Problem Set 5

Jadyn Gonzalez

10/3/2019

Question 1: Does increasing a movie's budget ever pay out?

- a) We are working with the movies data set again so we need to set our working directory and import the data.

```
setwd("~/Documents/MGSC310")
getwd()
```

```
## [1] "/Users/jgonzalez/Documents/MGSC310"
```

```
movies = read.csv("movie_metadata.csv")
```

- b) Using the given code we'll clean up the data and simplify some variables. First we'll remove NaN values.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
movies = movies[!is.na(movies$budget),]
movies = movies[!is.na(movies$gross),]
```

Then we'll remove "" and "Not Rated" values from 'content_rating'.

```
movies = movies[(movies$content_rating != "" & movies$content_rating != "Not Rated"), ]
```

We need to remove some outliers in 'budget'.

```
movies = movies[movies$budget<4e+8,]
```

We'll create new simplified versions of some variables.

```r
movies$grossM = movies$gross/1e+6
movies$budgetM = movies$budget/1e+6
movies$profitM = movies$grossM-movies$budgetM
```

Use fct_lump() to create lump rating into four factor levels.

```r
movies$rating_simple = fct_lump(movies$content_rating, n = 4)
```

And create our train and test sets.

```r
set.seed(2019)
train_indx = sample(1:nrow(movies), 0.8 * nrow(movies), replace=FALSE)
train = movies[train_indx, ]
test = movies[-train_indx, ]
```

- c) Now we'll regress 'grossM' against 'budgetM' and 'imdb_score'.

```r
mod1 = lm(grossM ~ budgetM + imdb_score, data = train)
summary(mod1)
```

```
##
## Call:
## lm(formula = grossM ~ budgetM + imdb_score, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -389.14  -26.24  -10.30   14.87  490.14
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -74.51571    6.03477  -12.35   <2e-16 ***
## budgetM       1.00195    0.02187   45.82   <2e-16 ***
## imdb_score   13.59706    0.91594   14.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.53 on 3026 degrees of freedom
## Multiple R-squared:  0.4373, Adjusted R-squared:  0.437
## F-statistic:  1176 on 2 and 3026 DF,  p-value: < 2.2e-16
```

- d) We see that the coefficient for budgetM is 1.002 which is about 1. This shows a positive relationship between budget and profit. It's estimated that for every million dollars we add to our budget, profit is expected to increase by just over a million dollars.

- e) We'll run the same model but add the square of 'budgetM' as a variable.

```r
mod2 = lm(grossM ~ budgetM + I(budgetM^2) + imdb_score, data = train)
summary(mod2)
```

```
##
## Call:
```

```
## lm(formula = grossM ~ budgetM + I(budgetM^2) + imdb_score, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -321.12  -26.00   -9.78   15.26  503.61
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -7.902e+01  6.220e+00  -12.70  < 2e-16 ***
## budgetM       1.141e+00  5.231e-02   21.82  < 2e-16 ***
## I(budgetM^2) -7.864e-04  2.684e-04   -2.93  0.00341 **
## imdb_score    1.388e+01  9.197e-01   15.09  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 52.47 on 3025 degrees of freedom
## Multiple R-squared:  0.4389, Adjusted R-squared:  0.4384
## F-statistic: 788.8 on 3 and 3025 DF,  p-value: < 2.2e-16
```

- f) Now we'll look at the marginal impact of 'budgetM'.

```r
library(margins)
margins(mod2, at=list(budgetM = c(25,50,75,90,100,200,300)))
```

```
## Average marginal effects at specified values
```

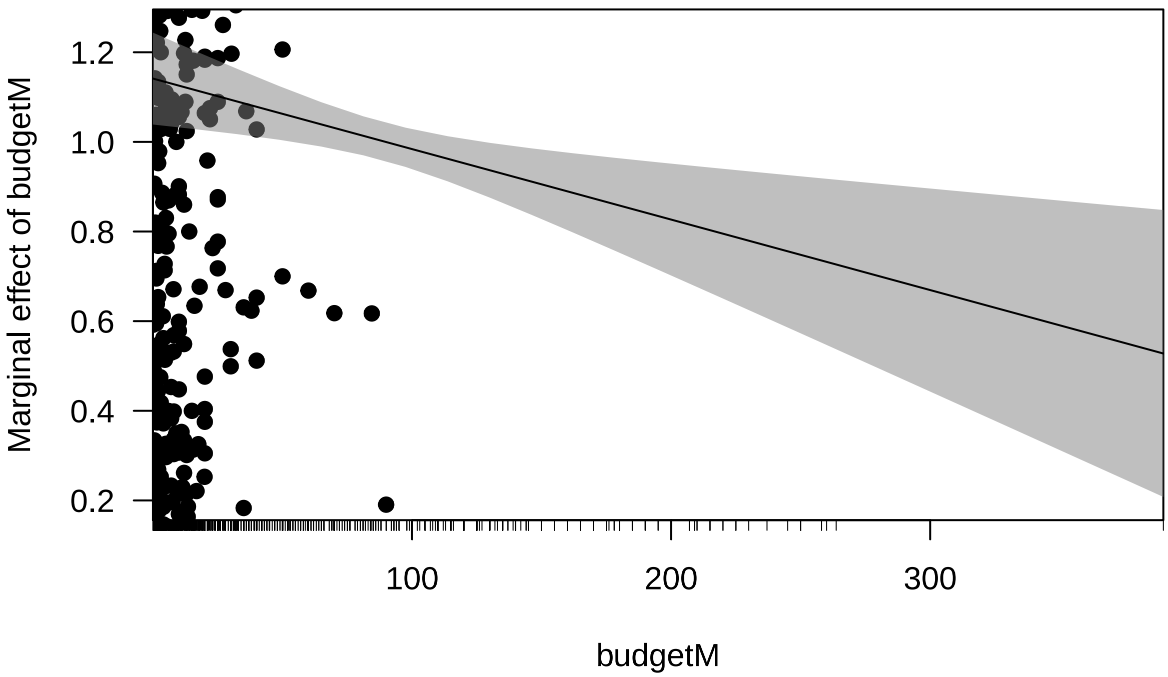
```
## lm(formula = grossM ~ budgetM + I(budgetM^2) + imdb_score, data = train)
```

```
##  at(budgetM) budgetM imdb_score
##           25  1.1019      13.88
##           50  1.0626      13.88
##           75  1.0233      13.88
##           90  0.9997      13.88
##          100  0.9839      13.88
##          200  0.8267      13.88
##          300  0.6694      13.88
```

From the output, it looks like it would make sense to increase budget at levels of 25, 50, and 75 million since adding 1 million to budget would expect to yield over 1 million in gross profit. Levels that are greater than 90 million expect to return less than a million for every million in budget.

- g) Plot of the marginal impact.

```r
cplot(mod2, x = 'budgetM', what = 'effect', scatter = TRUE)
```



It appears that budget only has a significant effect on gross profit for movies with less than 100 million in budget. Movies with budgets greater than 100 million do not observe a significant impact of budget on gross.

Question 2: Movie residuals and predicted values

- a) Let's regress gross profit on several variables and relevel to 'R' rated movies.

```
mod3 = lm(grossM ~ imdb_score + budgetM + I(budgetM^2) + relevel(rating_simple, ref = "R"), data = trai
summary(mod3)
```

```
##
## Call:
## lm(formula = grossM ~ imdb_score + budgetM + I(budgetM^2) + relevel(rating_simple,
##     ref = "R"), data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -316.51  -25.36   -7.99   15.35  503.19
##
## Coefficients:
##                                        Estimate Std. Error t value
## (Intercept)                          -9.386e+01  6.374e+00 -14.725
## imdb_score                            1.519e+01  9.230e-01  16.459
## budgetM                               1.020e+00  5.367e-02  19.011
```

```
## I(budgetM^2)                             -4.646e-04  2.679e-04  -1.735
## relevel(rating_simple, ref = "R")G        3.325e+01  6.540e+00   5.084
## relevel(rating_simple, ref = "R")PG       2.331e+01  2.867e+00   8.130
## relevel(rating_simple, ref = "R")PG-13  1.542e+01  2.247e+00   6.864
## relevel(rating_simple, ref = "R")Other  6.901e+00  7.644e+00   0.903
##                                          Pr(>|t|)
## (Intercept)                               < 2e-16 ***
## imdb_score                                < 2e-16 ***
## budgetM                                   < 2e-16 ***
## I(budgetM^2)                               0.0829 .
## relevel(rating_simple, ref = "R")G       3.92e-07 ***
## relevel(rating_simple, ref = "R")PG      6.17e-16 ***
## relevel(rating_simple, ref = "R")PG-13 8.12e-12 ***
## relevel(rating_simple, ref = "R")Other   0.3667
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 51.69 on 3021 degrees of freedom
## Multiple R-squared:  0.4561, Adjusted R-squared:  0.4548
## F-statistic: 361.8 on 7 and 3021 DF,  p-value: < 2.2e-16
```

- b) From the output, we see that the coefficient for a 'G' rated movie is 3.325. This means that compared to an R rated movie, a movie with a G rating is expected to have a gross profit of 3.325 million dollars higher.
- c) Now we'll run predictions on our train and test sets.

```
predicted_train = predict(mod3)
predicted_test = predict(mod3, newdata = test)
head(predicted_train, 5)
```

```
##      1381      1780      4295      4677      3926
##  81.63893  43.96550 -20.50903  11.73468  25.67834
```

```
head(predicted_test, 5)
```

```
##       24       27       32       33       44
## 247.6437 224.0332 217.9563 216.4370 250.9750
```

- d) We'll also calculate the residuals for both.

```
residuals_train = train$grossM - predicted_train
residuals_test = test$grossM - predicted_test
head(residuals_train, 5)
```

```
##       1381       1780       4295       4677       3926
## -13.430737 -13.277133  21.044283  -6.215762 -22.785754
```

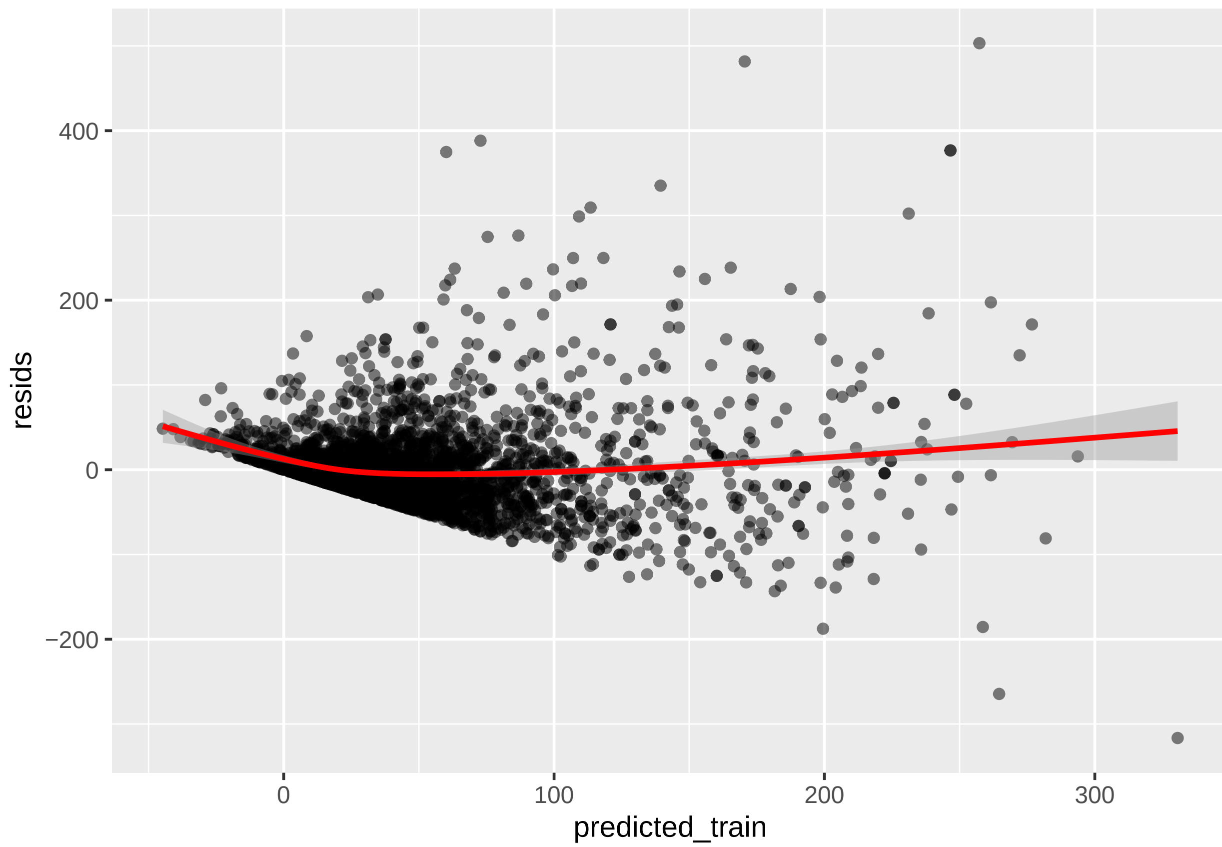
```
head(residuals_test, 5)
```

```
##        24         27         32         33         44
##  10.71165 434.63906 155.42161 192.55523 164.00952
```

- e) Plotting the values. We need to create dataframes to use in ggplot.

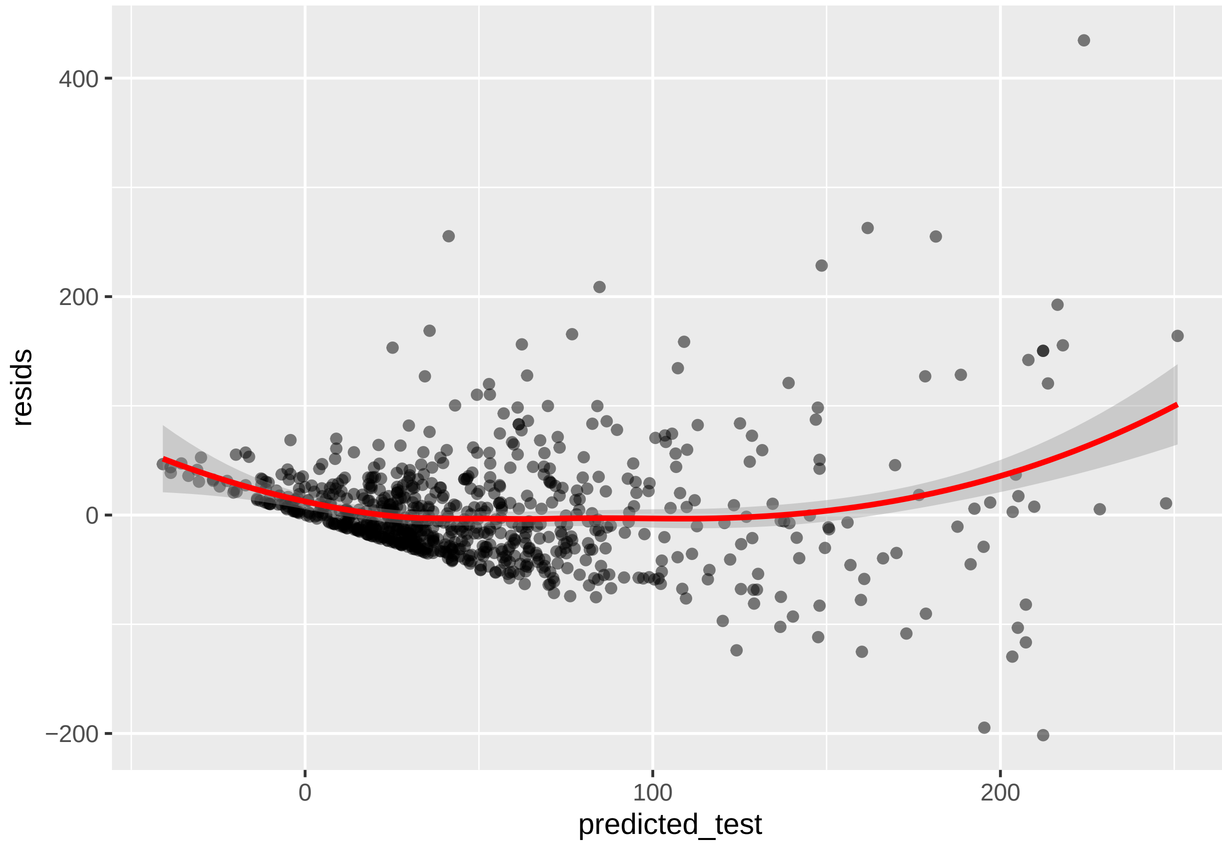
```
mod3_train_df = data.frame(
  resids = residuals_train,
  predicted = predicted_train
)
ggplot(mod3_train_df,aes(x=predicted_train,y=resids)) + geom_point(alpha=0.5) + geom_smooth(color="red"
```

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




```
mod3_test_df = data.frame(
  resids = residuals_test,
  predicted = predicted_test
)
ggplot(mod3_test_df,aes(x=predicted_test,y=resids)) + geom_point(alpha=0.5) + geom_smooth(color="red")
```

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

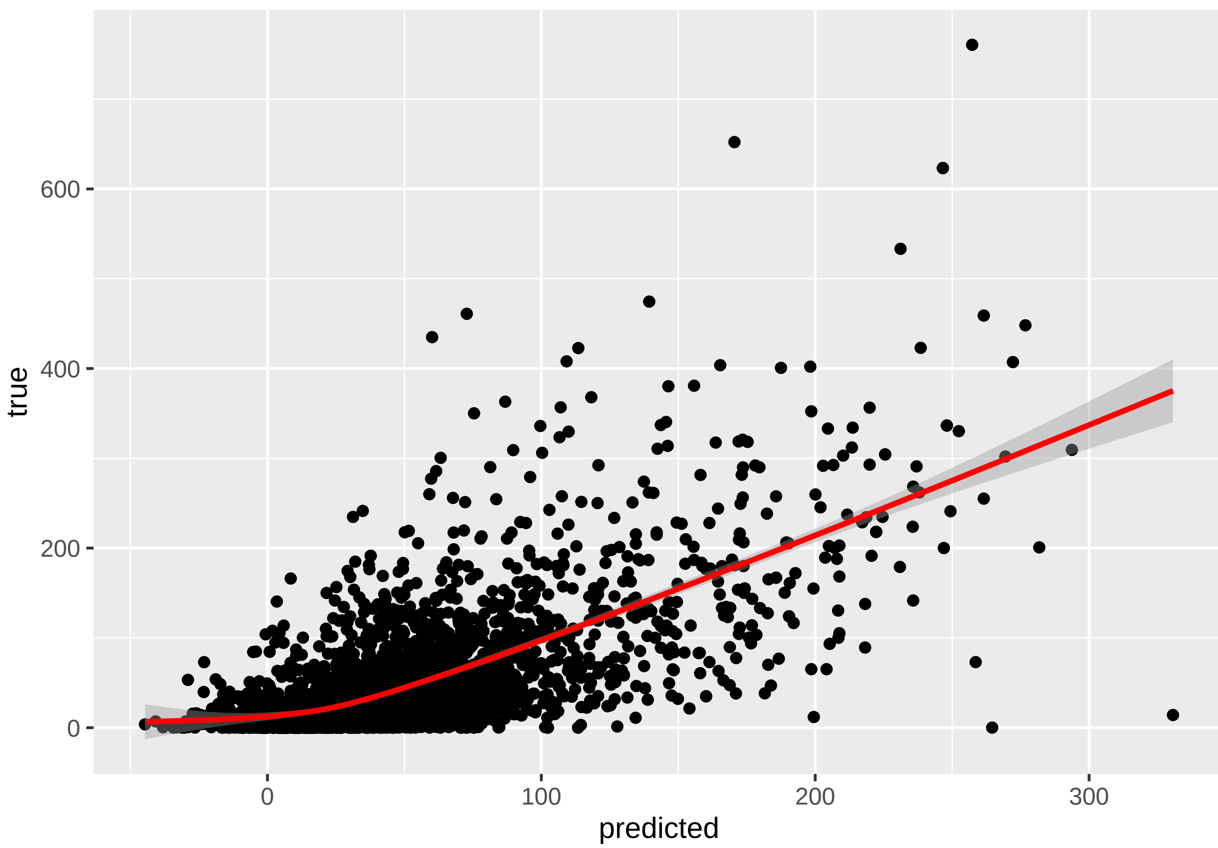


Both graphs have fan shaped error values which means we have heteroscedasticity.

- f) Plot of predicted values against true values

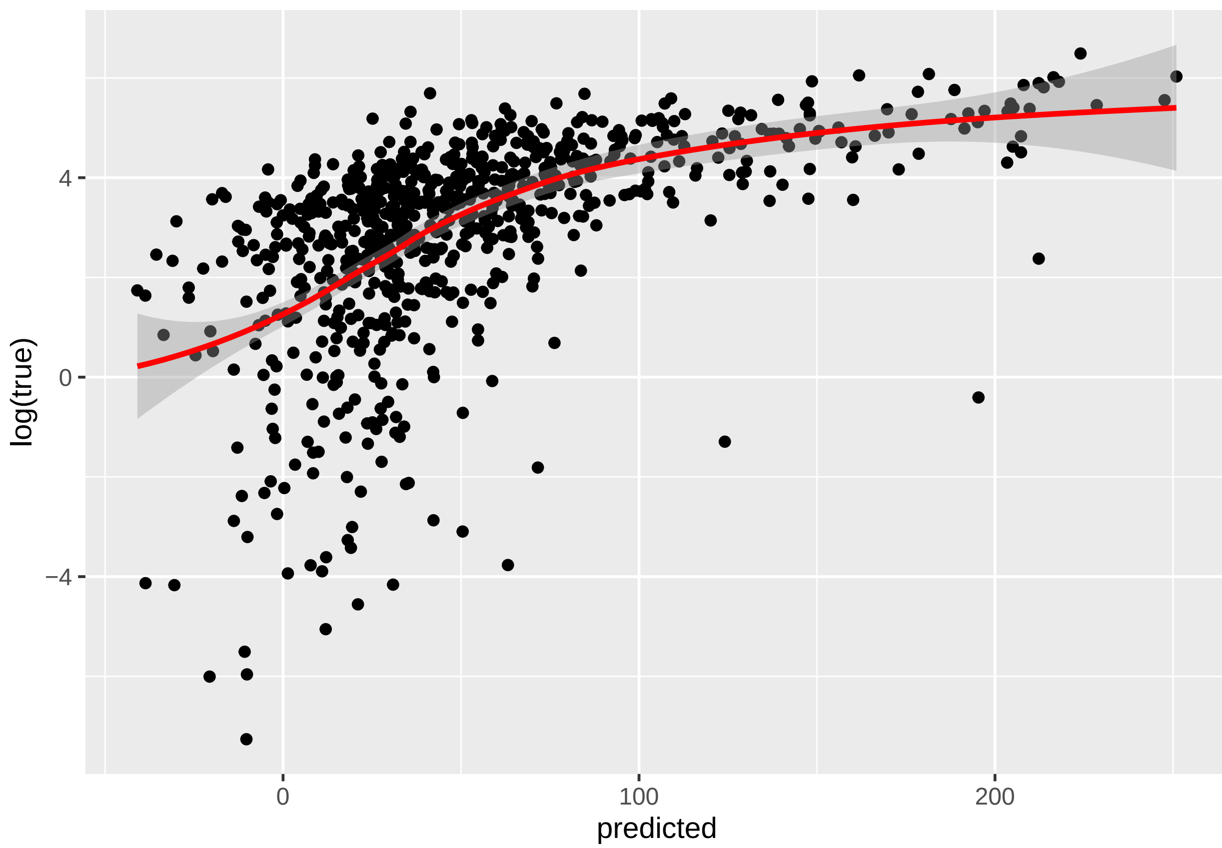
```
df1 = data.frame(
  predicted = predicted_train,
  true = train$grossM
)
df2 = data.frame(
  predicted = predicted_test,
  true = test$grossM
)

ggplot(df1, aes(x = predicted, y = true)) + geom_point() + geom_smooth(color = "red")
```

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




```r
ggplot(df2, aes(x = predicted, y = log(true))) + geom_point() + geom_smooth(color = "red")
```

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



- g) Calculating accuracy and RMSE.

```
library(forecast)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```

```
accuracy(predicted_train, train$grossM)
```

```
##                     ME     RMSE      MAE       MPE     MAPE
## Test set -5.687971e-13 51.62507 32.99394 -3680.763 7029.888
```

```r
accuracy(predicted_test, test$grossM)
```

```
##                ME     RMSE      MAE     MPE     MAPE
## Test set 2.352148 50.54496 33.27347 2243.78 7009.265
```

```r
RMSE = function(t,p) {
  sqrt(sum(((t - p)^2)) * (1/length(t)))
}

RMSE_train = RMSE(train$grossM, predicted_train)
RMSE_train
```

```
## [1] 51.62507
```

```r
RMSE_test = RMSE(test$grossM, predicted_test)
RMSE_test
```

```
## [1] 50.54496
```

While the model itself may not be the best fit for the data, which is indicated by the high RMSE, it does not appear that we are overfitting. Both our test and train sets have similar RMSE which is not an indication of overfitting. WE could try and normalize the data use log transformations etc. to try and get a better RMSE value.