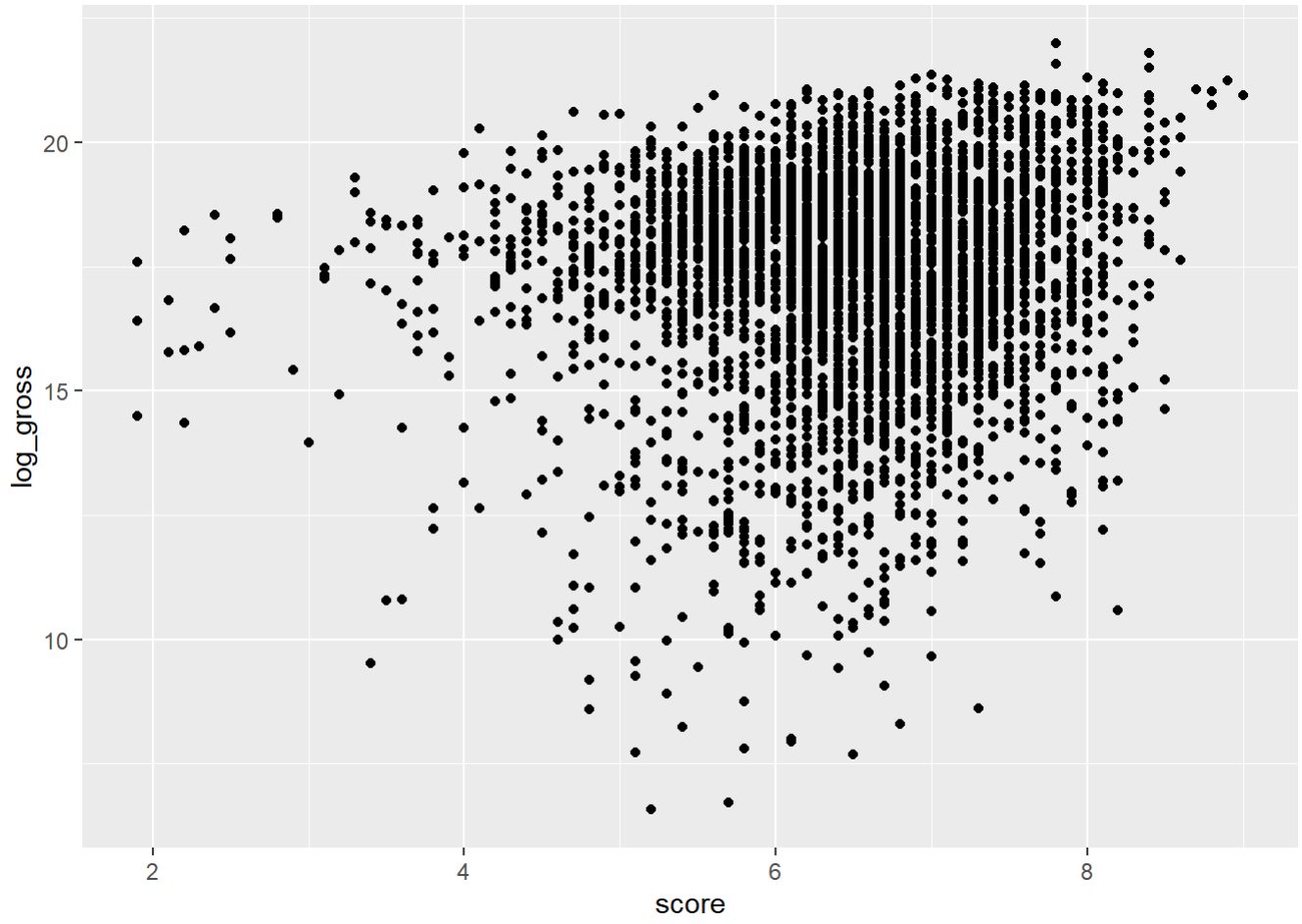



We'll need to apply log function to the 'gross' variable because of the presence of e. The presence of the Euler's constant means the variable is on an exponential scale, and we want to transform it to a log scale so it can conform to normality.

# Question 3 [2 points]

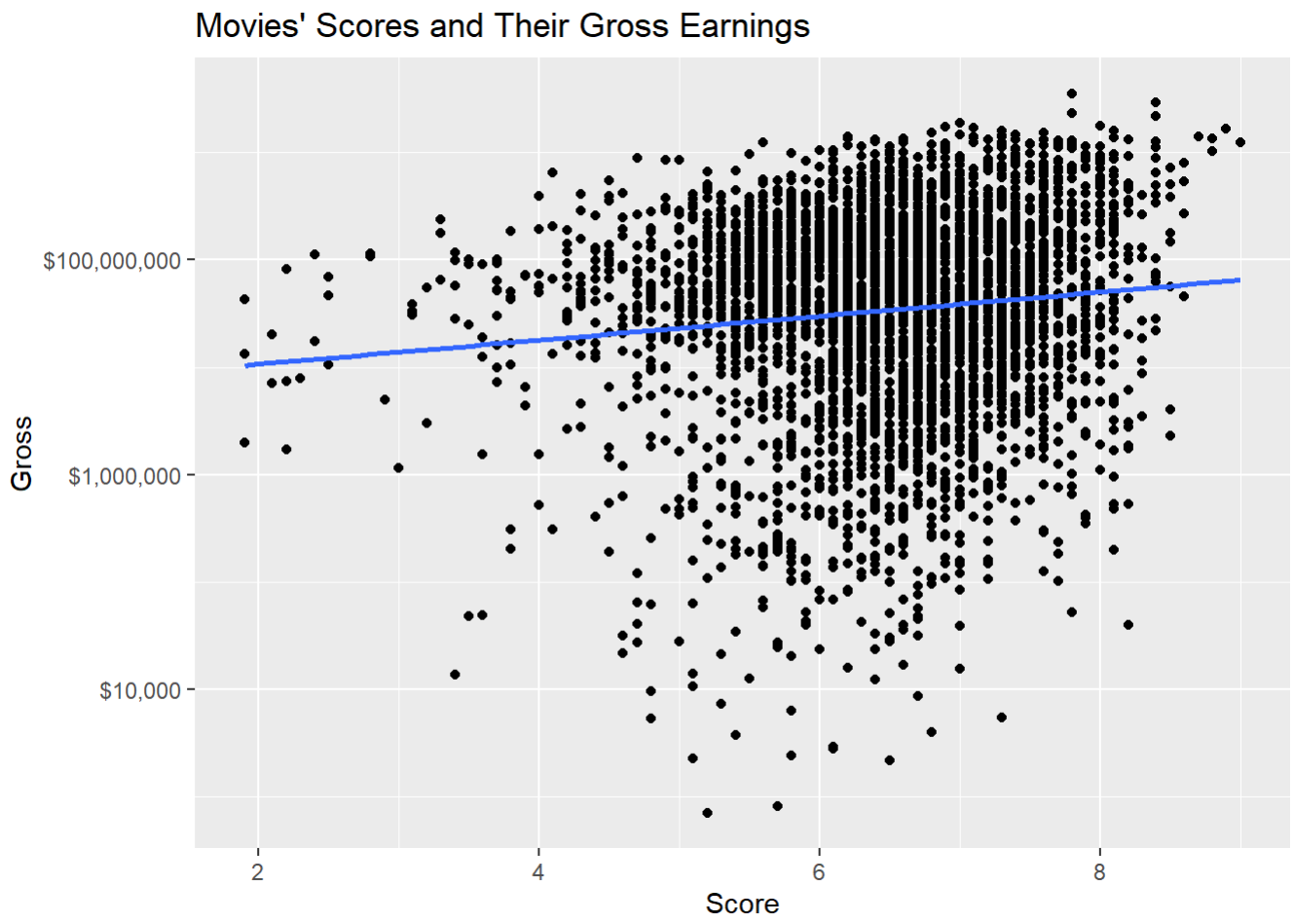
Now create a multivariate visualization of these two variables, making sure to put the independent variable on the x-axis and the dependent variable on the y-axis. Add the line of best fit. Make sure to log the data if you determined this was necessary in the previous question! Does the visualization support your hypothesis?

```
#drop nas
movies_analysis<-movies%>%mutate(log_gross=log(gross))%>%drop_na(gross, score)
# multivariate analysis
movies_analysis%>%
  ggplot(aes(x=score, y=log_gross))+
  geom_point()
```

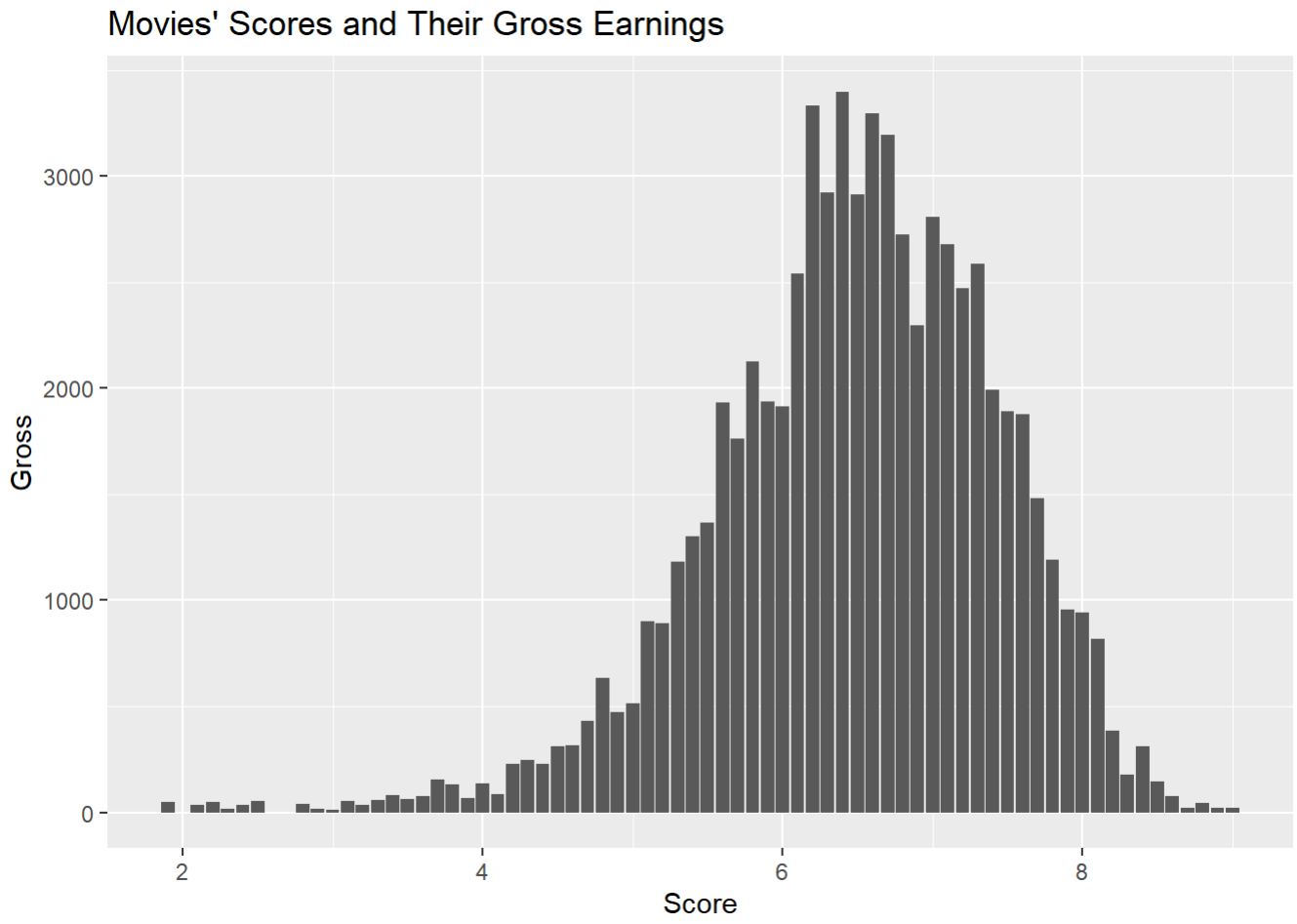



```
#multivariate analysis-- we want to see how score impacts gross, so our dependent variable is gr
oss, and independent is score
movies_analysis%>%
  ggplot(aes(x=score,y=gross))+
  geom_point()+
  scale_y_continuous(trans="log",
                     labels=label_dollar(),
                     breaks=log_breaks())+
  labs(title="Movies' Scores and Their Gross Earnings",
       x="Score",
       y="Gross")+
  geom_smooth(method="lm", se=F)
```

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




```
#But this doesn't look great, let's look as a bar
movies_analysis %>%
  ggplot(aes(x=score, y=log_gross))+
  geom_bar(stat="identity")+
  labs(title="Movies' Scores and Their Gross Earnings",
       x="Score",
       y="Gross")
```




Yes, our linear model supports our hypothesis! We see a moderately positive association between the two variables of score and gross.

# Question 4 [2 points]

Now estimate the regression using the `lm()` function. Describe the output of the model in English, talking about the intercept, the slope, and the statistical significance.

```
model_gross_score <- lm(formula = log_gross ~ score, # Write the regression formula here
                        data = movies_analysis)

summary(model_gross_score)
```

```
##
## Call:
## lm(formula = log_gross ~ score, data = movies_analysis)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4413  -1.1336   0.4662   1.5388   4.3091
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 15.67373    0.23785  65.897  < 2e-16 ***
## score        0.25745    0.03639   7.075 1.76e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.197 on 4003 degrees of freedom
## Multiple R-squared:  0.01235,    Adjusted R-squared:  0.0121
## F-statistic: 50.05 on 1 and 4003 DF,  p-value: 1.762e-12
```

We observe when the score is 0, the gross is 16m dollars. This doesn't exactly makes sense, though, since films that score poorly shouldn't gross that much money. An increase in 1 unit of the movie's score is associated with a $26m increase in gross earnings, and the score variable is statistically significant to gross earnings– we're 99.999% confident.

# Question 5 [2 points]

Now calculate the model's prediction errors and create both a univariate and multivariate visualization of them. Based on these analyses, would you say that your model does a good job predicting how much money a movie makes? **Make sure to reference both the univariate and multivariate visualization of the errors!**

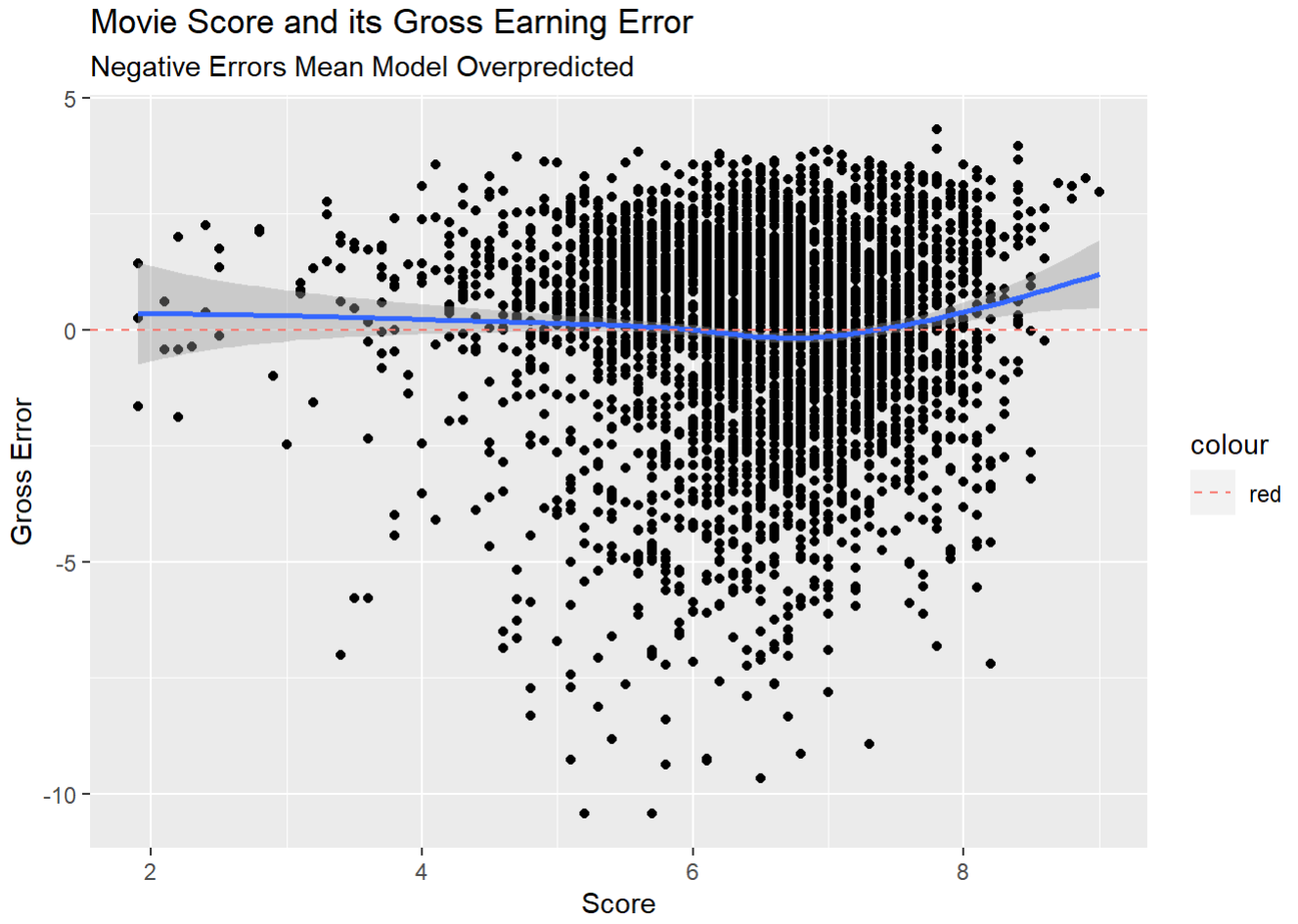
```
movies_analysis <- movies_analysis %>%
  mutate(preds = predict(model_gross_score)) %>% # Add the predicted values from the model as it
s own column called "preds"
  mutate(errors = log_gross-preds) # Calculate the errors as Y - predicted Y

# Look at data
glimpse(movies_analysis%>%select(errors)) # continuous
```

```
## Rows: 4,005
## Columns: 1
## $ errors <dbl> 0.4075826, 0.1606562, 2.5376329, 0.8718073, -1.5535898, 0.11057…
```

```
# Univariate
movies_analysis %>%
  ggplot(aes(x=errors)) + # Visualize the errors
  geom_density(alpha=0.4) + # Choose the best geom_... for this data
  geom_vline(data=movies_analysis, aes(xintercept=0, color="red"), linetype="dashed") + # Add a vertical dashed line at zero
  labs(title = 'Gross Earnings Error in Movies', # Write clear labels for the title, subtitle, and axes
       subtitle = 'Observed minus predicted',
       x = 'Error',
       y = 'Proportion of Movies')
```

```
# Multivariate
movies_analysis %>%
  ggplot(aes(x=score, y=errors)) + # Visualize the audience score and the model's errors
  geom_point() + # Choose the best geom_... for this data
  geom_smooth() + # Add a line of best fit (this can be curvey)
  geom_hline(data=movies_analysis, aes(yintercept=0, color="red"), linetype="dashed") + # Add a horizontal dashed line at zero
  labs(title = 'Movie Score and its Gross Earning Error', # Write clear labels for the title, subtitle, and axes
       subtitle = 'Negative Errors Mean Model Overpredicted',
       x = 'Score',
       y = 'Gross Error')
```

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




Our model does a decent job, but looking to the univariate analysis it could be much better. Our error data is centered around zero, but we can notice an obvious leftward skew, indicating we frequently overestimate how much a movie will gross. In terms of the multivariate analysis, we observe the same trend. As score rises, our model frequently overestimates the gross earnings, with most of the data points falling below our regression line.

# Question 6 [2 points]

Calculate the RMSE in the full data. Then calculate the RMSE using 100-fold cross validation with an 80-20 split and take the average of the 100 estimates. Which value is larger? Why?

```
# RMSE Full Data
movies__analysis<-movies_analysis %>%
  mutate(se=errors^2) %>% # Calculate the squared errors
  mutate(mse=mean(se))%>%# Calculate the mean squared errors
  mutate(rmse=sqrt(mse)) # Calculate the square root of the mean squared errors

# RMSE 100-fold CV
set.seed(123)
cvRes <- NULL # Instantiate an empty object to store data from the loop
for(i in 1:100) { # Loop 100 times
  inds <- sample(x=1:nrow(movies_analysis), # Sample from the row numbers of the movies_analysis
dataframe
                 size=round(nrow(movies_analysis)*0.8),# Set the size to be 80% of the total row
s (don't forget to round()!)
                 replace=F) # Sample WITHOUT replacement

  #create training and test sets
  train <- movies_analysis %>% slice(inds) # Use the 80% to get the training data
  test <- movies_analysis %>% slice(-inds) # Drop the 80% to get the test data

  # run model
  m <- lm(formula = log_gross~score, # Write the regression formula here
          data = train) # Train the model on the train data

  # calculating RMSE
  test$preds <- predict(m, newdata=test) # Generate predicted values from the model

  e <- test$log_gross-test$preds # Calculate the errors as the true Y minus the predicted Y
  se <- e^2 # Square the errors
  mse <- mean(se)# Take the mean of the squared errors
  rmse <- sqrt(mse)# Take the square root of the mean of the squared errors
  cvRes <- c(rmse, cvRes) # Append the rmse to the cvRes object
}

mean(cvRes) # Calculate the average RMSE from 100-fold cross validation
```

```
## [1] 2.198704
```

The RMSE is larger in the 100 sample cross-validation than from earlier, but this is just a feature of CV– it doesn't mean we're overconfident. By employing the CV method, we ensure that we don't overfit our regression model to the data.

# Question 7 [5 EC Points]

Using the same process as described in the preceding questions, answer the following research question: “Do movies that have a higher Bechdel Score ( `bechdel_score` ) make more money ( `gross` )?” Make sure to include:

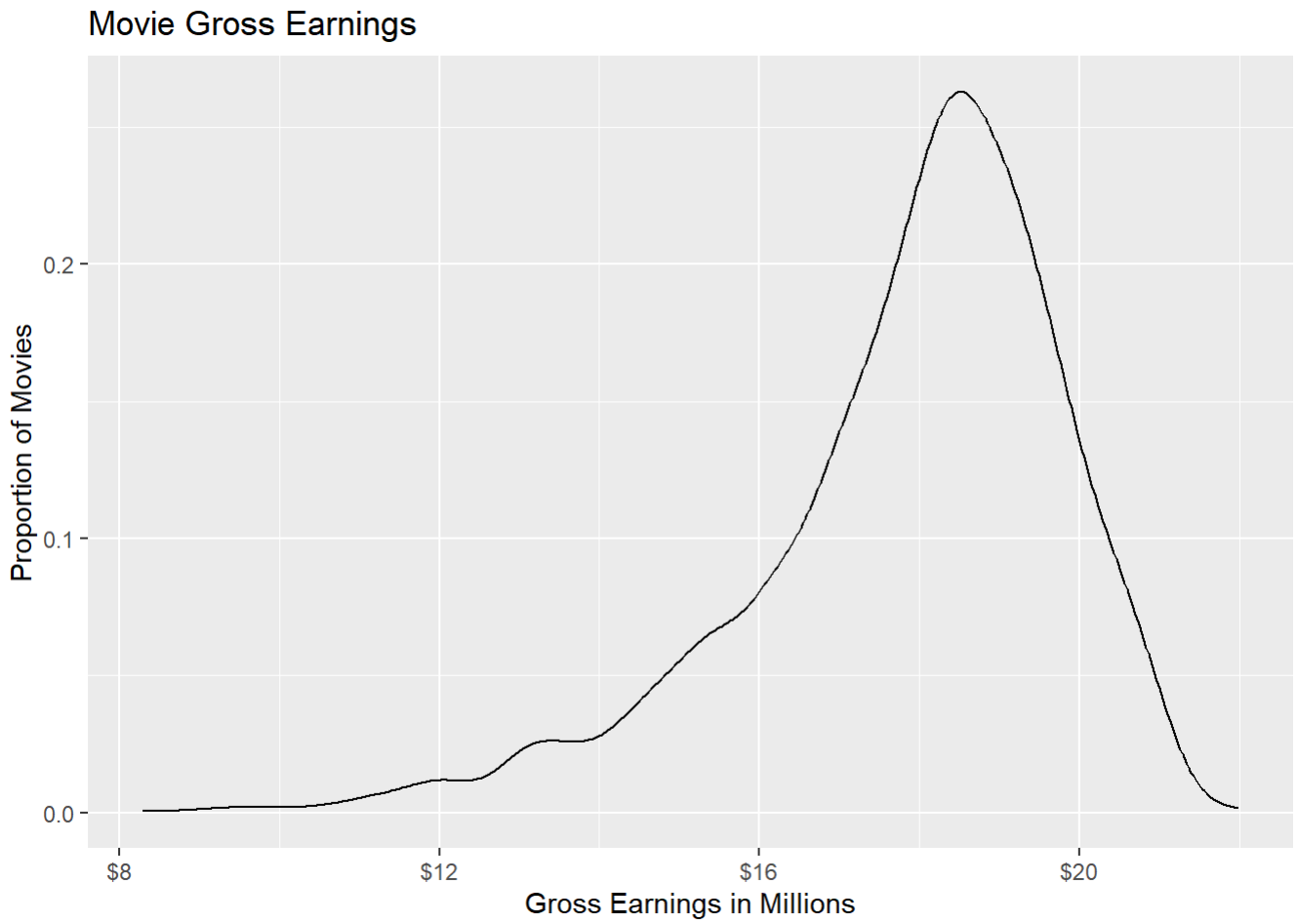
- A theory and hypothesis (write-up required: 2 to 3 sentences)
- Univariate AND multivariate visualizations of both the $X$ and $Y$ variables (no write-up required)
- A regression model using the `lm()` function (write-up required)
- Univariate and multivariate visualizations of the errors (write-up required: 2 to 3 sentences)
- Analysis of RMSE using 100-fold cross validation with an 80-20 split (no write-up required)

> Theory: The higher a movie scores on the bechdel test, the lower its gross earnings will be. Hypothesis: There is a negative relationship between the bechdel test and gross earnings.

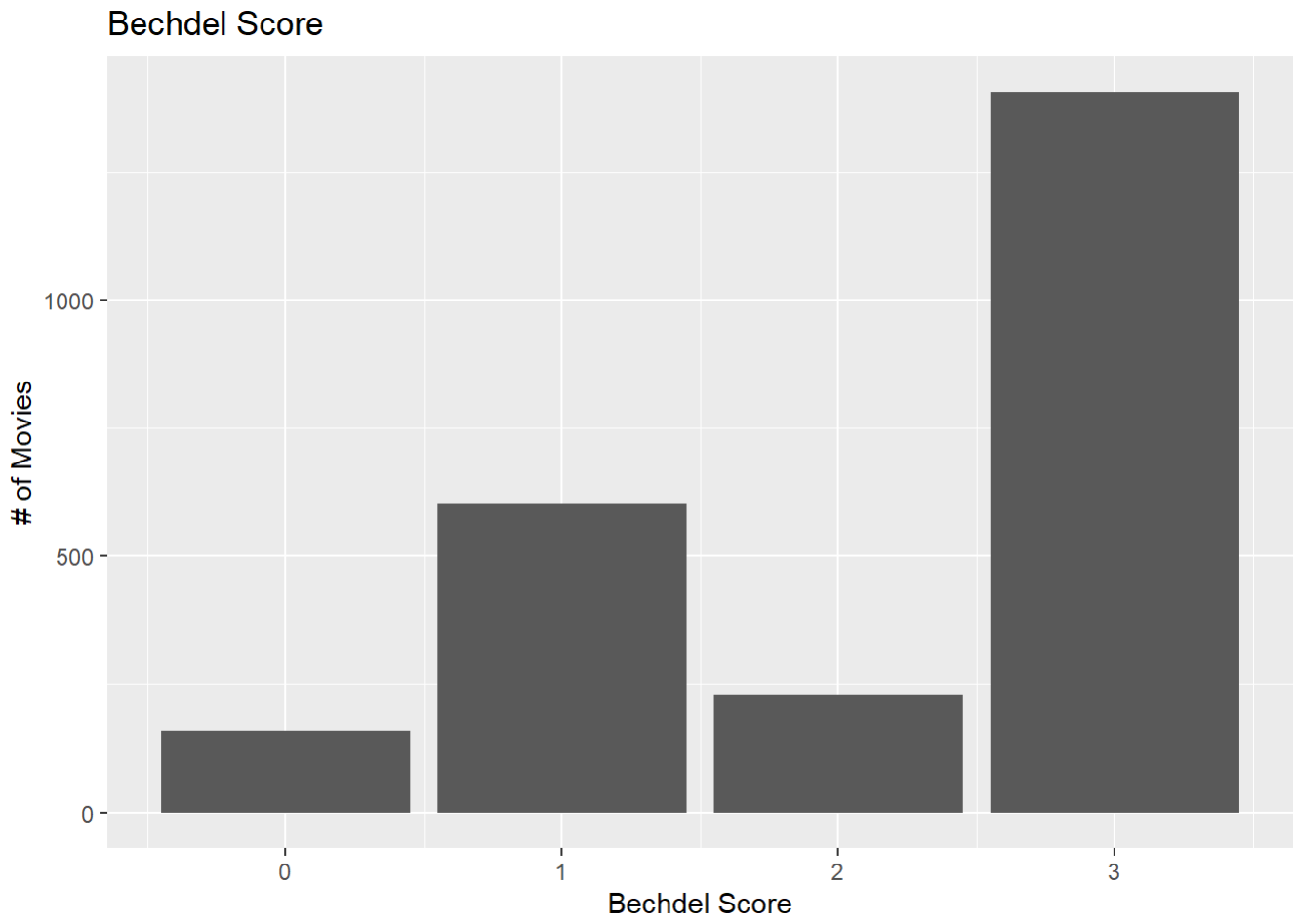
```
# Let's look at our data first.
glimpse(movies_analysis%>%select(bechdel_score, gross))
```

```
## Rows: 4,005
## Columns: 2
## $ bechdel_score <dbl> 3, 3, 0, NA, 3, 1, NA, 2, 3, 2, 3, 1, 2, 3, 3, 0, NA, NA…
## $ gross         <dbl> 73677478, 53278578, 723586629, 129918034, 11490339, 6226…
```

```
#drop missingness
movies_analysis<-movies_analysis %>% drop_na(gross,bechdel_score)

# Univariate of Gross
movies_analysis %>%
  ggplot(aes(x=log_gross))+
  geom_density()+
  labs(title="Movie Gross Earnings",
       x="Gross Earnings in Millions",
       y="Proportion of Movies")+
  scale_x_continuous(labels=label_dollar())
```




```
# Univariate of Bechdel Score
movies_analysis %>%
  ggplot(aes(x=bechdel_score))+
  geom_bar()+
  labs(title="Bechdel Score",
       x="Bechdel Score",
       y="# of Movies")
```




```
# INSERT CODE FOR MULTIVARIATE VISUALIZATION HERE
movies_analysis %>%
  ggplot(aes(x=factor(bechdel_score), y=log_gross))+
  geom_violin()+
  labs(title="Movie Bechdel Scores and Their Gross Earnings",
       x="Score",
       y="Gross")+
  scale_y_continuous(labels=label_dollar())
```


```
# Since we already removed missingness and logged our gross variable, we can move straight to es
timating our regression
model1<-lm(log_gross~bechdel_score, data=movies_analysis)
summary(model1)
```

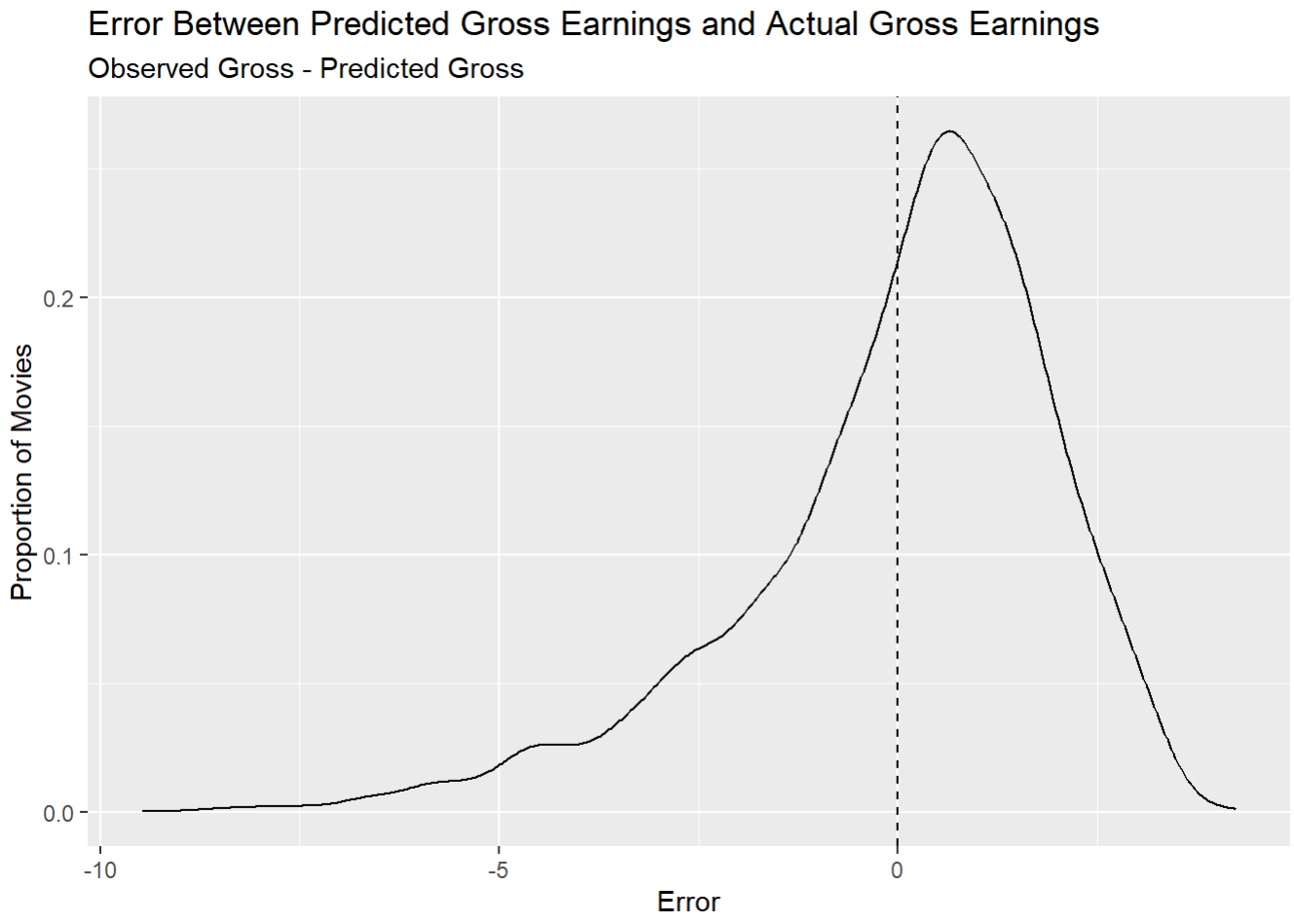
```
##
## Call:
## lm(formula = log_gross ~ bechdel_score, data = movies_analysis)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.4747 -0.9196  0.3796  1.3447  4.2295
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   18.04380    0.09552 188.903   <2e-16 ***
## bechdel_score -0.09411    0.03927  -2.397   0.0166 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.981 on 2395 degrees of freedom
## Multiple R-squared:  0.002393,   Adjusted R-squared:  0.001976
## F-statistic: 5.744 on 1 and 2395 DF,  p-value: 0.01662
```

For a bechdel score of 0, the gross earnings of a film are $18m, and every unit increase in the bechdel score is associated with a decrease in gross earnings by $8.97m.

```
# INSERT CODE TO VISUALIZE THE ERRORS HERE
# find errors first--
movies_analysis$predictions<-predict(model1)
movies_analysis$errors<-movies_analysis$log_gross-movies_analysis$predictions

# take a look at them
movies_analysis%>%
  select(bechdel_score,log_gross, predictions, errors)
```

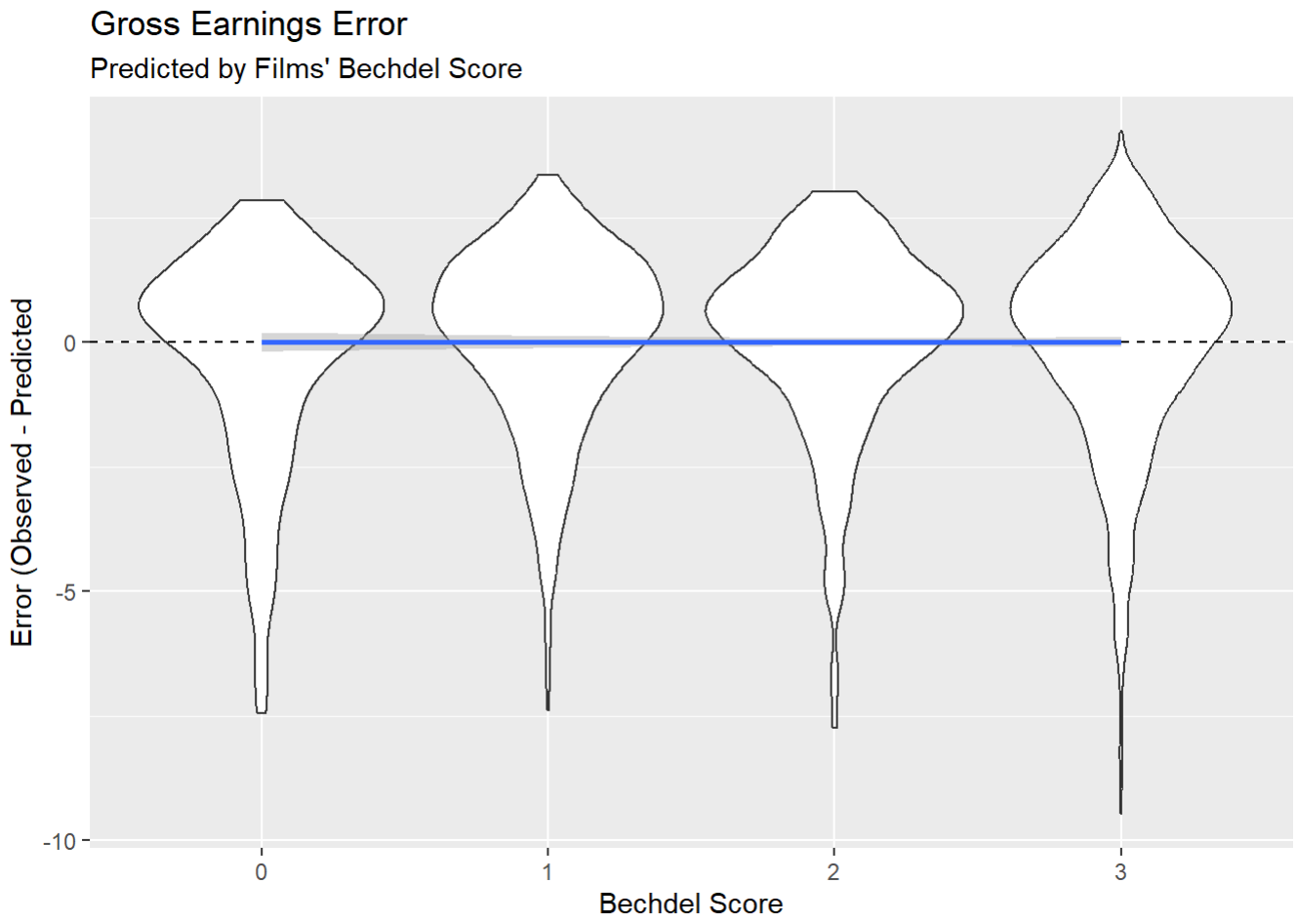
```
## # A tibble: 2,397 × 4
##    bechdel_score log_gross predictions   errors
##            <dbl>     <dbl>       <dbl>    <dbl>
##  1             3      18.1        17.8  0.354
##  2             3      17.8        17.8  0.0296
##  3             0      20.4        18.0  2.36
##  4             3      16.3        17.8 -1.50
##  5             1      17.9        17.9 -0.00276
##  6             2      20.3        17.9  2.46
##  7             3      19.9        17.8  2.12
##  8             2      20.1        17.9  2.20
##  9             3      19.0        17.8  1.23
## 10             1      19.9        17.9  2.00
## # … with 2,387 more rows
```

```
# univariate viz of error
movies_analysis%>%
  ggplot(aes(x=errors))+
  geom_density()+
  labs(title="Error Between Predicted Gross Earnings and Actual Gross Earnings",
       subtitle="Observed Gross - Predicted Gross",
       x="Error",
       y="Proportion of Movies")+
  geom_vline(data=movies_analysis, aes(xintercept=mean(errors)), linetype="dashed")
```



```
# multivariate viz
violin<-movies_analysis%>%
  ggplot(aes(x=factor(bechdel_score), y=errors))+
  geom_violin()+
  labs(title="Gross Earnings Error",
       subtitle="Predicted by Films' Bechdel Score",
       x="Bechdel Score",
       y="Error (Observed - Predicted")+
  geom_hline(yintercept=0, linetype="dashed")
violin+geom_smooth(method="lm", aes(group=1))
```

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



```
# CALCULATE THE FULL RMSE HERE
movies_analysis<-movies_analysis %>%
  mutate(se1=errors^2) %>% # Calculate the squared errors
  mutate(mse1=mean(se))%>%# Calculate the mean squared errors
  mutate(rmse1=sqrt(mse)) # Calculate the square root of the mean squared errors

#just taking a look
movies_analysis%>%
  select(se1, mse1, rmse1)
```

```
## # A tibble: 2,397 × 3
##           se1  mse1 rmse1
##         <dbl> <dbl> <dbl>
##  1 0.125       4.57  2.14
##  2 0.000875    4.57  2.14
##  3 5.55        4.57  2.14
##  4 2.26        4.57  2.14
##  5 0.00000760  4.57  2.14
##  6 6.07        4.57  2.14
##  7 4.51        4.57  2.14
##  8 4.82        4.57  2.14
##  9 1.52        4.57  2.14
## 10 3.99        4.57  2.14
## # … with 2,387 more rows
```

```
# CALCULATE THE CV RMSE HERE
set.seed(123)
cvRes2<-NULL
for(i in 1:100){
  #take the sample and store it
  samp<-sample(x=1:nrow(movies_analysis),
               size=round(nrow(movies_analysis)*0.8),
               replace=F)

  #training and testing data
  sample_train<-movies_analysis%>%slice(samp)
  sample_test<-movies_analysis%>%slice(-samp)

  #run model with training data
  bech_model<-lm(formula=log_gross~bechdel_score,
                 data=sample_train)

  #calculate rmse
  sample_test$predictions <- predict(bech_model, newdata=sample_test) # Generate predicted value
s from the model

  bech_e <- sample_test$log_gross-sample_test$predictions # Calculate the errors as the true Y m
inus the predicted Y
  bech_se <- bech_e^2 # Square the errors
  bech_mse <- mean(bech_se)# Take the mean of the squared errors
  bech_rmse <- sqrt(bech_mse)# Take the square root of the mean of the squared errors
  cvRes2 <- c(bech_rmse, cvRes2) # Append the rmse to the cvRes object

}
mean(cvRes2)
```

```
## [1] 1.962773
```

Our multivariate visualization shows us that most films, regardless of bechdel score, tend to have error roughly centered around 0; however, there is increasing variance and skew as the bechdel score grows. Films with bechdel scores of 0 and 1 are skewed towards overpredicting gross earnings, while scores of 2 and 3 are heavily skewed towards overpredicting gross earnings.