Modeling_Katrina

Katrina Truebebach

March 18, 2019

```
rm(list = ls())
```

Load cleaned data

```
load(file = '~/DS5110/data/proj_cleaned_dta.RData')
```

Fit Model with Genre Variables vs Real Revenue

Step Wise Selection

End model includes (in order of steps): 'Adventure', 'Action', 'Family', 'Mystery', 'Documentary', 'Drama', 'History', 'Romance'

Dependent variable is log(real_gross). Makes model look better *and* a lot of the relationships with other variables are more linear with log, so we will need to use this as y variable in the main model.

This model selection by and large makes sense. All included variables are significant at some level.
However, according to Qiang's graphs in EDA, some of the included genres do not make a real difference to real_gross. Especially History. Also, some genres that look like they would make a signficant difference are not included. For example, Animation.

Thoughts:

- There are a few genres that define almost all of the movies (For example, almost 80% of the movies are either Adventure, Action, Romance, or Drama). Thus, the relationship between revenue and some genres can be explained by other generes. For example, 93 out of 99 Animation movies are also Family. So Animation's effect on revenue may already by captured by Family, which is included in the model.
- On the flip side, History is included even though it seems to have a negligable effect on revenue based on the EDA bar graphs. I don't have a great explanation for this other than it was close to the cutoff RMSE for being included. 53 out of 55 History movies are also Drama. So unclear why included.

Also, the residuals are debatably random vs included and excluded variables in model (not sure if these are not-random enough to matter – see graphs).
More concerning is the fact that the residuals themselves are not Normal. See QQ-Plot (close-ish. . .)

```
train %>% filter(Animation == 1, Family == 1) %>% count() # 93
train %>% filter(Animation == 1) %>% count() # 99

train %>% filter(History == 1) %>% count() # 55
train %>% filter(History == 1, Drama == 1) %>% count() # 53
```

Which genres should we be using?
Note: not using the step() function because can't fit and find RMSE on different datasets (train, valid)

```
# version of train set with just genre columns to loop through
train_genre <- train %>% select(Action, Adventure, Animation, Biography, Comedy, Crime, Documentary,
                Drama, Family, Fantasy, History, Horror, Music, Musical, Mystery,
                Romance, SciFi, Sport, Thriller, War, Western)

# calculate log(real_gross)
```

```r
train <- train %>% mutate(real_gross_log = log(real_gross))
valid <- valid %>% mutate(real_gross_log = log(real_gross))

# function to automate each step of stepwise variable selection
step_wise_step <- function(genre_lst = NULL, formula = NULL) {
  # if first step
  if (length(genre_lst) == 0) {
    # rmse with each variable against real_gross
    rmse_genre <- sapply(names(train_genre), function(var) {
      rmse(lm(as.formula(str_c('real_gross_log ~', var)), data = train), data = valid)
    })
  # if > first step: exclude variables from genre_lst from data and include in model formula
  } else {
    rmse_genre <- sapply(names(train_genre %>% select(-genre_lst)), function(var) {
      rmse(lm(as.formula(str_c('real_gross_log ~', formula, ' + ', var)),
              data = train), data = valid)
    })
  }
  # return the name and value of the genre that resulted in the lowest RMSE
  return(rmse_genre[which.min(rmse_genre)])
}

# loop through each step wise loop
step_wise_loop <- function() {
  # list to store min RMSE from each step in
  rmse_lst <- c()

  # first step: no genre_lst or formula (default values NULL)
  min_genre <- step_wise_step()
  print(names(min_genre))

  # add to list of genres, formula, and min RMSE list
  genre_lst <- names(min_genre)
  formula <- str_c(names(min_genre))
  rmse_lst <- c(rmse_lst, min(min_genre))

  # loop through until have considered every genre variable
  for (i in seq(1:(ncol(train_genre)-1))) {
    print(i)
    # step
    min_genre <- step_wise_step(genre_lst = genre_lst, formula = formula)
    print(min_genre)

    # add to lists
    genre_lst <- c(genre_lst, names(min_genre))
    formula <- str_c(formula, ' + ', names(min_genre))
    rmse_lst <- c(rmse_lst, min(min_genre))
  }
  return(rmse_lst)
}

# step wise implement
# return list of all min RMSE from each step -> graph
```

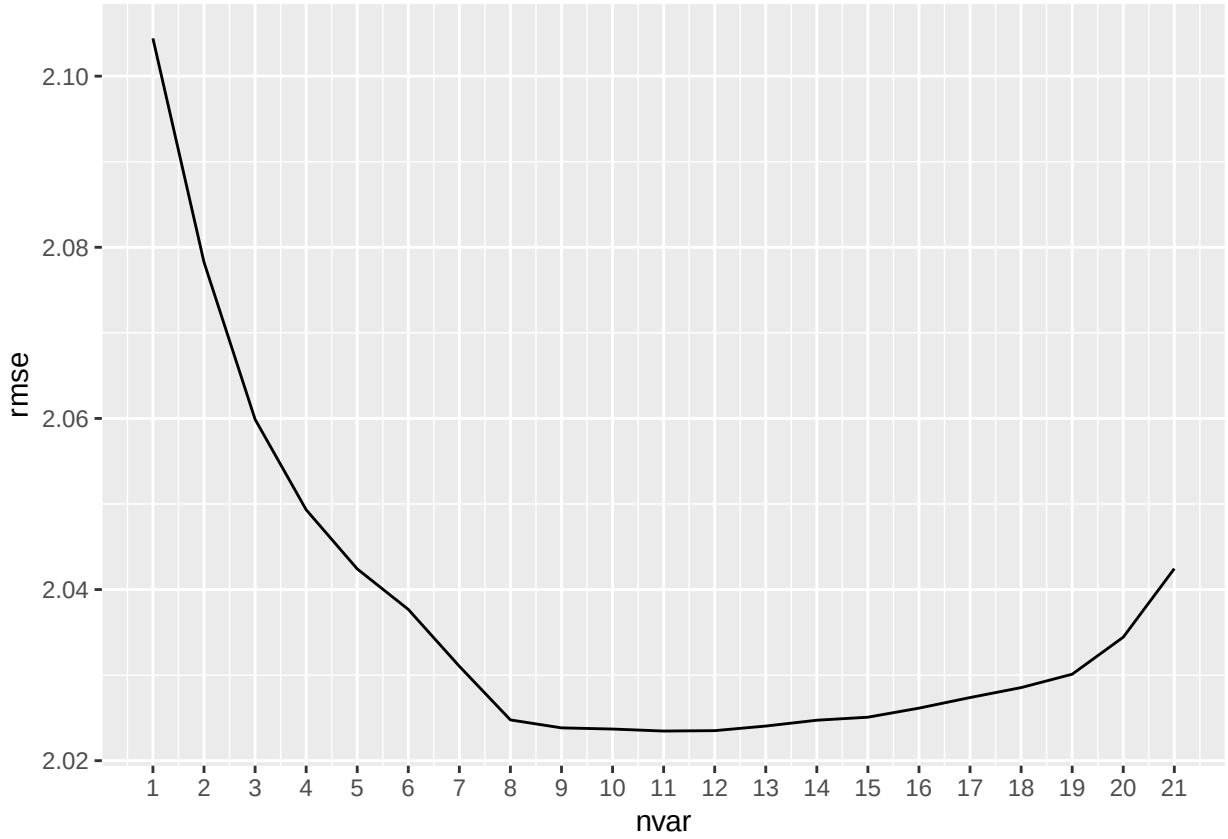
```
rmse_lst <- step_wise_loop()
```

```
## [1] "Adventure"
## [1] 1
##   Action
## 2.078261
## [1] 2
##   Family
## 2.059892
## [1] 3
##  Mystery
## 2.049328
## [1] 4
## Documentary
##    2.042411
## [1] 5
##   Drama
## 2.03768
## [1] 6
##  History
## 2.031034
## [1] 7
##  Romance
## 2.024762
## [1] 8
##      War
## 2.023833
## [1] 9
##    SciFi
## 2.023698
## [1] 10
##    Crime
## 2.023463
## [1] 11
##    Sport
## 2.023509
## [1] 12
##  Fantasy
## 2.024045
## [1] 13
##   Music
## 2.02473
## [1] 14
##  Musical
## 2.025082
## [1] 15
## Biography
##  2.026141
## [1] 16
##   Comedy
## 2.027379
## [1] 17
##  Horror
## 2.02854
```

```
## [1] 18
## Animation
##  2.030104
## [1] 19
## Thriller
## 2.034433
## [1] 20
##  Western
## 2.042443
```

Graph RMSE vs number of variables: how many to include?
Specify 'final' model

```
# graph RMSE at each step
fit_rmse <- tibble(nvar = 1:length(rmse_lst),
                   rmse = rmse_lst)
ggplot(fit_rmse) + geom_line(aes(x = nvar, y = rmse))+
  scale_x_continuous(breaks = seq(1, length(rmse_lst), by = 1))
```




```
# after var 8, decreases too small or increase

# model based off of step wise
# HOWEVER some of these variables are insignificant
  # (see pvalues and graphs from Qiang's EDA where barely any difference in revenue from genre)
mod <- lm(real_gross_log ~ Adventure + Action + Family + Mystery +
            Documentary + Drama + History + Romance,
          data = train)
```

```
summary(mod)
```

```
##
## Call:
## lm(formula = real_gross_log ~ Adventure + Action + Family + Mystery +
##     Documentary + Drama + History + Romance, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.4316 -0.7523  0.3915  1.3052  4.0615
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.62606    0.09336 178.093  < 2e-16 ***
## Adventure    0.57443    0.13767   4.173 3.15e-05 ***
## Action       0.85492    0.12393   6.899 7.15e-12 ***
## Family       1.08300    0.15541   6.969 4.42e-12 ***
## Mystery      0.42172    0.16239   2.597  0.00948 **
## Documentary -2.50278    0.30128  -8.307  < 2e-16 ***
## Drama       -0.53082    0.10172  -5.218 2.01e-07 ***
## History      1.11219    0.27919   3.984 7.05e-05 ***
## Romance      0.23773    0.11304   2.103  0.03559 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.002 on 1875 degrees of freedom
## Multiple R-squared:  0.159,  Adjusted R-squared:  0.1554
## F-statistic:  44.3 on 8 and 1875 DF,  p-value: < 2.2e-16
```

```
rmse(mod, data = valid)
```

```
## [1] 2.024762
```

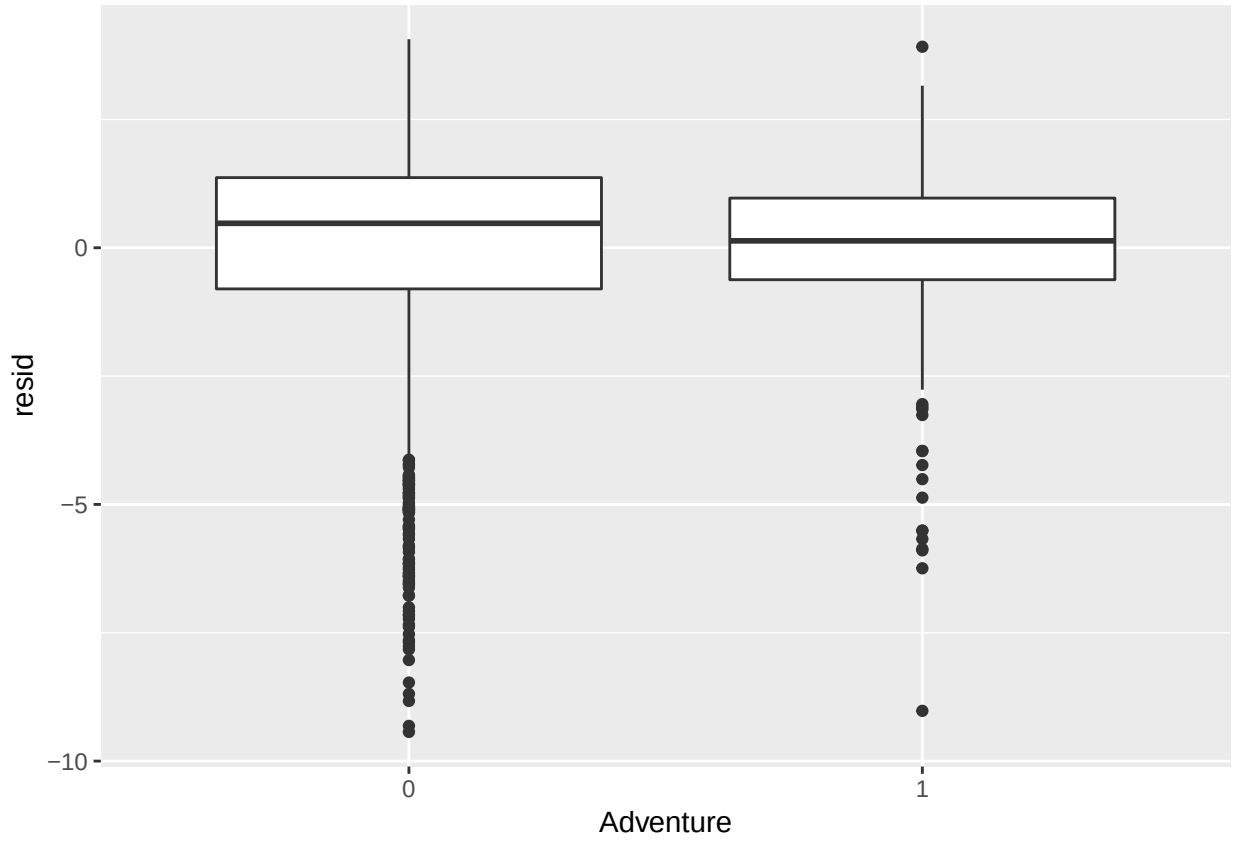
```
# list of these variables for future use
genre_xvar <- c('Adventure', 'Action', 'Family', 'Mystery',
                'Documentary', 'Drama', 'History', 'Romance')
```

Graph variables in and out of model against residuals. Most are fairly evenly distributed around residual. Worst is probably Western.

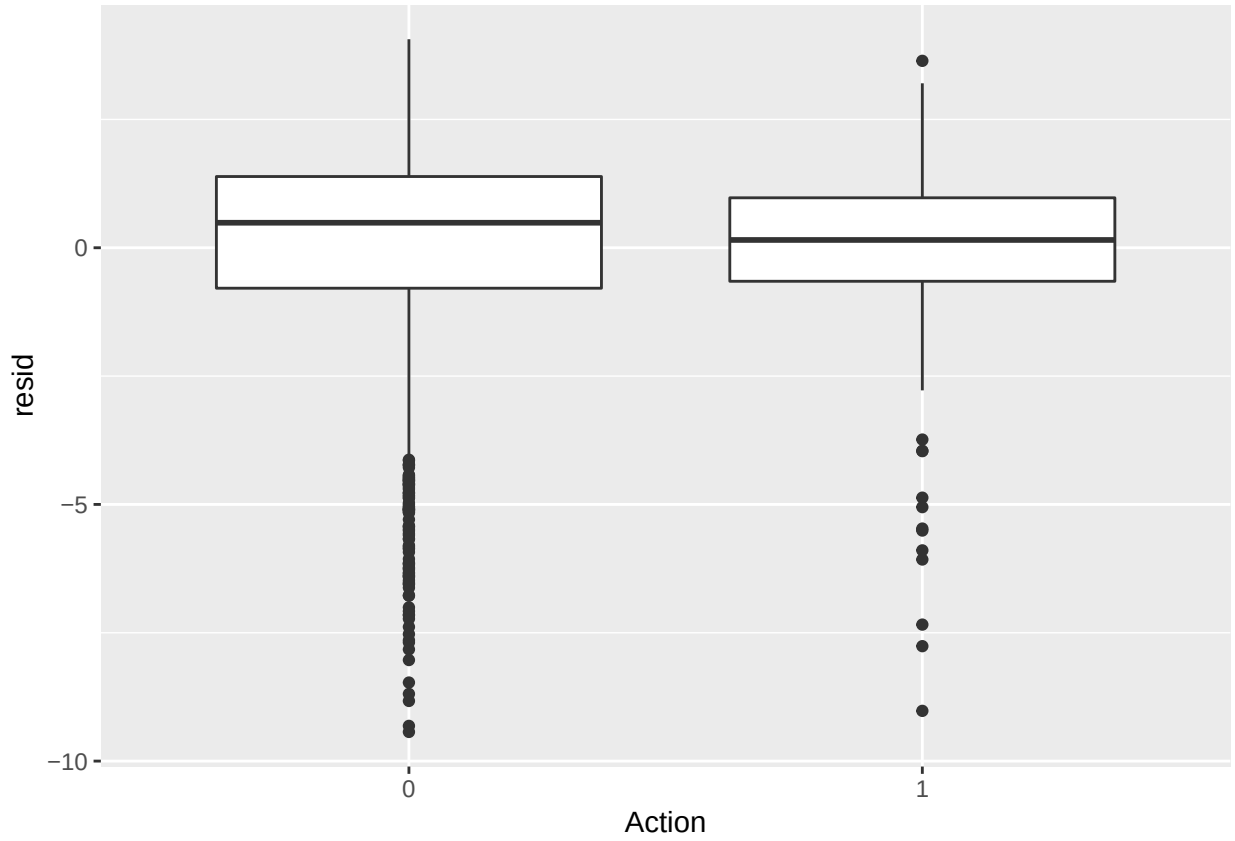
```
# graph residuals against each variable included in the model
# most look random except Adventure
train <- train %>%
  add_residuals(mod) %>%
  mutate_at(vars(Action, Adventure, Animation, Biography, Comedy, Crime, Documentary,
                 Drama, Family, Fantasy, History, Horror, Music, Musical, Mystery,
                 Romance, SciFi, Sport, Thriller, War, Western),
            funs(as.factor(.)))

lapply(genre_xvar, function(var) {
  train %>%
    ggplot() +
    geom_boxplot(aes_string(var, y = 'resid'))
})
```

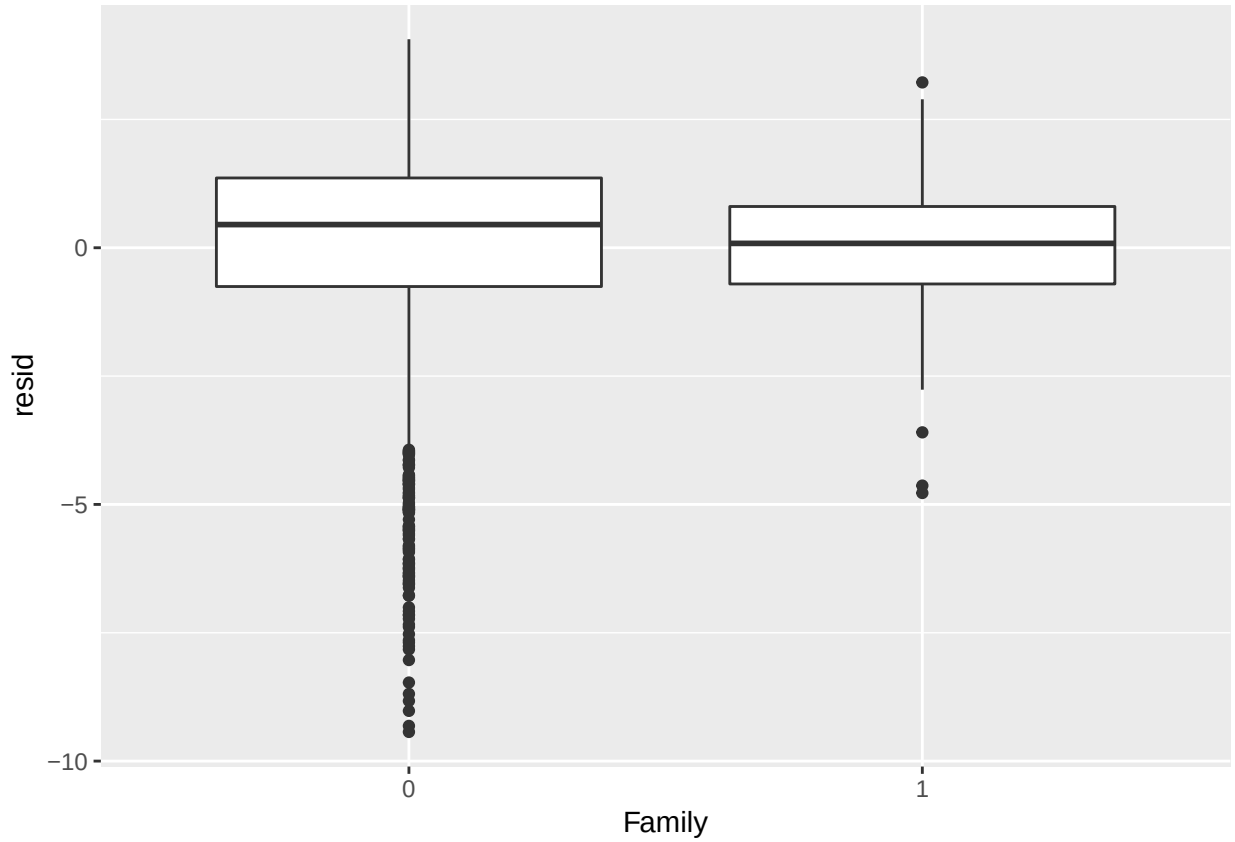
```
## [[1]]
```



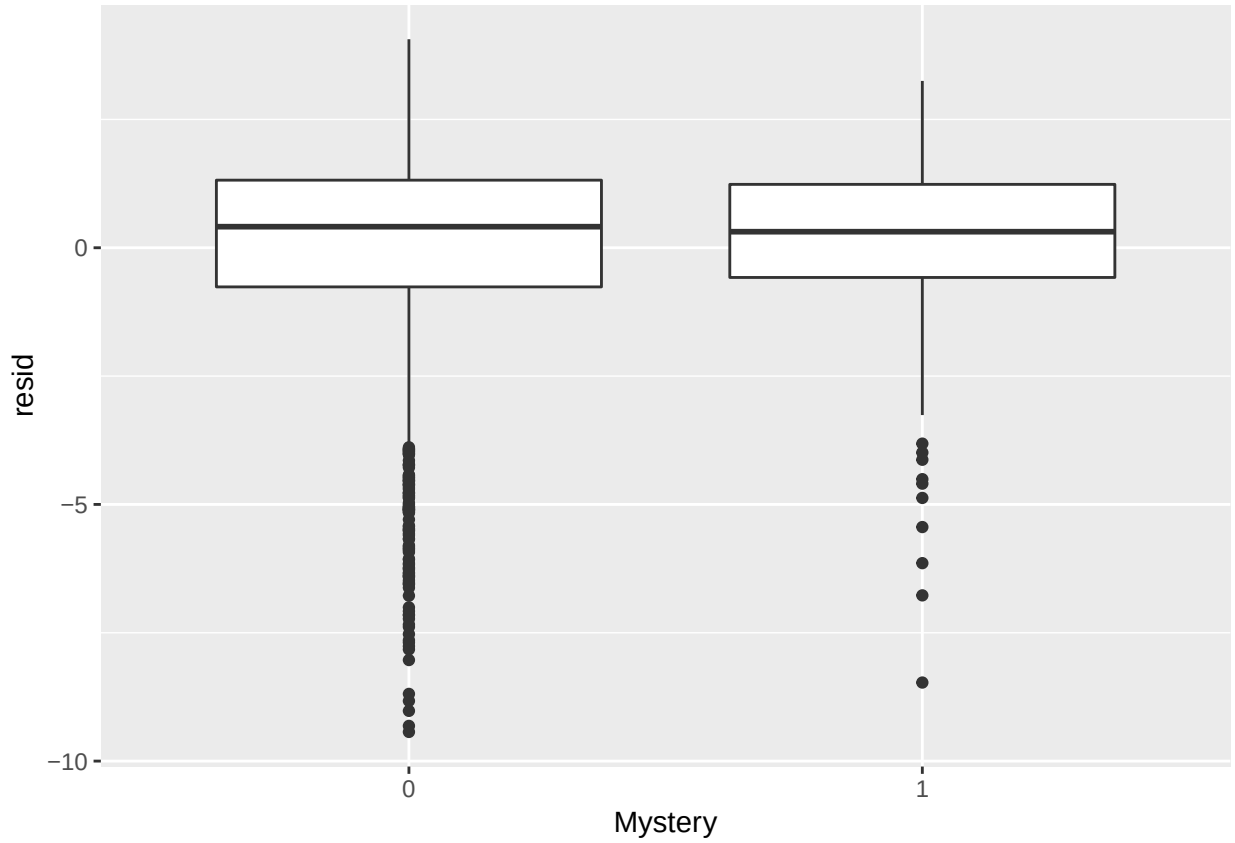

```
##
## [[2]]
```



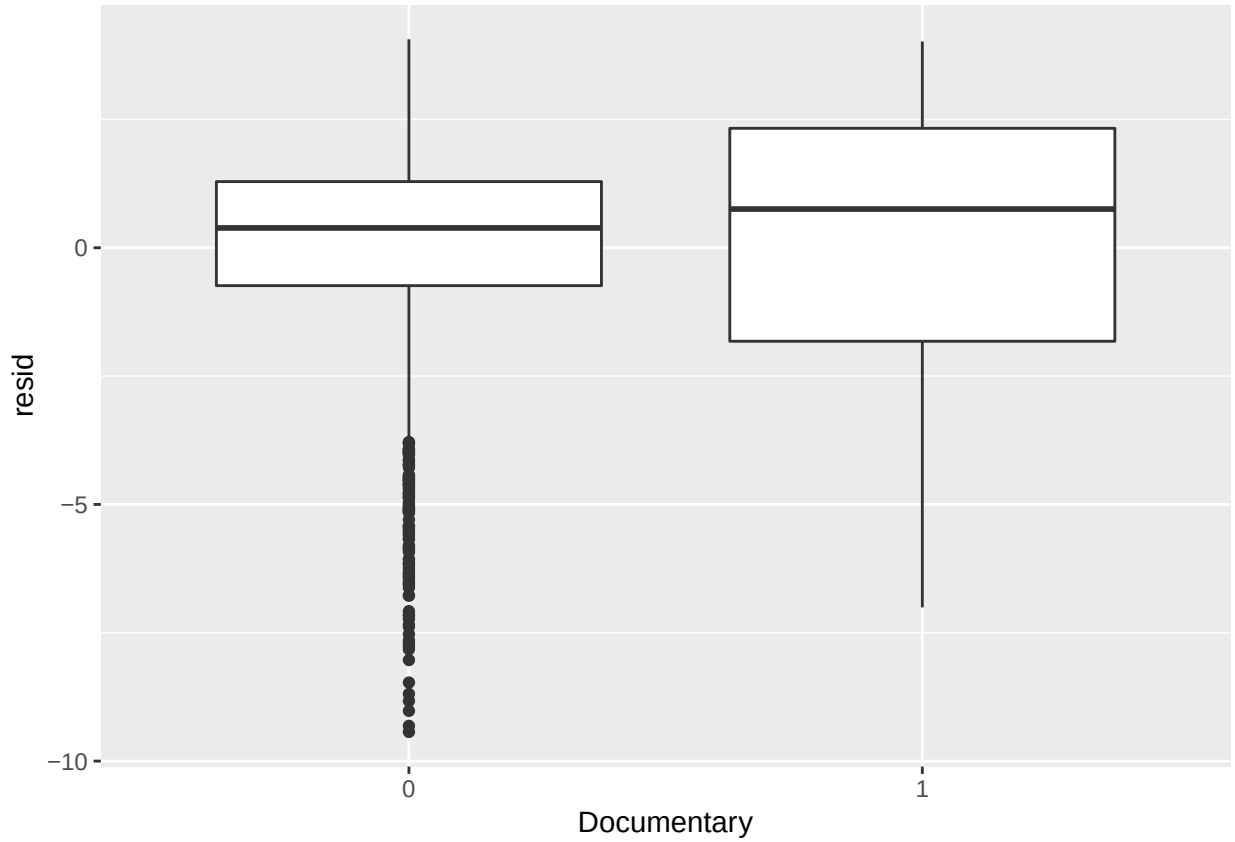

```
##
## [[3]]
```



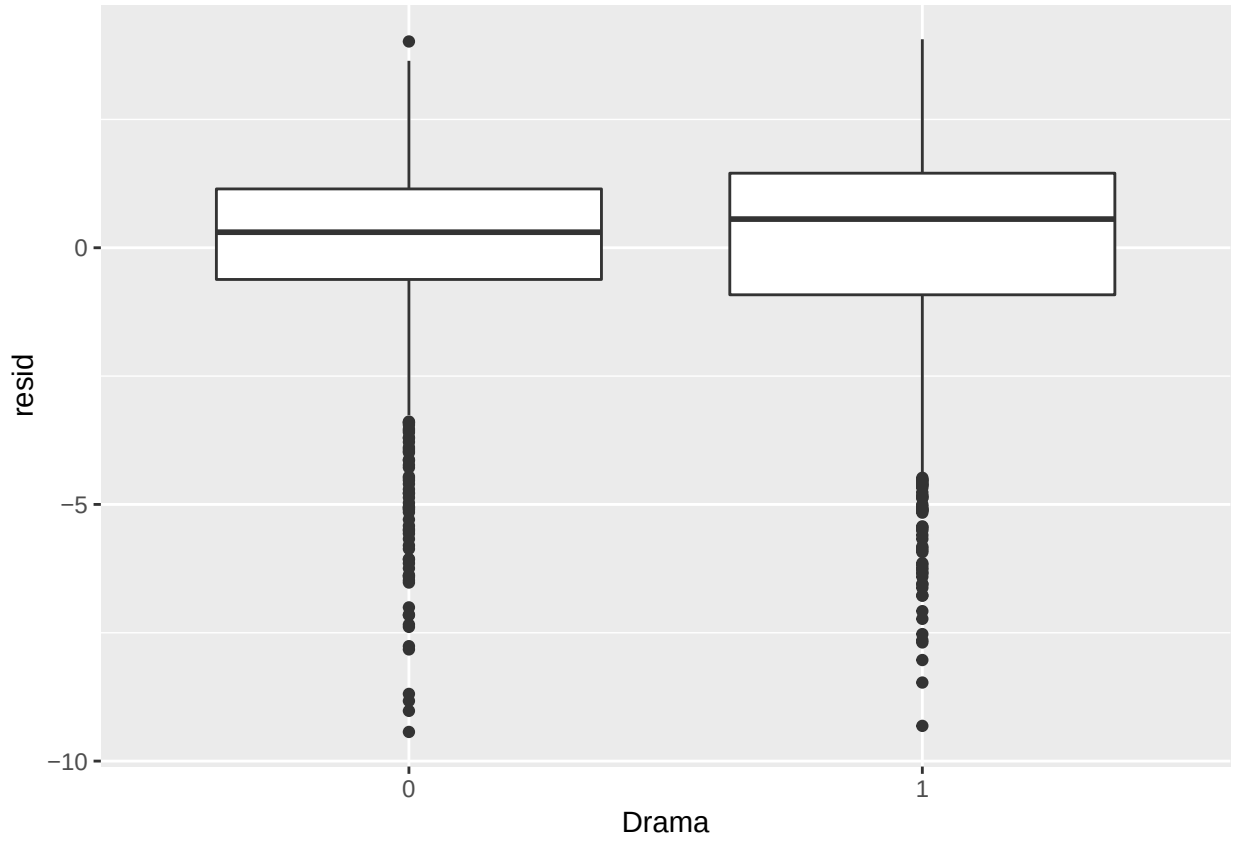

```
##
## [[4]]
```



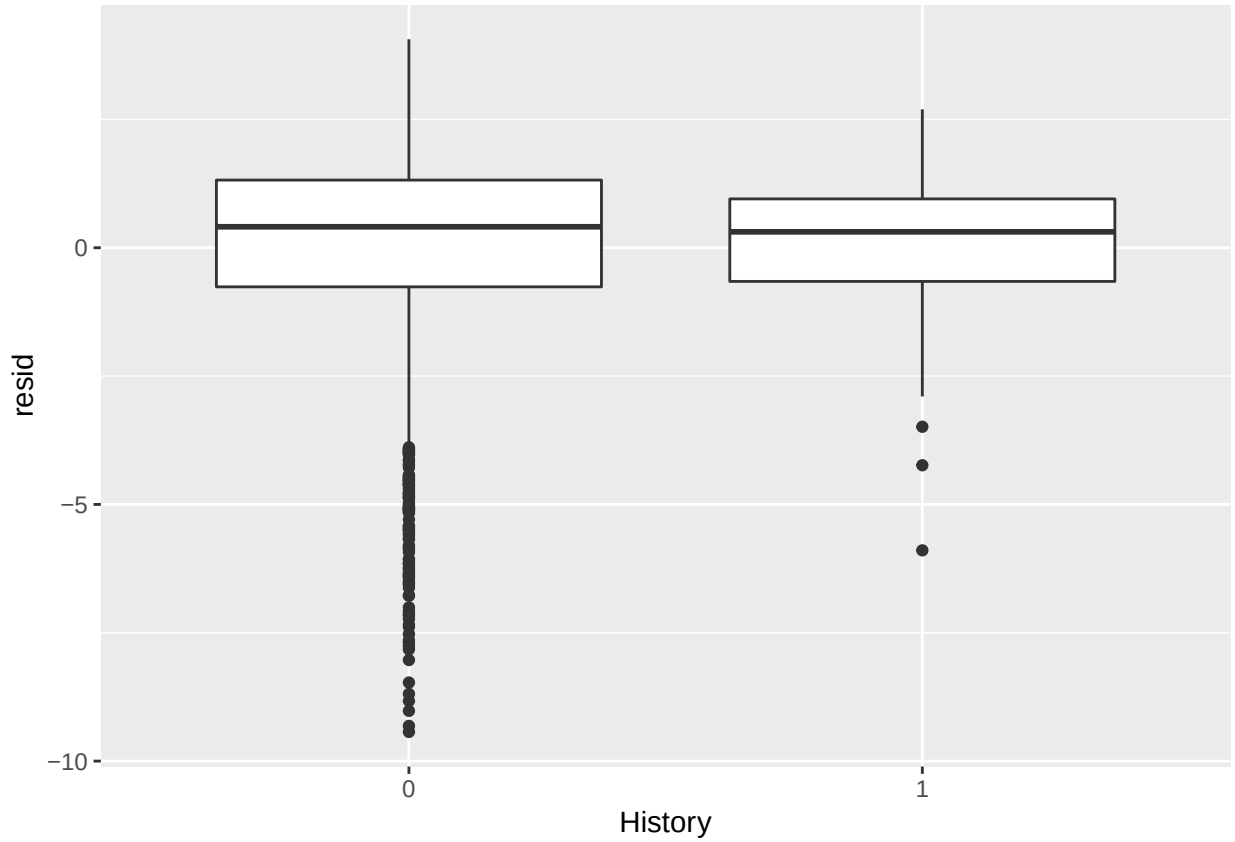

```
##
## [[5]]
```



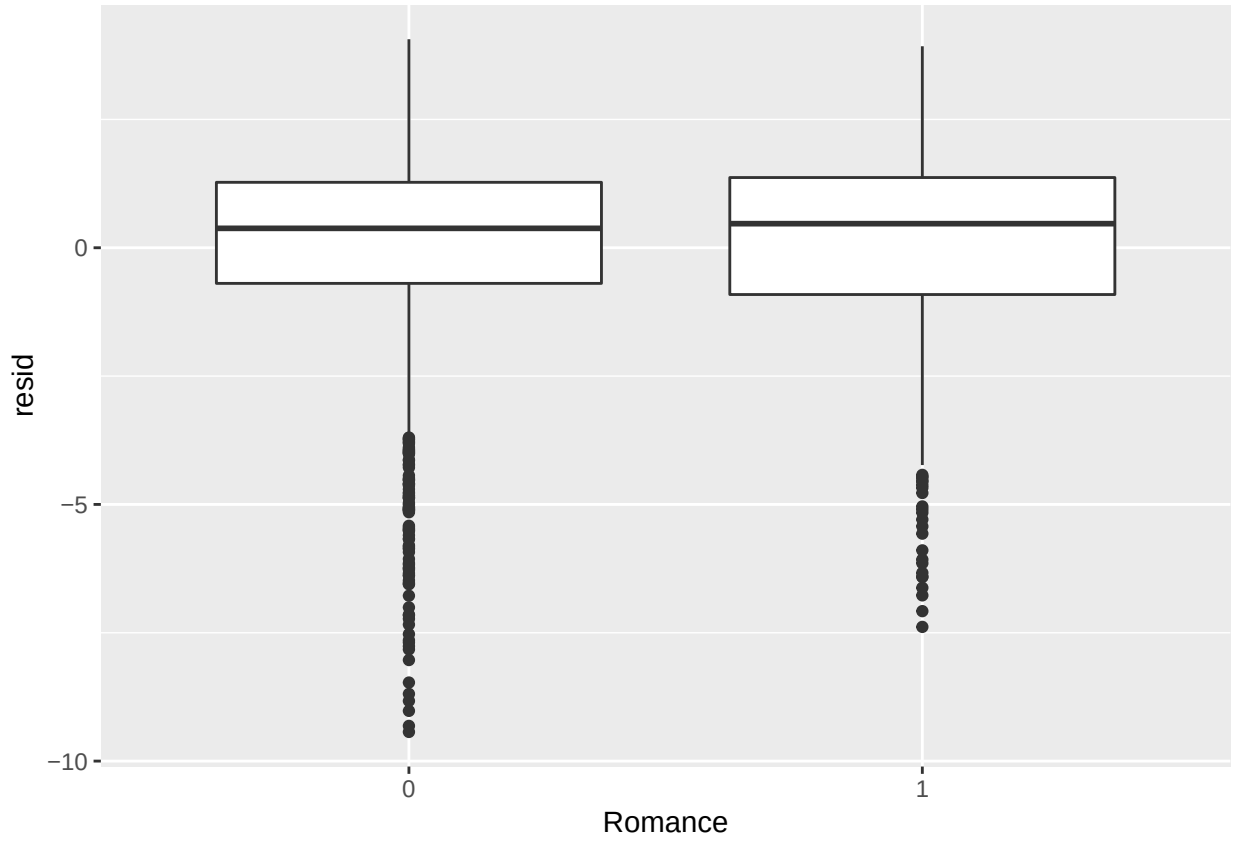

```
##
## [[6]]
```




```
##
## [[7]]
```

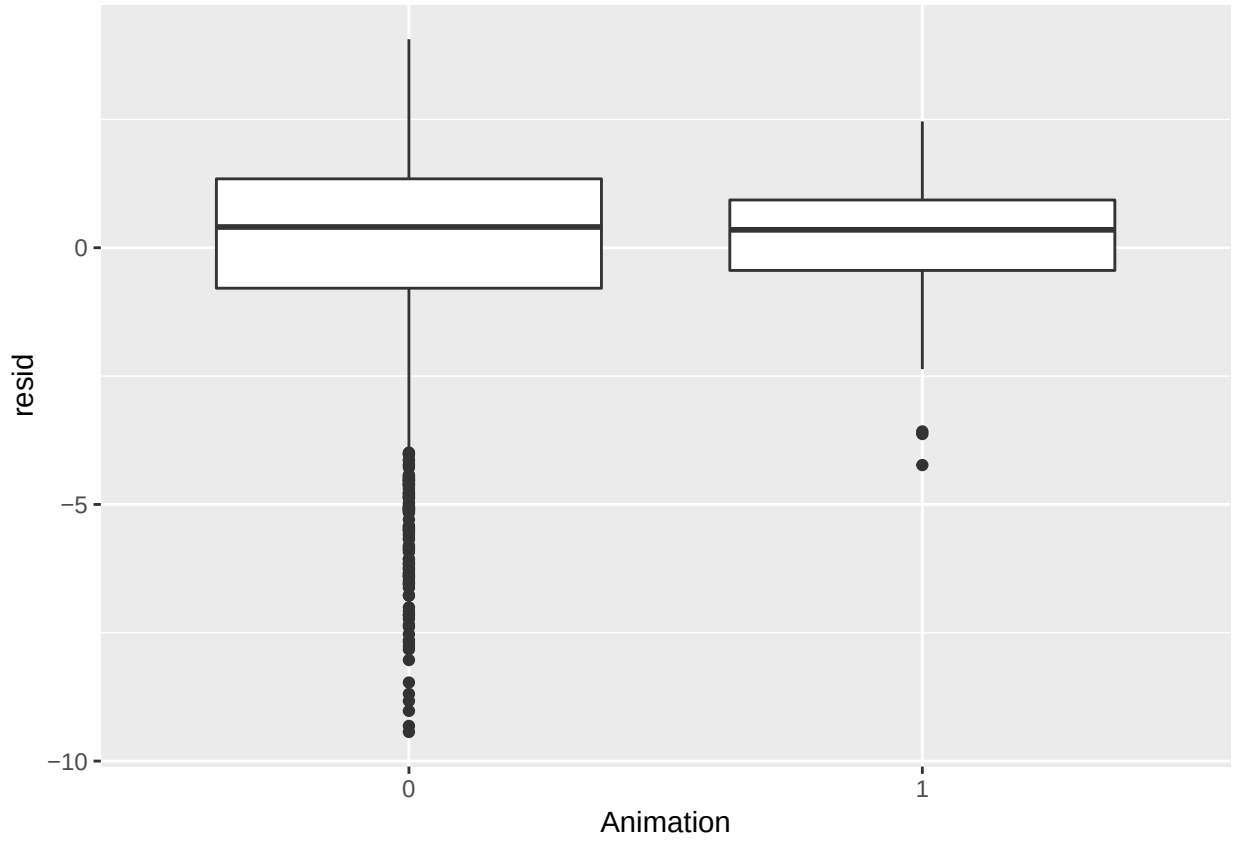



```
##
## [[8]]
```

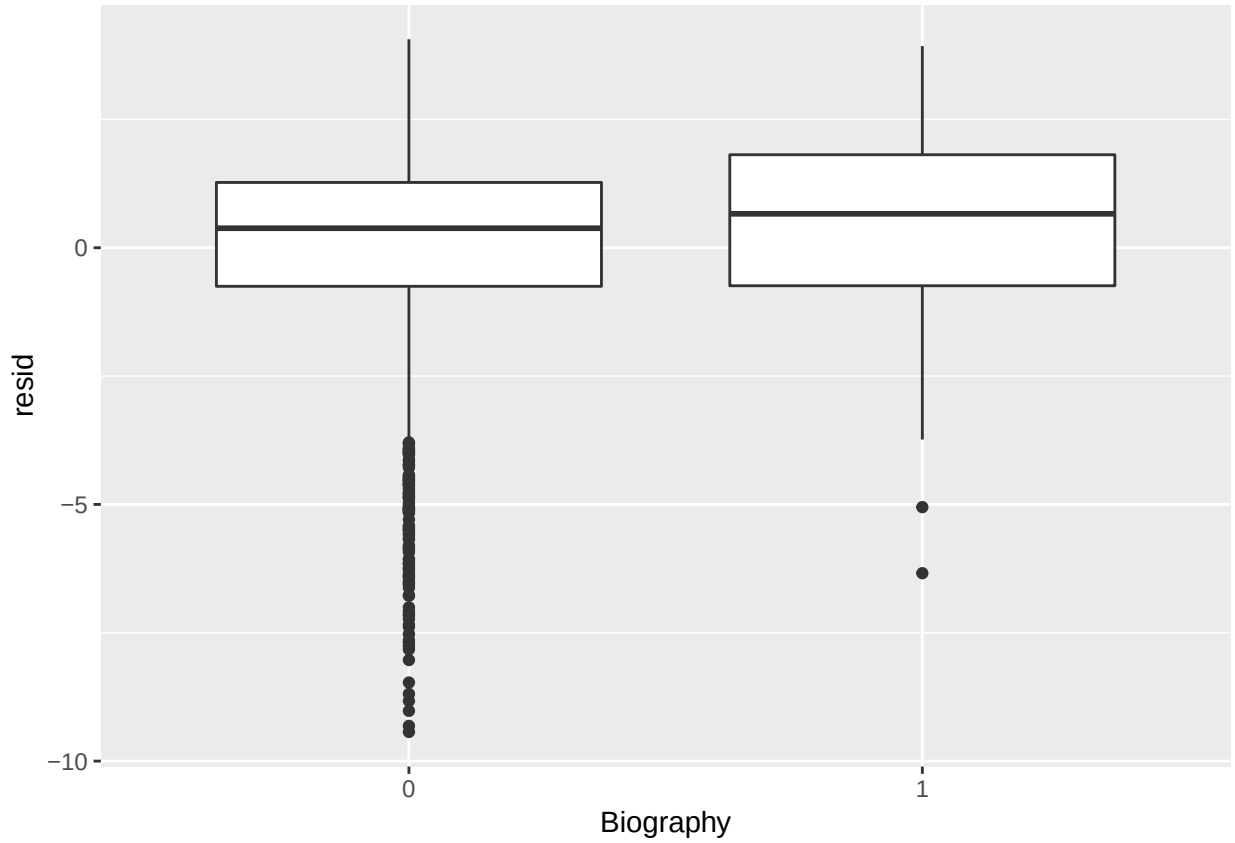



```
# graph residuals against each genre not included in the model
# several are questionable if random. Especially Animation.
lapply(names(train_genre %>% select(-genre_xvar)), function(var) {
  train %>%
    ggplot() +
    geom_boxplot(aes_string(var, y = 'resid'))
})
```

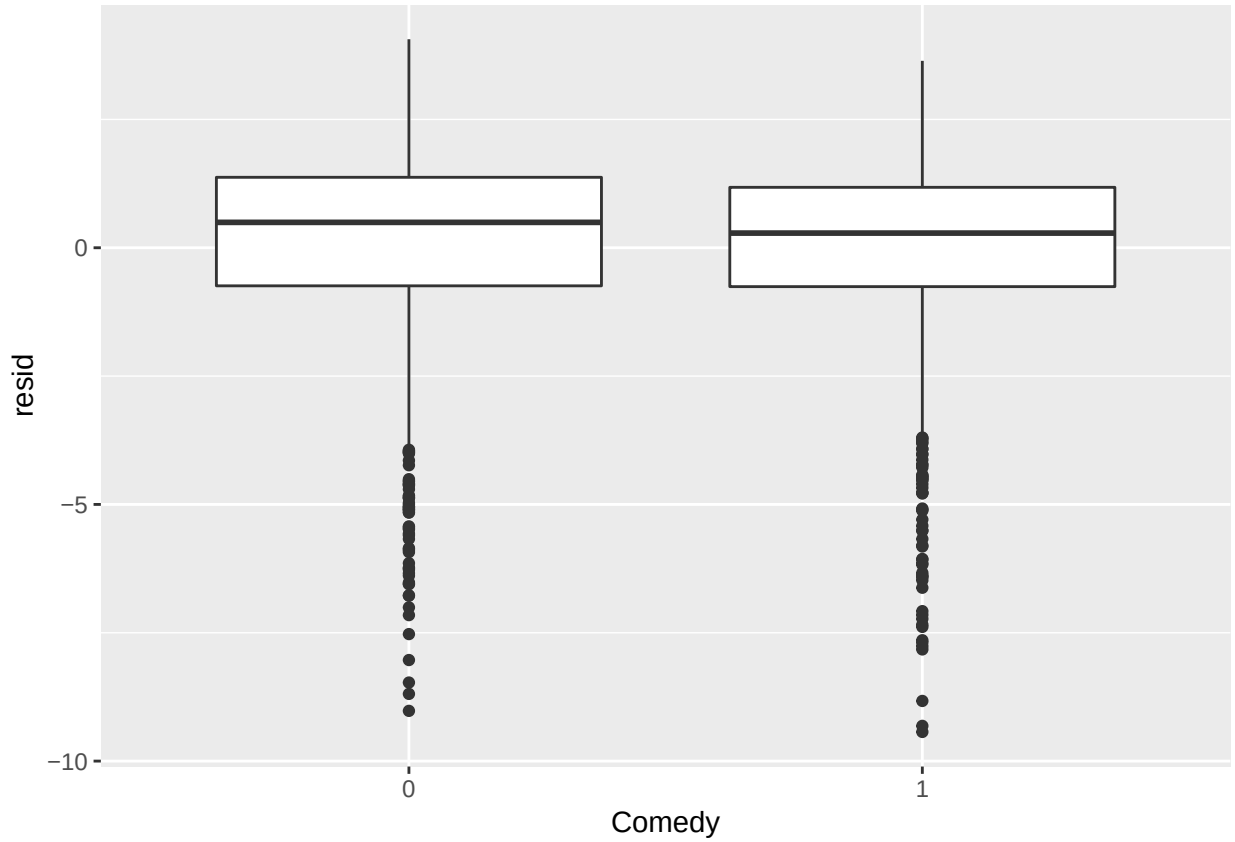
```
## [[1]]
```



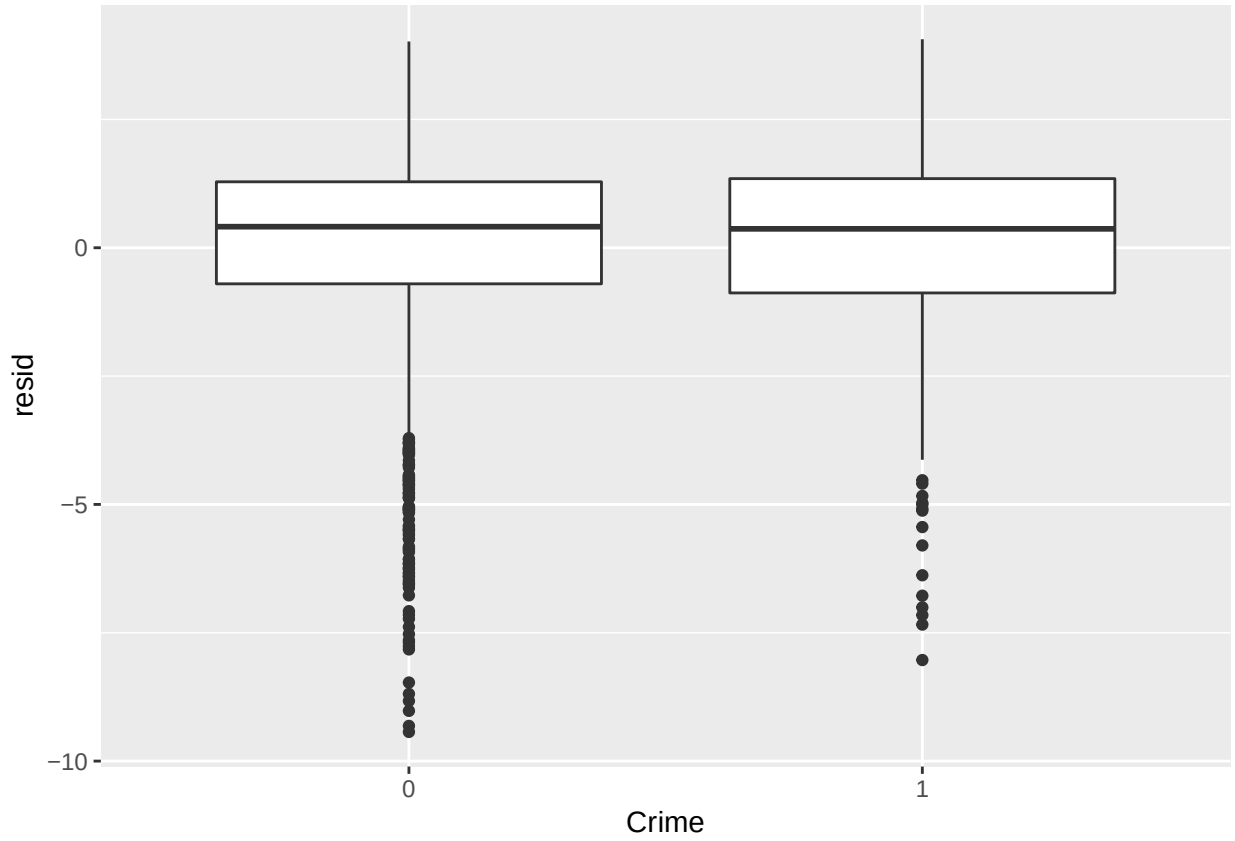

```
##
## [[2]]
```



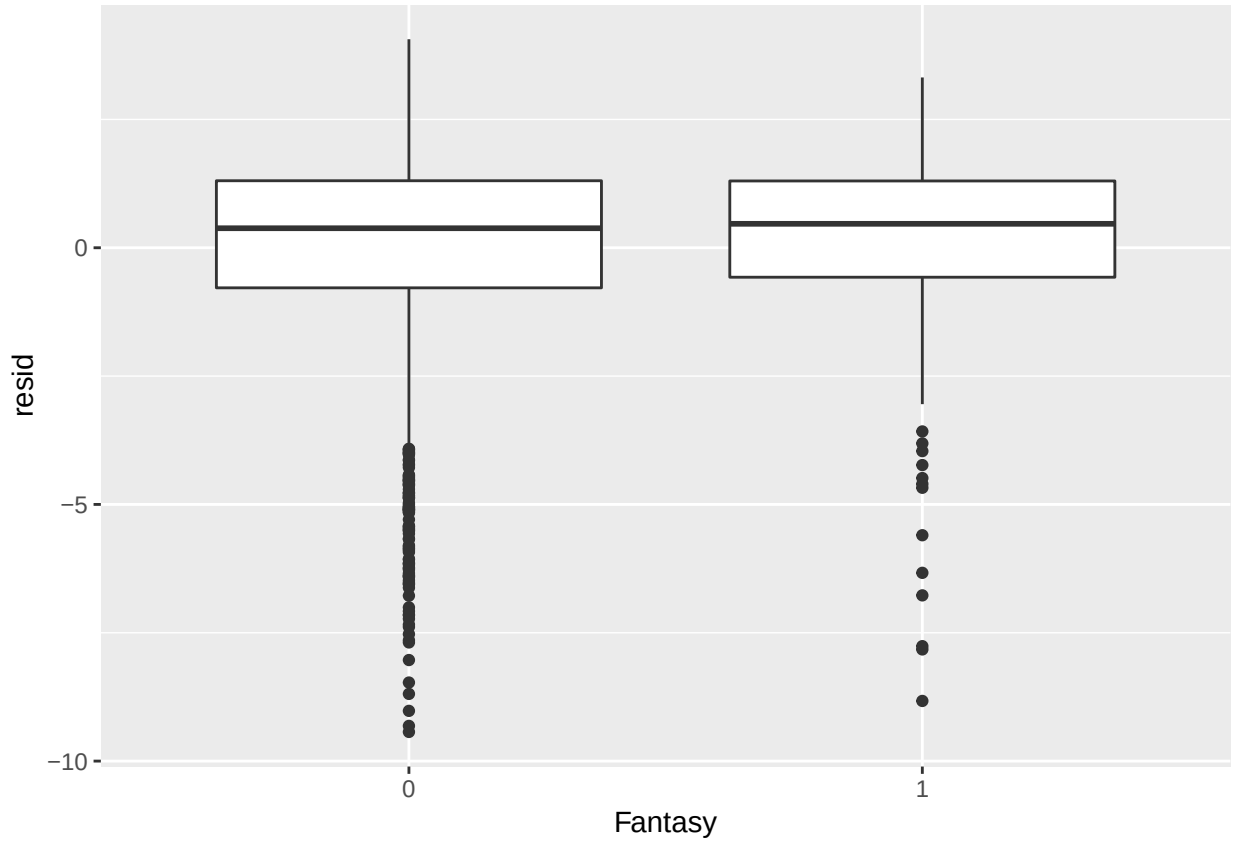

```
##
## [[3]]
```



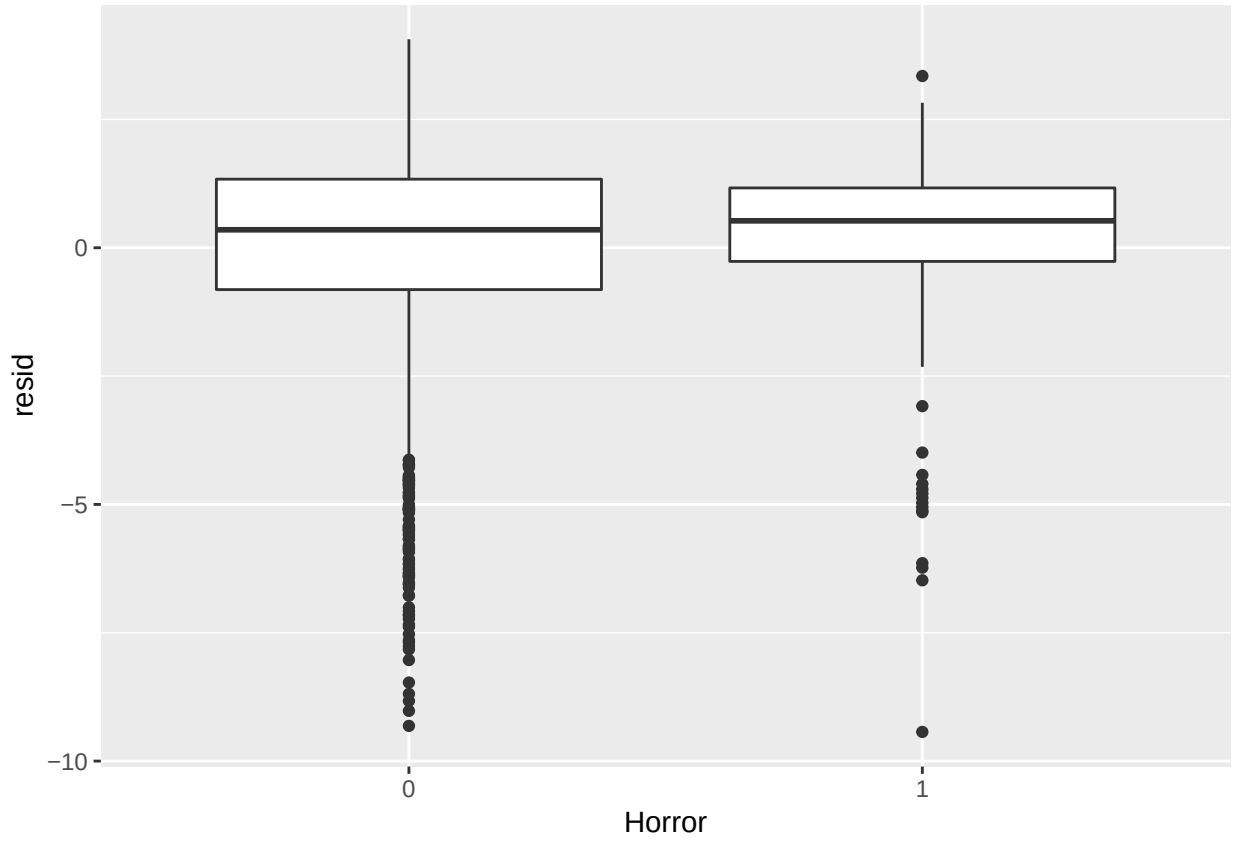

```
##
## [[4]]
```



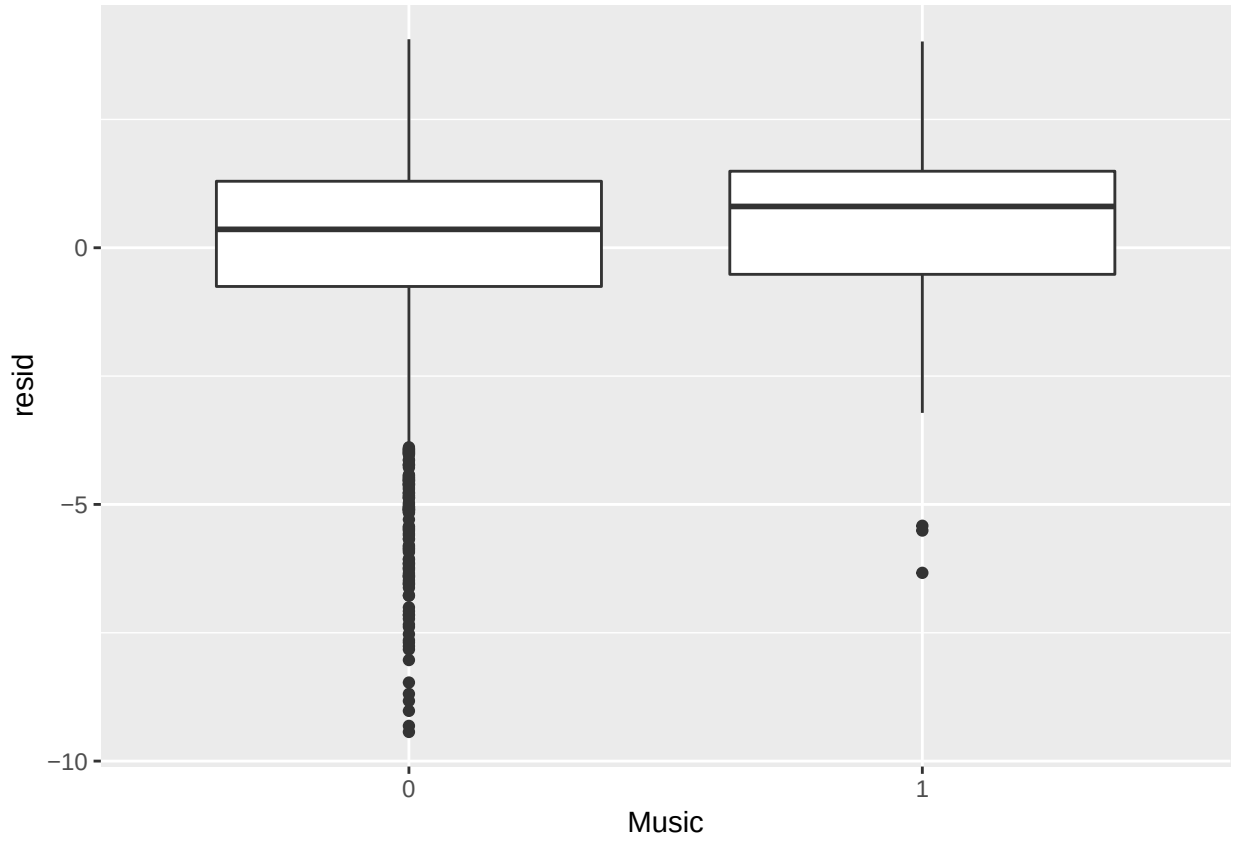

```
##
## [[5]]
```



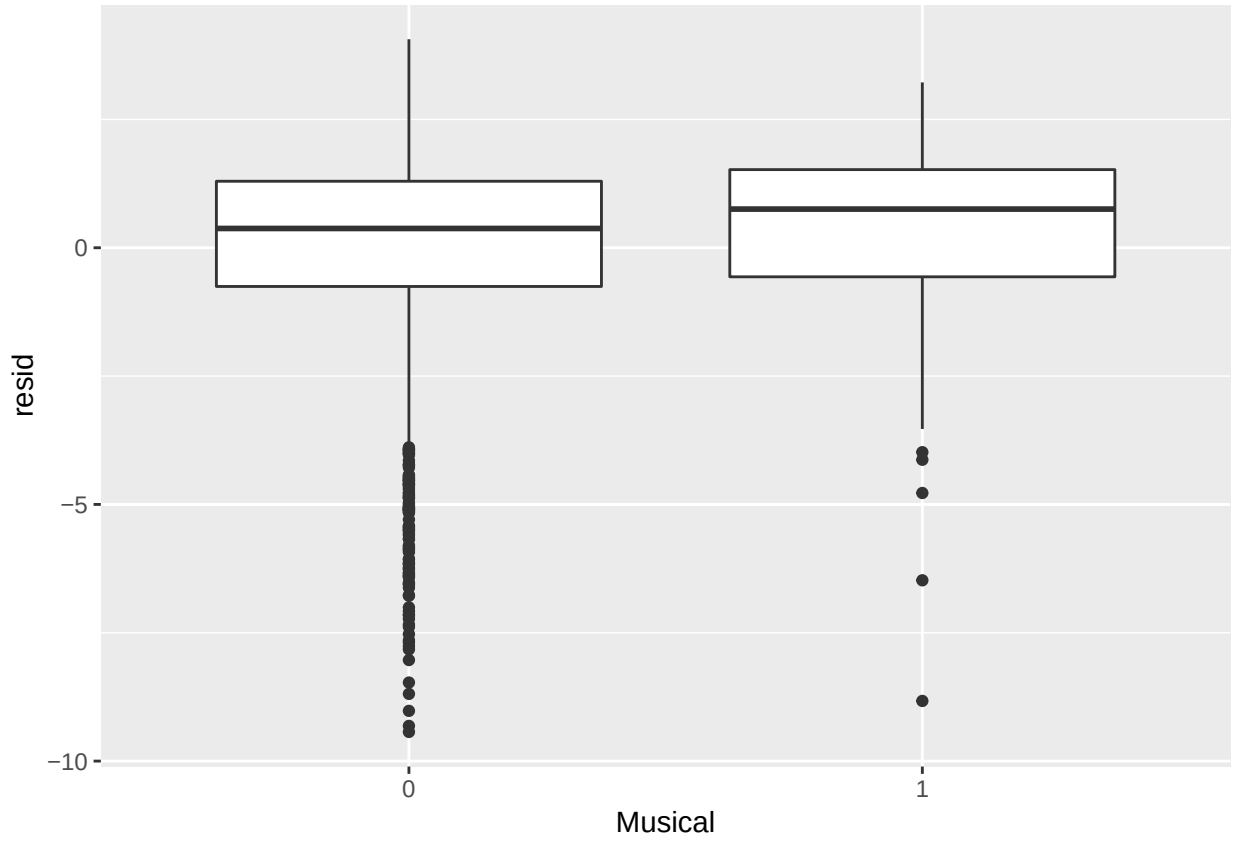

```
##
## [[6]]
```



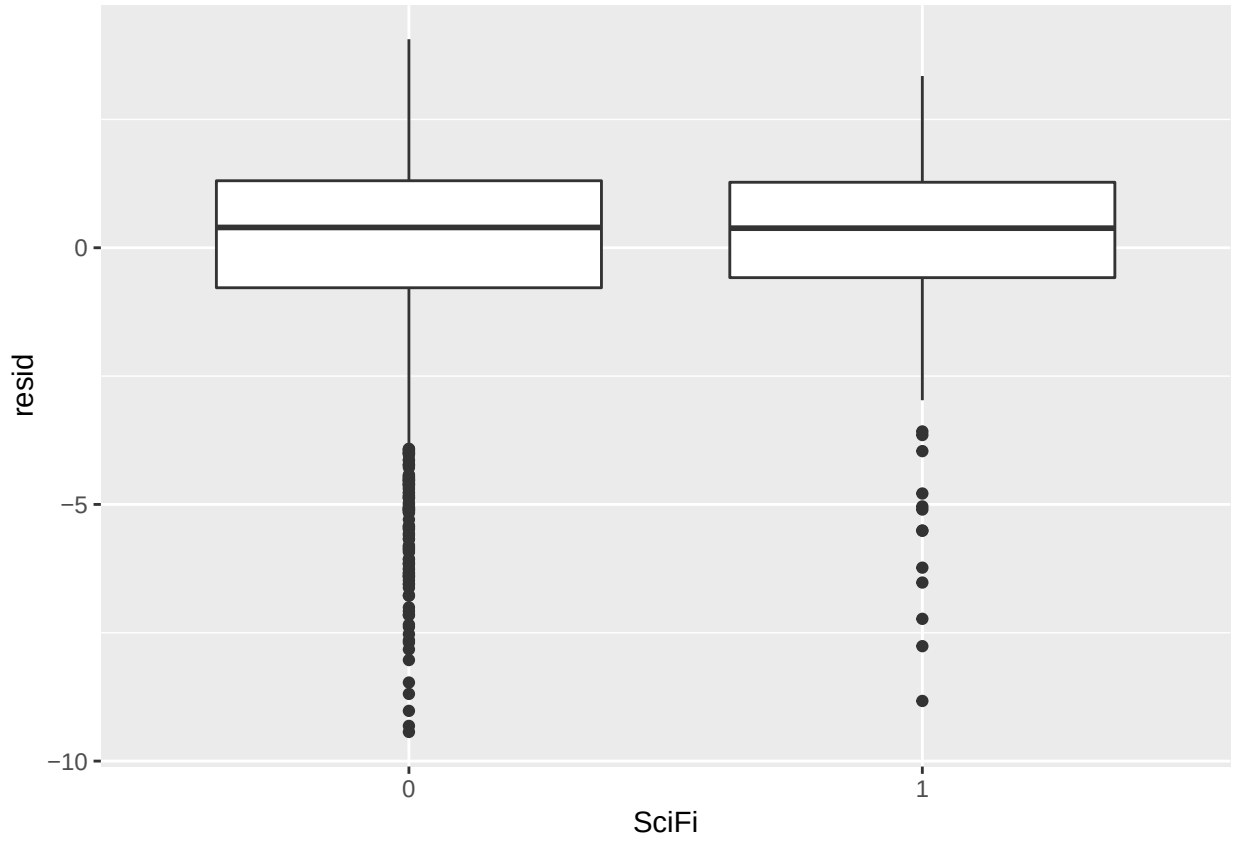

```
##
## [[7]]
```



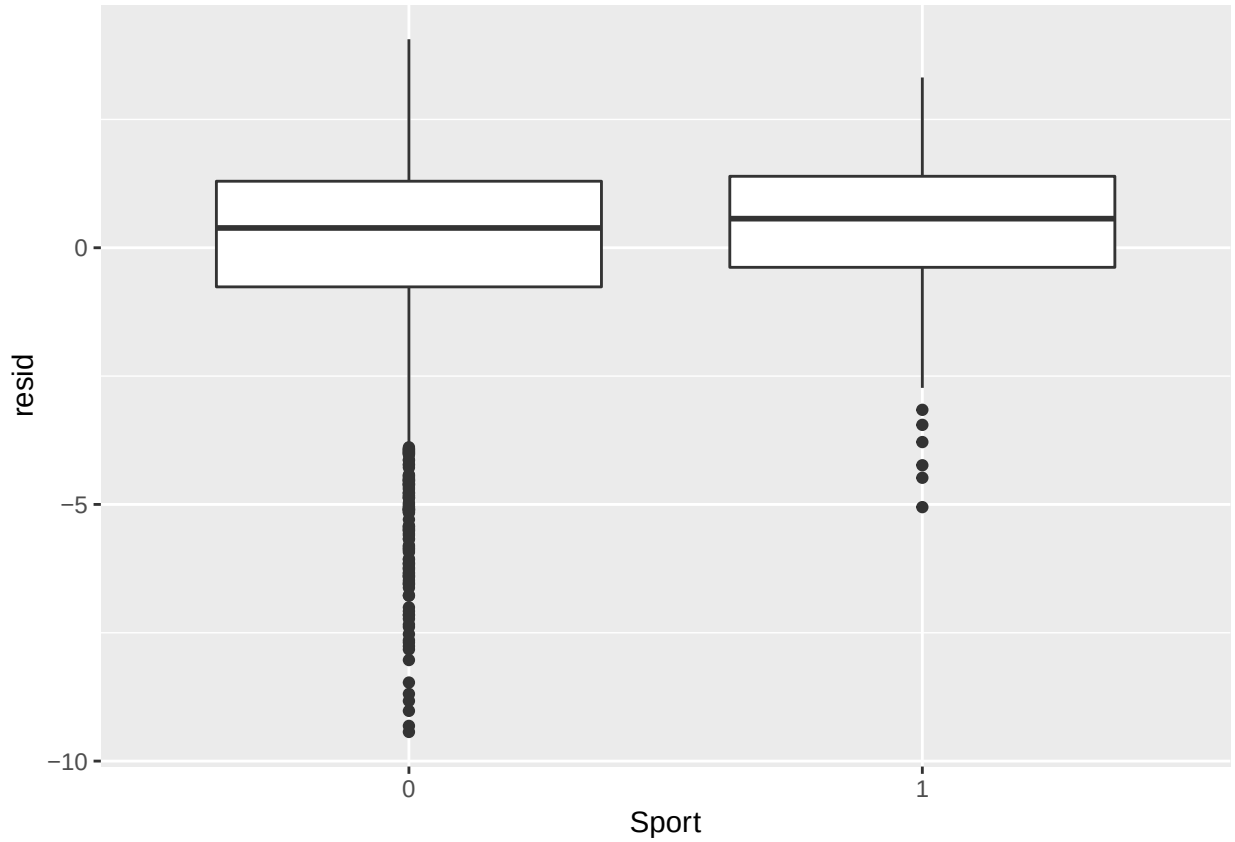

```
##
## [[8]]
```



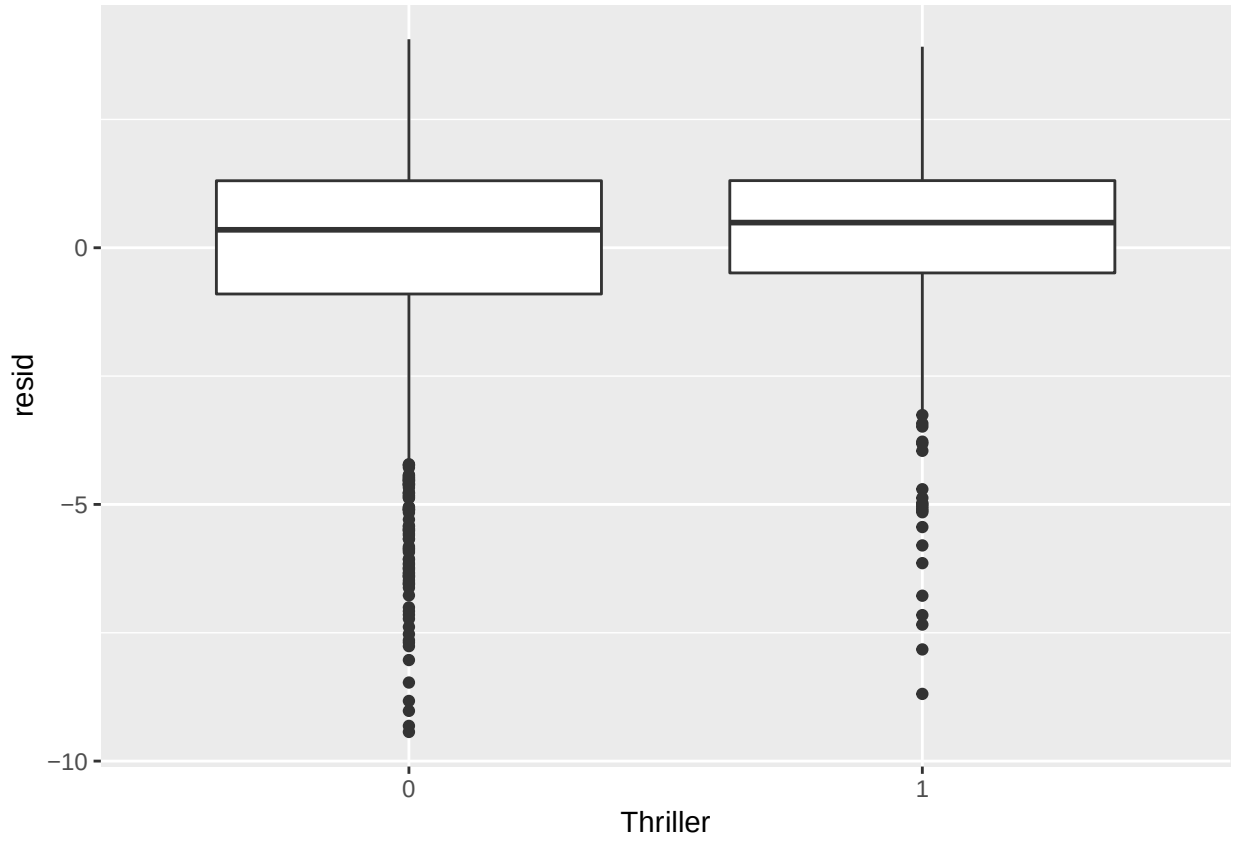

```
##
## [[9]]
```



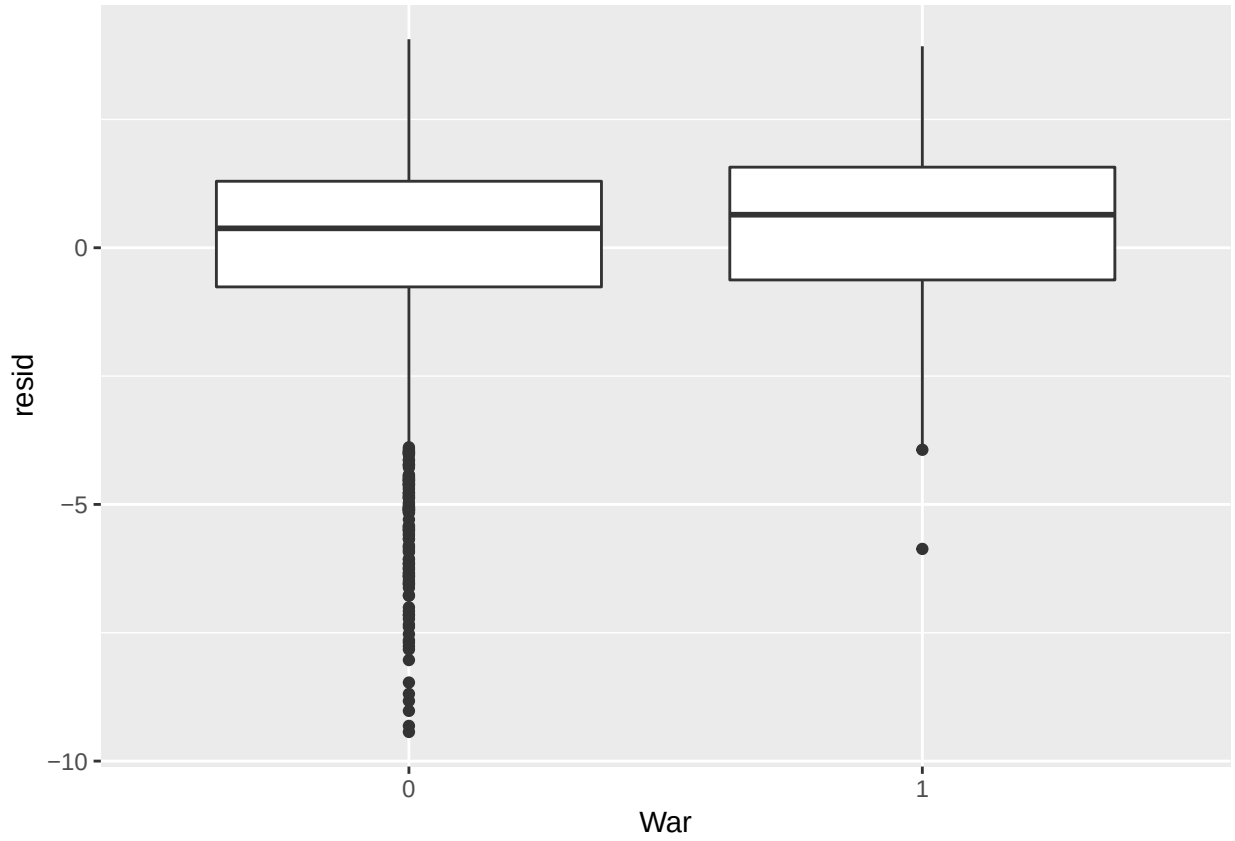

```
##
## [[10]]
```



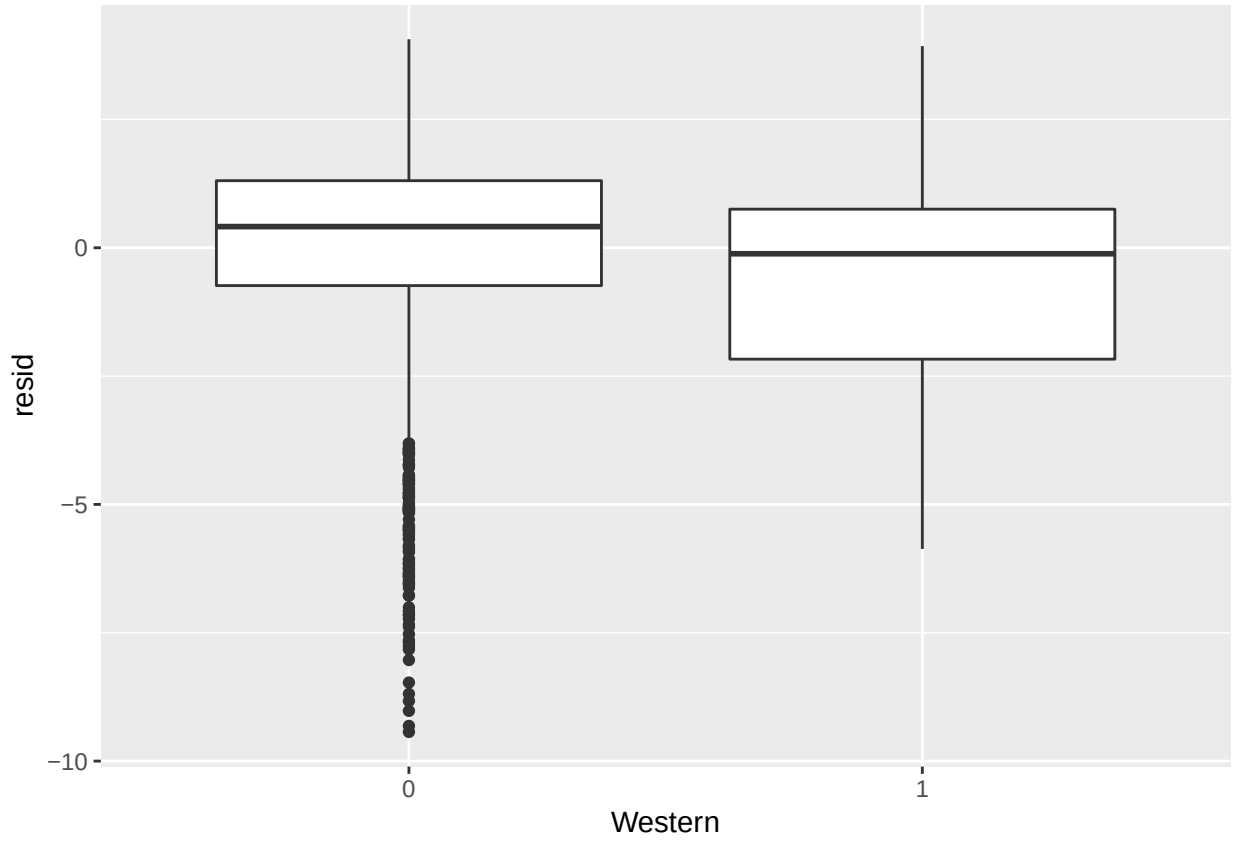

```
##
## [[11]]
```




```
##
## [[12]]
```

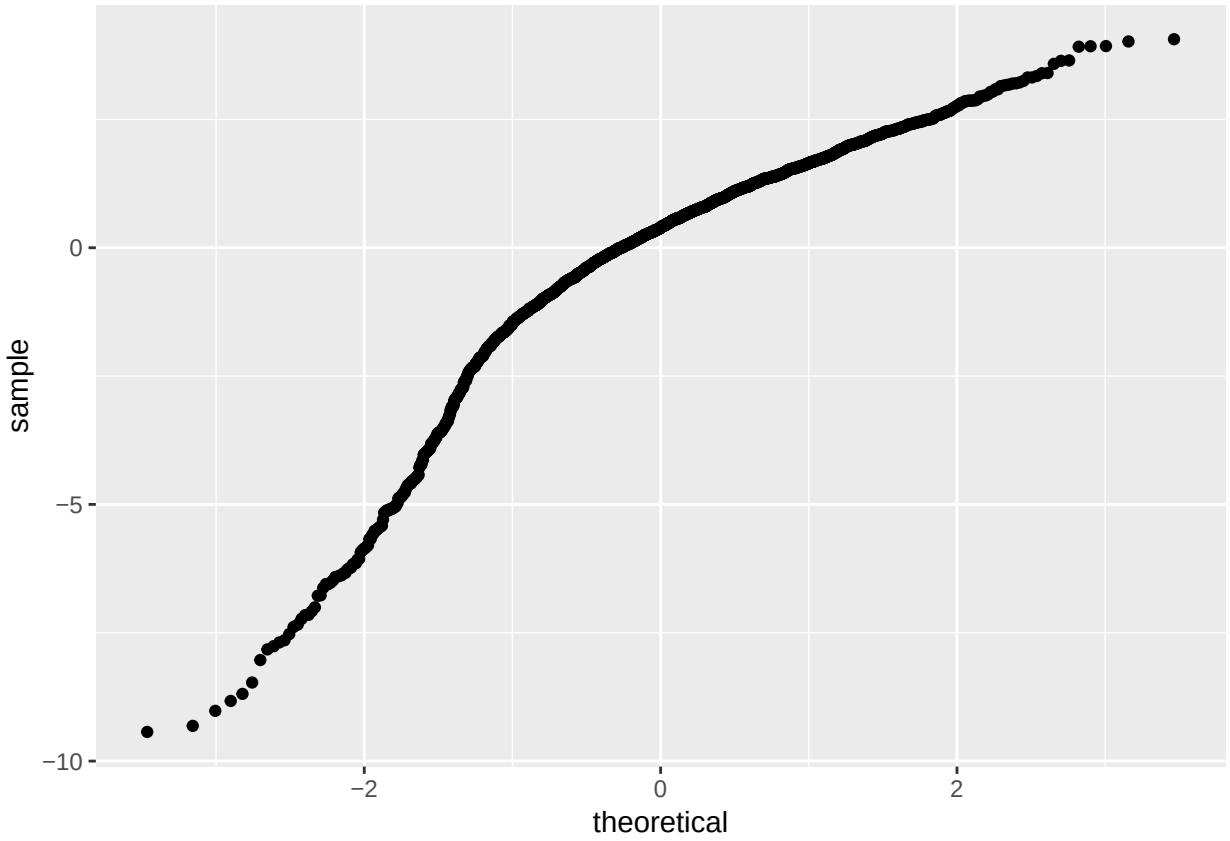



```
##
## [[13]]
```



Plot QQ plot for residuals. Not normally distributed, but close-ish.

```
# residuals themselves are NOT normally distributed
# qq plot
train %>% ggplot() +
  geom_qq(aes(sample = resid))
```



Glmnet: sparse

Quickly try this new method from class instead of stepwise. The sparse version does give us a lot of the same variables as stepwise. Good sign!
Can't do statistical tests, so not useful for analysis, but can use to aid justification.

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.5.3
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following object is masked from 'package:tidyr':
##
##     expand
```

```
## Loading required package: foreach
```

```
## Warning: package 'foreach' was built under R version 3.5.3
```

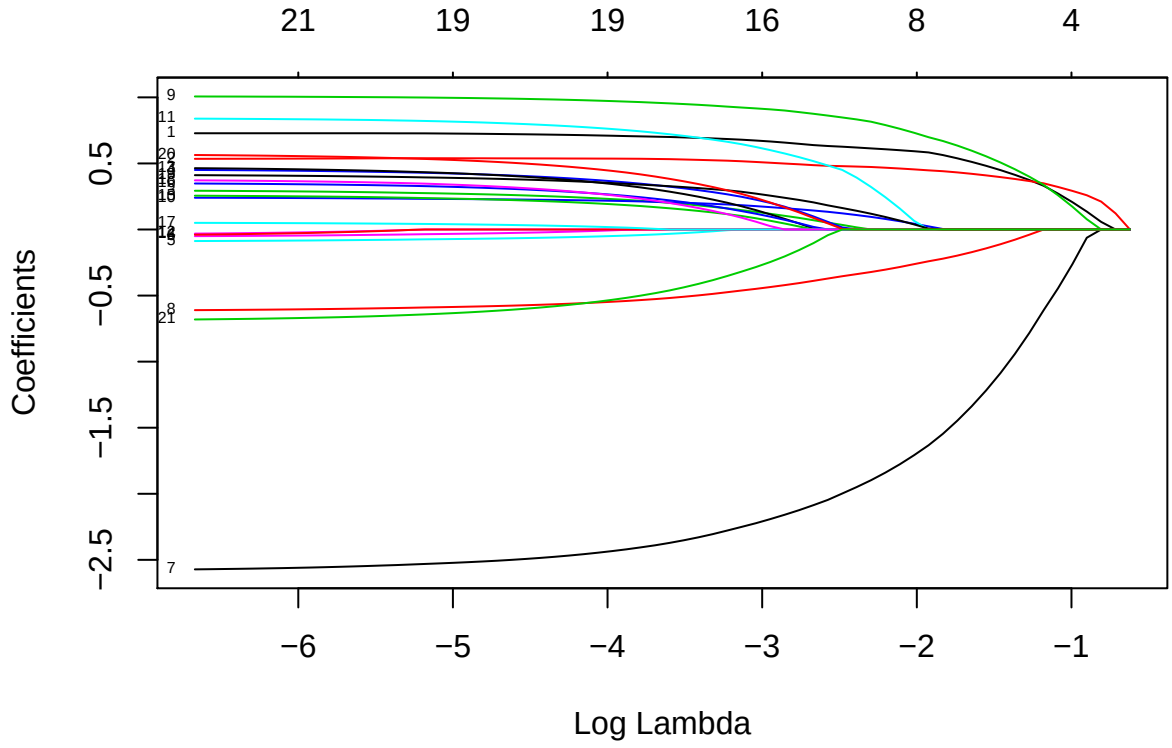
```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

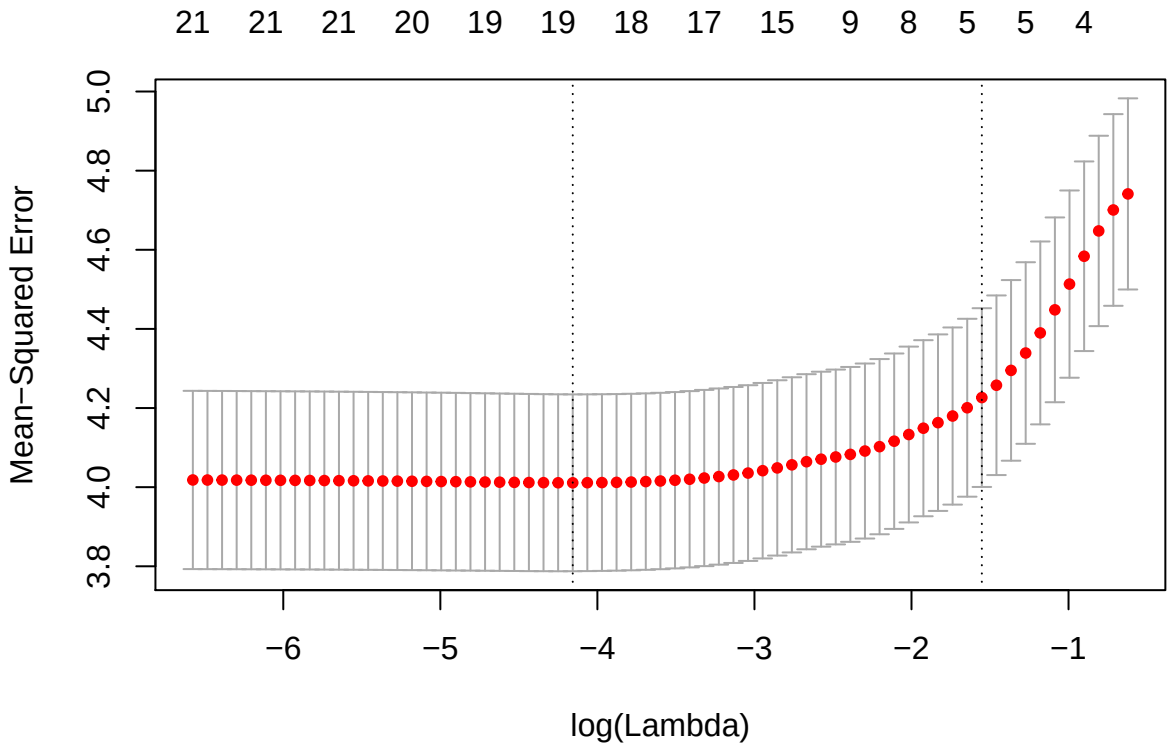
```
## Loaded glmnet 2.0-16
```

```r
# matrix of x and y variables
x <- as.matrix(train_genre)
y <- as.matrix(train$real_gross_log)

# glmnet process form class
mod <- glmnet(x, y, family = 'gaussian')
plot(mod, xvar = 'lambda', label = TRUE)
```




```r
mod2 <- cv.glmnet(x, y)
plot(mod2)
```




```
coef(mod2, s = 'lambda.min') # use min lambda
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) 16.53355864
## Action       0.71395632
## Adventure    0.53706521
## Animation    0.24100827
## Biography    0.38072732
## Comedy      -0.05552159
## Crime       -0.01350923
## Documentary -2.45736280
## Drama       -0.55721639
## Family       0.97836651
## Fantasy      0.22086352
## History      0.77520854
## Horror       .
## Music        0.36592441
## Musical      .
## Mystery      0.20234863
## Romance      0.27900549
## SciFi        0.02341647
## Sport        0.28039372
## Thriller     0.36192549
## War          0.46550531
## Western     -0.55895101
```

```r
coef(mod2, s = 'lambda.1se') # use most sparse
```

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 16.7121209
## Action       0.4819914
## Adventure    0.4095830
## Animation    .
## Biography    .
## Comedy       .
## Crime        .
## Documentary -1.2180058
## Drama       -0.1428957
## Family       0.5539177
## Fantasy      .
## History      .
## Horror       .
## Music        .
## Musical      .
## Mystery      .
## Romance      .
## SciFi        .
## Sport        .
## Thriller     .
## War          .
## Western      .
```