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() # 101

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_only <- 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
# df_vars is the dataset with only the relevant variables
# var_lst is the list of variables that are in the base model
# formula is the formula with those variables besides the y variable
step_wise_step <- function(df_vars, var_lst = NULL, formula = NULL) {
  # if first step
  if (length(var_lst) == 0) {
    # rmse with each variable against real_gross
    rmse_vars <- sapply(names(df_vars), function(var) {
      # rmse of model
      rmse(lm(as.formula(str_c('real_gross_log ~', var)), data = train), data = valid)
    })
  # if > first step: exclude variables from var_lst from data and include in model formula
  } else {
    rmse_vars <- sapply(names(df_vars %>% select(-var_lst)), function(var) {
      # rmse of model
      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_vars[which.min(rmse_vars)])
}

# function to loop through each step wise loop
# adding optional starting vars and formula in case want to build off of an existing formula
step_wise_loop <- function(df_vars, starting_vars = NULL, starting_formula = NULL) {
  # list to store min RMSE from each step in
  rmse_lst <- c()

  # first step: no genre_lst or formula (default values NULL)
  min_rmse_var <- step_wise_step(df = df_vars, var_lst = starting_vars, formula = starting_formula)
  print(min_rmse_var)

  # add to list of genres, formula, and min RMSE list
  var_lst <- c(starting_vars, names(min_rmse_var))
  formula <- str_c(starting_formula, '+', names(min_rmse_var))
  rmse_lst <- c(rmse_lst, min(min_rmse_var))

  # if have starting variables, take those out of the number we are iterating through
  if (!is.null(starting_vars)) {
    df_vars_seq <- df_vars %>% select(-starting_vars)
  } else {
    df_vars_seq <- df_vars
  }
  # loop through until have considered every variable
  for (i in seq(1:(ncol(df_vars_seq)-1))) {
    print(i)
    # step
    min_rmse_var <- step_wise_step(df = df_vars, var_lst = var_lst, formula = formula)
```

```
    print(min_rmse_var)

    # add to lists
    var_lst <- c(var_lst, names(min_rmse_var))
    formula <- str_c(formula, ' + ', names(min_rmse_var))
    rmse_lst <- c(rmse_lst, min(min_rmse_var))
  }
  return(rmse_lst)
}
```

```
# step wise implement
# return list of all min RMSE from each step -> graph
rmse_lst <- step_wise_loop(df = train_genre_only)
```

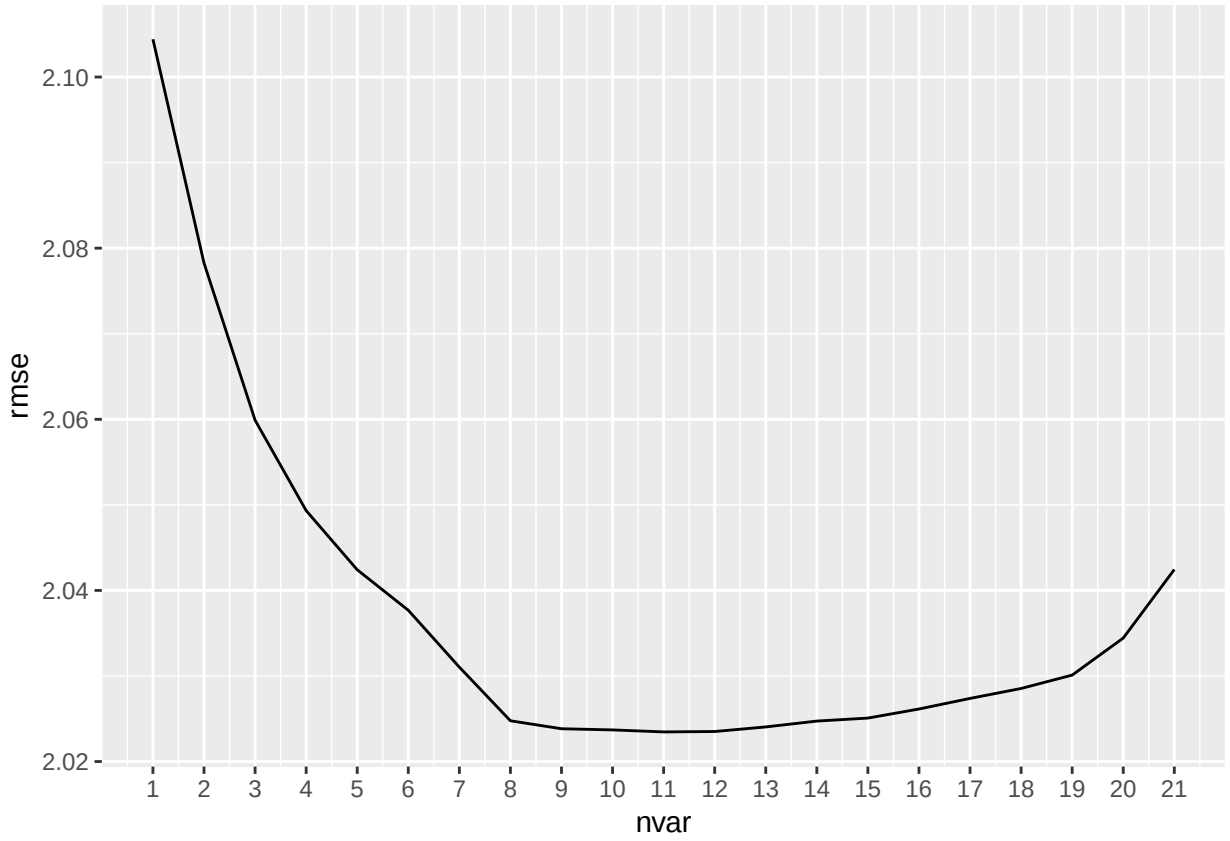
```
## Adventure
##  2.104417
## [1] 1
##   Action
## 2.078261
## [1] 2
##   Family
## 2.059892
## [1] 3
##  Mystery
## 2.049328
## [1] 4
## Documentary
##    2.042411
## [1] 5
##   Drama
## 2.03768
## [1] 6
##  History
## 2.031034
## [1] 7
##  Romance
## 2.024762
## [1] 8
##      War
## 2.023833
## [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_genre <- lm(real_gross_log ~ Adventure + Action + Family + Mystery +
            Documentary + Drama + History + Romance,
          data = train)

summary(mod_genre)
```

```
##
## 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 ***
## Adventure1    0.57443    0.13767   4.173 3.15e-05 ***
## Action1       0.85492    0.12393   6.899 7.15e-12 ***
## Family1       1.08300    0.15541   6.969 4.42e-12 ***
```

```
## Mystery1      0.42172    0.16239   2.597  0.00948 **
## Documentary1 -2.50278    0.30128  -8.307  < 2e-16 ***
## Drama1       -0.53082    0.10172  -5.218 2.01e-07 ***
## History1      1.11219    0.27919   3.984 7.05e-05 ***
## Romance1      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_genre, data = valid)
```

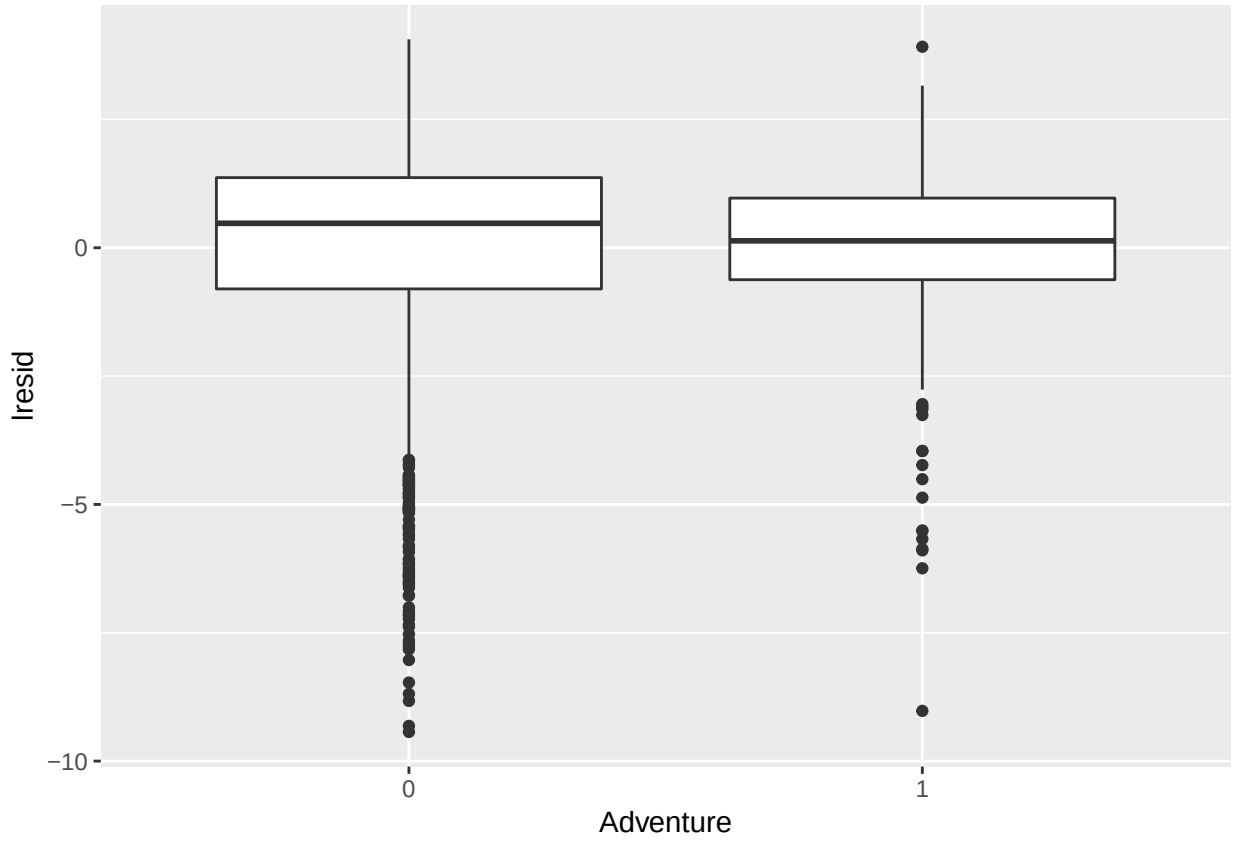
```
## [1] 2.024762
```

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

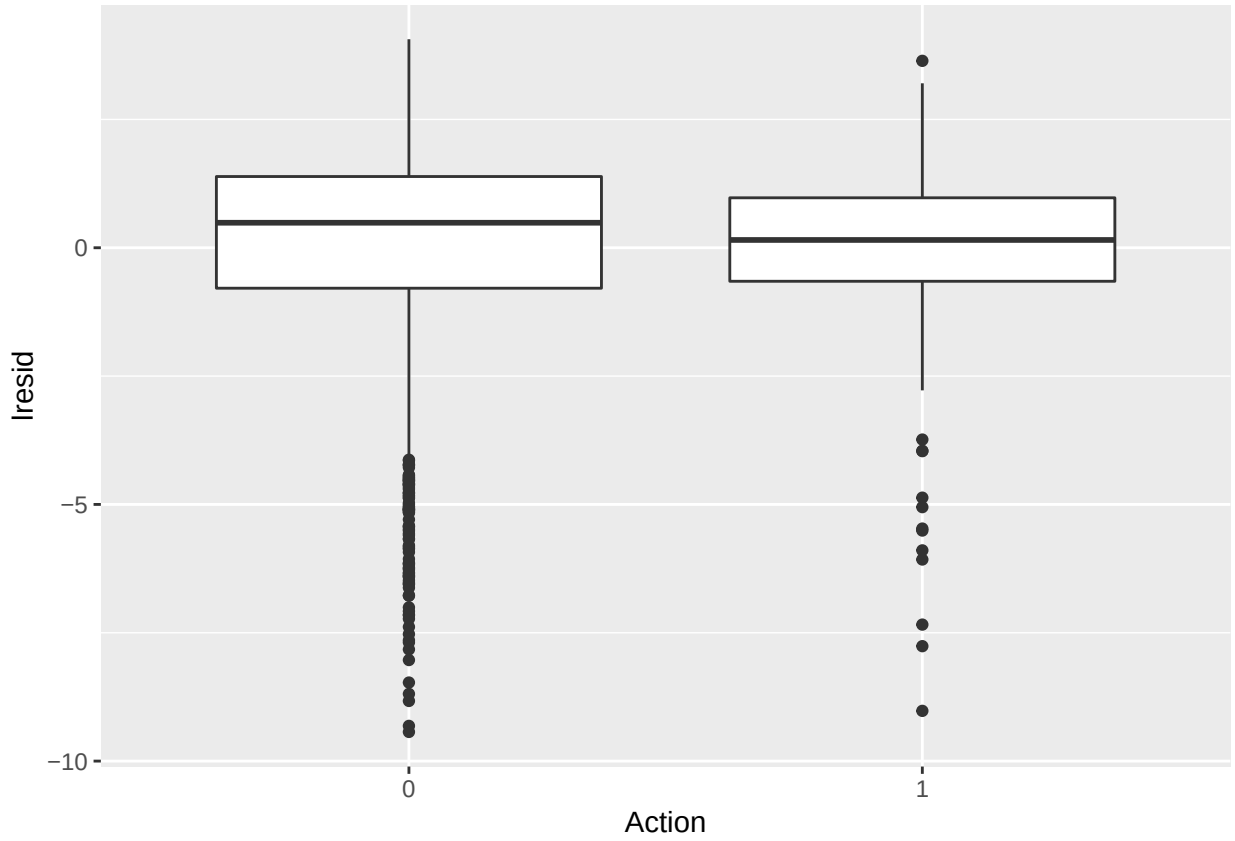
Graph variables in and out of model against residuals. Most are fairly evenly distributed around residual. Worst is probably Western.

```
# graph residuals against each variable included in the model
# most look random except Adventure
train_resid <- train %>%
  add_residuals(mod_genre, 'lresid')

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

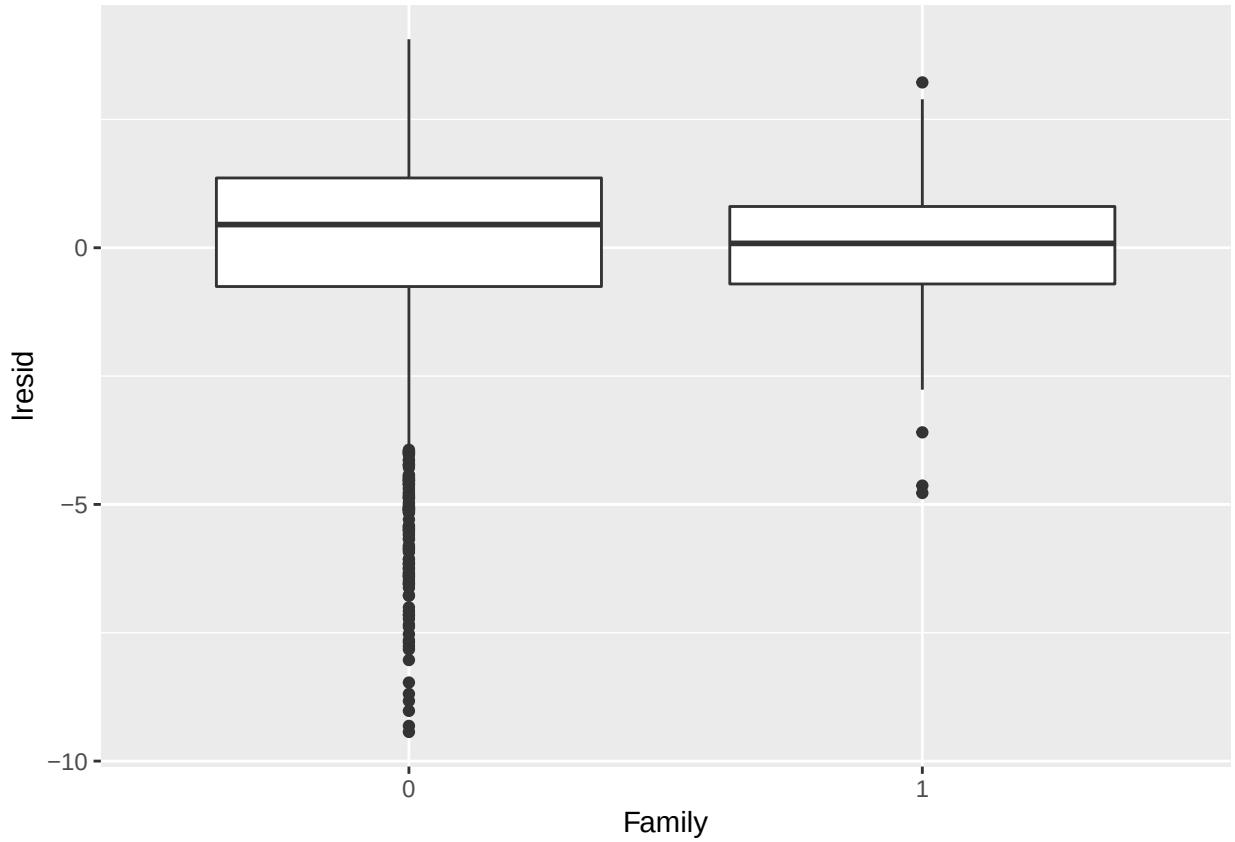
```
## [[1]]
```



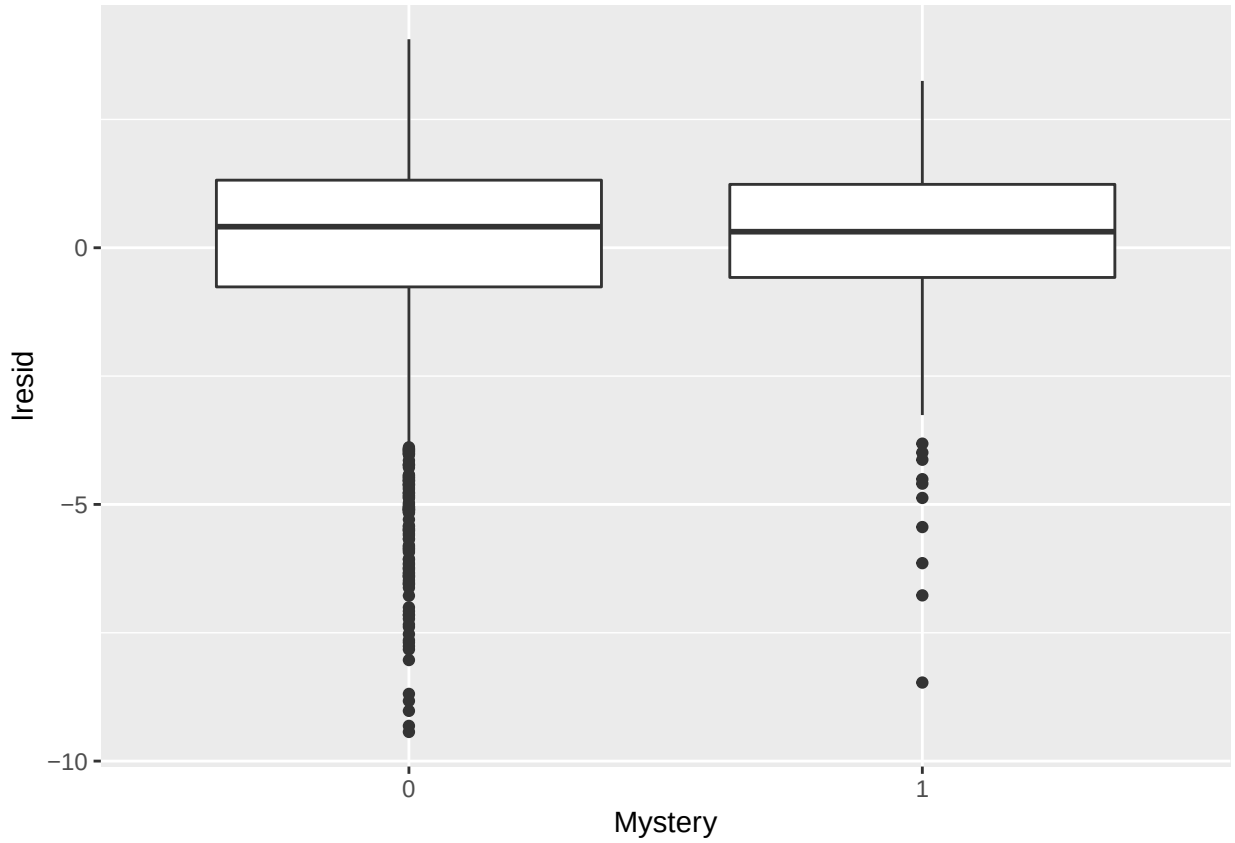

```
##
## [[2]]
```



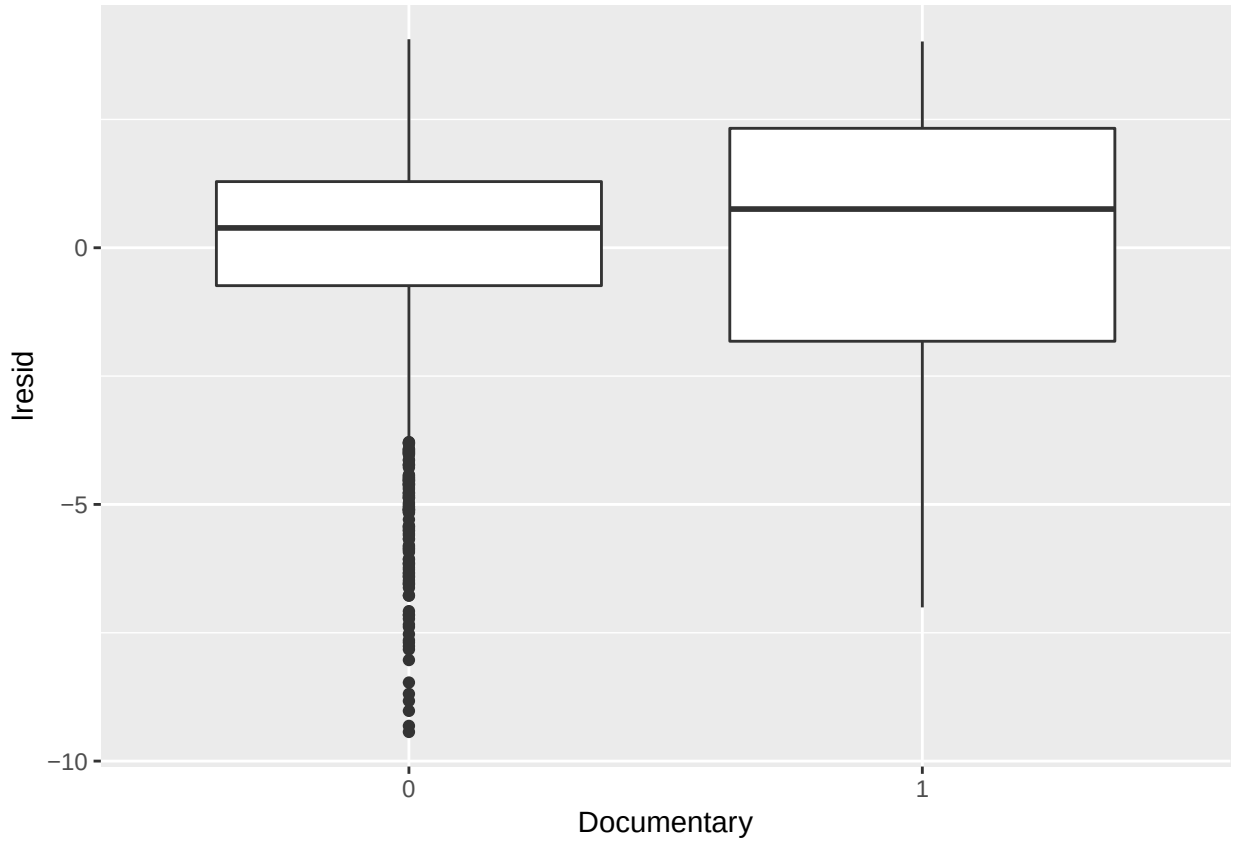

```
##
## [[3]]
```



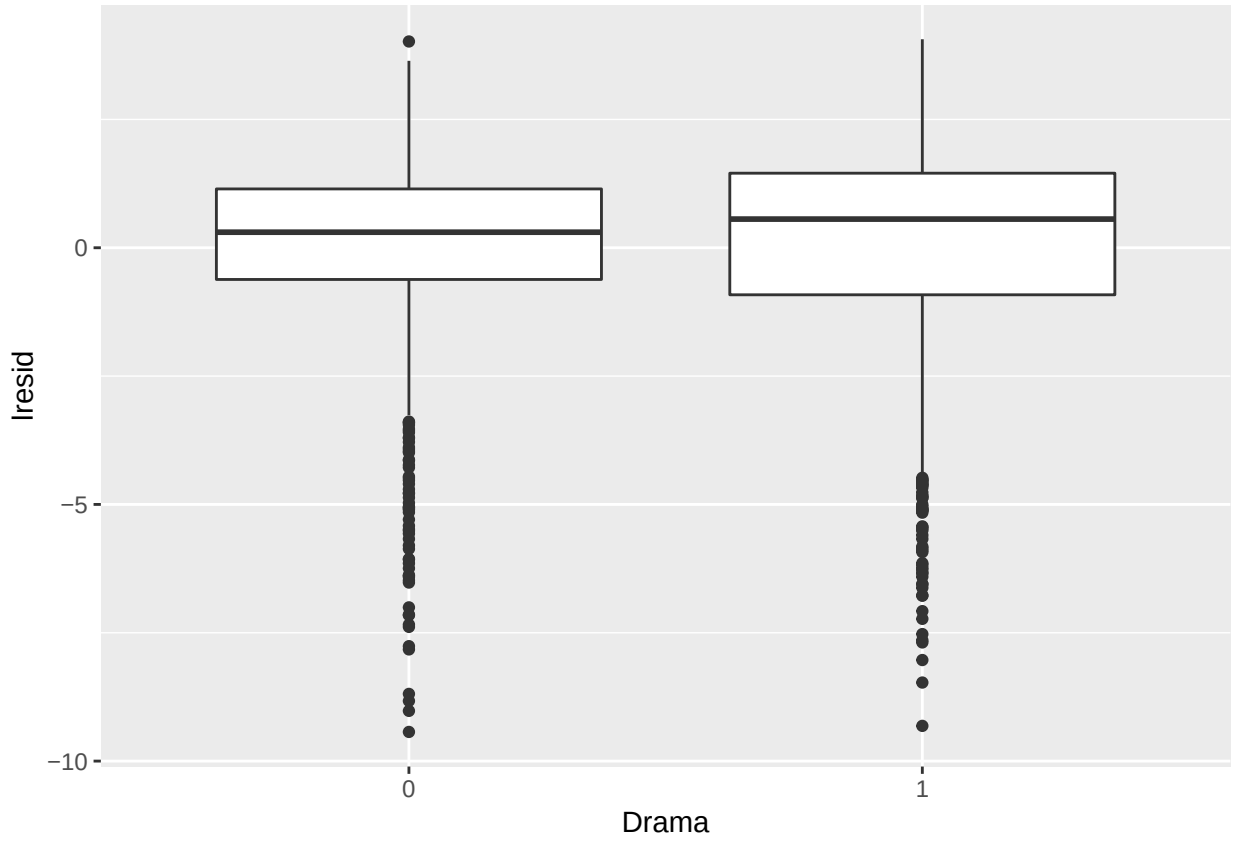

```
##
## [[4]]
```



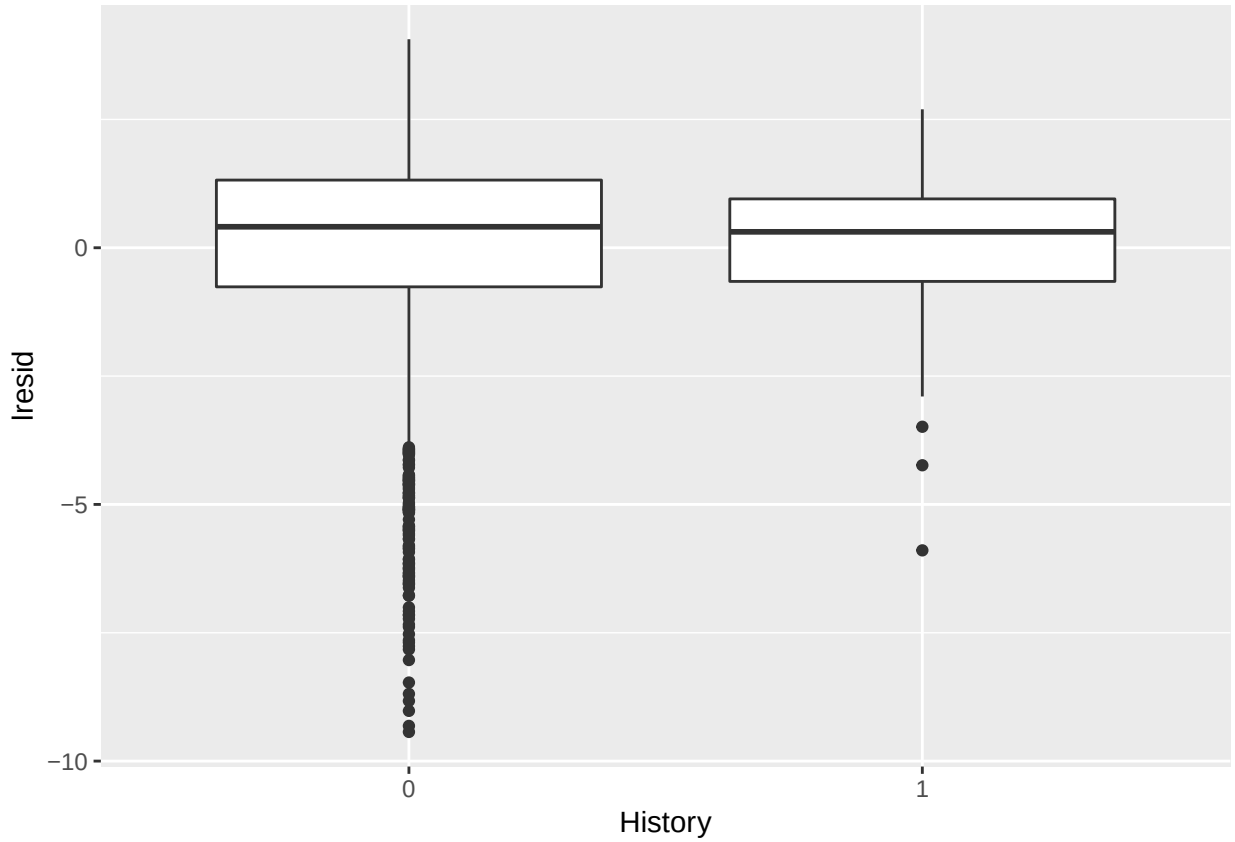

```
##
## [[5]]
```



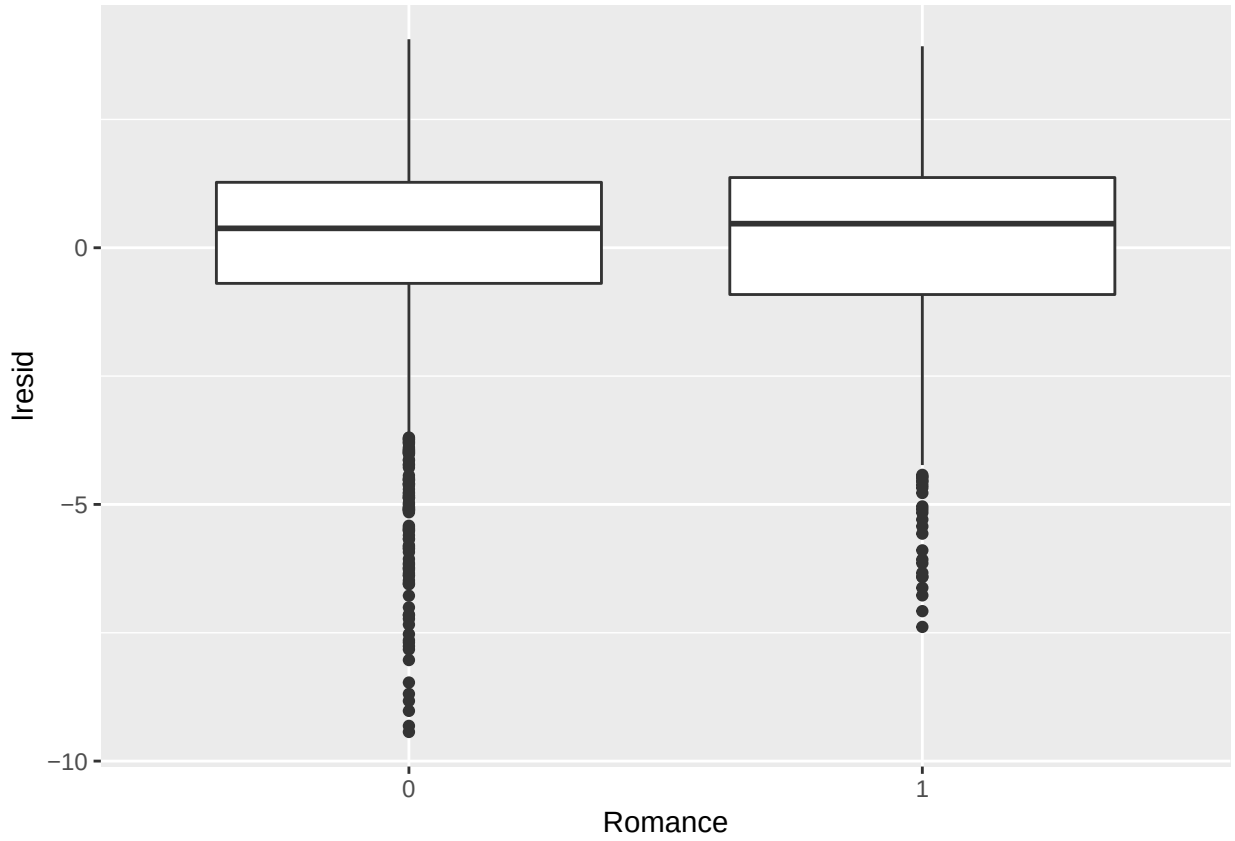

```
##
## [[6]]
```




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

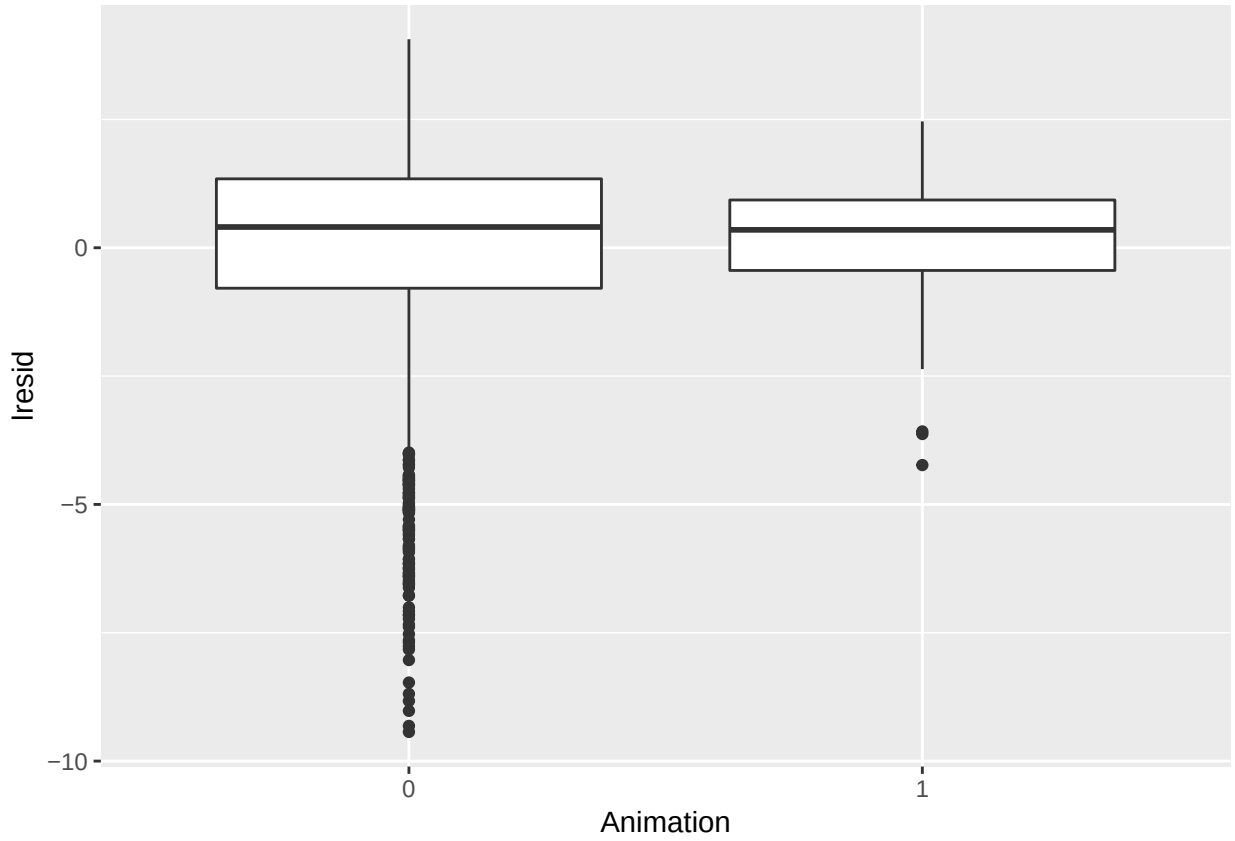



```
##
## [[8]]
```

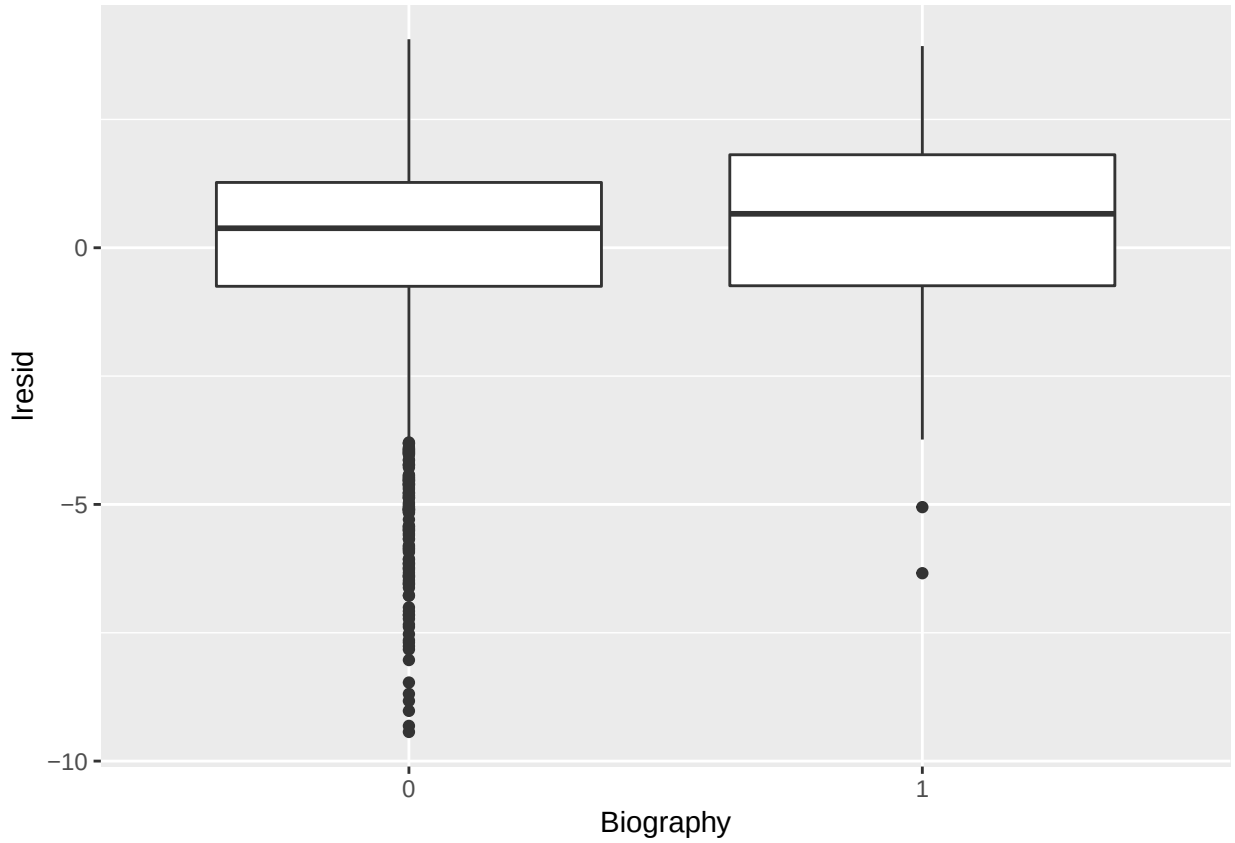



```
# graph residuals against each genre not included in the model
# several are questionable if random. Especially Animation.
lapply(names(train_genre_only %>% select(-genre_xvar)), function(var) {
  train_resid %>%
    ggplot() +
    geom_boxplot(aes_string(var, y = 'lresid'))
})
```

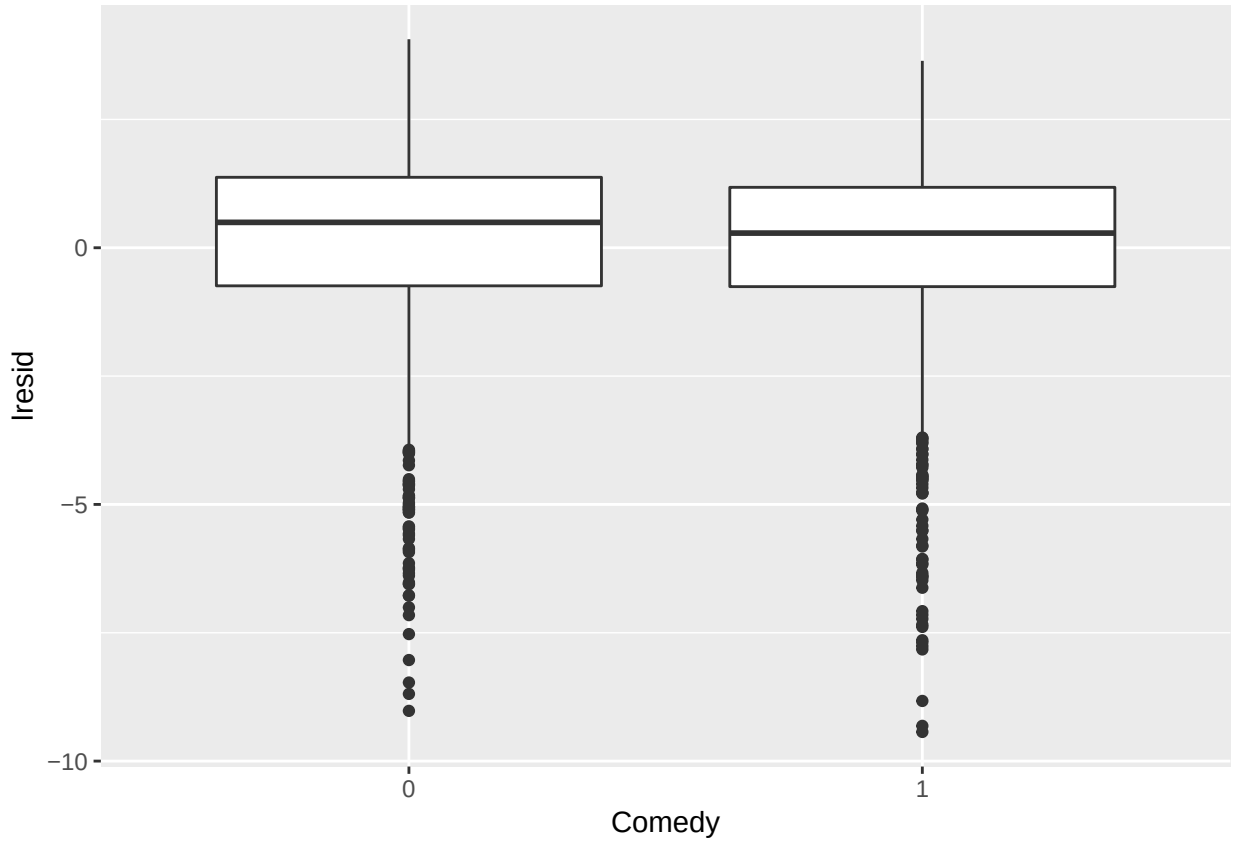
```
## [[1]]
```



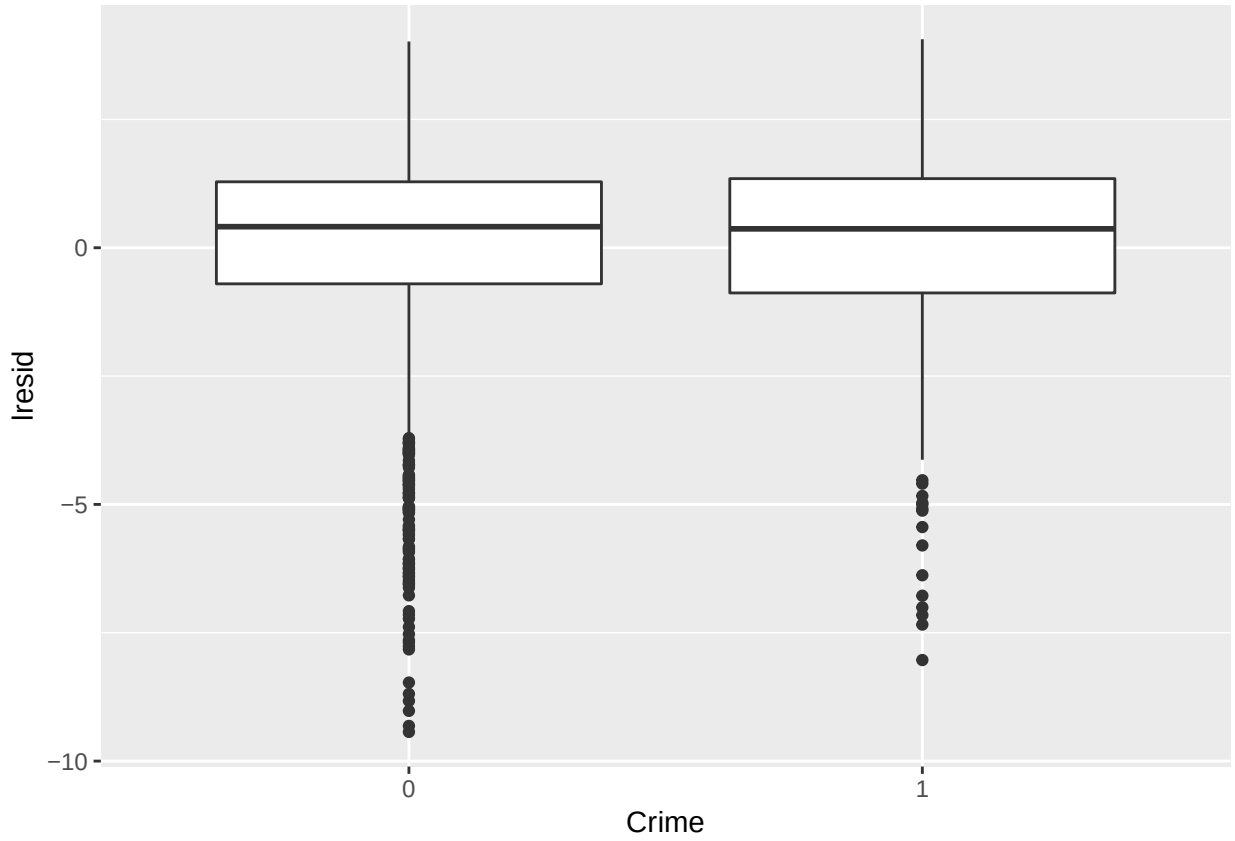

```
##
## [[2]]
```



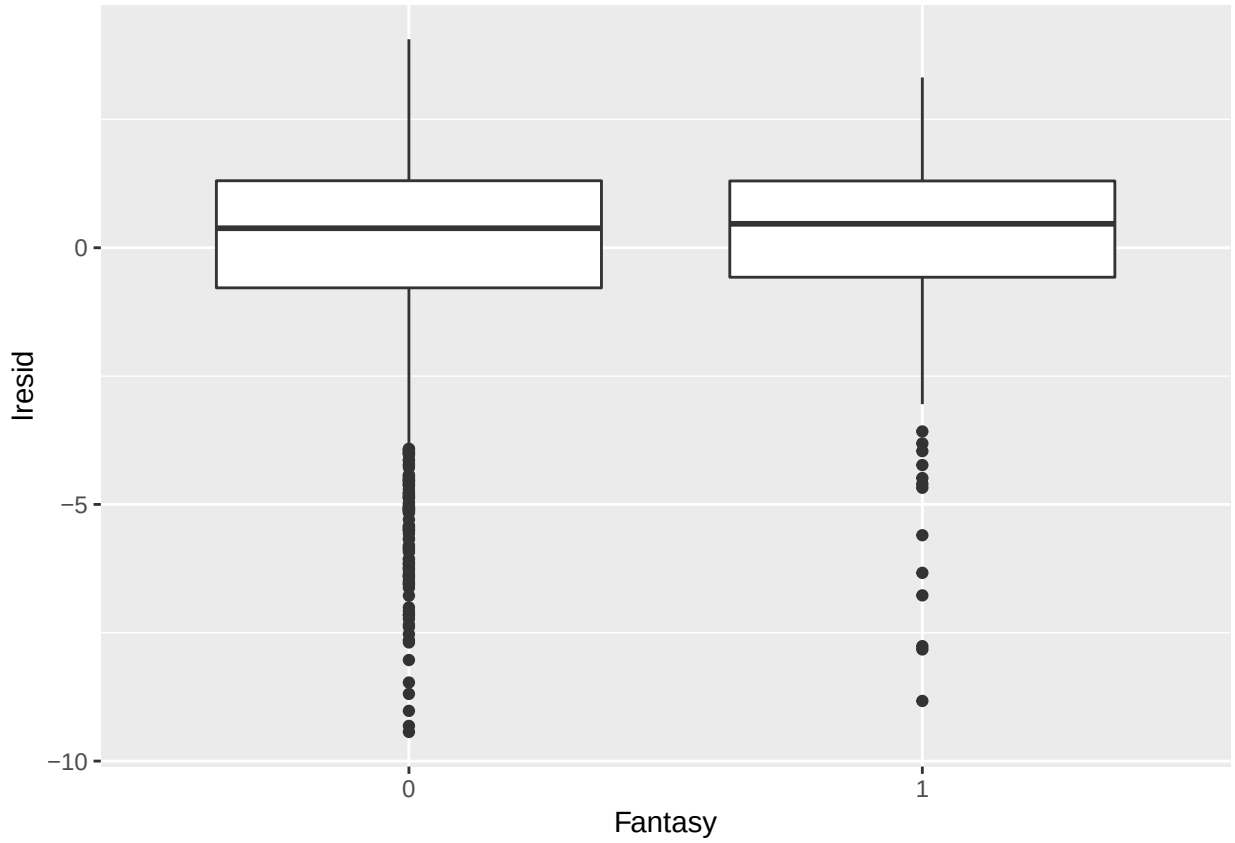

```
##
## [[3]]
```



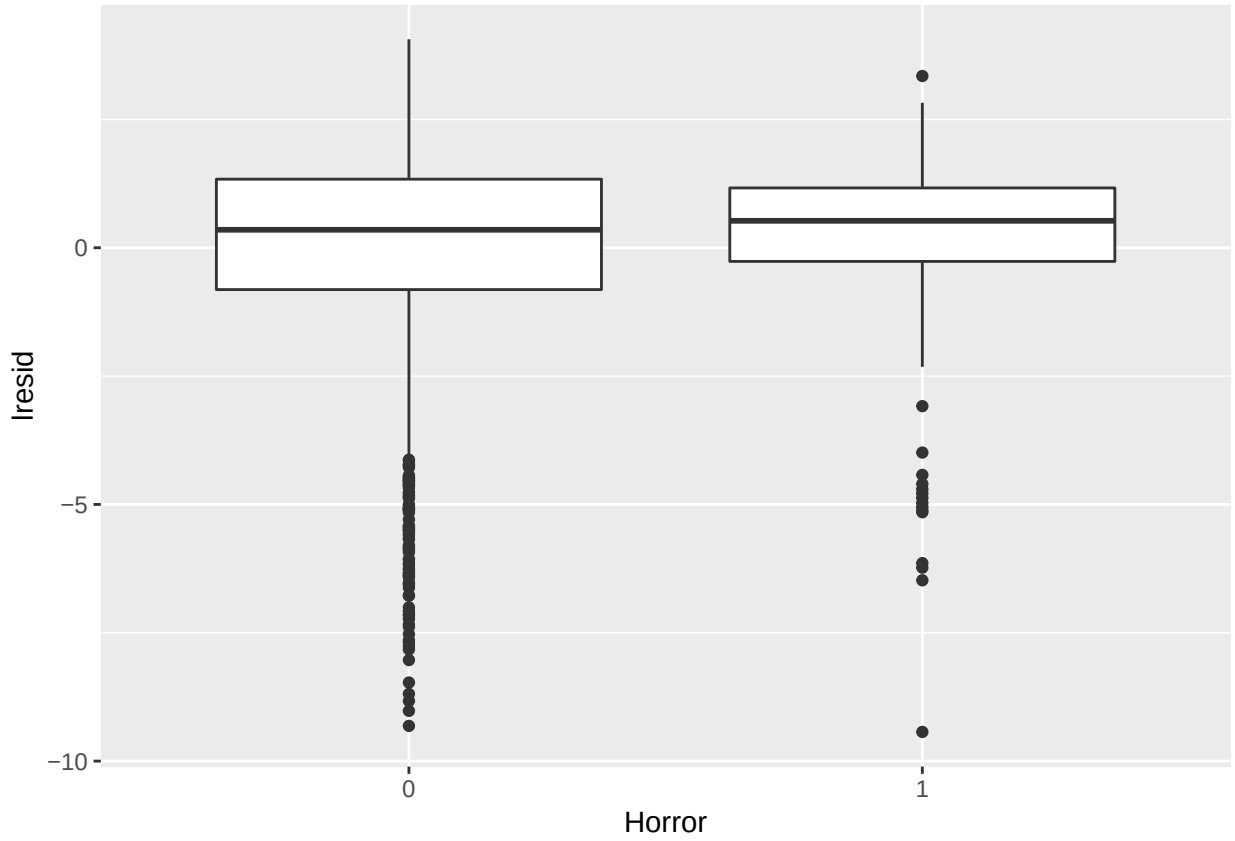

```
##
## [[4]]
```



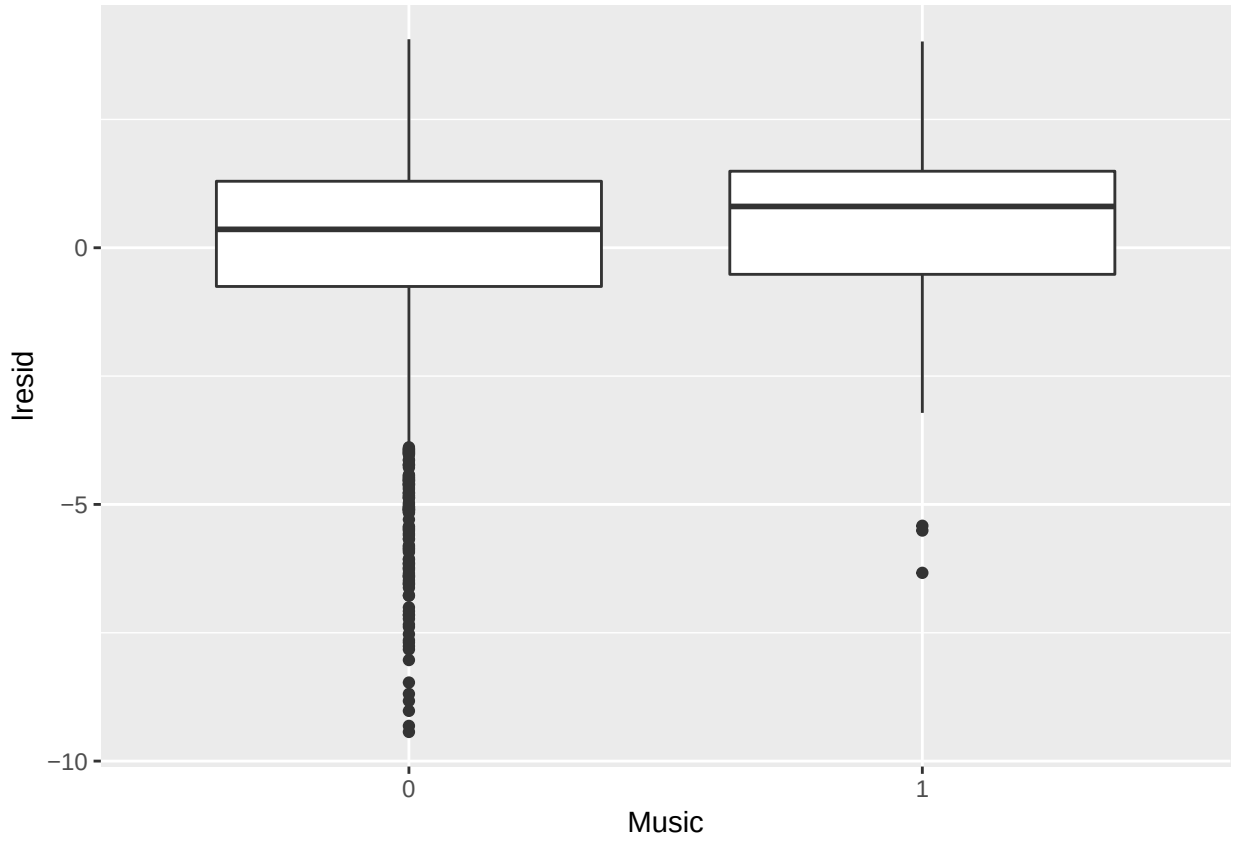

```
##
## [[5]]
```



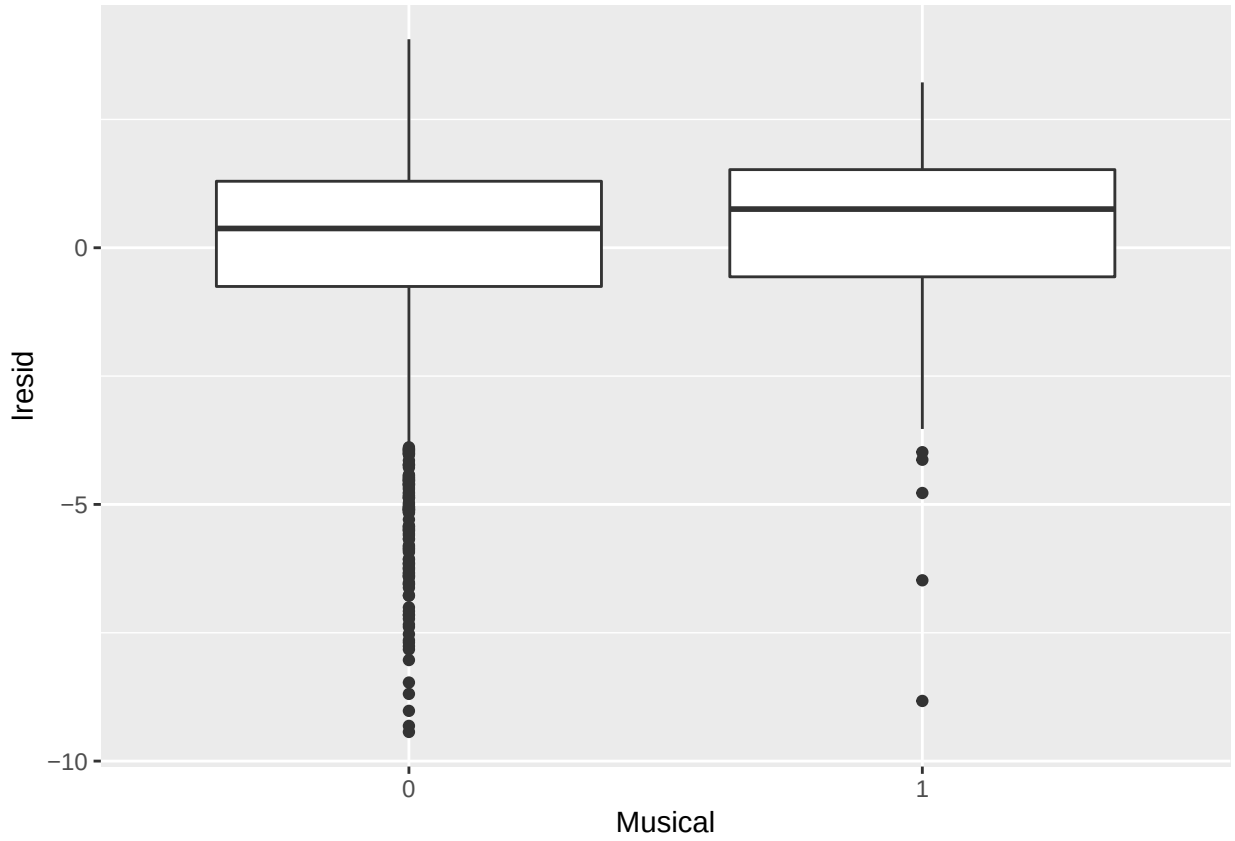

```
##
## [[6]]
```



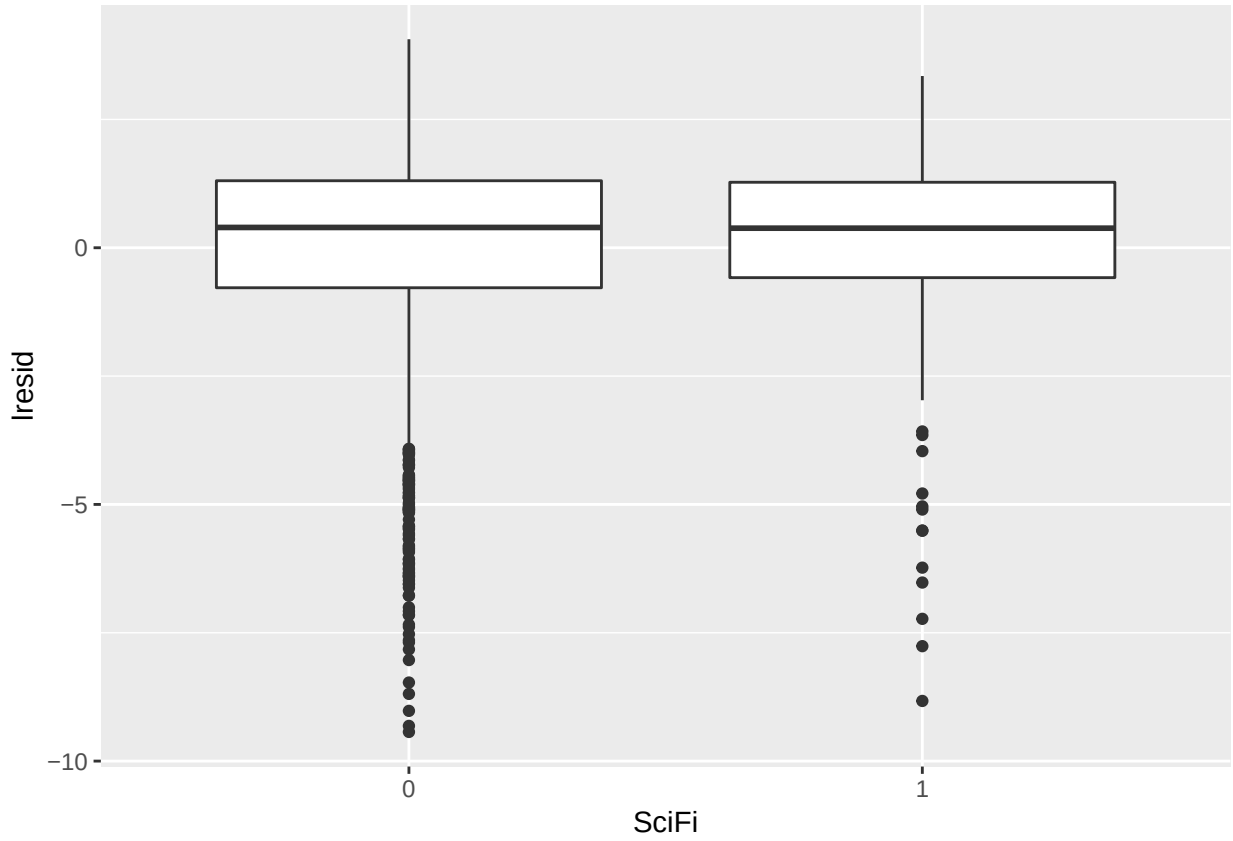

```
##
## [[7]]
```



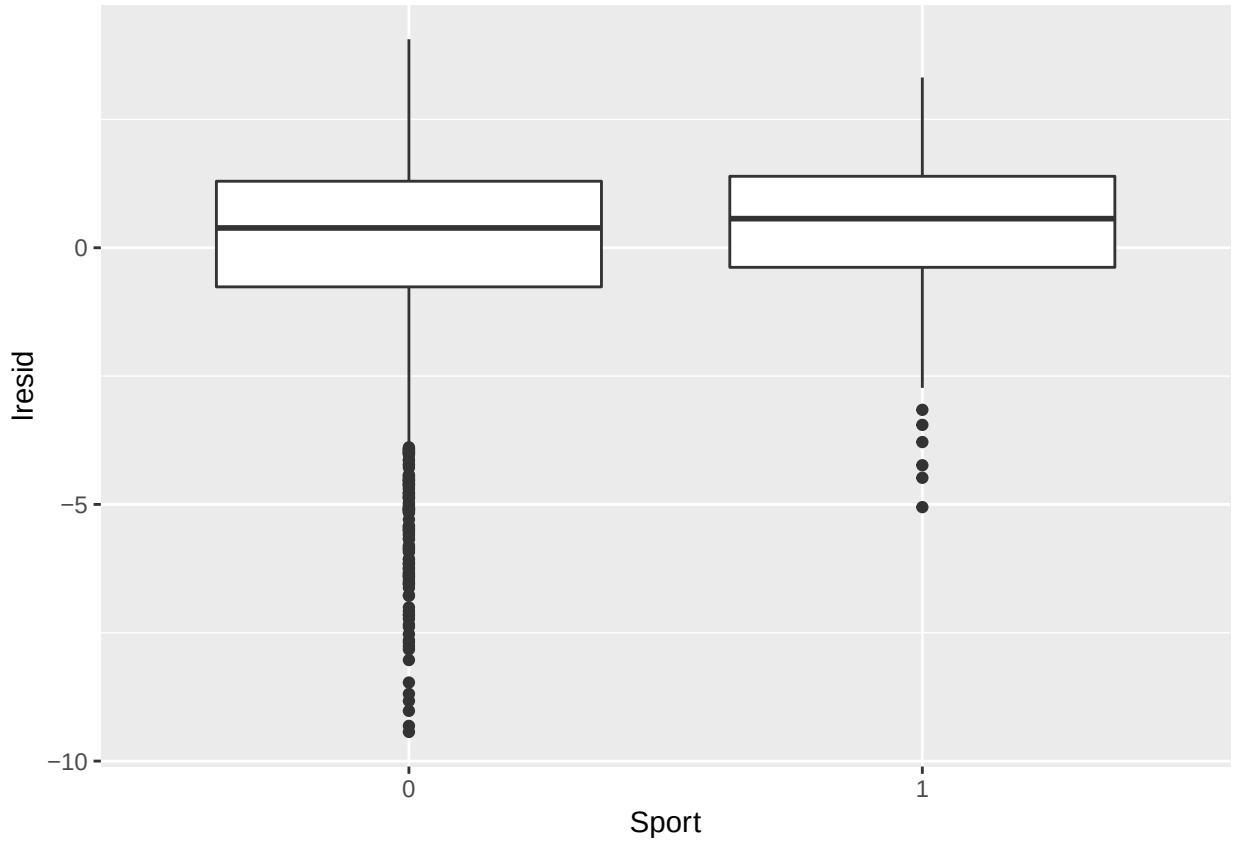

```
##
## [[8]]
```



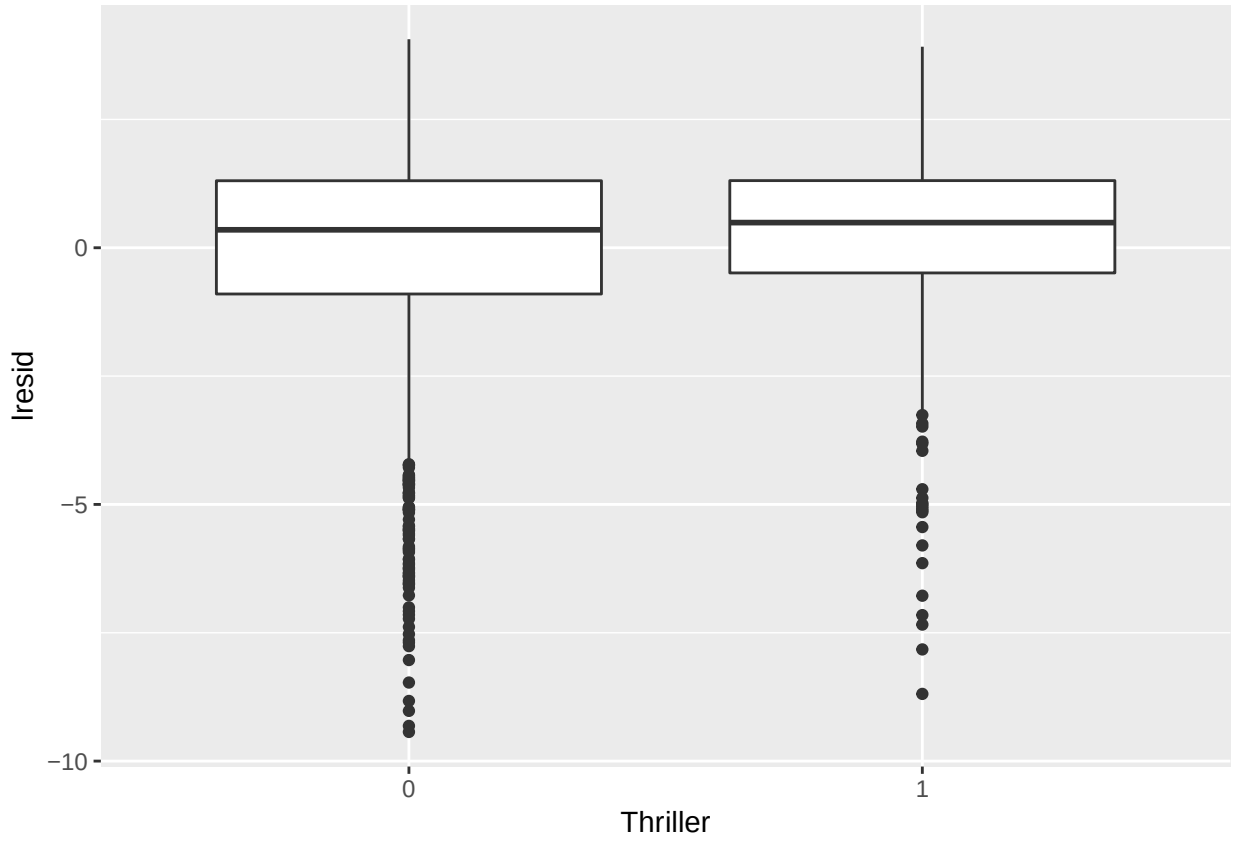

```
##
## [[9]]
```



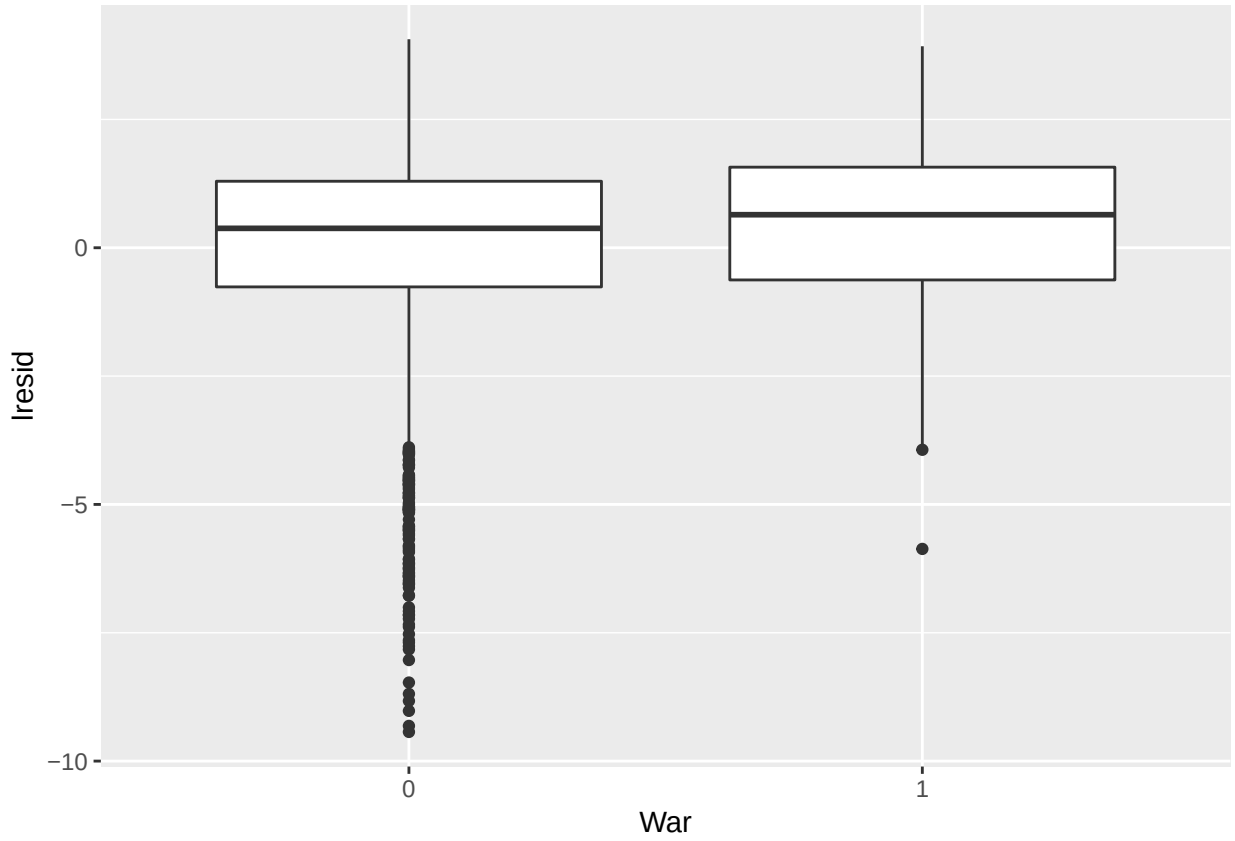

```
##
## [[10]]
```



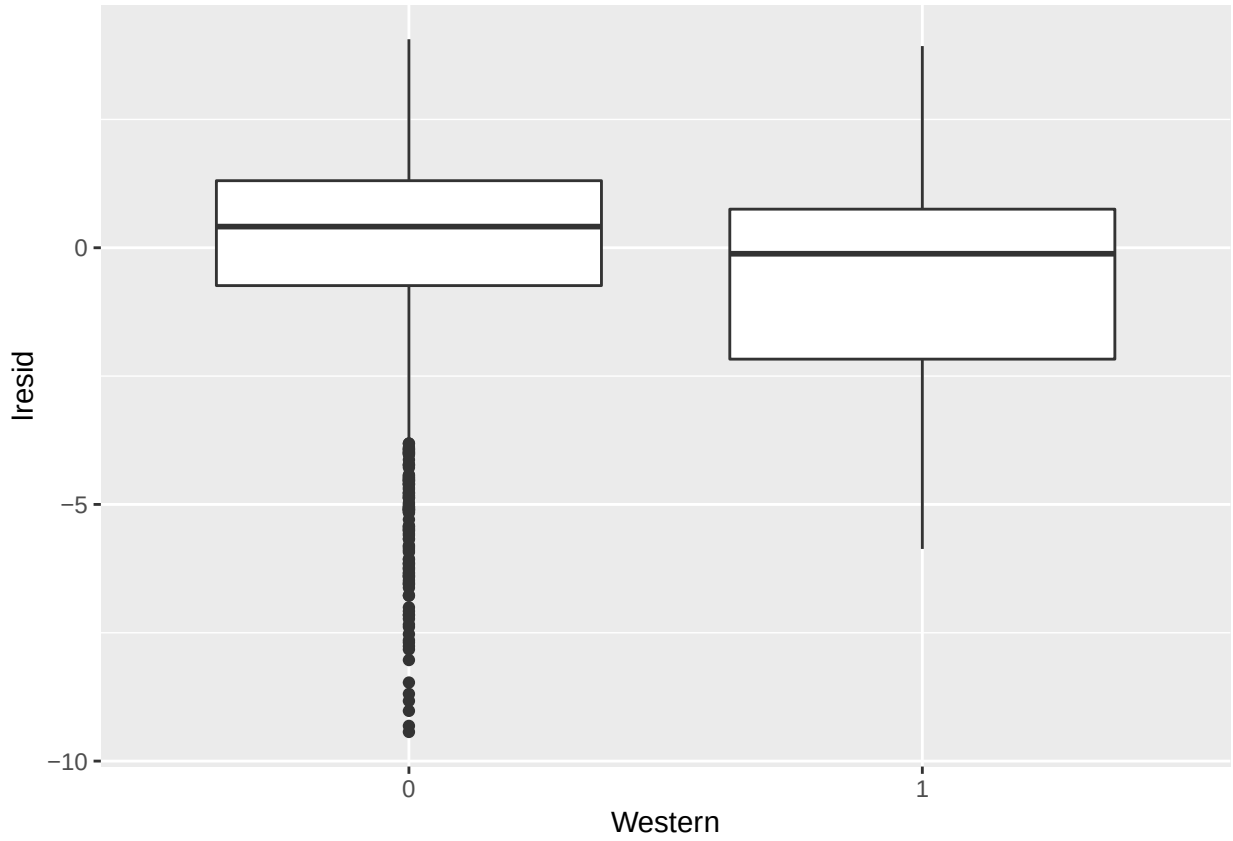

```
##
## [[11]]
```




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

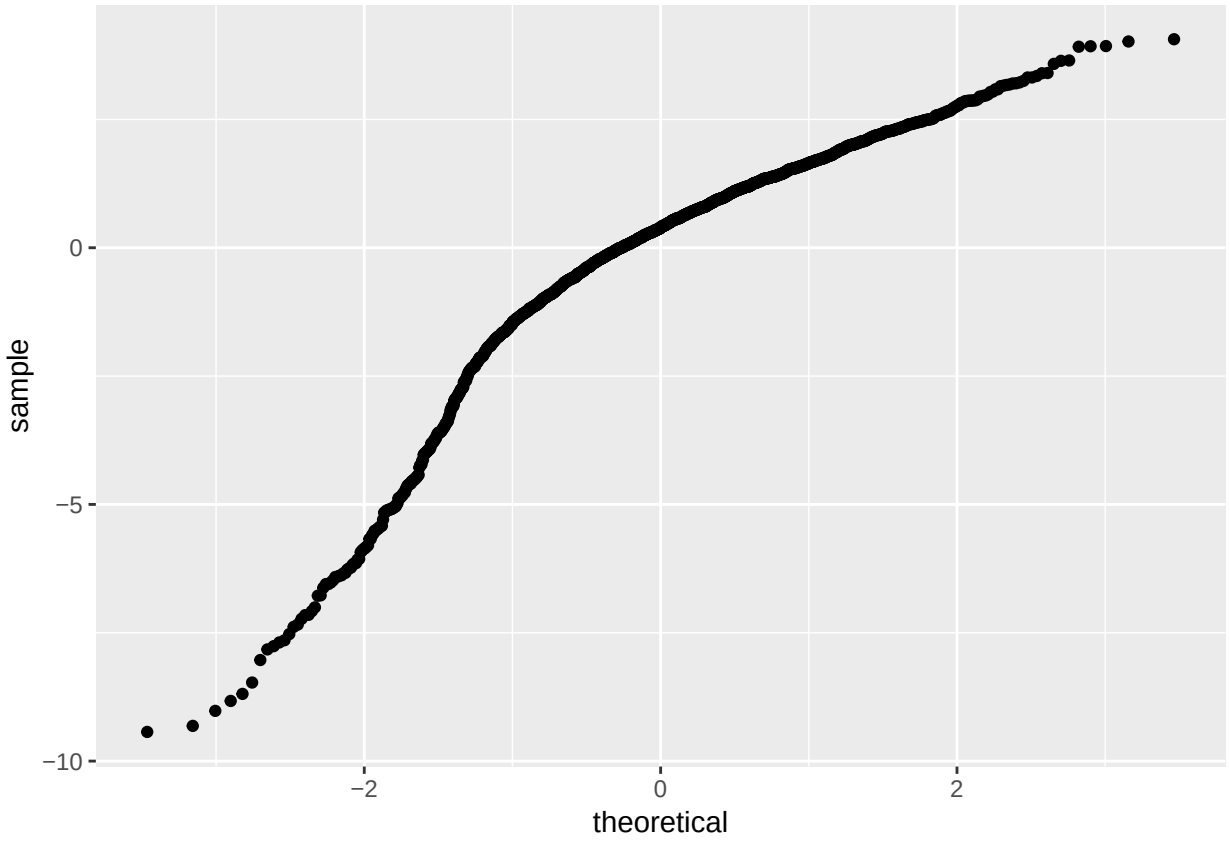



```
##
## [[13]]
```



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

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

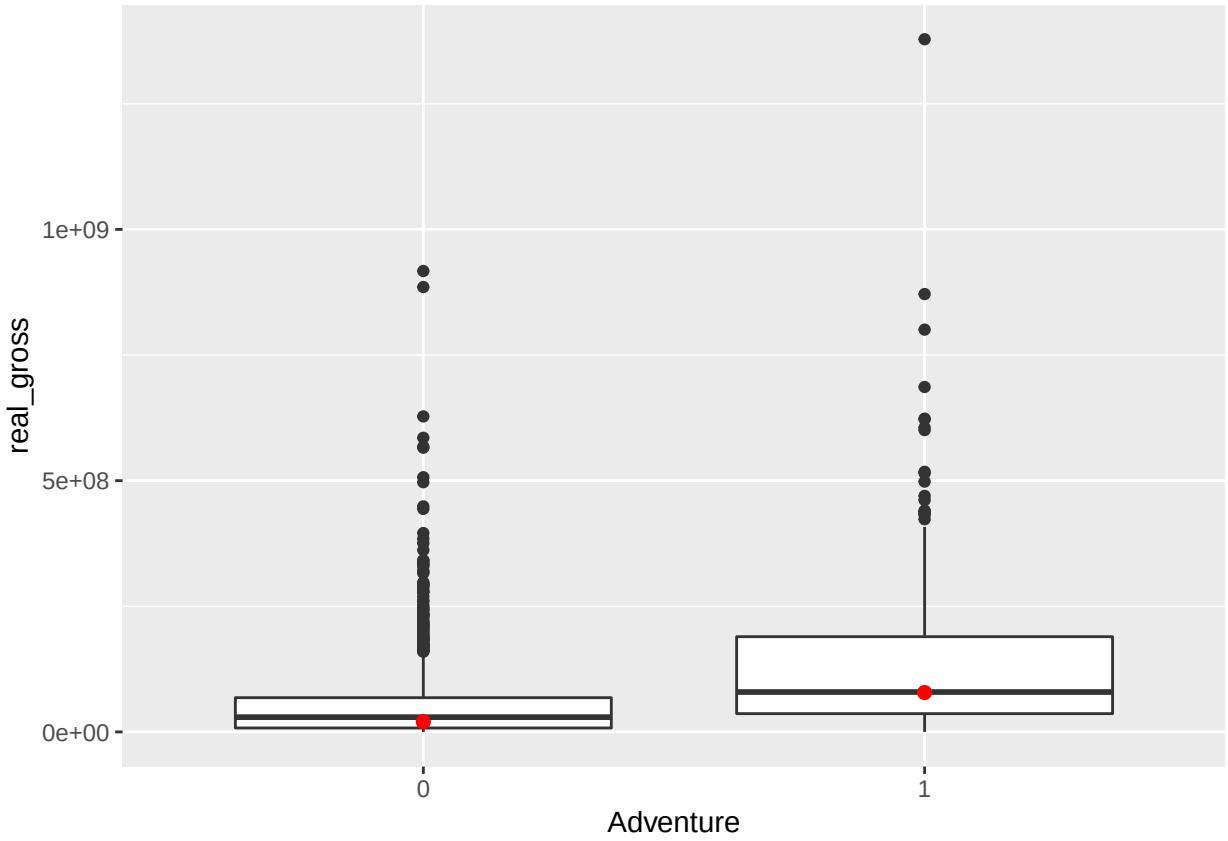


Plot Predictions

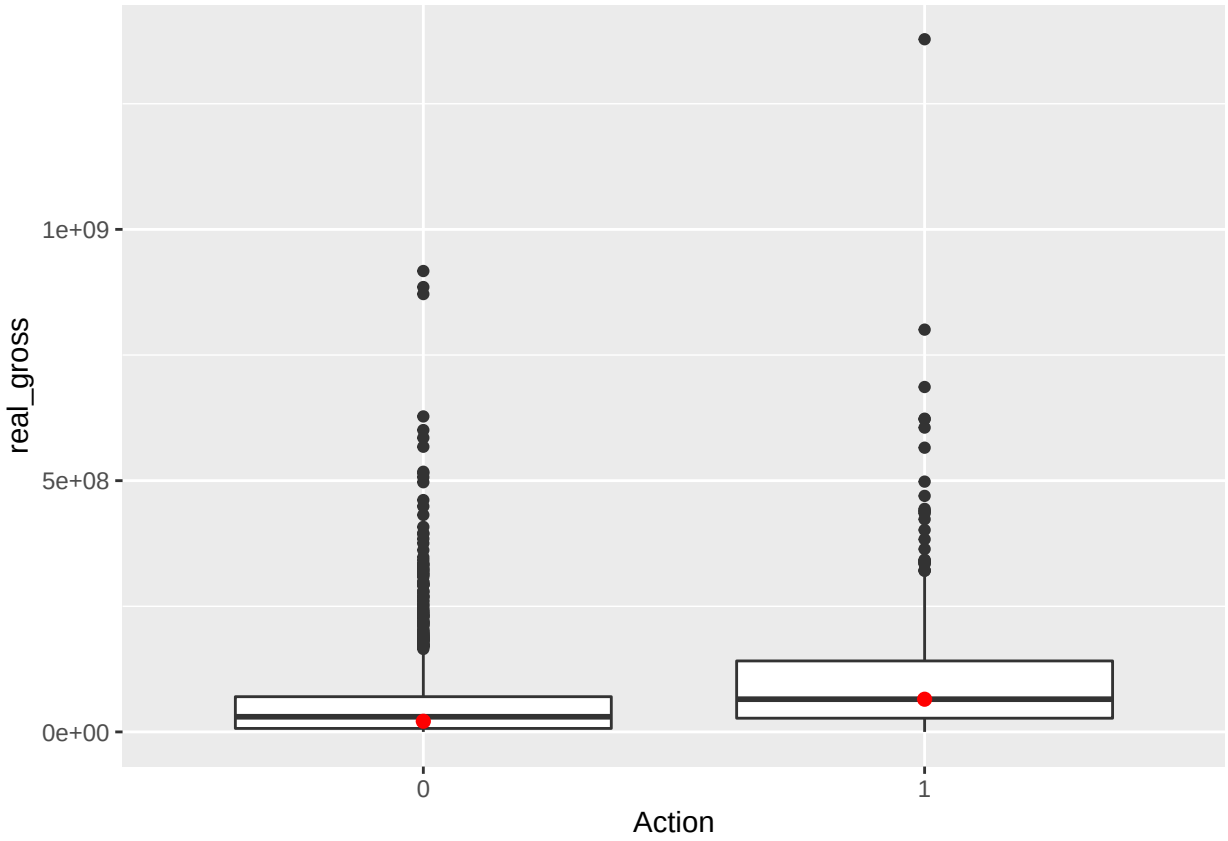
Plot prediction for mean real revenue against each genre included in the model. Looks pretty accurate!

```
train_pred <- train %>%
  add_predictions(mod_genre, 'lpred') %>%
  mutate(pred = exp(lpred))

lapply(genre_xvar, function(var) {
  train_pred %>%
    ggplot(aes_string(x = var)) +
    geom_boxplot(aes(y = real_gross)) +
    geom_point(data = train_pred %>% group_by(!!rlang::sym(var)) %>% summarize(mean = mean(pred)),
               aes(y = mean), color = 'red', size = 2)
})
```

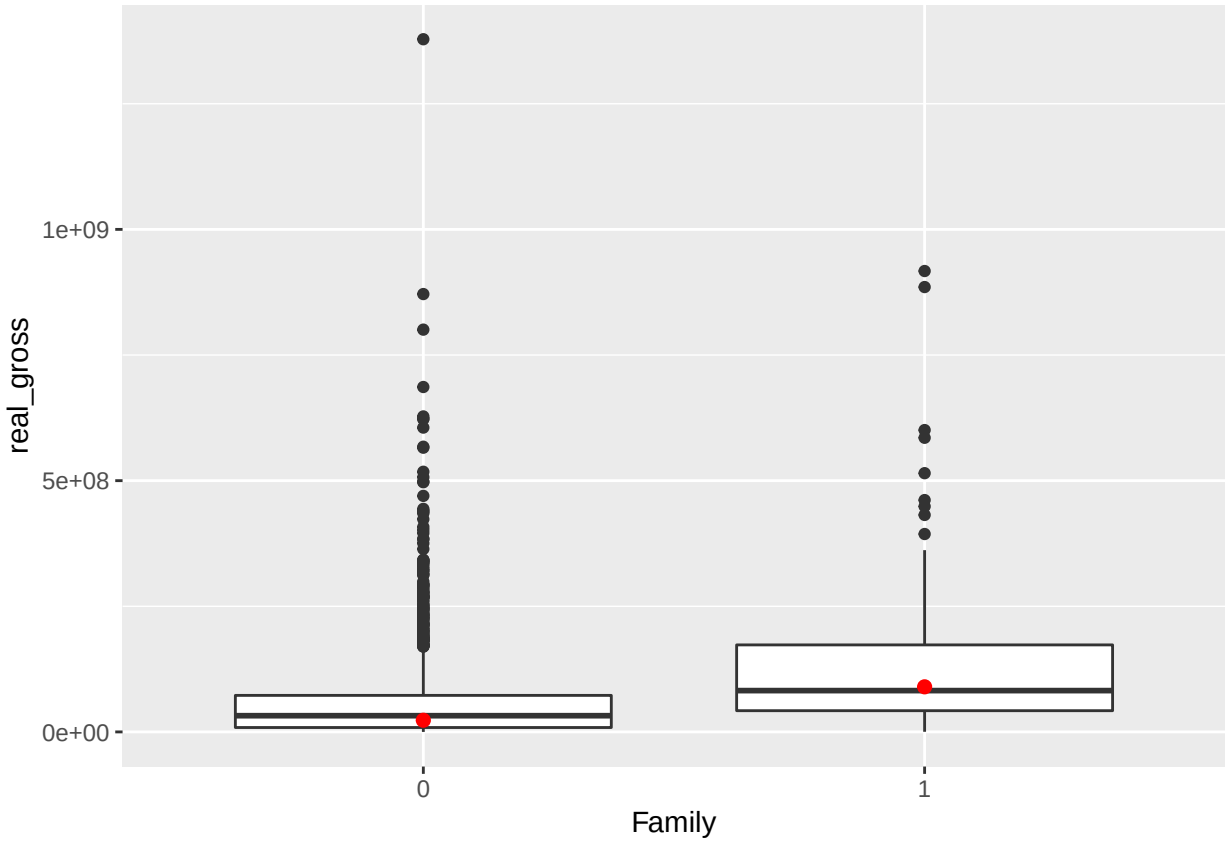
```
## [[1]]
```



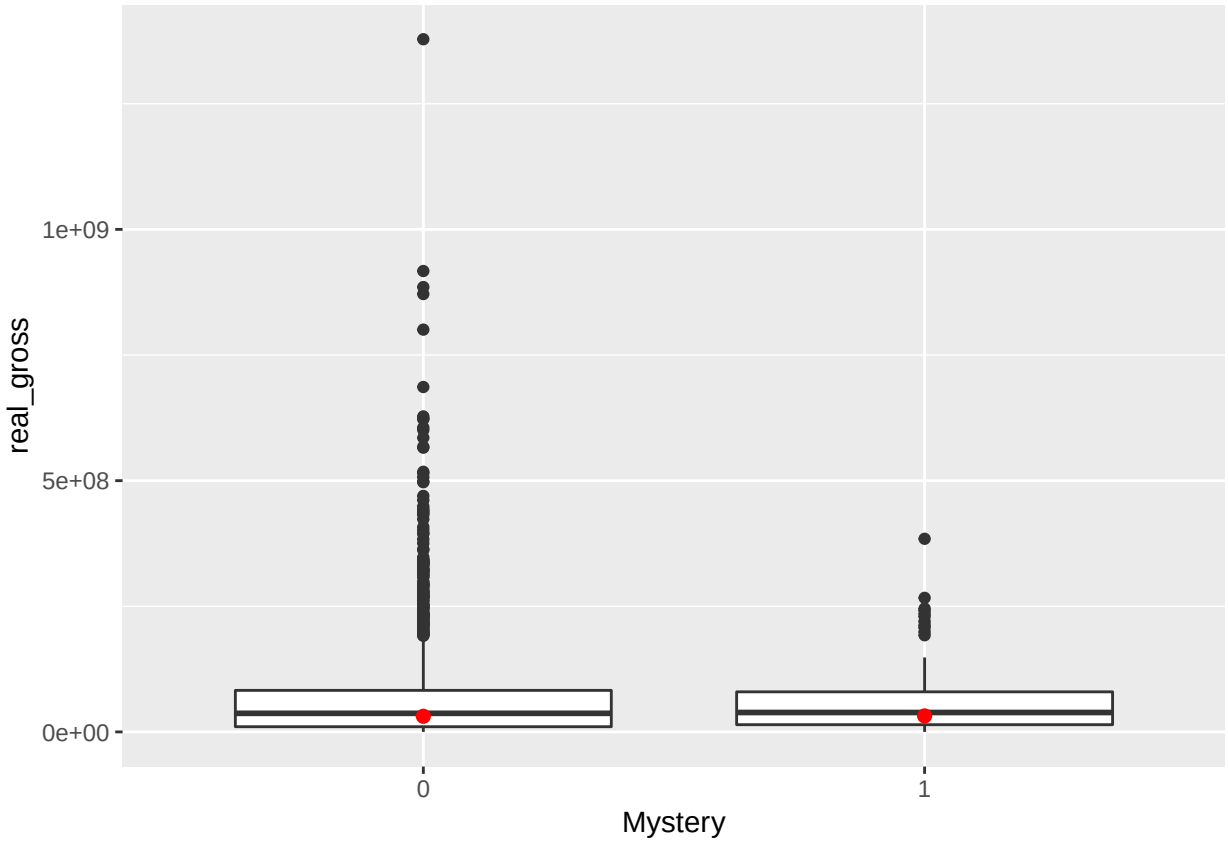

```
##
## [[2]]
```



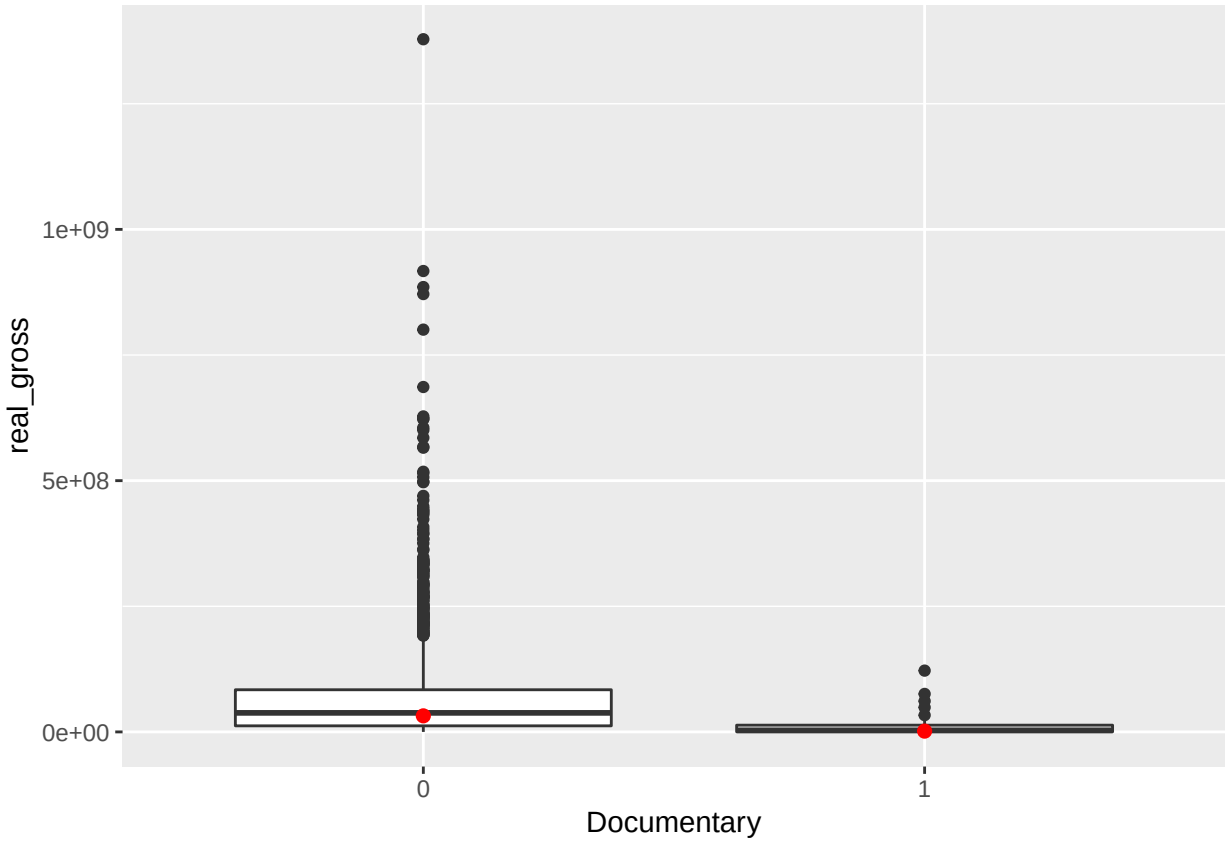

```
##
## [[3]]
```



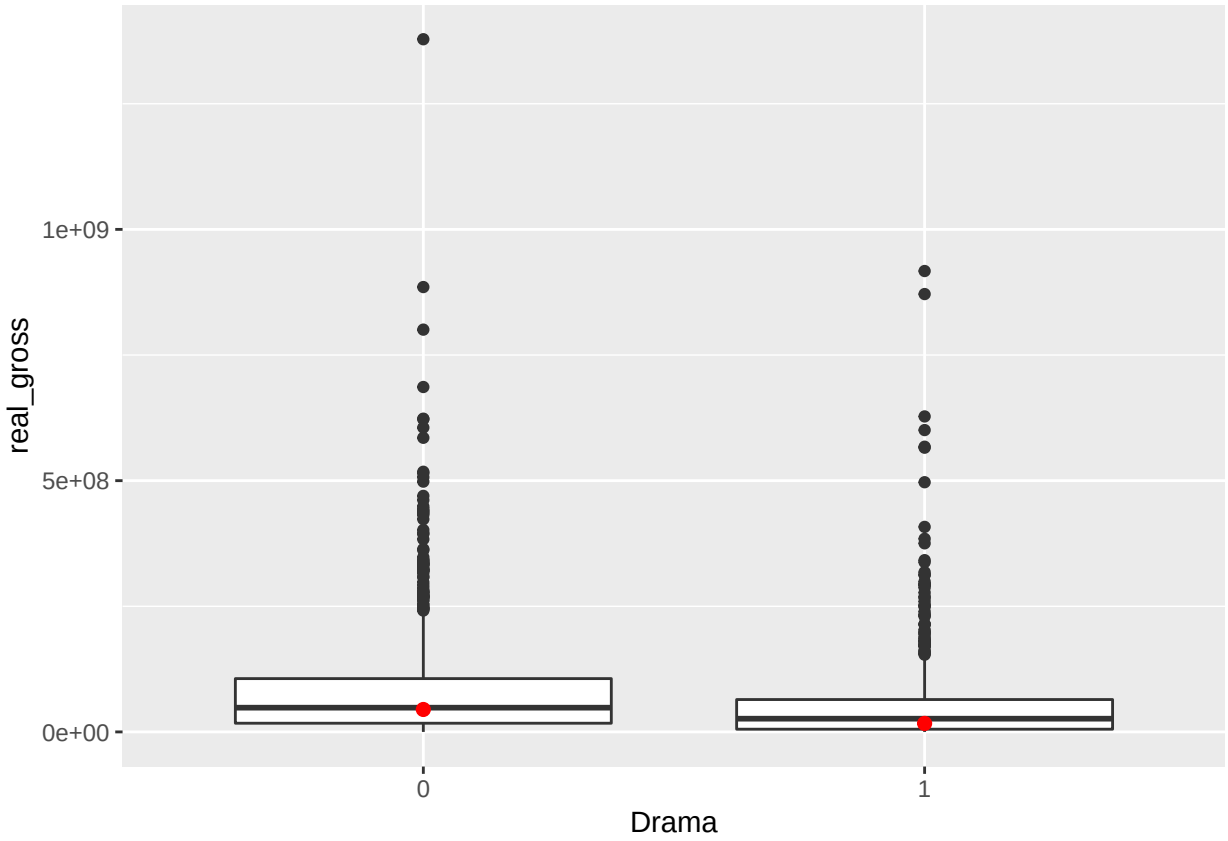

```
##
## [[4]]
```



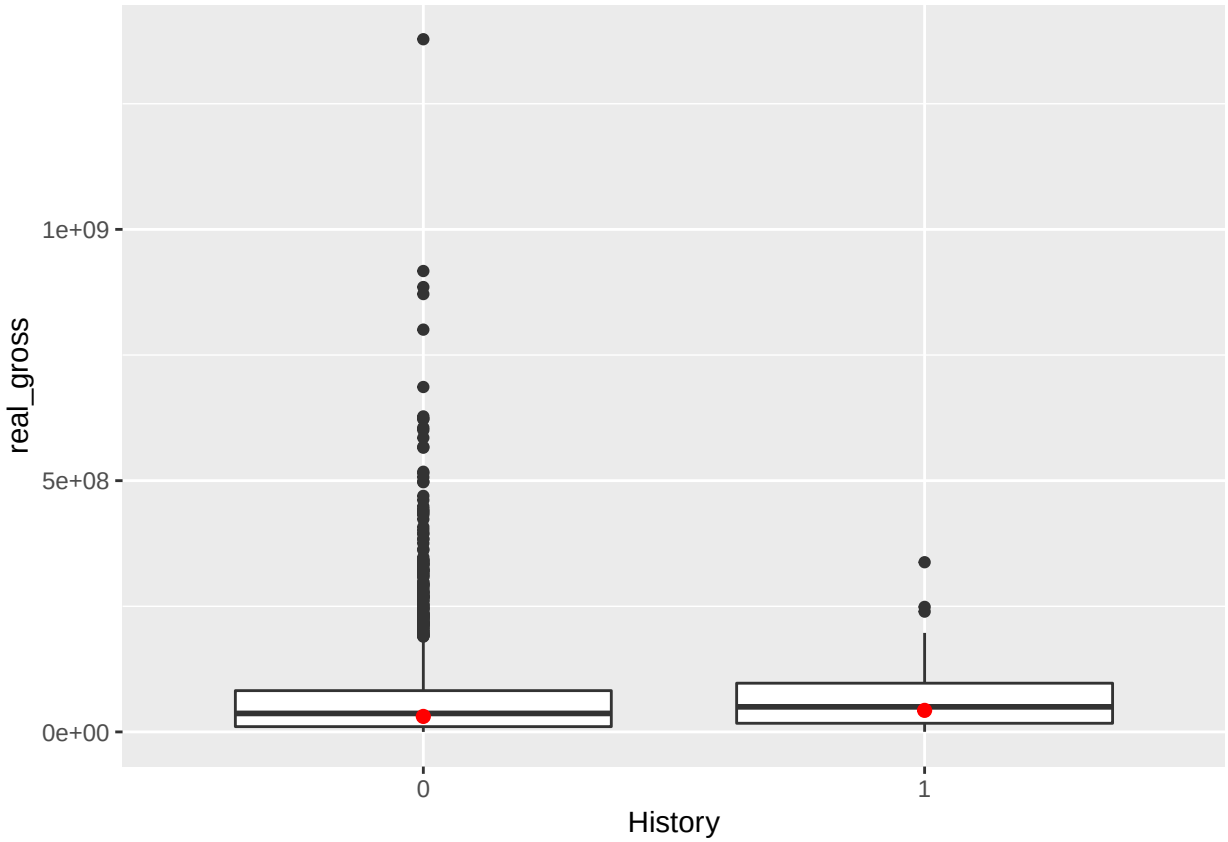

```
##
## [[5]]
```



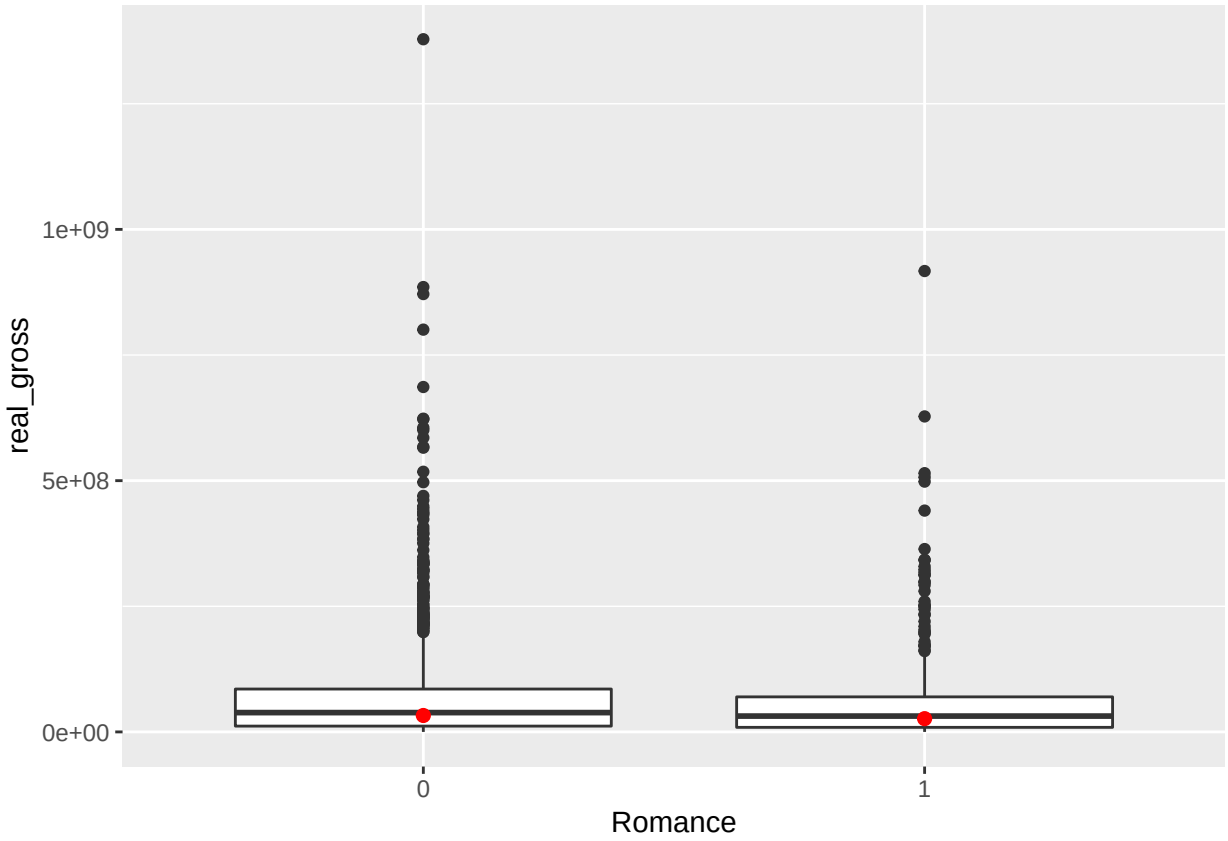

```
##
## [[6]]
```




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




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



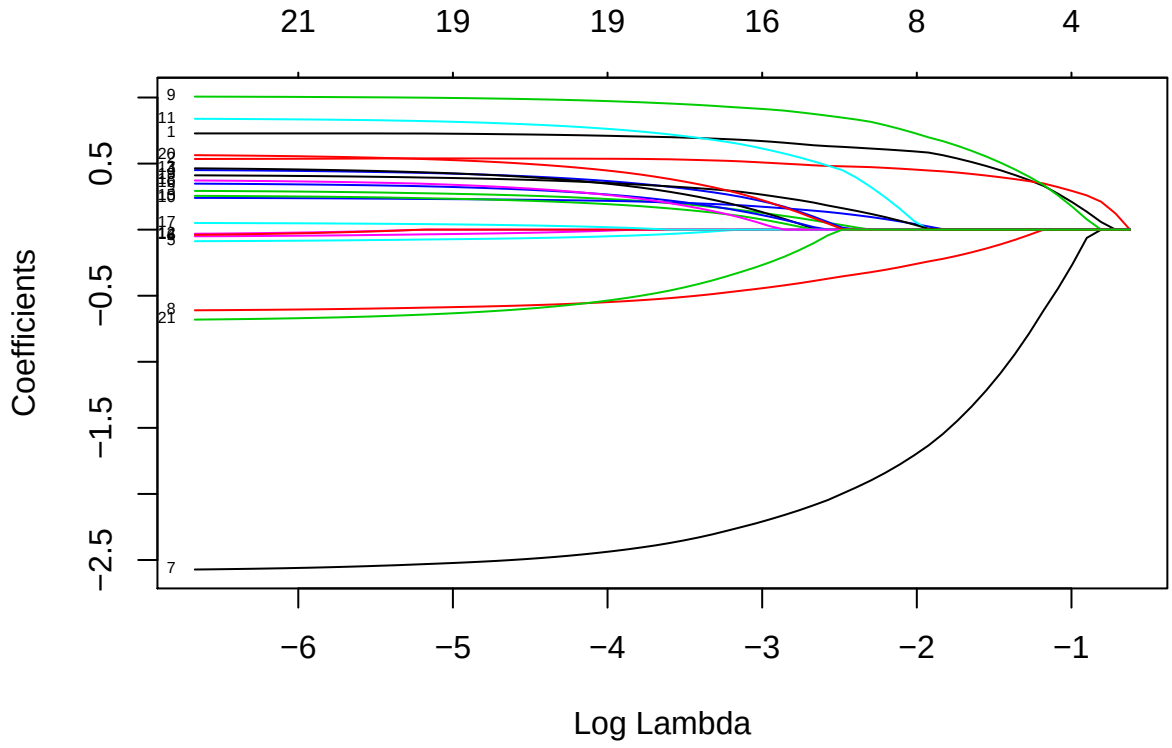
Glmnet: sparse

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.

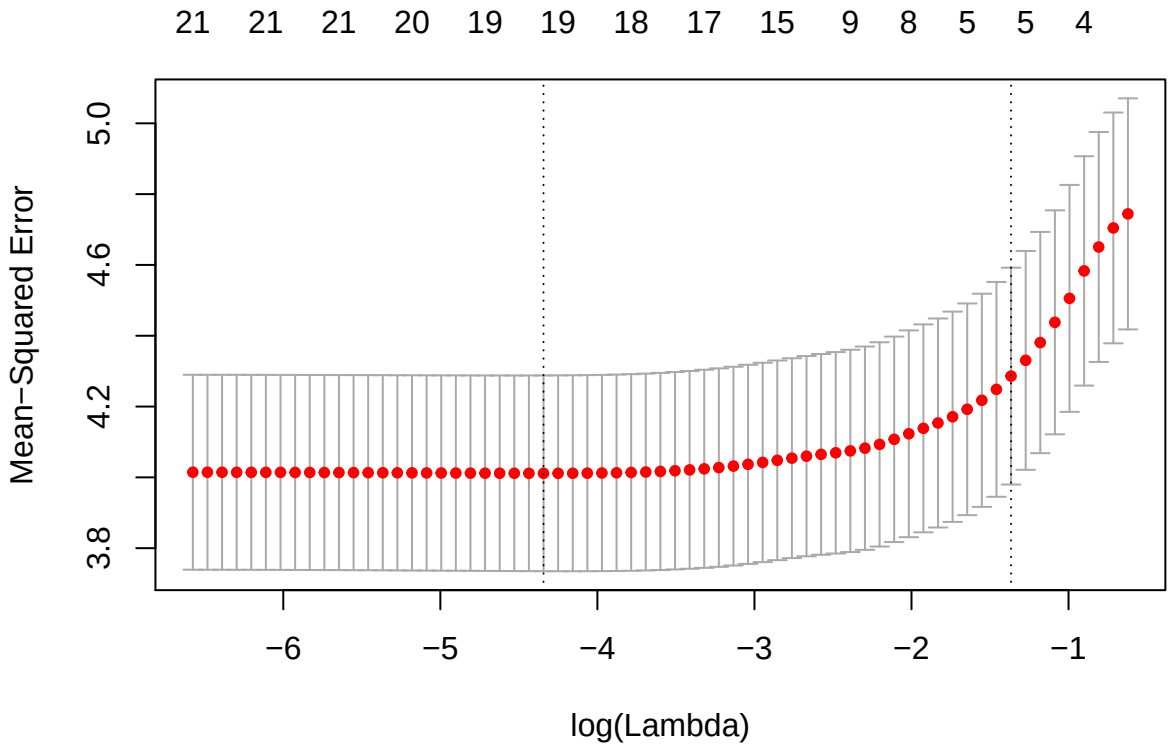
```
library(glmnet)
```

```
## 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
```

```
# matrix of x and y variables
x <- as.matrix(train_genre_only %>% mutate_all(funs(as.numeric(as.character(.)))))
y <- as.matrix(train$real_gross_log)

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




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




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

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 16.52926500
## Action       0.71752177
## Adventure    0.53762268
## Animation    0.25013799
## Biography    0.39362800
## Comedy      -0.06055871
## Crime       -0.01933750
## Documentary -2.47690355
## Drama       -0.56592977
## Family       0.98378232
## Fantasy      0.22385047
## History      0.78743189
## Horror       .
## Music        0.38385204
## Musical      .
## Mystery      0.21170857
## Romance      0.29206916
## SciFi        0.02834215
## Sport        0.29804928
## Thriller     0.37051979
## War          0.48368330
## Western     -0.58101894
```

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

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept) 16.70554759
## Action       0.41375453
## Adventure    0.38239659
## Animation    .
## Biography    .
## Comedy       .
## Crime        .
## Documentary -0.94219911
## Drama       -0.07720242
## Family       0.45609726
## Fantasy      .
## History      .
## Horror       .
## Music        .
## Musical      .
## Mystery      .
## Romance      .
## SciFi        .
## Sport        .
## Thriller     .
## War          .
## Western      .
```

Fit model with other variables

Plot residuals of other variables based on the genre model.

All of these plots indicate a relationship that is not fully represented in the model yet and thus all are valid candidates for including in the model (also given their relatively linear relationships)

For example, movies with lower budgets make less revenue than predicted by the genres in the model (negative residual) and movies with higher budgets make more revenue than predicted by genre (positive residual). Cast facebook likes, director facebook likes, and IMDB score follow a similar pattern.
Year is opposite where movies in more recent years have less revenue than predicted by genre

Content rating has more of a random relationship with the residual. Perhaps this is because genres and content ratings have some correlation (i.e. Family movies tend to be G or PG while Horror tend to be R) and thus this relationship may have already been captured.
There is some indication that R movies may make less revenue than predicted and PG-13 movies make more revenue than predicted.

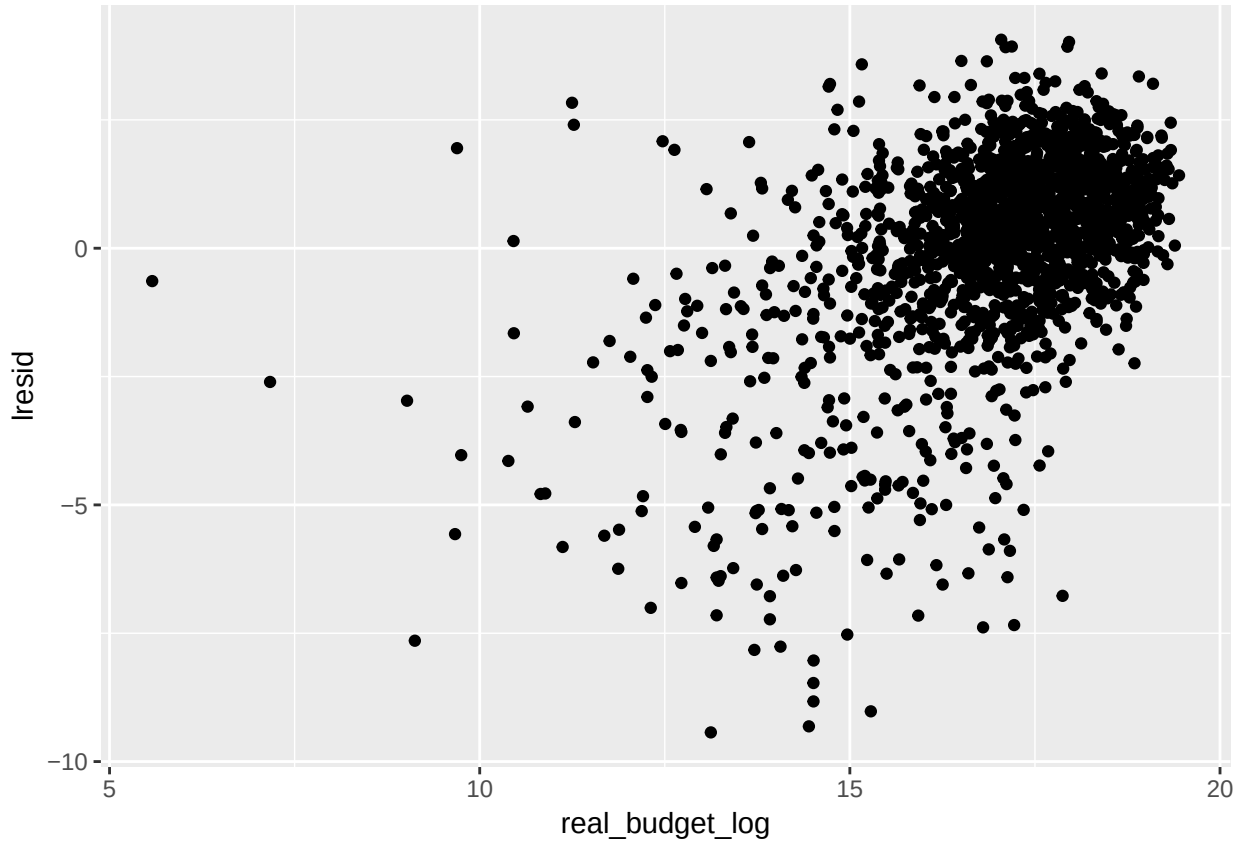
```
# get log versions of variables since residuals are log: same scale
train_resid <- train_resid %>%
  mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score', 'year'), funs(log = log(.)))

# graph each against log residual: continuous
lapply(c('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score', 'year'), function(var) {
  print(var)
  train_resid %>%
```

```
    ggplot() +
    geom_point(aes_string(str_c(var, '_log'), y = 'lresid'))
})
```

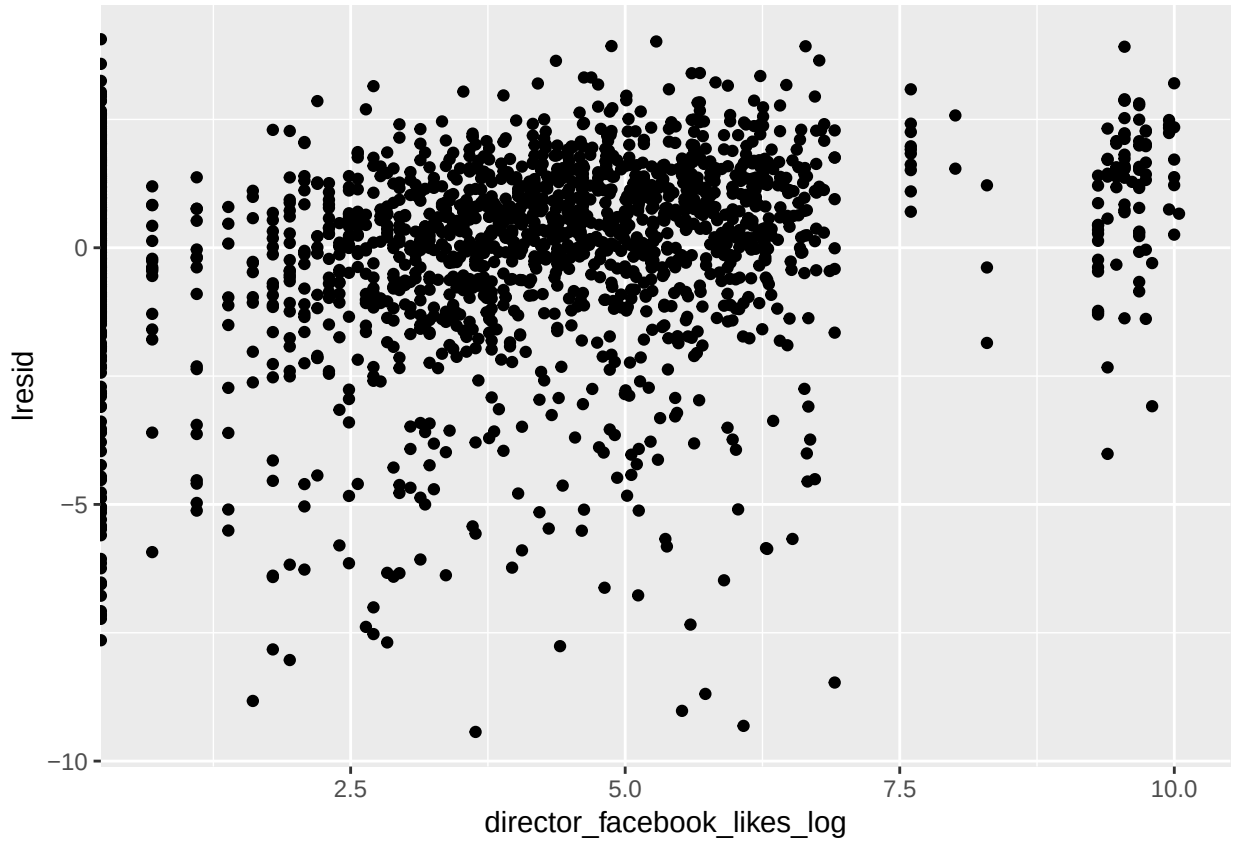
```
## [1] "real_budget"
## [1] "director_facebook_likes"
## [1] "cast_total_facebook_likes"
## [1] "imdb_score"
## [1] "year"
```

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

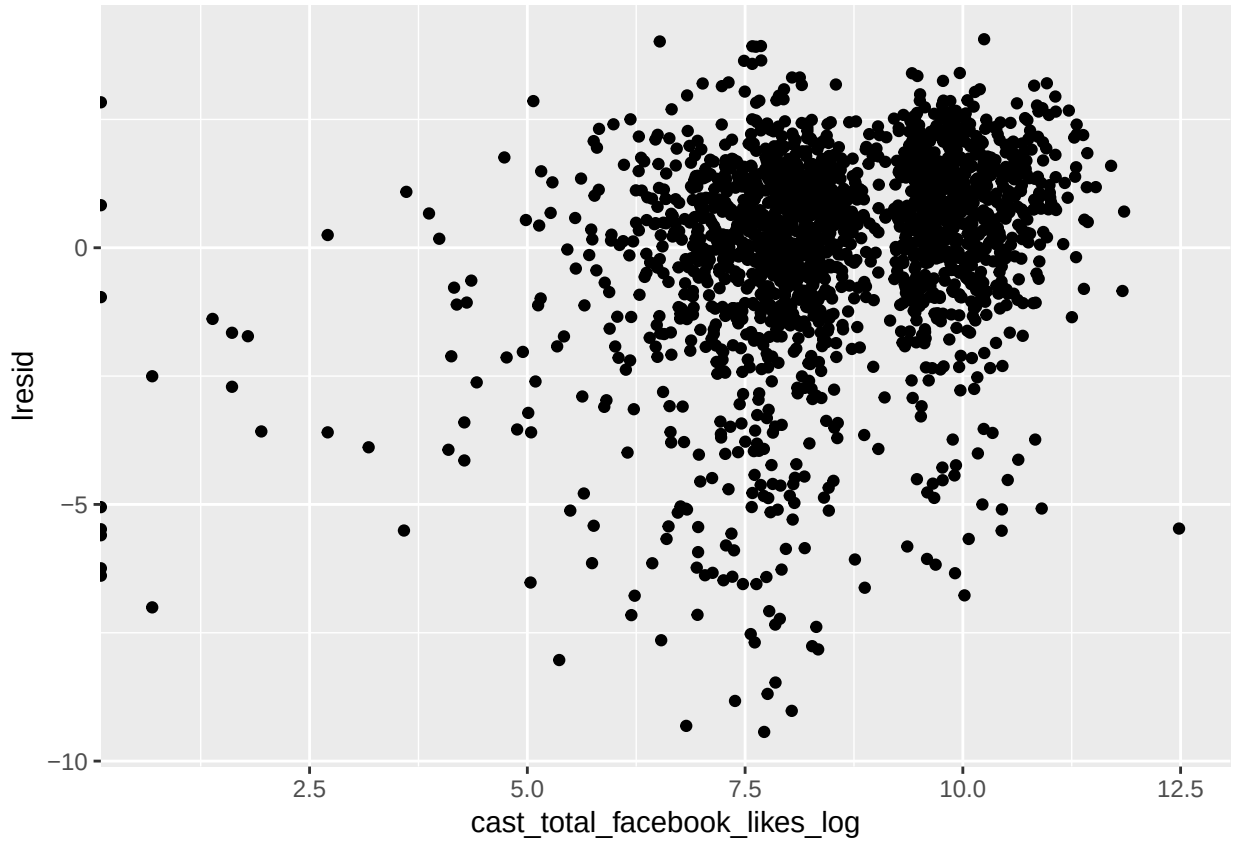
```
## Warning: Removed 102 rows containing missing values (geom_point).
```



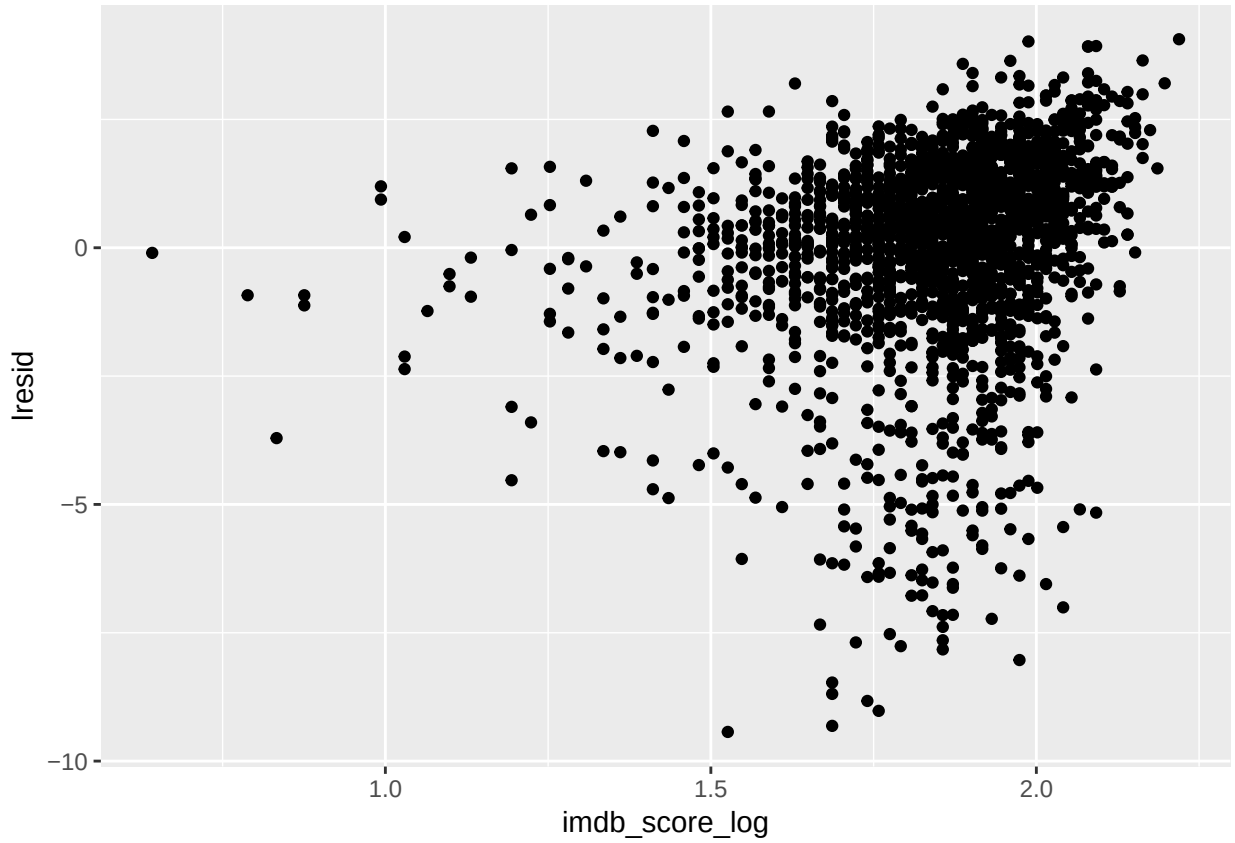

```
##
## [[2]]
```



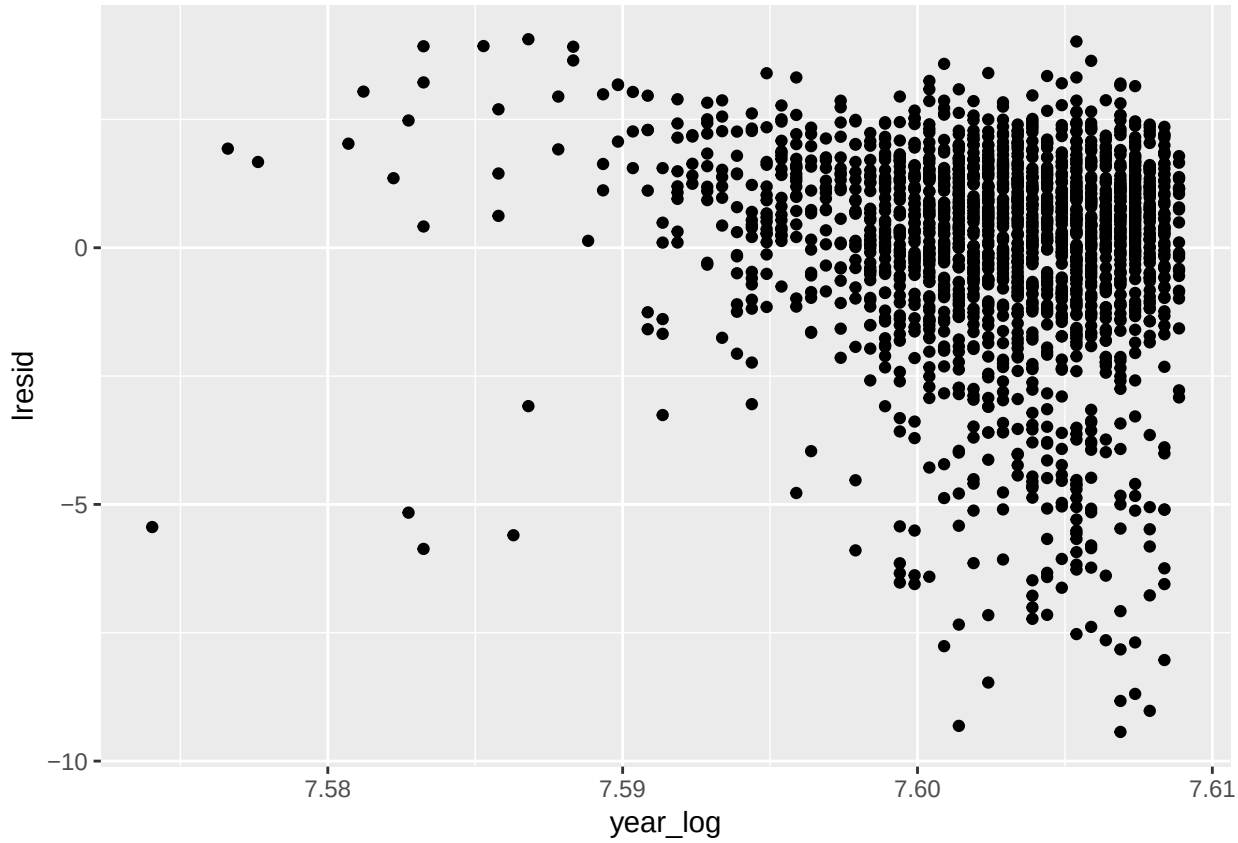

```
##
## [[3]]
```



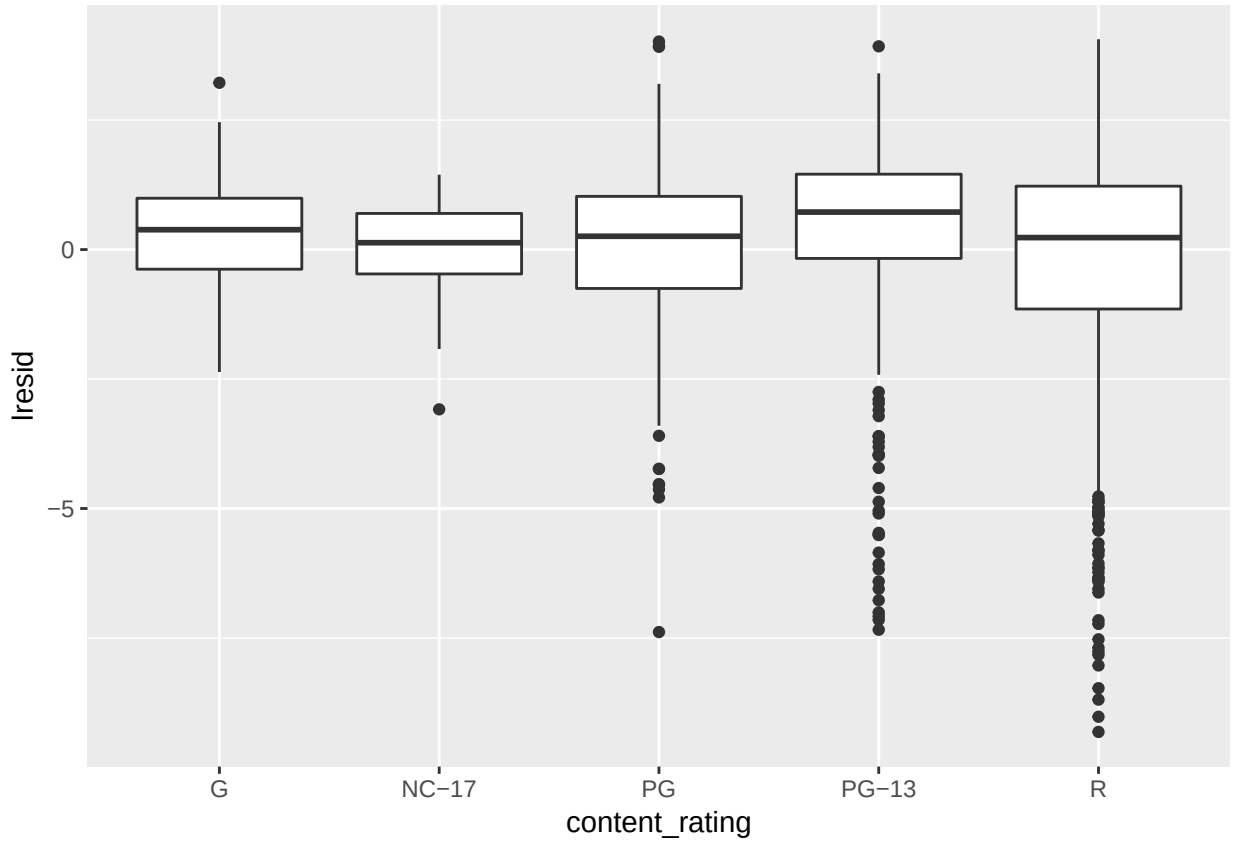

```
##
## [[4]]
```




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




```
# content rating: categorical
# note can't log categorical variables
train_resid %>%
  filter(!is.na(content_rating)) %>%
  ggplot() +
  geom_boxplot(aes(content_rating, lresid))
```



Try stepwise selection with these other variables given that none had fully random relationships with the residual from the genre model. Use the fitted genre model as a base.

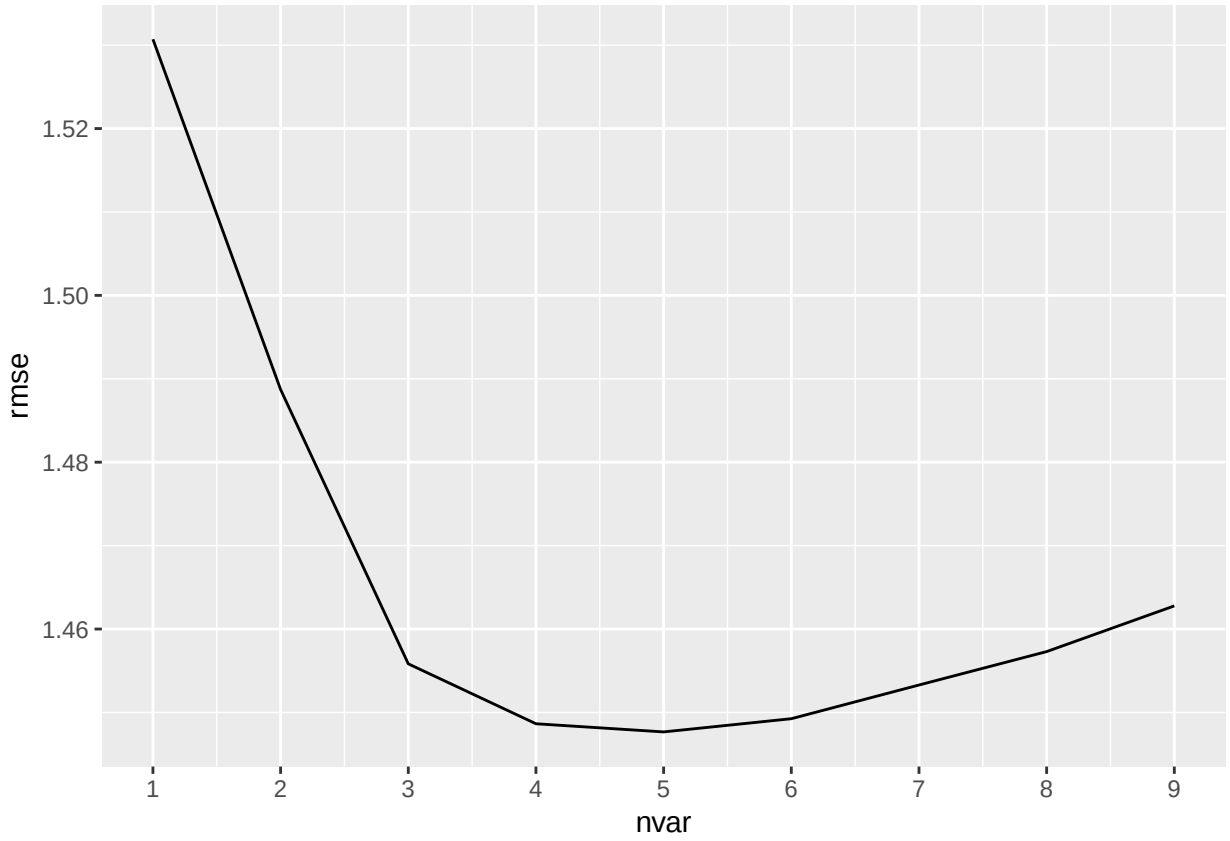
For factor variables (content_rating, total_oscars) use normal versions of variables.
For facebook likes, use log versions as those were more linear with log(real_gross).
For budget and IMDB score, I think log versions are better, but try the non-log versions too. Both had some linearity.
For year, use normal version.

```
# create log versions of continuous variables
# also turn -Inf from log(0) to NA
train <- train %>%
  mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score', 'year'), funs(log = log(.))) %>%
  mutate_at(vars(contains('log')), funs(ifelse(is.infinite(.), NA, .)))
valid <- valid %>%
  mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score', 'year'), funs(log = log(.))) %>%
  mutate_at(vars(contains('log')), funs(ifelse(is.infinite(.), NA, .)))

# starting formula: genre
starting_formula = 'Adventure + Action + Family + Mystery + Documentary + Drama + History + Romance'

# stepwise starting with genre
rmse_lst <- step_wise_loop(df = train %>% select(genre_xvar, content_rating, real_budget, year,
                                                 total_oscars_actor, total_oscars_director,
                                                 imdb_score_log, real_budget_log,
```

```
                                                director_facebook_likes_log, cast_total_facebook_likes
                           starting_vars = genre_xvar,
                           starting_formula = starting_formula)
```

```
## real_budget_log
##        1.530706
## [1] 1
## imdb_score_log
##       1.488713
## [1] 2
##     year
## 1.455831
## [1] 3
## content_rating
##       1.448646
## [1] 4
## total_oscars_director
##              1.447667
## [1] 5
## real_budget
##    1.449249
## [1] 6
## cast_total_facebook_likes_log
##                       1.45329
## [1] 7
## total_oscars_actor
##           1.457298
## [1] 8
## director_facebook_likes_log
##                     1.462775
```

```
# graph RMSE vs number of variables
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 4, decreases too small or increase

# model with extra 4 variables
mod_all <- lm(real_gross_log ~ Adventure + Action + Family + Mystery +
                  Documentary + Drama + History + Romance +
                  real_budget_log + imdb_score_log + year + content_rating,
          data = train)

summary(mod_all)
```

```
##
## Call:
## lm(formula = real_gross_log ~ Adventure + Action + Family + Mystery +
##     Documentary + Drama + History + Romance + real_budget_log +
##     imdb_score_log + year + content_rating, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5049 -0.5353  0.1766  0.7935  7.2505
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         65.056883   8.353061   7.788 1.16e-14 ***
## Adventure1          -0.289429   0.103546  -2.795 0.005245 **
## Action1             -0.003027   0.093066  -0.033 0.974058
## Family1              0.612841   0.175199   3.498 0.000481 ***
```

```
## Mystery1             0.056771   0.118605   0.479 0.632242
## Documentary1         0.244447   0.319790   0.764 0.444734
## Drama1              -0.675839   0.078066  -8.657  < 2e-16 ***
## History1            -0.060521   0.201827  -0.300 0.764318
## Romance1            -0.026950   0.084713  -0.318 0.750420
## real_budget_log      0.784444   0.028690  27.342  < 2e-16 ***
## imdb_score_log       2.505047   0.202432  12.375  < 2e-16 ***
## year                -0.032857   0.004171  -7.878 5.84e-15 ***
## content_ratingNC-17  0.483070   0.559116   0.864 0.387714
## content_ratingPG     0.146977   0.263565   0.558 0.577156
## content_ratingPG-13  0.461353   0.300150   1.537 0.124458
## content_ratingR     -0.003942   0.299416  -0.013 0.989498
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.414 on 1727 degrees of freedom
##   (141 observations deleted due to missingness)
## Multiple R-squared:  0.4763, Adjusted R-squared:  0.4717
## F-statistic: 104.7 on 15 and 1727 DF,  p-value: < 2.2e-16
```

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

```
## [1] 1.448646
```

TO DO

- Graph residuals of included and excluded variables against this model
- Some of the genre variables are now insigificant -> try full step wise from scratch. Some of these shouldn't be included?