Modeling_Katrina

Katrina Truebebach

March 18, 2019

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

Load cleaned data

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

```
# need to drop years before 1980: too sparse
  # most of those years have 1 or 0 observations. If including year in the model, we aren't getting any
# only necessary when including year in model, but hard to compare different models then b/c data diffe
train <- train %>% filter(as.integer(as.character(year)) >= 1980)
valid <- valid %>% filter(as.integer(as.character(year)) >= 1980)

# calculate logs
train <- train %>% mutate(real_gross_log = log10(real_gross))
valid <- valid %>% mutate(real_gross_log = log10(real_gross))
train <- train %>%
  mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score'), funs(log = log10(.)))
valid <- valid %>%
  mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score'), funs(log = log10(.)))
```

Write Functions to Automate

Write function to automate stepwise
Note: not using the step() function because can't fit and find RMSE on different datasets (train, valid)

```
# 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)
}
```

Function to graph the residuals from a model against all potential variables (included and excluded)

```
gr_resid <- function(mod) {
  # graph residuals
  # get log versions of variables since residuals are log: same scale
  df_resid <- train %>%
    add_residuals(mod, 'lresid') %>%
    mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
           'imdb_score'), funs(log = log10(.)))

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

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

  # categorical
  # can't log categorical variables
  lapply(c('content_rating', 'year', 'total_oscars_actor', 'total_oscars_director', all_genre_vars), fu
    print(df_resid %>%
      filter(!is.na(!!rlang::sym(var))) %>%
      ggplot() +
      geom_jitter(aes_string(var, 'lresid'), alpha = .3))
  })

  # qq plot of residuals
  df_resid %>% ggplot() +
    geom_qq(aes(sample = lresid))
}
```

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() # 52
train %>% filter(History == 1, Drama == 1) %>% count() # 51
```

Model Fit

Which genres should we be using?

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

train_genre_only <- train %>% select(all_genre_vars)

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

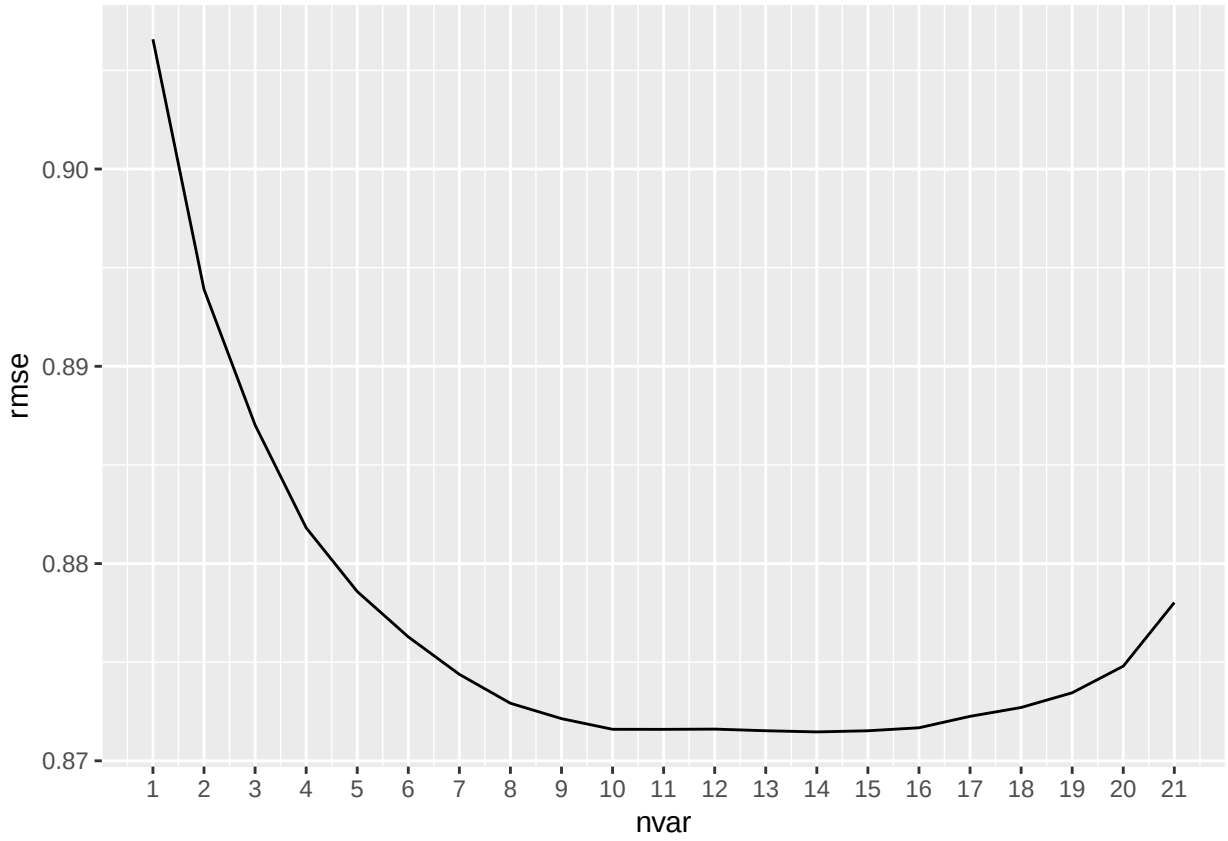
```
## Adventure
## 0.9065803
## [1] 1
##    Action
## 0.8939041
## [1] 2
##    Family
## 0.8870258
## [1] 3
##   Mystery
## 0.8818136
## [1] 4
##   Romance
## 0.8785826
## [1] 5
##     Drama
## 0.8762839
## [1] 6
##   History
## 0.8743844
## [1] 7
## Documentary
##   0.8729182
## [1] 8
##   Musical
## 0.8721386
## [1] 9
##       War
## 0.8715984
## [1] 10
##     SciFi
## 0.8715957
## [1] 11
##     Crime
## 0.8716081
## [1] 12
##   Fantasy
## 0.8715244
## [1] 13
##     Sport
## 0.8714654
## [1] 14
##     Music
```

```
## 0.8715242
## [1] 15
## Biography
## 0.8716754
## [1] 16
##    Comedy
## 0.8722553
## [1] 17
##    Horror
## 0.8727044
## [1] 18
## Animation
## 0.8734501
## [1] 19
##  Thriller
## 0.8747985
## [1] 20
##   Western
## 0.8780216
```

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 (debatably 10?)

# 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 + Romance + Drama + History + Do
          data = train)

summary(mod_genre)
```

```
##
## Call:
## lm(formula = real_gross_log ~ Adventure + Action + Family + Mystery +
##     Romance + Drama + History + Documentary, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0859 -0.3211  0.1680  0.5636  1.7697
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.21040    0.04041 178.418  < 2e-16 ***
## Adventure1    0.25572    0.05959   4.291 1.87e-05 ***
## Action1       0.37493    0.05337   7.025 2.99e-12 ***
## Family1       0.46184    0.06733   6.859 9.43e-12 ***
## Mystery1      0.19739    0.07021   2.811 0.004985 **
```

```
## Romance1      0.10130    0.04886   2.073 0.038297 *
## Drama1       -0.23074    0.04398  -5.247 1.73e-07 ***
## History1      0.45334    0.12298   3.686 0.000234 ***
## Documentary1 -1.10142    0.13044  -8.444  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8582 on 1842 degrees of freedom
## Multiple R-squared:  0.1648, Adjusted R-squared:  0.1612
## F-statistic: 45.44 on 8 and 1842 DF,  p-value: < 2.2e-16
```

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

```
## [1] 0.8729182
```

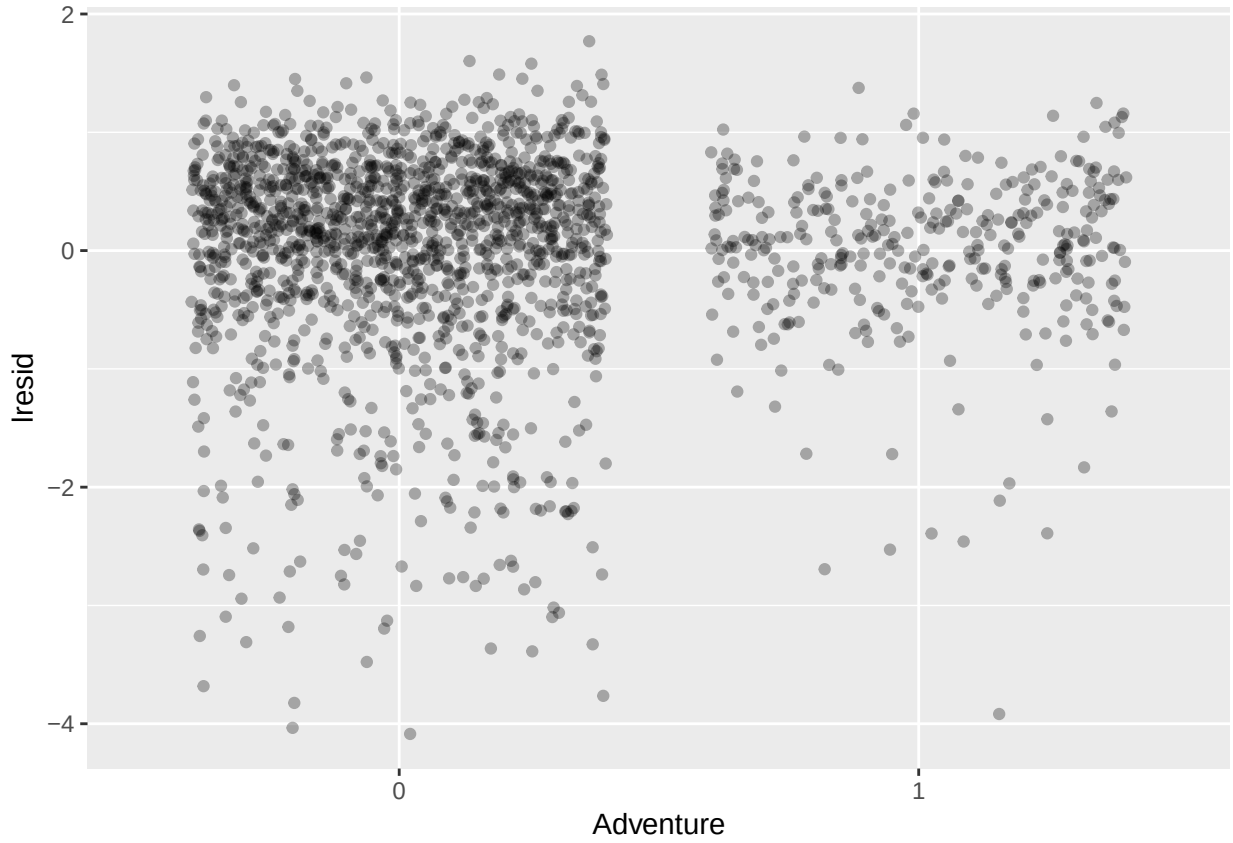
```
# list of these variables for future use
genre_xvar <- c('Adventure', 'Action', 'Family', 'Mystery',
                'Documentary', 'Drama', 'History', 'Romance')
```

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

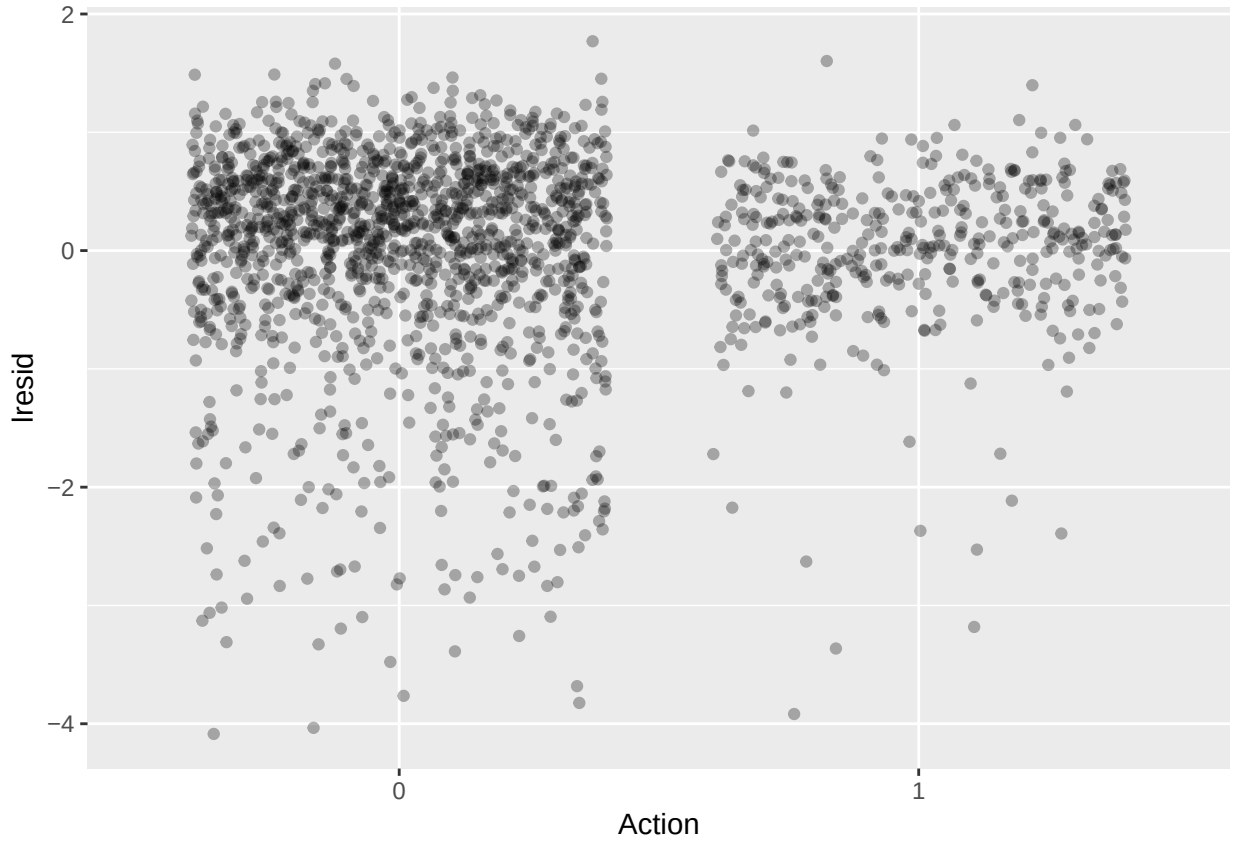
I like this geom jitter view better. Can see individual points. Most movies have some outliers where actual makes less money than predicted based on the included genres. But tricky because movies are multiple genres. So could be because that movie is also another genre that makes less money. Bulk of observations around zero.

```
# 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_jitter(aes_string(var, y = 'lresid'), alpha = .3)
})
```

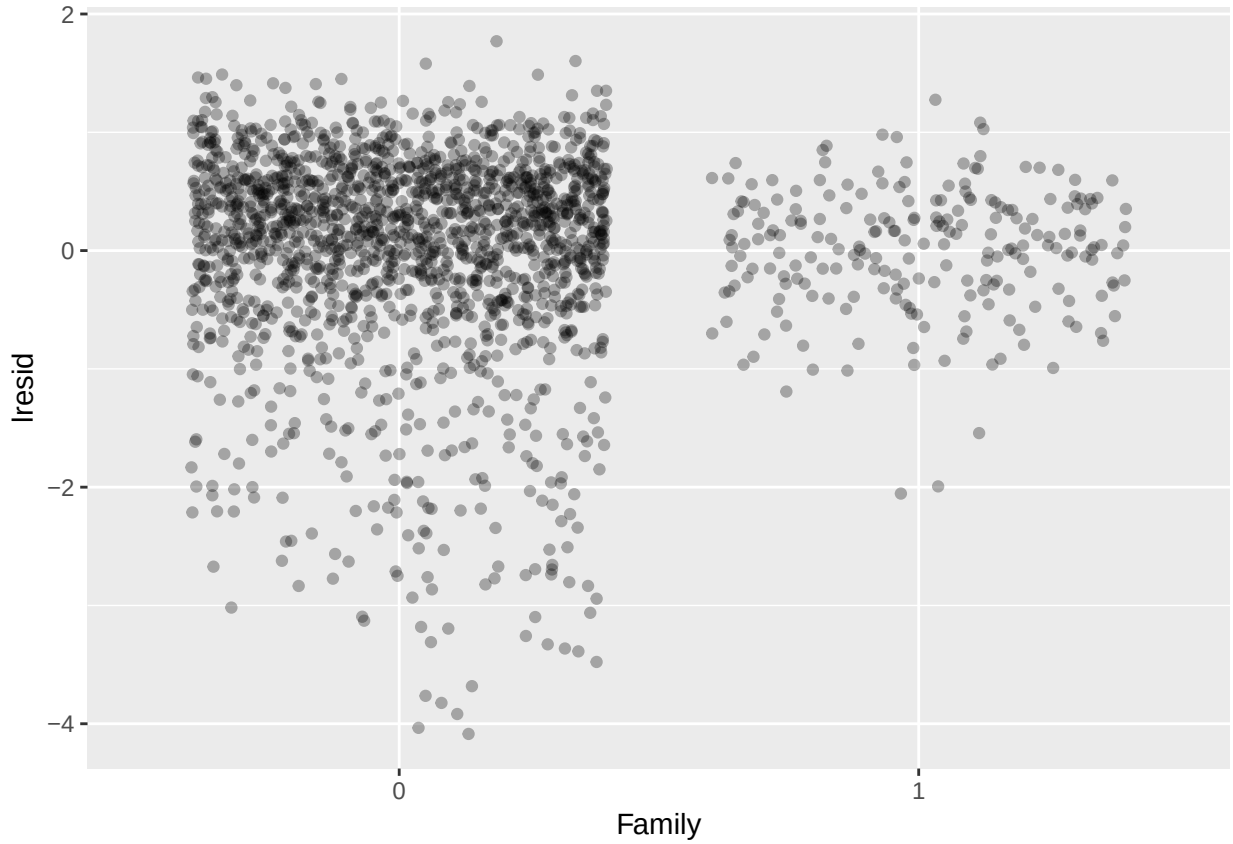
```
## [[1]]
```



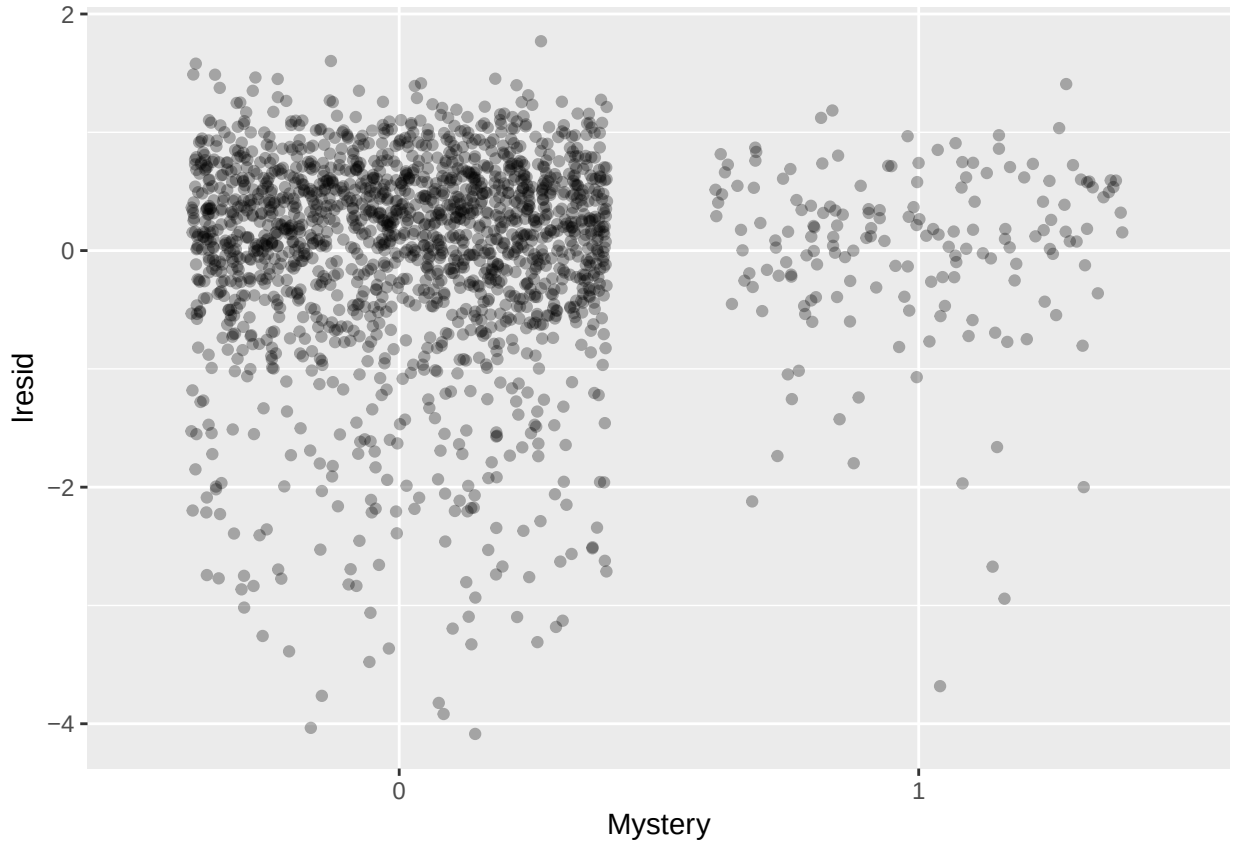

```
##
## [[2]]
```



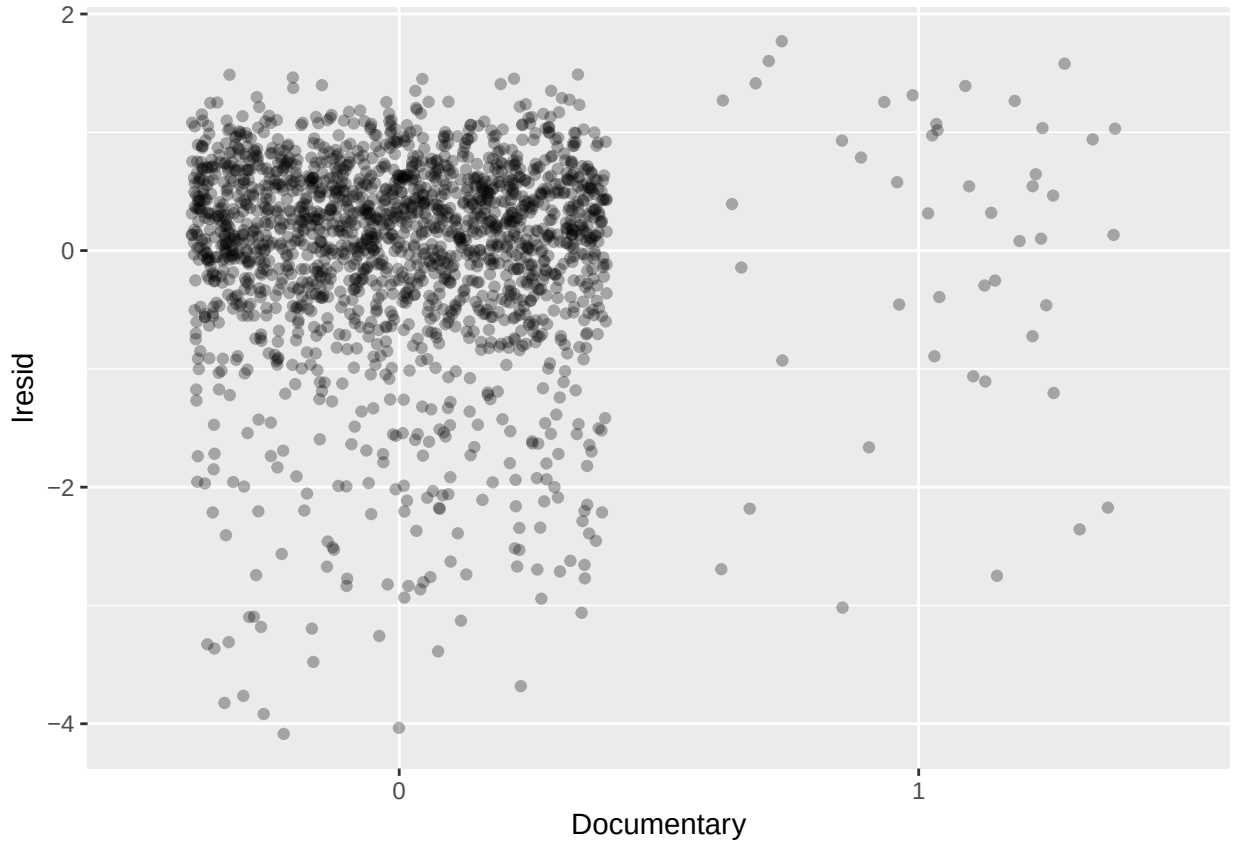

```
##
## [[3]]
```



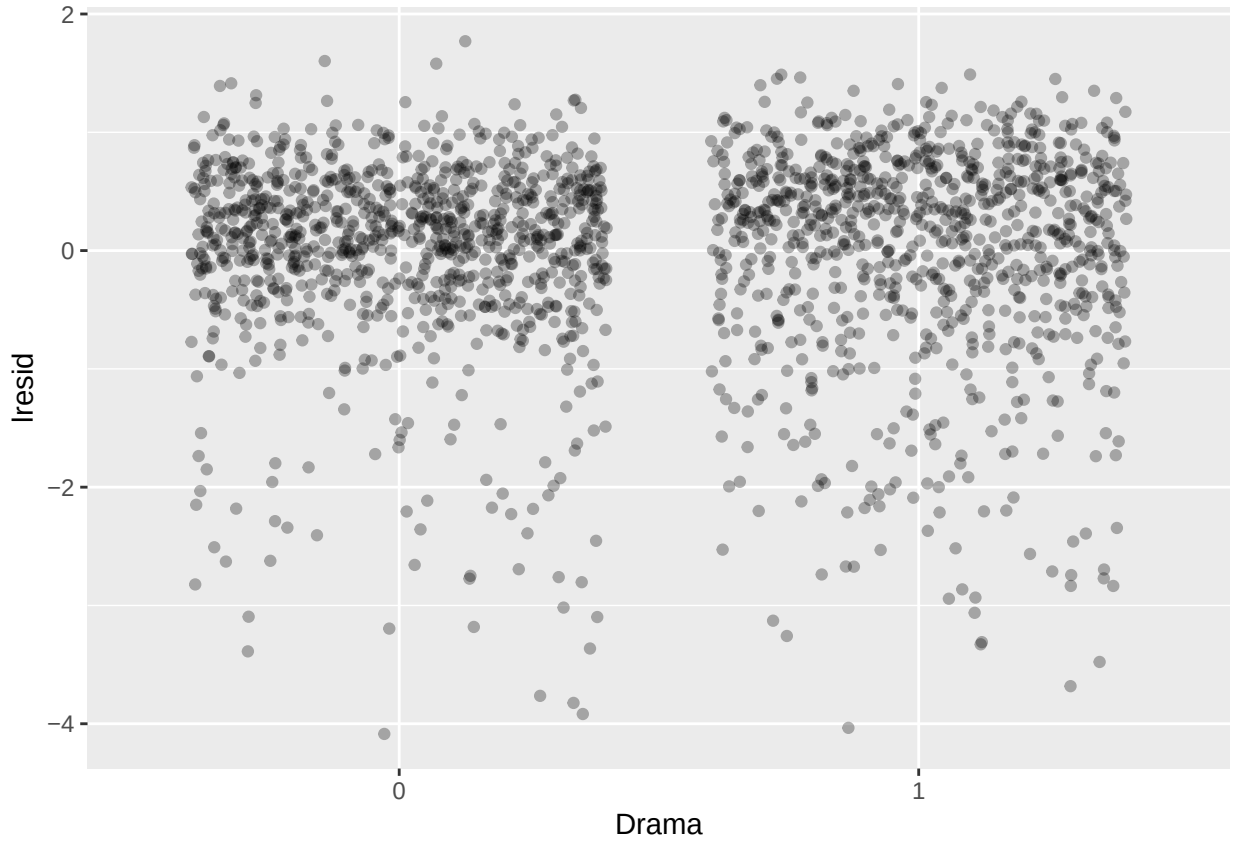

```
##
## [[4]]
```



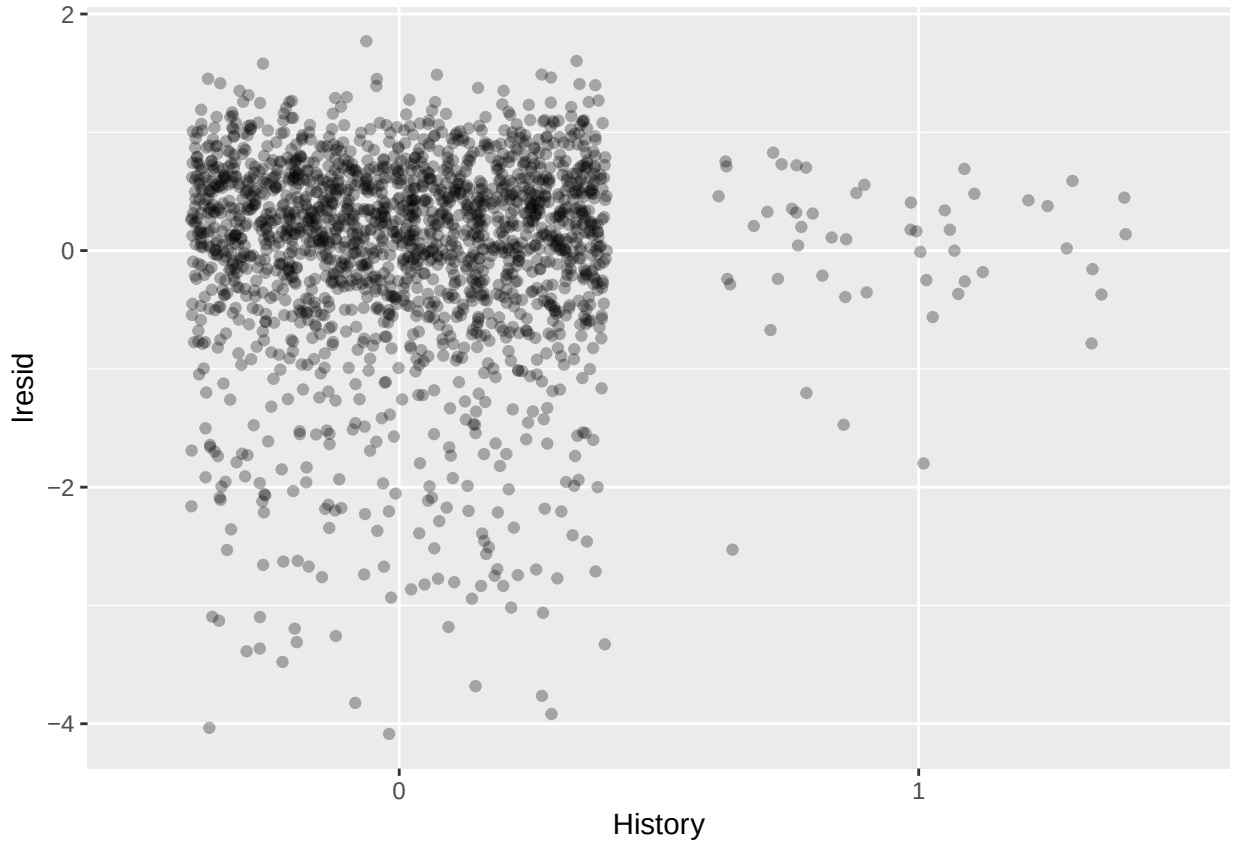

```
##
## [[5]]
```



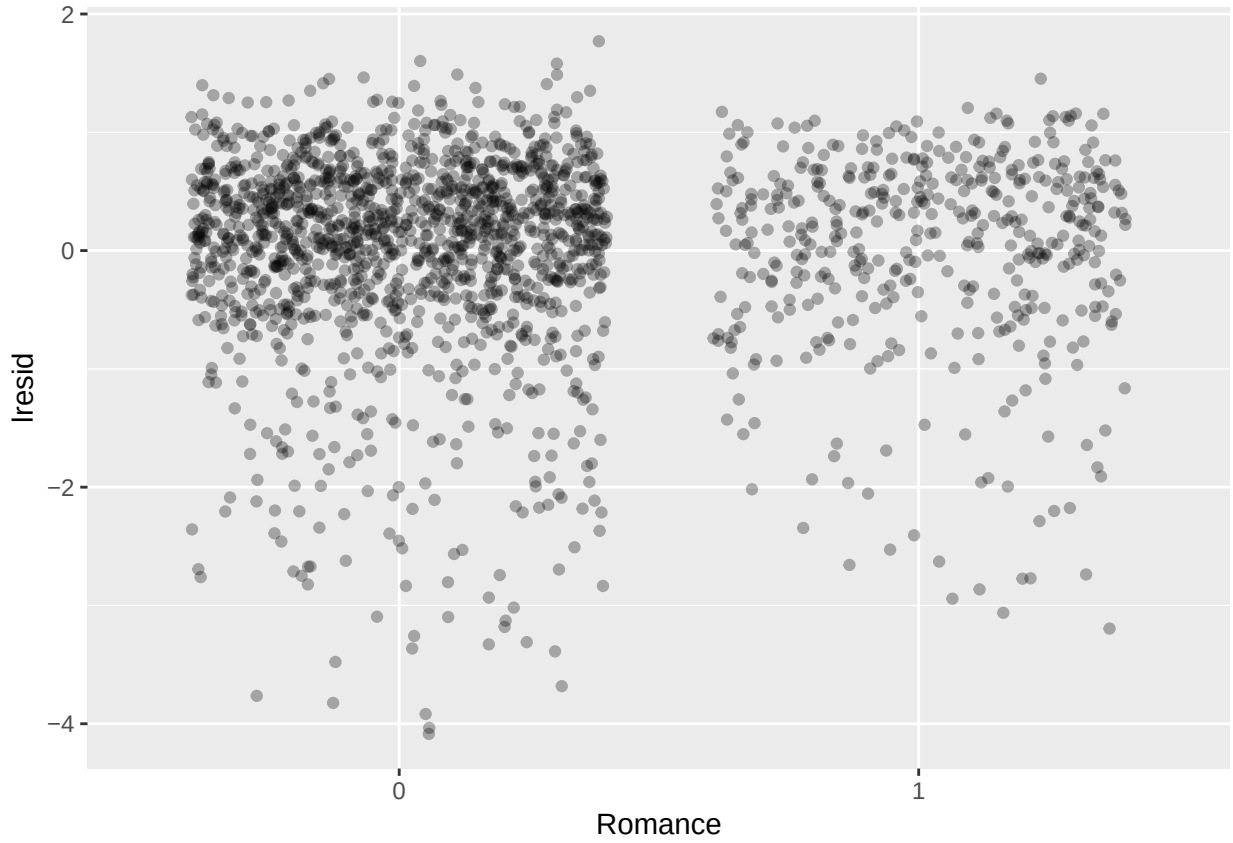

```
##
## [[6]]
```




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

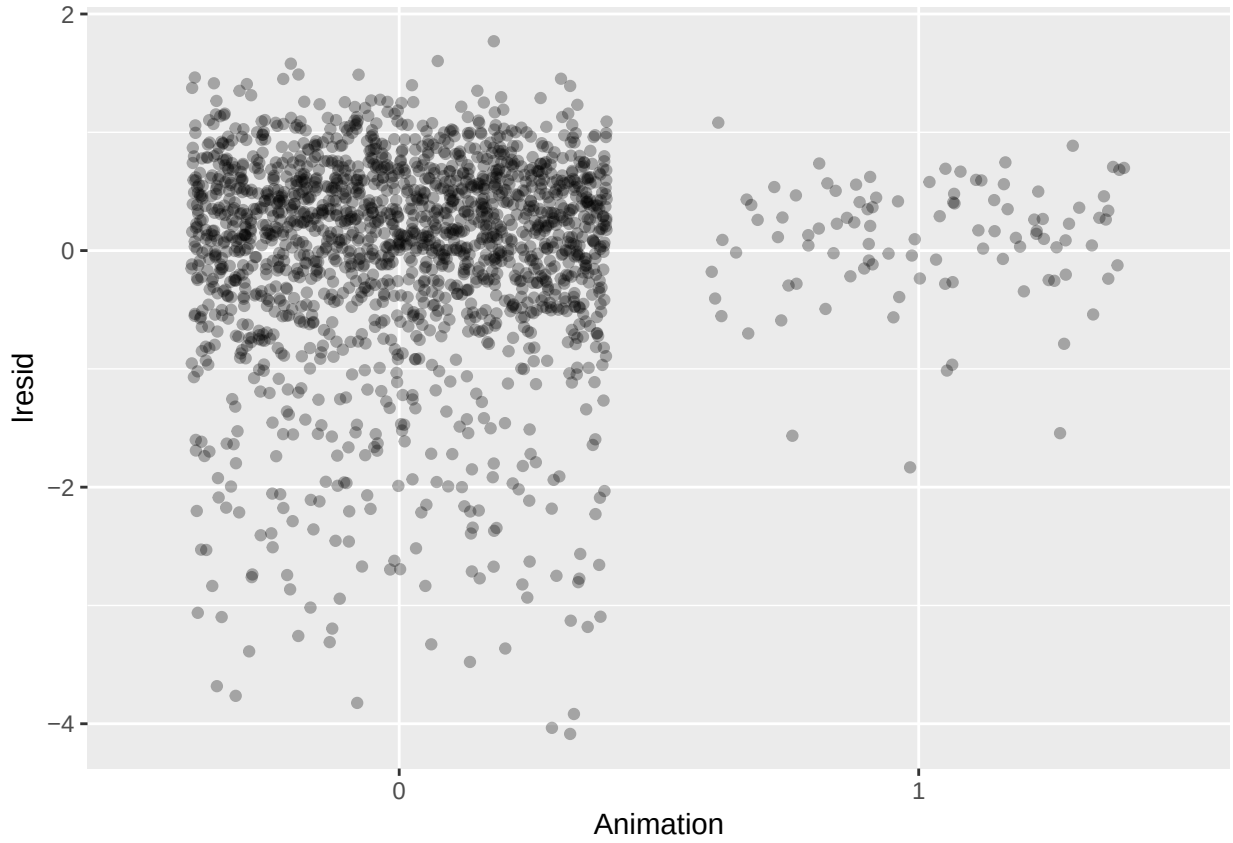



```
##
## [[8]]
```

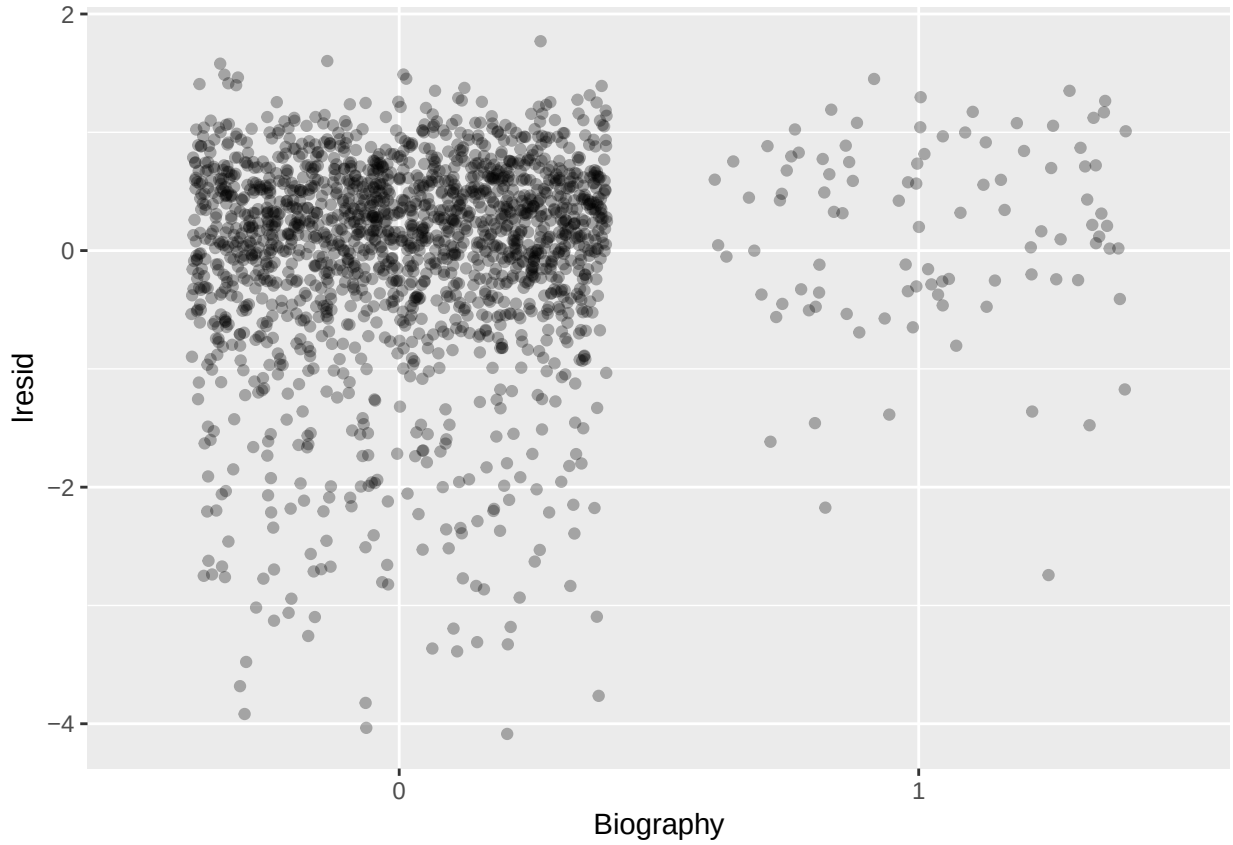



```
# graph residuals against each genre not included in the model
lapply(names(train_genre_only %>% select(-genre_xvar)), function(var) {
  train_resid %>%
    ggplot() +
    geom_jitter(aes_string(var, y = 'lresid'), alpha = .3)
})
```

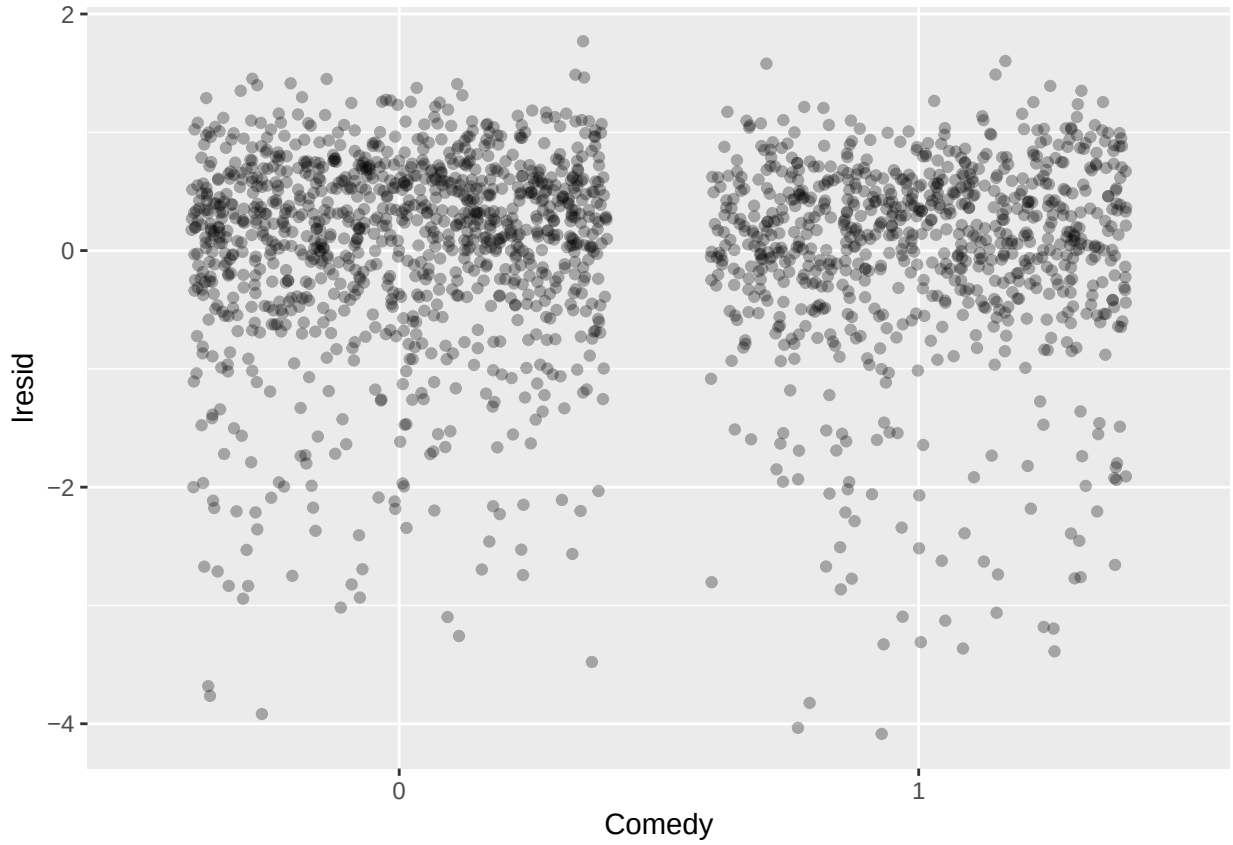
```
## [[1]]
```



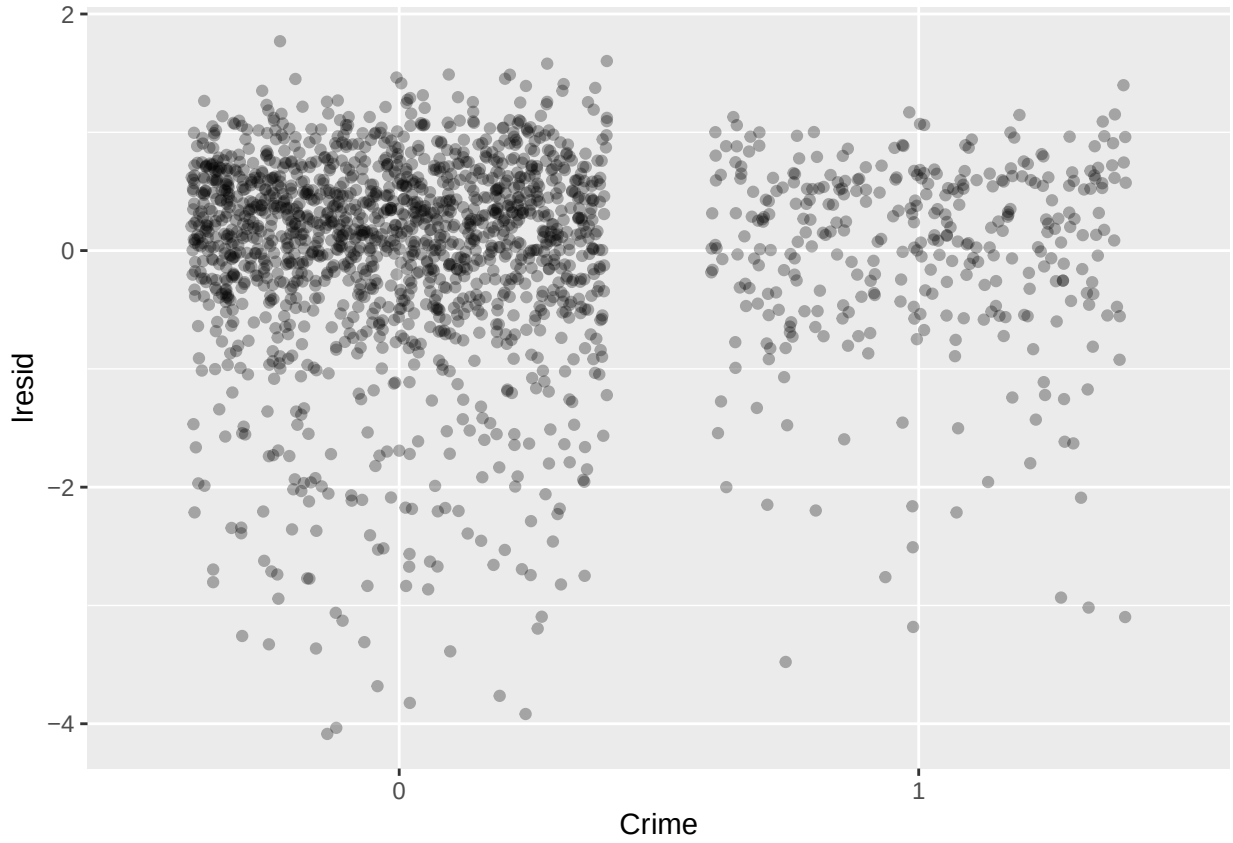

```
##
## [[2]]
```



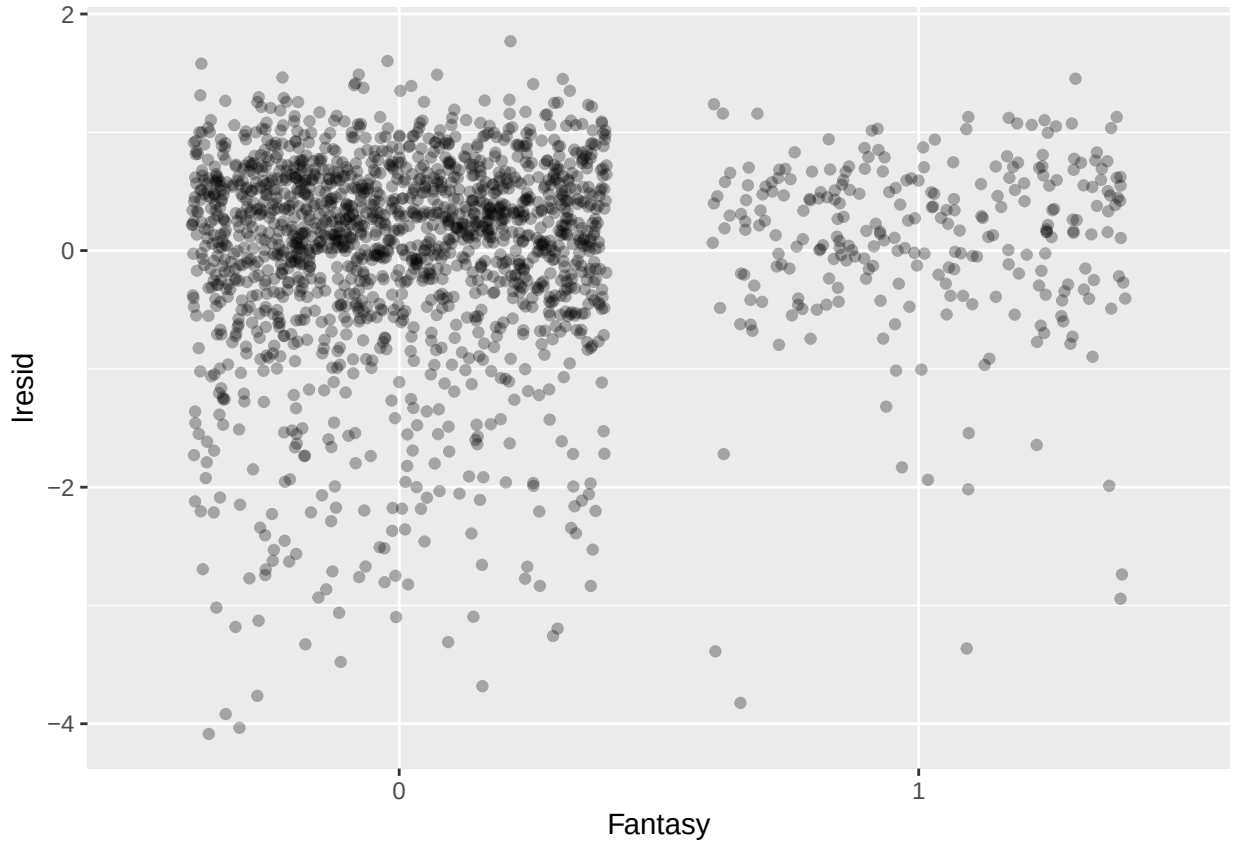

```
##
## [[3]]
```



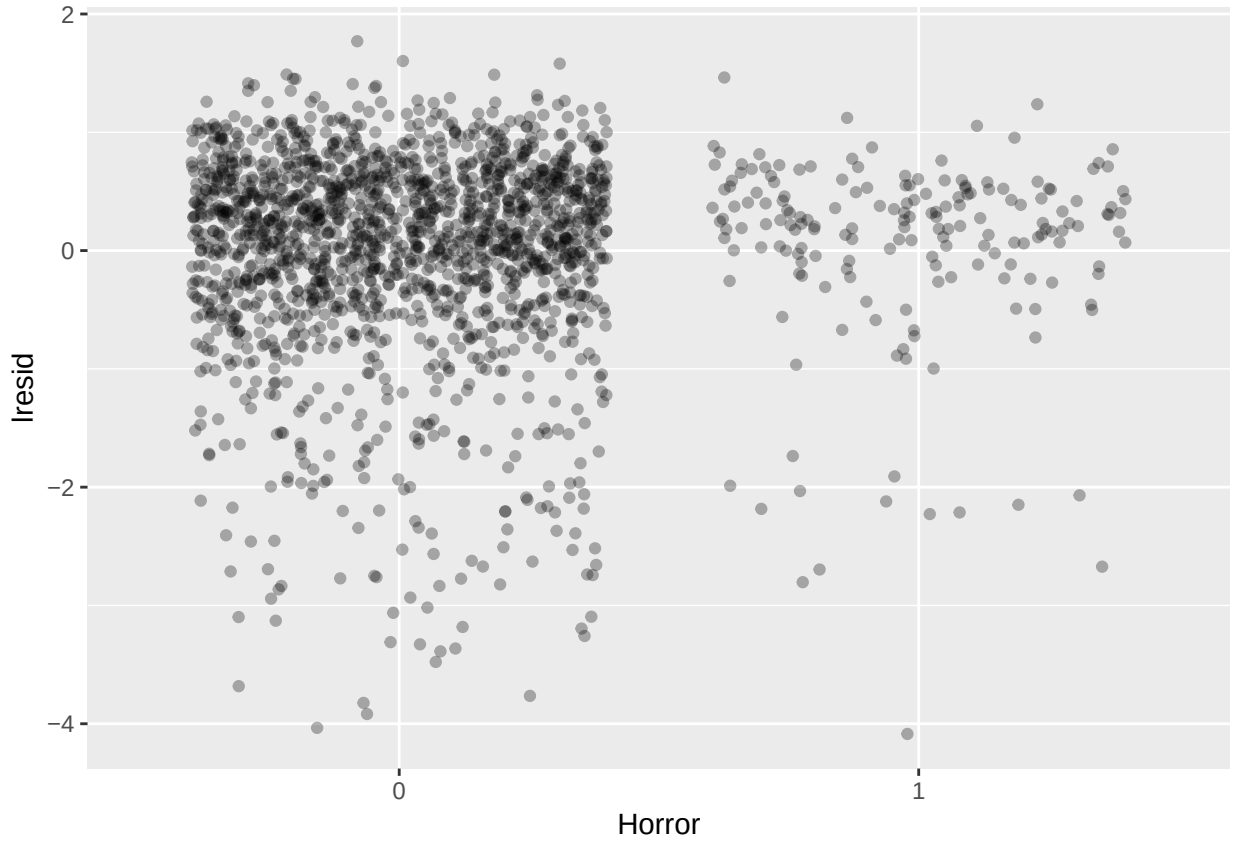

```
##
## [[4]]
```



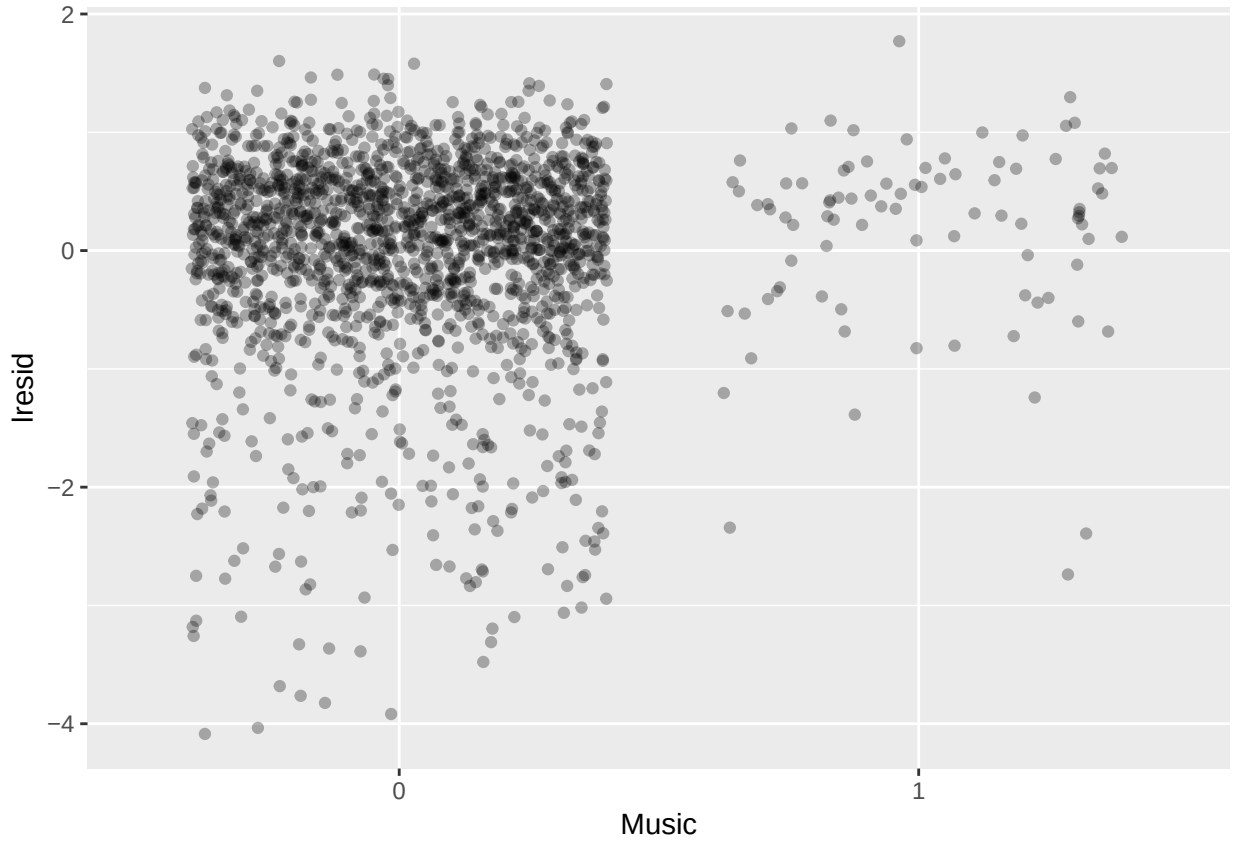

```
##
## [[5]]
```



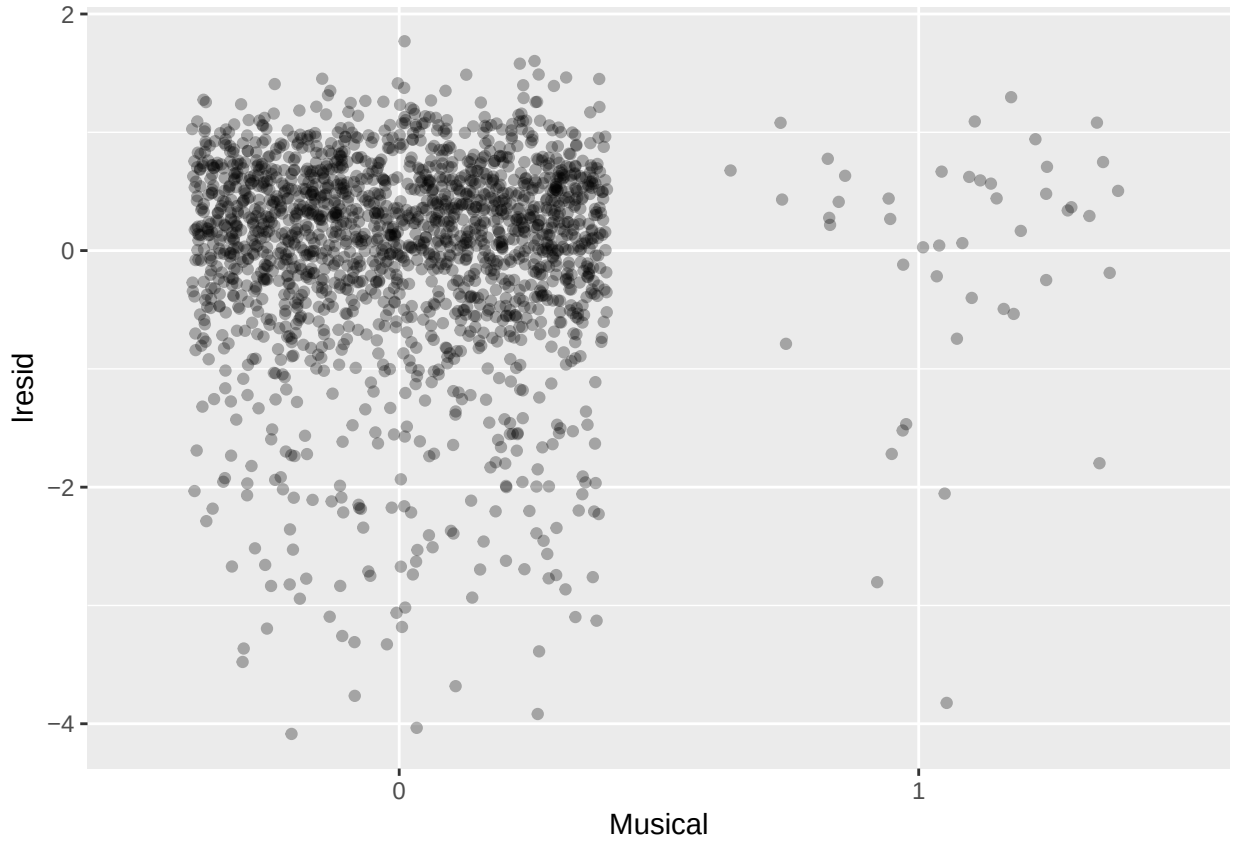

```
##
## [[6]]
```



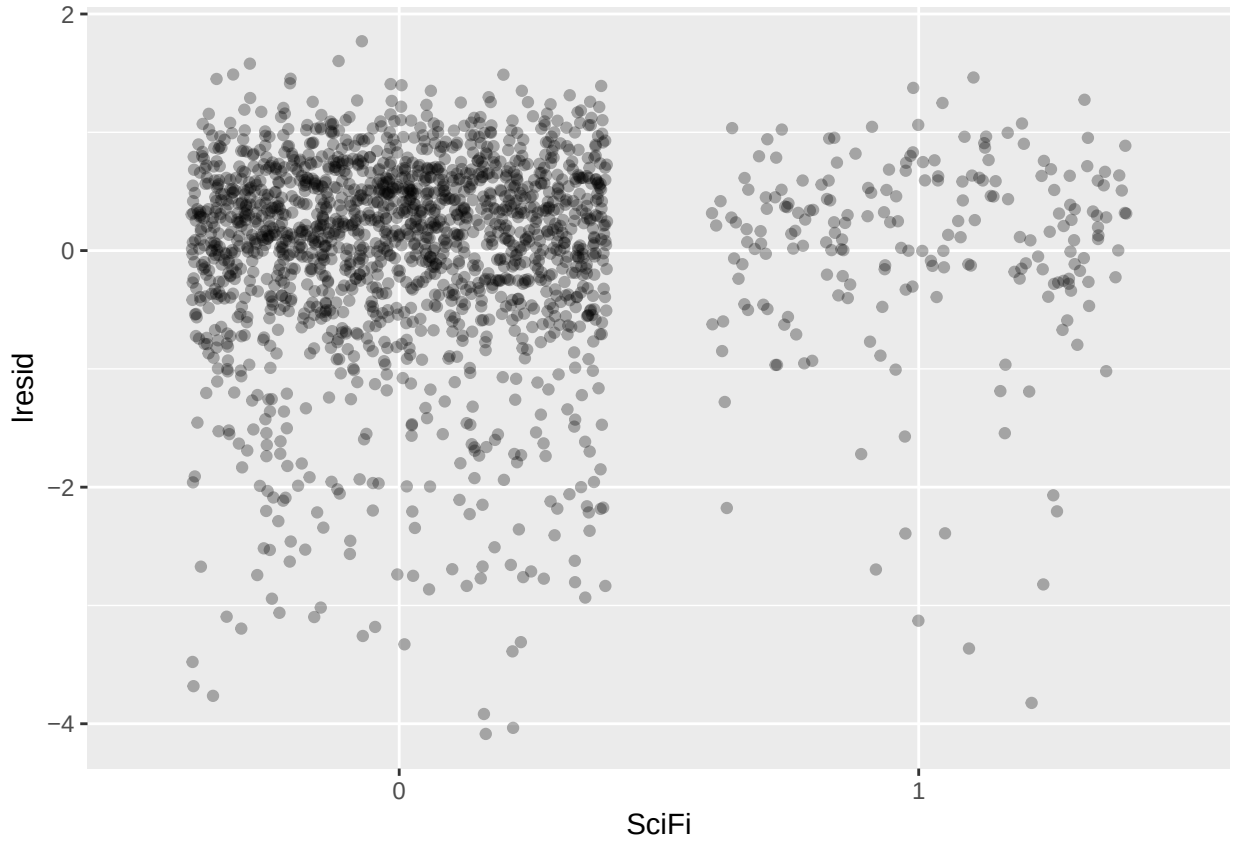

```
##
## [[7]]
```



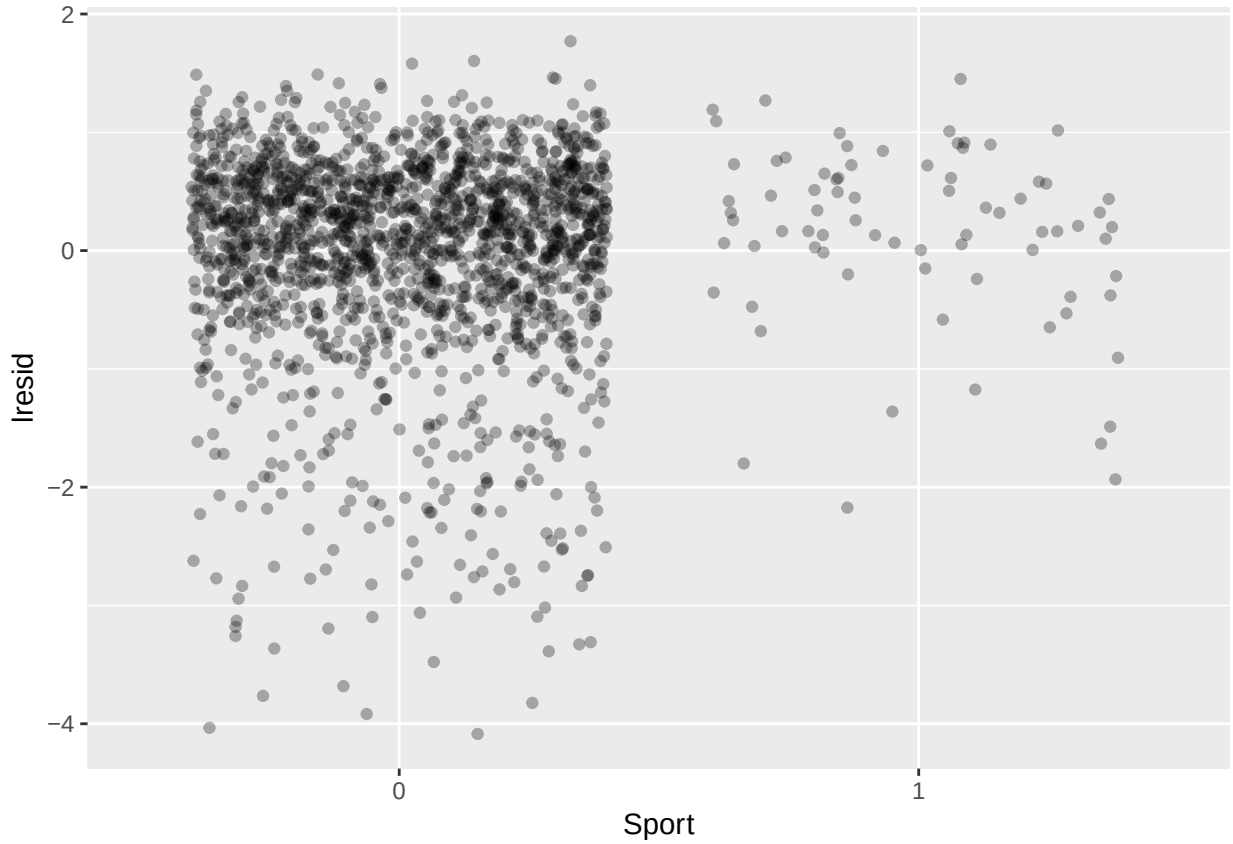

```
##
## [[8]]
```



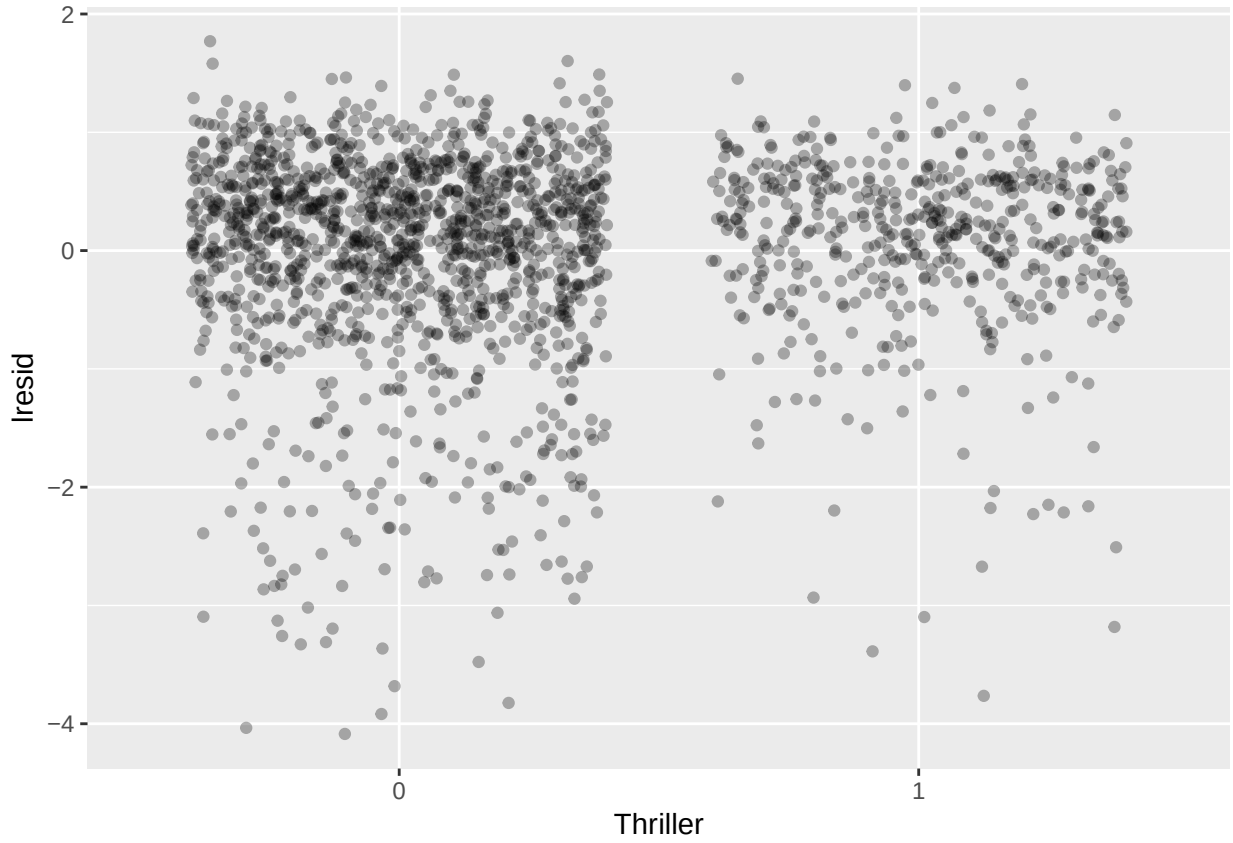

```
##
## [[9]]
```



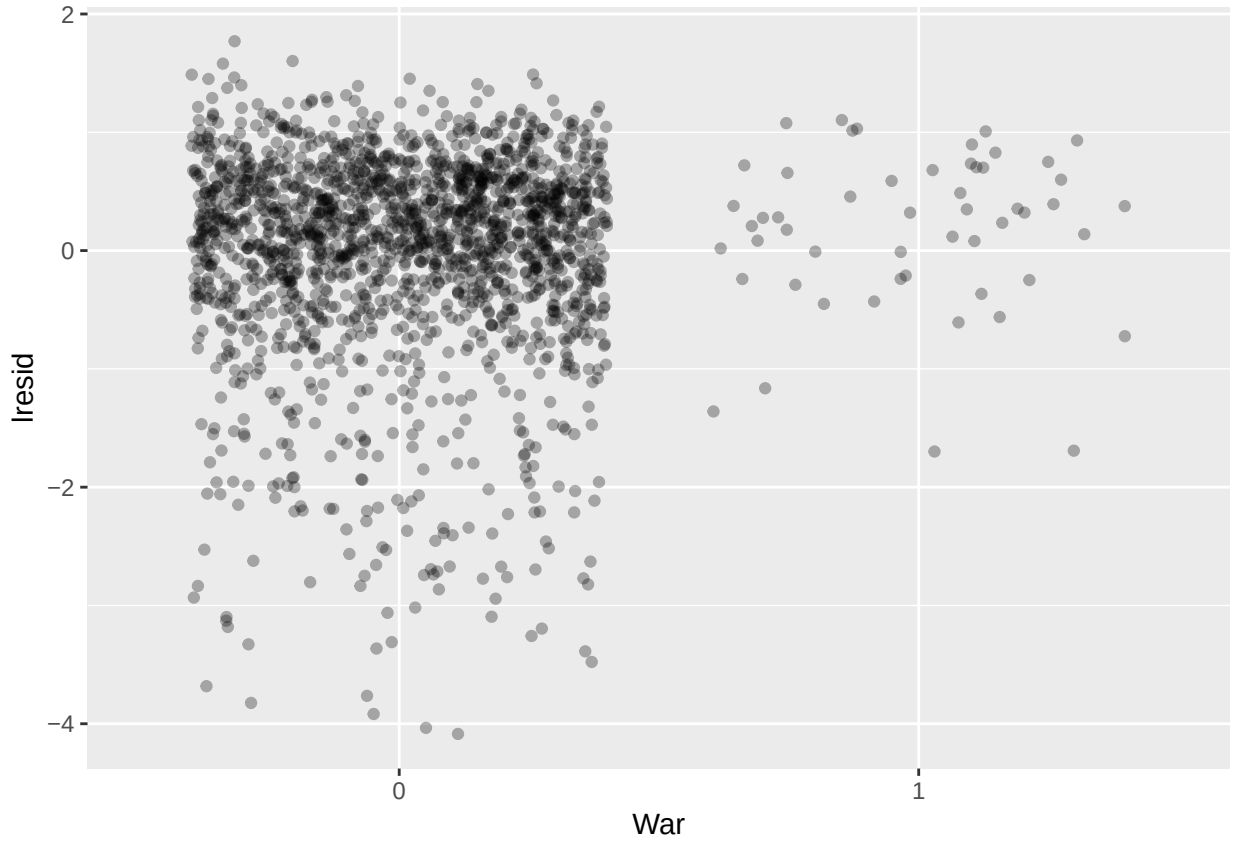

```
##
## [[10]]
```



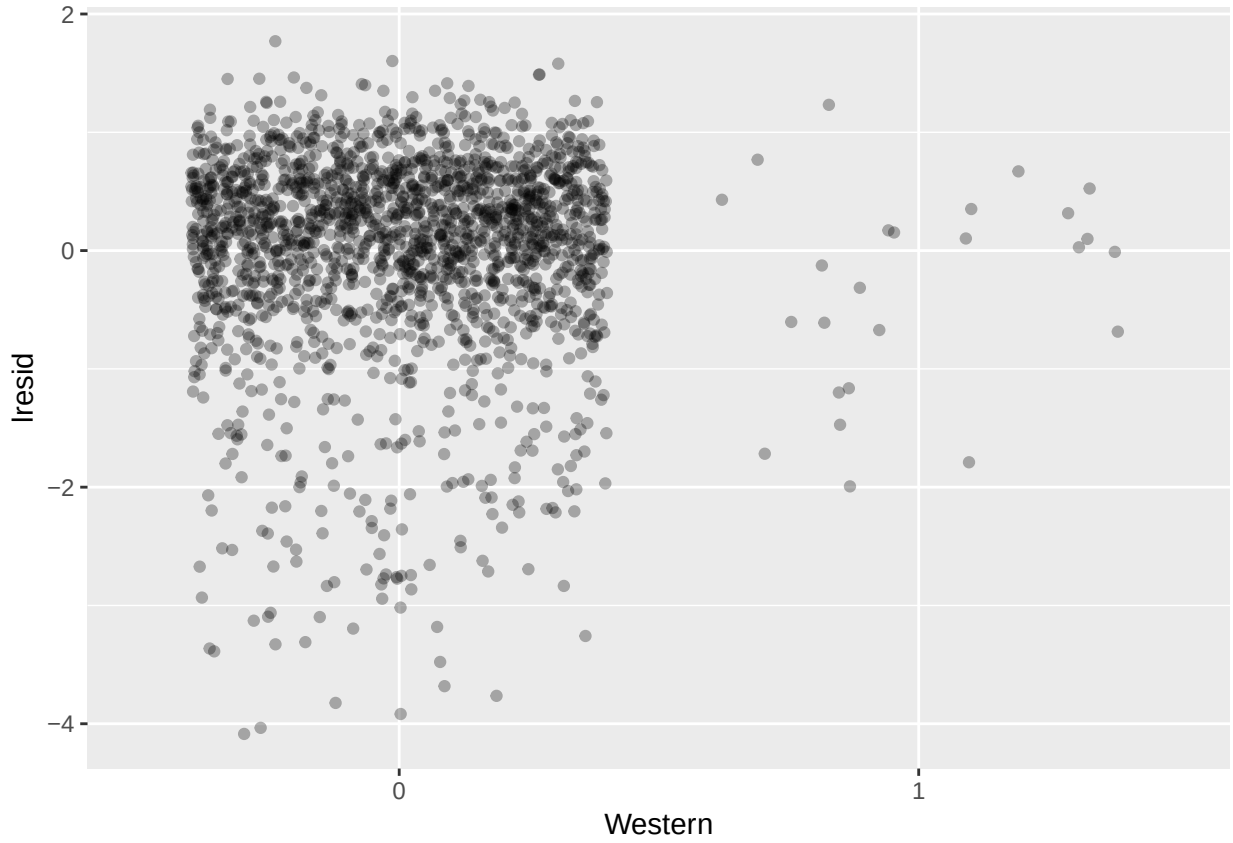

```
##
## [[11]]
```




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

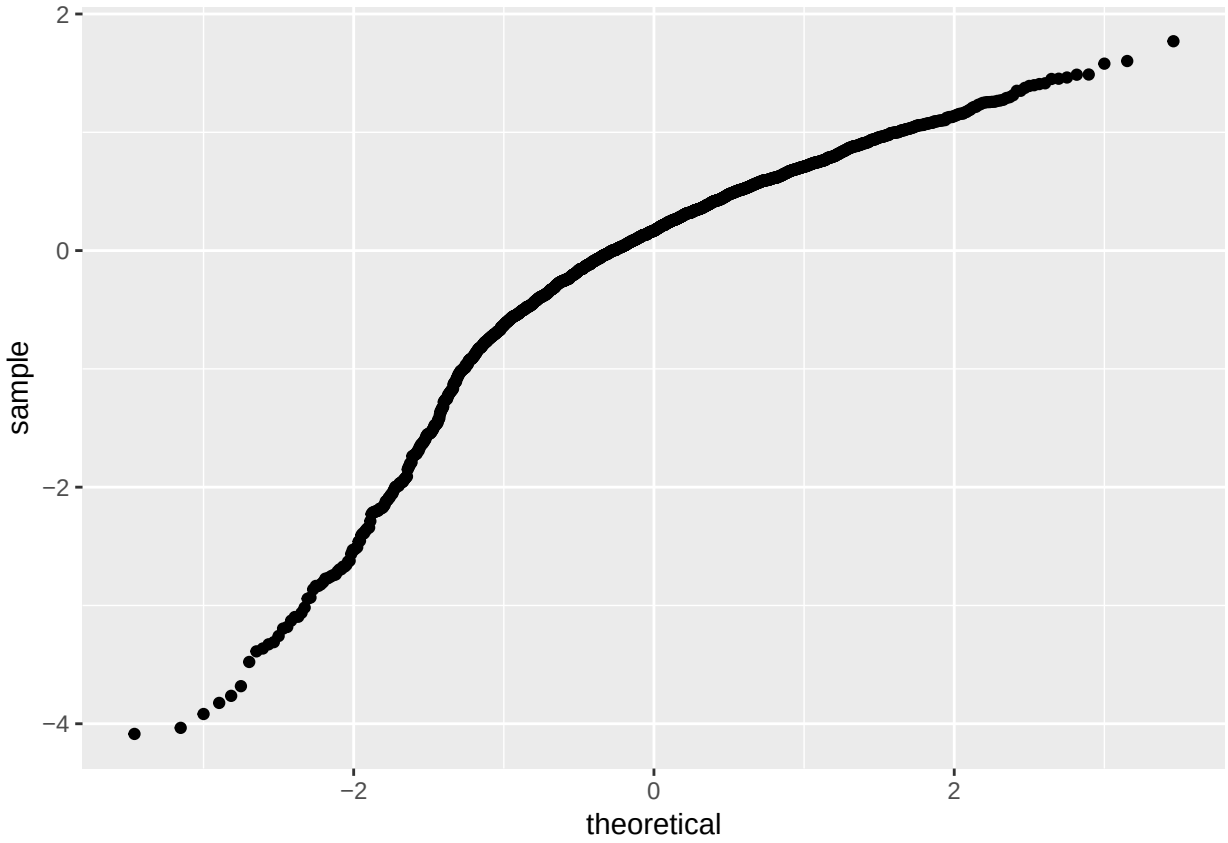



```
##
## [[13]]
```



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))
```

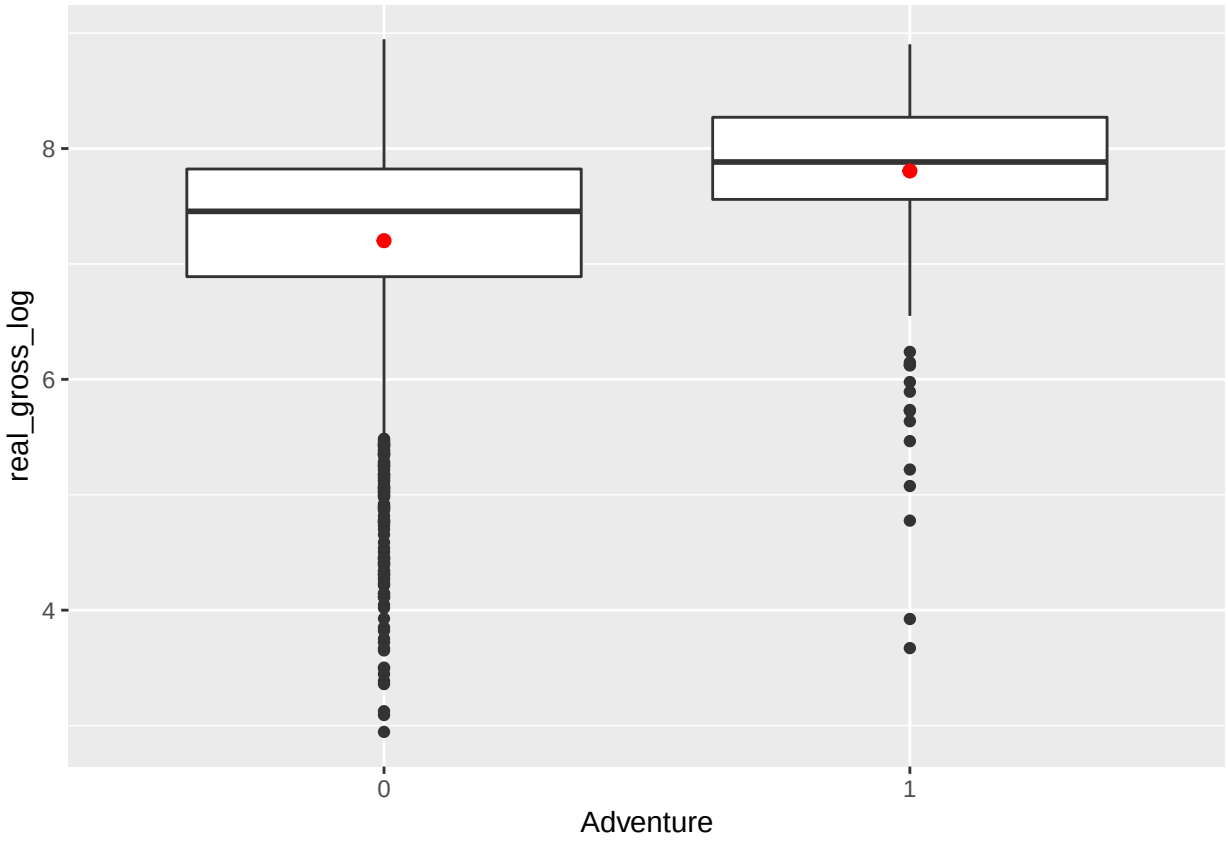


Plot Predictions

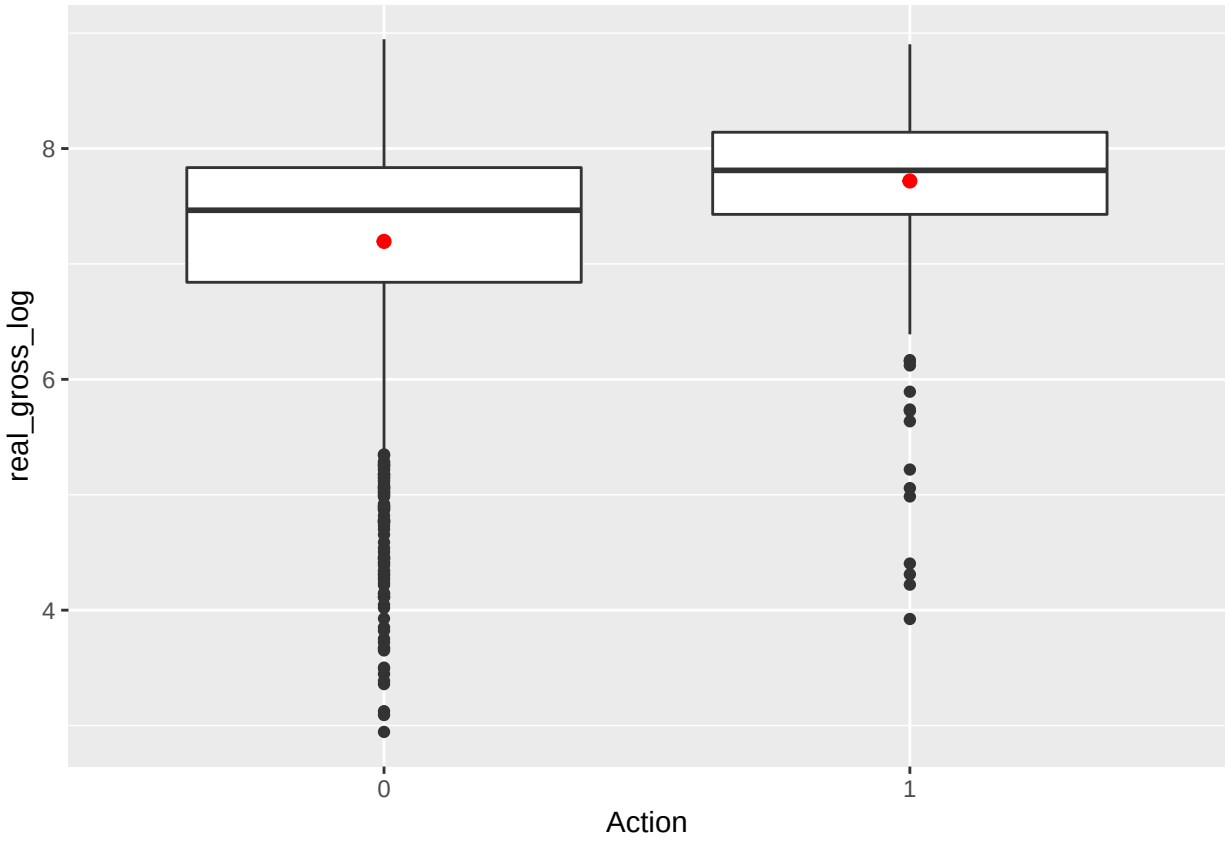
Plot prediction for mean real revenue against each genre included in the model.

```
train_pred <- train %>% add_predictions(mod_genre, 'lpred')

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

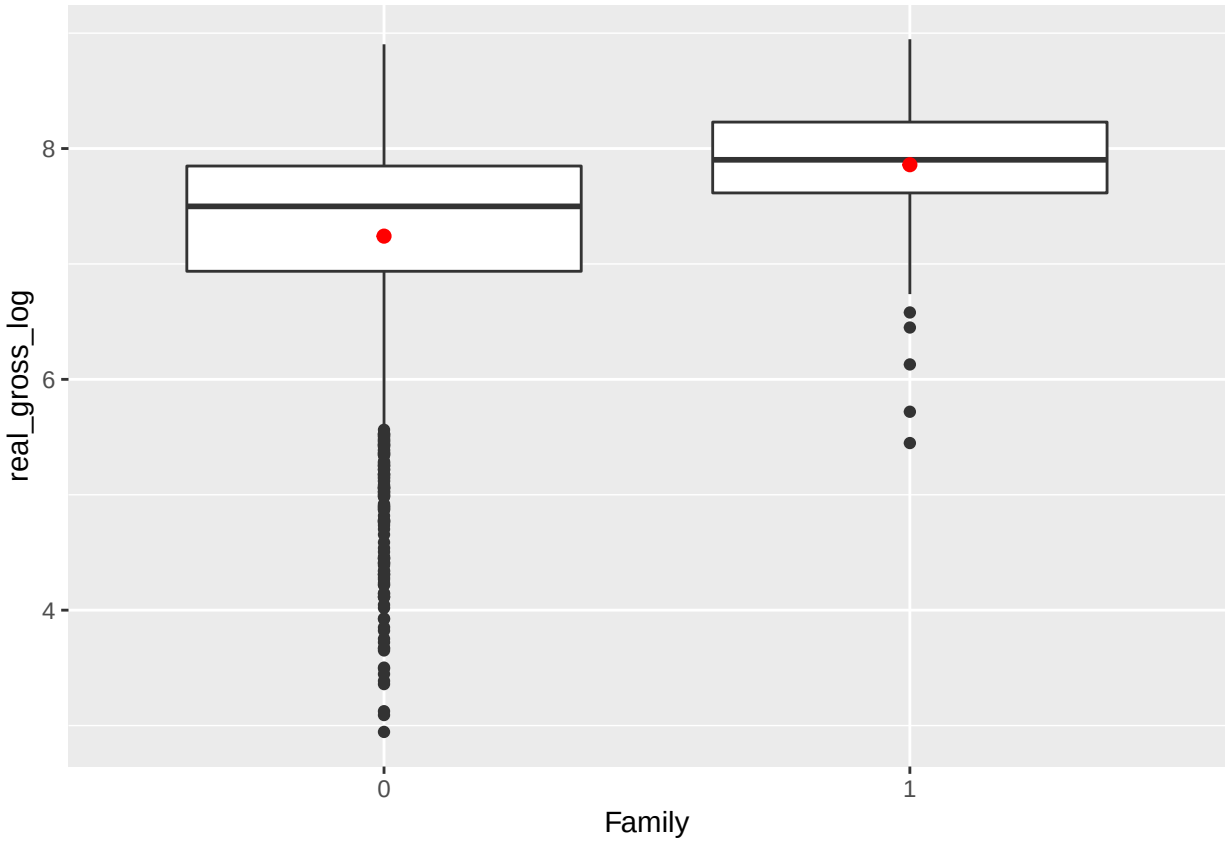
```
## [[1]]
```



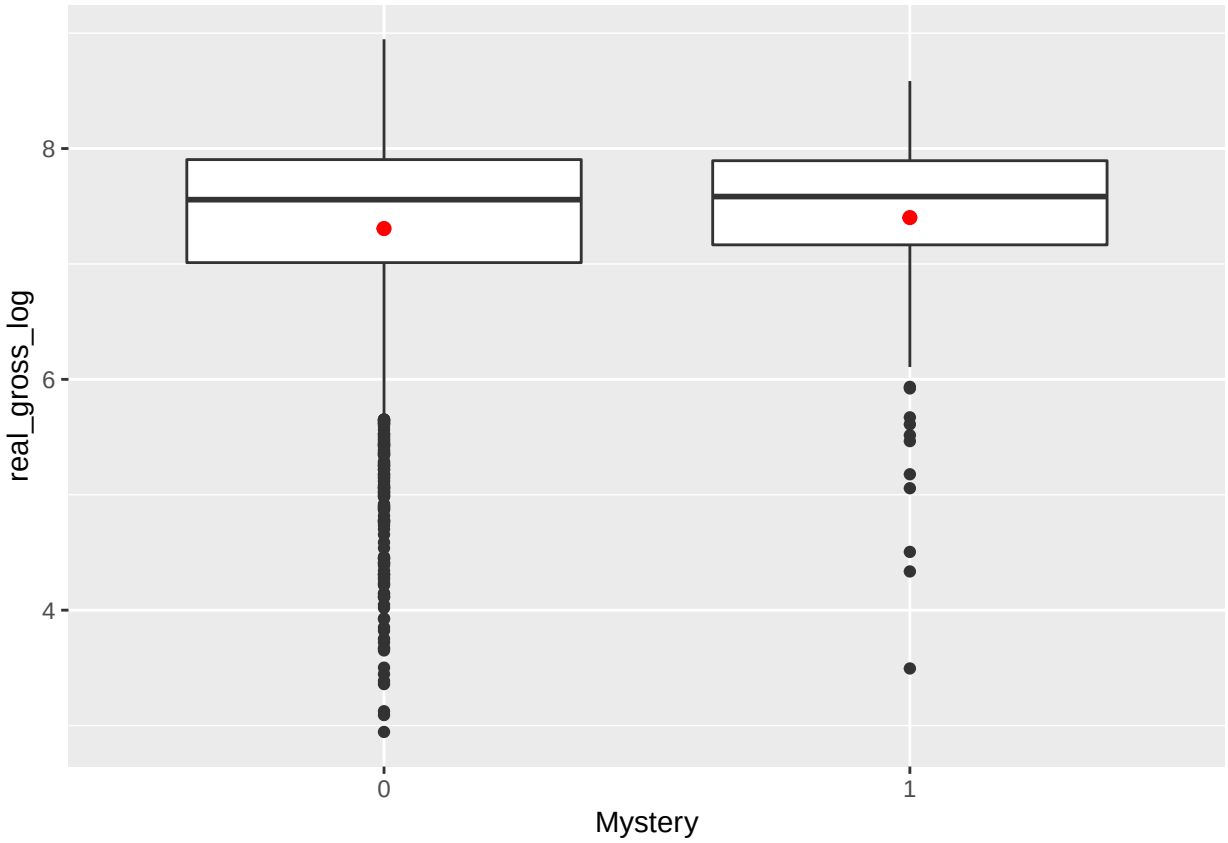

```
##
## [[2]]
```



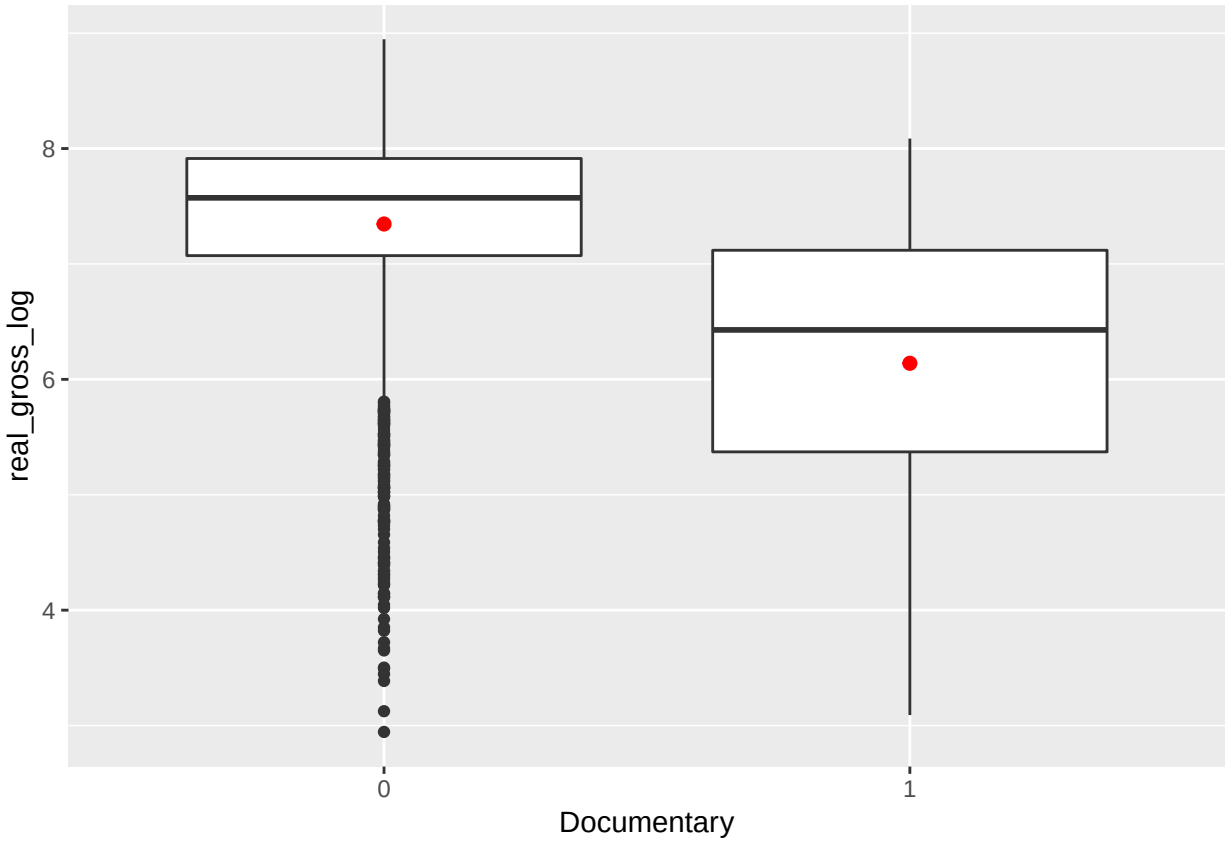

```
##
## [[3]]
```



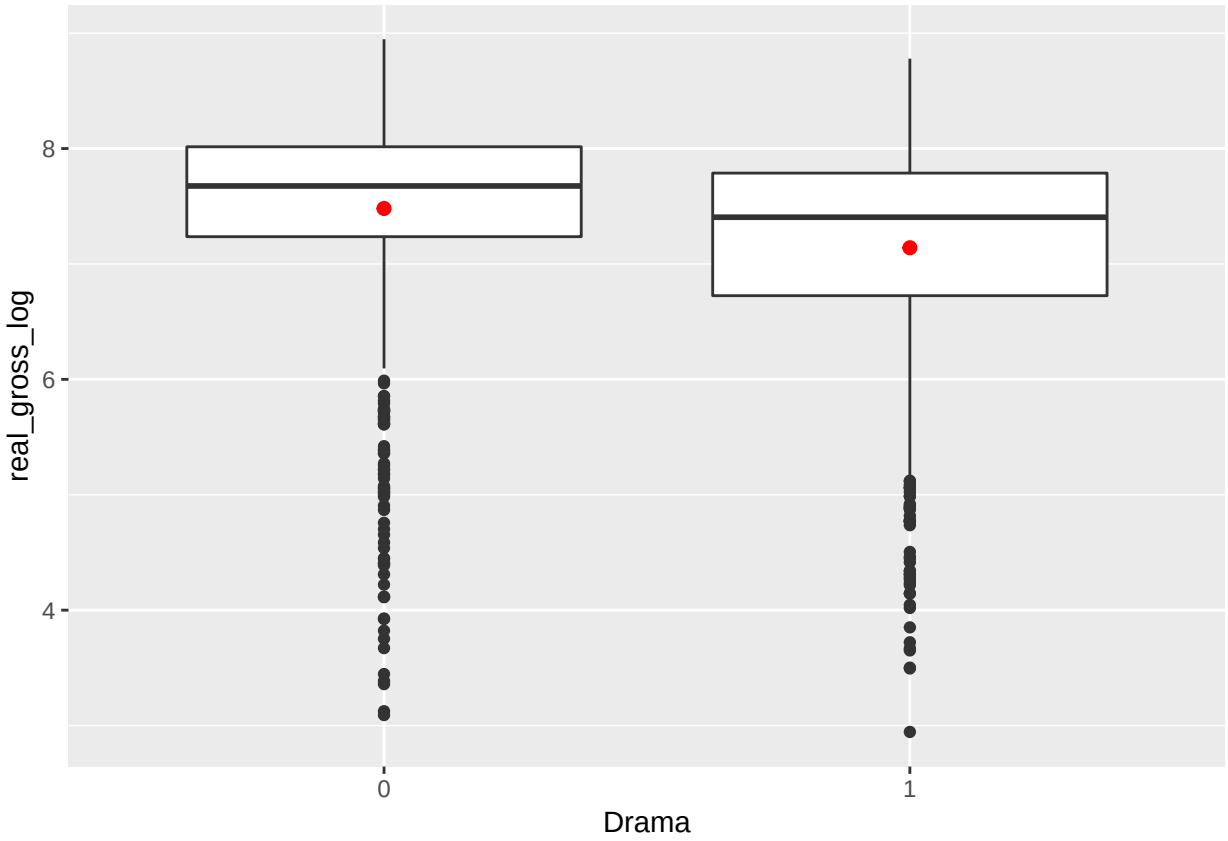

```
##
## [[4]]
```



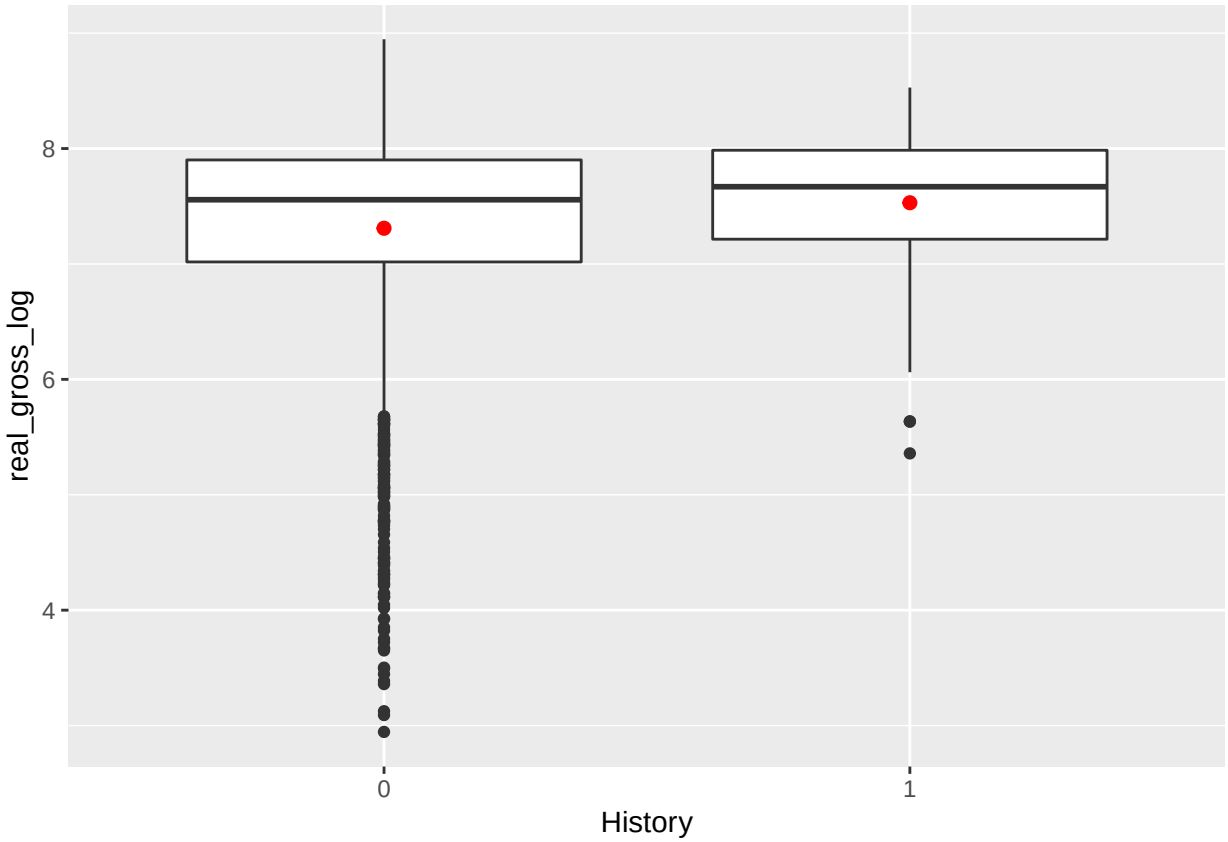

```
##
## [[5]]
```



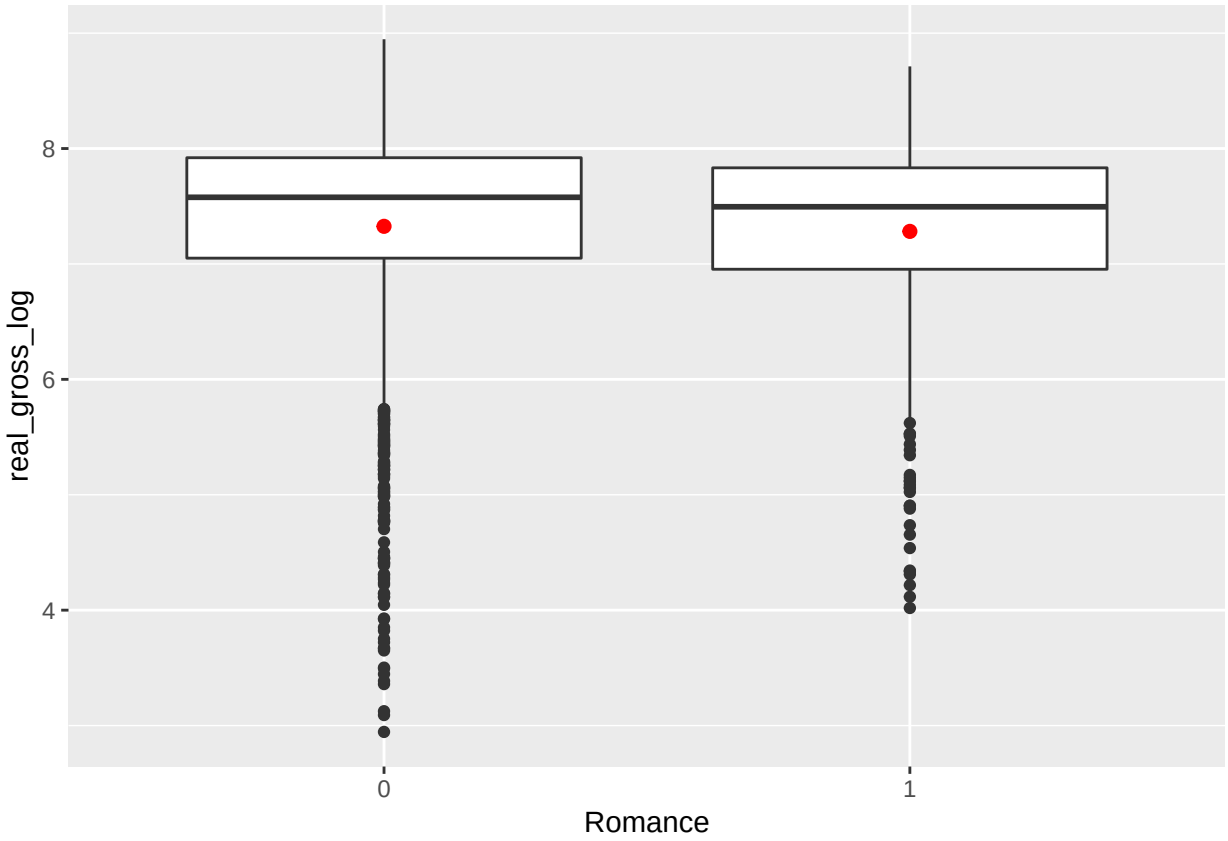

```
##
## [[6]]
```




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



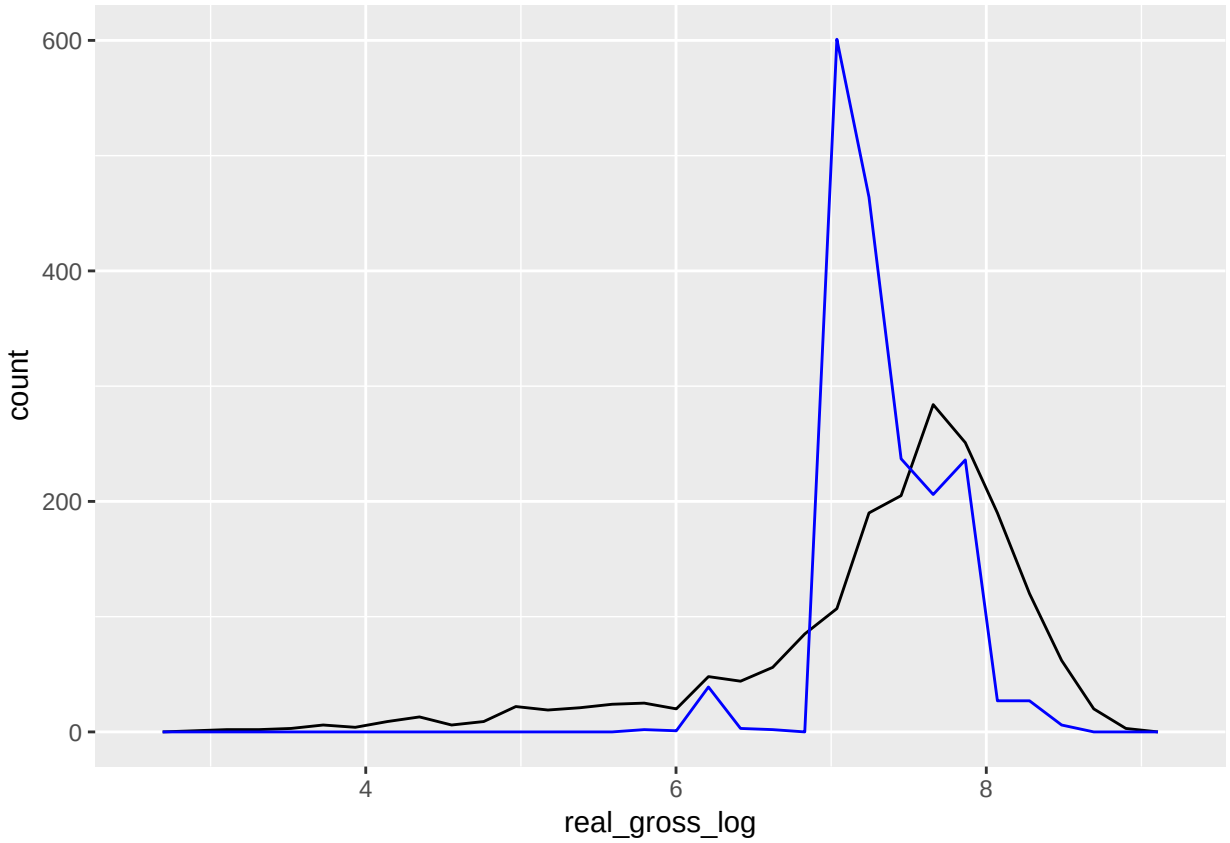

```
##
## [[8]]
```



Overall predictions: clearly not enough to just specify genres

```
train %>%
  add_predictions(mod_genre, 'lpred') %>%
  ggplot() +
  geom_freqpoly(aes(x = real_gross_log)) +
  geom_freqpoly(aes(x = lpred), color = 'blue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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
```

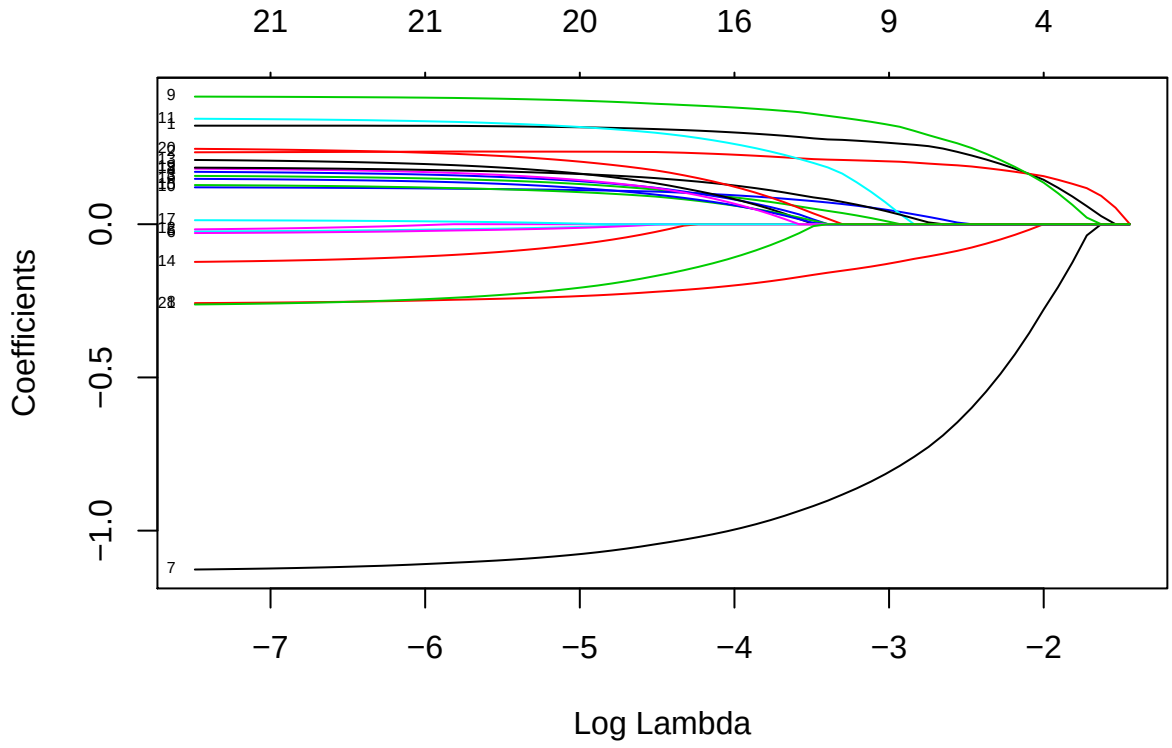
```
## Warning: package 'foreach' was built under R version 3.5.3
```

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

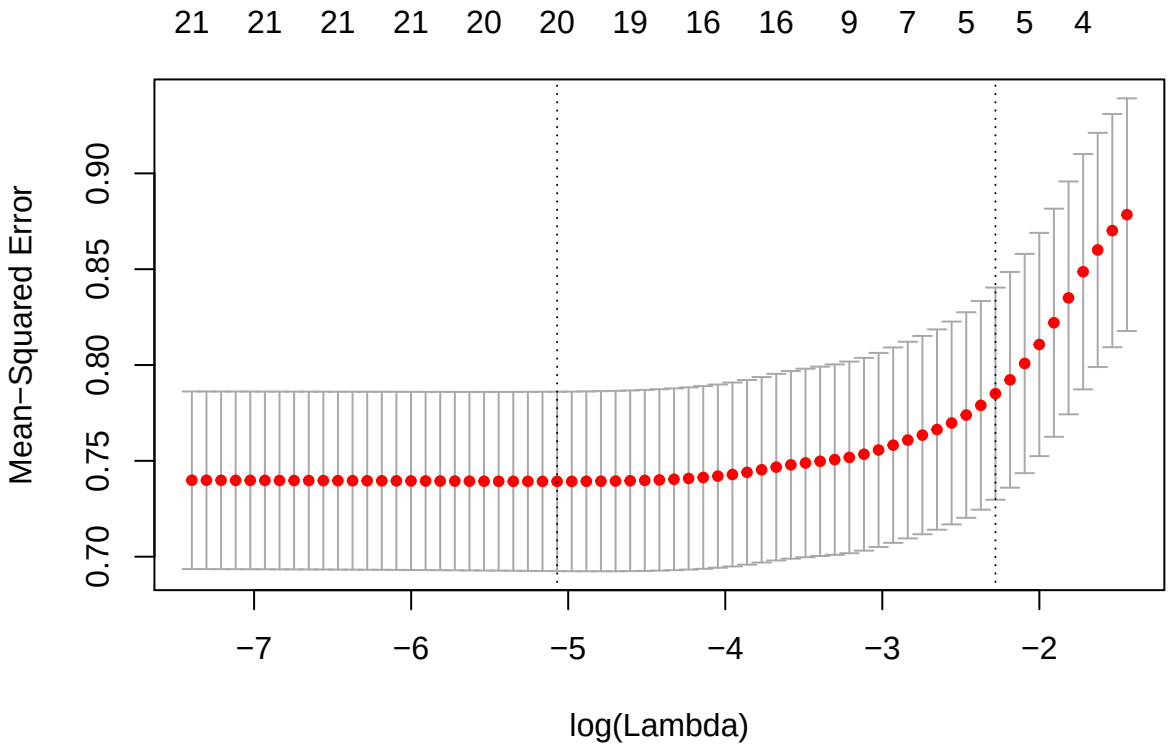
```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## 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)  7.158319467
## Action       0.316866270
## Adventure    0.236784610
## Animation    0.133081838
## Biography    0.142672310
## Comedy      -0.009843859
## Crime       -0.012866133
## Documentary -1.080069143
## Drama       -0.236062771
## Family       0.404262630
## Fantasy      0.111824039
## History      0.319273704
## Horror       .
## Music        0.168382172
## Musical     -0.069374568
## Mystery      0.106229872
## Romance      0.120292330
## SciFi        0.002684535
## Sport        0.146848534
## Thriller     0.165198422
## War          0.207283391
## Western     -0.211255187
```

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

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  7.24823459
## Action       0.19684629
## Adventure    0.17743337
## Animation    .
## Biography    .
## Comedy       .
## Crime        .
## Documentary -0.48840116
## Drama       -0.04977566
## Family       0.20946931
## 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.
Many years have revenue either higher or lower than that 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.

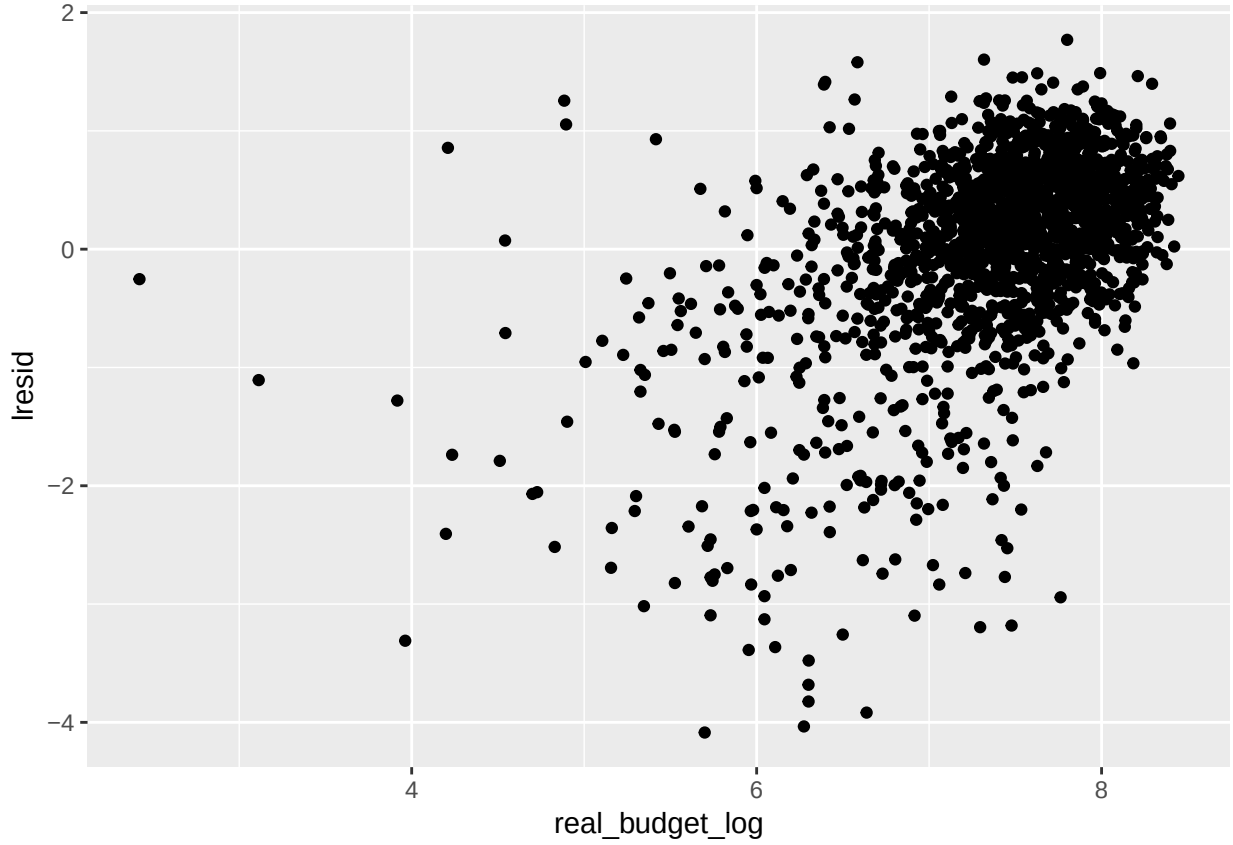
```
# get log versions of variables since residuals are log: same scale
train_resid <- train_resid %>%
  mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score'), funs(log = log10(.)))

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

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

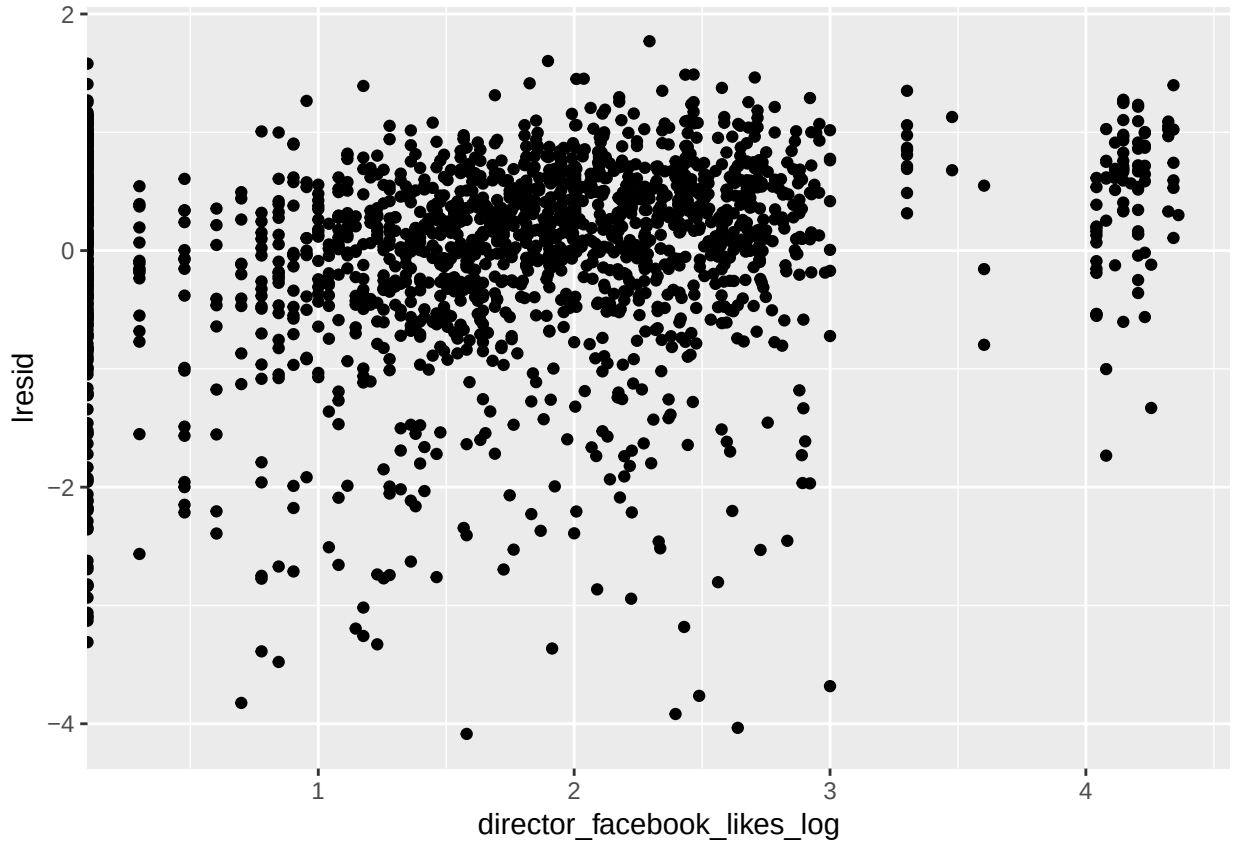
```
## [1] "real_budget"
## [1] "director_facebook_likes"
## [1] "cast_total_facebook_likes"
## [1] "imdb_score"
```

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

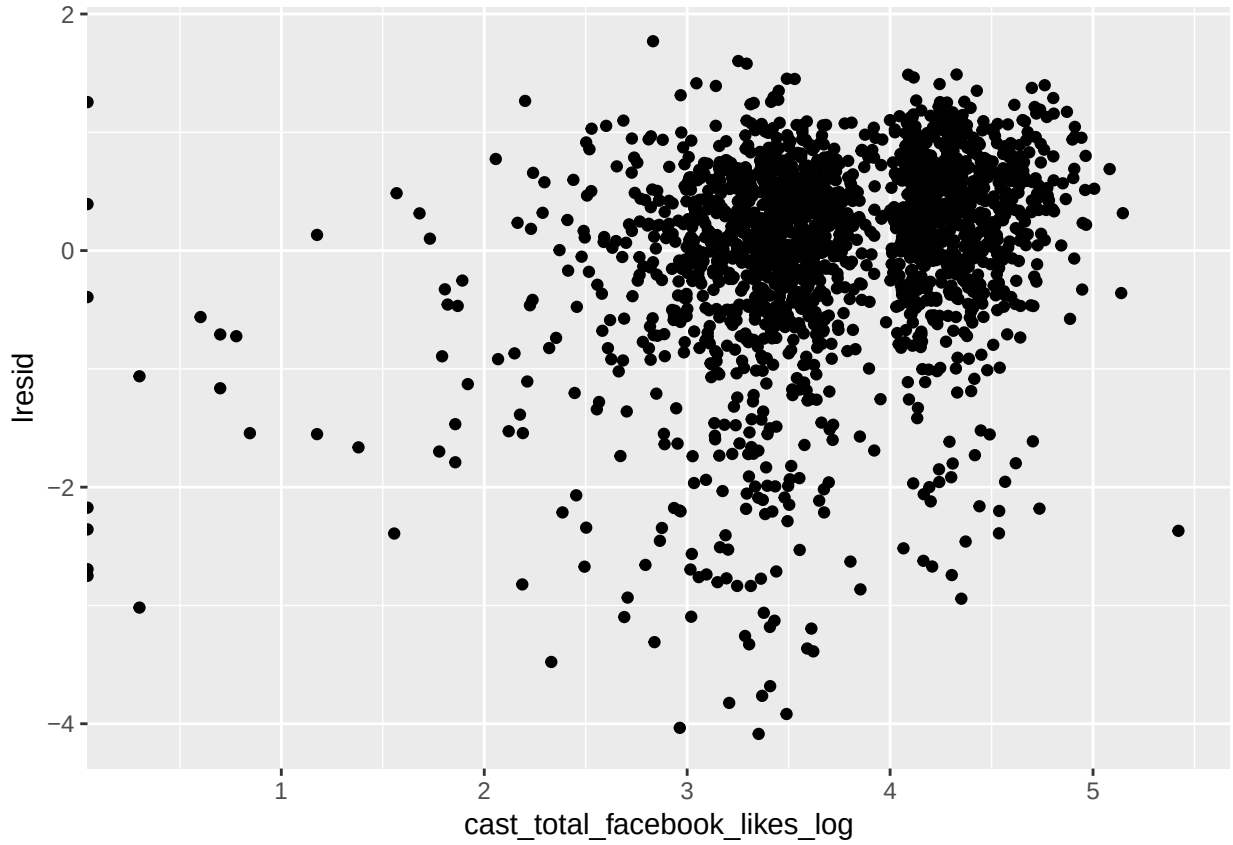
```
## Warning: Removed 102 rows containing missing values (geom_point).
```



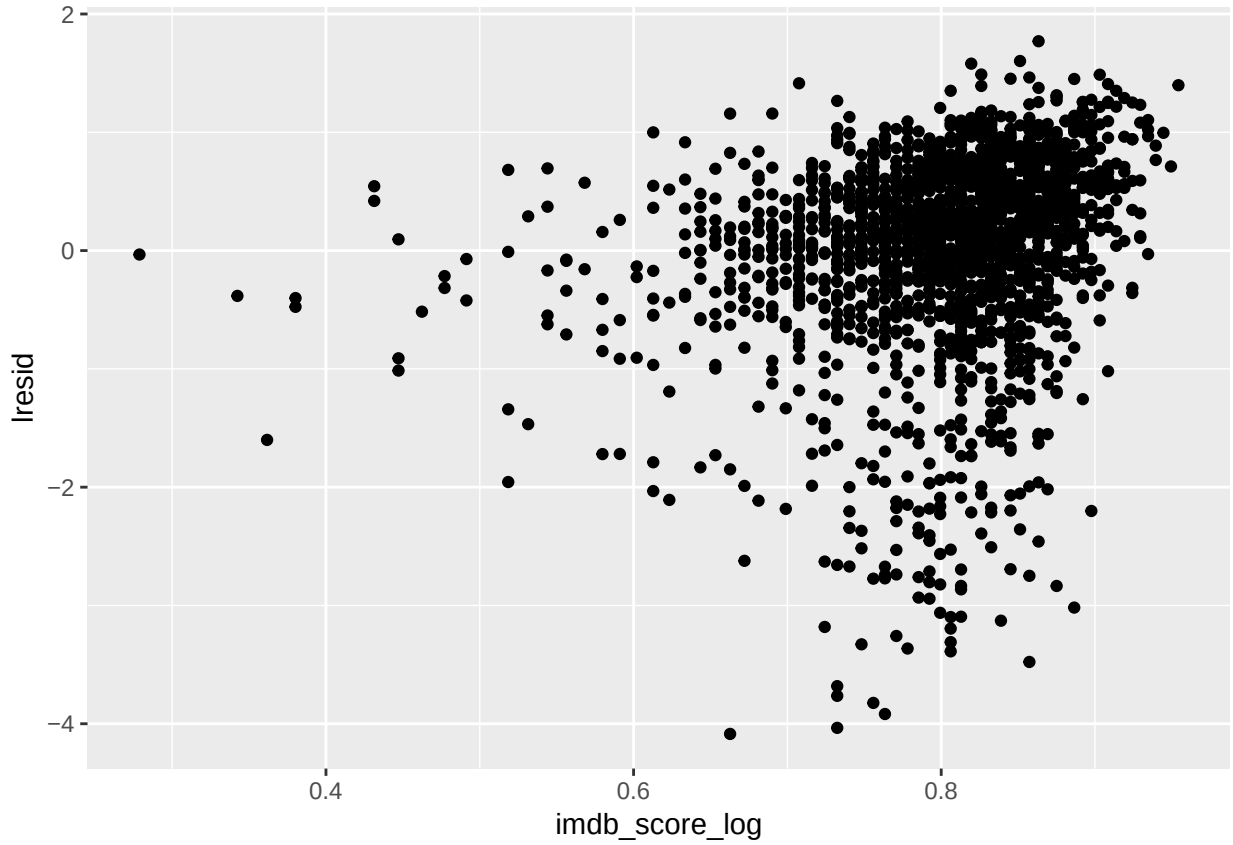

```
##
## [[2]]
```




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

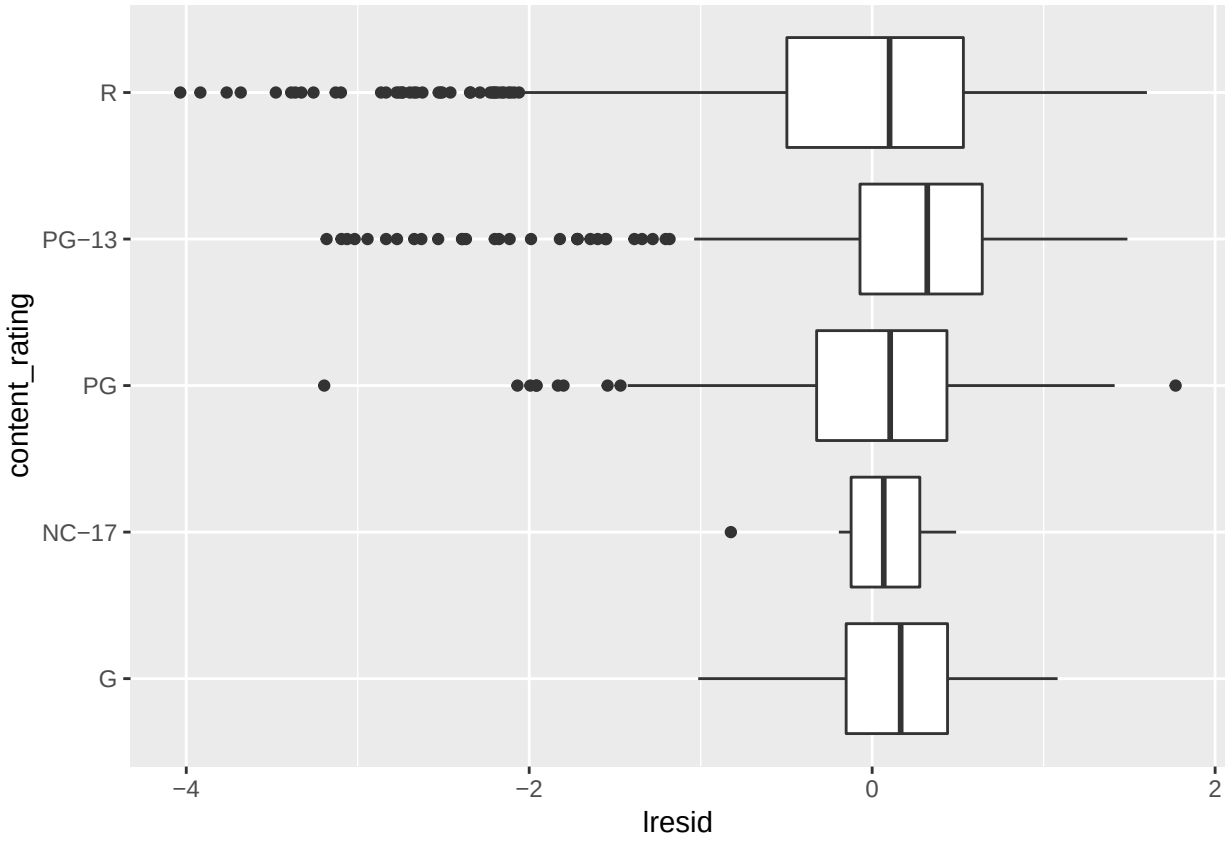



```
##
## [[4]]
```

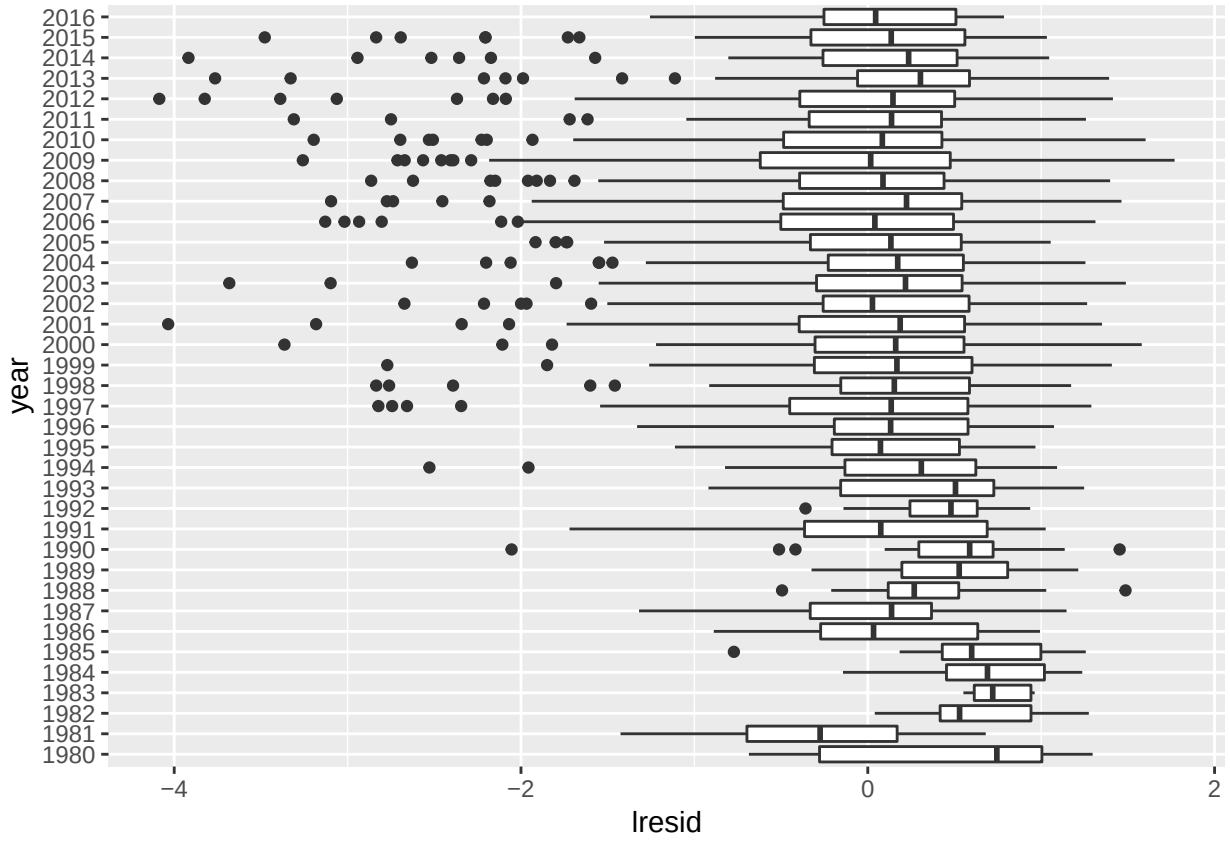



```
# categorical
# can't log categorical variables
lapply(c('content_rating', 'year'), function(var) {
  train_resid %>%
    filter(!is.na(!!rlang::sym(var))) %>%
    ggplot() +
    geom_boxplot(aes_string(var, 'lresid')) +
    coord_flip()
})
```

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




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



Stepwise: Genre as Base

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.

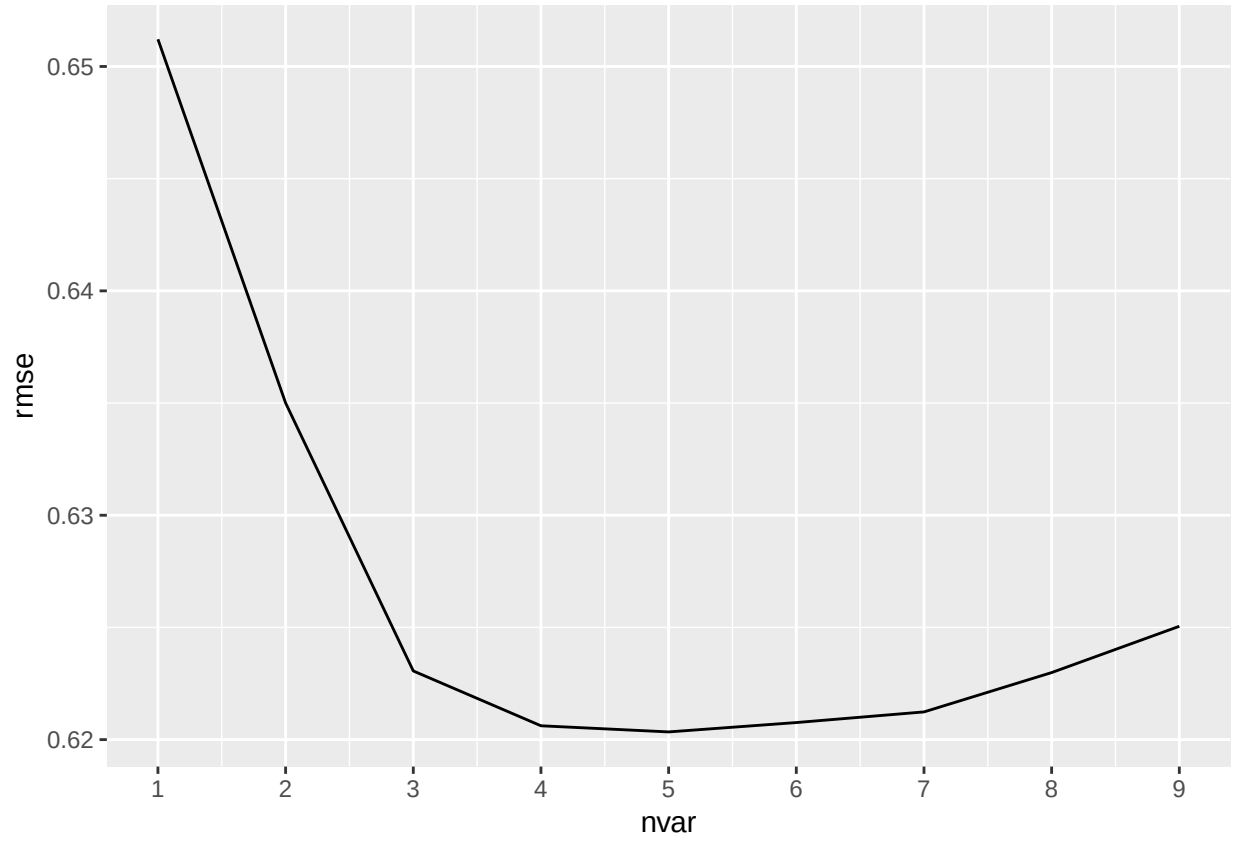
For factor variables (content_rating, total_oscars) use normal versions of variables.
For facebook likes, use log versions as those were more linear with log(real_gross).
For budget and IMDB score, I think log versions are better, but try the non-log versions too. Both had some linearity.
For year, use 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'), funs(log = log10(.))) %>%
  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'), funs(log = log10(.))) %>%
  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
##       0.6512142
## [1] 1
## imdb_score_log
##      0.6350029
## [1] 2
##      year
## 0.6230605
## [1] 3
## content_rating
##      0.6206177
## [1] 4
## total_oscars_director
##             0.6203438
## [1] 5
## cast_total_facebook_likes_log
##                      0.6207615
## [1] 6
## real_budget
##   0.6212351
## [1] 7
## total_oscars_actor
##          0.6229884
## [1] 8
## director_facebook_likes_log
##                   0.6250515
```

```
# 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
## -3.4485 -0.2238  0.0878  0.3407  3.3377
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.016486   0.394298  -0.042  0.96665
## Adventure1          -0.142890   0.045332  -3.152  0.00165 **
## Action1             -0.011306   0.040640  -0.278  0.78090
## Family1              0.311938   0.079689   3.914 9.43e-05 ***
```

```
## Mystery1             0.042585   0.051695   0.824  0.41019
## Documentary1         0.163354   0.141606   1.154  0.24884
## Drama1              -0.257951   0.034509  -7.475 1.24e-13 ***
## History1            -0.017739   0.089809  -0.198  0.84345
## Romance1             0.015177   0.037044   0.410  0.68208
## real_budget_log      0.812038   0.029073  27.931  < 2e-16 ***
## imdb_score_log       2.160658   0.206270  10.475  < 2e-16 ***
## year1981            -0.209854   0.368926  -0.569  0.56955
## year1982             0.182927   0.339404   0.539  0.58998
## year1983             0.247796   0.384325   0.645  0.51917
## year1984             0.500267   0.329092   1.520  0.12866
## year1985             0.288020   0.339707   0.848  0.39665
## year1986             0.065028   0.324926   0.200  0.84140
## year1987             0.172547   0.316476   0.545  0.58568
## year1988             0.147460   0.310528   0.475  0.63494
## year1989             0.279267   0.305019   0.916  0.36002
## year1990             0.062553   0.306795   0.204  0.83846
## year1991             0.044401   0.310206   0.143  0.88620
## year1992            -0.007279   0.315891  -0.023  0.98162
## year1993             0.025560   0.307311   0.083  0.93372
## year1994            -0.222206   0.301314  -0.737  0.46095
## year1995            -0.050133   0.293633  -0.171  0.86445
## year1996            -0.217635   0.286206  -0.760  0.44712
## year1997            -0.171164   0.286195  -0.598  0.54988
## year1998            -0.272722   0.285619  -0.955  0.33980
## year1999            -0.270022   0.282260  -0.957  0.33889
## year2000            -0.173448   0.283525  -0.612  0.54078
## year2001            -0.287595   0.281570  -1.021  0.30721
## year2002            -0.262480   0.281560  -0.932  0.35135
## year2003            -0.250251   0.282916  -0.885  0.37653
## year2004            -0.266357   0.282475  -0.943  0.34585
## year2005            -0.255937   0.281463  -0.909  0.36332
## year2006            -0.346753   0.281963  -1.230  0.21895
## year2007            -0.347511   0.283316  -1.227  0.22015
## year2008            -0.374356   0.281500  -1.330  0.18375
## year2009            -0.371088   0.280547  -1.323  0.18611
## year2010            -0.402398   0.281138  -1.431  0.15253
## year2011            -0.279867   0.283485  -0.987  0.32367
## year2012            -0.184045   0.281991  -0.653  0.51406
## year2013            -0.120662   0.281346  -0.429  0.66807
## year2014            -0.154492   0.283282  -0.545  0.58558
## year2015            -0.267933   0.285375  -0.939  0.34793
## year2016            -0.176387   0.301510  -0.585  0.55862
## content_ratingNC-17 -0.036289   0.275191  -0.132  0.89510
## content_ratingPG    -0.035081   0.119686  -0.293  0.76948
## content_ratingPG-13  0.139695   0.137054   1.019  0.30822
## content_ratingR     -0.052452   0.136983  -0.383  0.70184
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6071 on 1668 degrees of freedom
##   (132 observations deleted due to missingness)
## Multiple R-squared:  0.495,  Adjusted R-squared:  0.4799
## F-statistic: 32.71 on 50 and 1668 DF,  p-value: < 2.2e-16
```

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

```
## [1] 0.6206177
```

```r
# when consider the factors as one variable, they are significant
anova(mod_all)
```

```
## Analysis of Variance Table
##
## Response: real_gross_log
##                    Df Sum Sq Mean Sq  F value    Pr(>F)
## Adventure           1  75.64   75.64 205.2075 < 2.2e-16 ***
## Action              1  24.10   24.10  65.3761 1.178e-15 ***
## Family              1  32.07   32.07  86.9917 < 2.2e-16 ***
## Mystery             1   2.40    2.40   6.5082   0.01083 *
## Documentary         1   7.70    7.70  20.8945 5.210e-06 ***
## Drama               1  14.60   14.60  39.6072 3.960e-10 ***
## History             1   5.95    5.95  16.1431 6.134e-05 ***
## Romance             1   1.86    1.86   5.0542   0.02470 *
## real_budget_log     1 347.64  347.64 943.0834 < 2.2e-16 ***
## imdb_score_log      1  46.38   46.38 125.8221 < 2.2e-16 ***
## year               36  33.63    0.93   2.5341 1.764e-06 ***
## content_rating      4  10.83    2.71   7.3440 7.330e-06 ***
## Residuals        1668 614.85    0.37
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# number of observations
  # lost about 150 observations to missings
nobs(mod_all)
```

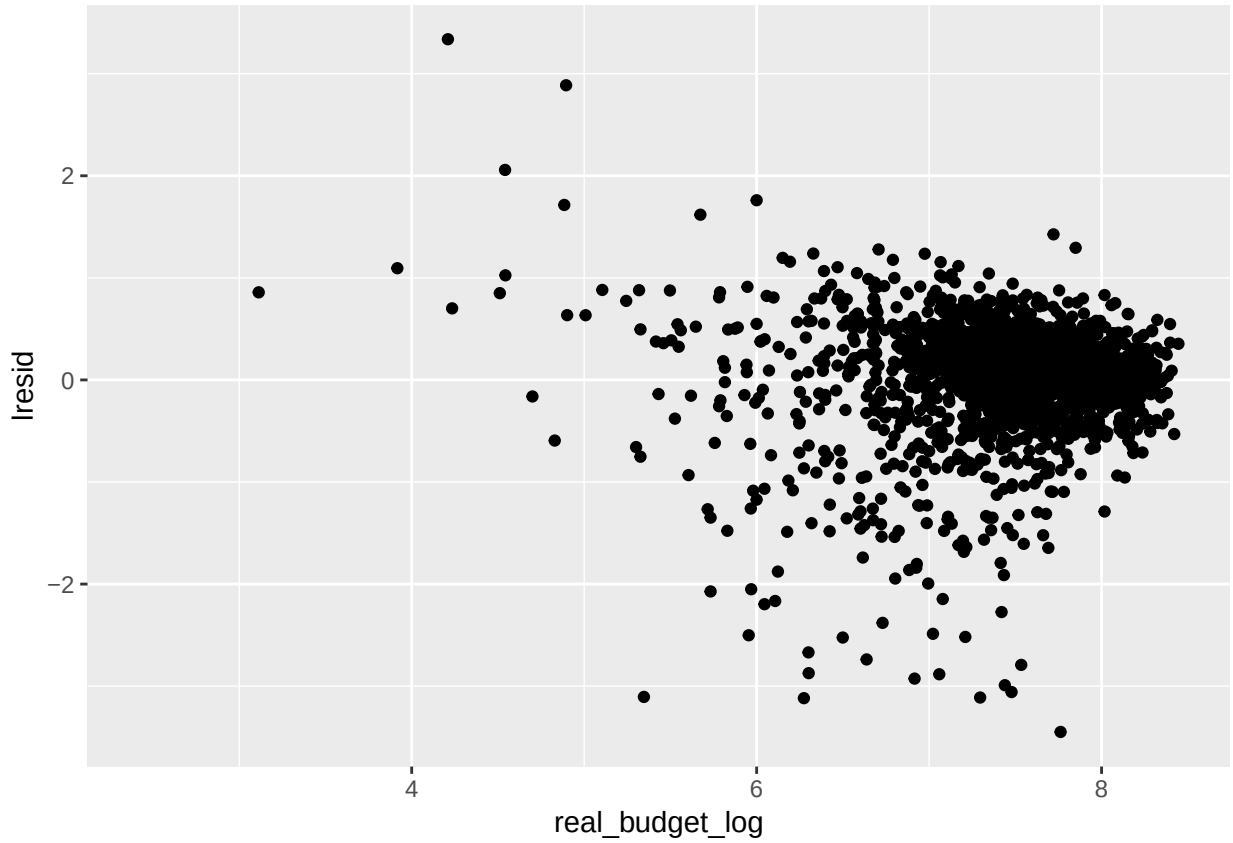
```
## [1] 1719
```

New Residuals

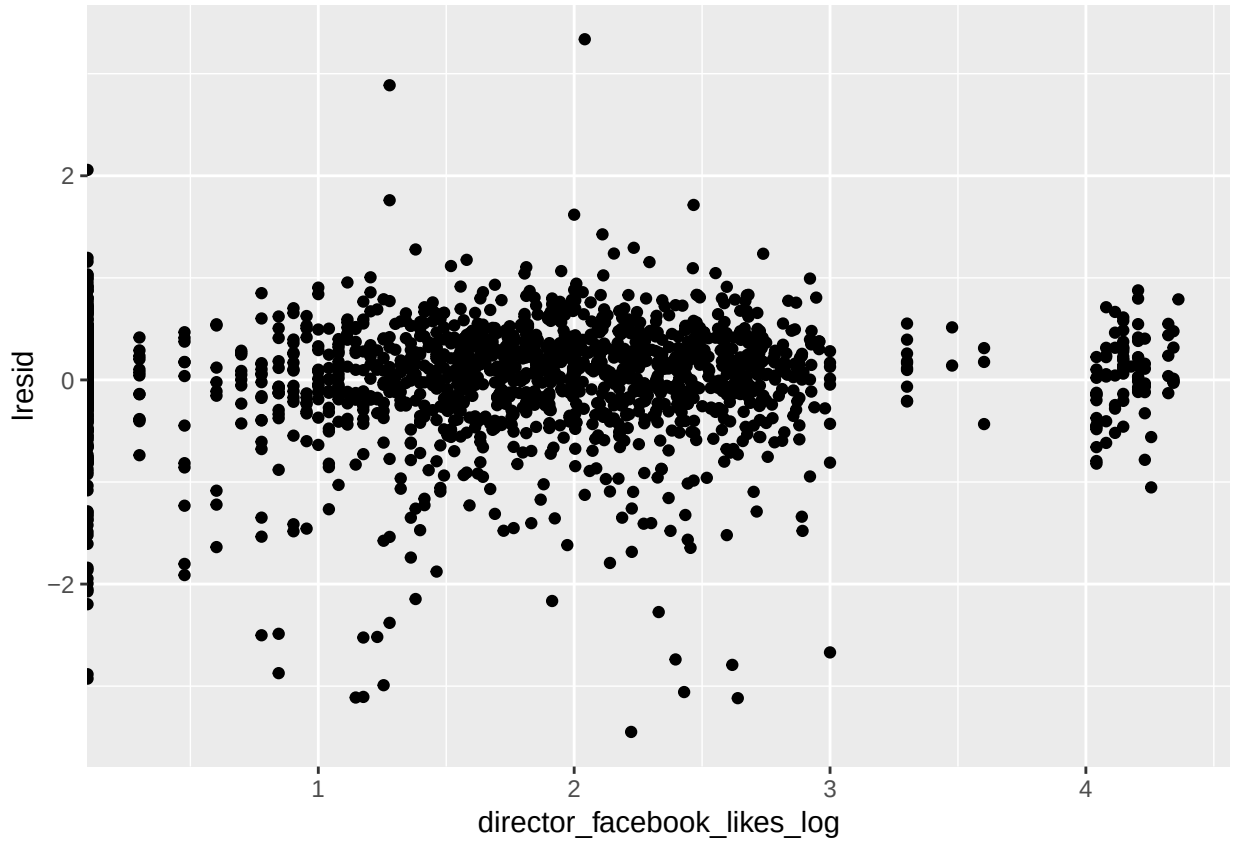
Graph residuals of included and excluded variables: have we captured all of the relationships?

```r
gr_resid(mod_all)
```

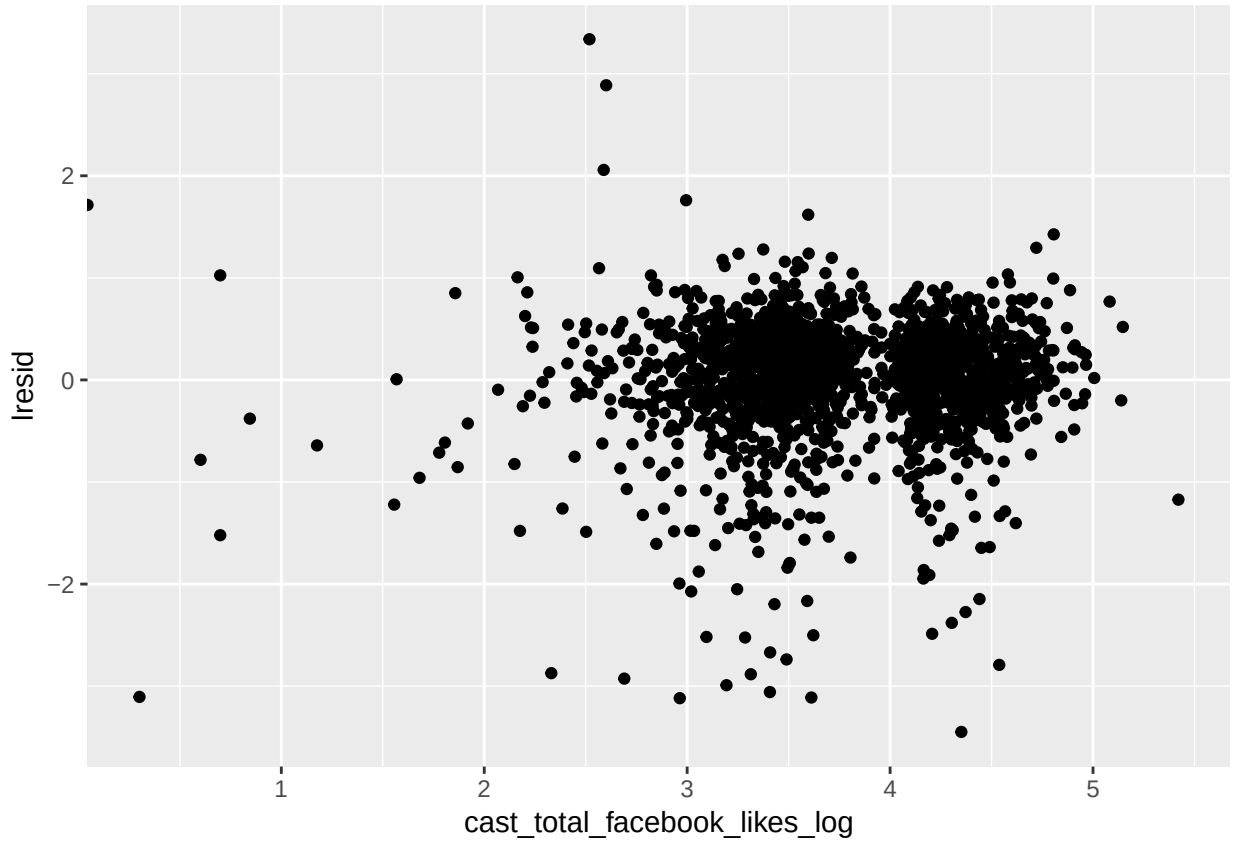
```
## Warning: Removed 132 rows containing missing values (geom_point).
```



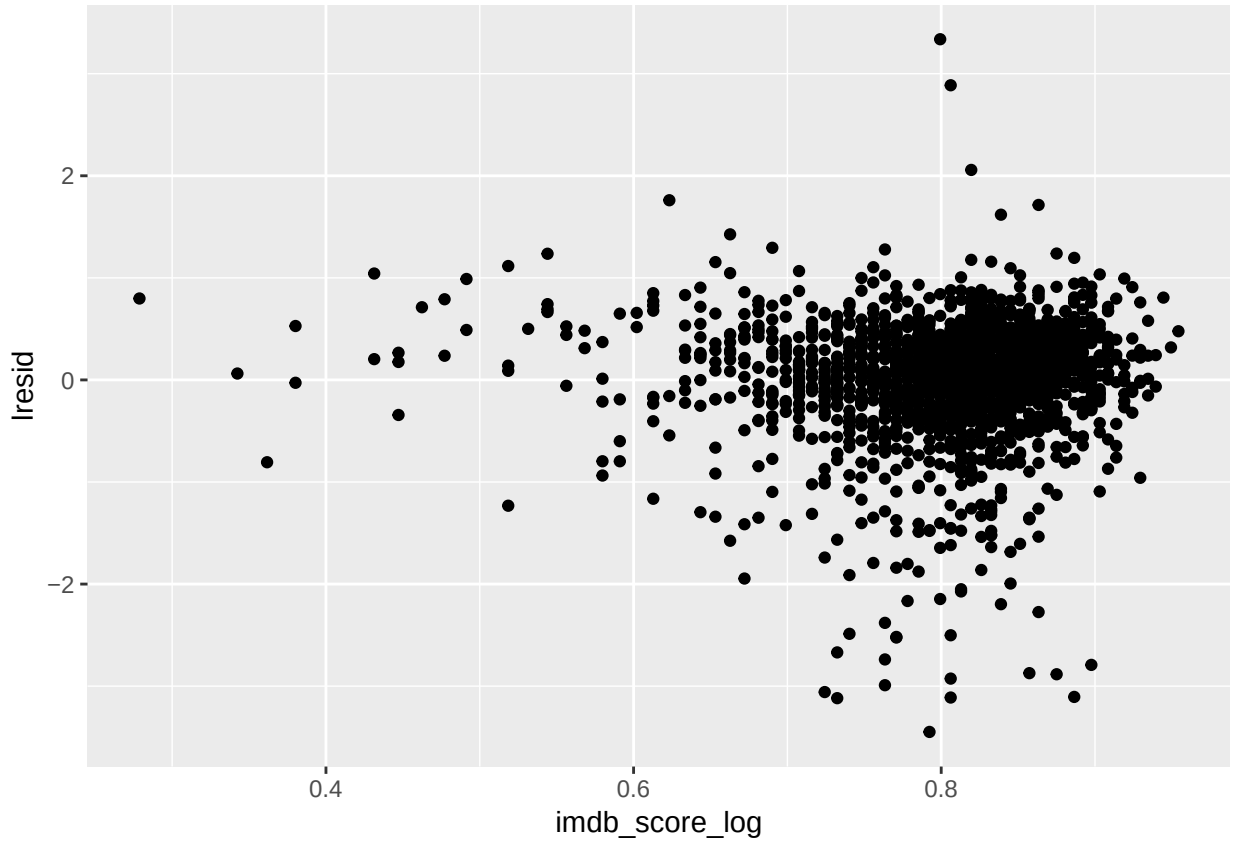

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



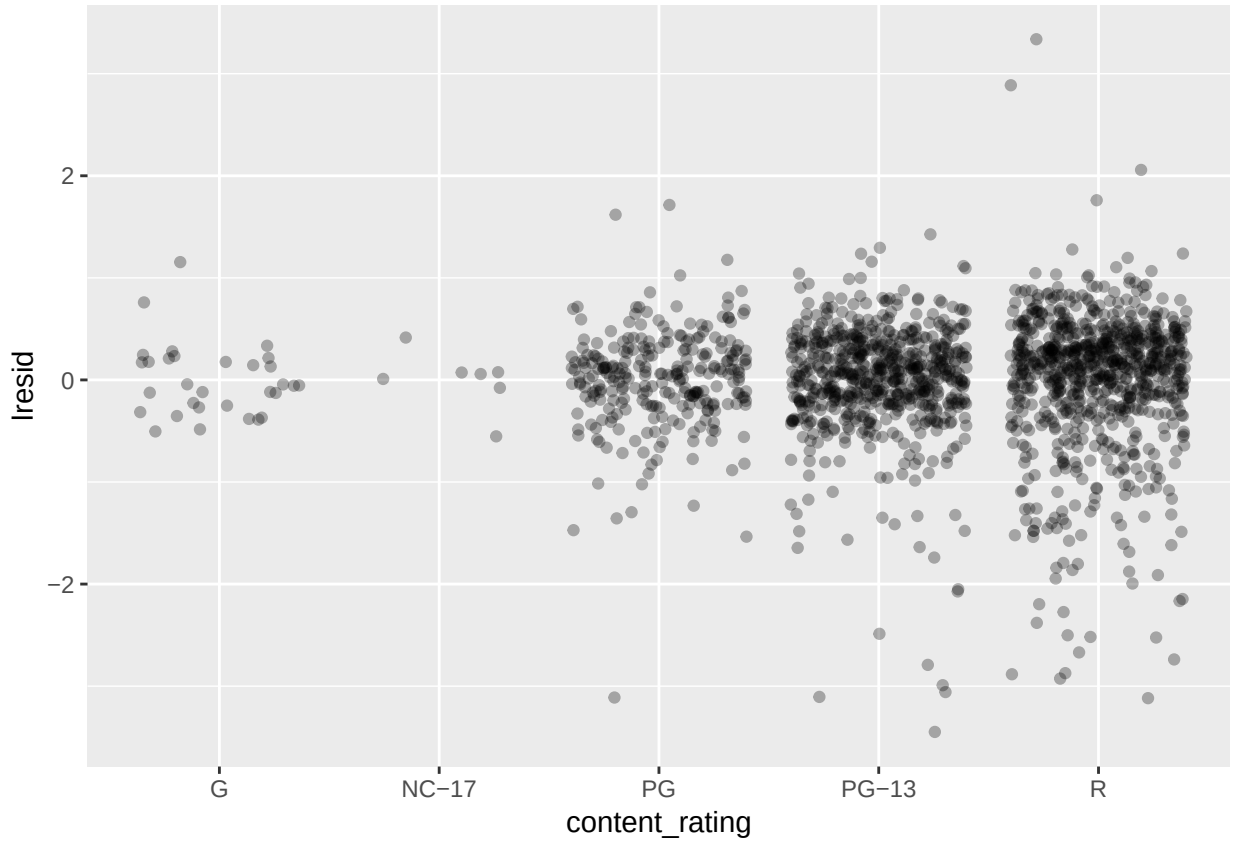

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



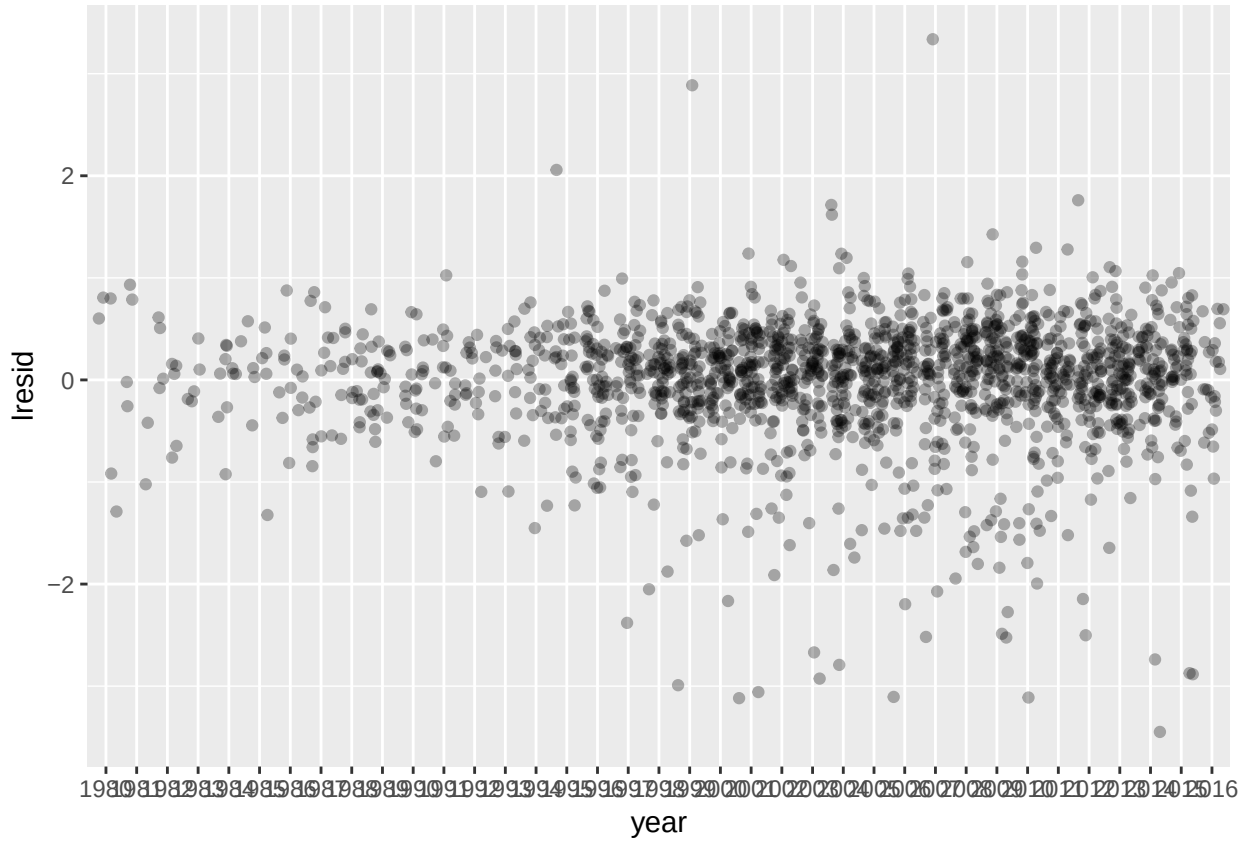

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



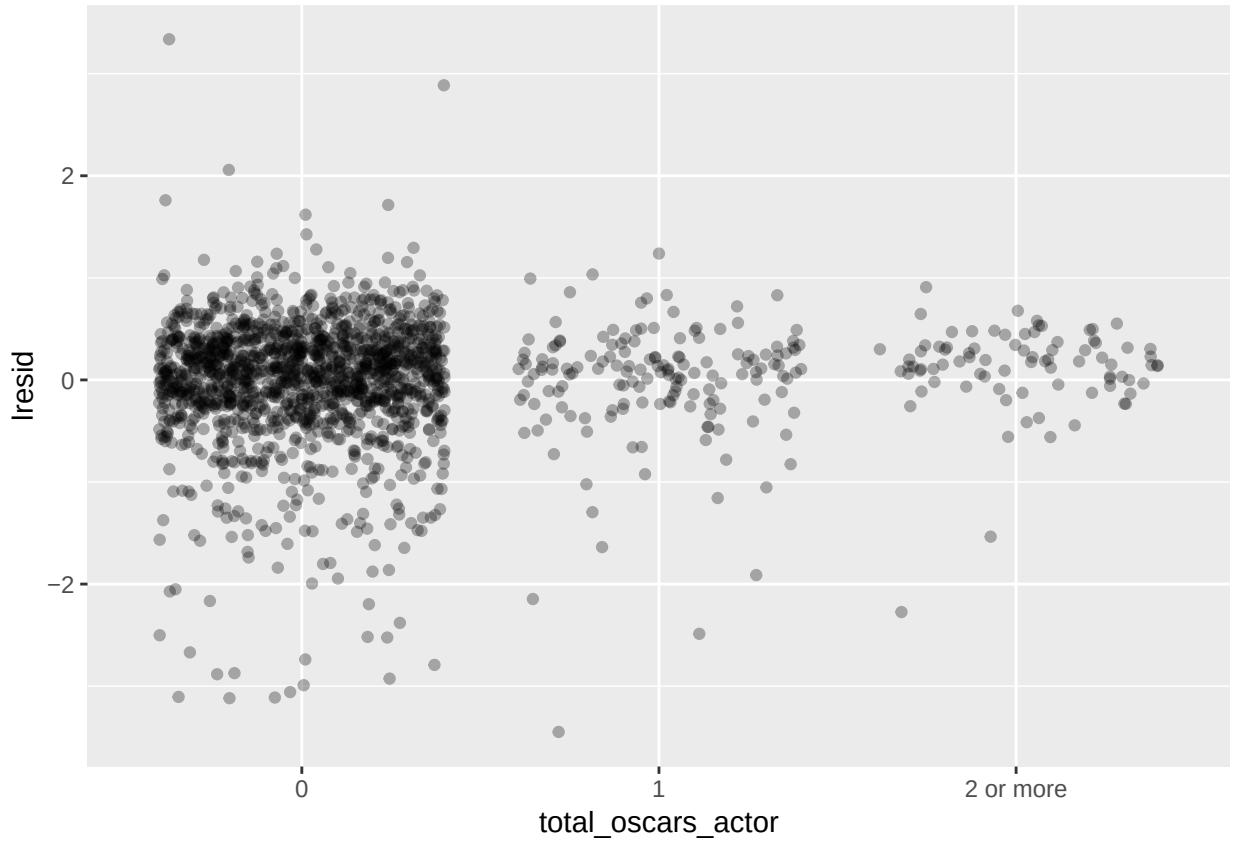

```
## Warning: Removed 94 rows containing missing values (geom_point).
```



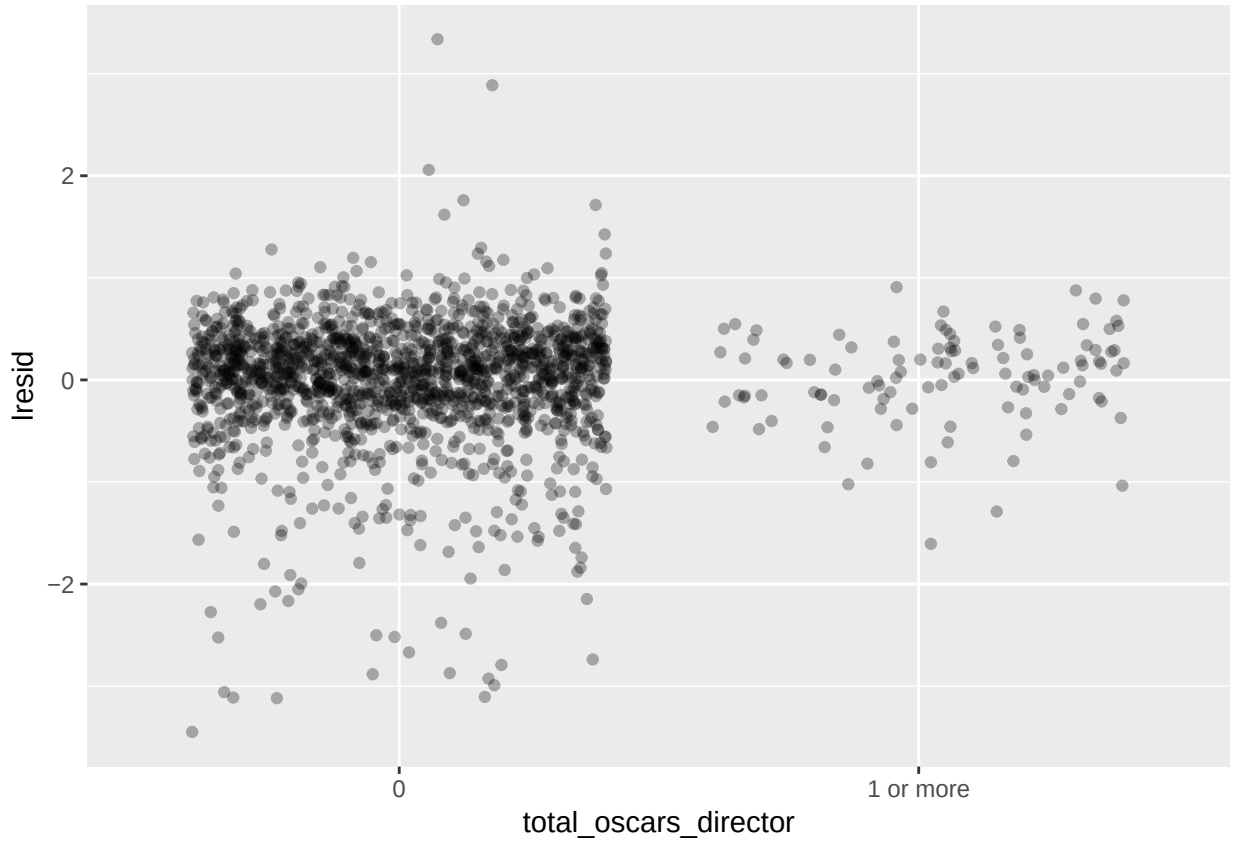

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



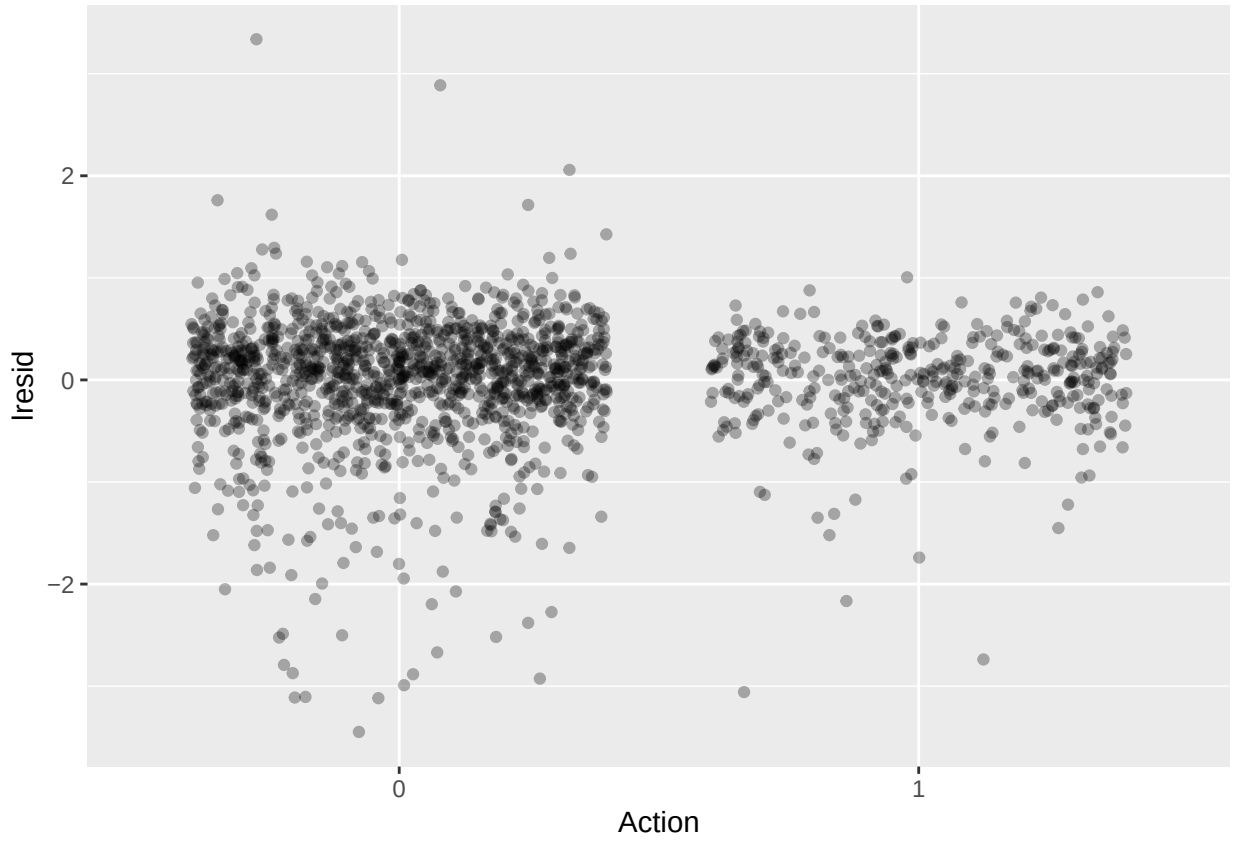

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



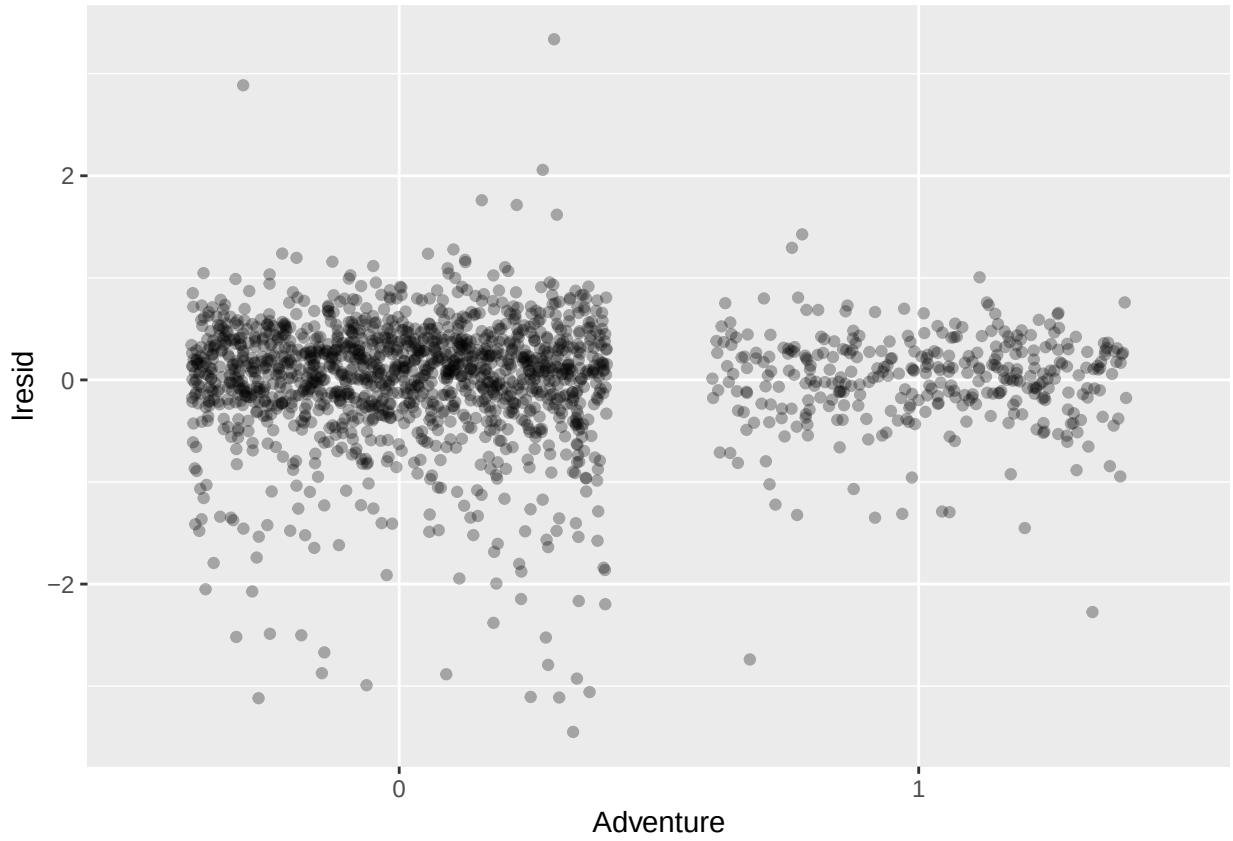

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



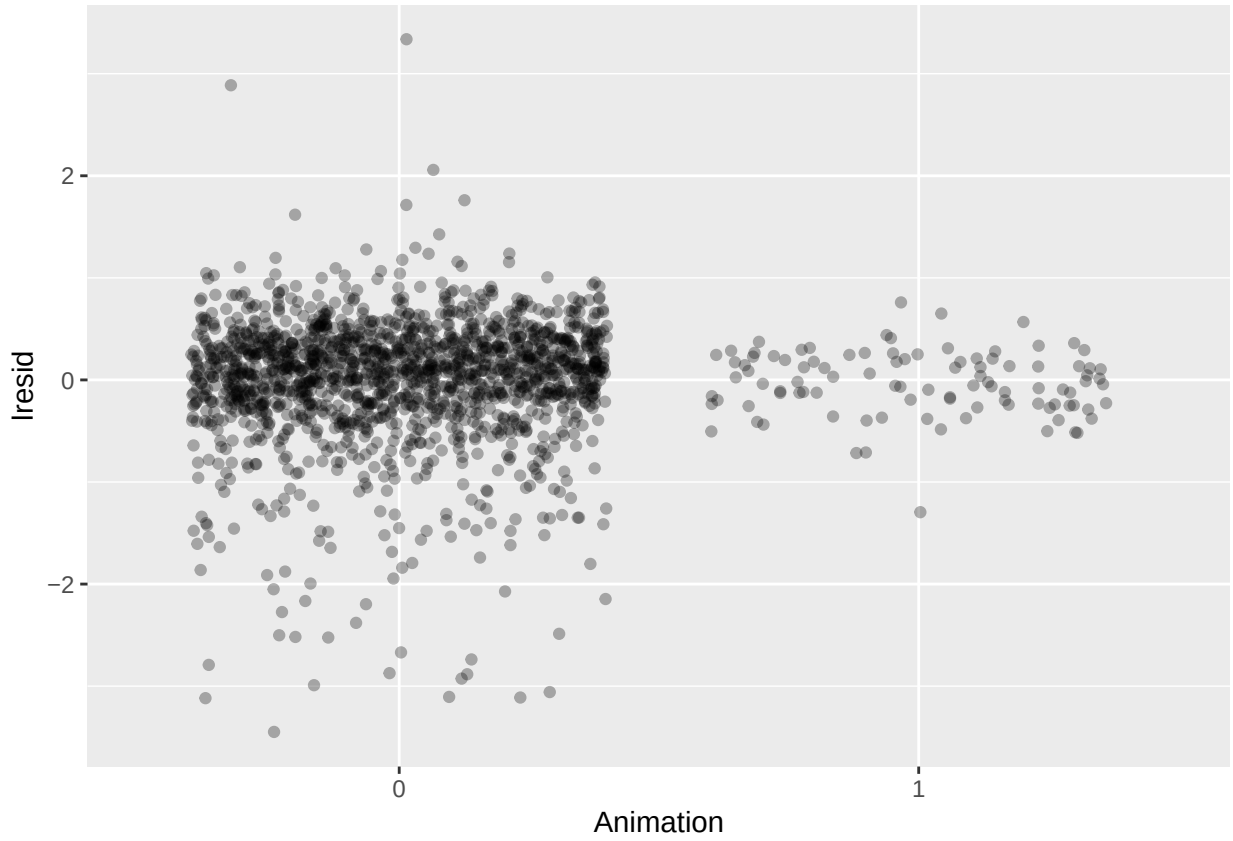

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



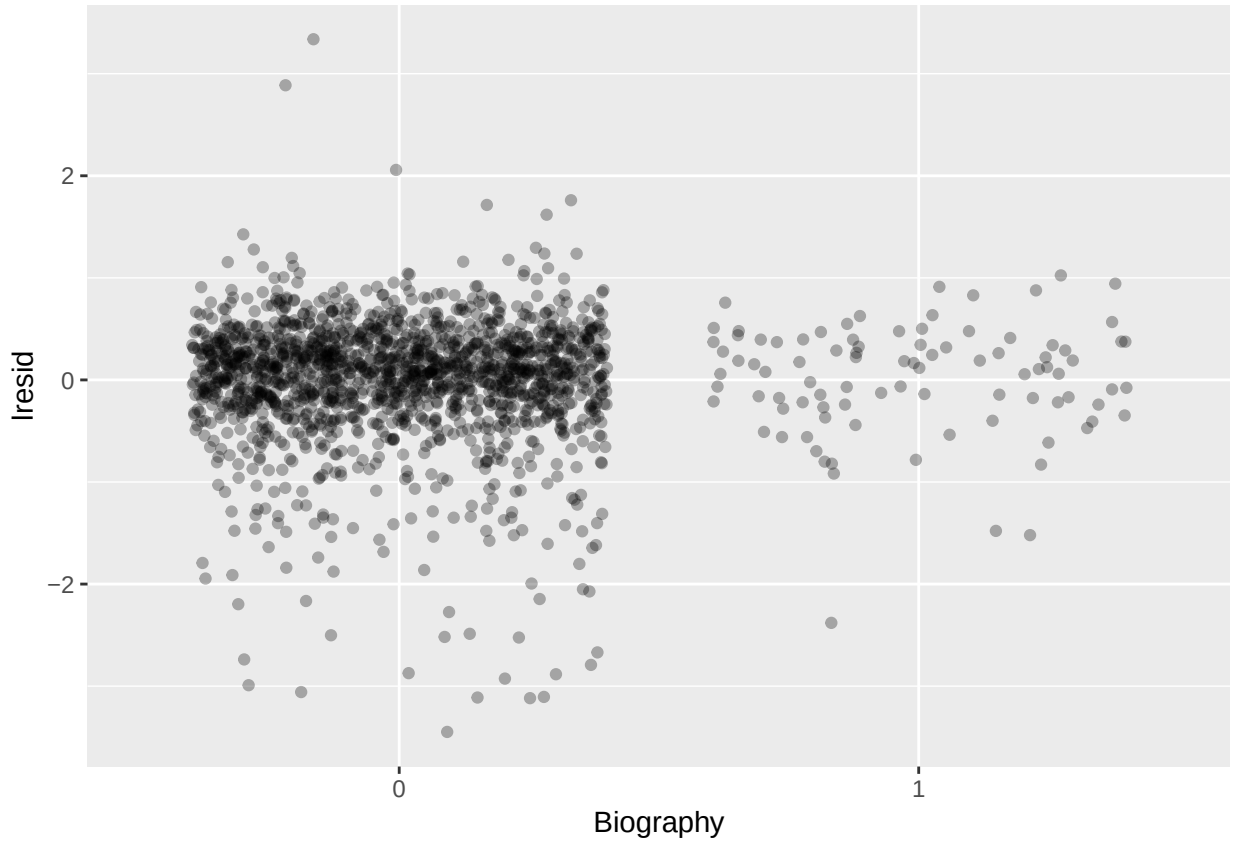

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



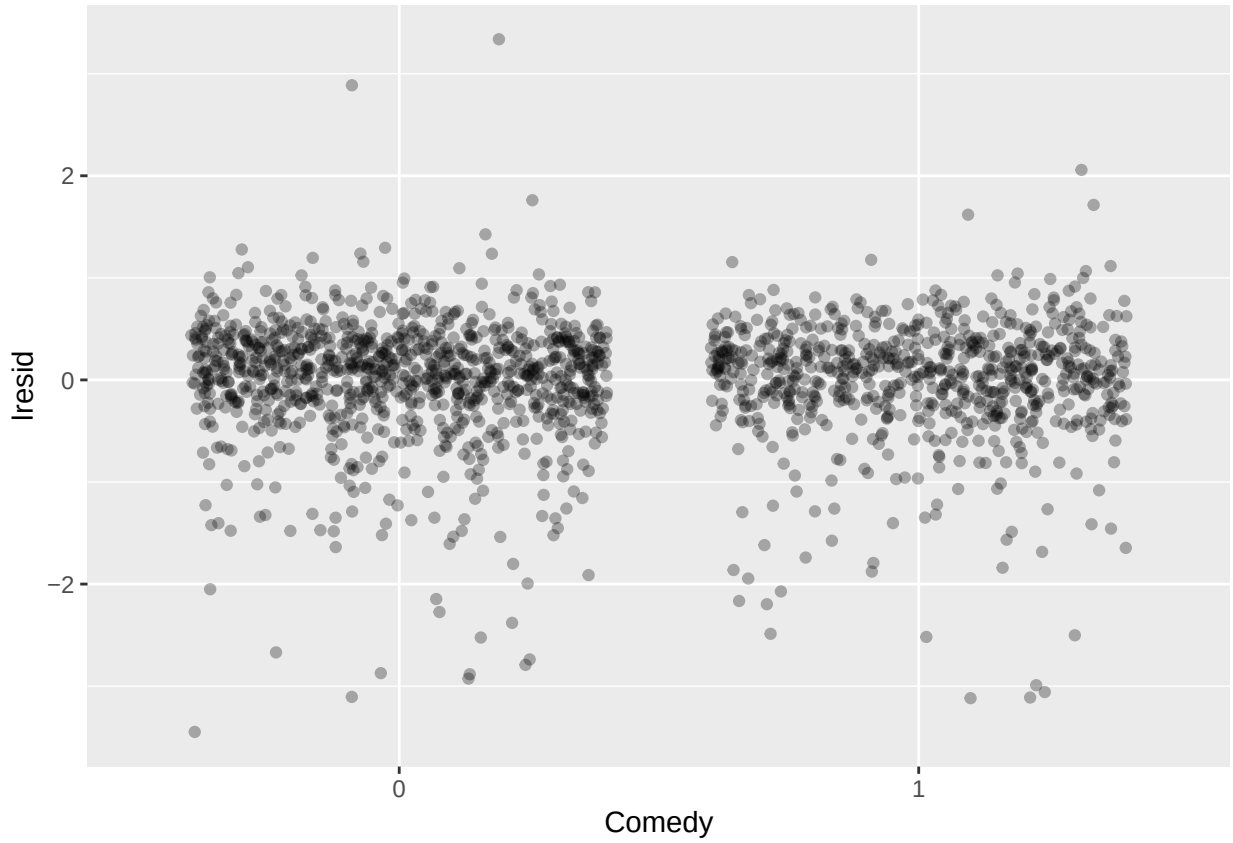

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



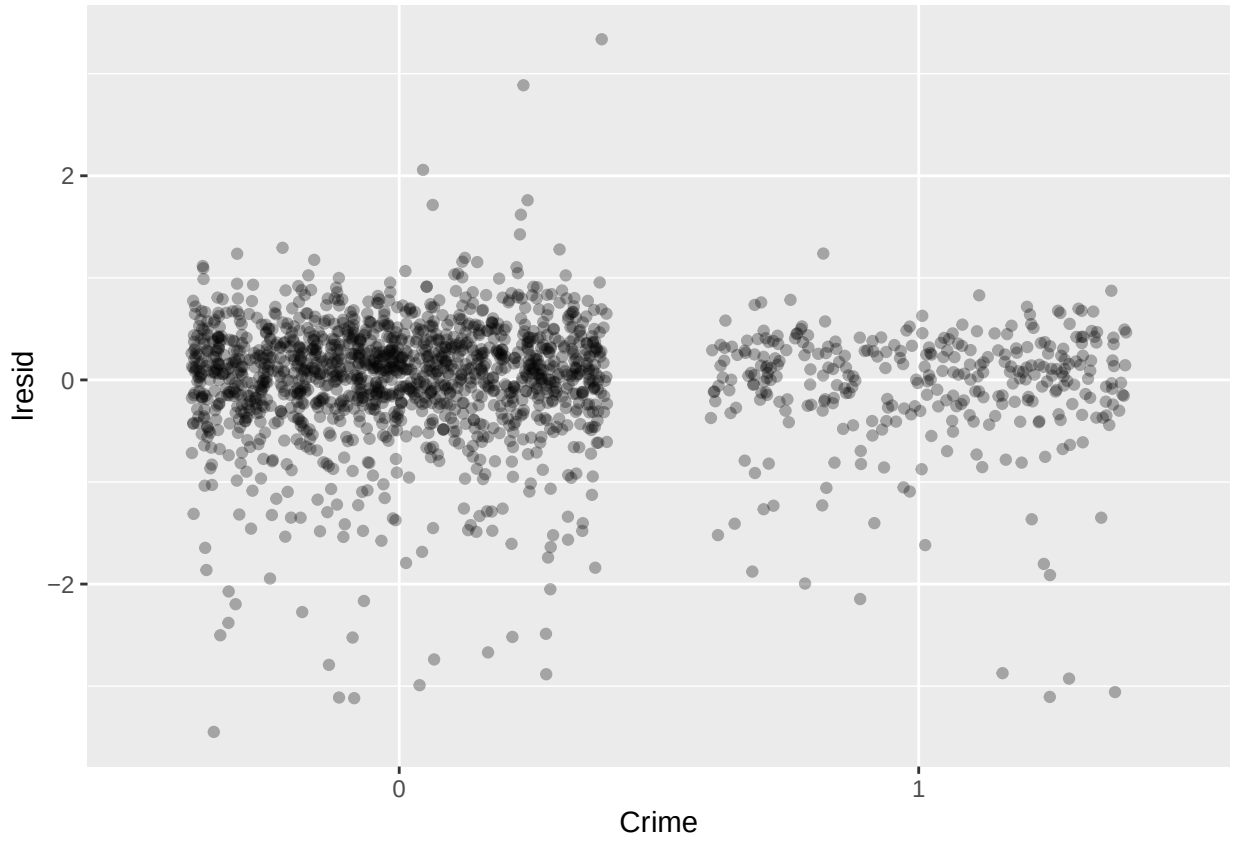

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



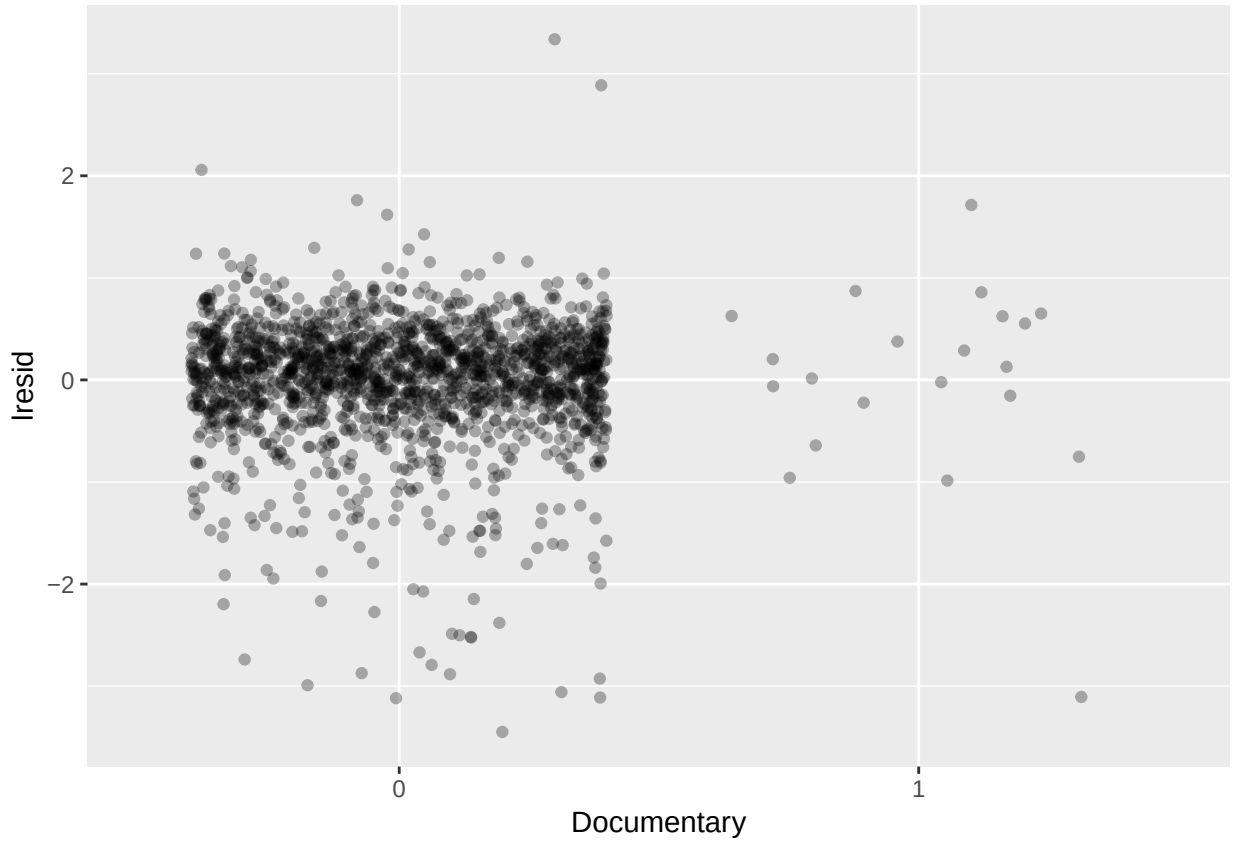

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



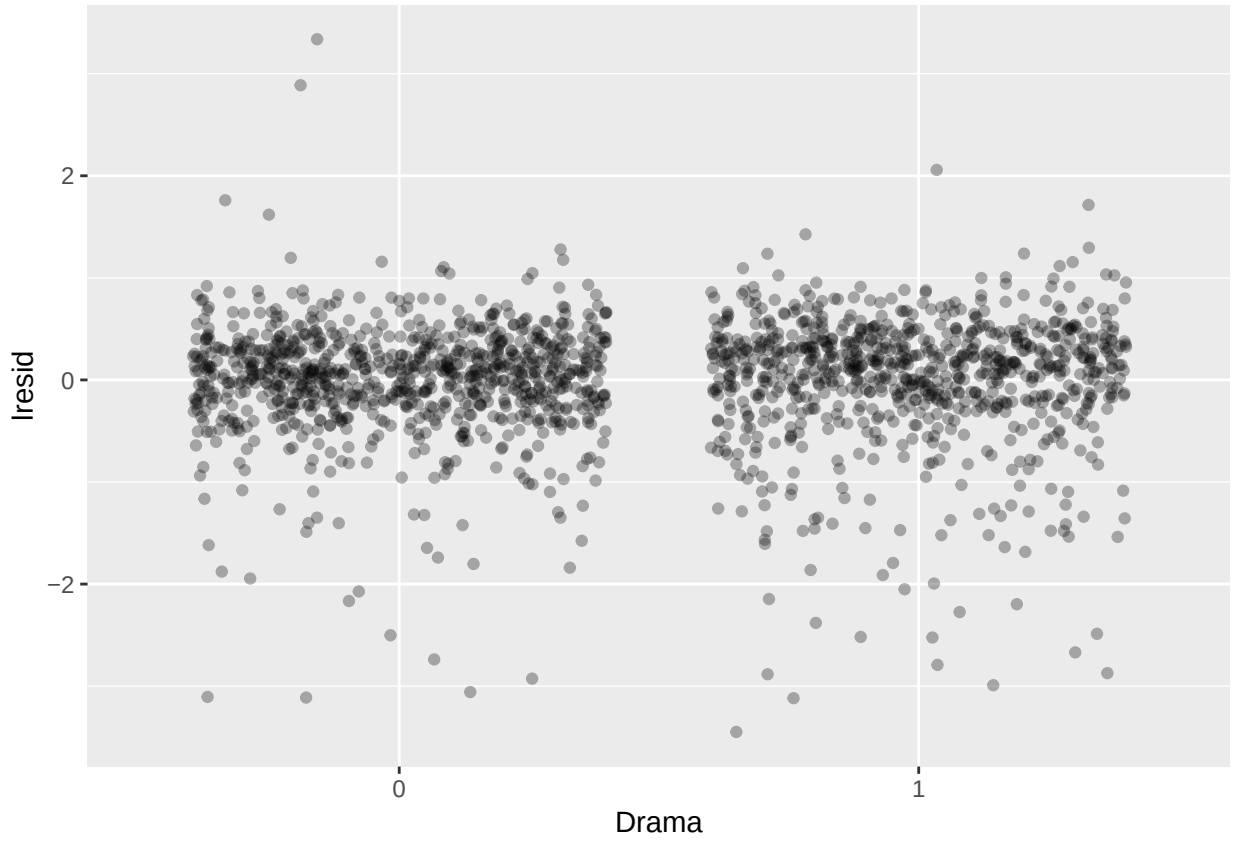

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



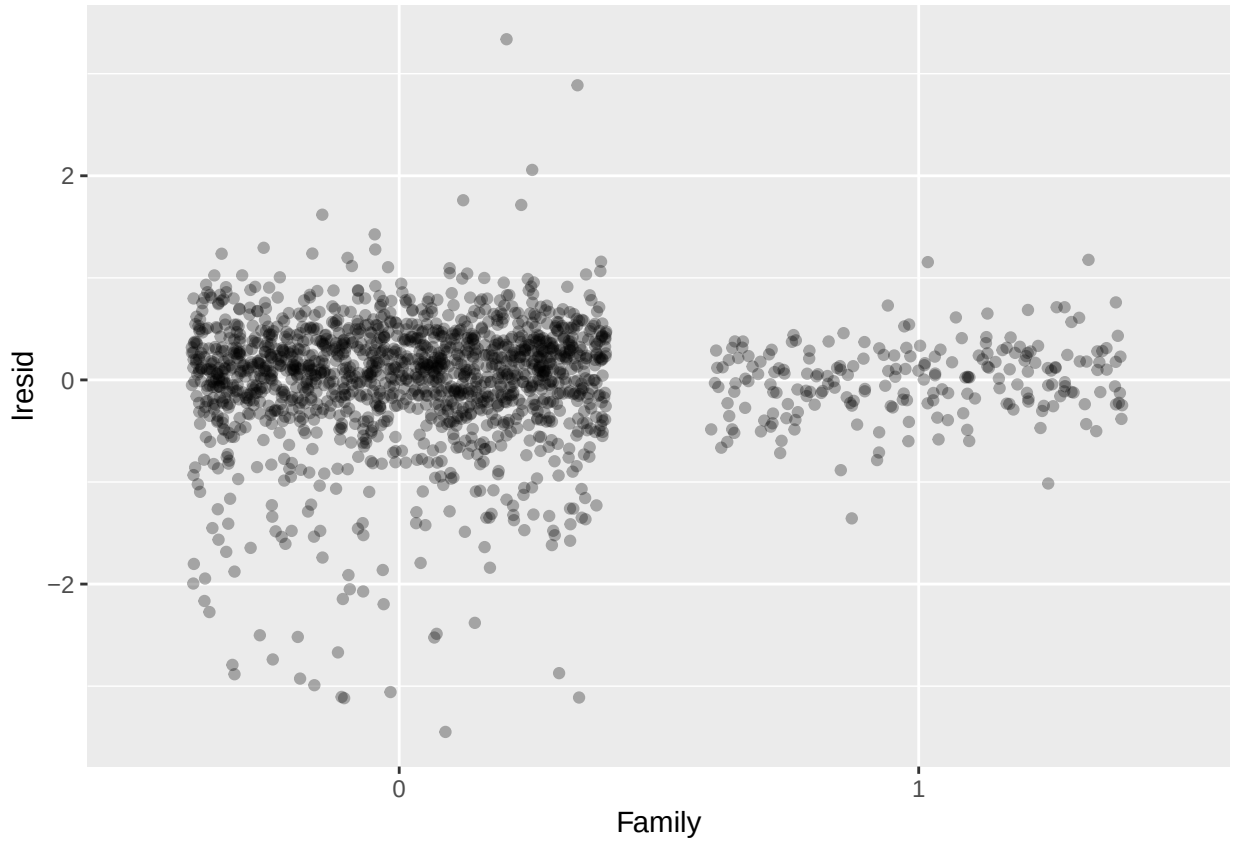

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



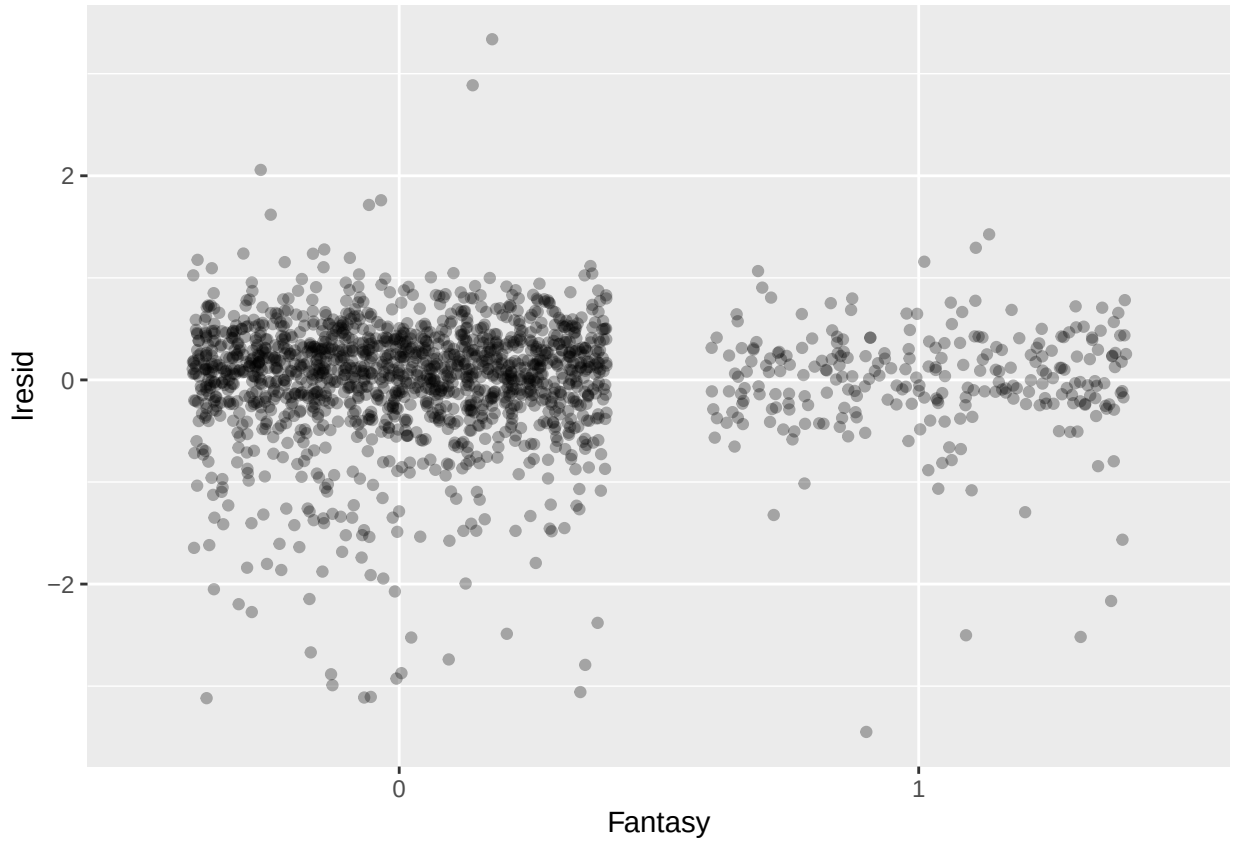

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



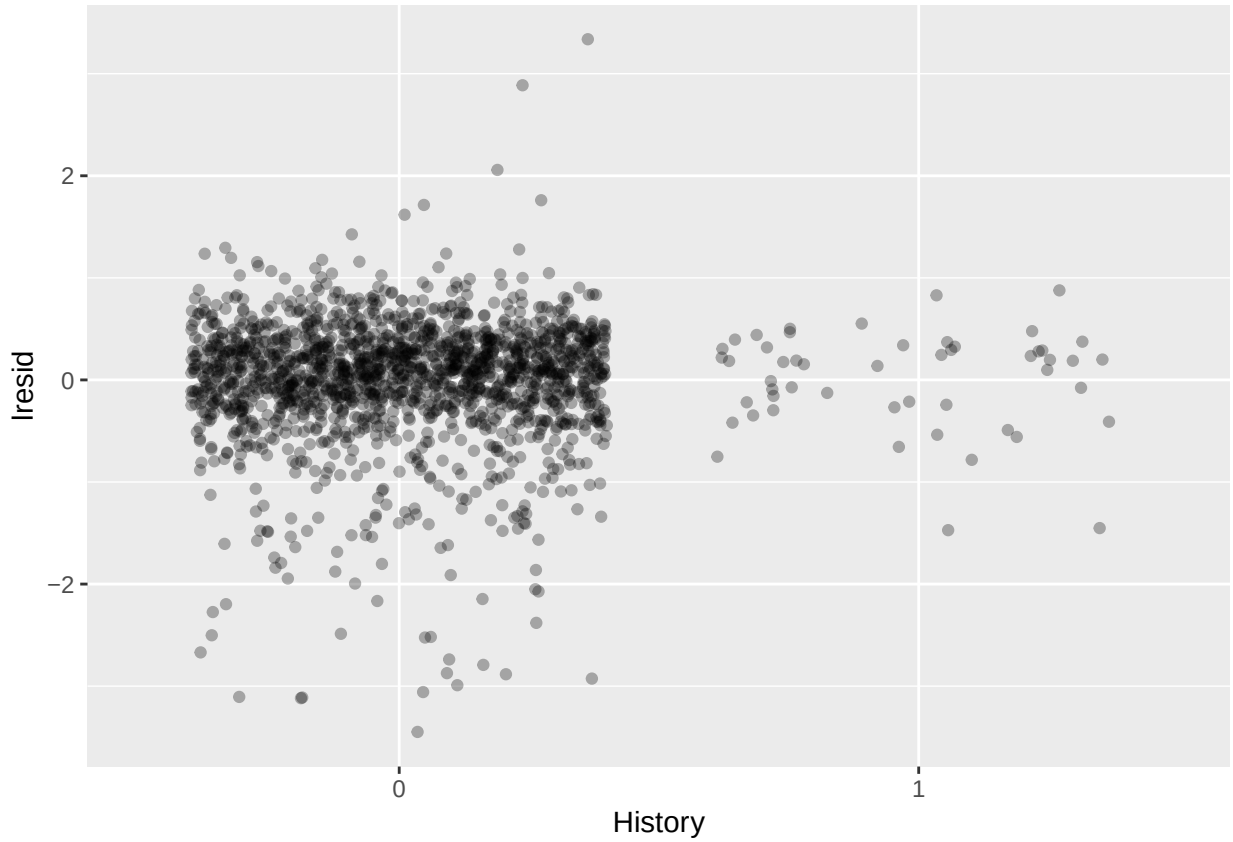

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



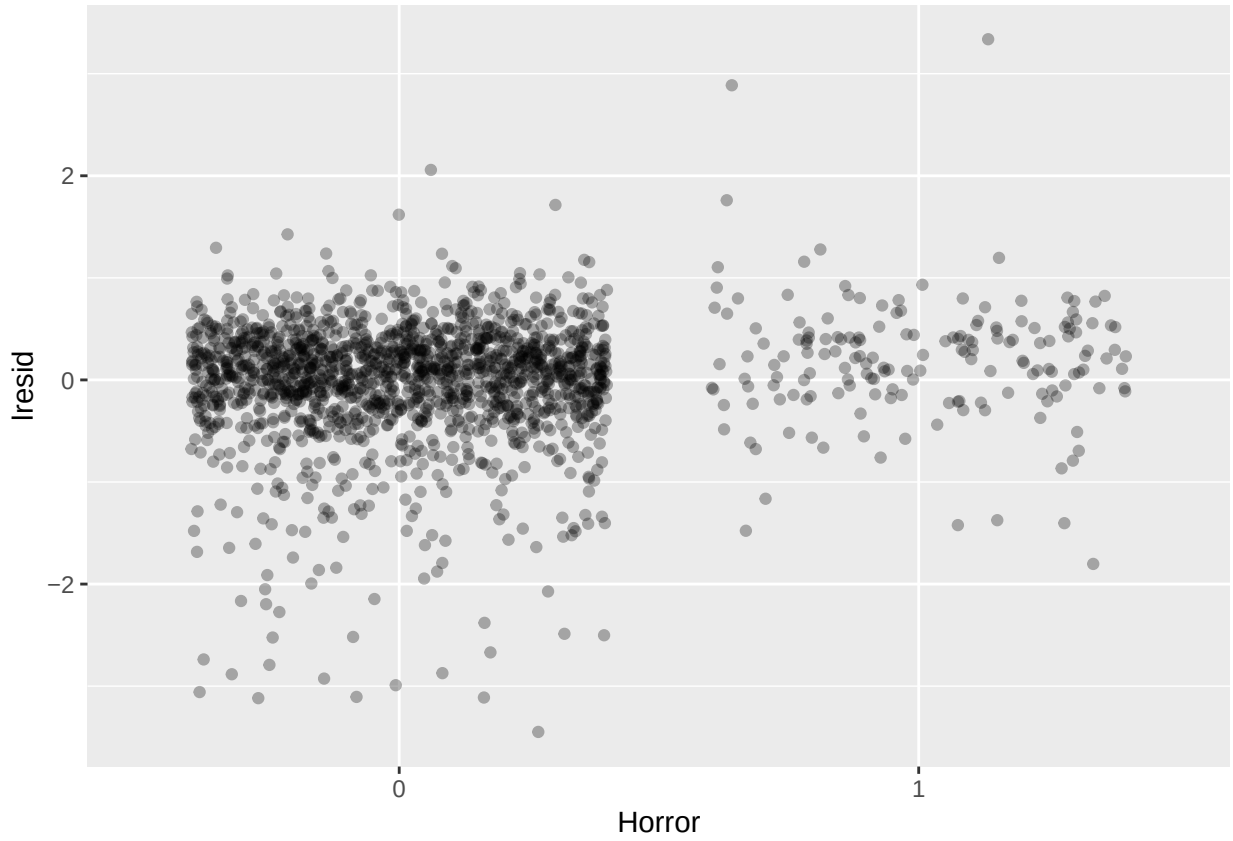

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



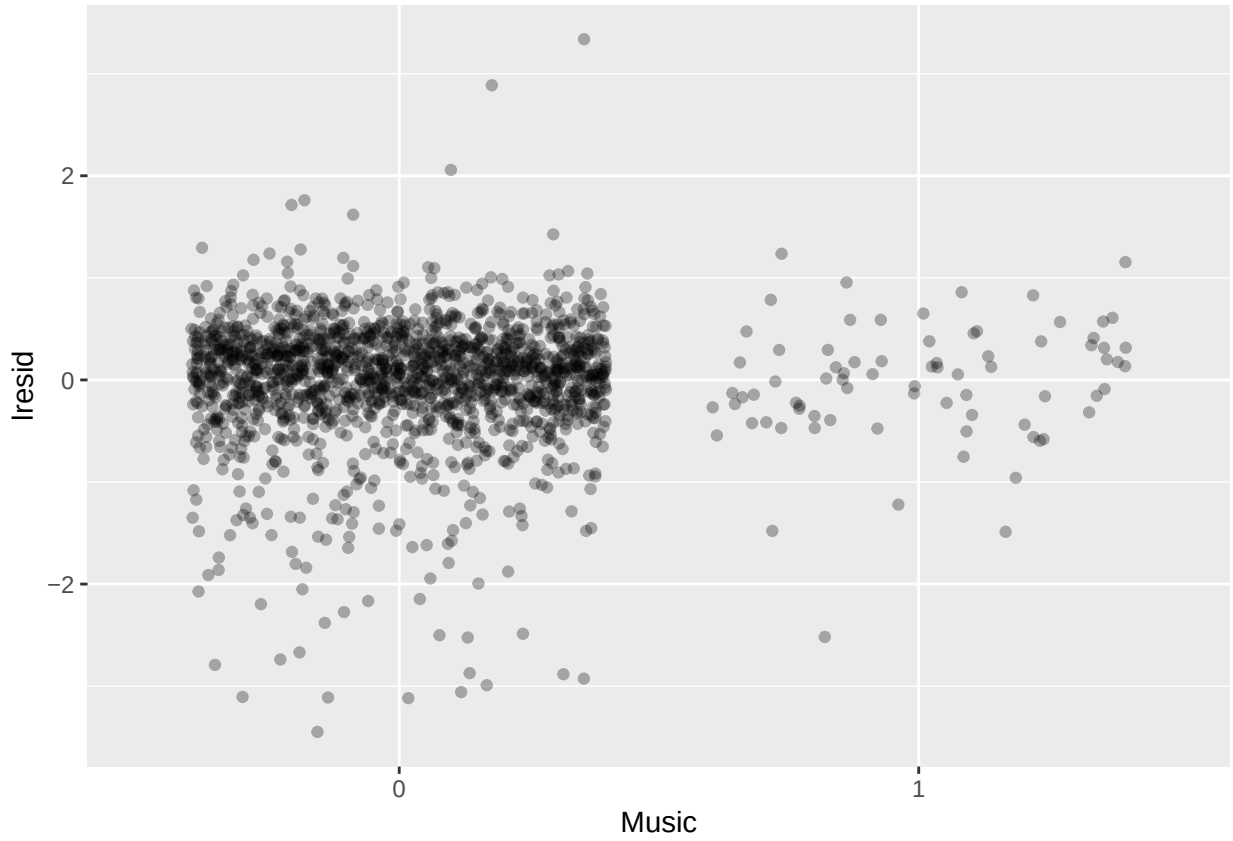

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



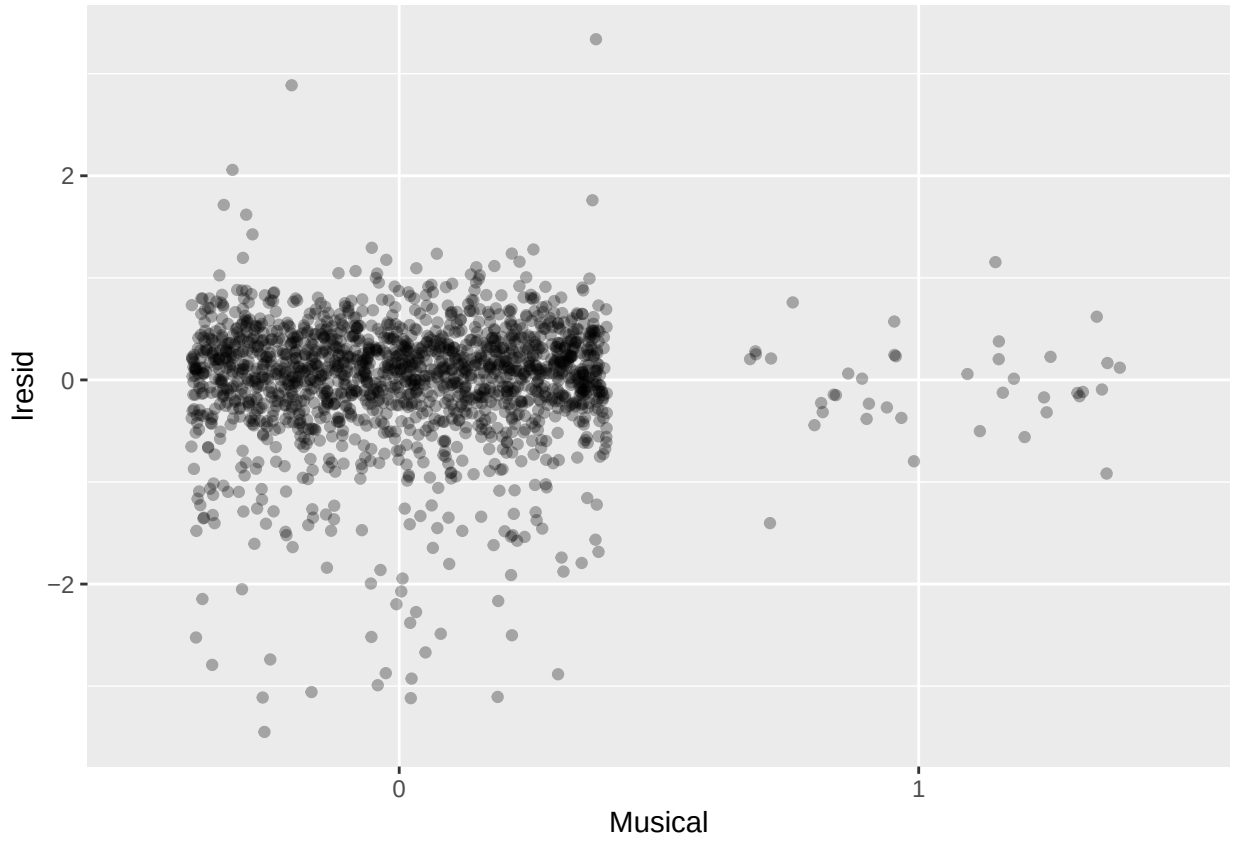

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



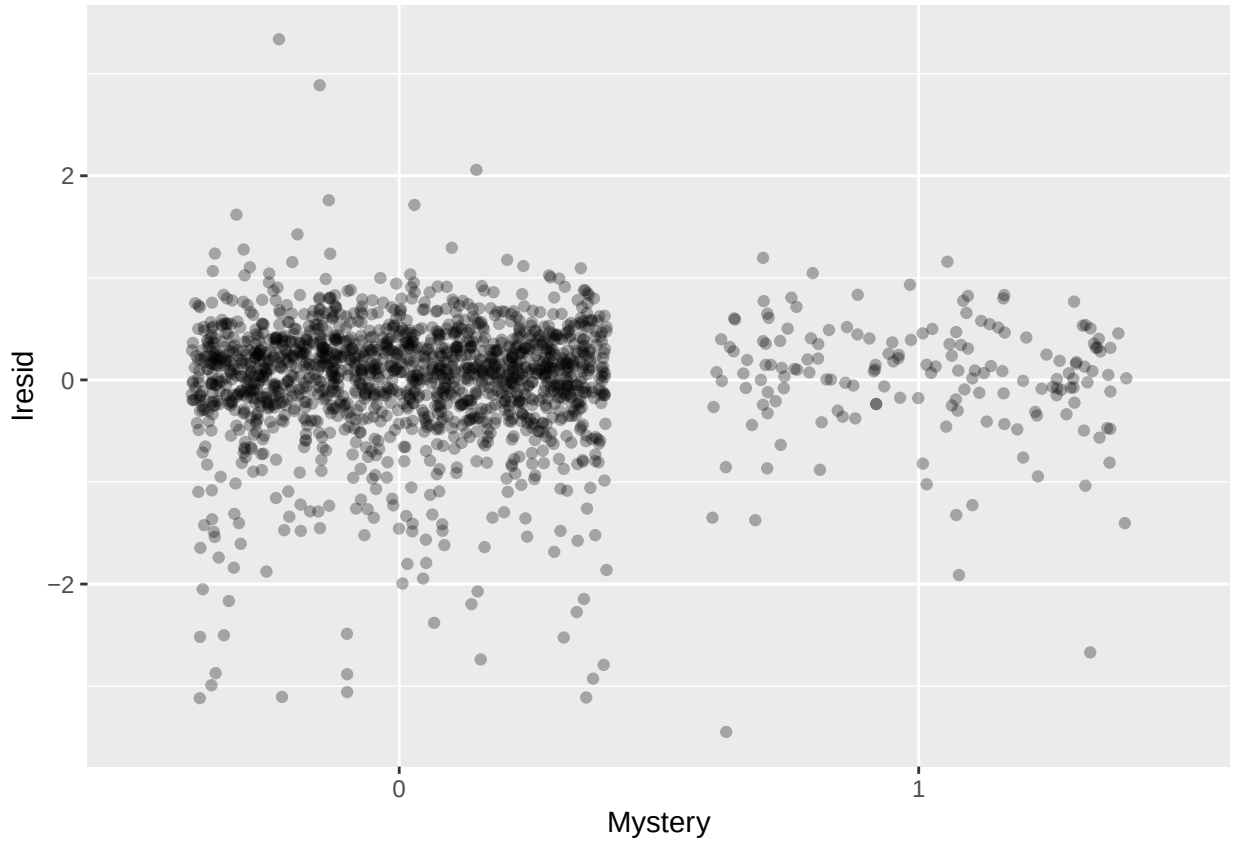

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



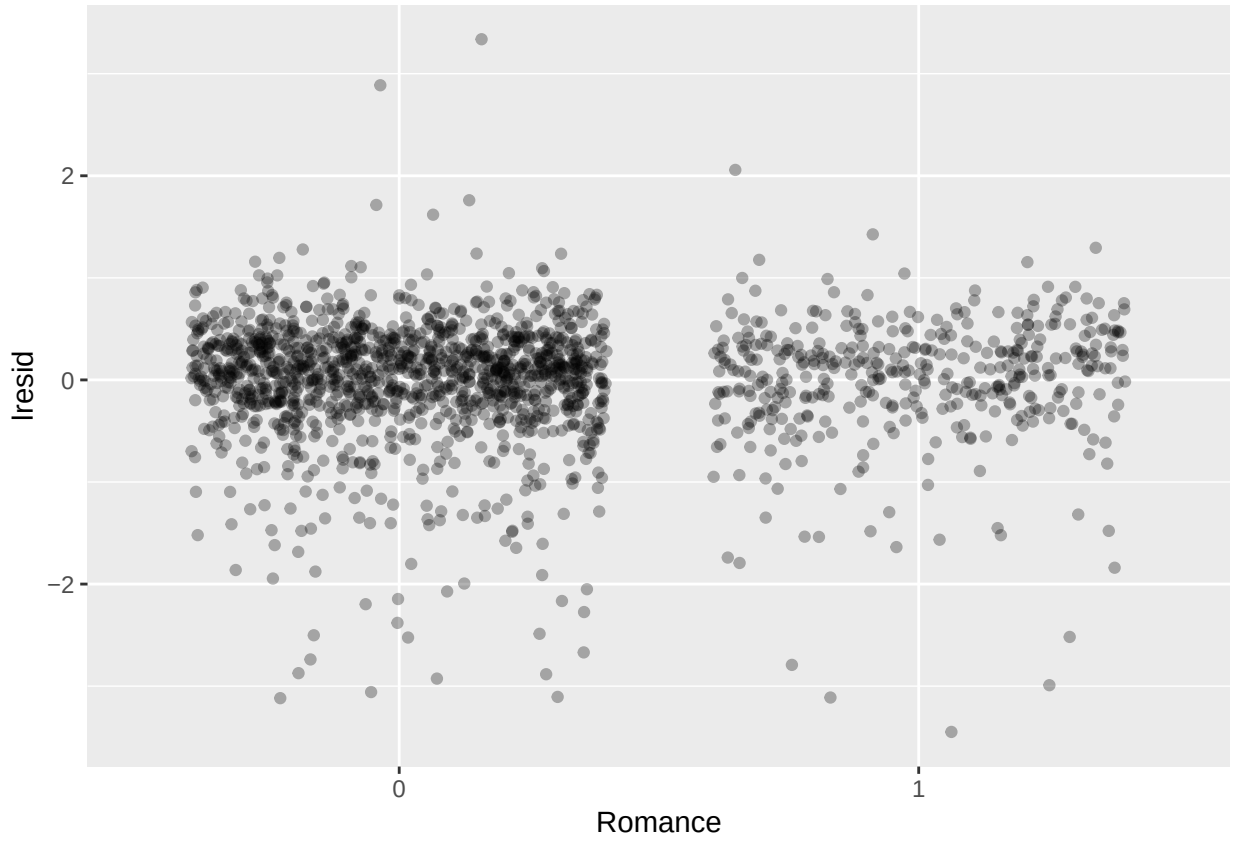

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



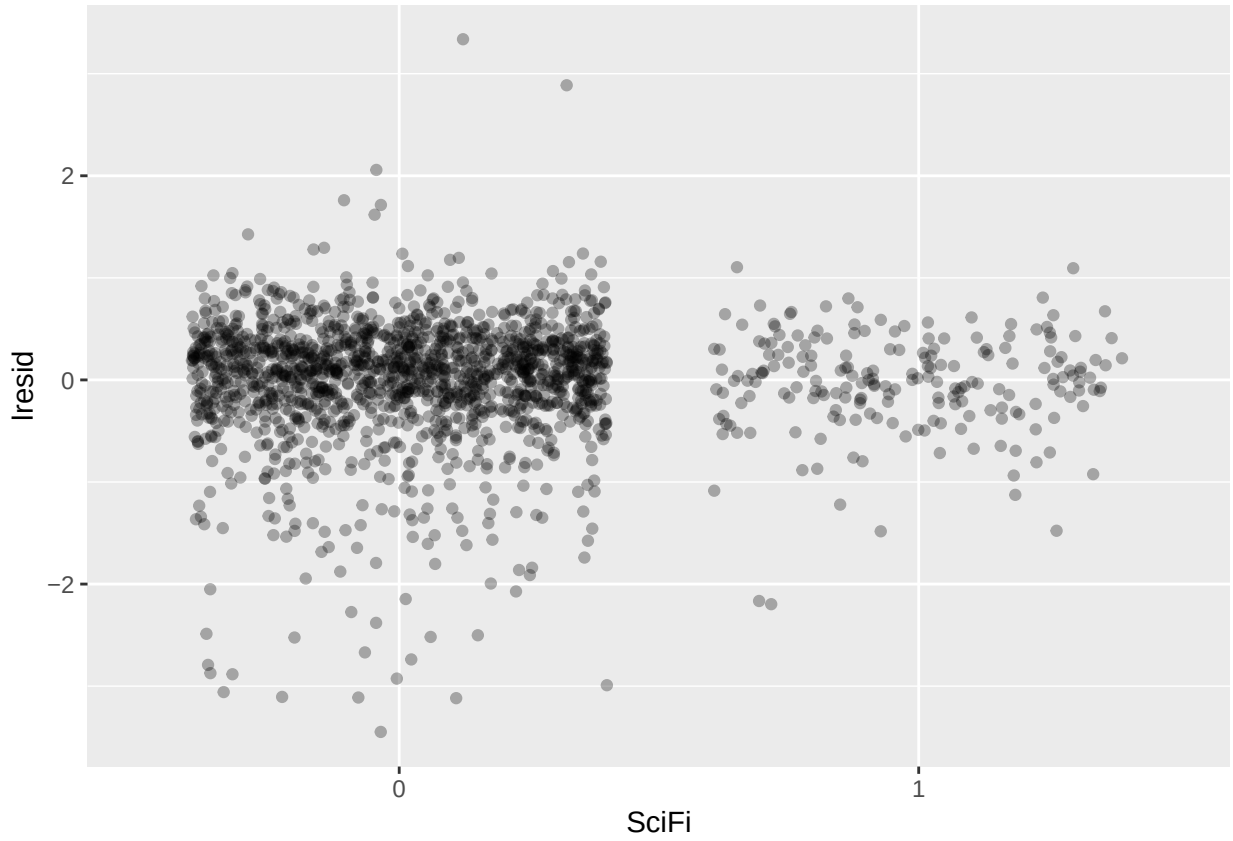

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



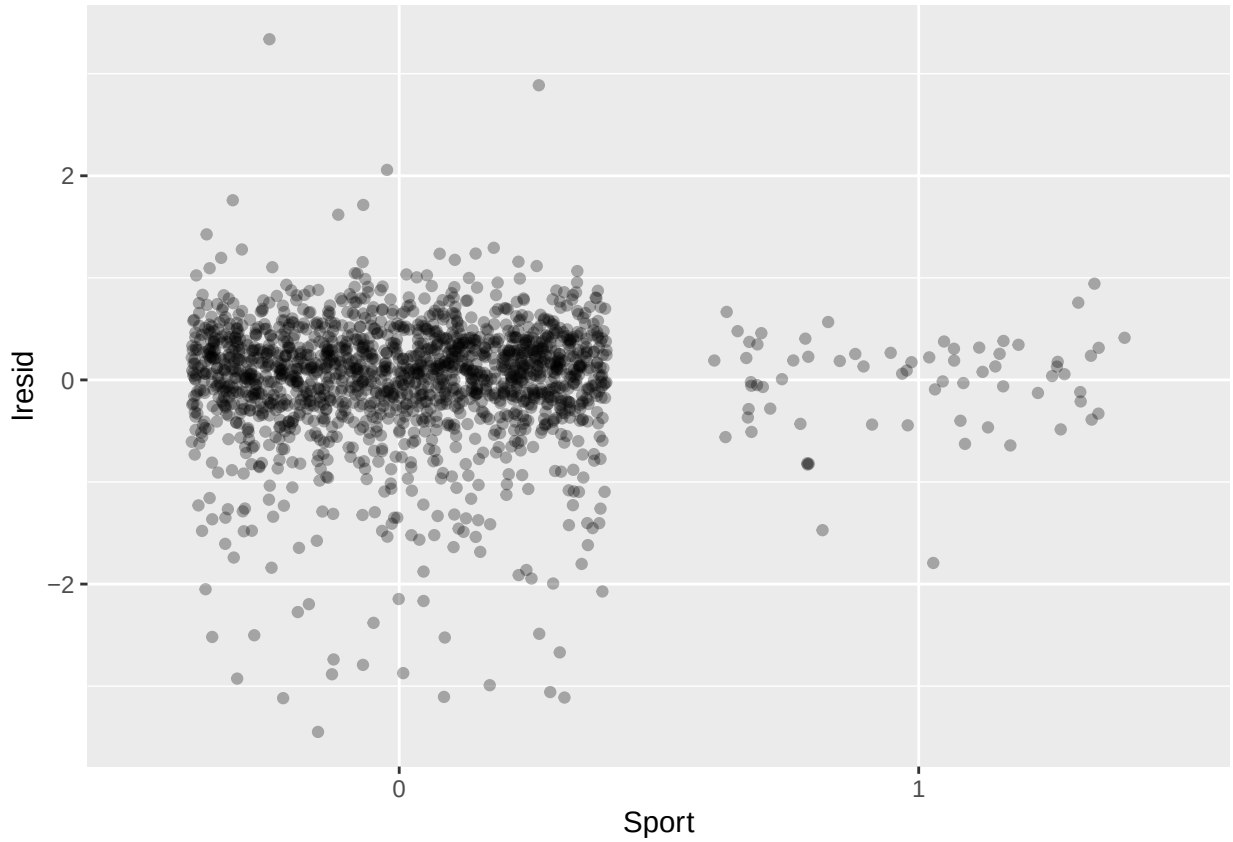

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



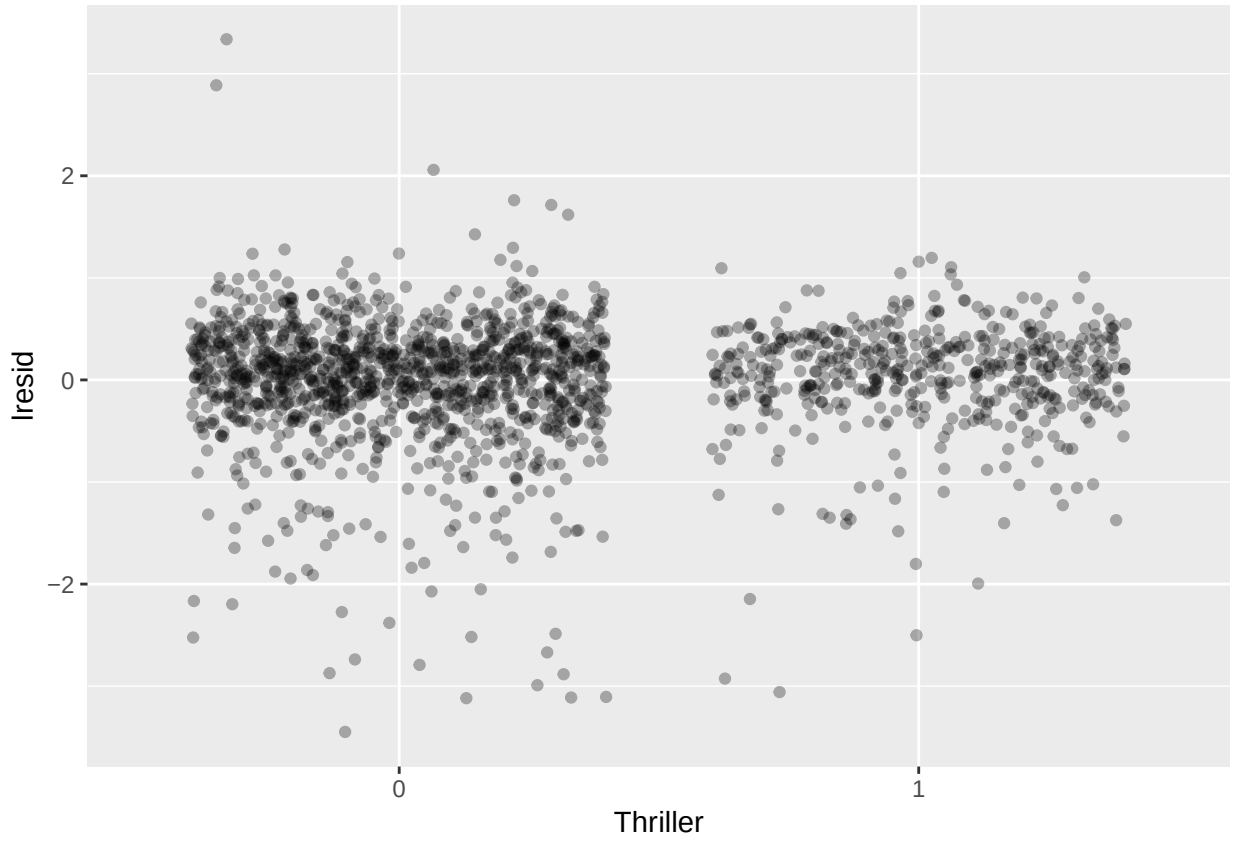

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



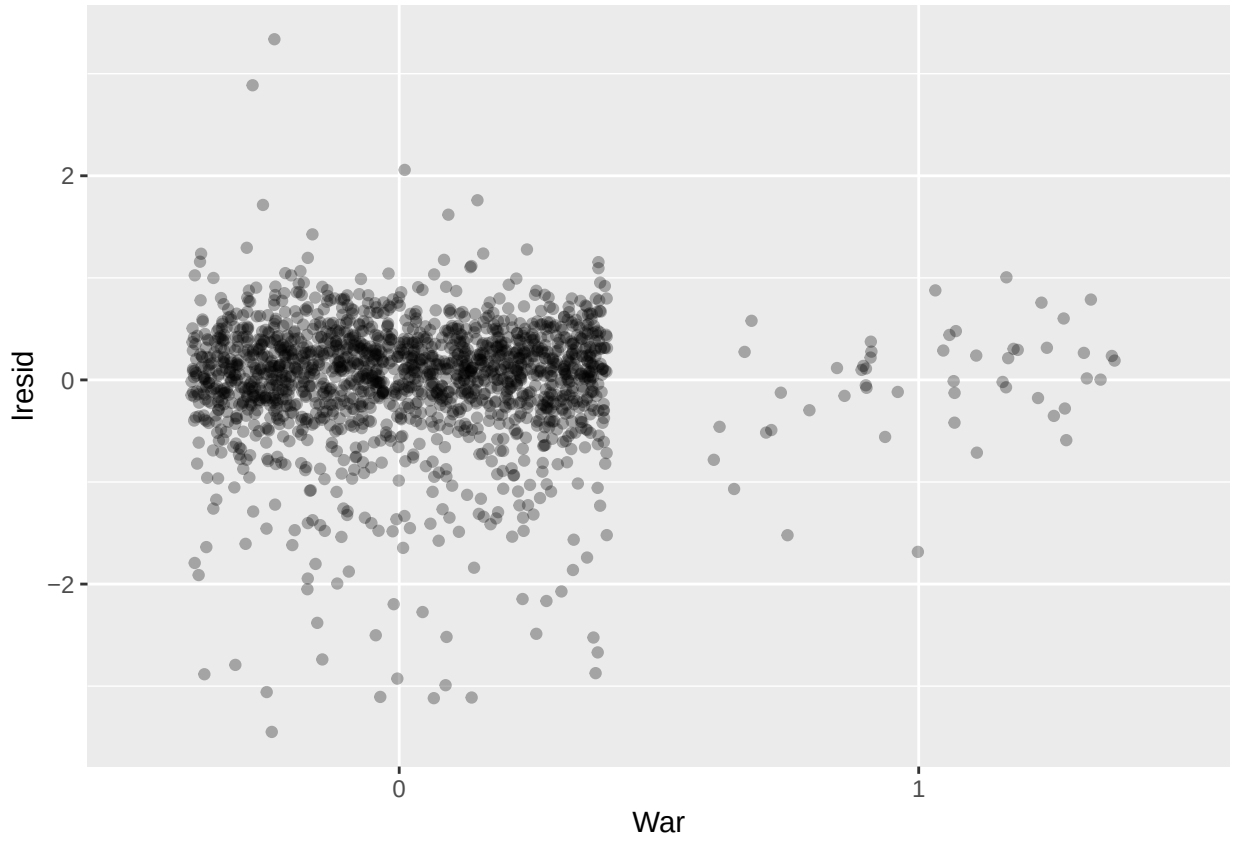

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



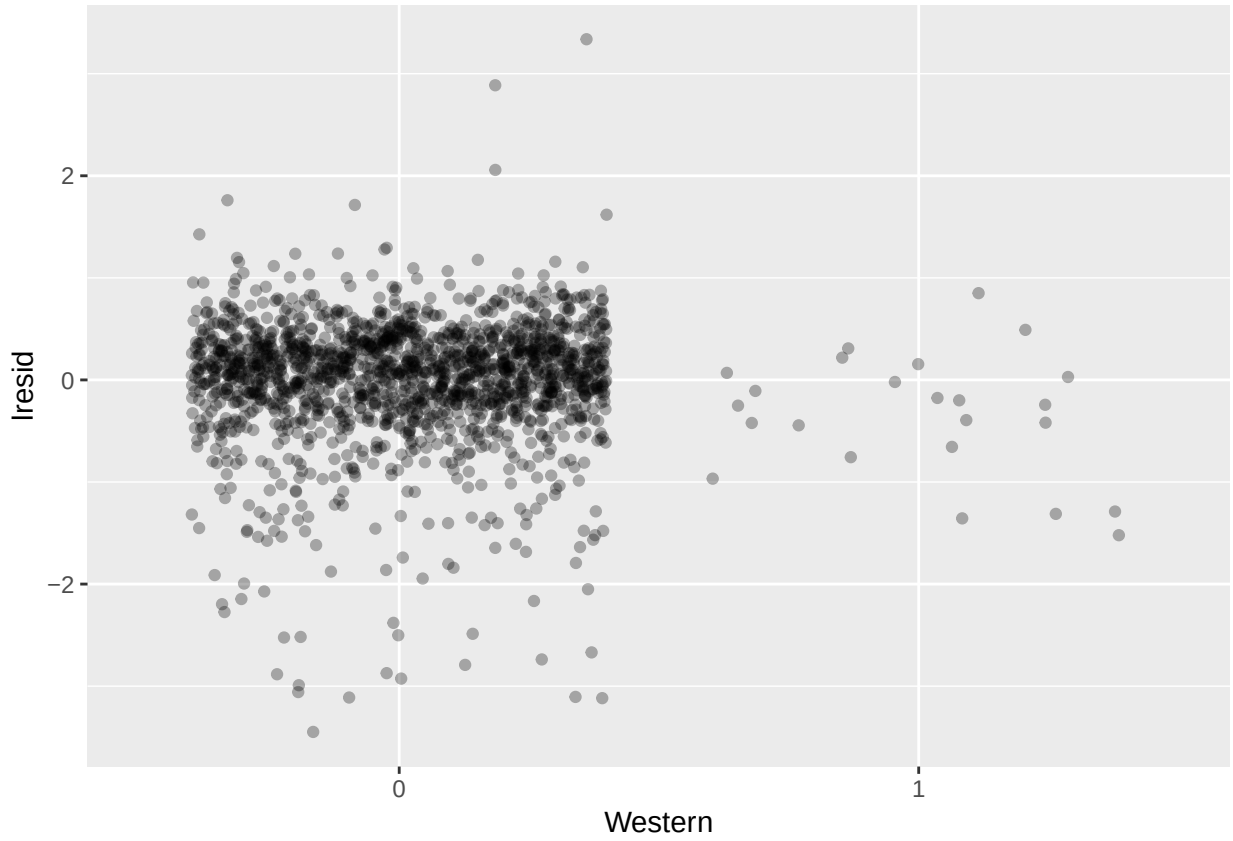

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



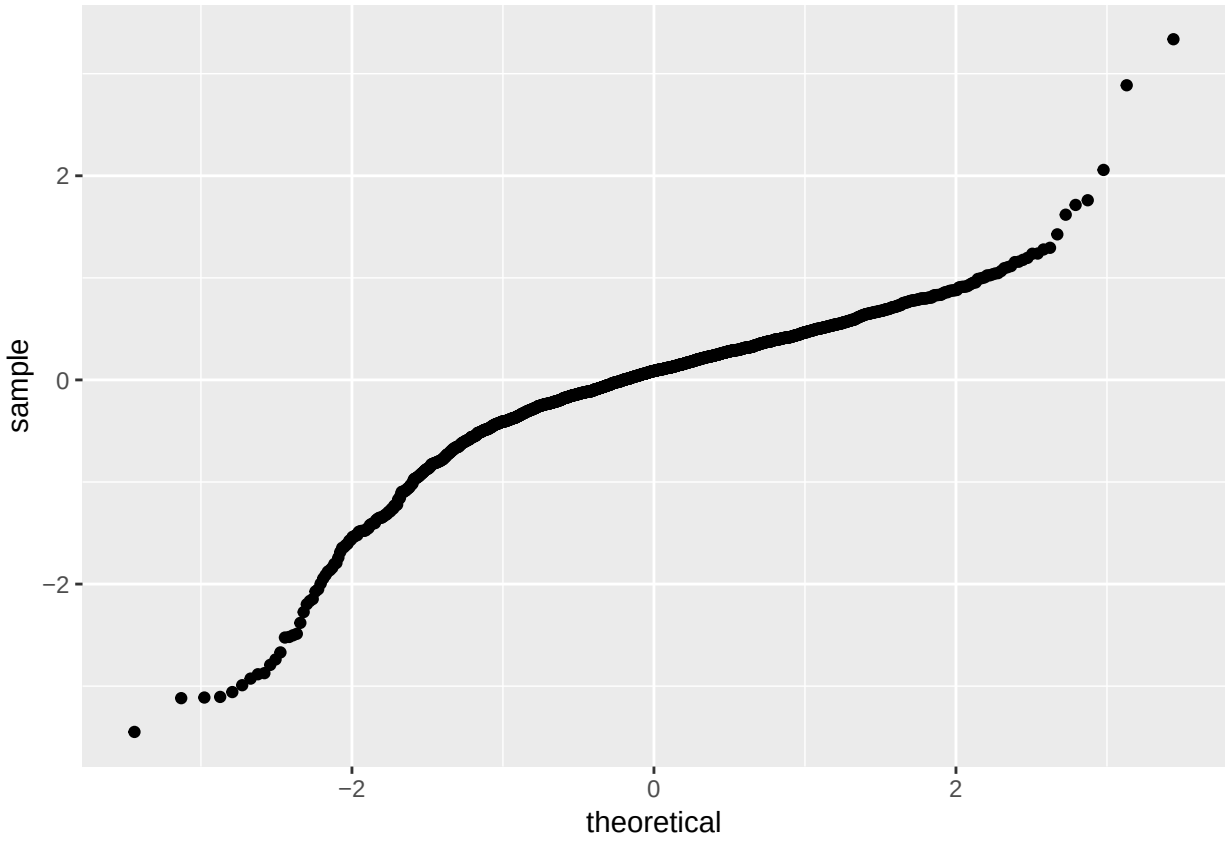

```
## Warning: Removed 132 rows containing missing values (geom_point).
```




```
## Warning: Removed 132 rows containing missing values (geom_point).
```




```
## Warning: Removed 132 rows containing non-finite values (stat_qq).
```




```
# two points with consistently really high residuals (greater than 2) i.e. based on all of their factor
  ### make a graph labeling these and pointing out
# often many points with large negative results less than -2: often get movies that are a flop. High bu
train %>%
  add_residuals(mod_all, 'lresid') %>%
  filter(lresid > 2.1)
```

```
## # A tibble: 2 x 58
##   director_name num_critic_for_~ director_facebo~ actor_3_faceboo~
##   <chr>                   <dbl>            <dbl>            <dbl>
## 1 Daniel Myrick             360               19               39
## 2 Oren Peli                 409              110               21
## # ... with 54 more variables: actor_2_name <chr>,
## #   actor_1_facebook_likes <dbl>, gross <dbl>, genres <chr>,
## #   actor_1_name <chr>, movie_title <chr>,
## #   cast_total_facebook_likes <dbl>, actor_3_name <chr>,
## #   num_user_for_reviews <dbl>, language <chr>, country <chr>,
## #   content_rating <fct>, budget <dbl>, year <fct>,
## #   actor_2_facebook_likes <dbl>, imdb_score <dbl>,
## #   movie_facebook_likes <dbl>, Action <fct>, Adventure <fct>,
## #   Animation <fct>, Biography <fct>, Comedy <fct>, Crime <fct>,
## #   Documentary <fct>, Drama <fct>, Family <fct>, Fantasy <fct>,
## #   History <fct>, Horror <fct>, Music <fct>, Musical <fct>,
## #   Mystery <fct>, Romance <fct>, SciFi <fct>, Sport <fct>,
## #   Thriller <fct>, War <fct>, Western <fct>, gdpd <dbl>,
## #   real_budget <dbl>, real_gross <dbl>, rev_budget <dbl>,
```

```
## #   profit_dum <dbl>, total_oscars_actor1 <dbl>,
## #   total_oscars_actor2 <dbl>, total_oscars_actor3 <dbl>,
## #   total_oscars_director <fct>, total_oscars_actor <fct>,
## #   real_gross_log <dbl>, real_budget_log <dbl>,
## #   director_facebook_likes_log <dbl>,
## #   cast_total_facebook_likes_log <dbl>, imdb_score_log <dbl>,
## #   lresid <dbl>
```

```
train %>%
  add_residuals(mod_all, 'lresid') %>%
  filter(lresid < -2)
```

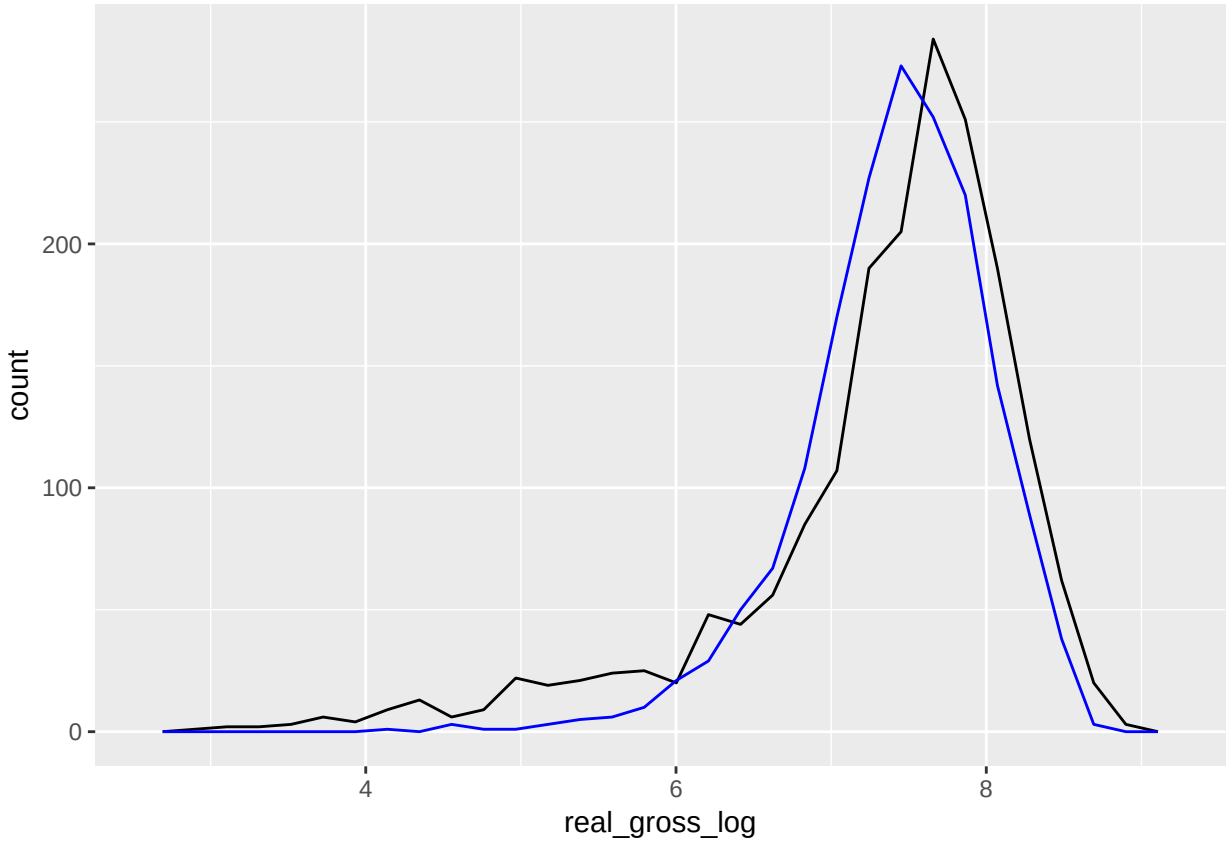
```
## # A tibble: 23 x 58
##    director_name num_critic_for_~ director_facebo~ actor_3_faceboo~
##    <chr>                    <dbl>            <dbl>            <dbl>
##  1 Akiva Goldsm~              189              167              778
##  2 Nick Cassave~              177              415              545
##  3 John Hillcoat              355              214             3000
##  4 George Gallo                45              269              384
##  5 John Duigan                 16               18              186
##  6 Anand Tucker               134               14              356
##  7 Jake Paltrow                50               17               66
##  8 Timothy Hines                1                0              247
##  9 Michael Mere~               23                7              875
## 10 Damian Nieman               25                0               83
## # ... with 13 more rows, and 54 more variables: actor_2_name <chr>,
## #   actor_1_facebook_likes <dbl>, gross <dbl>, genres <chr>,
## #   actor_1_name <chr>, movie_title <chr>,
## #   cast_total_facebook_likes <dbl>, actor_3_name <chr>,
## #   num_user_for_reviews <dbl>, language <chr>, country <chr>,
## #   content_rating <fct>, budget <dbl>, year <fct>,
## #   actor_2_facebook_likes <dbl>, imdb_score <dbl>,
## #   movie_facebook_likes <dbl>, Action <fct>, Adventure <fct>,
## #   Animation <fct>, Biography <fct>, Comedy <fct>, Crime <fct>,
## #   Documentary <fct>, Drama <fct>, Family <fct>, Fantasy <fct>,
## #   History <fct>, Horror <fct>, Music <fct>, Musical <fct>,
## #   Mystery <fct>, Romance <fct>, SciFi <fct>, Sport <fct>,
## #   Thriller <fct>, War <fct>, Western <fct>, gdpd <dbl>,
## #   real_budget <dbl>, real_gross <dbl>, rev_budget <dbl>,
## #   profit_dum <dbl>, total_oscars_actor1 <dbl>,
## #   total_oscars_actor2 <dbl>, total_oscars_actor3 <dbl>,
## #   total_oscars_director <fct>, total_oscars_actor <fct>,
## #   real_gross_log <dbl>, real_budget_log <dbl>,
## #   director_facebook_likes_log <dbl>,
## #   cast_total_facebook_likes_log <dbl>, imdb_score_log <dbl>,
## #   lresid <dbl>
```

Prediction

```
train %>%
  add_predictions(mod_all, 'lpred') %>%
  ggplot() +
  geom_freqpoly(aes(x = real_gross_log)) +
  geom_freqpoly(aes(x = lpred), color = 'blue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 132 rows containing non-finite values (stat_bin).
```



Fit Model W/O Assuming Genre

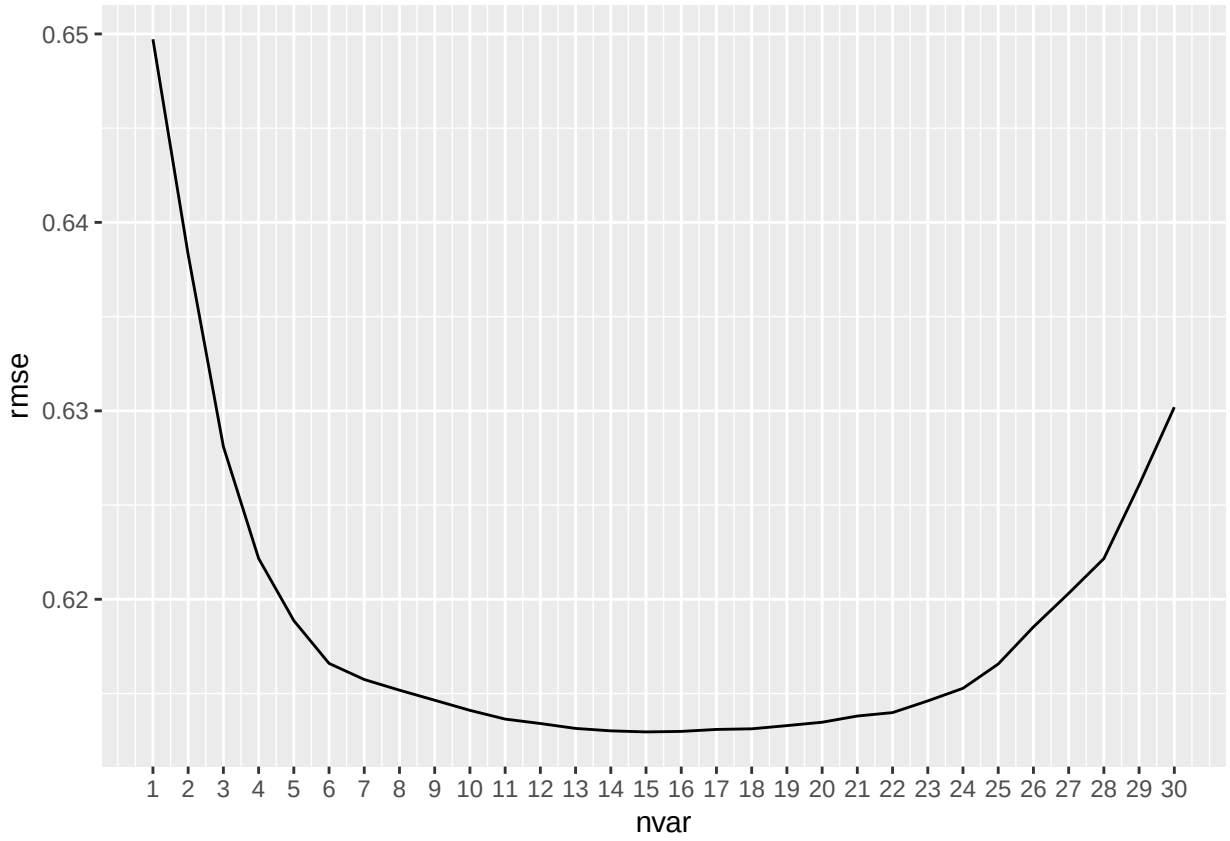
```
# stepwise
# ALL potentially relevant variables
rmse_lst <- step_wise_loop(df = train %>% select(all_genre_vars, content_rating, real_budget, year,
                                                 total_oscars_actor, total_oscars_director,
                                                 imdb_score_log, real_budget_log,
                                                 director_facebook_likes_log, cast_total_facebook_likes
```

```
## real_budget_log
##       0.6497192
## [1] 1
## imdb_score_log
##      0.6382915
## [1] 2
##      year
## 0.6281035
## [1] 3
##    Comedy
## 0.6221681
## [1] 4
```

```
## content_rating
##      0.6188688
## [1] 5
##   Mystery
## 0.6165971
## [1] 6
## Biography
## 0.6157421
## [1] 7
##   Musical
## 0.6151764
## [1] 8
##     Sport
## 0.6146438
## [1] 9
##     Crime
## 0.6141069
## [1] 10
## Documentary
##   0.6136432
## [1] 11
##     Music
## 0.6134072
## [1] 12
##     SciFi
## 0.6131476
## [1] 13
##    Action
## 0.6130183
## [1] 14
##   History
## 0.6129618
## [1] 15
## total_oscars_director
##             0.6129888
## [1] 16
##  Fantasy
## 0.613094
## [1] 17
## real_budget
##   0.6131265
## [1] 18
## Animation
## 0.6132966
## [1] 19
##   Romance
## 0.6134755
## [1] 20
##       War
## 0.6138056
## [1] 21
##     Drama
## 0.6139871
## [1] 22
```

```
##  Thriller
## 0.6146081
## [1] 23
## cast_total_facebook_likes_log
##                     0.6152761
## [1] 24
## total_oscars_actor
##          0.6165643
## [1] 25
##    Family
## 0.6185332
## [1] 26
## director_facebook_likes_log
##                   0.6203153
## [1] 27
##    Horror
## 0.6221584
## [1] 28
##   Western
## 0.6260611
## [1] 29
## Adventure
## 0.6301929
```

```r
# 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 6, decreases too small or increase

mod_all2 <- lm(real_gross_log ~ real_budget_log + imdb_score_log + year + Comedy + content_rating + Mys
               data = train)

summary(mod_all2) # still a lot of insignificant
```

```
##
## Call:
## lm(formula = real_gross_log ~ real_budget_log + imdb_score_log +
##     year + Comedy + content_rating + Mystery, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5488 -0.2234  0.0941  0.3462  3.5384
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.290225   0.396873   0.731  0.46471
## real_budget_log     0.827045   0.026177  31.594  < 2e-16 ***
## imdb_score_log      1.710134   0.201331   8.494  < 2e-16 ***
## year1981           -0.254810   0.376404  -0.677  0.49852
## year1982            0.296995   0.346191   0.858  0.39107
## year1983            0.246156   0.392671   0.627  0.53083
## year1984            0.506559   0.335606   1.509  0.13139
## year1985            0.345134   0.346559   0.996  0.31945
```

```
## year1986             0.079291   0.331303   0.239  0.81088
## year1987             0.212157   0.322316   0.658  0.51048
## year1988             0.227782   0.315951   0.721  0.47105
## year1989             0.277787   0.311206   0.893  0.37219
## year1990             0.162527   0.312245   0.521  0.60278
## year1991             0.019237   0.315800   0.061  0.95143
## year1992             0.039691   0.321044   0.124  0.90162
## year1993             0.005509   0.312227   0.018  0.98593
## year1994            -0.199457   0.306368  -0.651  0.51511
## year1995            -0.051840   0.298095  -0.174  0.86196
## year1996            -0.220062   0.291028  -0.756  0.44966
## year1997            -0.138061   0.291017  -0.474  0.63527
## year1998            -0.266021   0.290209  -0.917  0.35946
## year1999            -0.248466   0.286897  -0.866  0.38659
## year2000            -0.155180   0.288267  -0.538  0.59043
## year2001            -0.261791   0.286269  -0.914  0.36059
## year2002            -0.237985   0.286021  -0.832  0.40550
## year2003            -0.193529   0.287432  -0.673  0.50085
## year2004            -0.213507   0.287003  -0.744  0.45703
## year2005            -0.226318   0.285994  -0.791  0.42886
## year2006            -0.316366   0.286338  -1.105  0.26937
## year2007            -0.284521   0.287894  -0.988  0.32316
## year2008            -0.335182   0.285977  -1.172  0.24134
## year2009            -0.342031   0.285352  -1.199  0.23084
## year2010            -0.350964   0.285611  -1.229  0.21931
## year2011            -0.232108   0.287990  -0.806  0.42038
## year2012            -0.124210   0.286558  -0.433  0.66474
## year2013            -0.046933   0.286103  -0.164  0.86972
## year2014            -0.112968   0.288063  -0.392  0.69499
## year2015            -0.223600   0.290358  -0.770  0.44136
## year2016            -0.086210   0.306680  -0.281  0.77866
## Comedy1              0.083397   0.032751   2.546  0.01097 *
## content_ratingNC-17 -0.160263   0.272686  -0.588  0.55680
## content_ratingPG    -0.118755   0.119473  -0.994  0.32037
## content_ratingPG-13 -0.156513   0.116101  -1.348  0.17782
## content_ratingR     -0.325984   0.116336  -2.802  0.00514 **
## Mystery1             0.088559   0.053305   1.661  0.09683 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6203 on 1674 degrees of freedom
##   (132 observations deleted due to missingness)
## Multiple R-squared:  0.471,  Adjusted R-squared:  0.4571
## F-statistic: 33.87 on 44 and 1674 DF,  p-value: < 2.2e-16
```

```
rmse(mod_all2, data = valid) # fit is somewhat better (and fewer variables)
```

```
## [1] 0.6165971
```

```
# when consider factors as one variable, they are significant
anova(mod_all2)
```

```
## Analysis of Variance Table
##
## Response: real_gross_log
```

```
##                   Df Sum Sq Mean Sq   F value    Pr(>F)
## real_budget_log    1 491.67  491.67 1277.7221 < 2.2e-16 ***
## imdb_score_log     1  28.18   28.18   73.2444 < 2.2e-16 ***
## year              36  36.51    1.01    2.6359 5.733e-07 ***
## Comedy             1   4.01    4.01   10.4110  0.001277 **
## content_rating     4  12.05    3.01    7.8296 2.994e-06 ***
## Mystery            1   1.06    1.06    2.7601  0.096829 .
## Residuals       1674 644.16    0.38
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# number of observations
nobs(mod_all2)
```

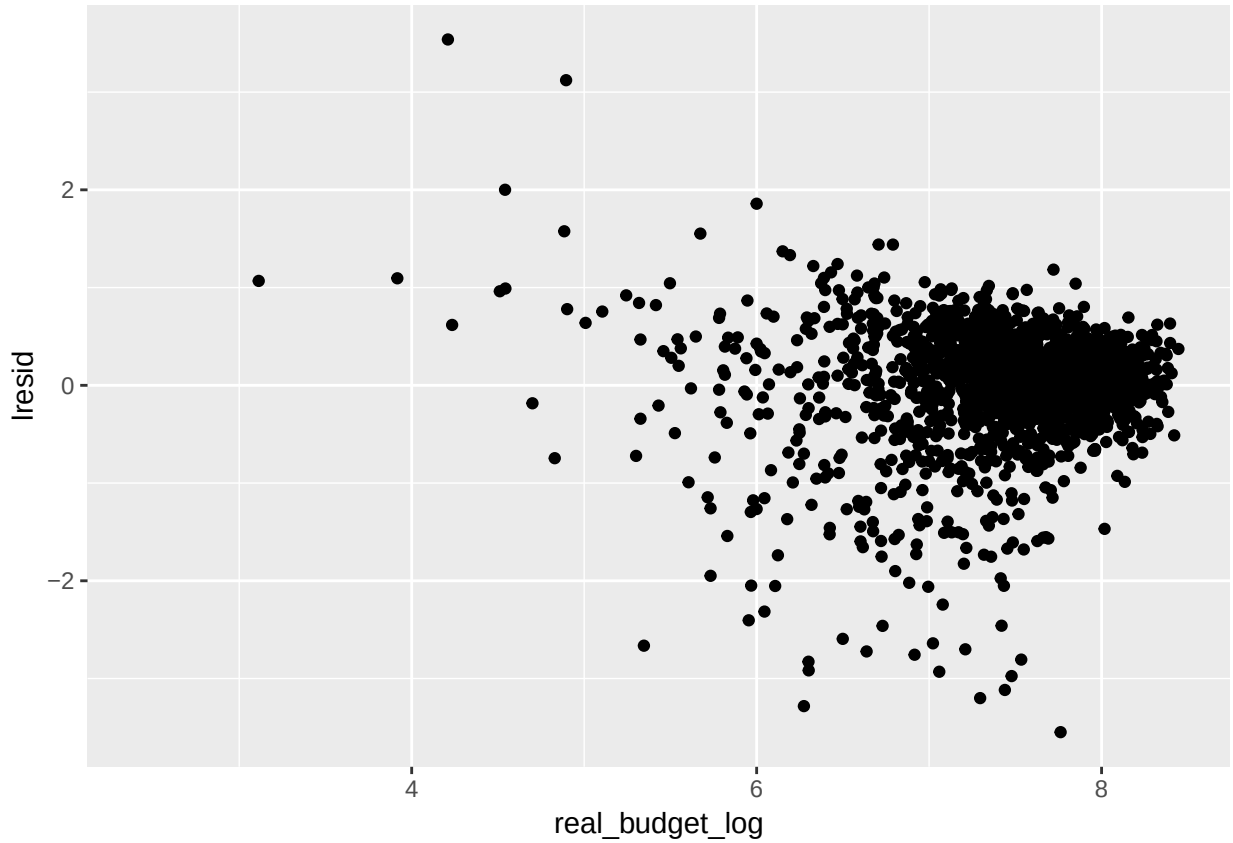
```
## [1] 1719
```

New Residuals

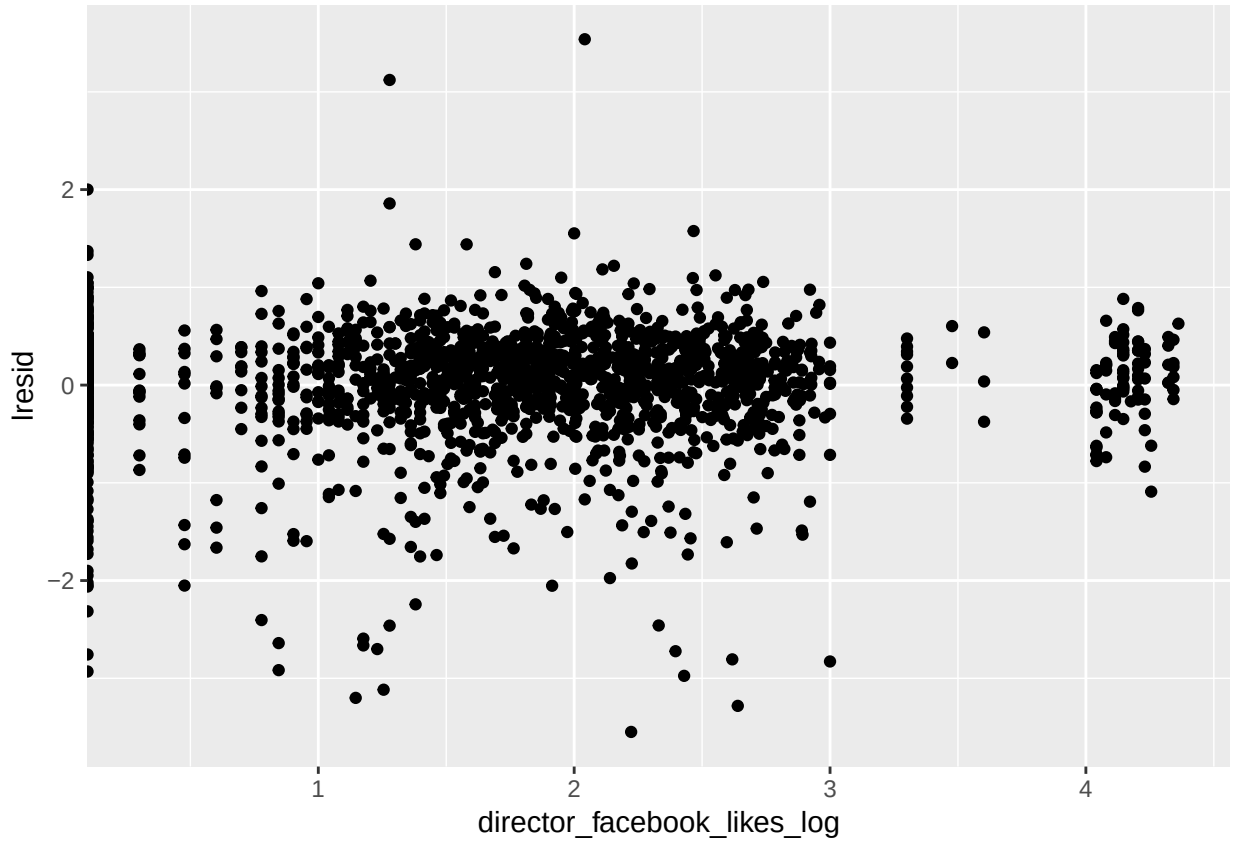
Graph residuals of included and excluded variables: have we captured all of the relationships?

```
gr_resid(mod_all2)
```

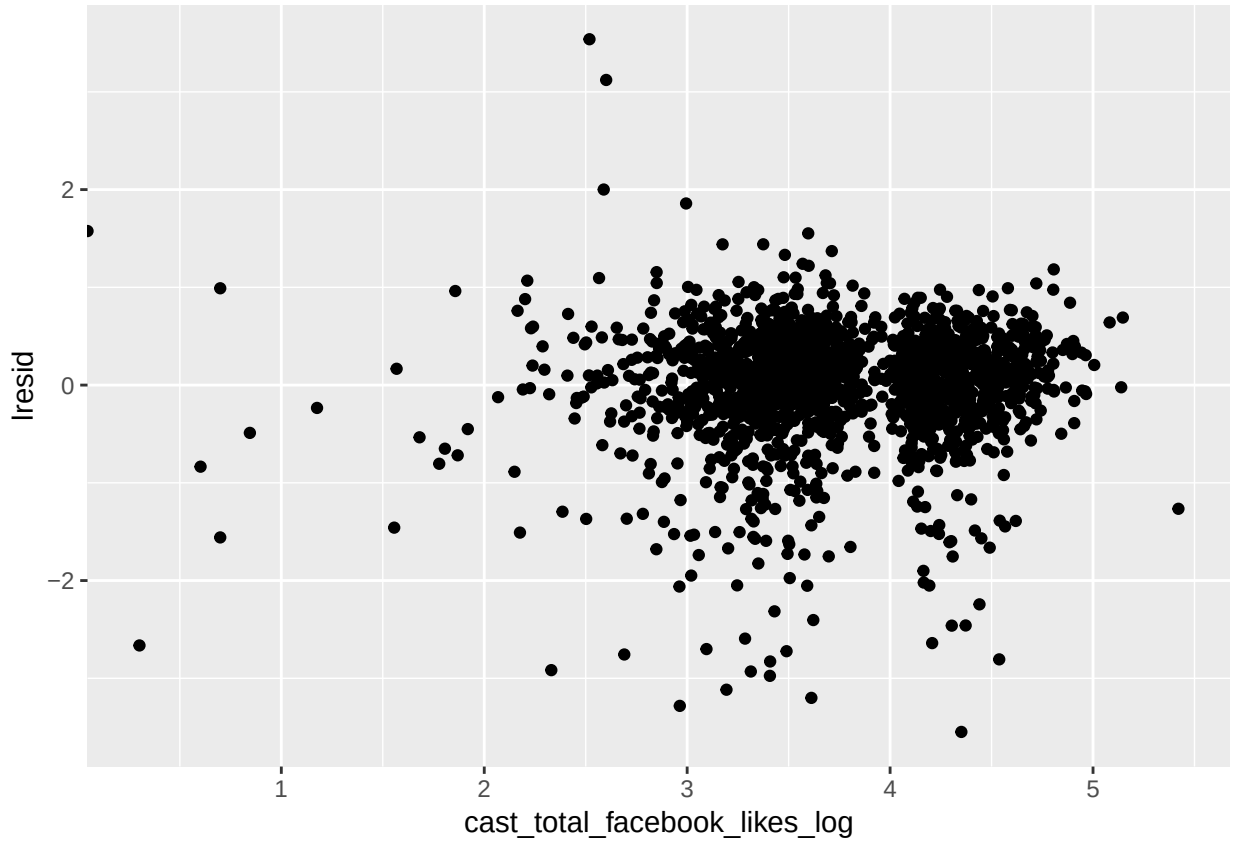
```
## Warning: Removed 132 rows containing missing values (geom_point).
```



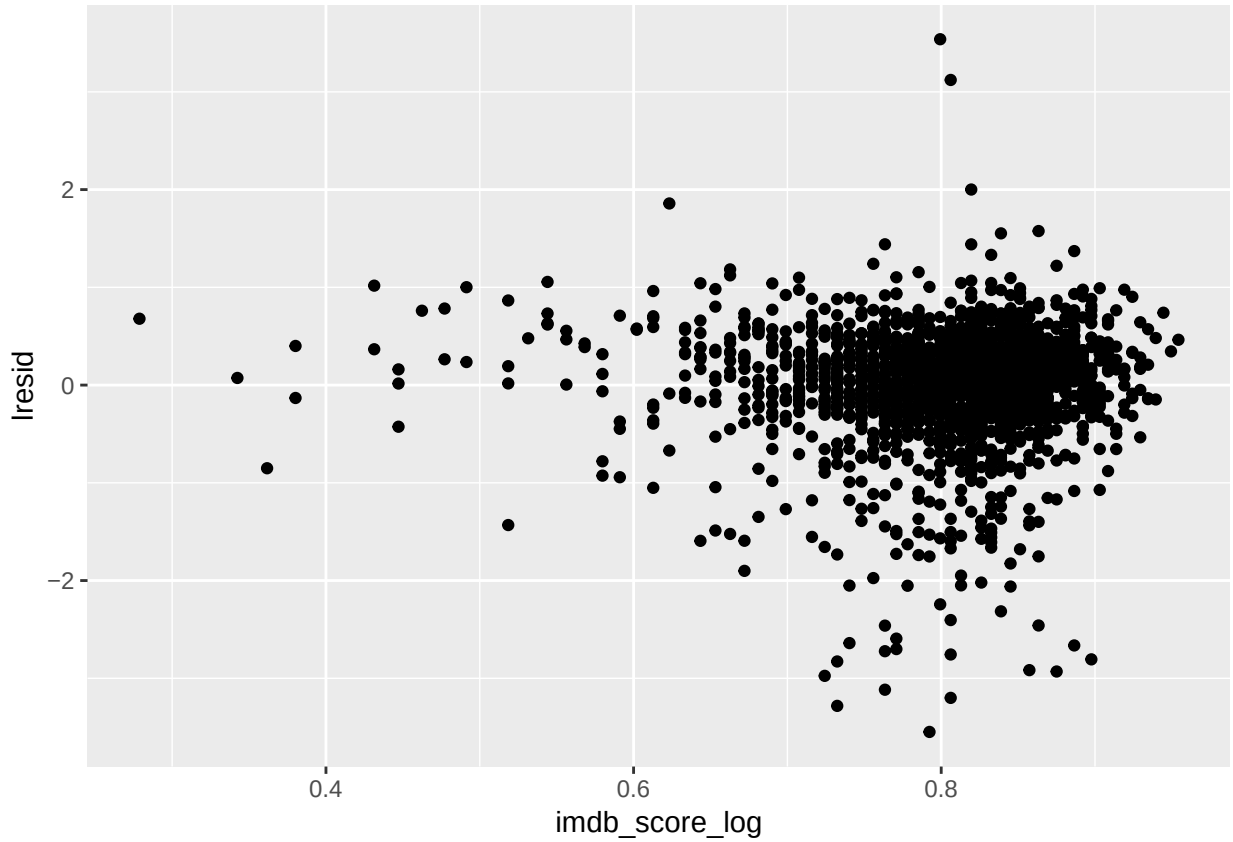

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



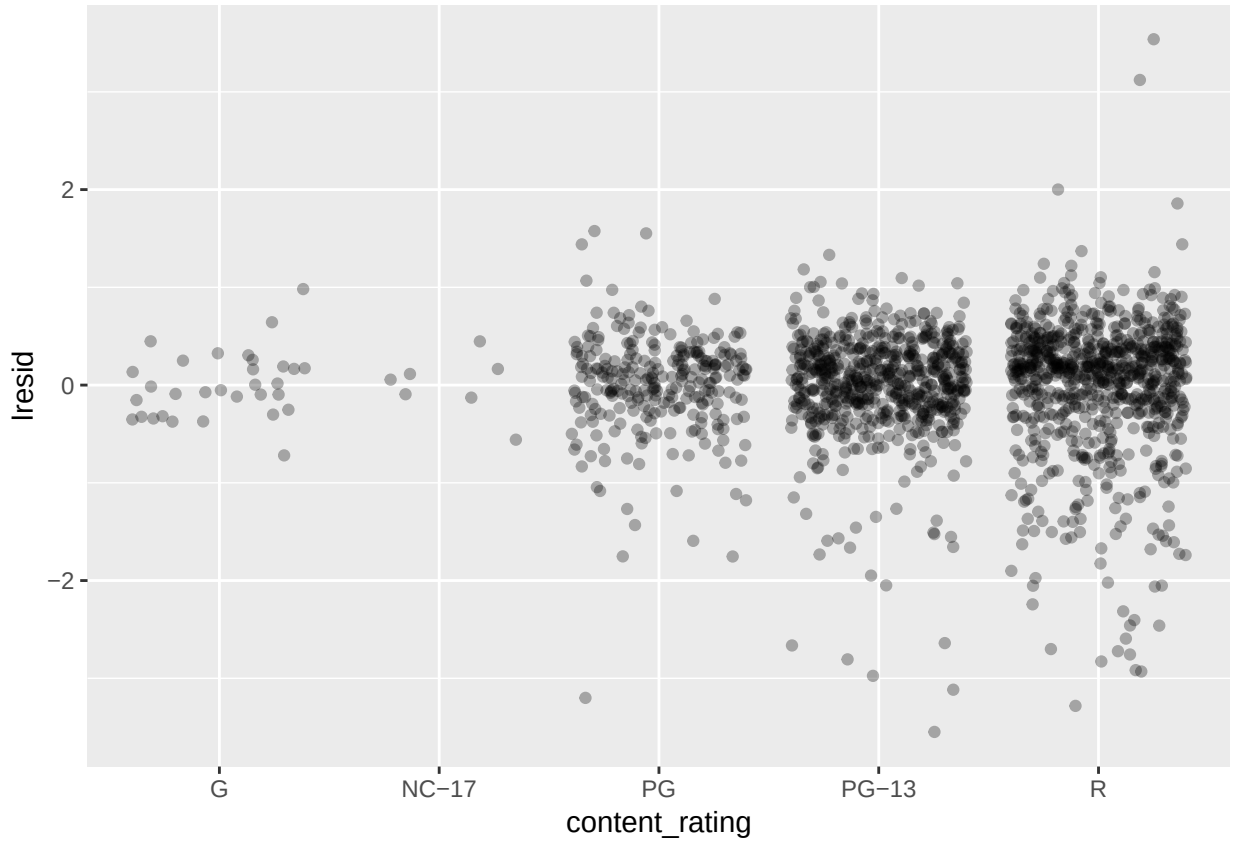

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



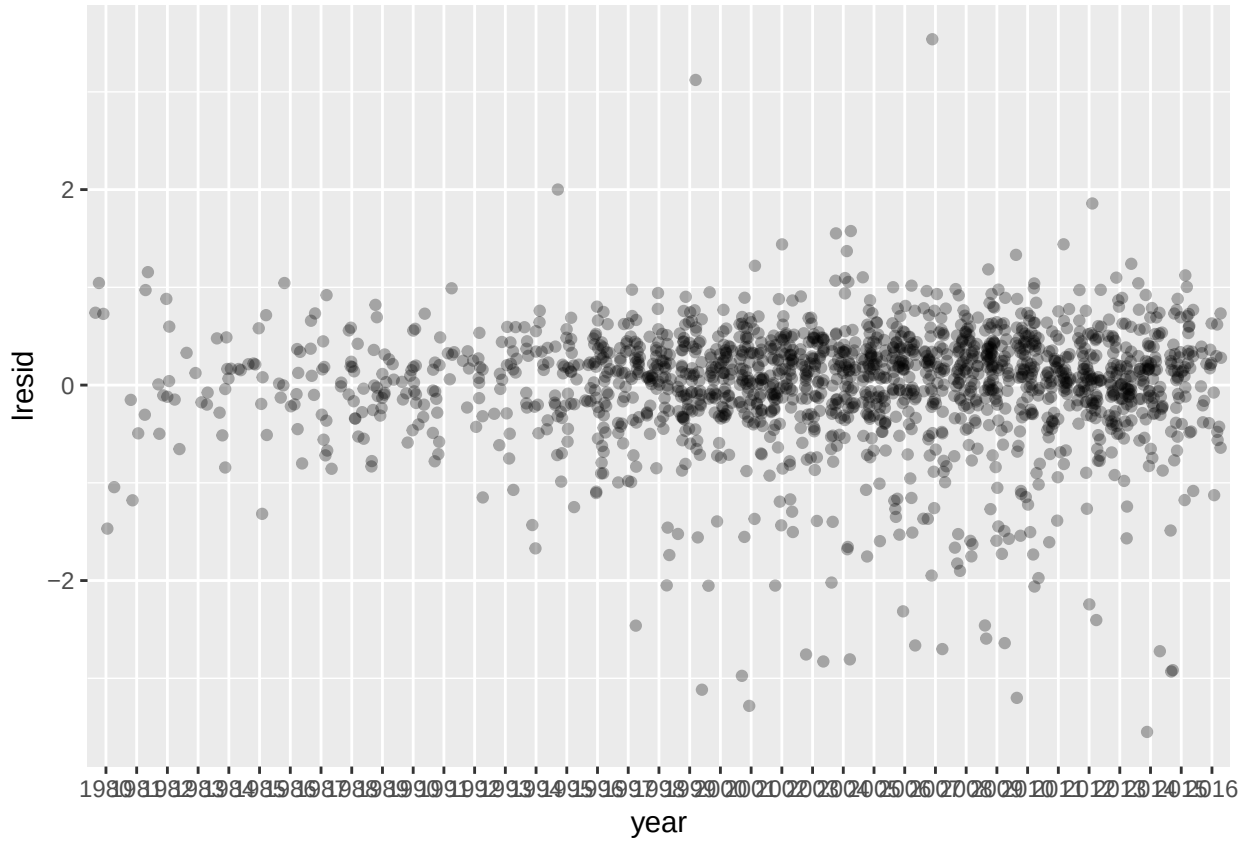

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



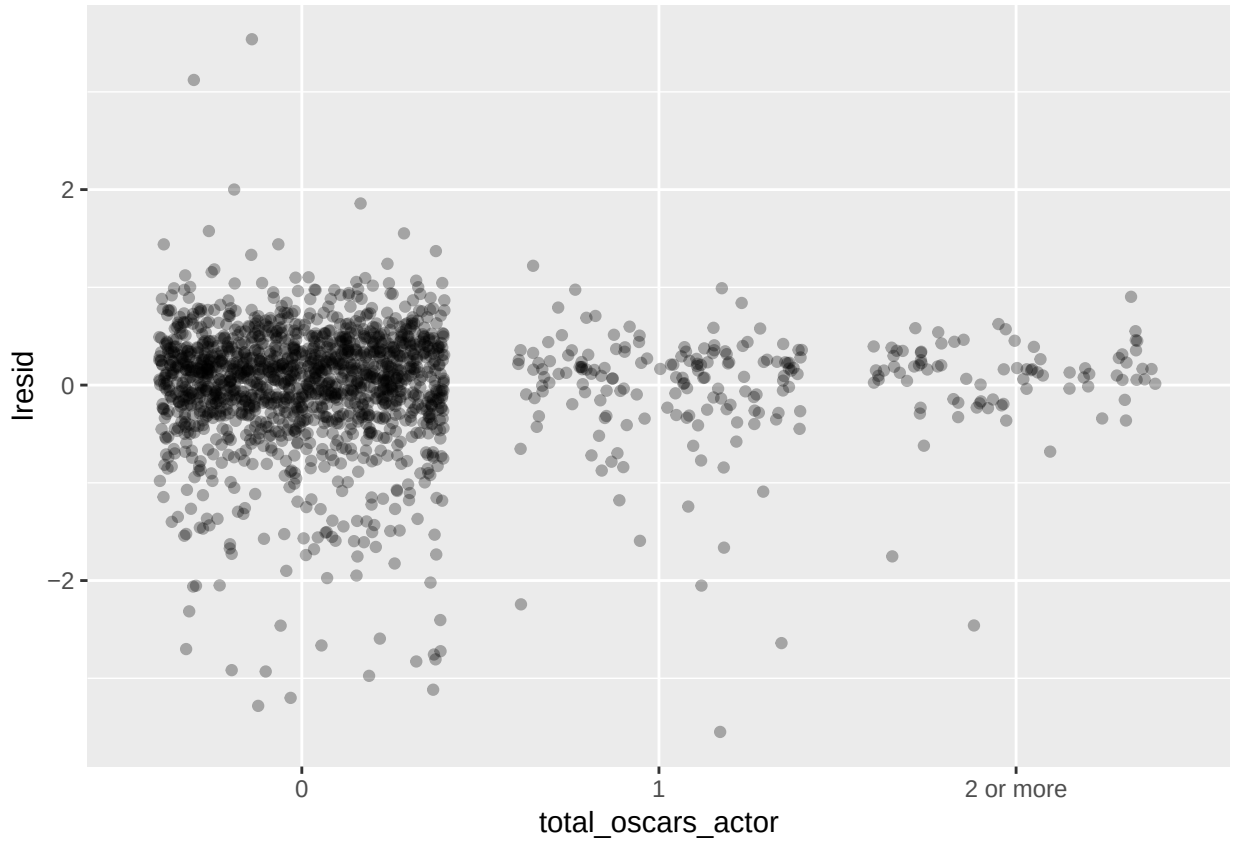

```
## Warning: Removed 94 rows containing missing values (geom_point).
```



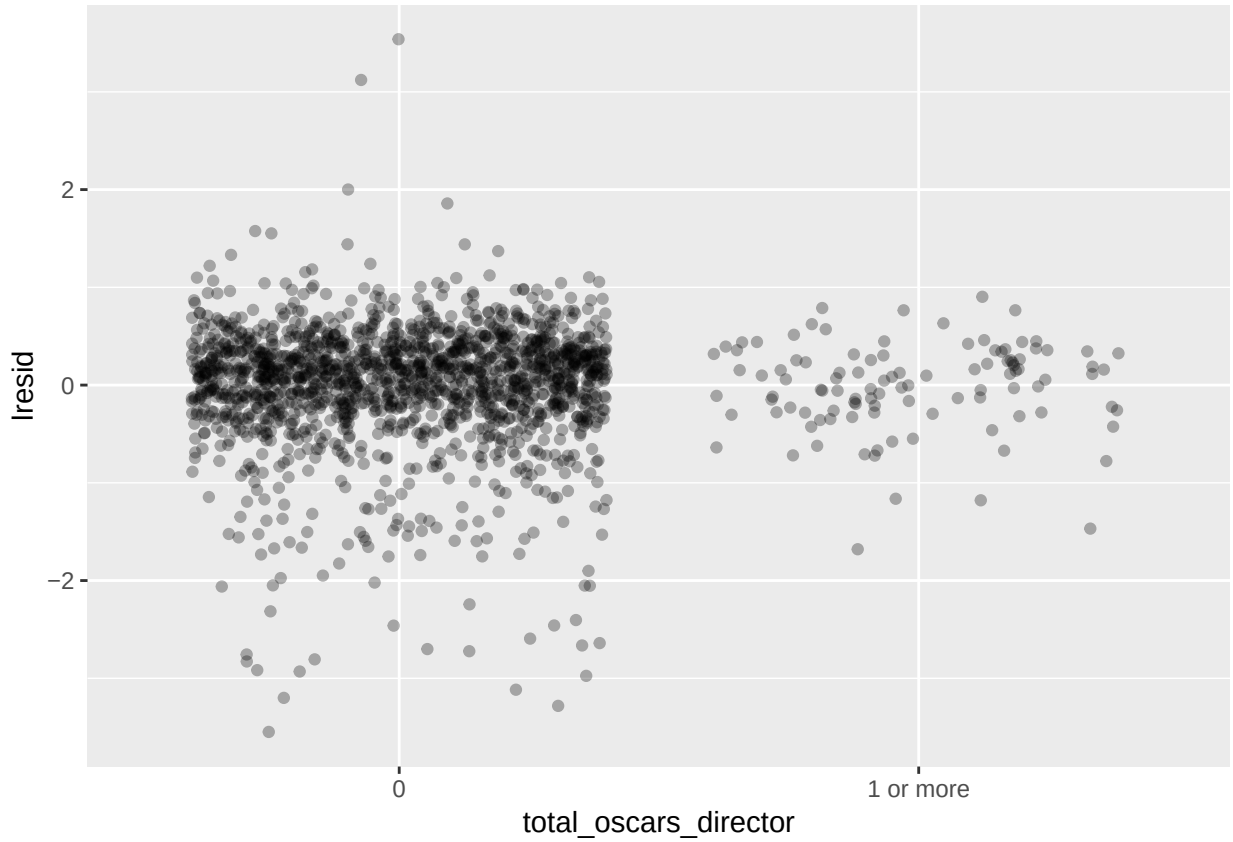

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



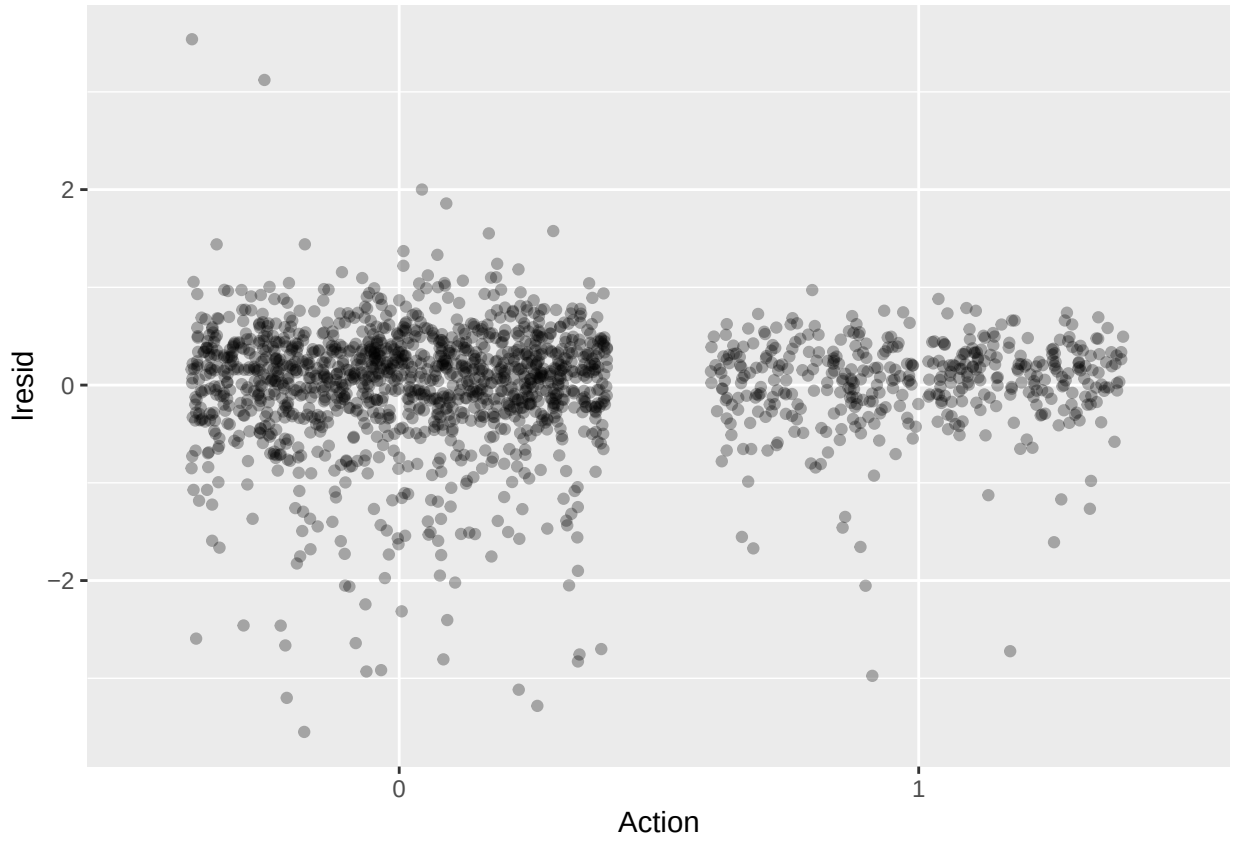

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



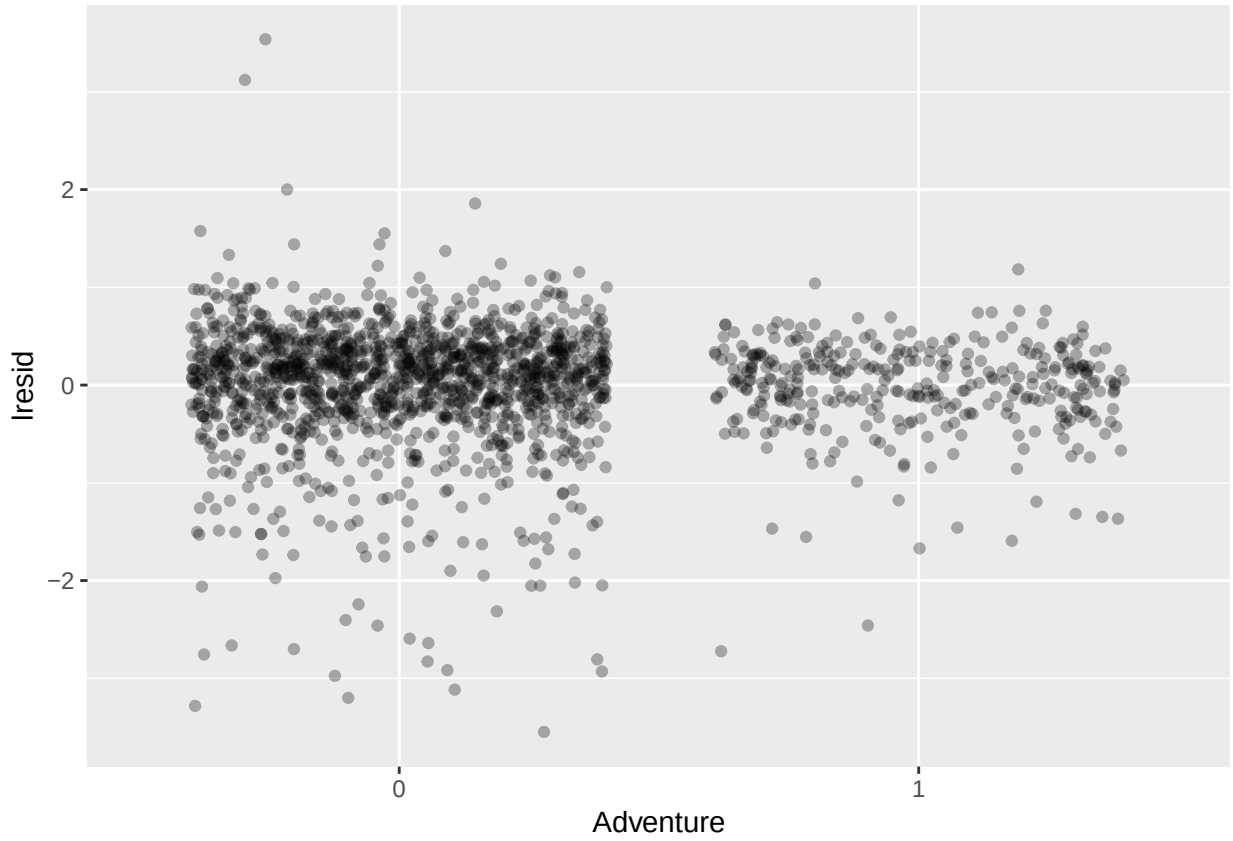

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



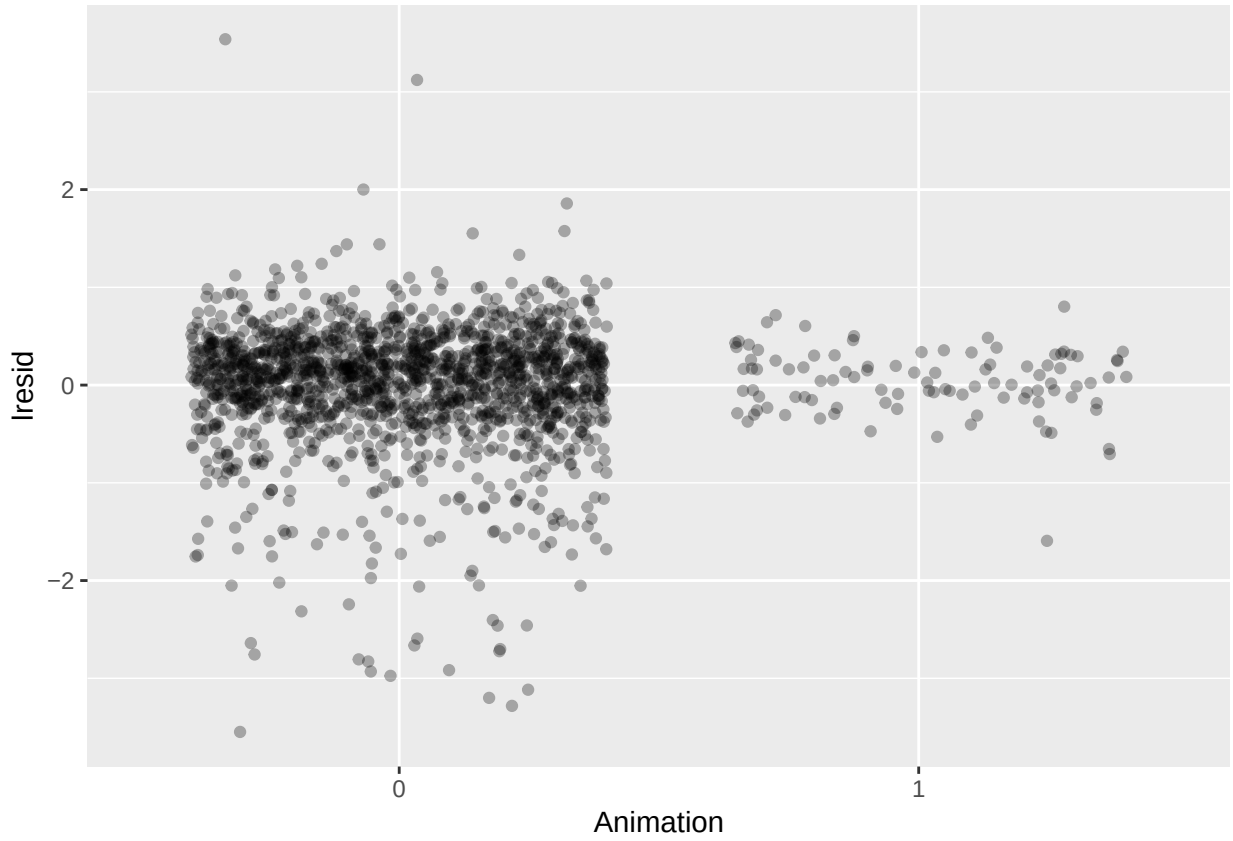

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



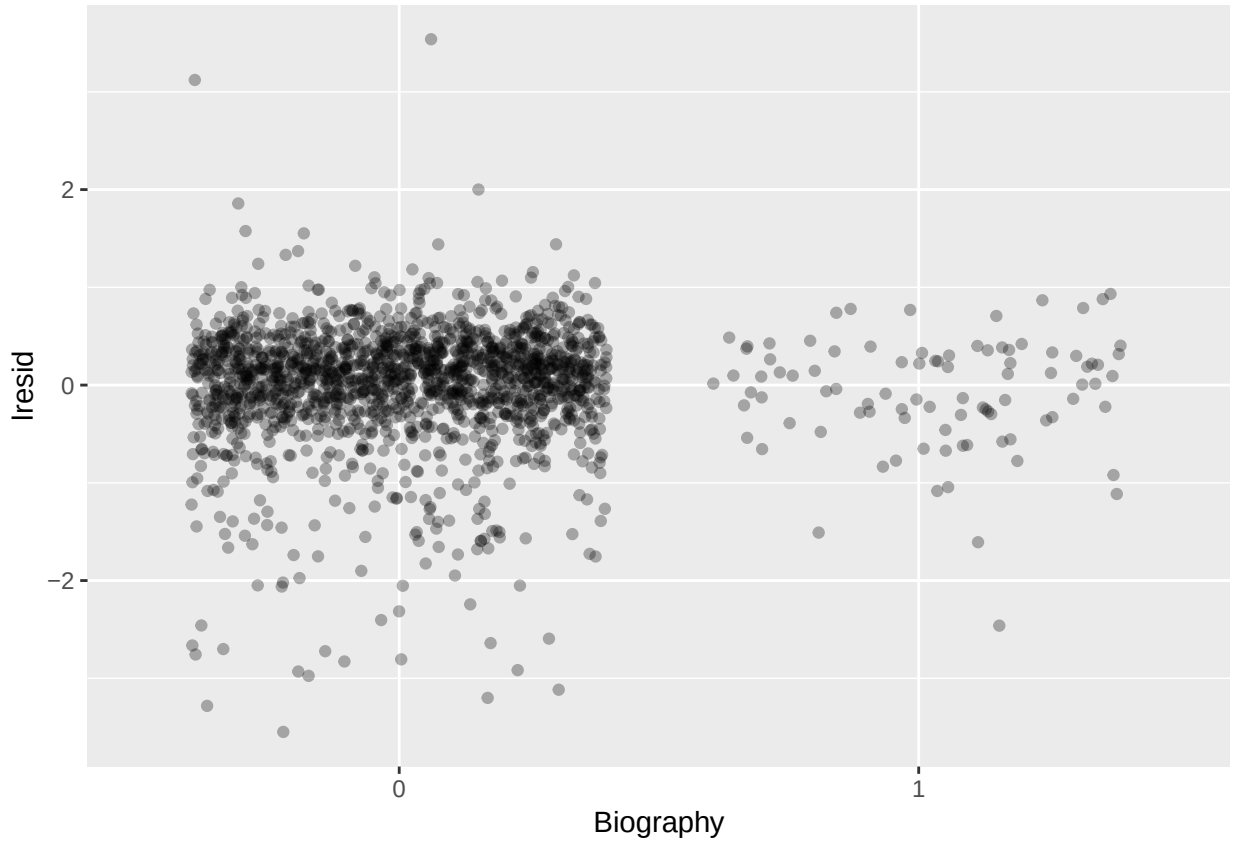

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



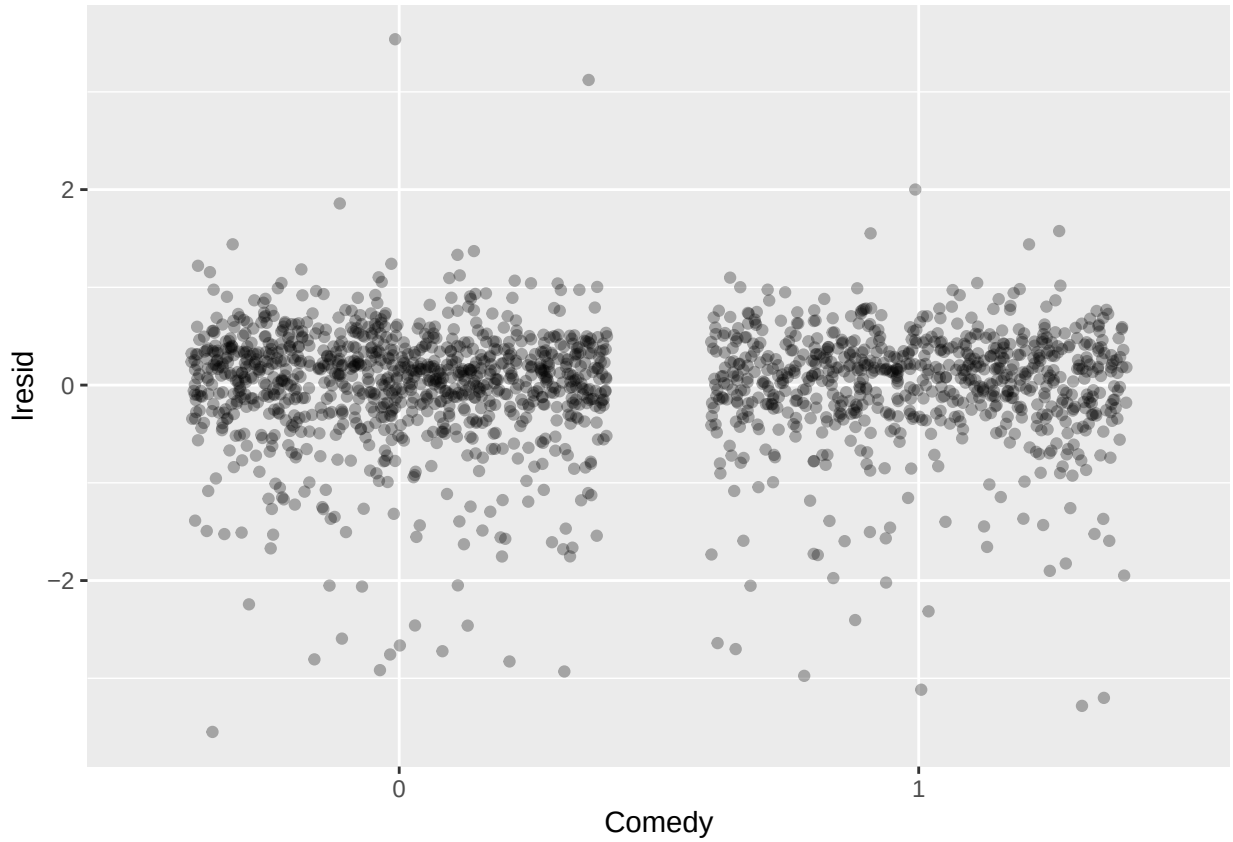

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



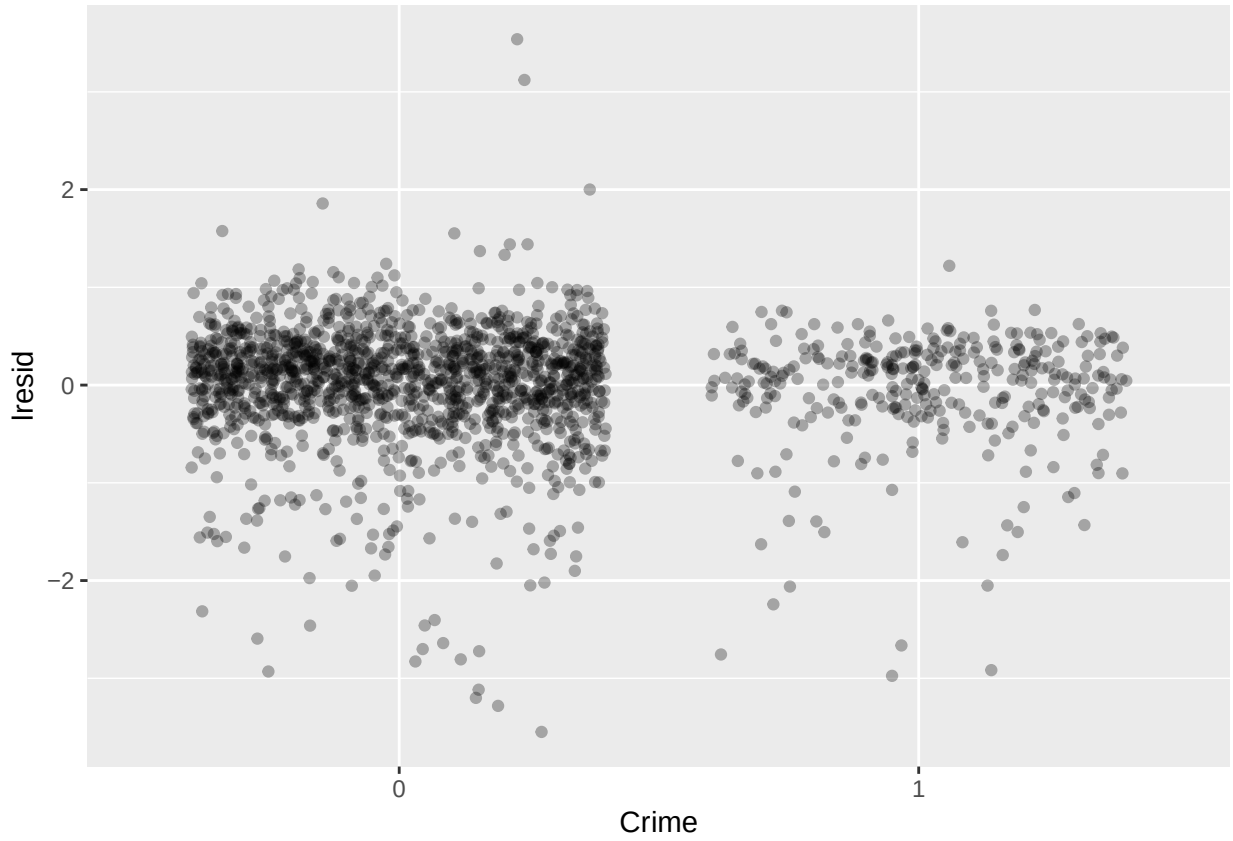

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



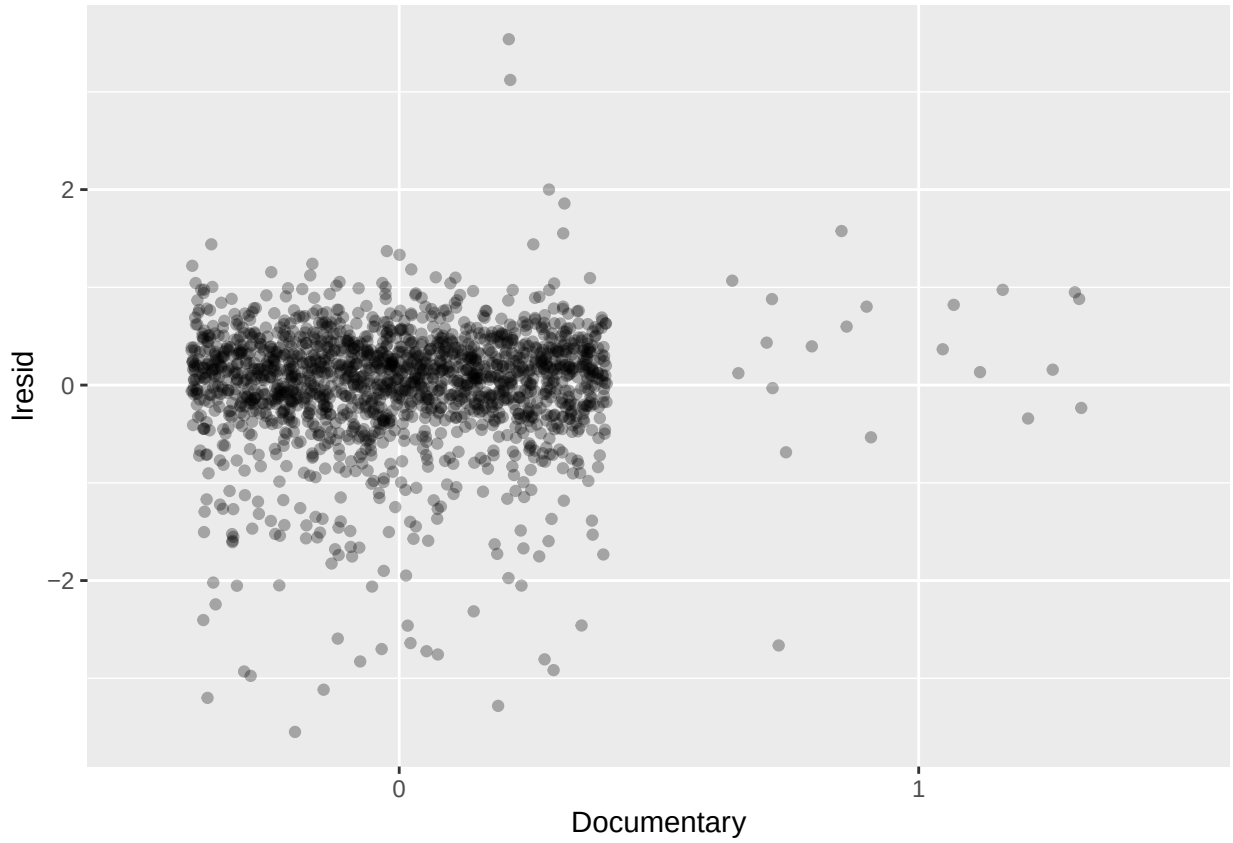

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



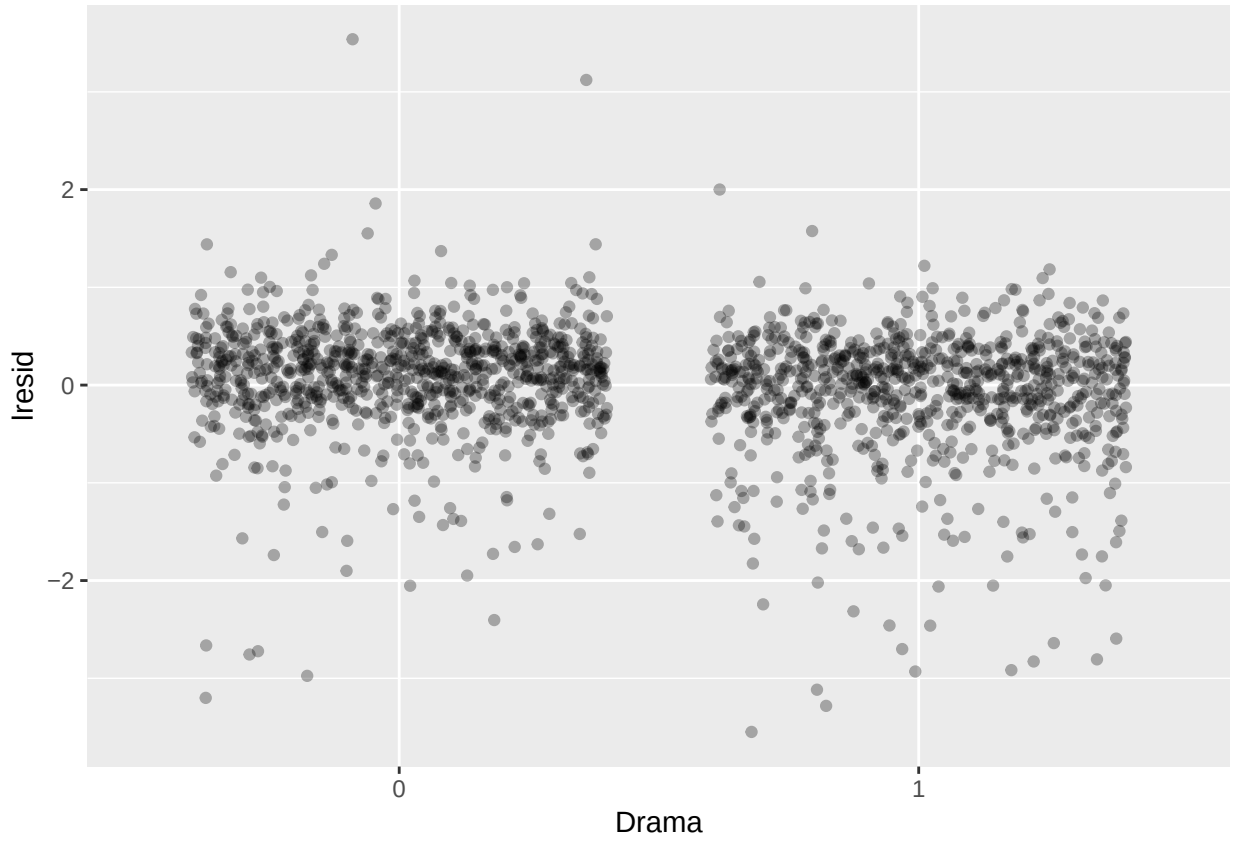

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



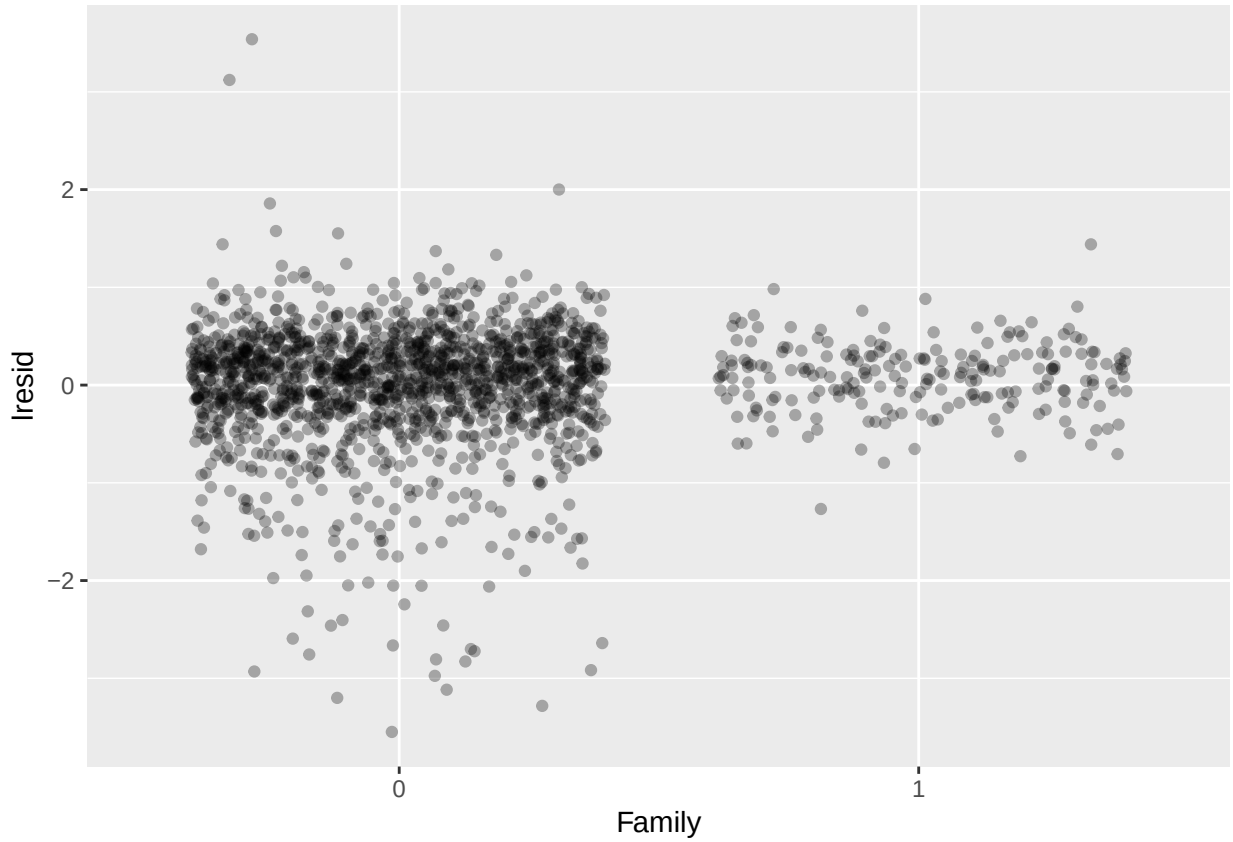

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



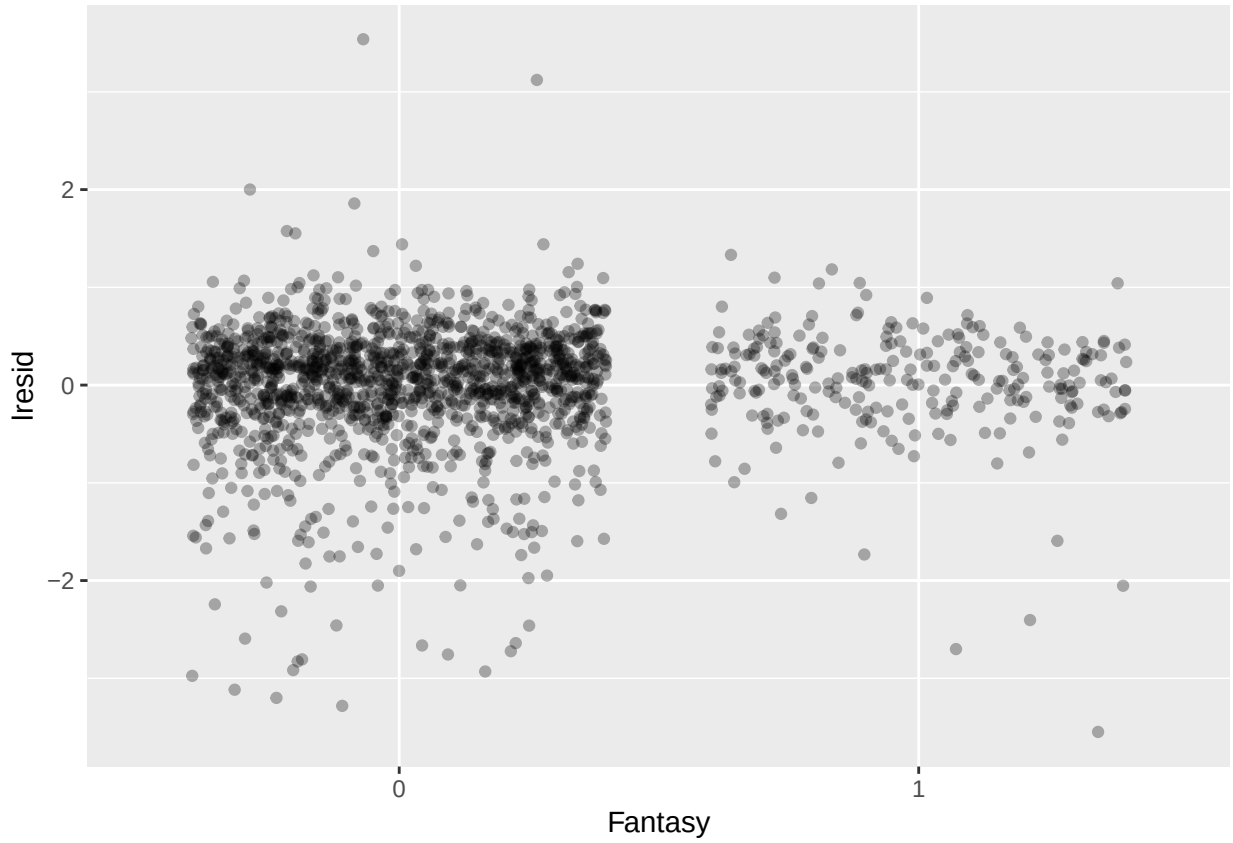

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



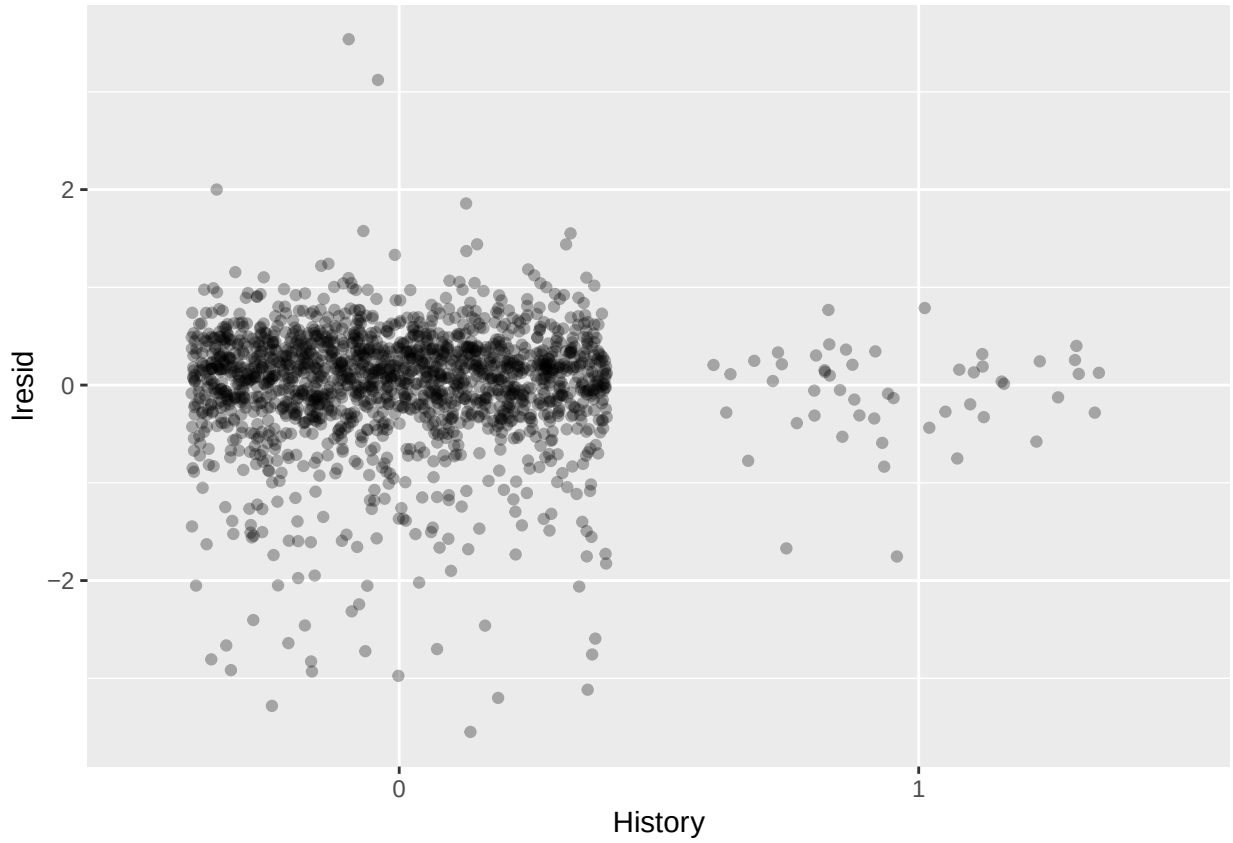

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



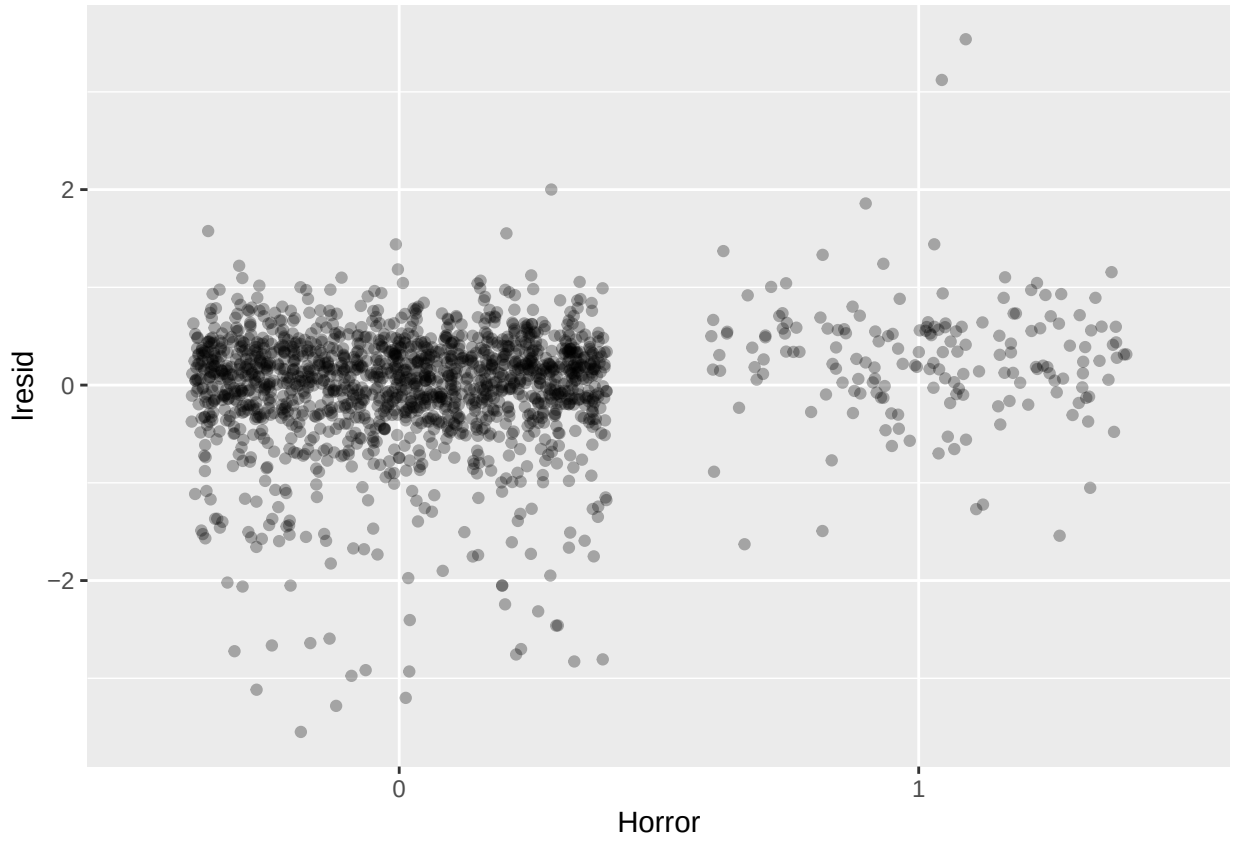

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



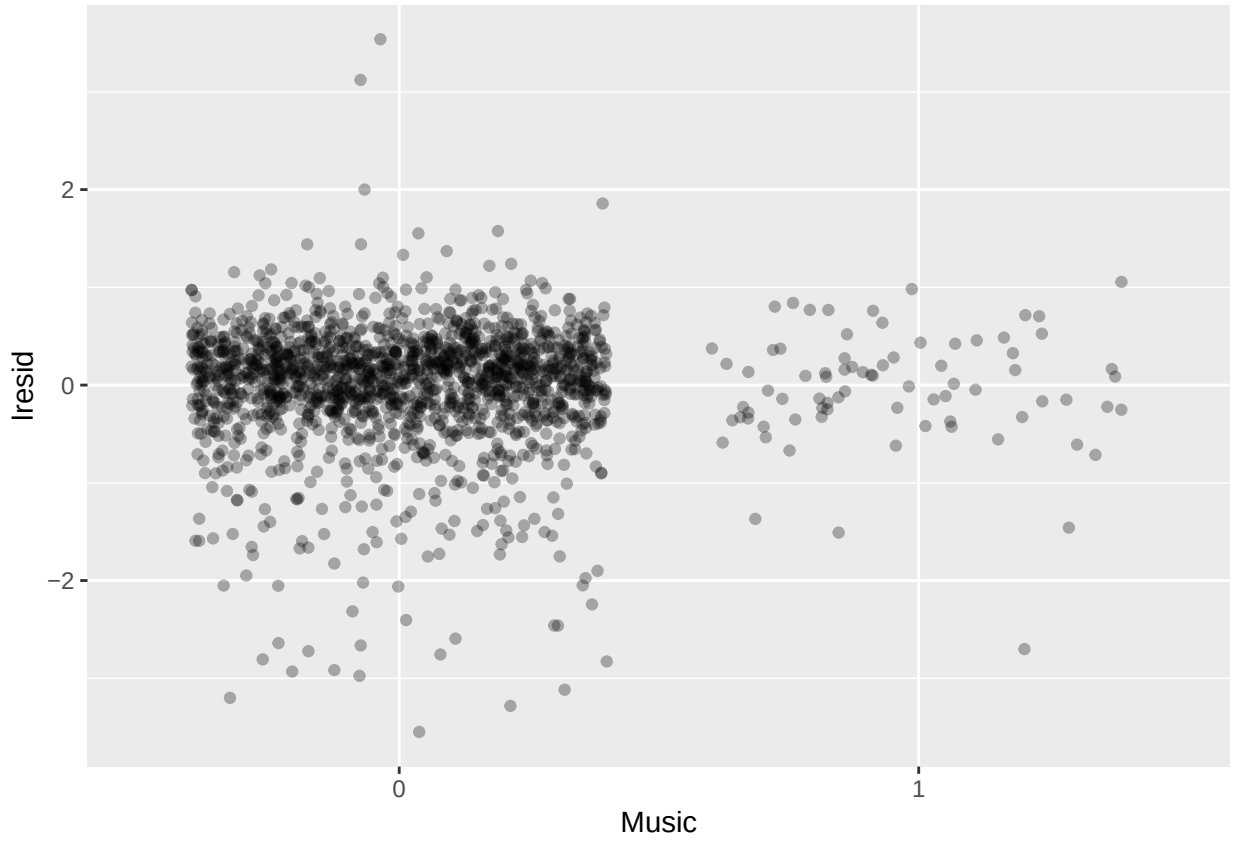

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



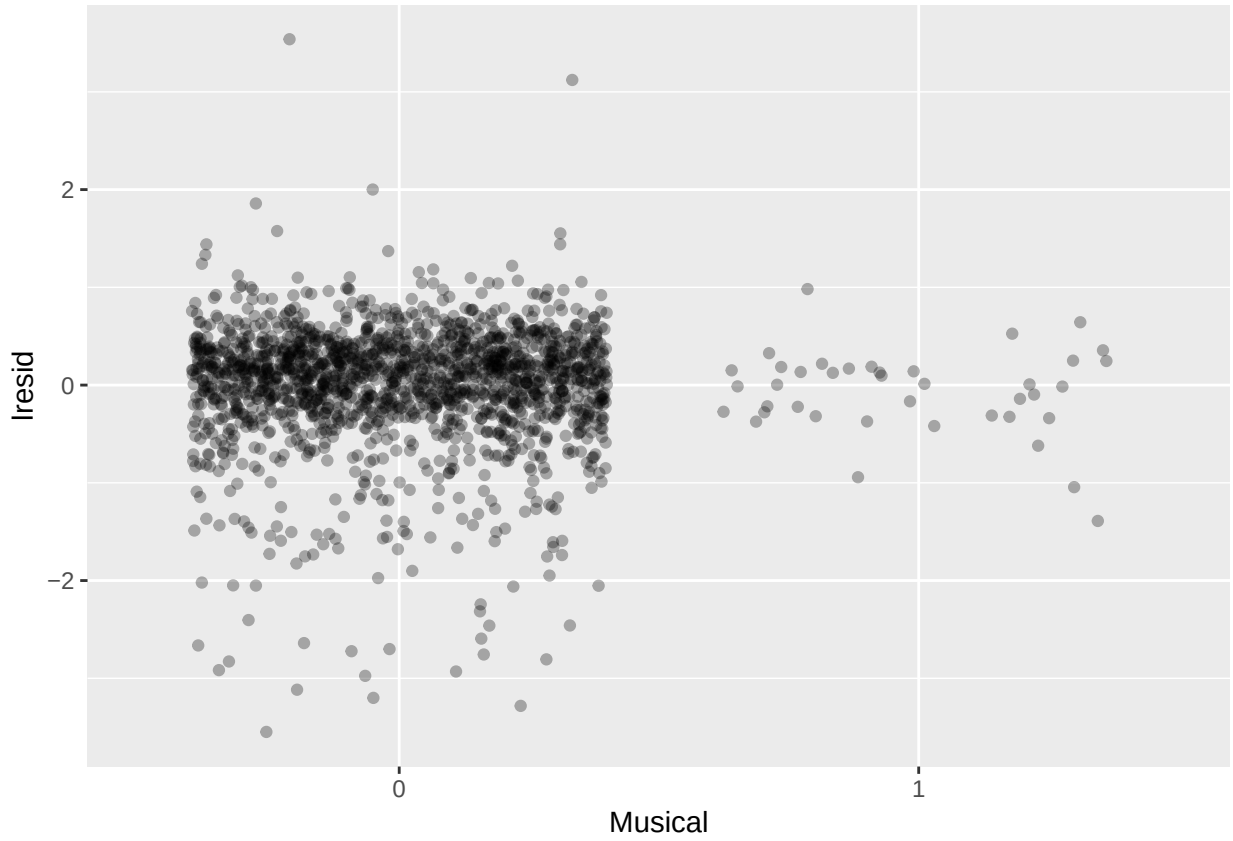

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



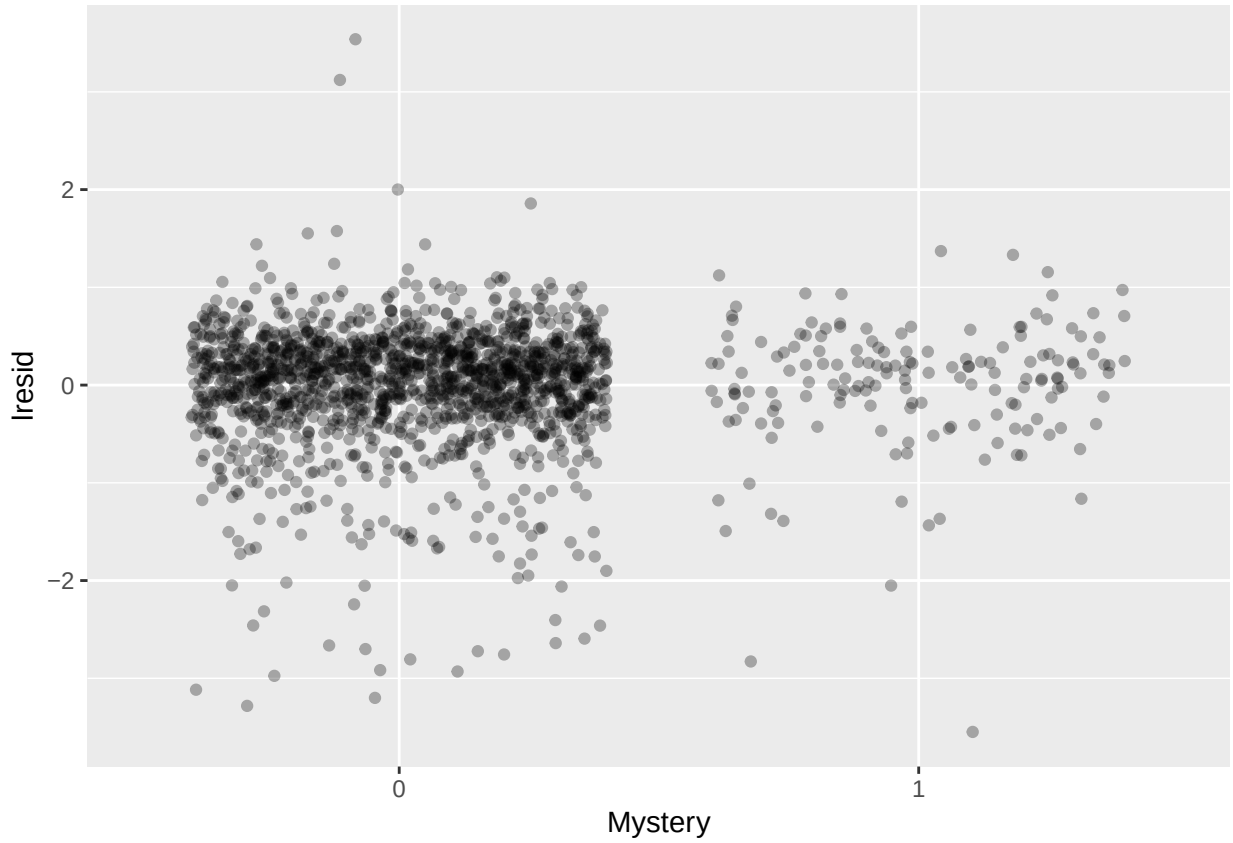

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



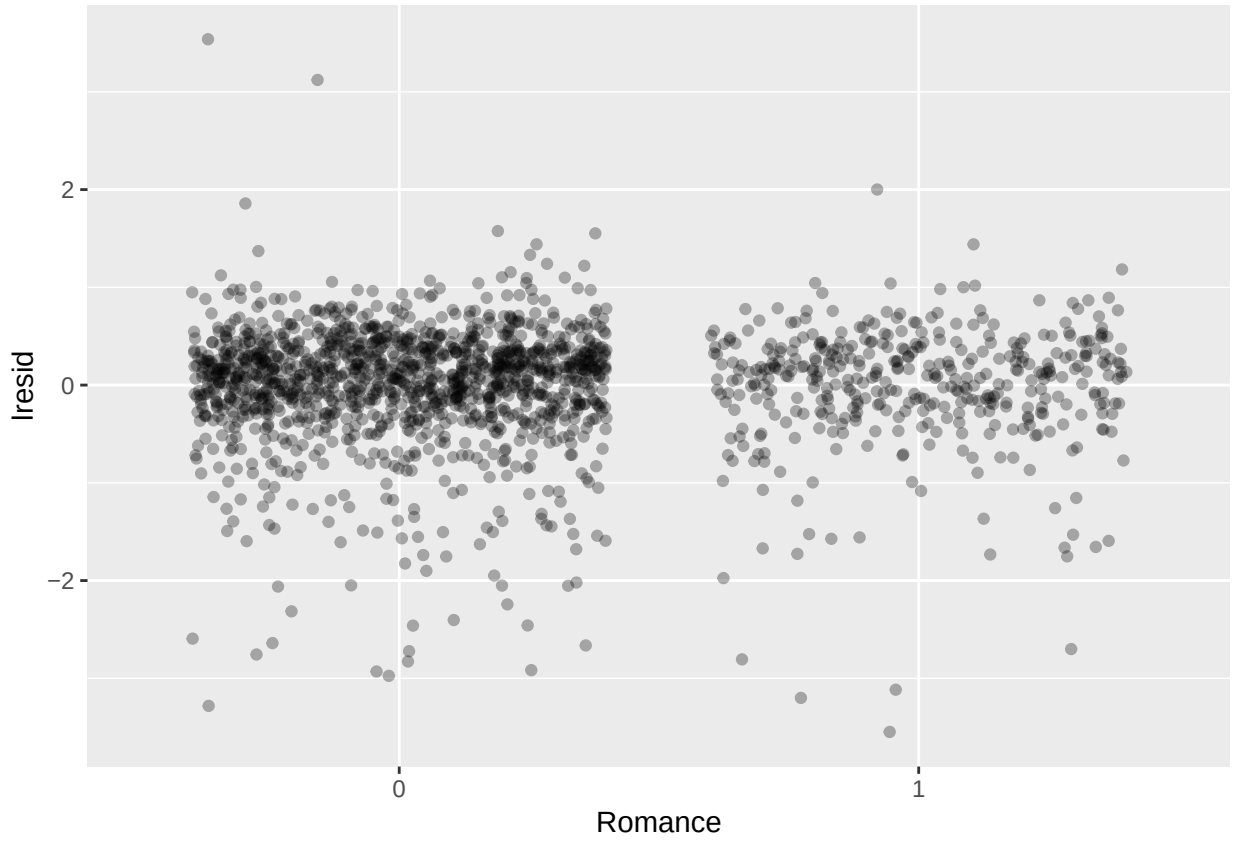

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



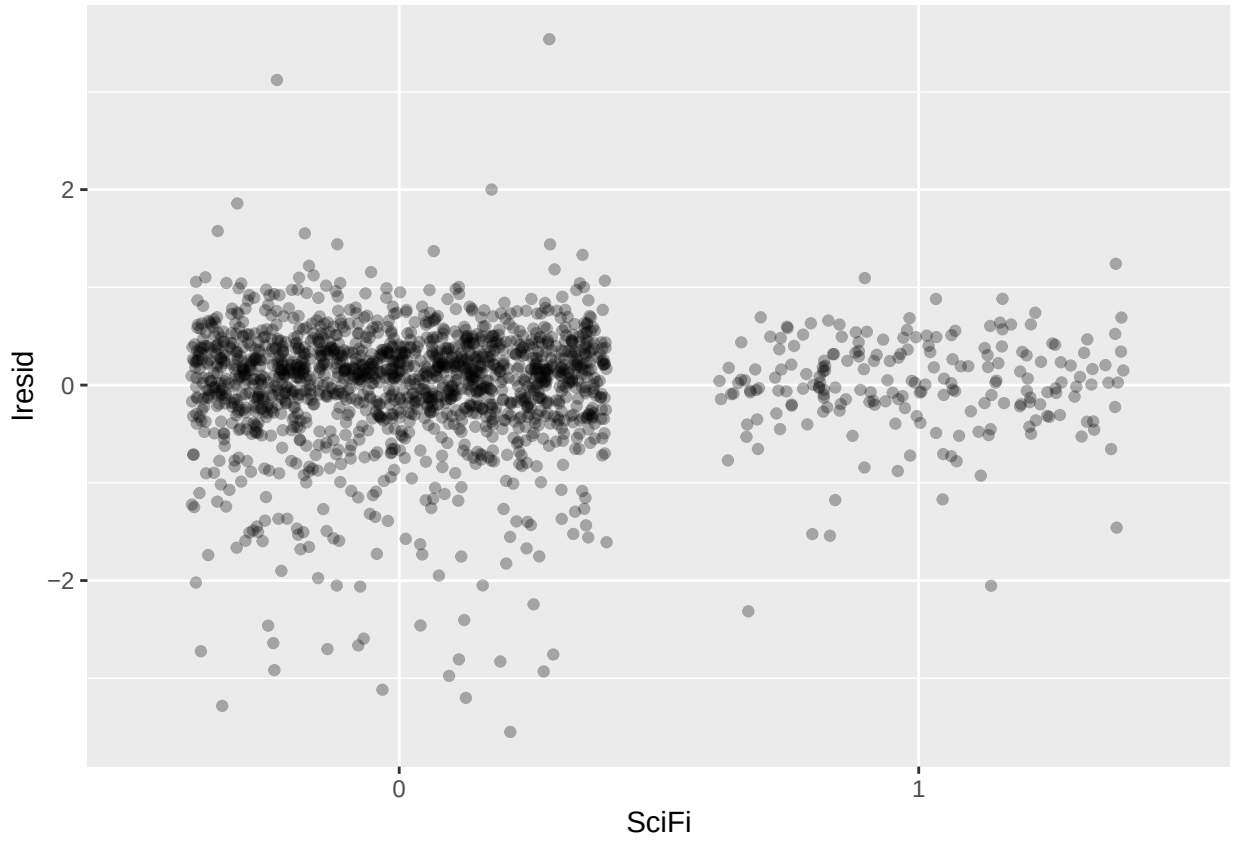

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



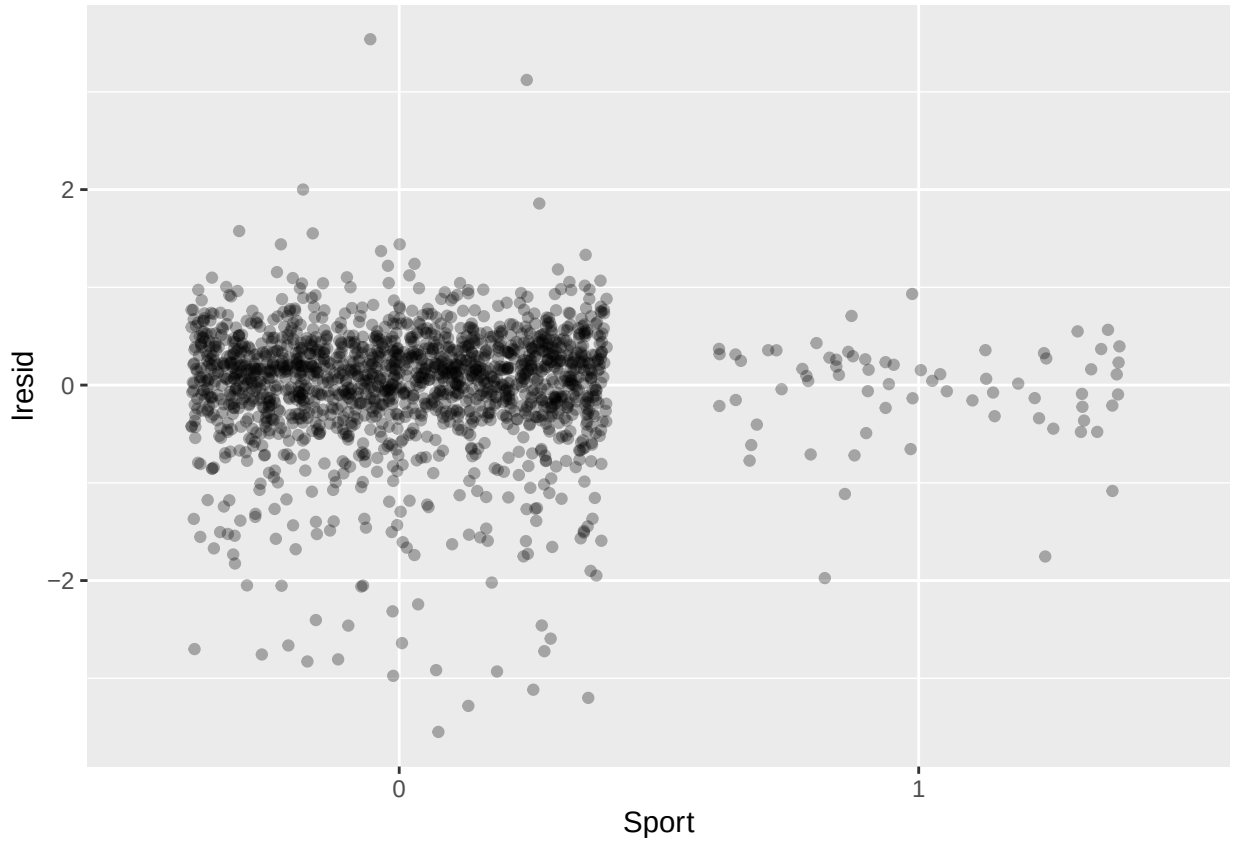

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



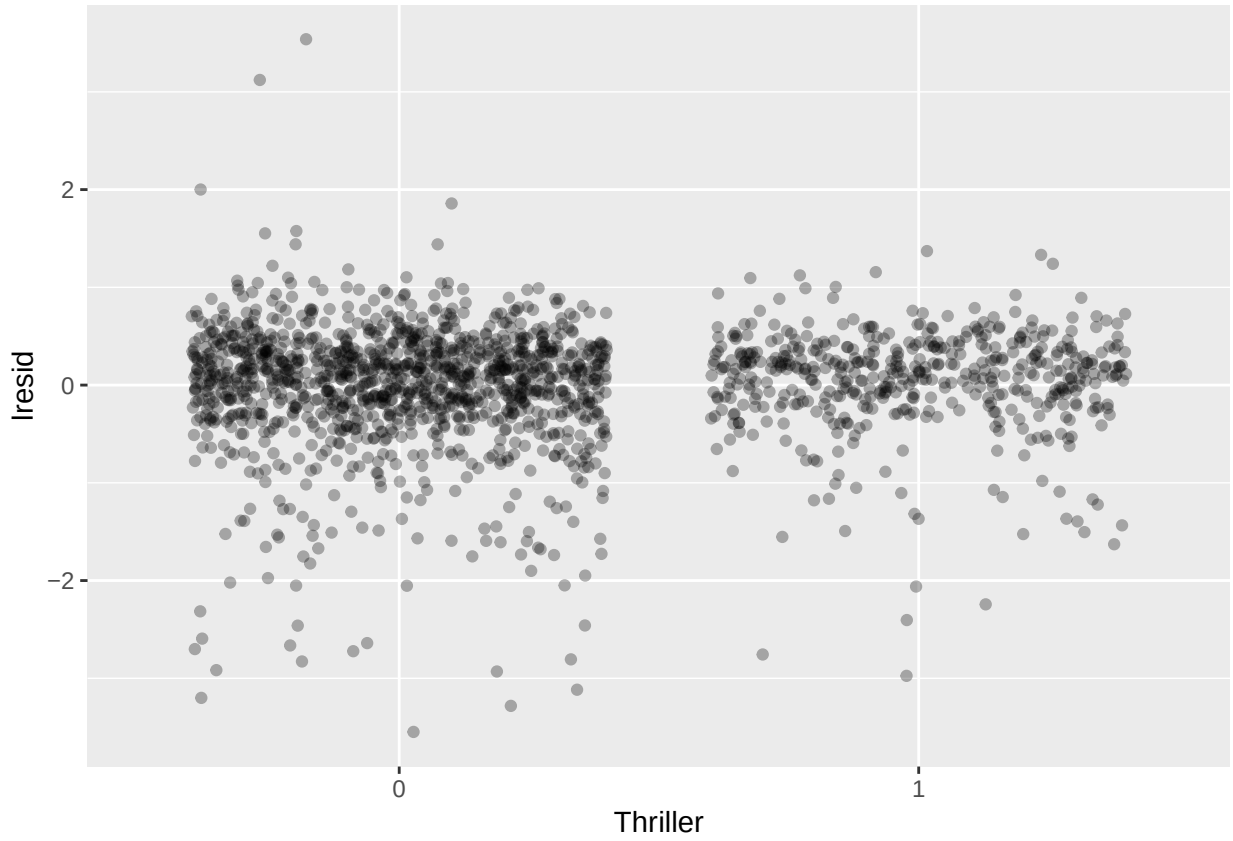

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



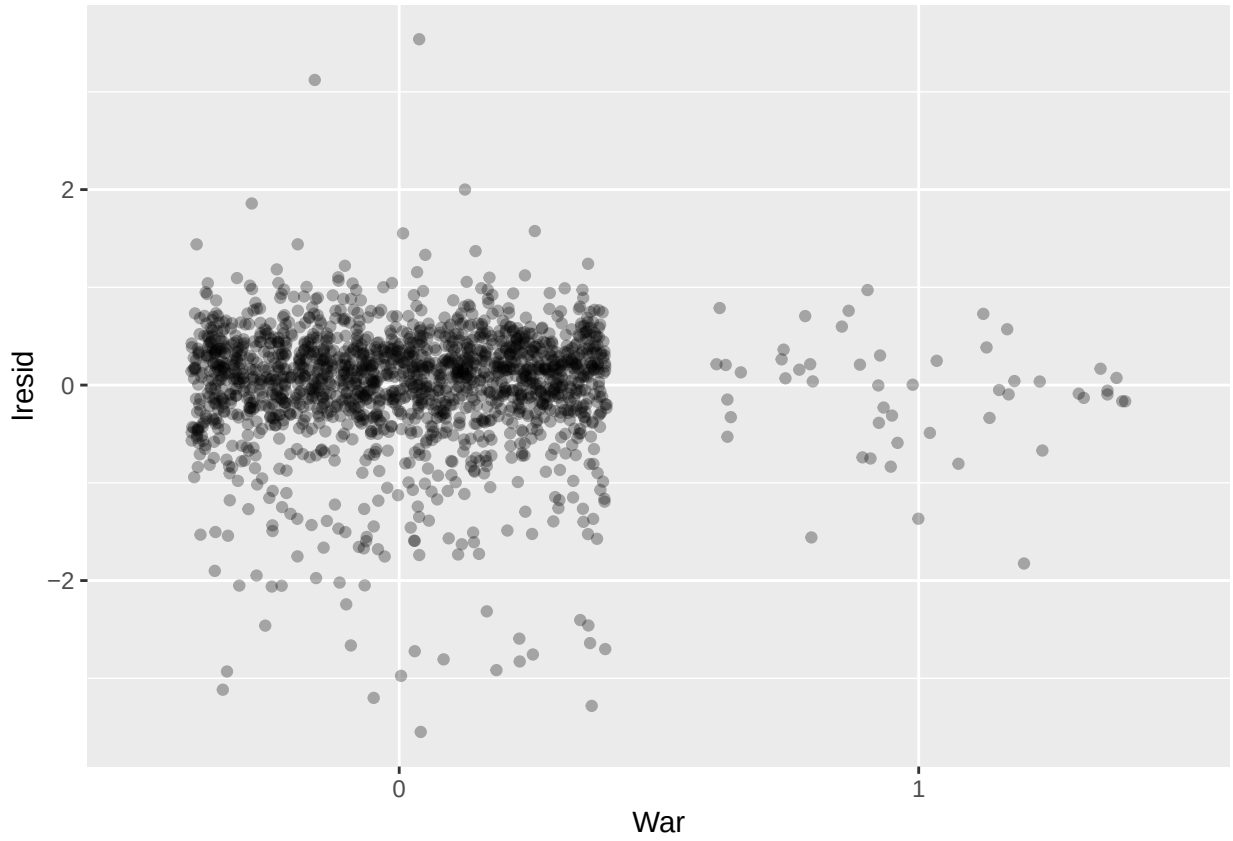

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



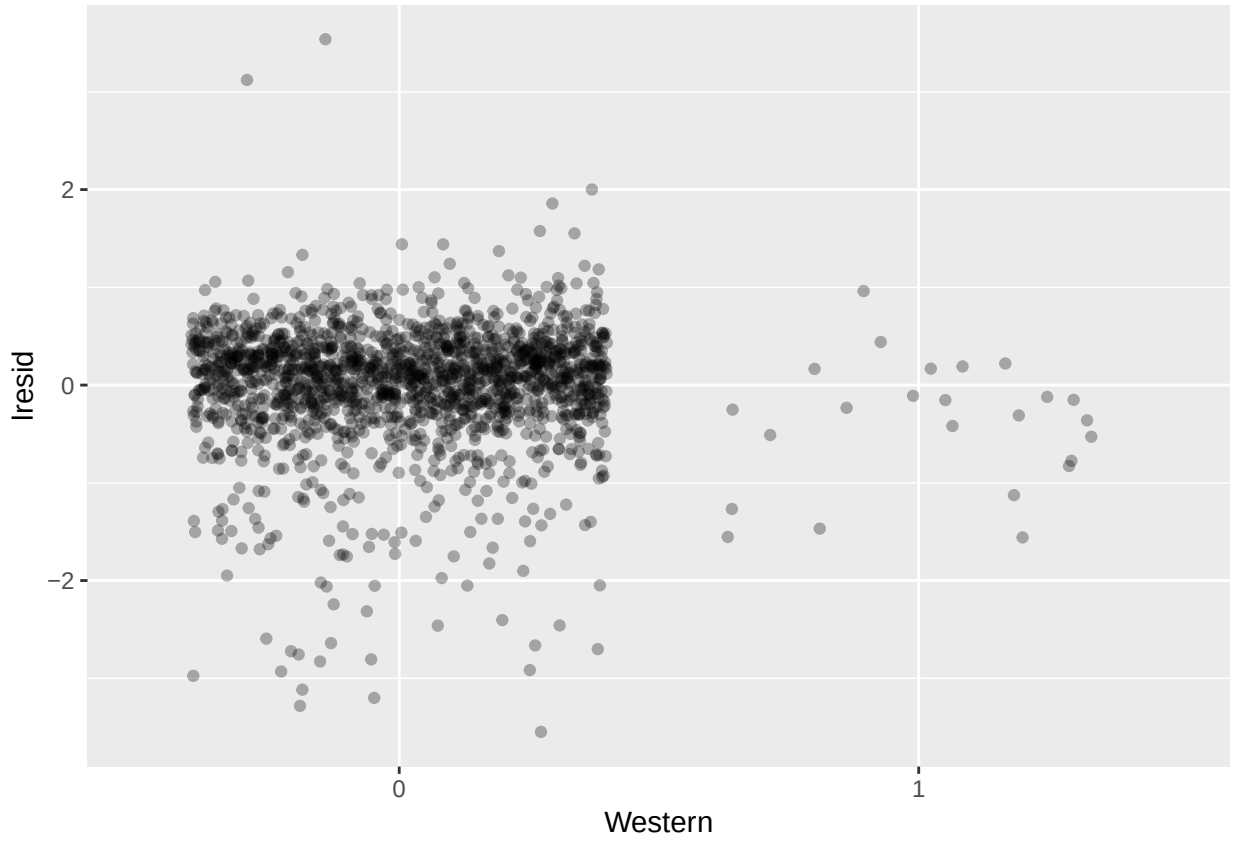

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



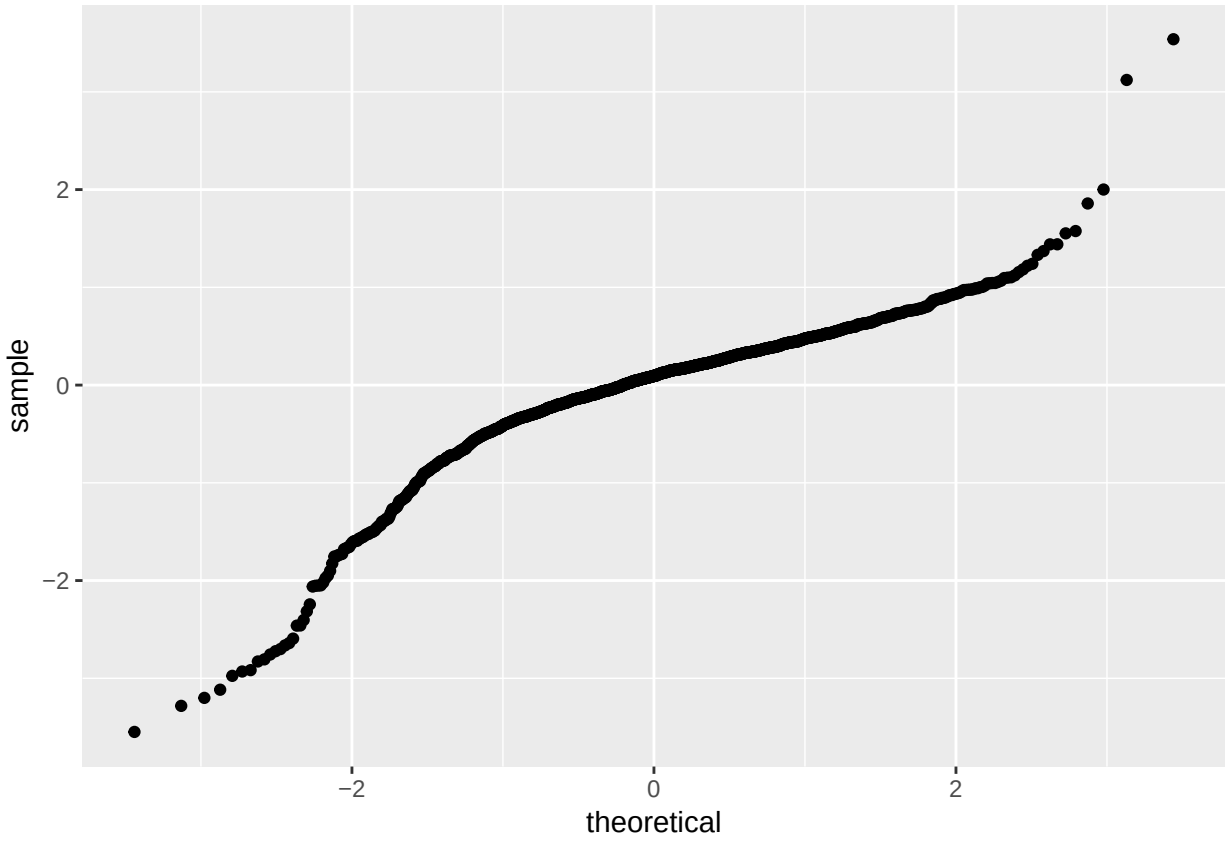

```
## Warning: Removed 132 rows containing missing values (geom_point).
```




```
## Warning: Removed 132 rows containing missing values (geom_point).
```




```
## Warning: Removed 132 rows containing non-finite values (stat_qq).
```




```
# again blair and paranormal
train %>%
  add_residuals(mod_all2, 'lresid') %>%
  filter(lresid > 2.1)
```

```
## # A tibble: 2 x 58
##   director_name num_critic_for_~ director_facebo~ actor_3_faceboo~
##   <chr>                    <dbl>            <dbl>            <dbl>
## 1 Daniel Myrick              360               19               39
## 2 Oren Peli                  409              110               21
## # ... with 54 more variables: actor_2_name <chr>,
## #   actor_1_facebook_likes <dbl>, gross <dbl>, genres <chr>,
## #   actor_1_name <chr>, movie_title <chr>,
## #   cast_total_facebook_likes <dbl>, actor_3_name <chr>,
## #   num_user_for_reviews <dbl>, language <chr>, country <chr>,
## #   content_rating <fct>, budget <dbl>, year <fct>,
## #   actor_2_facebook_likes <dbl>, imdb_score <dbl>,
## #   movie_facebook_likes <dbl>, Action <fct>, Adventure <fct>,
## #   Animation <fct>, Biography <fct>, Comedy <fct>, Crime <fct>,
## #   Documentary <fct>, Drama <fct>, Family <fct>, Fantasy <fct>,
## #   History <fct>, Horror <fct>, Music <fct>, Musical <fct>,
## #   Mystery <fct>, Romance <fct>, SciFi <fct>, Sport <fct>,
## #   Thriller <fct>, War <fct>, Western <fct>, gdpd <dbl>,
## #   real_budget <dbl>, real_gross <dbl>, rev_budget <dbl>,
## #   profit_dum <dbl>, total_oscars_actor1 <dbl>,
## #   total_oscars_actor2 <dbl>, total_oscars_actor3 <dbl>,
## #   total_oscars_director <fct>, total_oscars_actor <fct>,
```

```
## #   real_gross_log <dbl>, real_budget_log <dbl>,
## #   director_facebook_likes_log <dbl>,
## #   cast_total_facebook_likes_log <dbl>, imdb_score_log <dbl>,
## #   lresid <dbl>
```

```
train %>%
  add_residuals(mod_all2, 'lresid') %>%
  filter(lresid < -2)
```

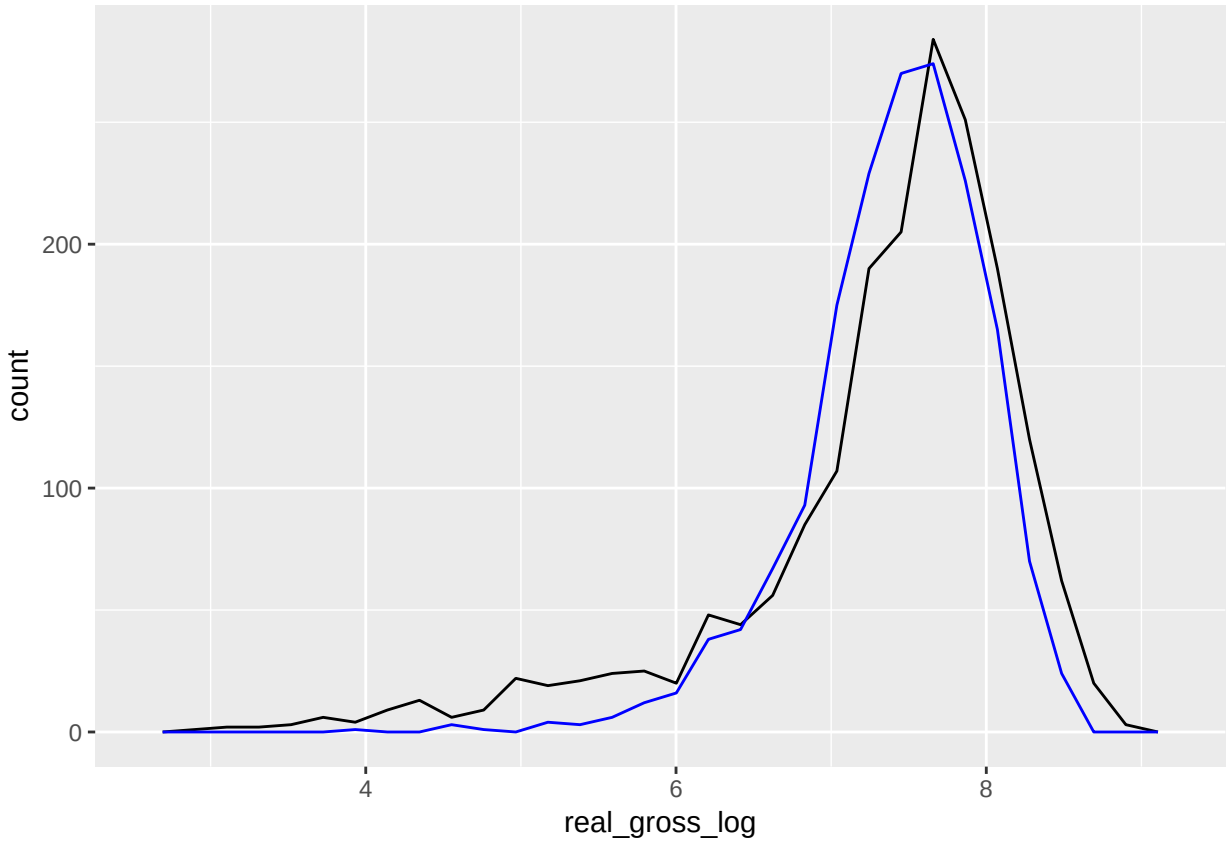
```
## # A tibble: 25 x 58
##    director_name num_critic_for_~ director_facebo~ actor_3_faceboo~
##    <chr>                   <dbl>            <dbl>            <dbl>
##  1 Akiva Goldsm~             189              167              778
##  2 Nick Cassave~             177              415              545
##  3 John Hillcoat             355              214             3000
##  4 George Gallo               45              269              384
##  5 John Duigan                16               18              186
##  6 Daniel Algra~              42                3              442
##  7 Anand Tucker              134               14              356
##  8 Jake Paltrow               50               17               66
##  9 Timothy Hines               1                0              247
## 10 Michael Mere~              23                7              875
## # ... with 15 more rows, and 54 more variables: actor_2_name <chr>,
## #   actor_1_facebook_likes <dbl>, gross <dbl>, genres <chr>,
## #   actor_1_name <chr>, movie_title <chr>,
## #   cast_total_facebook_likes <dbl>, actor_3_name <chr>,
## #   num_user_for_reviews <dbl>, language <chr>, country <chr>,
## #   content_rating <fct>, budget <dbl>, year <fct>,
## #   actor_2_facebook_likes <dbl>, imdb_score <dbl>,
## #   movie_facebook_likes <dbl>, Action <fct>, Adventure <fct>,
## #   Animation <fct>, Biography <fct>, Comedy <fct>, Crime <fct>,
## #   Documentary <fct>, Drama <fct>, Family <fct>, Fantasy <fct>,
## #   History <fct>, Horror <fct>, Music <fct>, Musical <fct>,
## #   Mystery <fct>, Romance <fct>, SciFi <fct>, Sport <fct>,
## #   Thriller <fct>, War <fct>, Western <fct>, gdpd <dbl>,
## #   real_budget <dbl>, real_gross <dbl>, rev_budget <dbl>,
## #   profit_dum <dbl>, total_oscars_actor1 <dbl>,
## #   total_oscars_actor2 <dbl>, total_oscars_actor3 <dbl>,
## #   total_oscars_director <fct>, total_oscars_actor <fct>,
## #   real_gross_log <dbl>, real_budget_log <dbl>,
## #   director_facebook_likes_log <dbl>,
## #   cast_total_facebook_likes_log <dbl>, imdb_score_log <dbl>,
## #   lresid <dbl>
```

Prediction

```
train %>%
  add_predictions(mod_all2, 'lpred') %>%
  ggplot() +
  geom_freqpoly(aes(x = real_gross_log)) +
  geom_freqpoly(aes(x = lpred), color = 'blue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 132 rows containing non-finite values (stat_bin).
```



Fit really simple model: Anova compare

Budget always very significant and decreases RMSE significantly.
Use this to compare with other more complex models via arima

```
# filter so that same sample size as mod_all models so can compare
mod_simple <- lm(real_gross_log ~ real_budget_log, data = train %>% filter(!is.na(content_rating)))

# anova does show that the more complex models do explain real_gross_log better than simple
anova(mod_all, mod_simple)
```

```
## Analysis of Variance Table
##
## Model 1: real_gross_log ~ Adventure + Action + Family + Mystery + Documentary +
##     Drama + History + Romance + real_budget_log + imdb_score_log +
##     year + content_rating
## Model 2: real_gross_log ~ real_budget_log
##   Res.Df    RSS  Df Sum of Sq      F    Pr(>F)
## 1   1668 614.85
## 2   1717 725.98 -49   -111.12 6.1524 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(mod_all2, mod_simple)
```

```
## Analysis of Variance Table
```

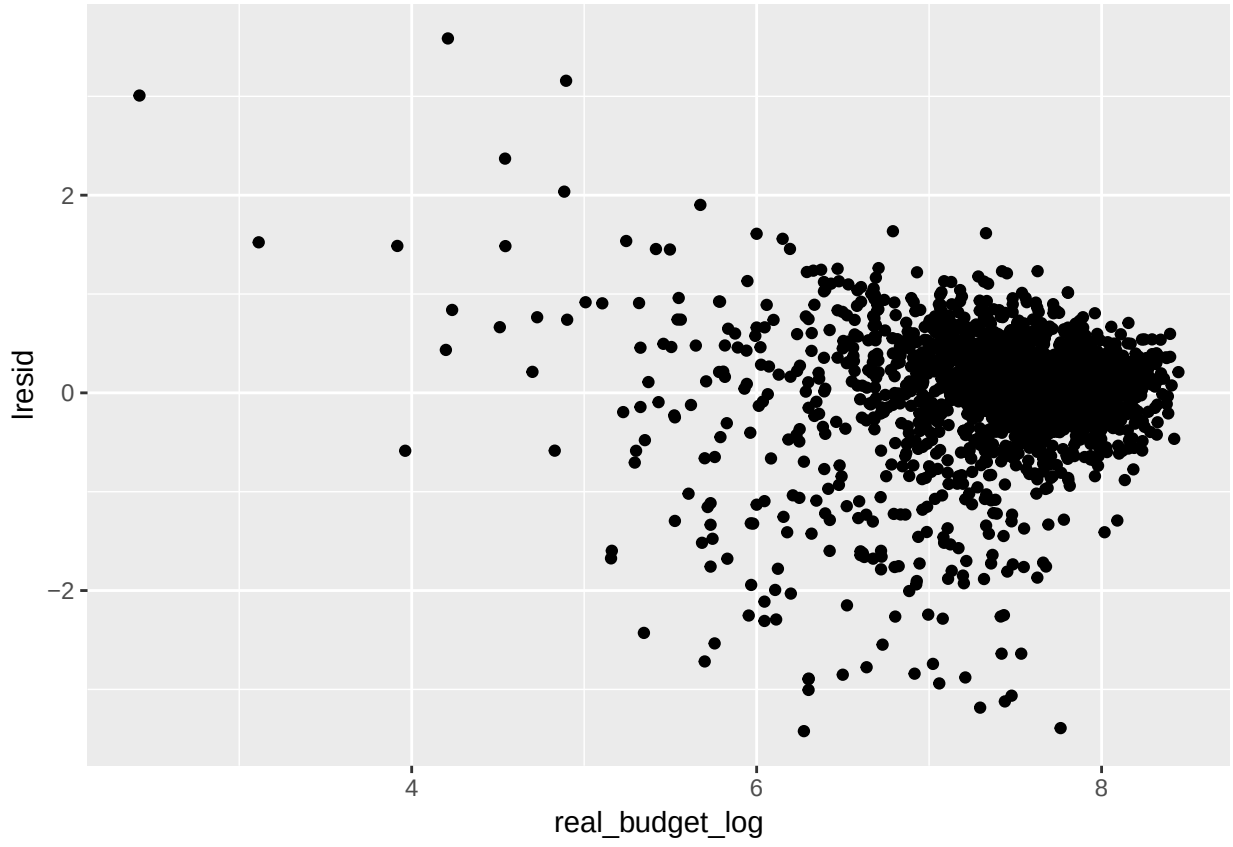
```
##
## Model 1: real_gross_log ~ real_budget_log + imdb_score_log + year + Comedy +
##     content_rating + Mystery
## Model 2: real_gross_log ~ real_budget_log
##   Res.Df    RSS  Df Sum of Sq      F    Pr(>F)
## 1   1674 644.16
## 2   1717 725.98 -43   -81.819 4.9448 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Start with budget

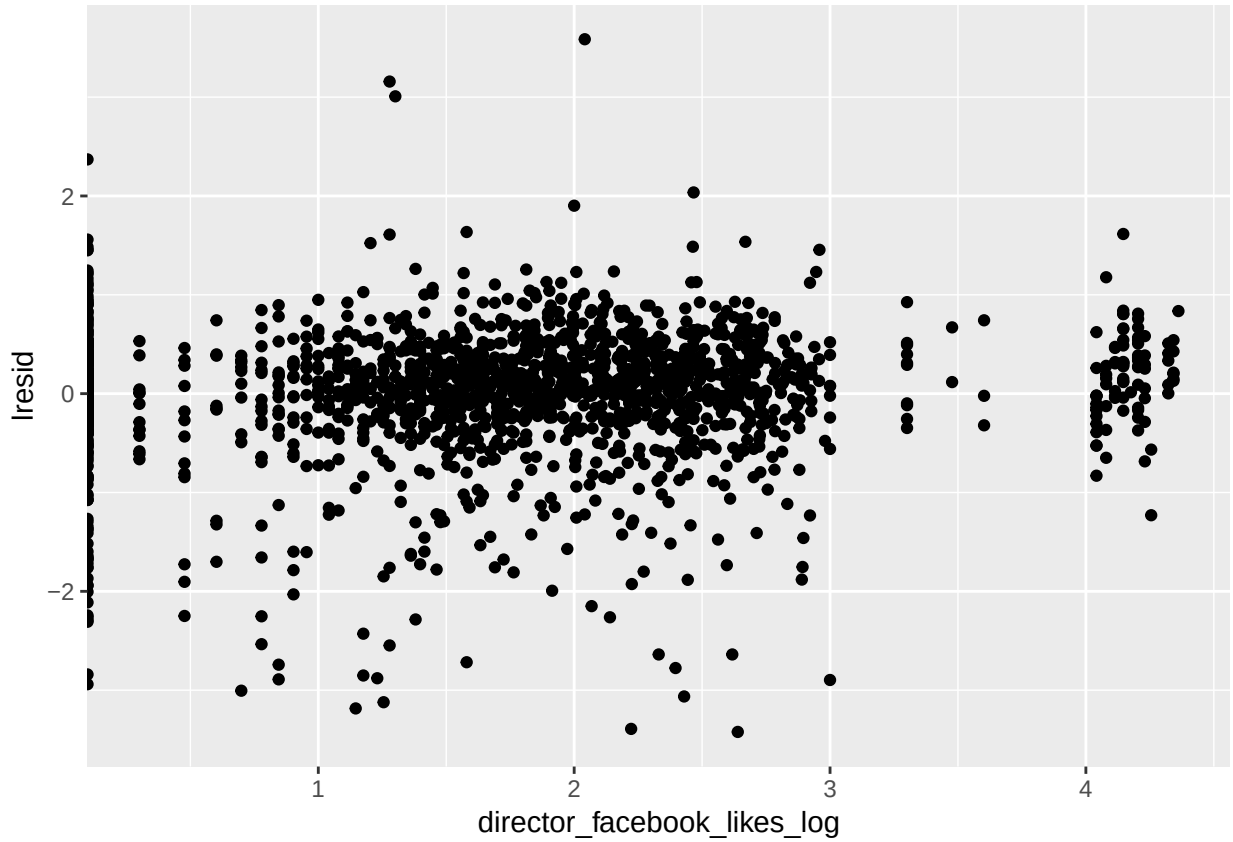
Look at residual plots and determine what other variables should be added
Budget has the most obvious relationship with revenue, both graphically and logically

```
mod_simple <- lm(real_gross_log ~ real_budget_log, data = train)

gr_resid(mod_simple)
```

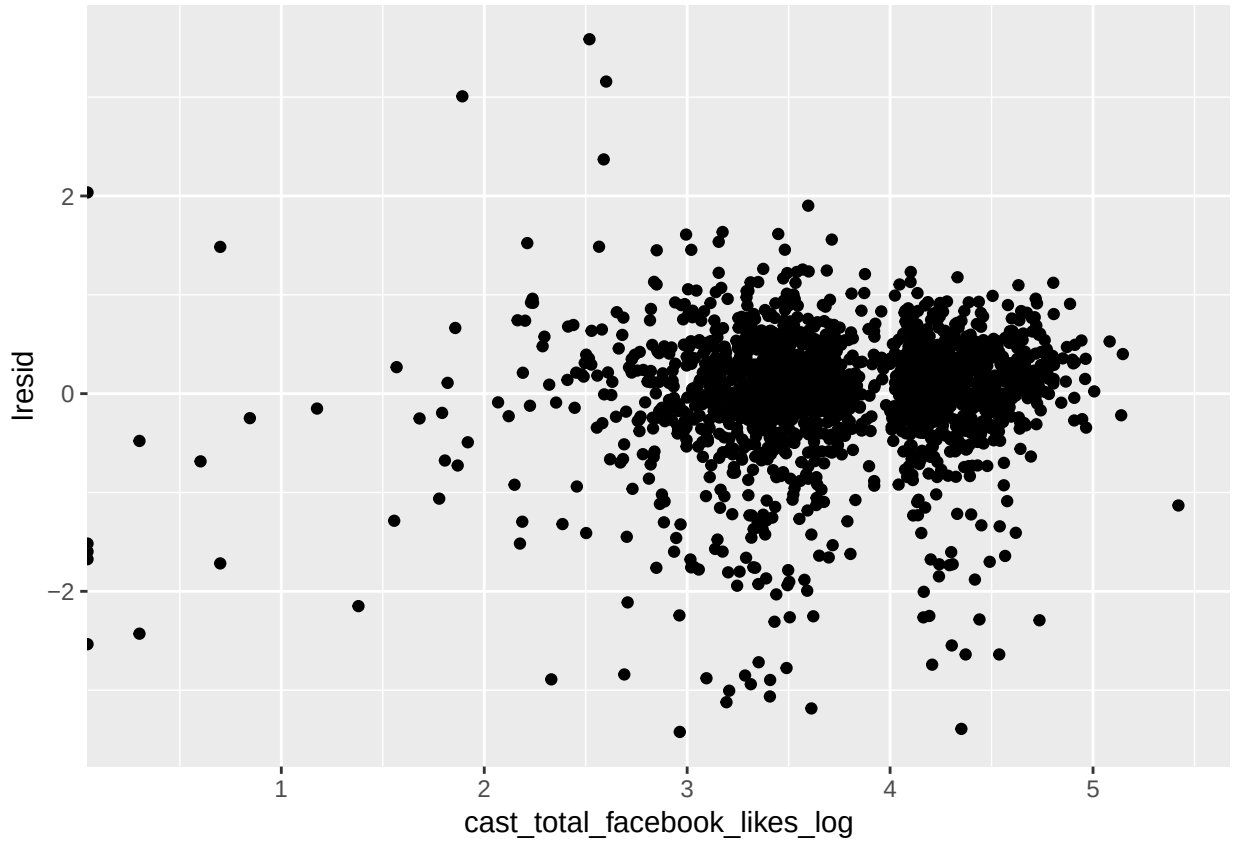
```
## Warning: Removed 102 rows containing missing values (geom_point).
```



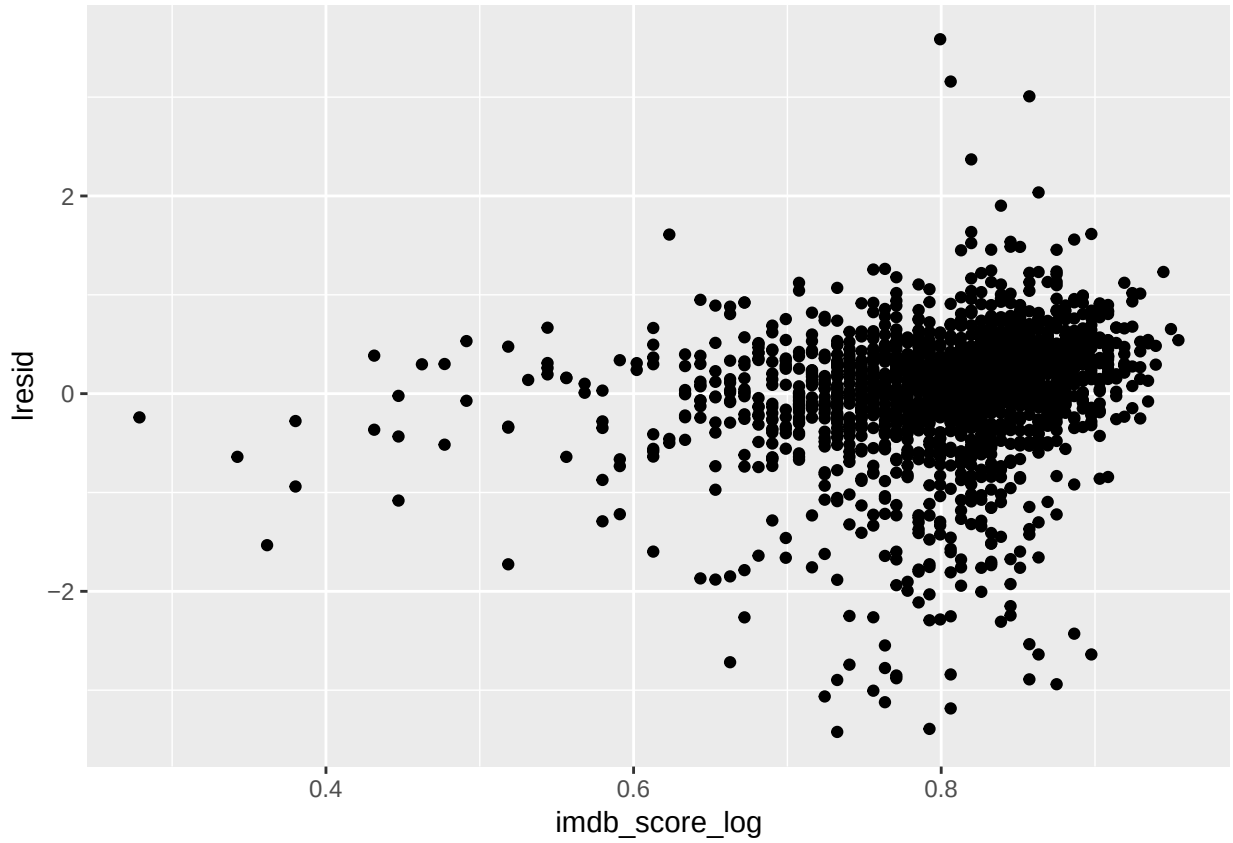

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



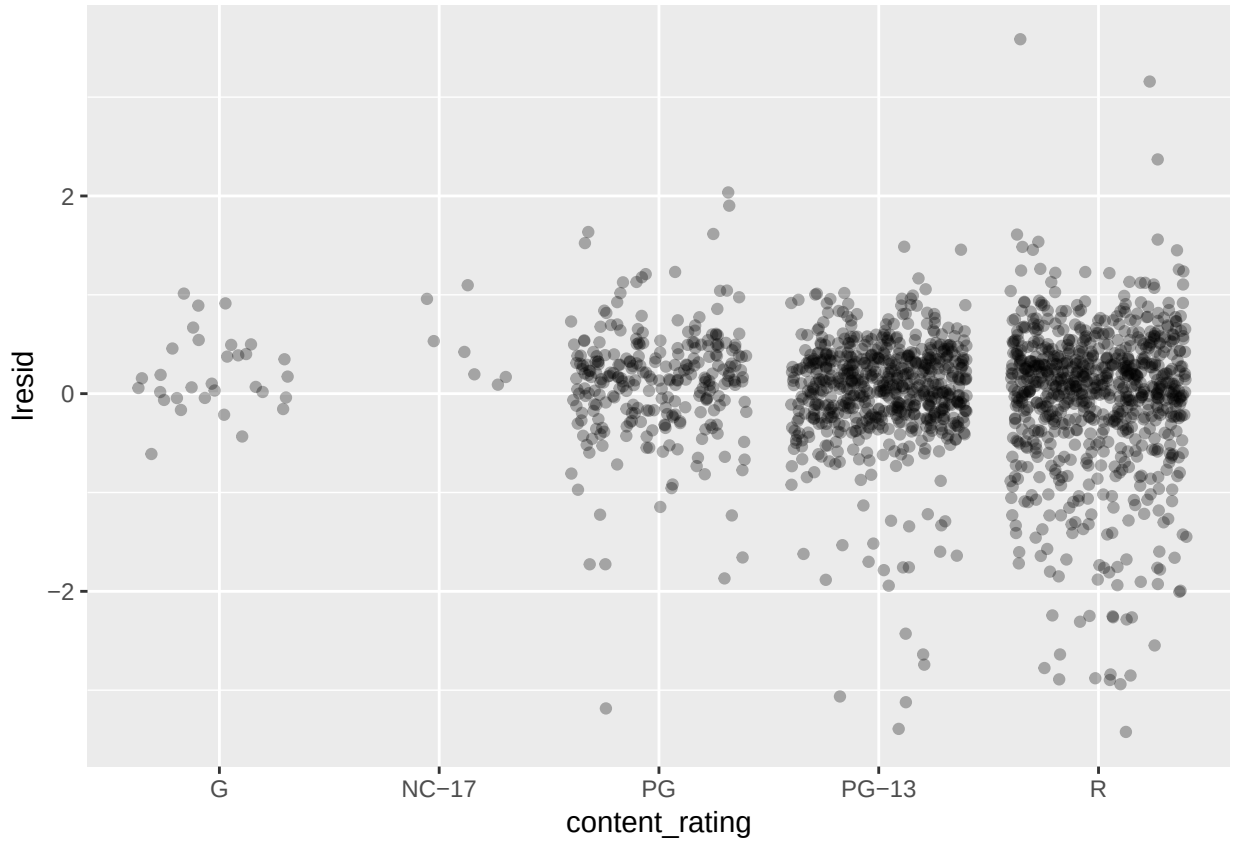

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



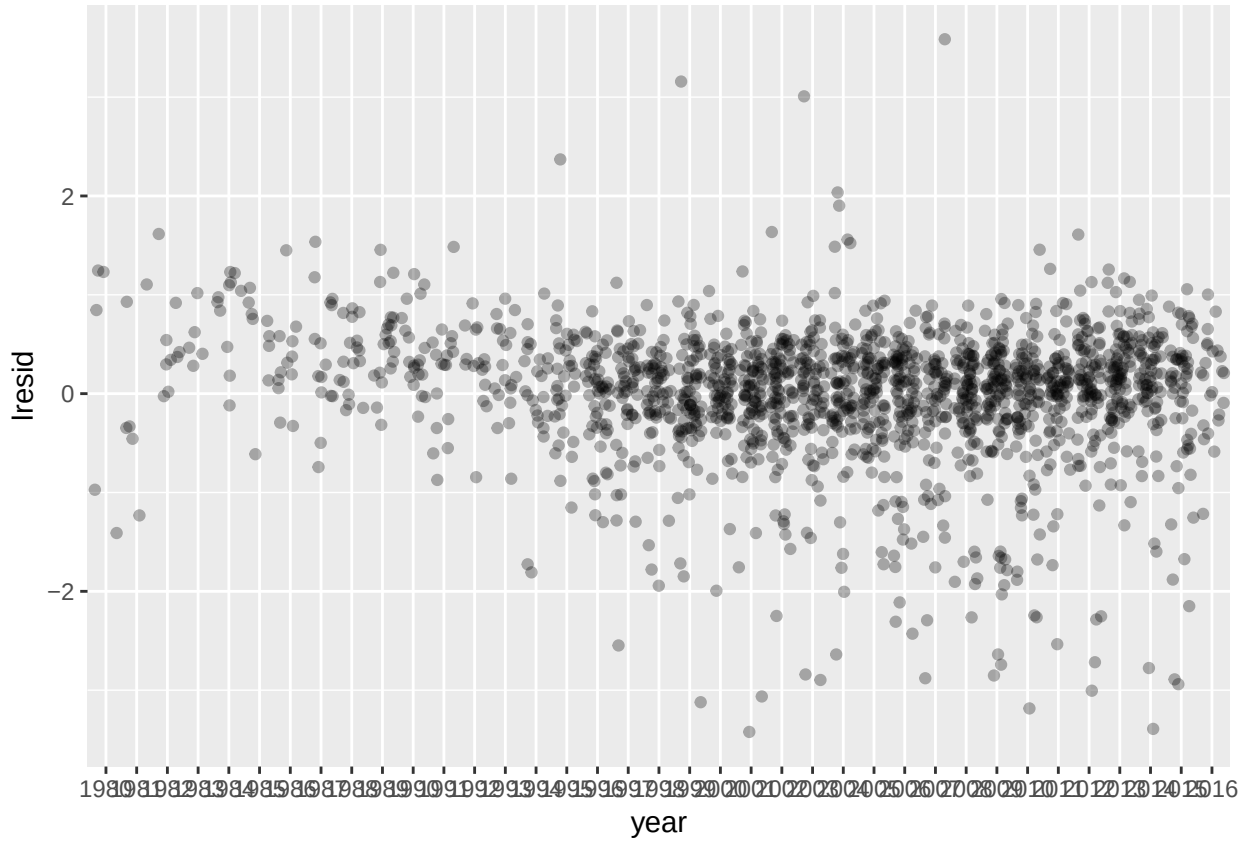

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



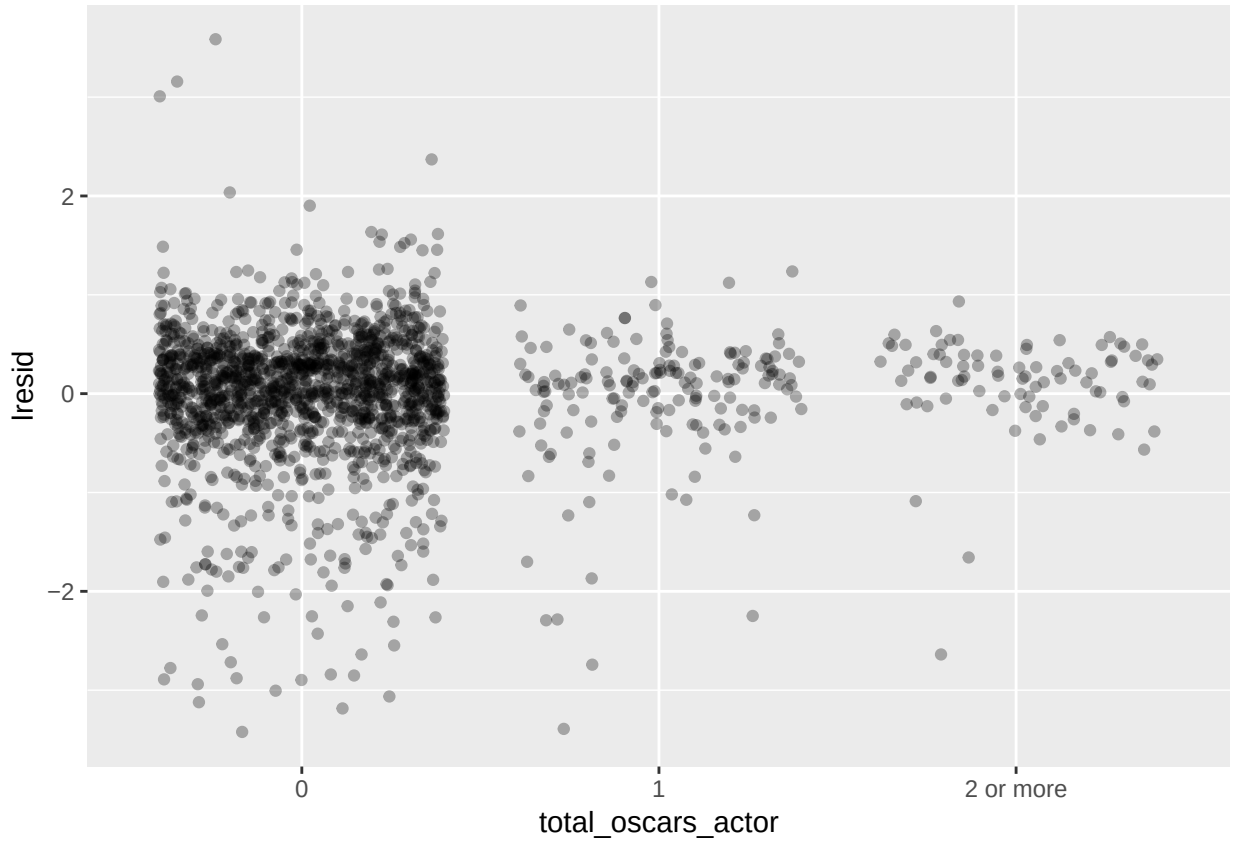

```
## Warning: Removed 94 rows containing missing values (geom_point).
```



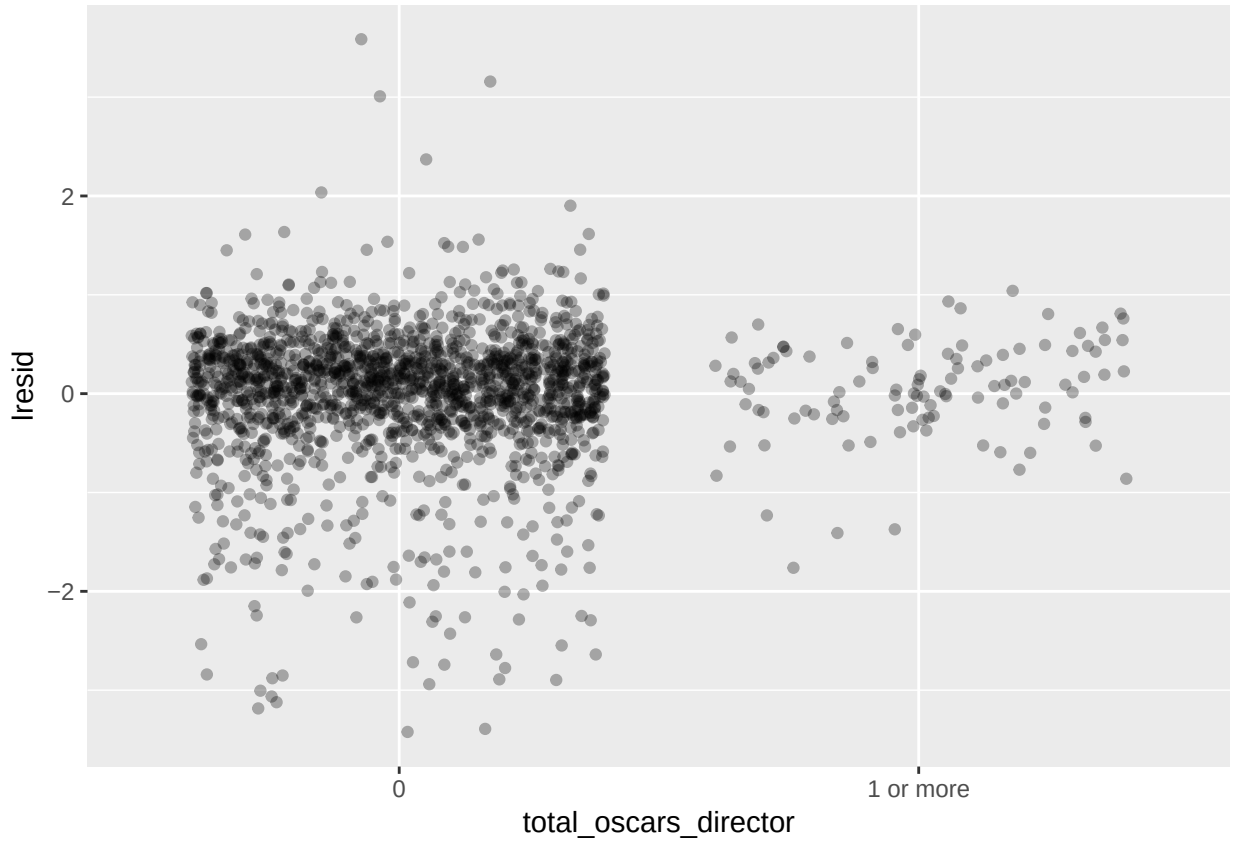

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



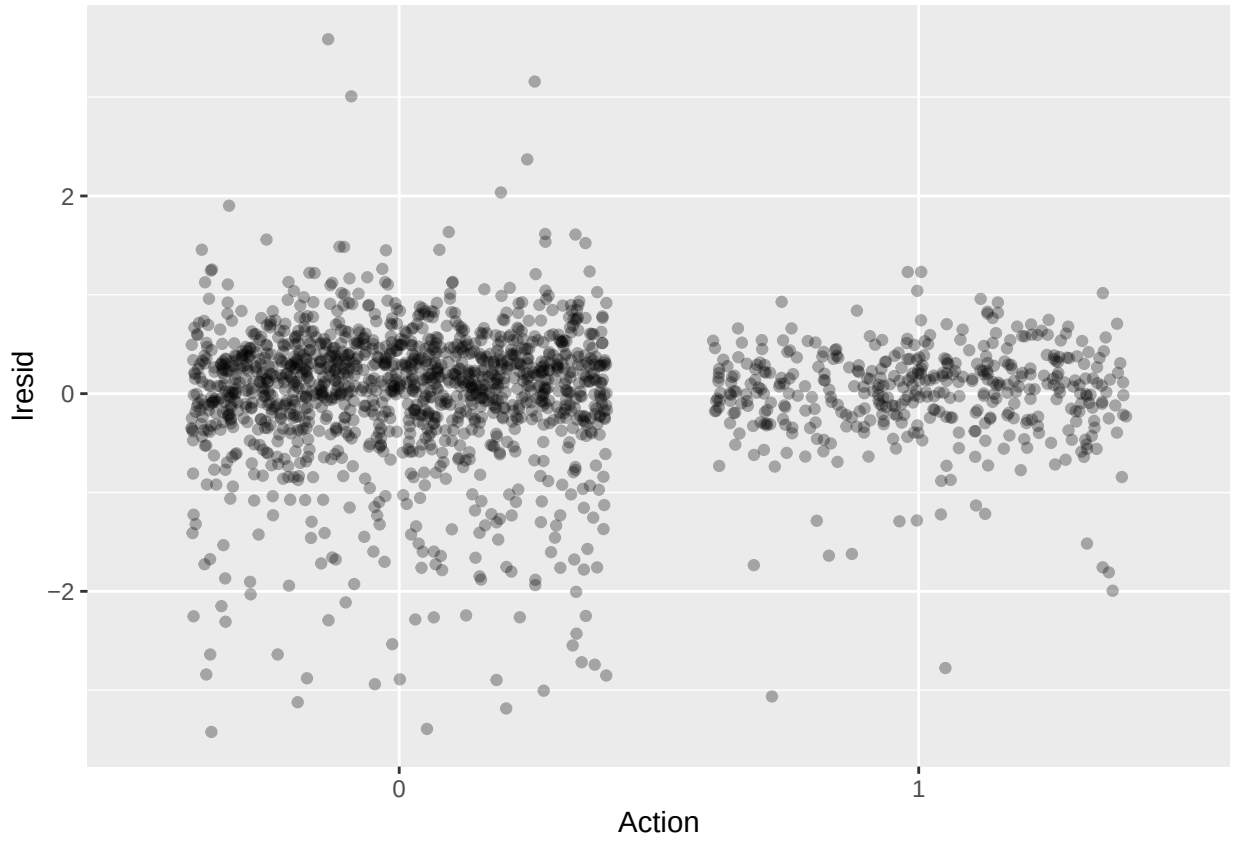

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



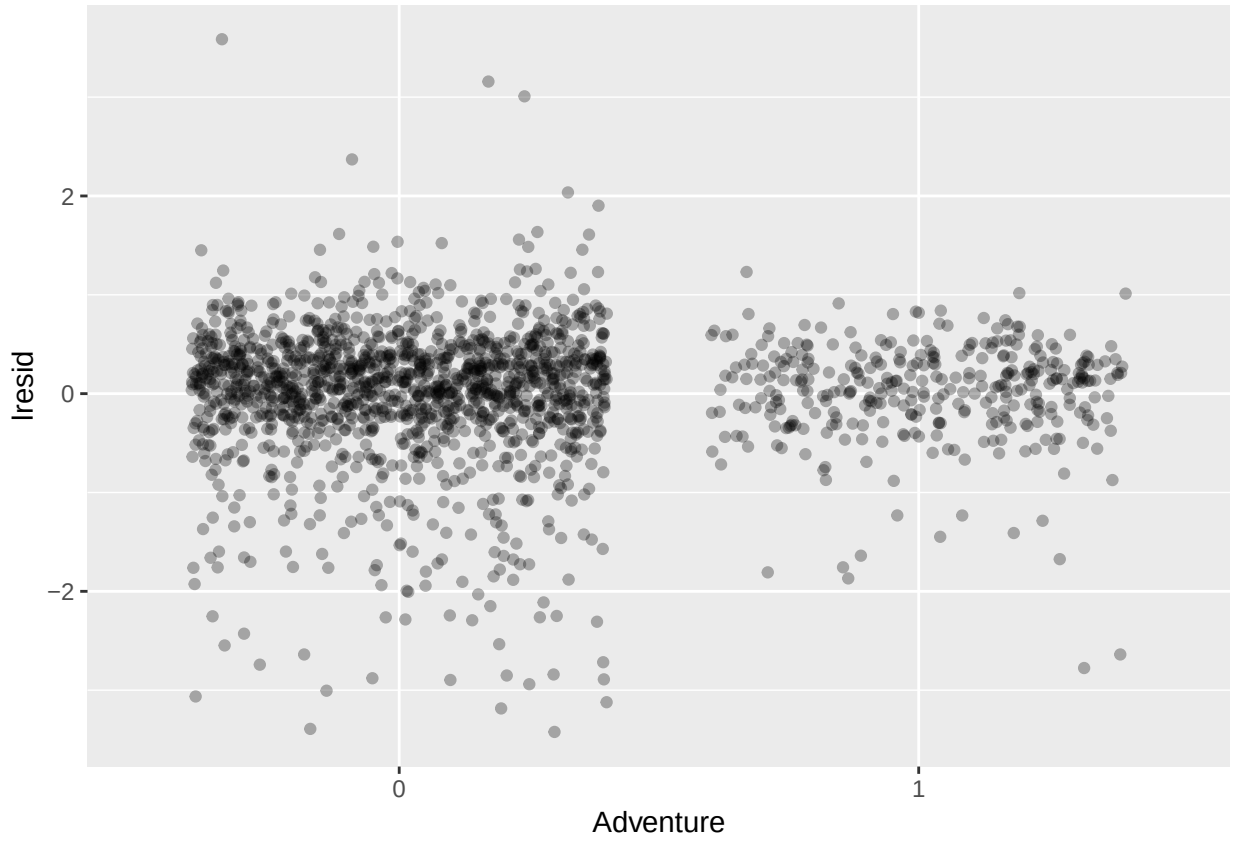

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



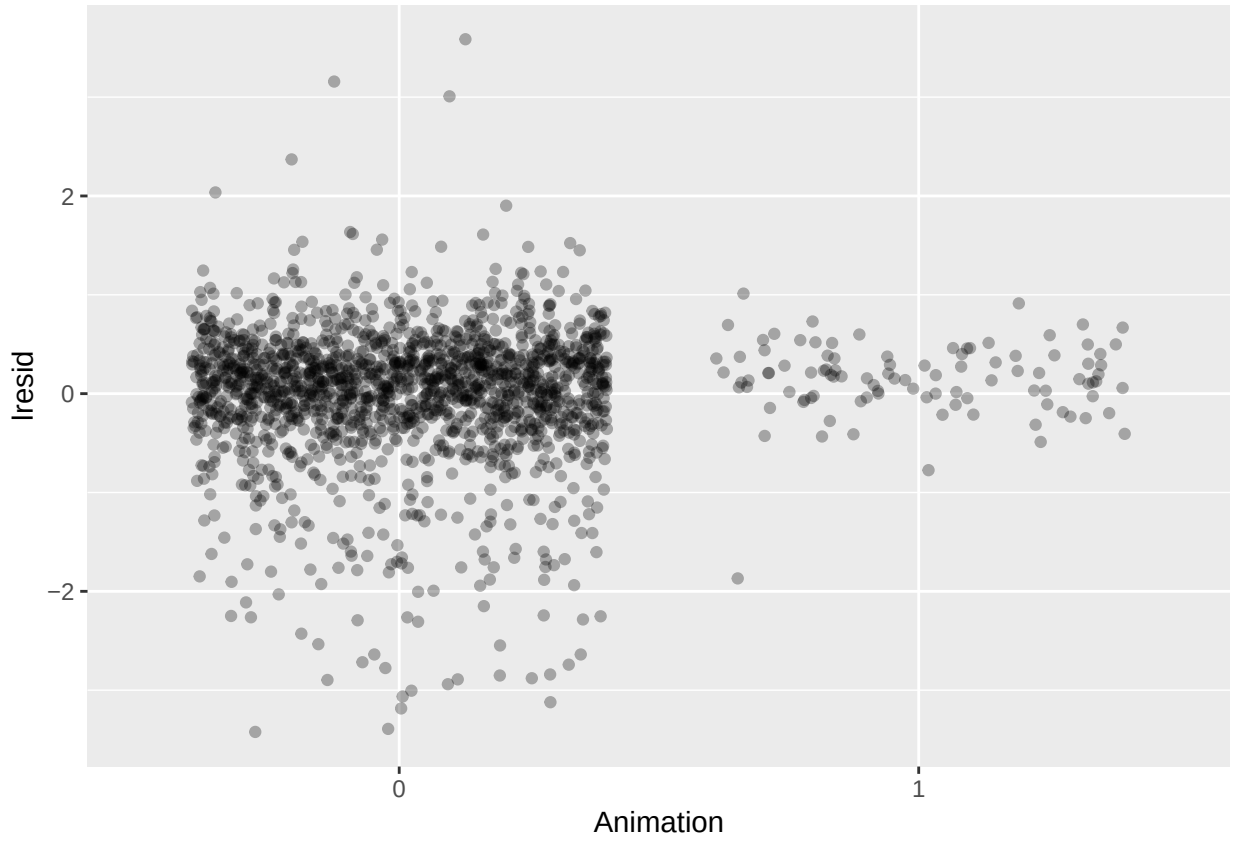

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



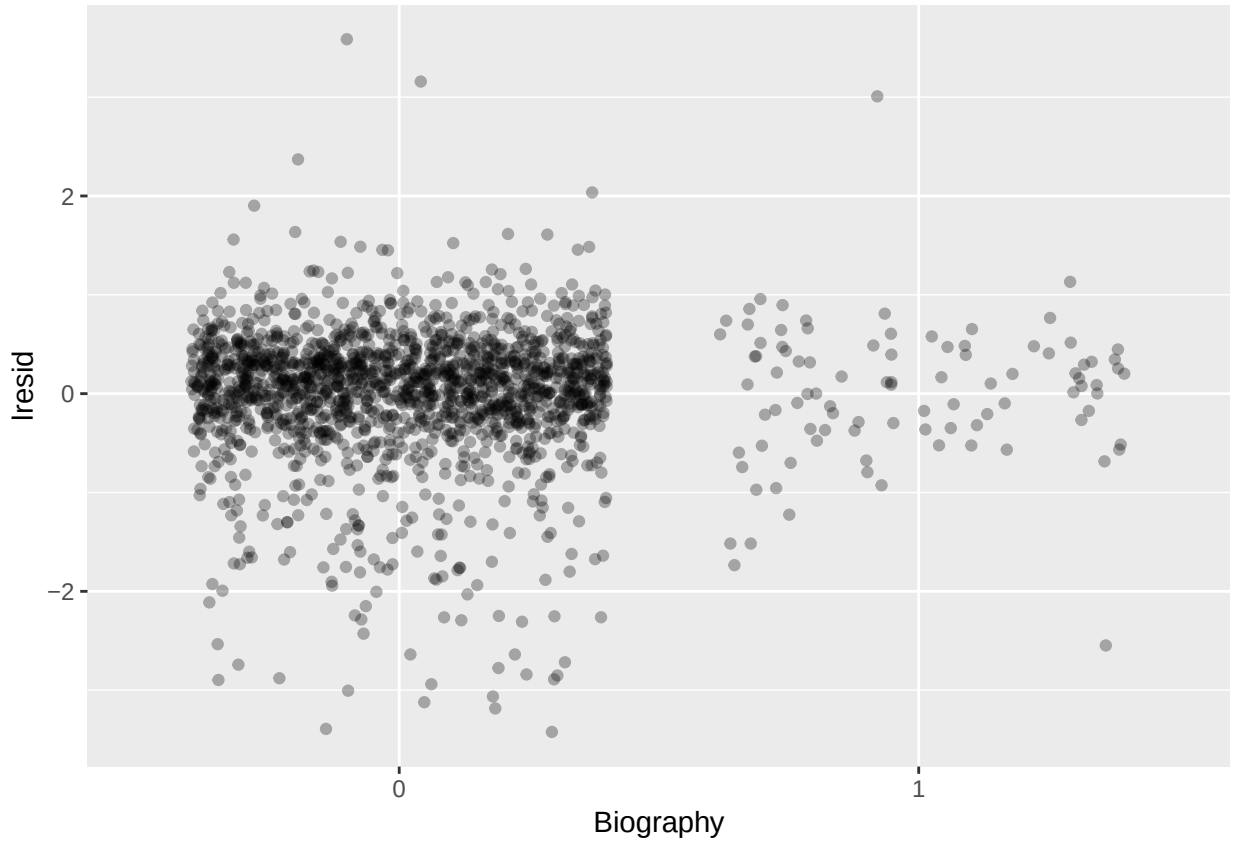

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



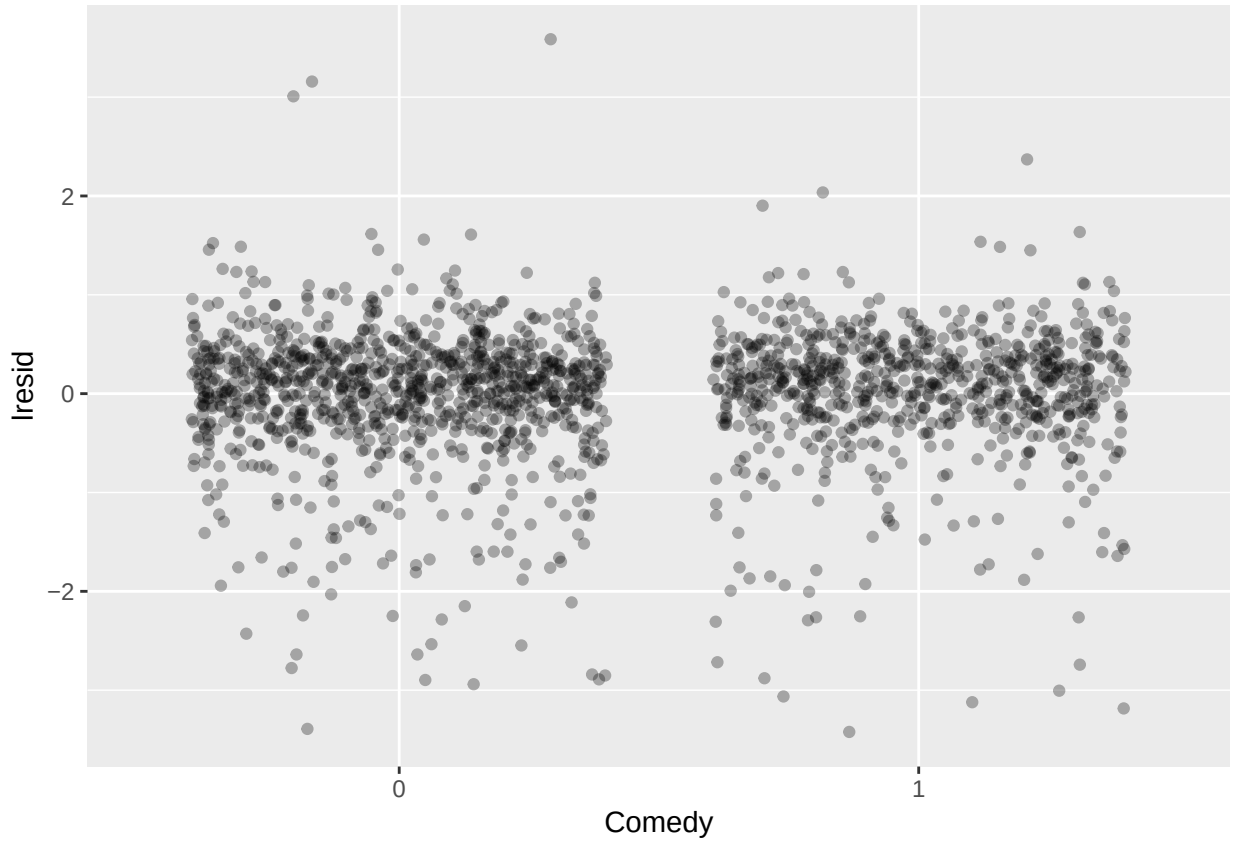

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



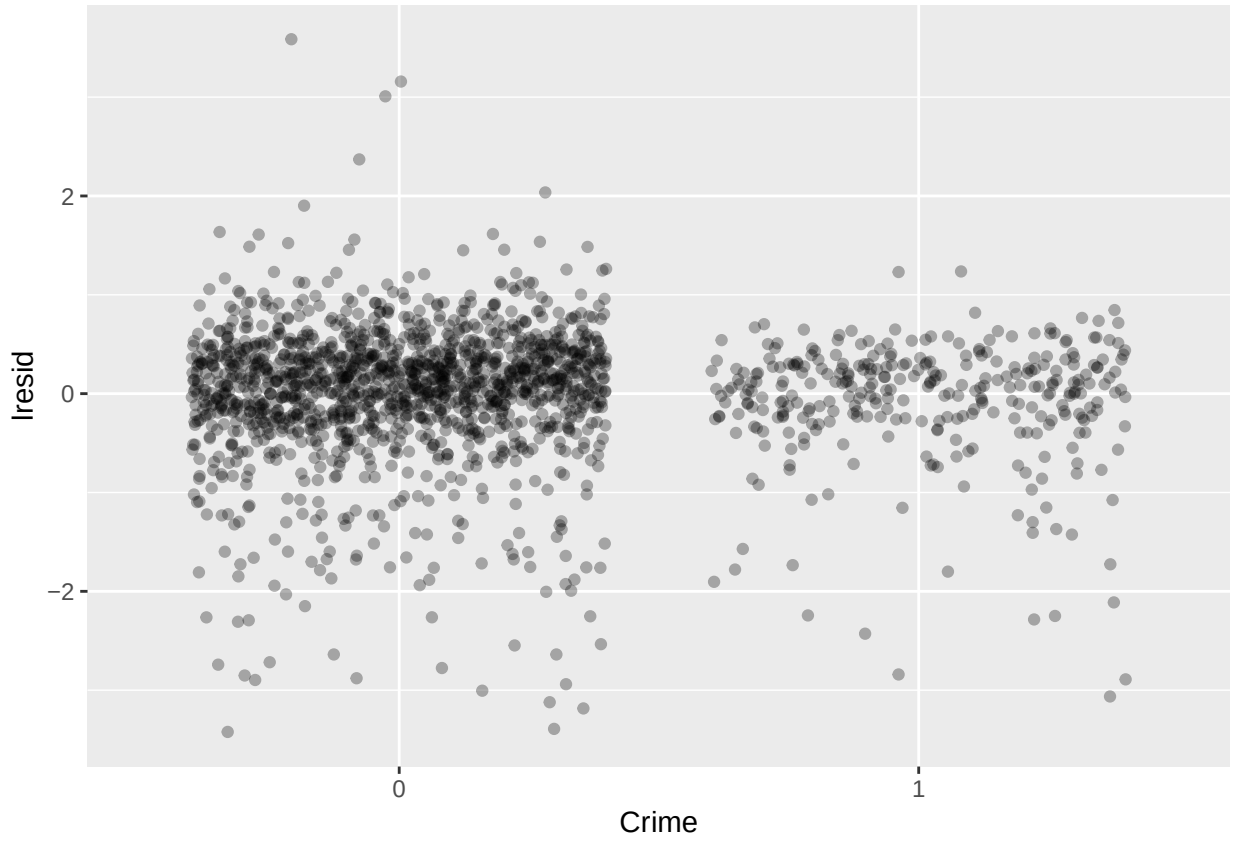

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



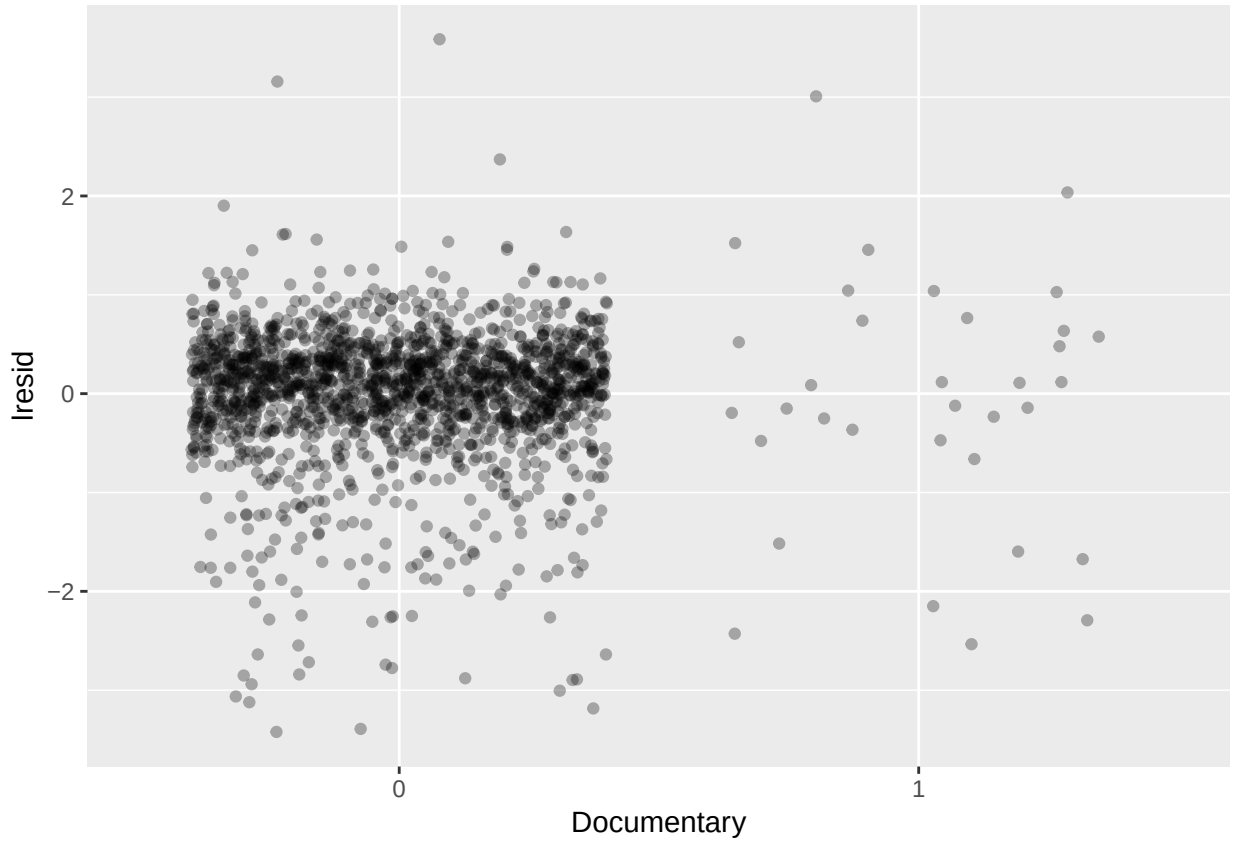

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



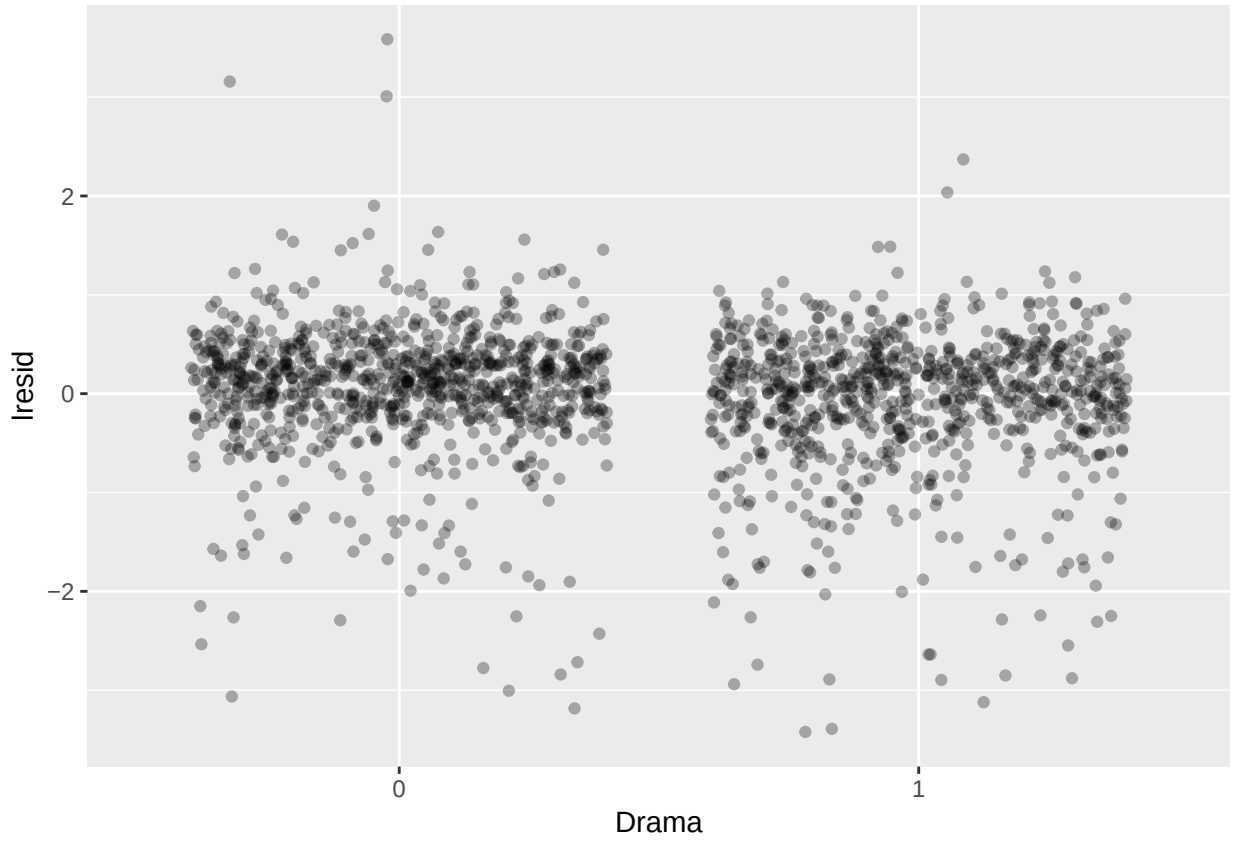

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



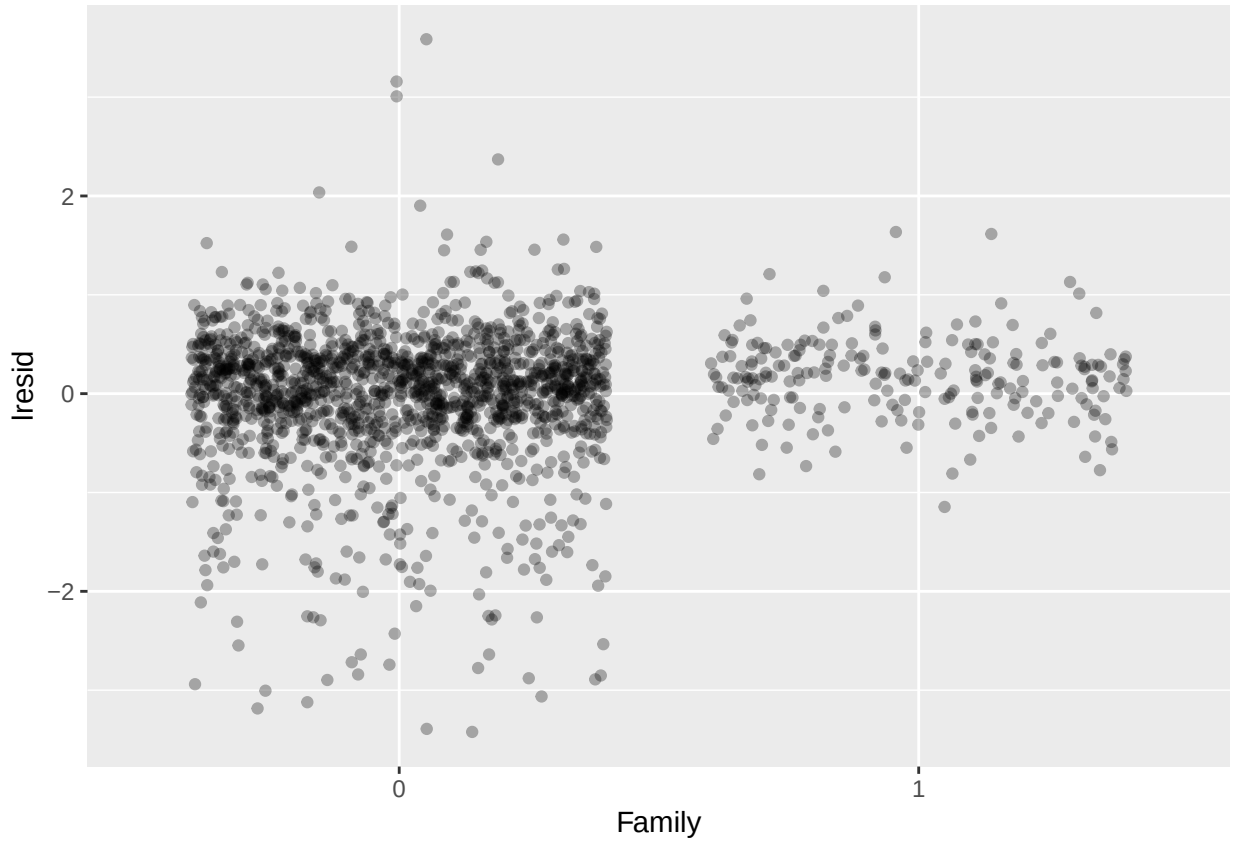

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



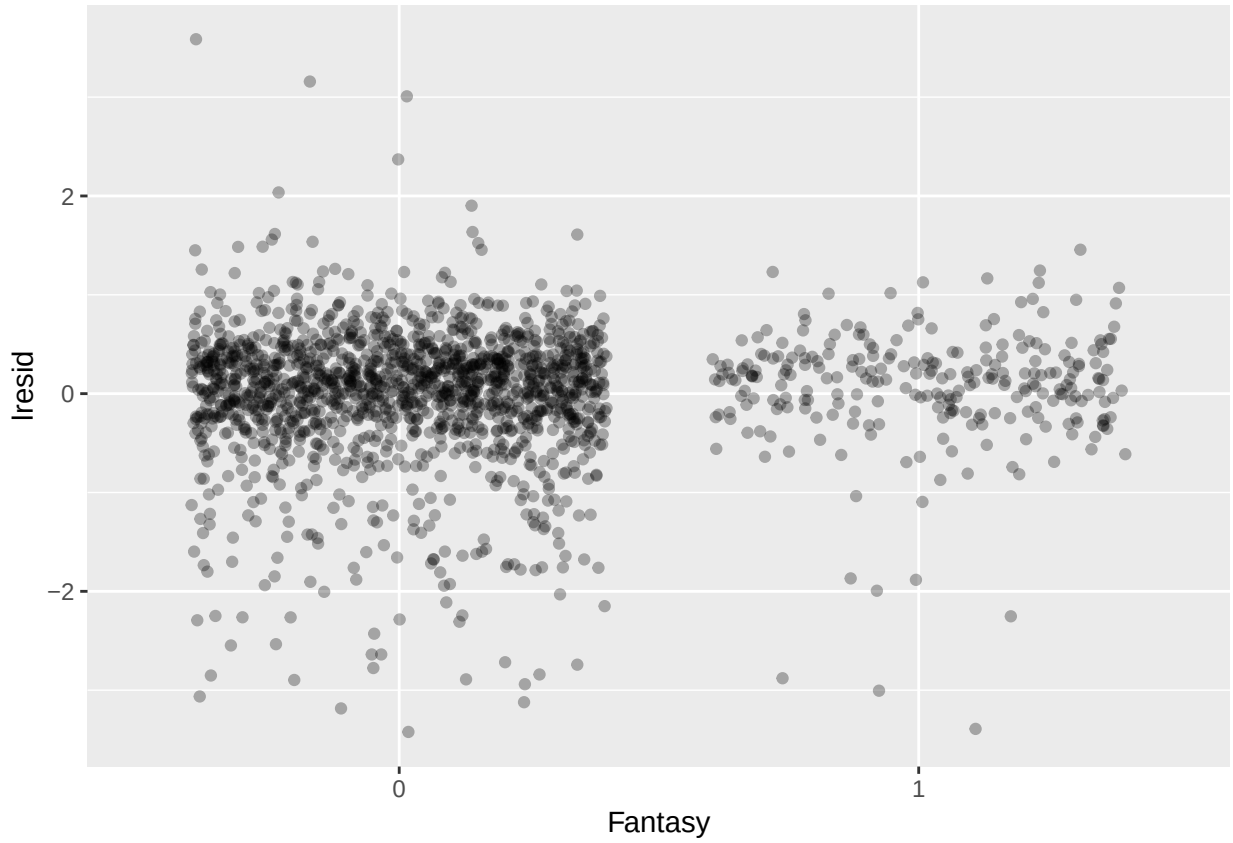

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



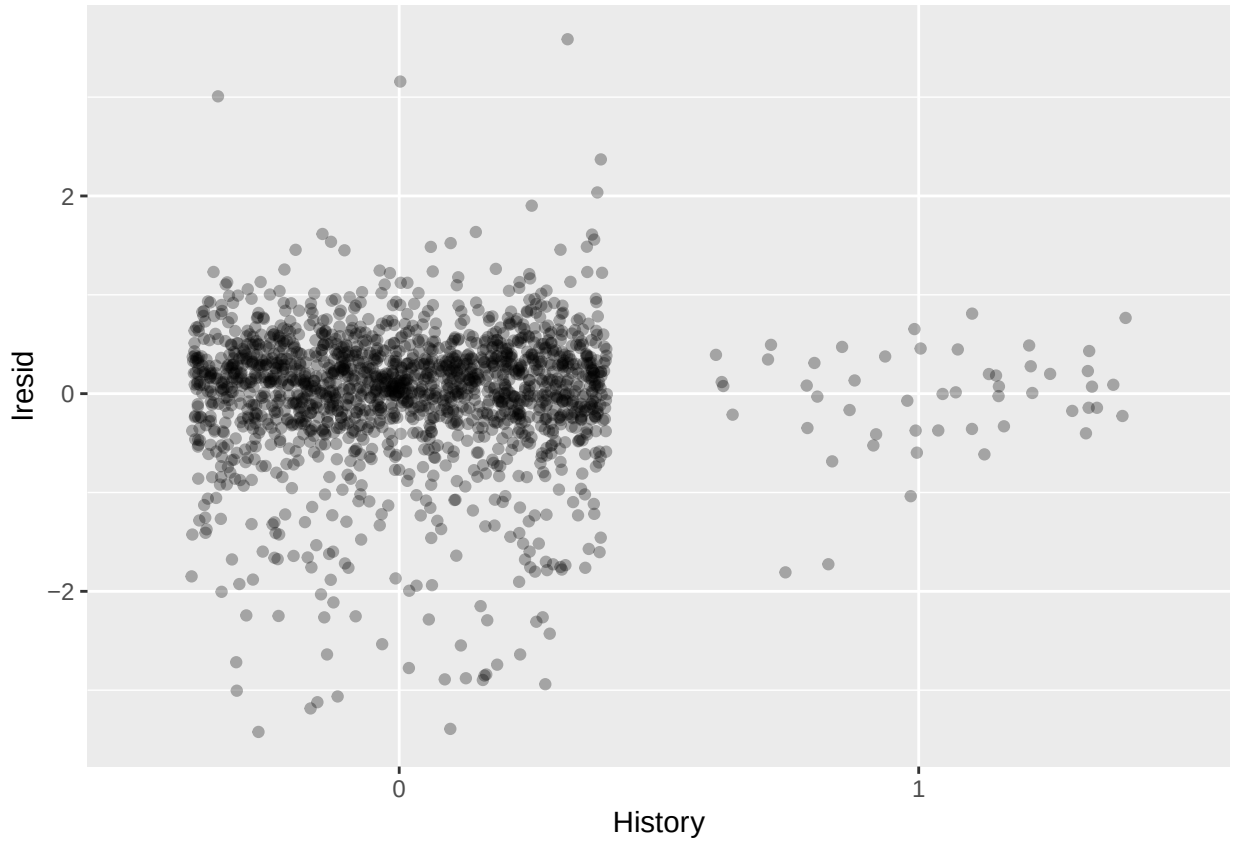

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



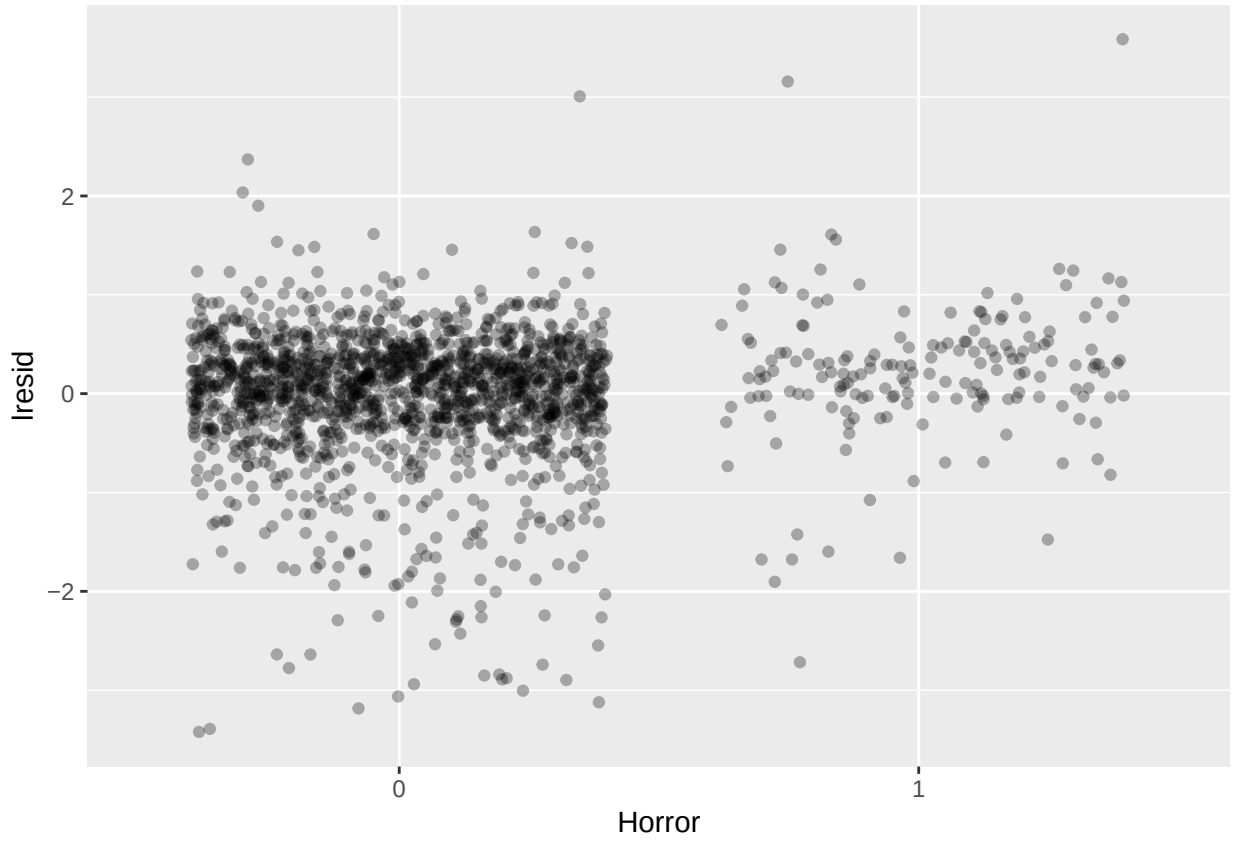

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



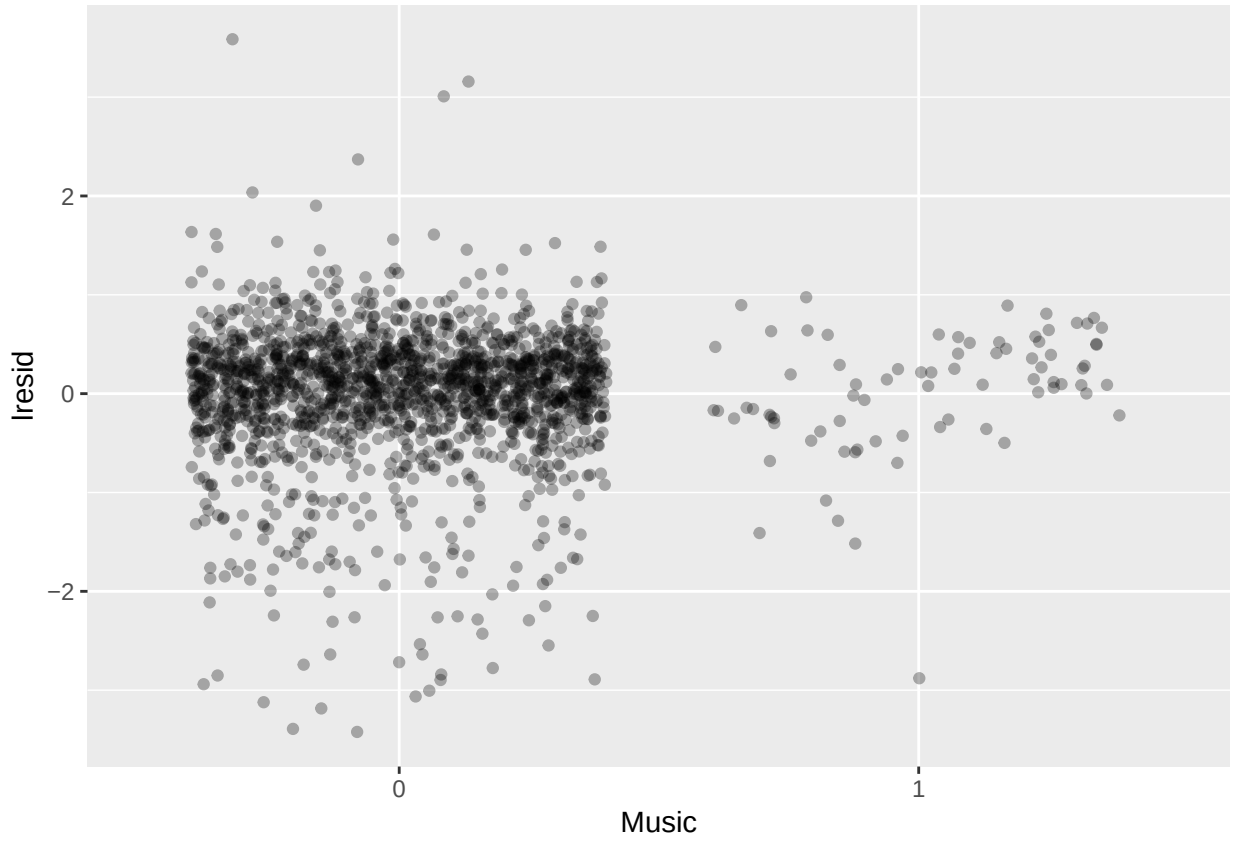

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



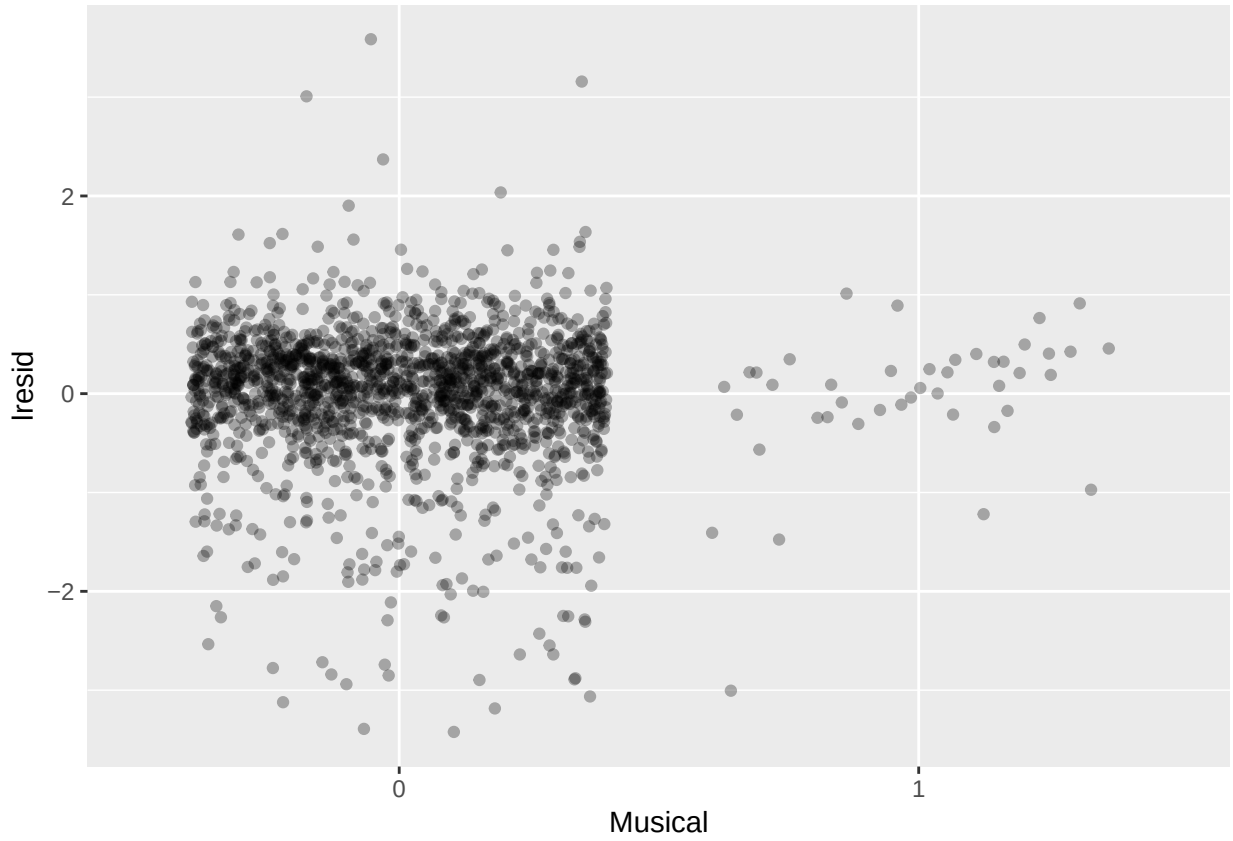

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



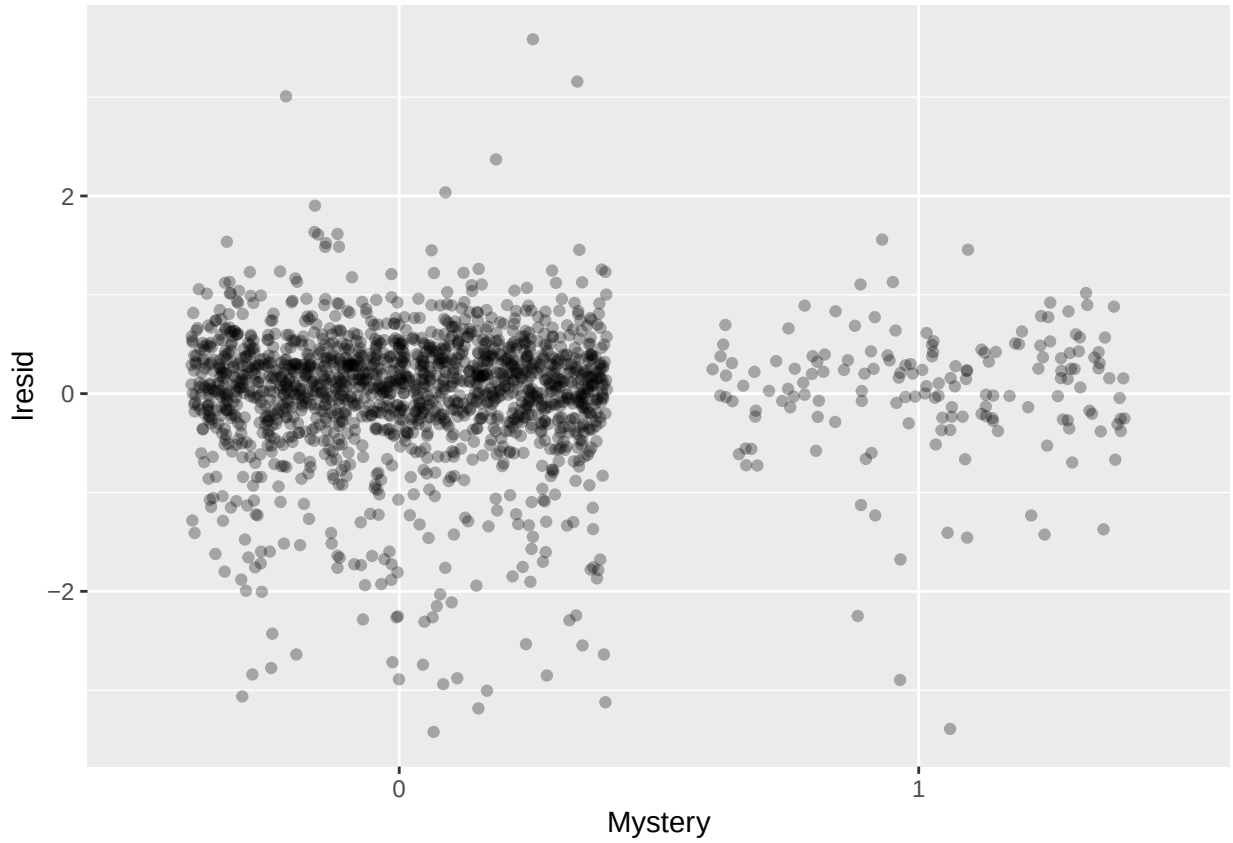

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



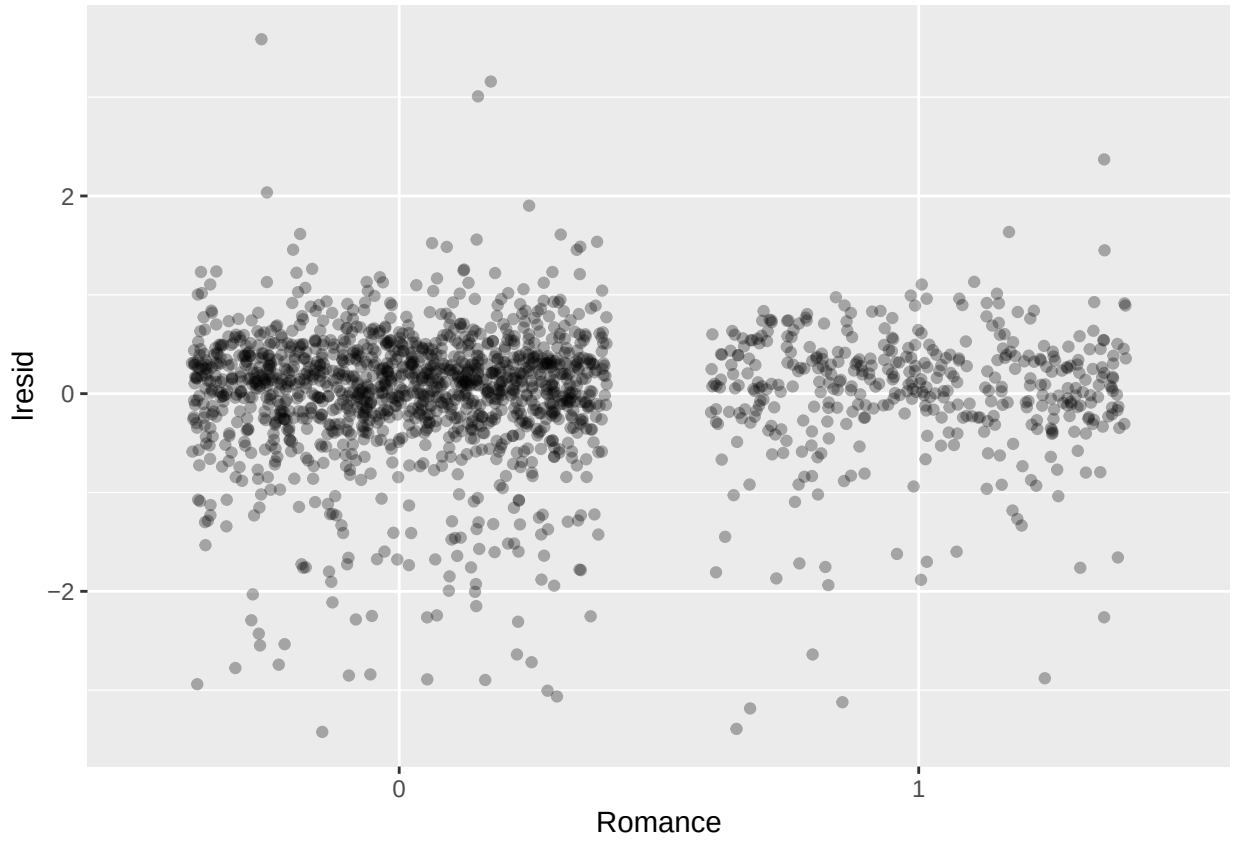

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



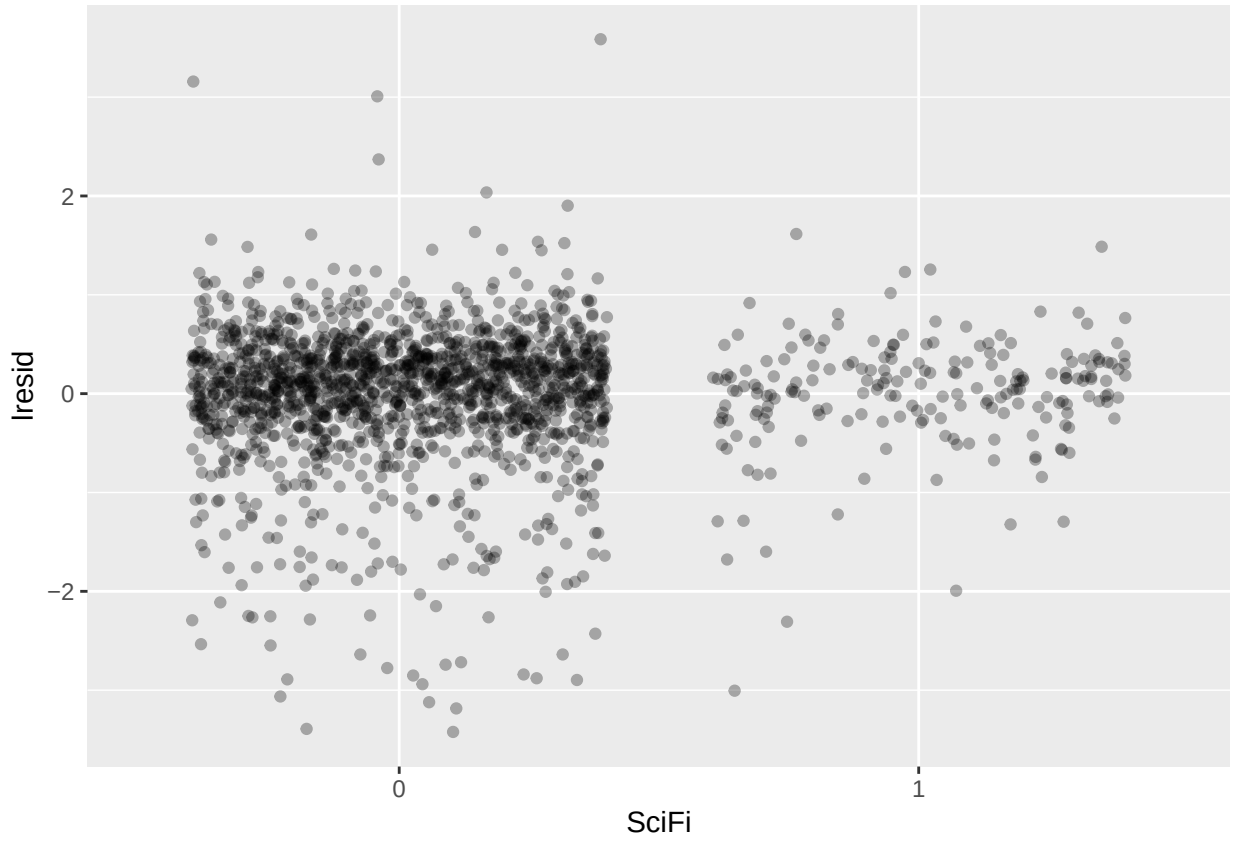

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



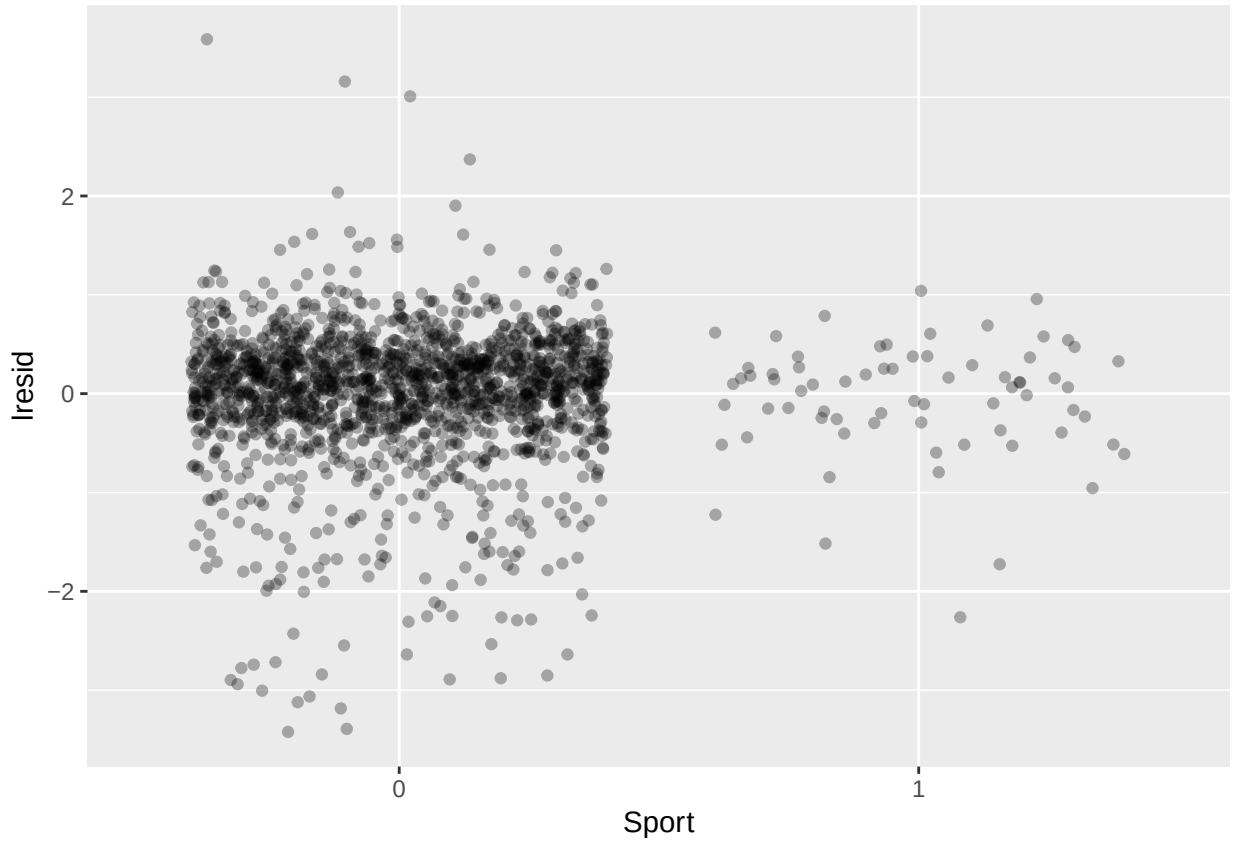

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



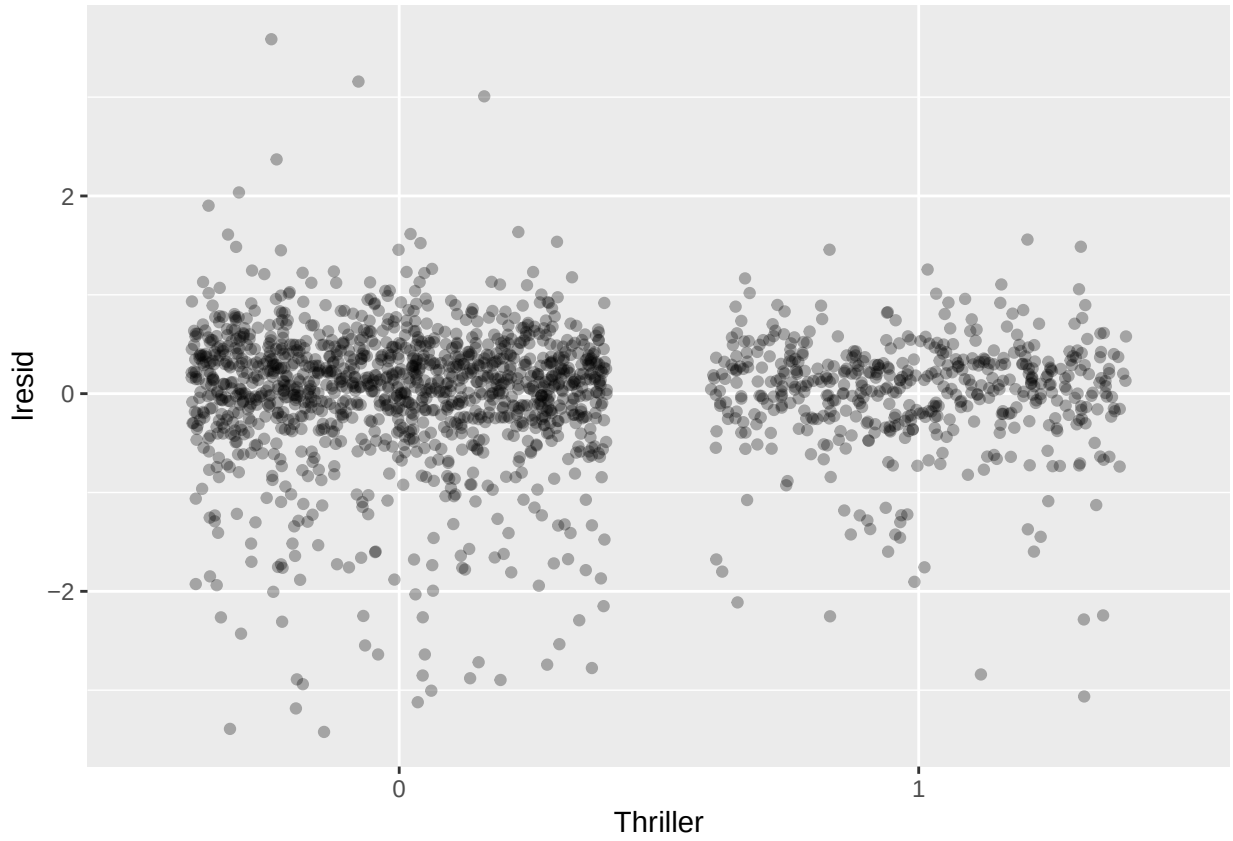

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



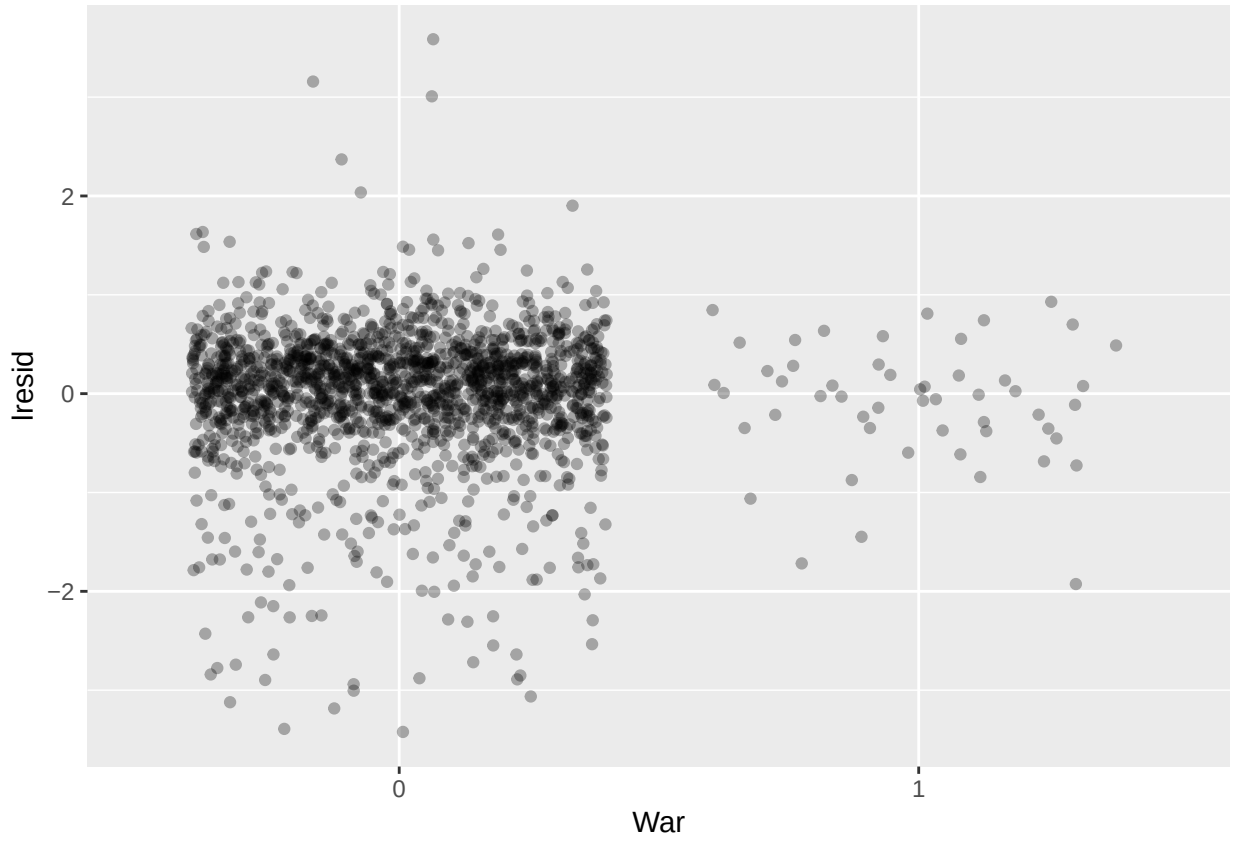

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



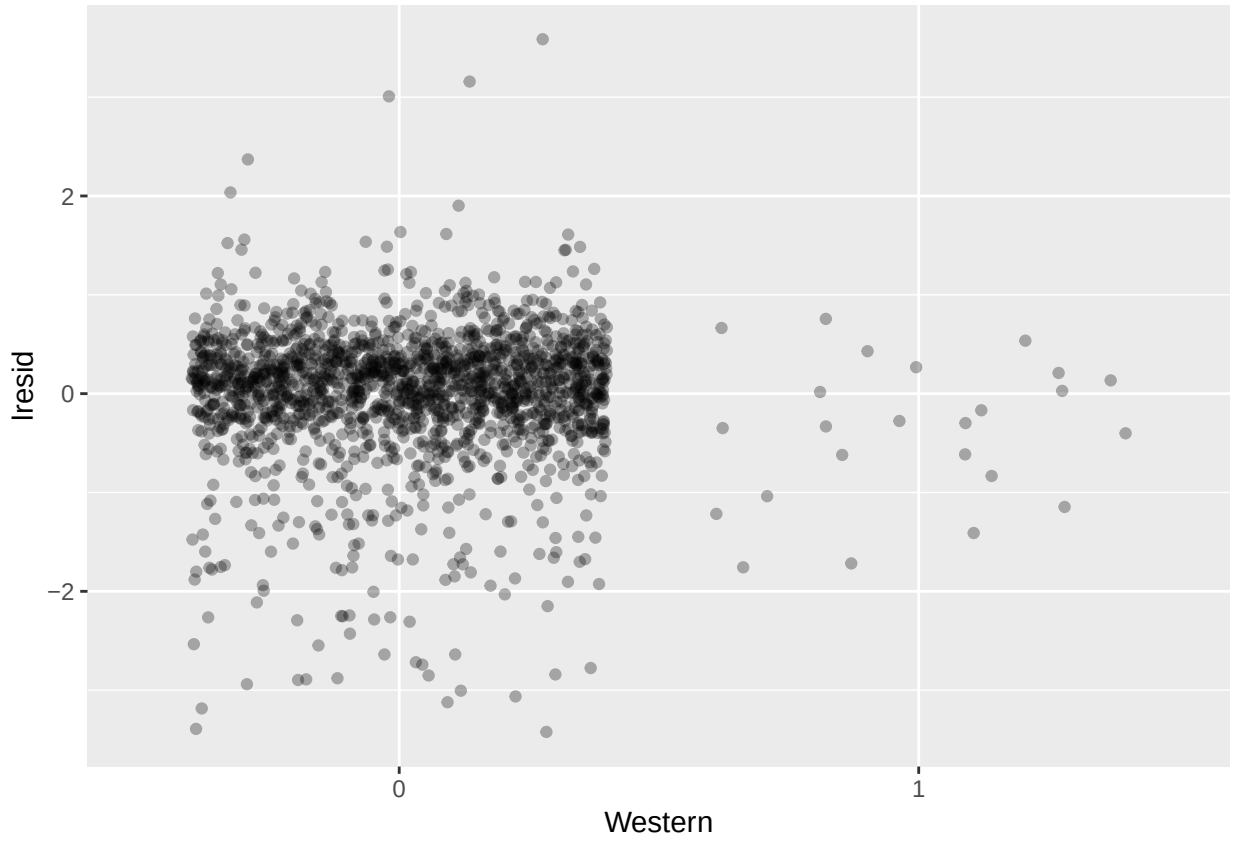

```
## Warning: Removed 102 rows containing missing values (geom_point).
```



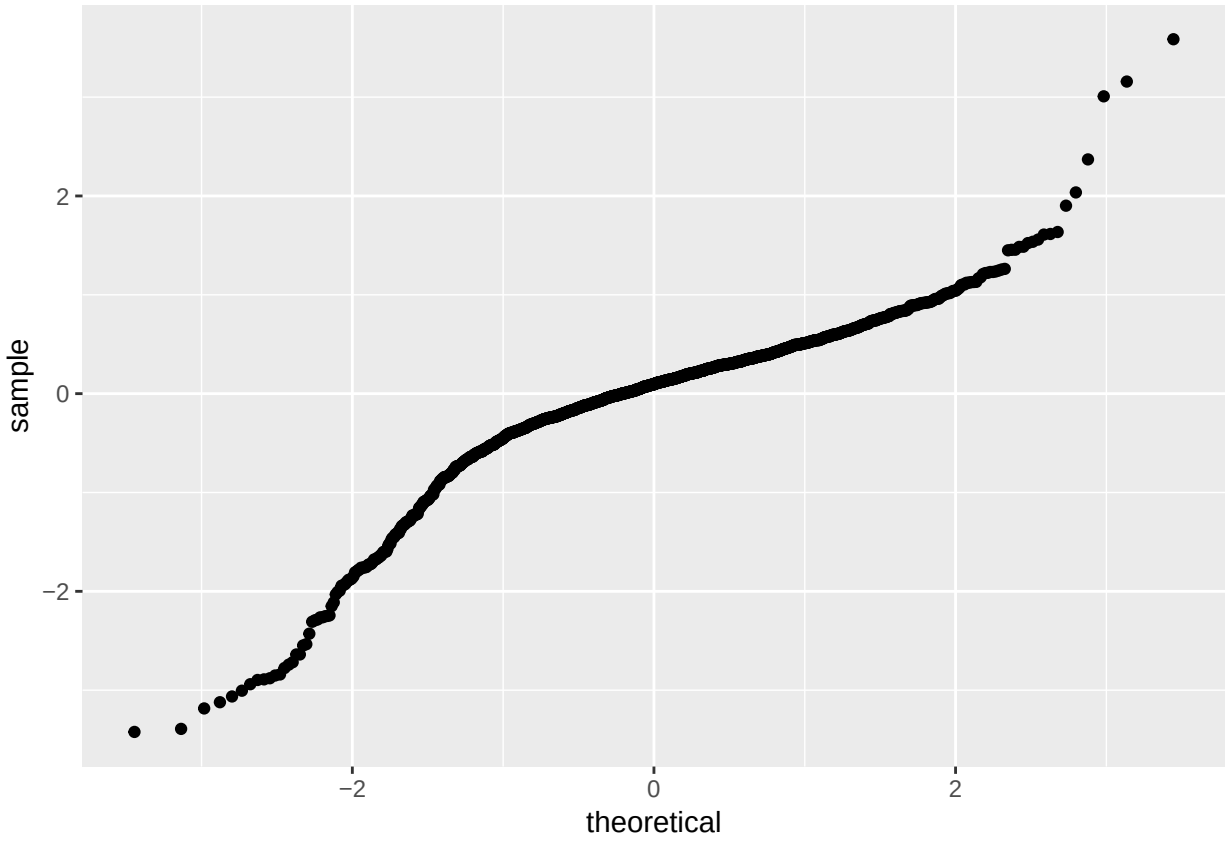

```
## Warning: Removed 102 rows containing missing values (geom_point).
```




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




```
## Warning: Removed 102 rows containing non-finite values (stat_qq).
```



Use variables with non-random relationships to the residuals.

```
mod_simple_plus <- lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating +
                        Family + Horror, data = train)

summary(mod_simple_plus)
```

```
##
## Call:
## lm(formula = real_gross_log ~ real_budget_log + imdb_score_log +
##     year + content_rating + Family + Horror, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4657 -0.2254  0.0845  0.3428  3.1910
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -0.01551    0.38954  -0.040    0.968
## real_budget_log      0.81650    0.02597  31.440  < 2e-16 ***
## imdb_score_log       1.90618    0.19916   9.571  < 2e-16 ***
## year1981            -0.16844    0.37031  -0.455    0.649
## year1982             0.21167    0.34124   0.620    0.535
## year1983             0.21119    0.38653   0.546    0.585
## year1984             0.51790    0.33037   1.568    0.117
## year1985             0.26373    0.34127   0.773    0.440
## year1986             0.06613    0.32617   0.203    0.839
```

```
## year1987              0.12252    0.31799   0.385    0.700
## year1988              0.09277    0.31192   0.297    0.766
## year1989              0.29569    0.30645   0.965    0.335
## year1990              0.07858    0.30783   0.255    0.799
## year1991              0.01181    0.31163   0.038    0.970
## year1992             -0.03313    0.31691  -0.105    0.917
## year1993             -0.05160    0.30813  -0.167    0.867
## year1994             -0.24165    0.30249  -0.799    0.424
## year1995             -0.08662    0.29423  -0.294    0.769
## year1996             -0.25017    0.28719  -0.871    0.384
## year1997             -0.15542    0.28722  -0.541    0.589
## year1998             -0.29490    0.28660  -1.029    0.304
## year1999             -0.26826    0.28327  -0.947    0.344
## year2000             -0.18925    0.28459  -0.665    0.506
## year2001             -0.28716    0.28262  -1.016    0.310
## year2002             -0.26763    0.28261  -0.947    0.344
## year2003             -0.22398    0.28405  -0.789    0.431
## year2004             -0.24424    0.28332  -0.862    0.389
## year2005             -0.27229    0.28249  -0.964    0.335
## year2006             -0.34943    0.28297  -1.235    0.217
## year2007             -0.34075    0.28438  -1.198    0.231
## year2008             -0.37503    0.28252  -1.327    0.185
## year2009             -0.37781    0.28163  -1.342    0.180
## year2010             -0.39954    0.28206  -1.417    0.157
## year2011             -0.28393    0.28449  -0.998    0.318
## year2012             -0.17336    0.28324  -0.612    0.541
## year2013             -0.10621    0.28265  -0.376    0.707
## year2014             -0.16106    0.28447  -0.566    0.571
## year2015             -0.29355    0.28662  -1.024    0.306
## year2016             -0.16115    0.30283  -0.532    0.595
## content_ratingNC-17 -0.18595    0.27781  -0.669    0.503
## content_ratingPG    -0.02590    0.11953  -0.217    0.829
## content_ratingPG-13  0.12934    0.13640   0.948    0.343
## content_ratingR     -0.07096    0.13545  -0.524    0.600
## Family1              0.33683    0.07868   4.281 1.97e-05 ***
## Horror1              0.34213    0.05192   6.590 5.89e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6106 on 1674 degrees of freedom
##   (132 observations deleted due to missingness)
## Multiple R-squared:  0.4874, Adjusted R-squared:  0.4739
## F-statistic: 36.17 on 44 and 1674 DF,  p-value: < 2.2e-16
```

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

```
## [1] 0.6293731
```

```
anova(mod_simple_plus)
```

```
## Analysis of Variance Table
##
## Response: real_gross_log
##                   Df Sum Sq Mean Sq   F value    Pr(>F)
## real_budget_log    1 491.67  491.67 1318.5462 < 2.2e-16 ***
```

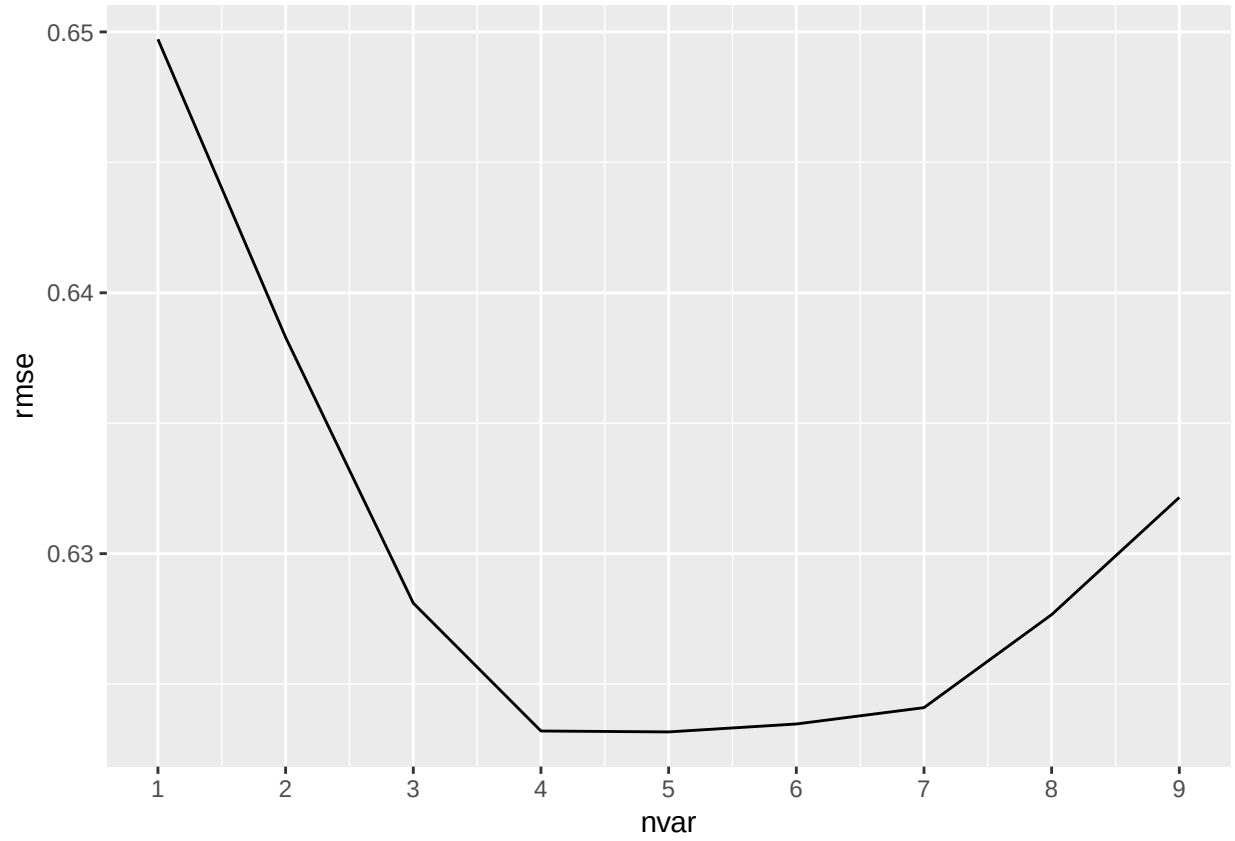
```
## imdb_score_log     1  28.18   28.18   75.5846 < 2.2e-16 ***
## year              36  36.51    1.01    2.7201 2.232e-07 ***
## content_rating     4  14.13    3.53    9.4767 1.417e-07 ***
## Family             1   6.74    6.74   18.0651 2.252e-05 ***
## Horror             1  16.19   16.19   43.4245 5.886e-11 ***
## Residuals       1674 624.21    0.37
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Stepwise just with those variables with non-random relationships with residual

```
# stepwise
# ALL potentially relevant variables
rmse_lst <- step_wise_loop(df = train %>% select(real_budget_log, imdb_score_log, year,
                                                 content_rating, Family, Western, Fantasy, Horror,
                                                 Documentary))
```

```
## real_budget_log
##       0.6497192
## [1] 1
## imdb_score_log
##      0.6382915
## [1] 2
##      year
## 0.6281035
## [1] 3
## content_rating
##       0.623201
## [1] 4
## Documentary
##   0.6231664
## [1] 5
##   Fantasy
## 0.6234698
## [1] 6
##    Family
## 0.6240969
## [1] 7
##   Western
## 0.6276628
## [1] 8
##    Horror
## 0.6321549
```

```
# 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

mod_simple_plus2 <- lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating,
               data = train)

summary(mod_simple_plus2)
```

```
##
## Call:
## lm(formula = real_gross_log ~ real_budget_log + imdb_score_log +
##     year + content_rating, data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4983 -0.2230  0.0973  0.3455  3.4758
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         0.444778   0.391868   1.135  0.25653
## real_budget_log     0.821725   0.026058  31.534  < 2e-16 ***
## imdb_score_log      1.626517   0.199024   8.172 5.89e-16 ***
## year1981           -0.238907   0.376557  -0.634  0.52587
## year1982            0.297881   0.346750   0.859  0.39043
## year1983            0.235272   0.393319   0.598  0.54981
## year1984            0.504909   0.336175   1.502  0.13331
## year1985            0.335770   0.346892   0.968  0.33322
```

```
## year1986             0.111891   0.331663   0.337  0.73589
## year1987             0.212045   0.322869   0.657  0.51143
## year1988             0.226528   0.316488   0.716  0.47424
## year1989             0.285472   0.311721   0.916  0.35991
## year1990             0.178997   0.312725   0.572  0.56714
## year1991             0.014859   0.316338   0.047  0.96254
## year1992             0.045736   0.321585   0.142  0.88692
## year1993            -0.003696   0.312675  -0.012  0.99057
## year1994            -0.184559   0.306846  -0.601  0.54761
## year1995            -0.042381   0.298579  -0.142  0.88714
## year1996            -0.208917   0.291490  -0.717  0.47365
## year1997            -0.131576   0.291460  -0.451  0.65173
## year1998            -0.246108   0.290619  -0.847  0.39721
## year1999            -0.232583   0.287331  -0.809  0.41837
## year2000            -0.141456   0.288697  -0.490  0.62421
## year2001            -0.245089   0.286678  -0.855  0.39271
## year2002            -0.222859   0.286430  -0.778  0.43665
## year2003            -0.176337   0.287847  -0.613  0.54022
## year2004            -0.193313   0.287370  -0.673  0.50123
## year2005            -0.210828   0.286412  -0.736  0.46177
## year2006            -0.308781   0.286785  -1.077  0.28177
## year2007            -0.267002   0.288303  -0.926  0.35452
## year2008            -0.323941   0.286439  -1.131  0.25825
## year2009            -0.325844   0.285759  -1.140  0.25434
## year2010            -0.338852   0.286066  -1.185  0.23637
## year2011            -0.217375   0.288402  -0.754  0.45112
## year2012            -0.110811   0.287004  -0.386  0.69948
## year2013            -0.037216   0.286569  -0.130  0.89669
## year2014            -0.103257   0.288515  -0.358  0.72047
## year2015            -0.216276   0.290823  -0.744  0.45718
## year2016            -0.069711   0.307146  -0.227  0.82048
## content_ratingNC-17 -0.197326   0.272730  -0.724  0.46946
## content_ratingPG    -0.123677   0.119665  -1.034  0.30150
## content_ratingPG-13 -0.175106   0.115920  -1.511  0.13109
## content_ratingR     -0.346131   0.115941  -2.985  0.00287 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6214 on 1676 degrees of freedom
##   (132 observations deleted due to missingness)
## Multiple R-squared:  0.4685, Adjusted R-squared:  0.4552
## F-statistic: 35.18 on 42 and 1676 DF,  p-value: < 2.2e-16
```

```
rmse(mod_simple_plus2, data = valid)
```

```
## [1] 0.623201
```

```
# when consider factors as one variable, they are significant
anova(mod_simple_plus2)
```

```
## Analysis of Variance Table
##
## Response: real_gross_log
##                   Df Sum Sq Mean Sq   F value    Pr(>F)
## real_budget_log    1 491.67  491.67 1273.3487 < 2.2e-16 ***
```

```
## imdb_score_log     1  28.18   28.18   72.9937 < 2.2e-16 ***
## year              36  36.51    1.01    2.6269 6.334e-07 ***
## content_rating     4  14.13    3.53    9.1519 2.589e-07 ***
## Residuals       1676 647.14    0.39
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# number of observations
nobs(mod_simple_plus2)
```

```
## [1] 1719
```

```
# compare with anova
anova(mod_simple_plus, mod_simple_plus2)
```

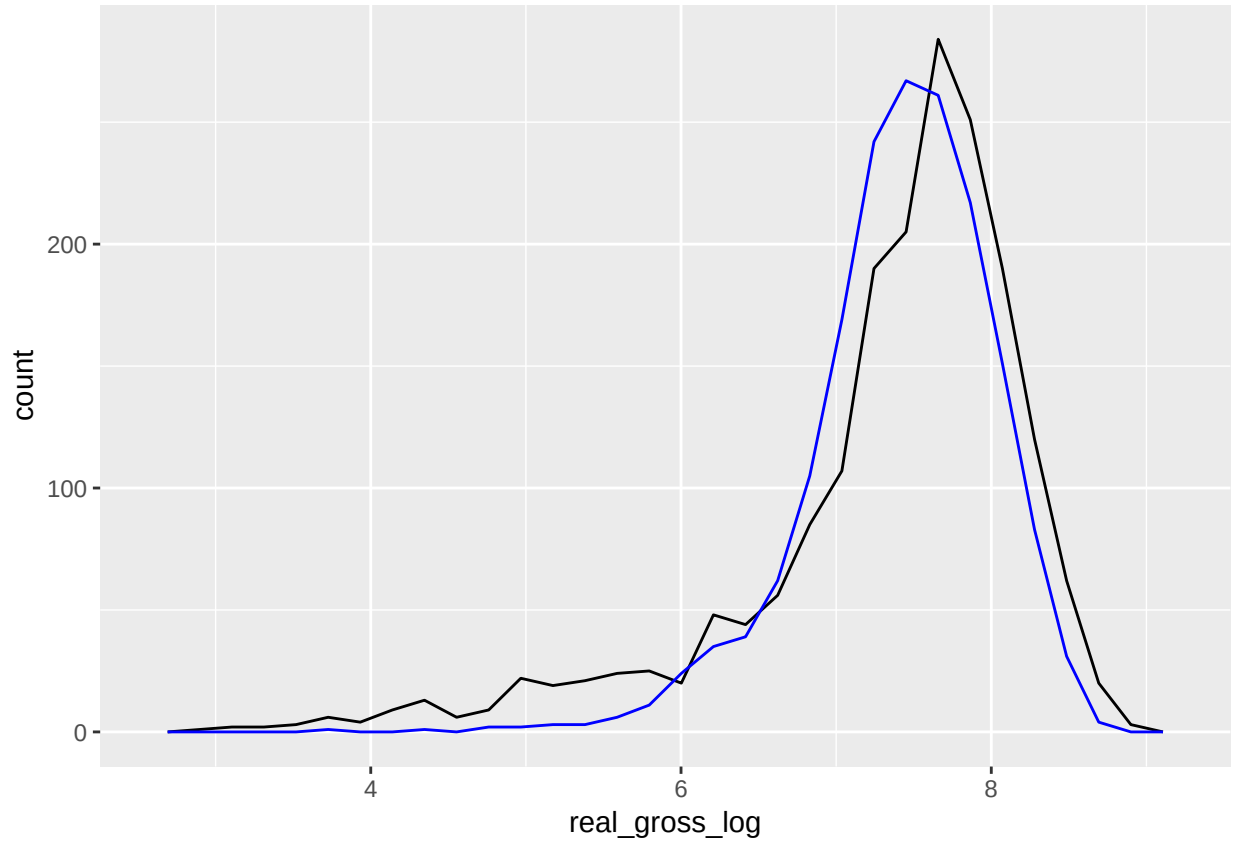
```
## Analysis of Variance Table
##
## Model 1: real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating +
##     Family + Horror
## Model 2: real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1   1674 624.21
## 2   1676 647.14 -2   -22.929 30.745 7.71e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Predictions

```
train %>%
  add_predictions(mod_simple_plus, 'lpred') %>%
  ggplot() +
  geom_freqpoly(aes(x = real_gross_log)) +
  geom_freqpoly(aes(x = lpred), color = 'blue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

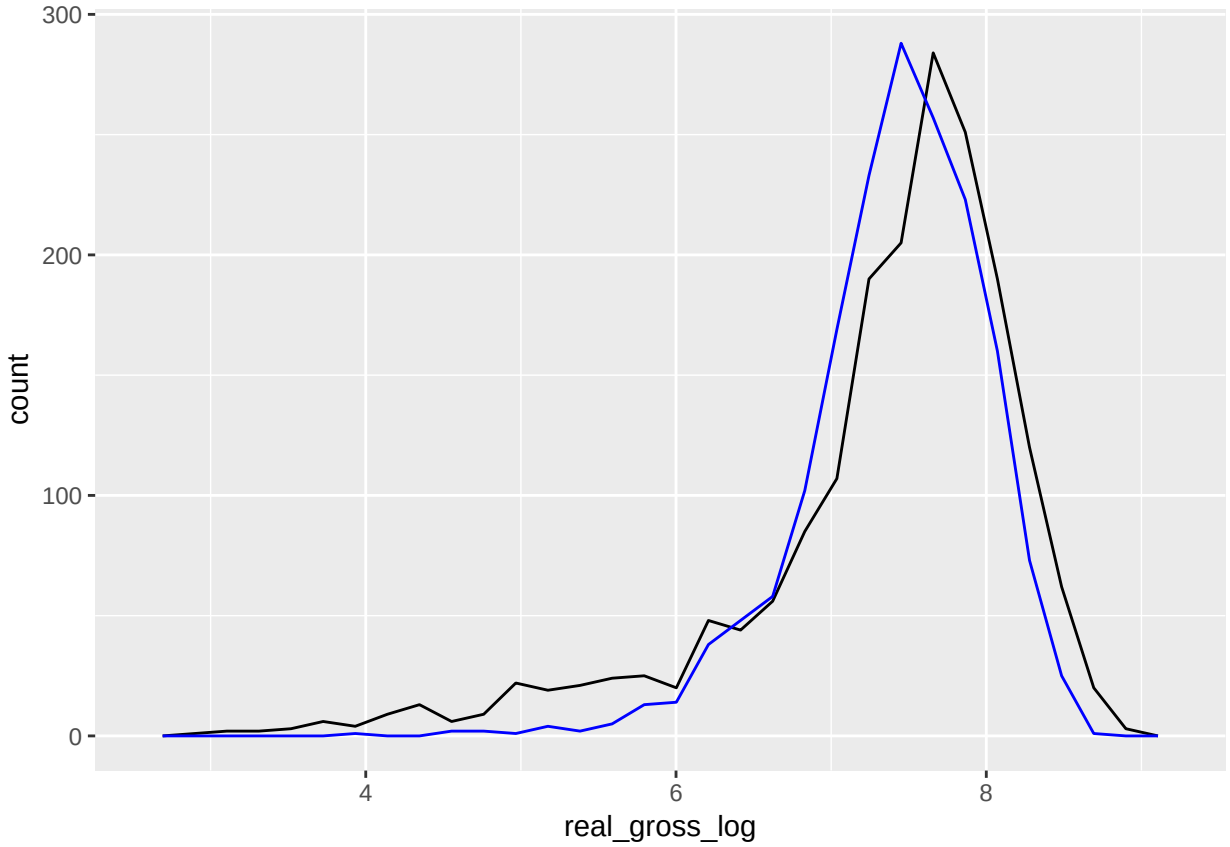
```
## Warning: Removed 132 rows containing non-finite values (stat_bin).
```




```
train %>%
  add_predictions(mod_simple_plus2, 'lpred') %>%
  ggplot() +
  geom_freqpoly(aes(x = real_gross_log)) +
  geom_freqpoly(aes(x = lpred), color = 'blue')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

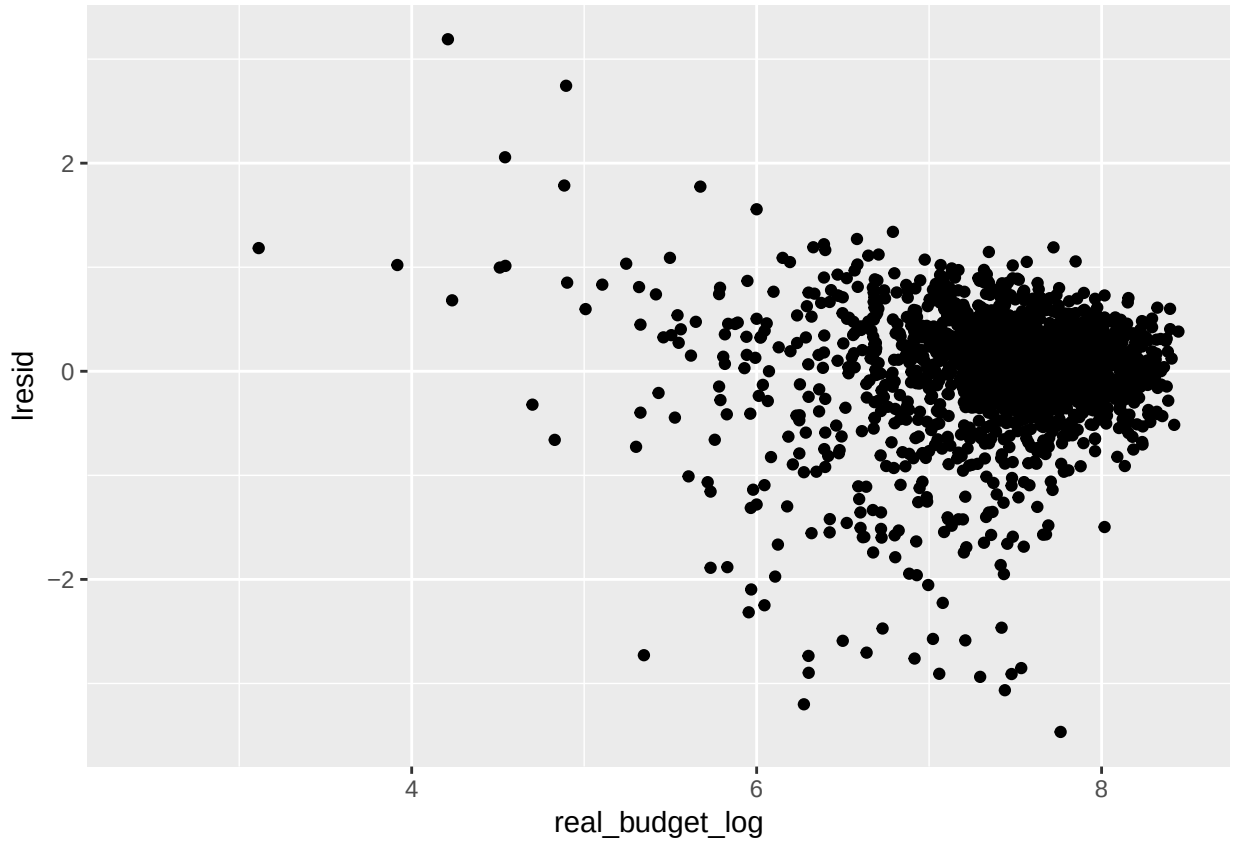
```
## Warning: Removed 132 rows containing non-finite values (stat_bin).
```



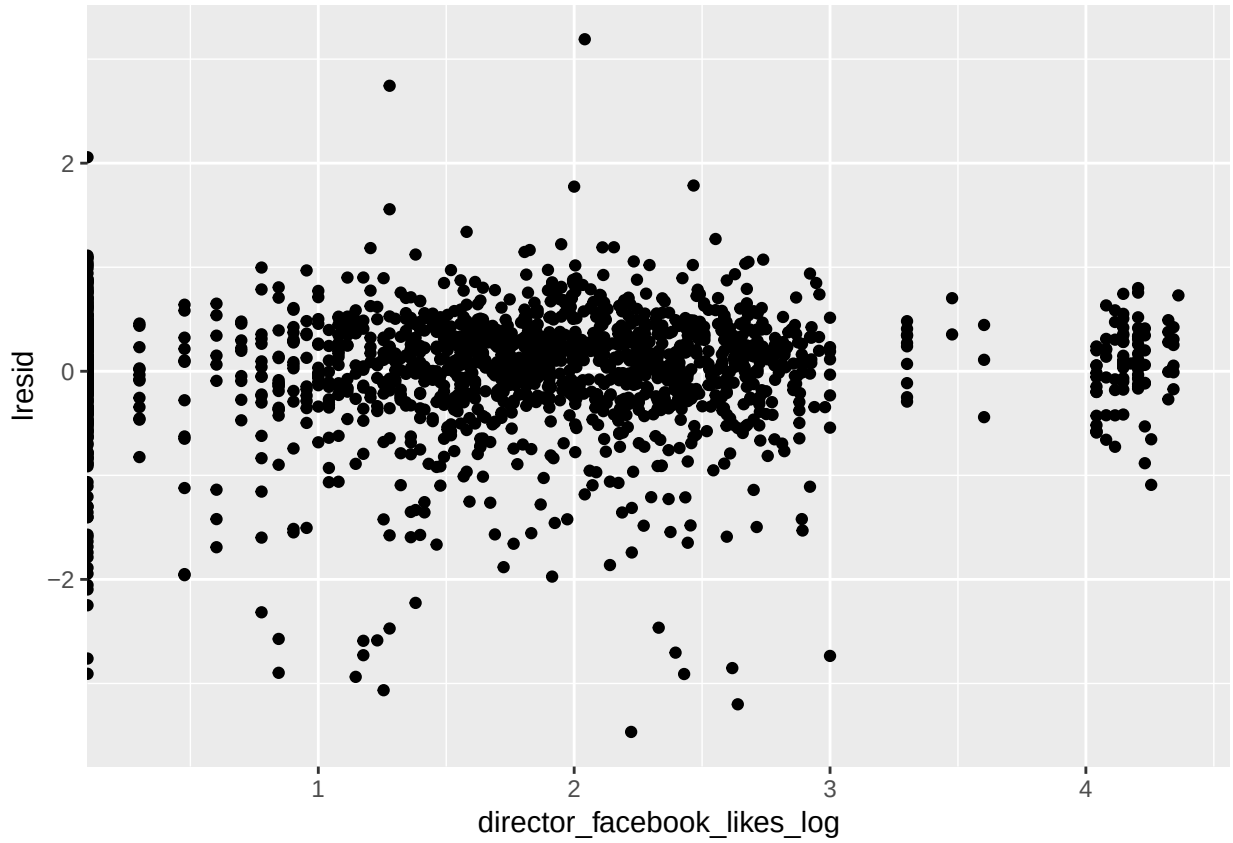
Residuals

```
gr_resid(mod_simple_plus)
```

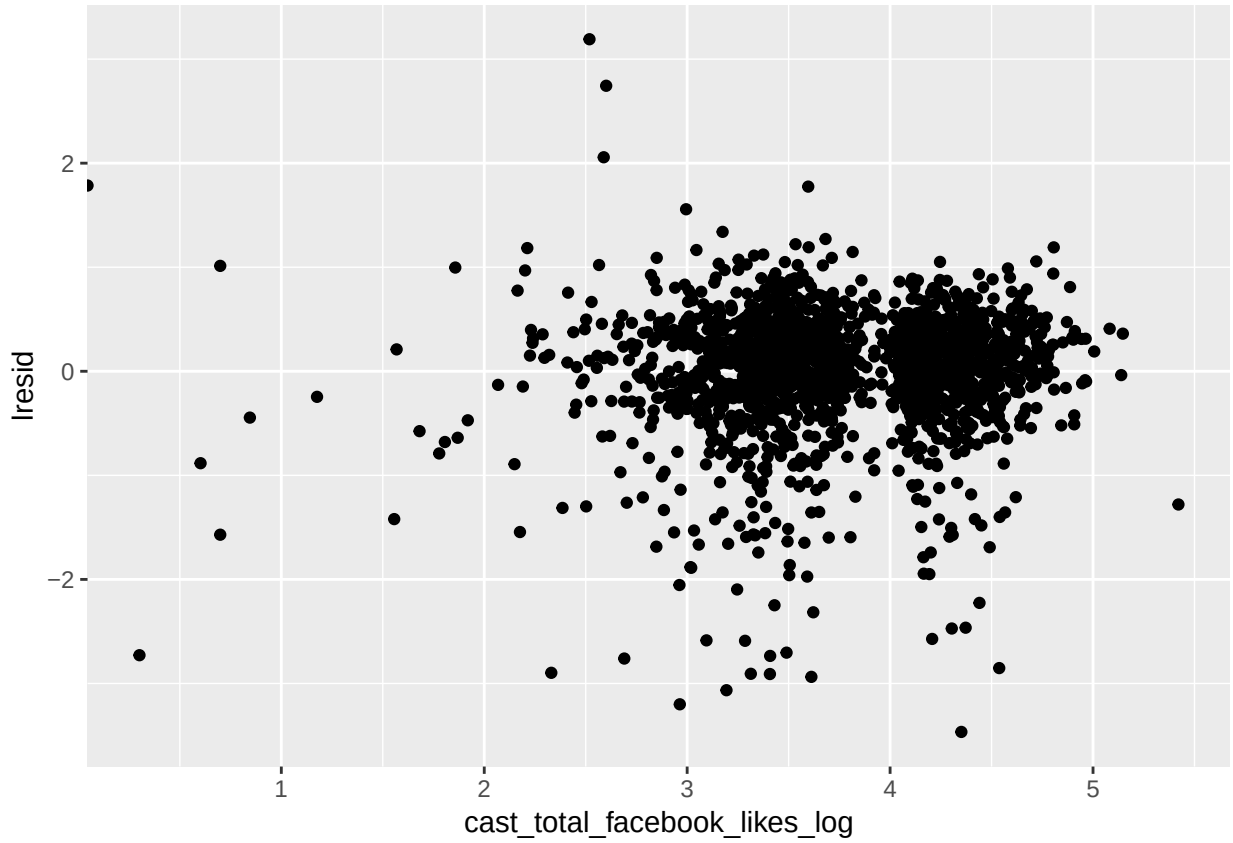
```
## Warning: Removed 132 rows containing missing values (geom_point).
```



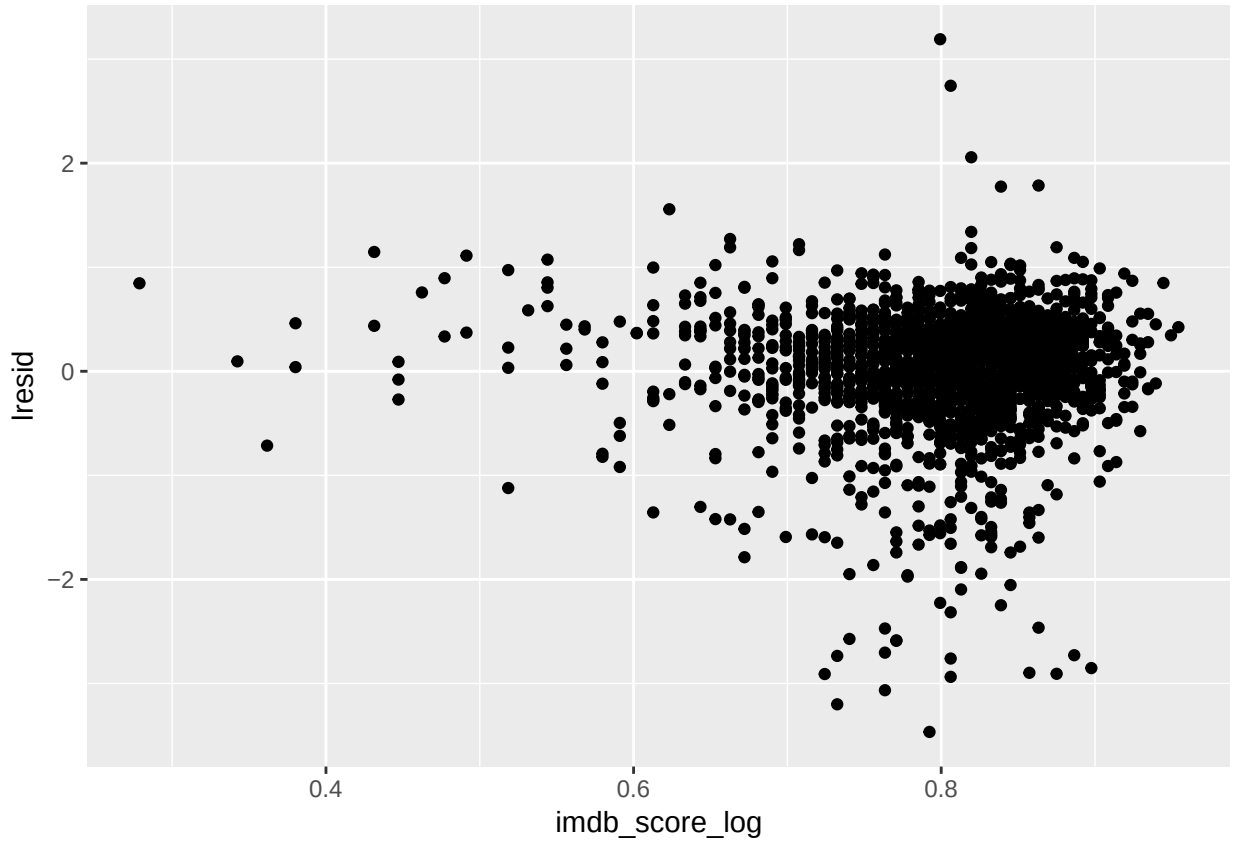

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



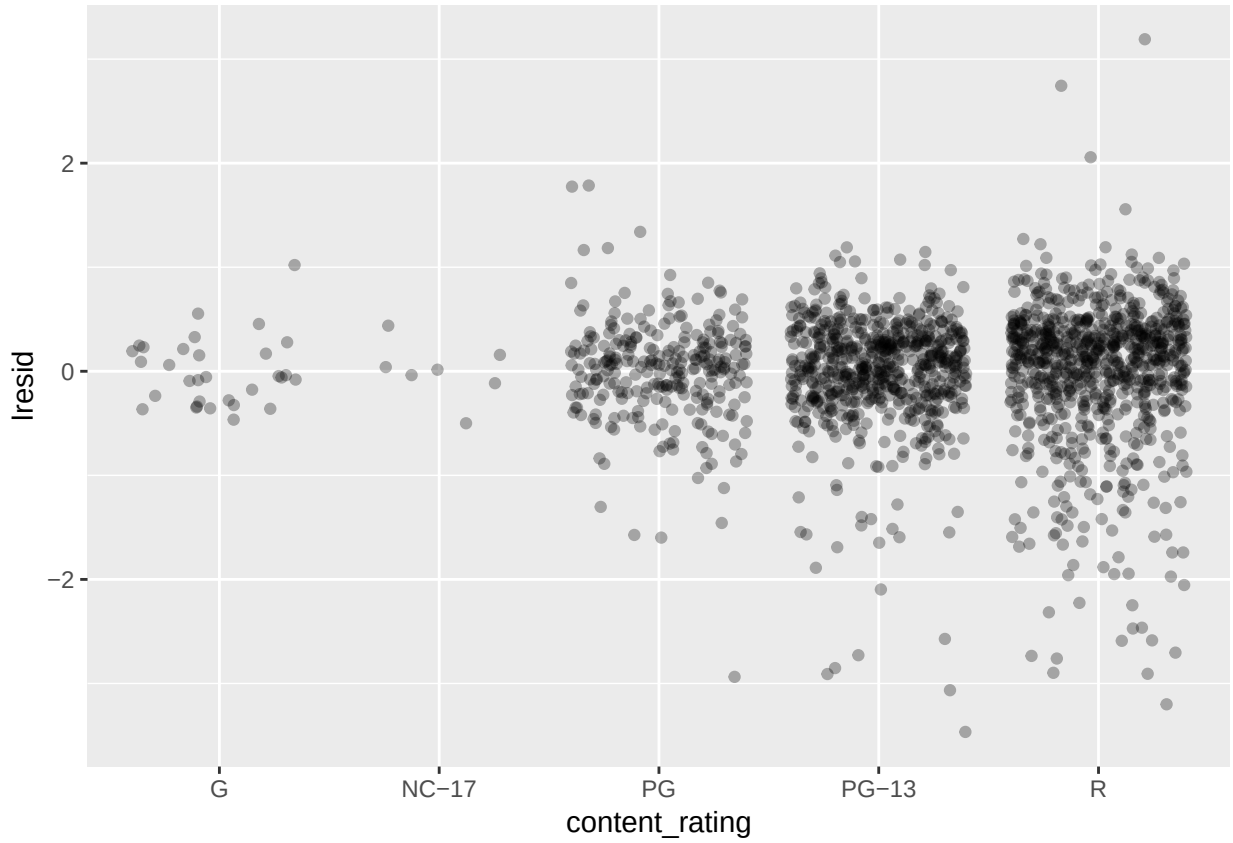

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



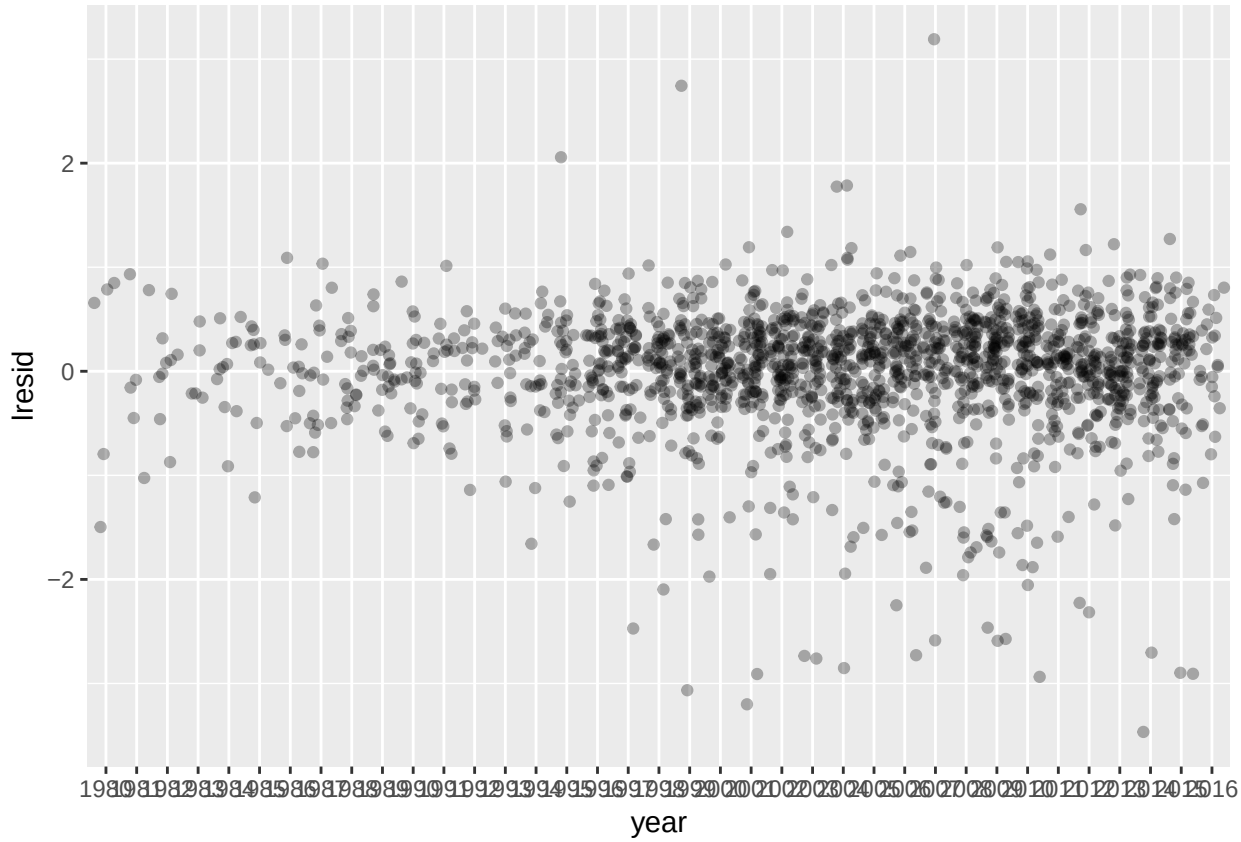

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



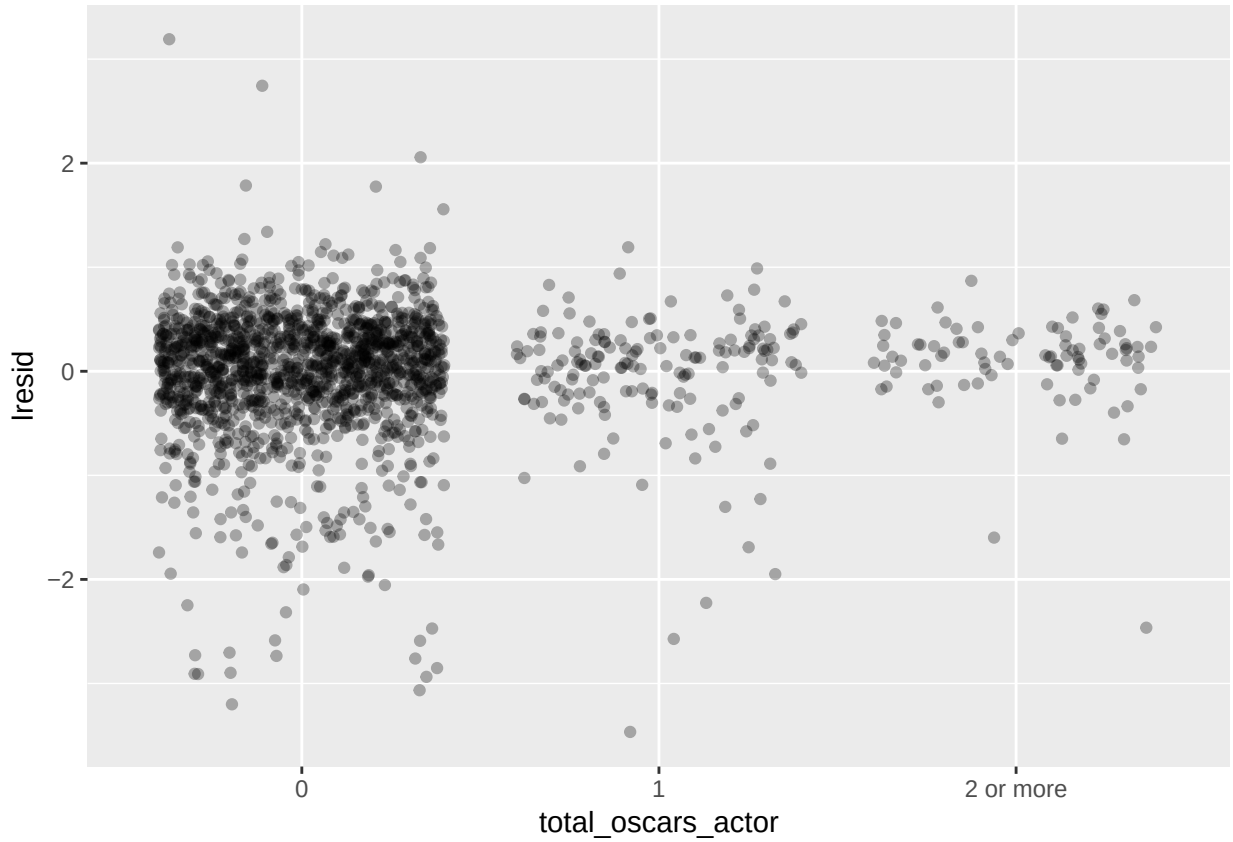

```
## Warning: Removed 94 rows containing missing values (geom_point).
```



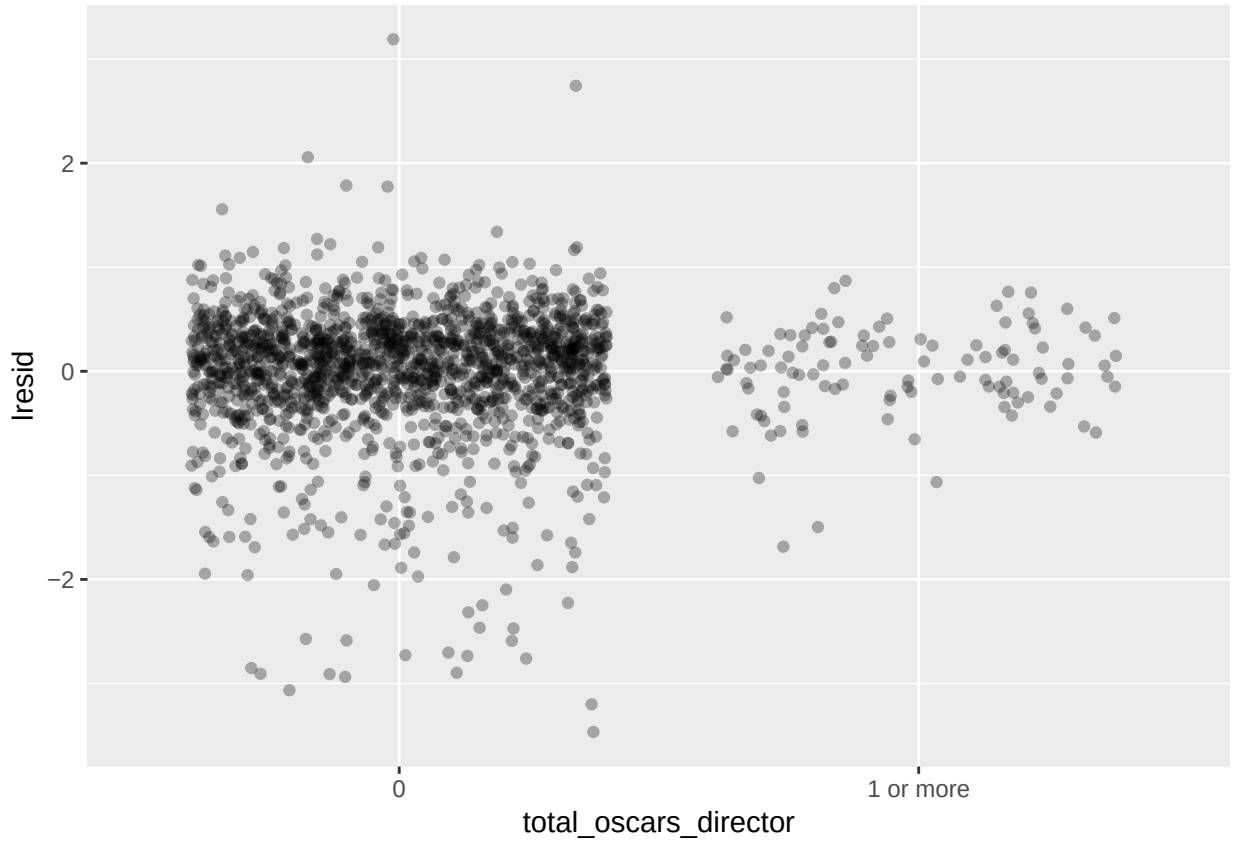

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



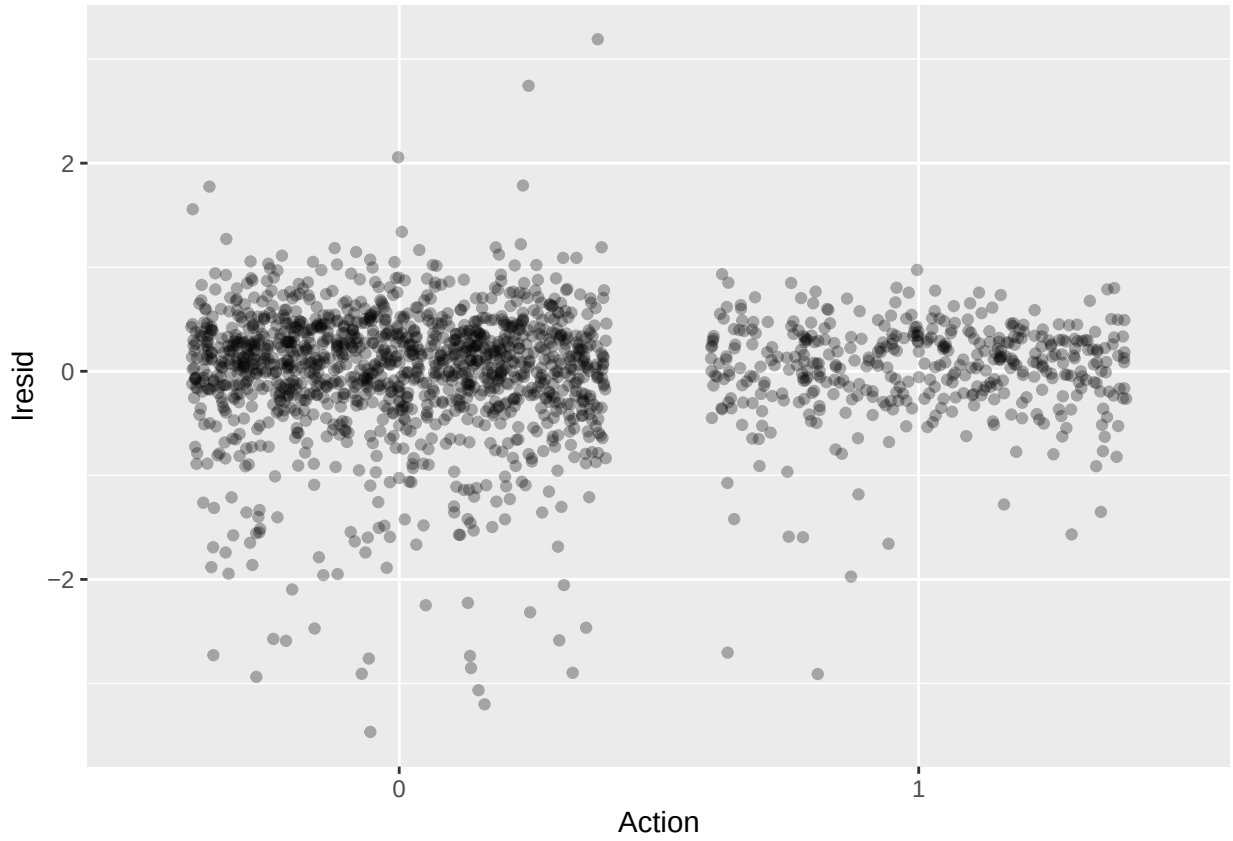

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



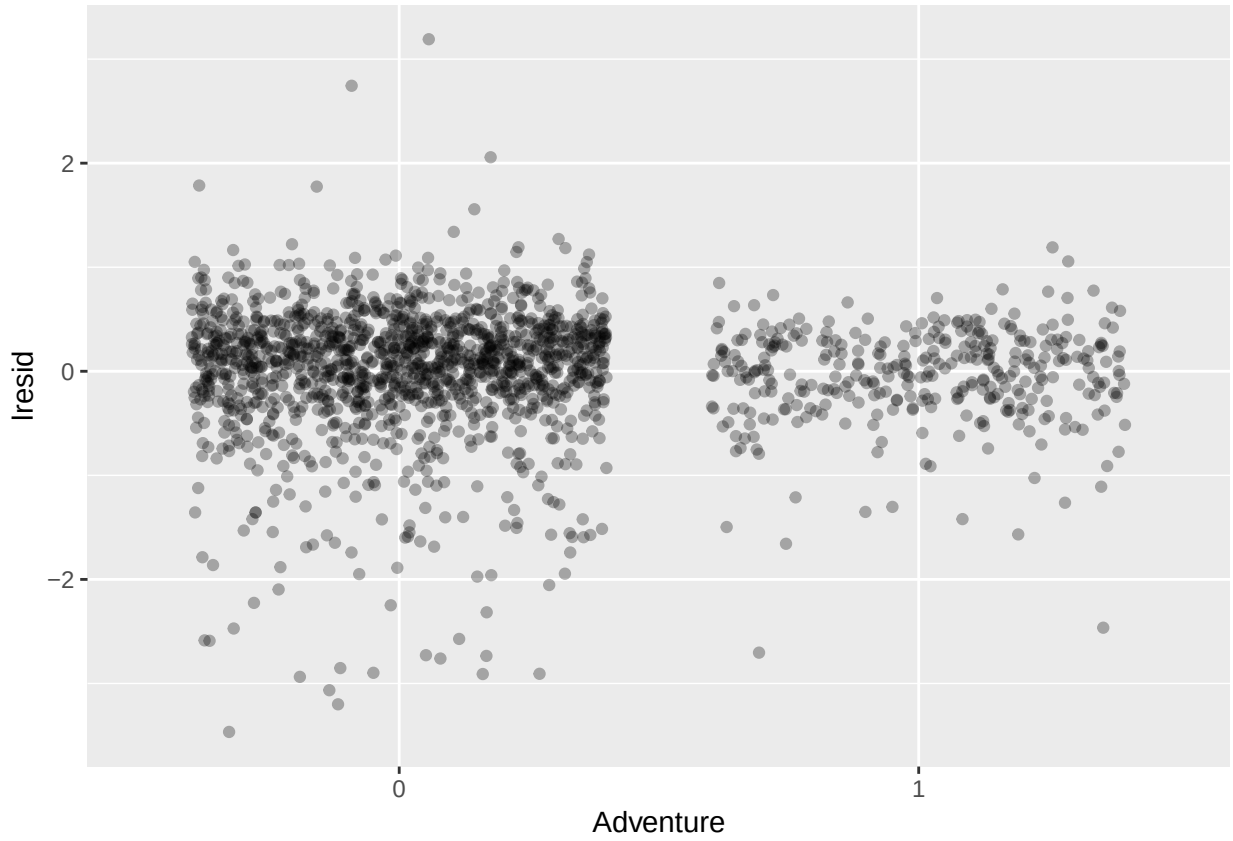

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



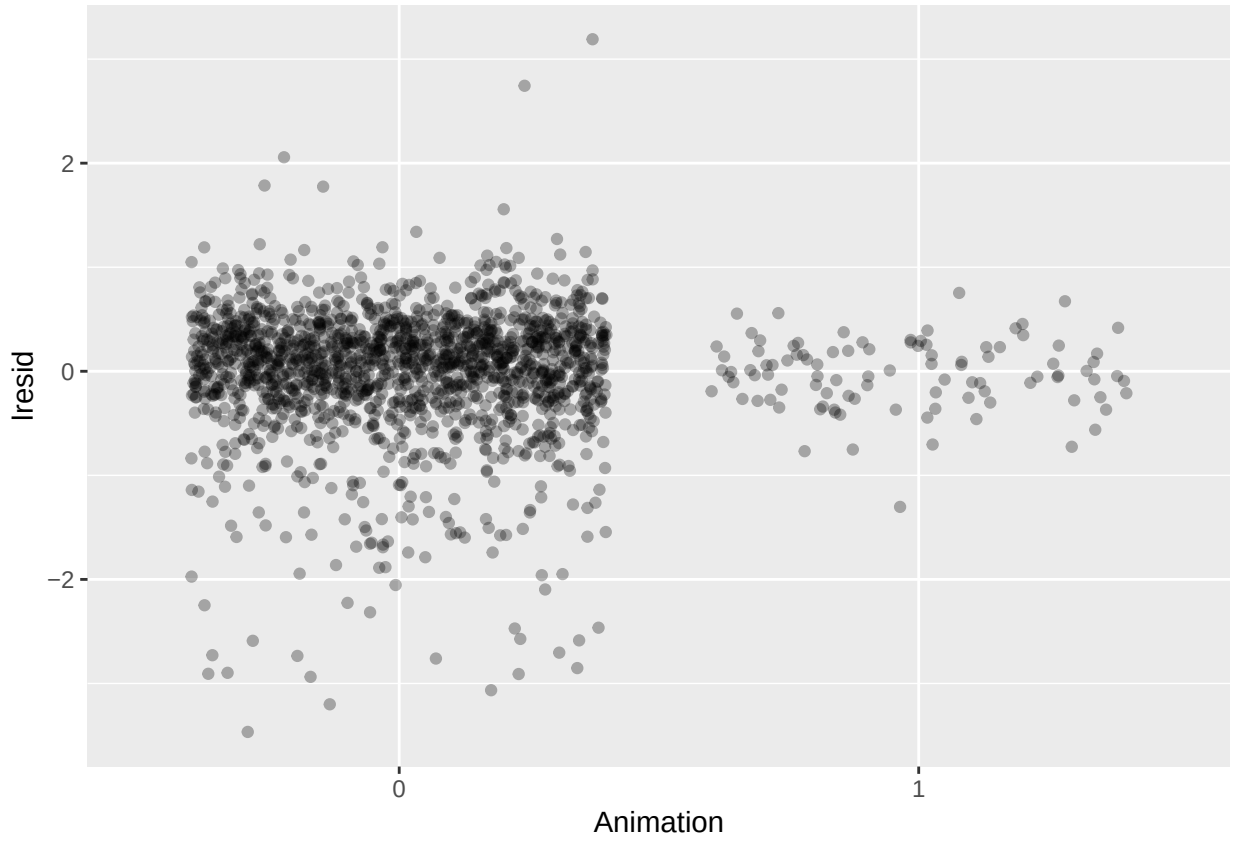

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



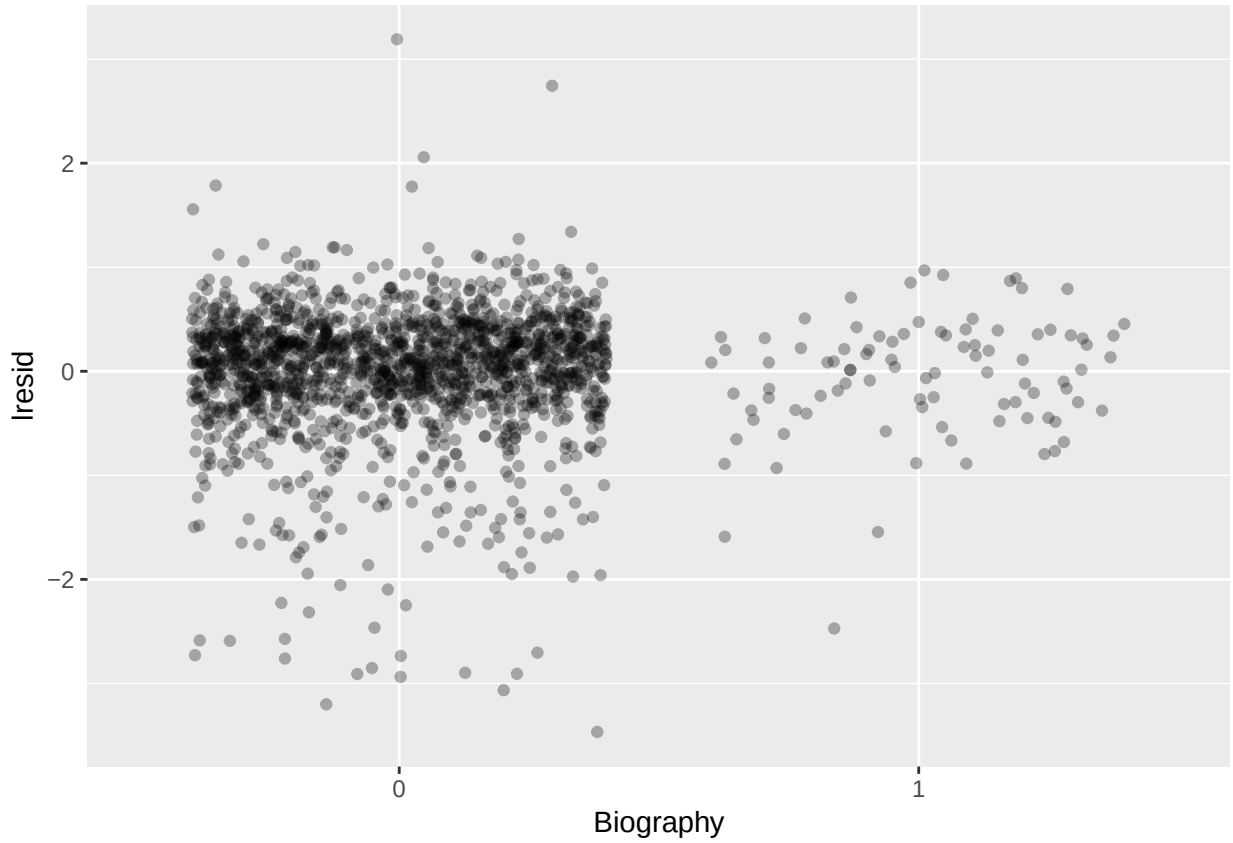

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



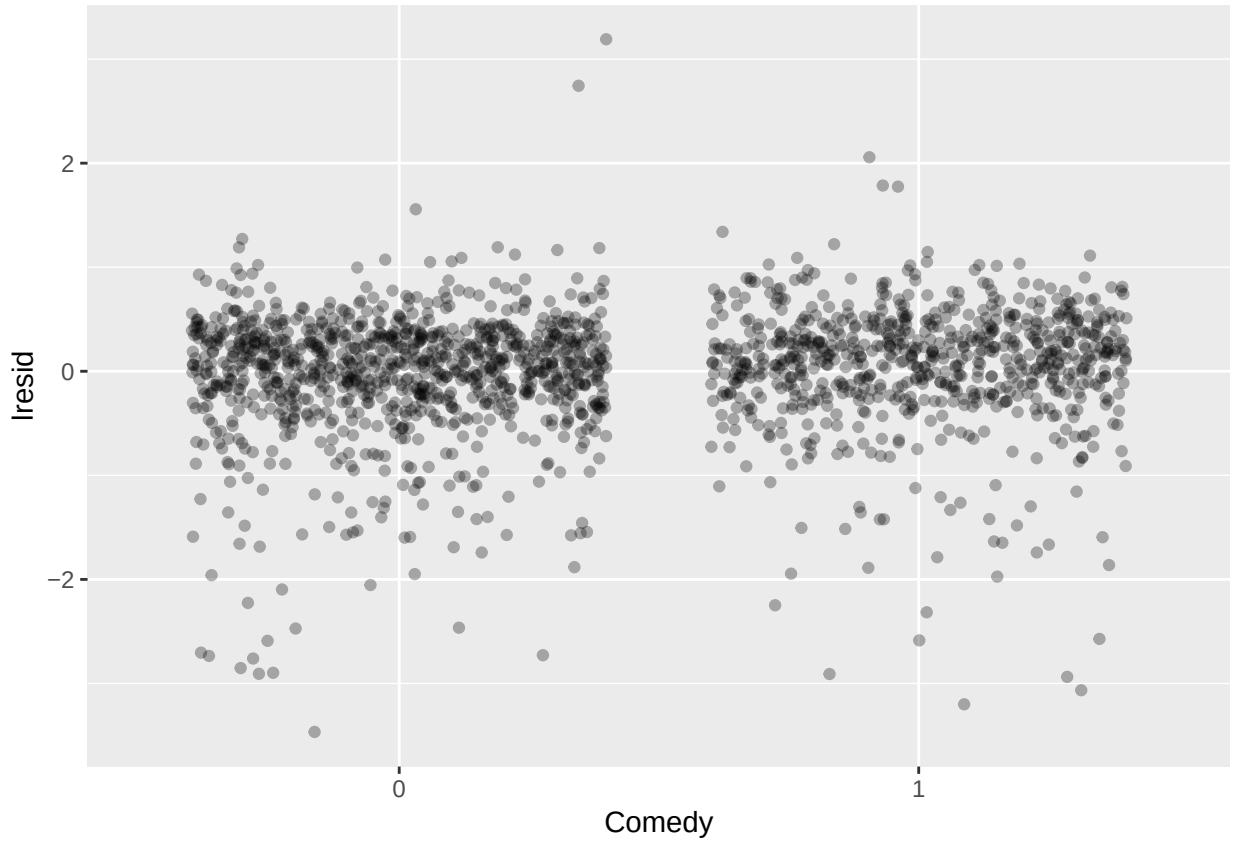

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



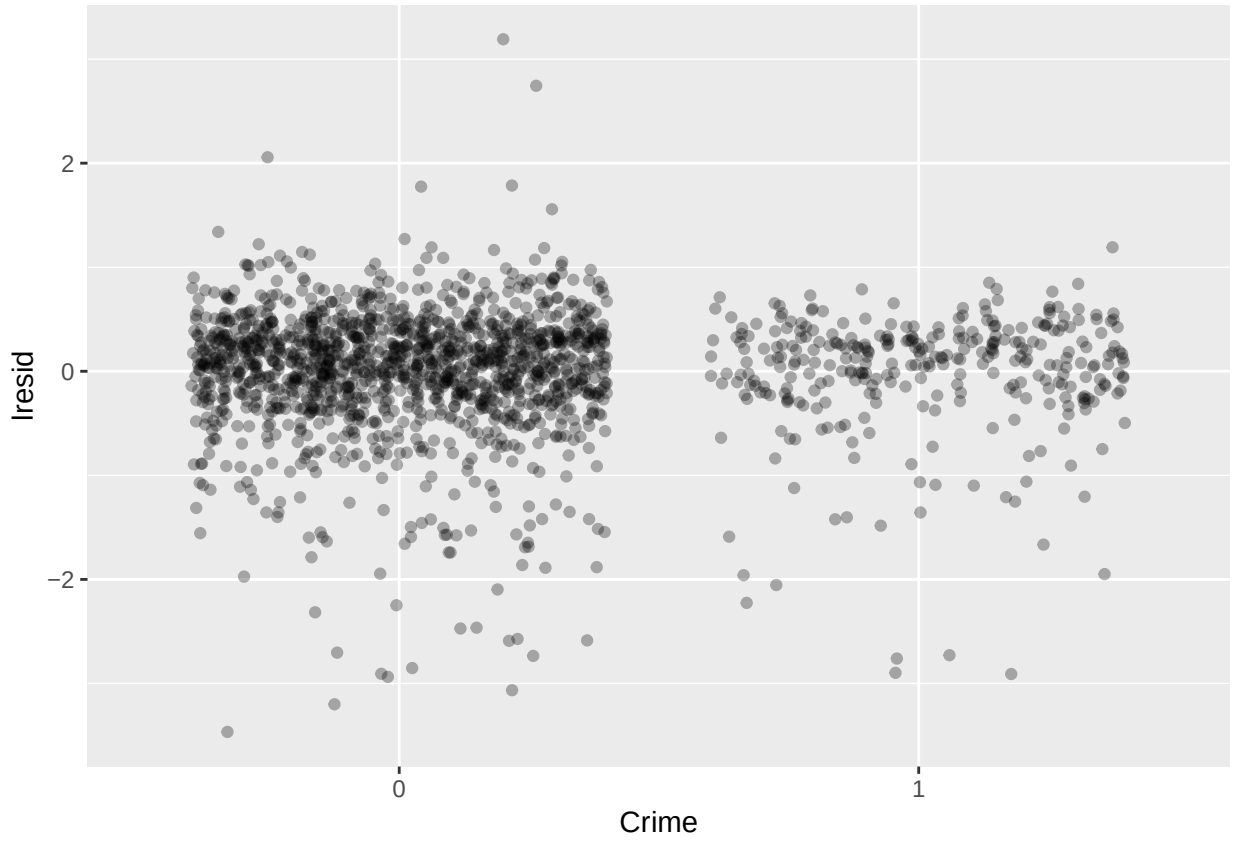

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



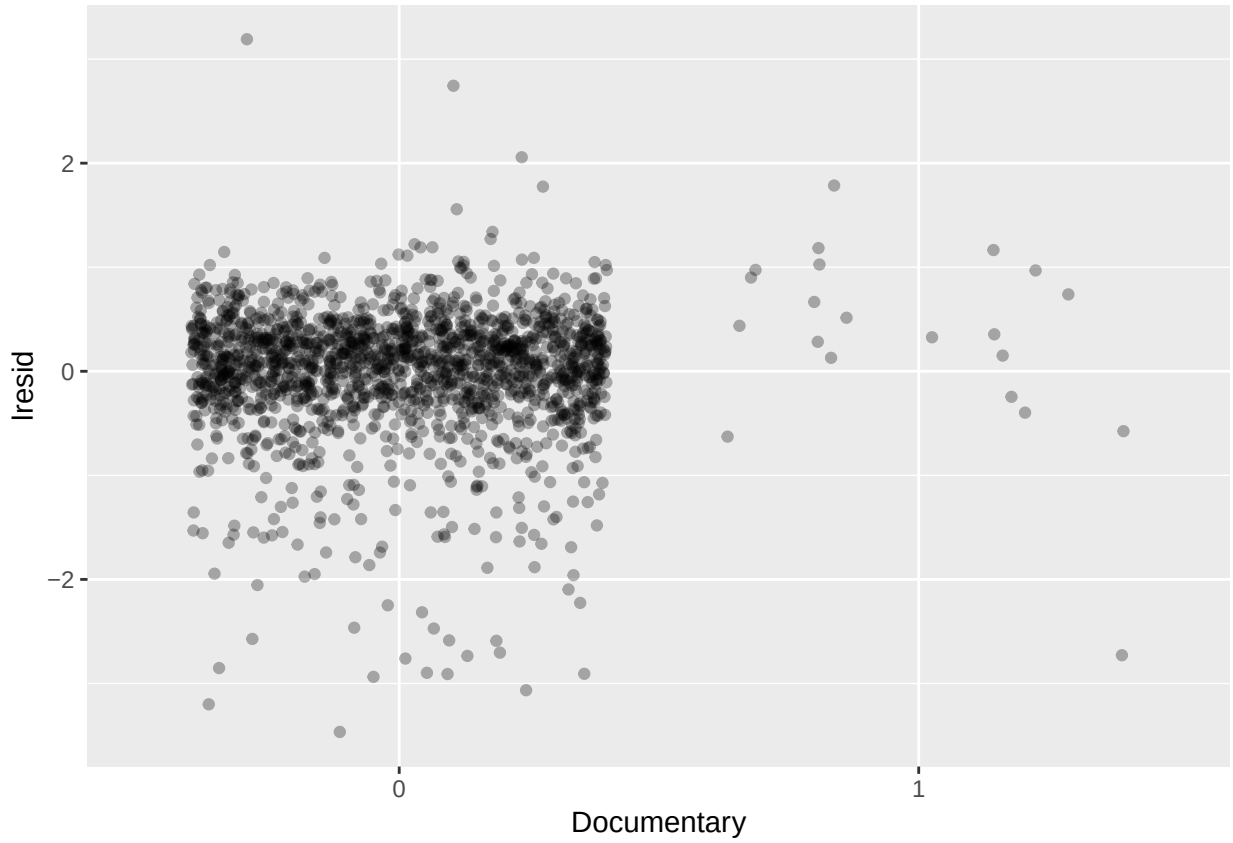

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



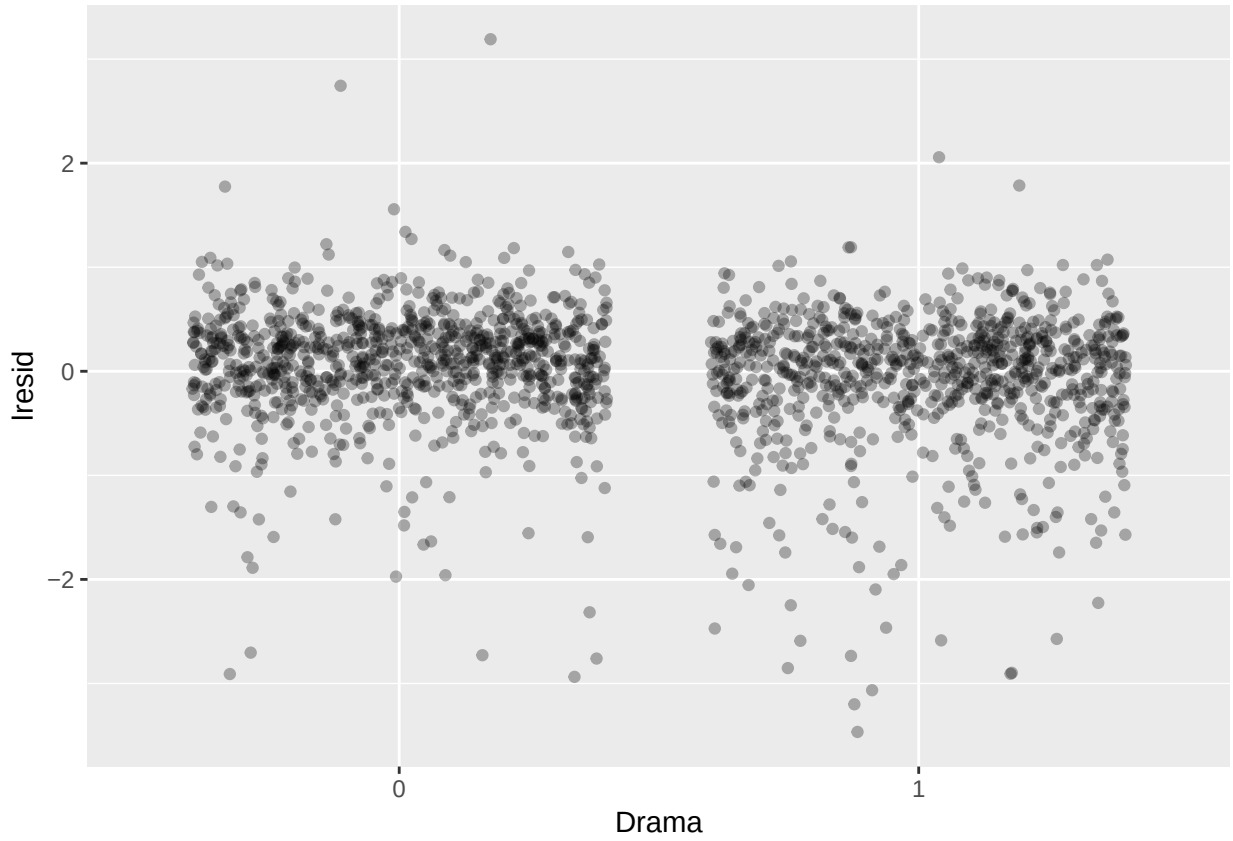

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



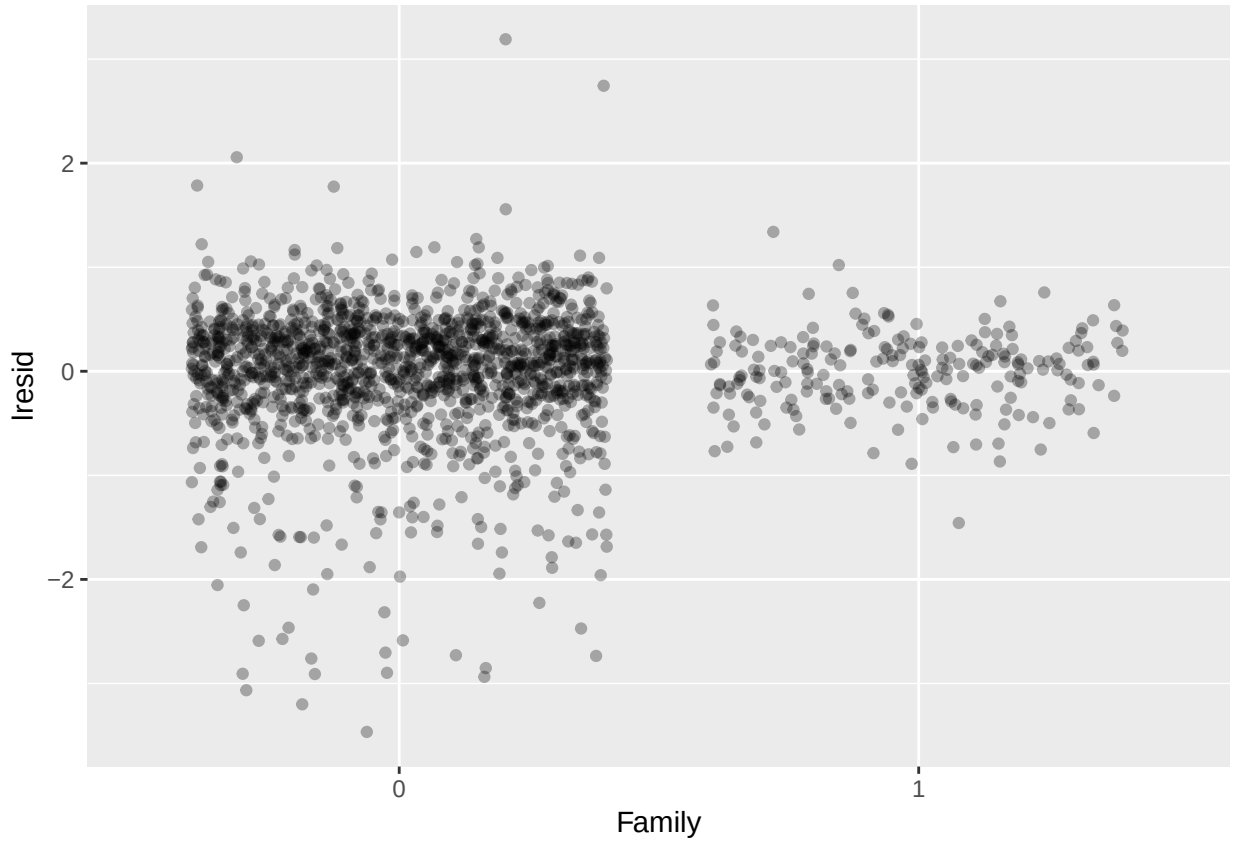

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



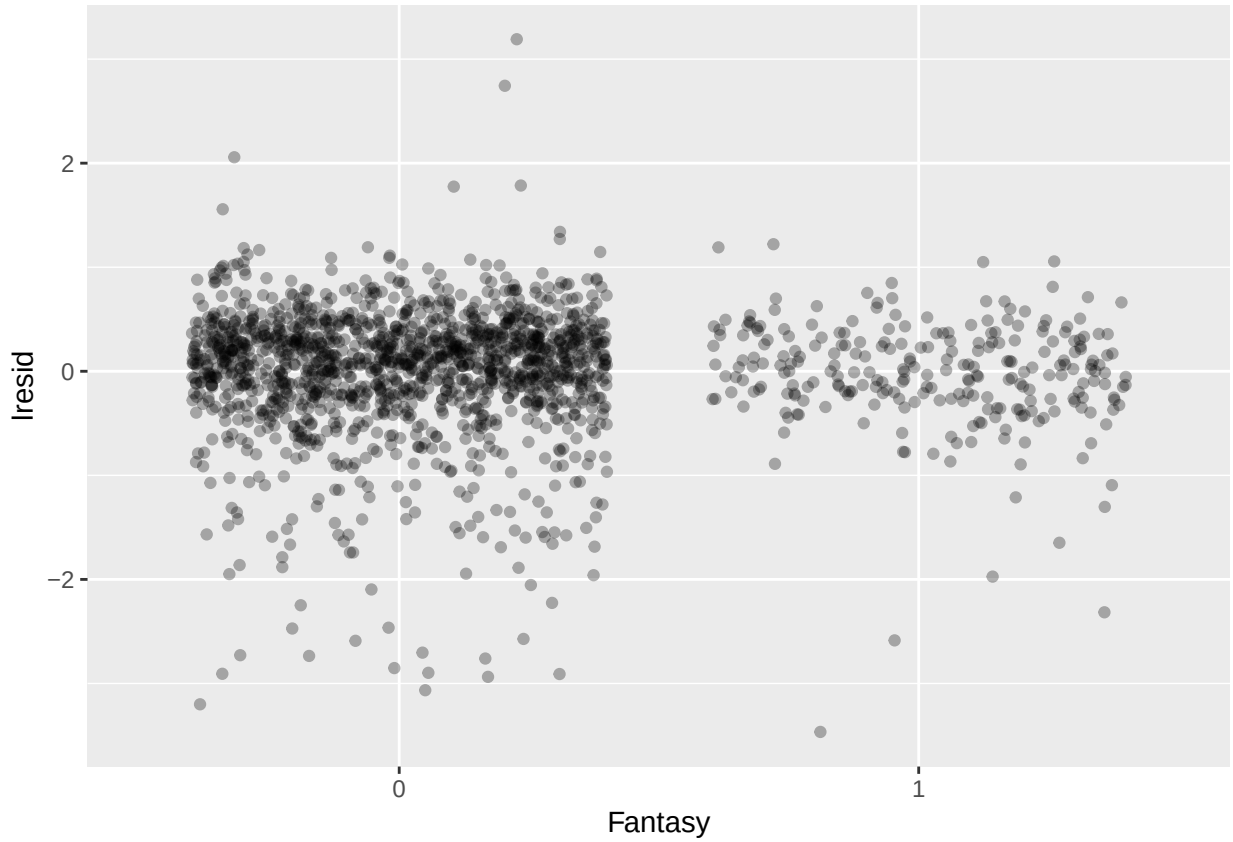

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



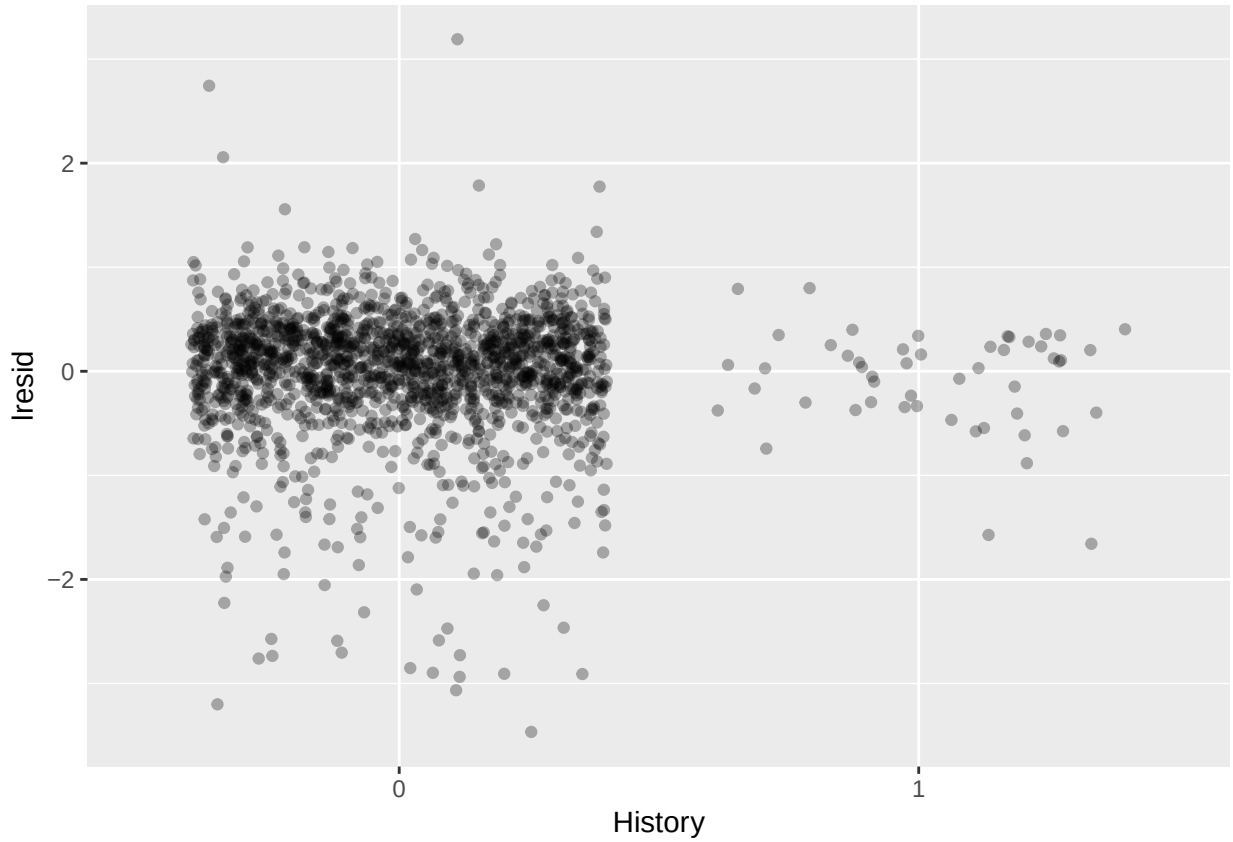

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



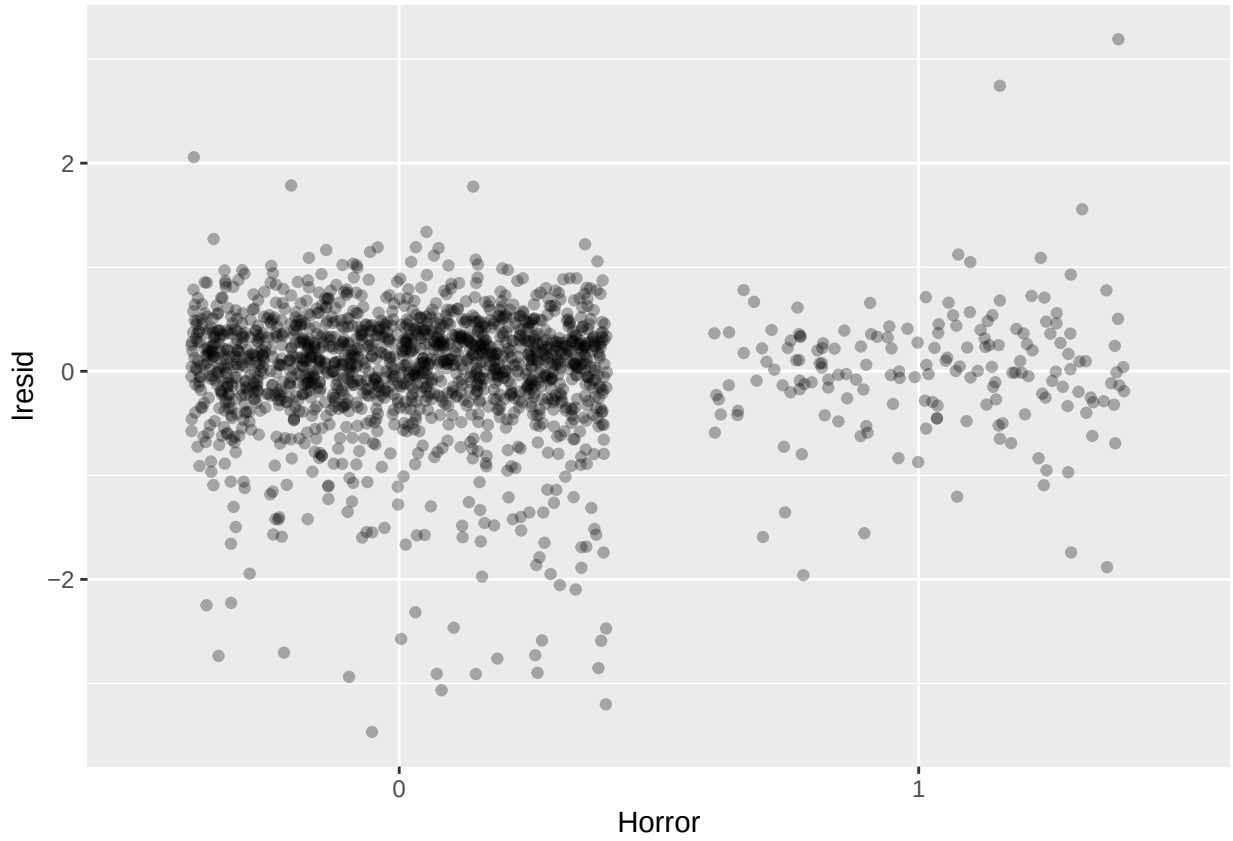

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



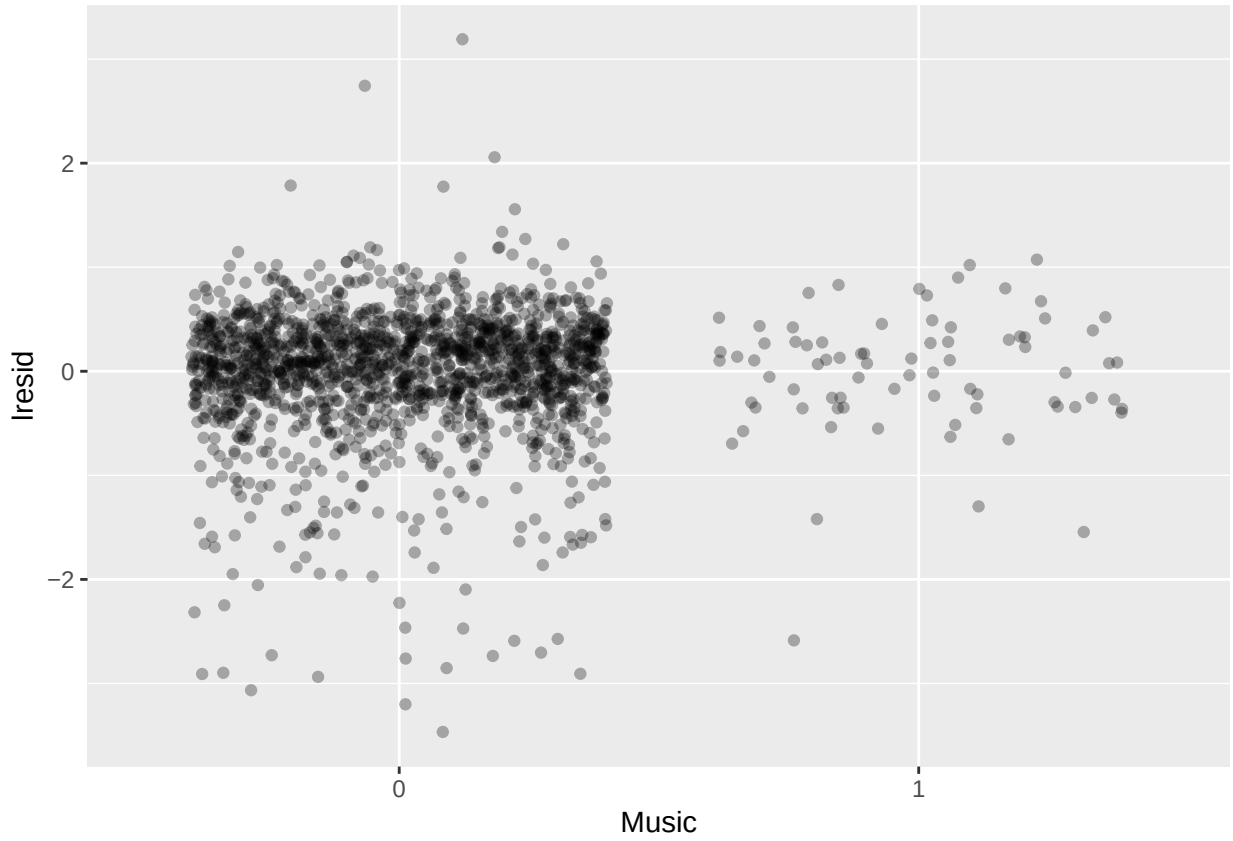

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



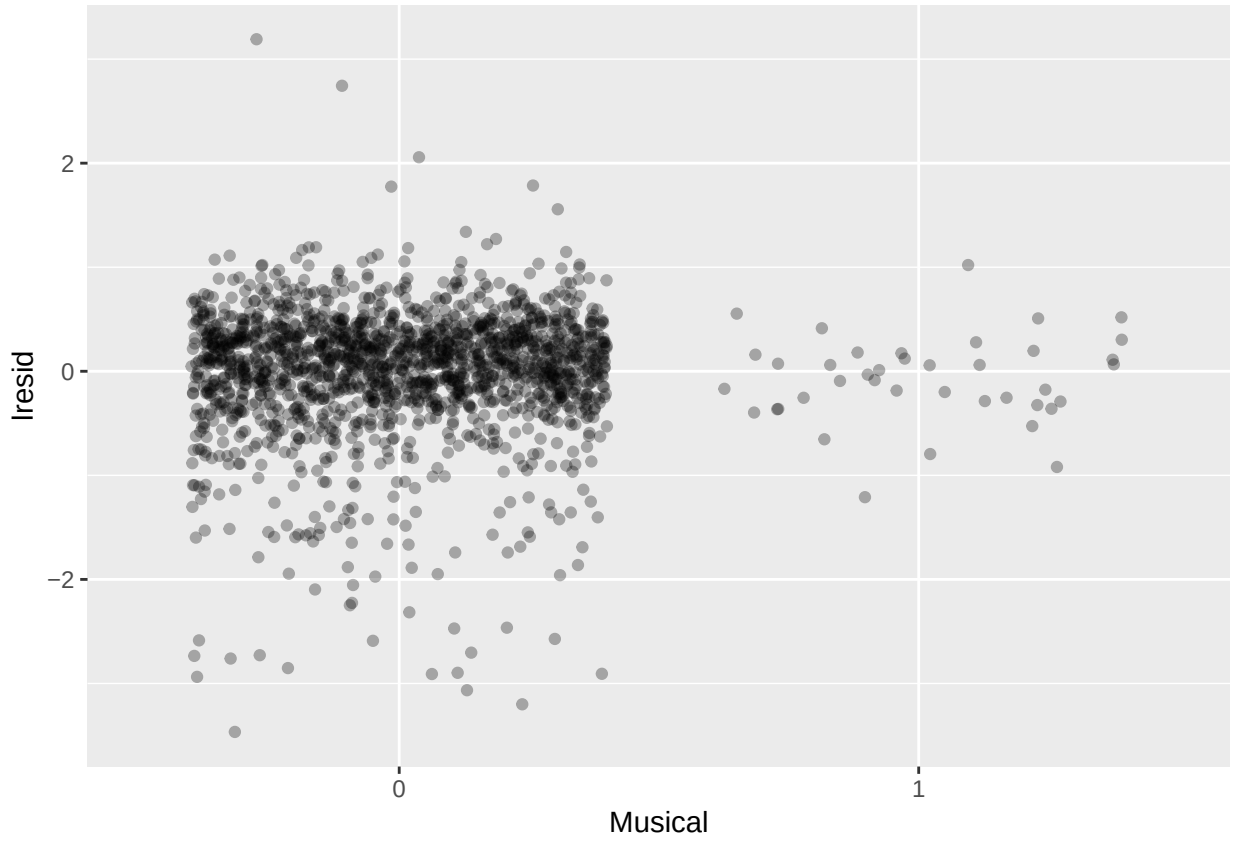

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



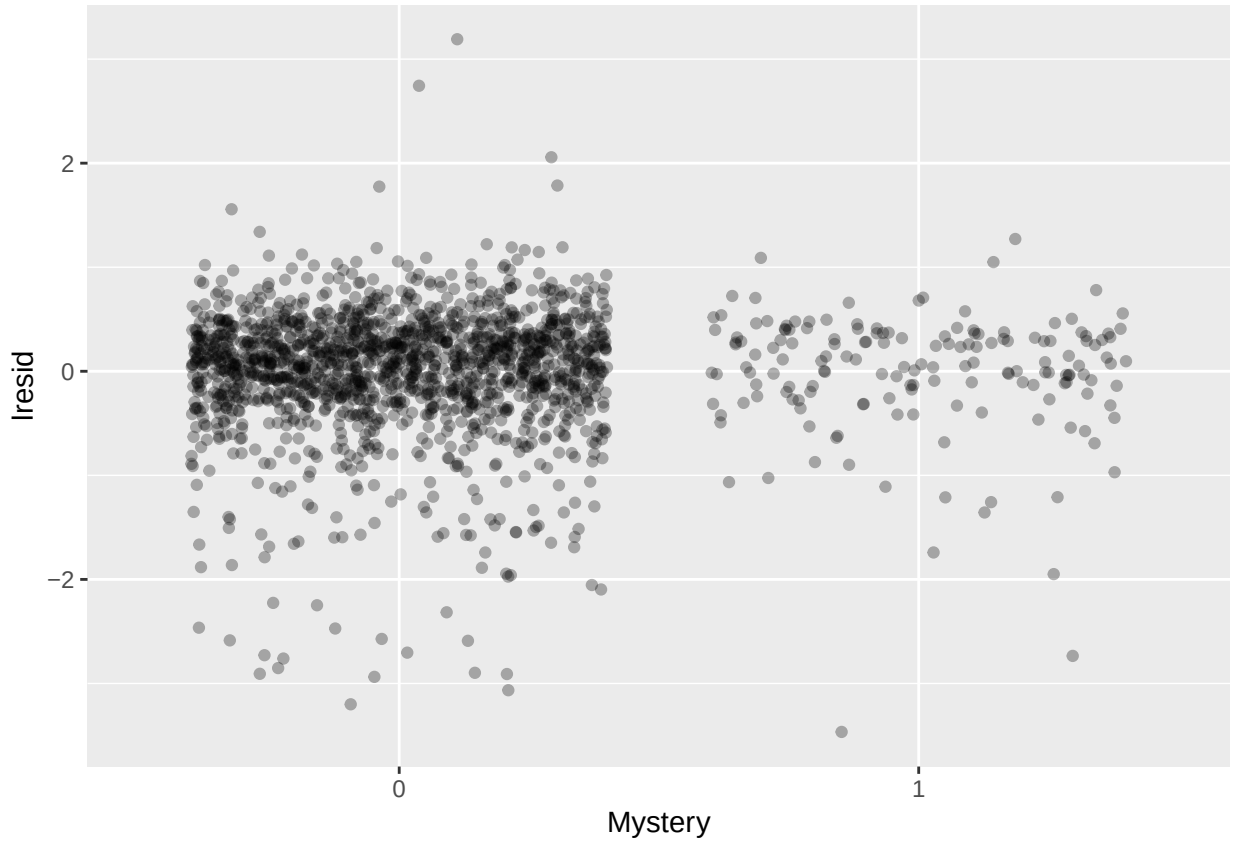

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



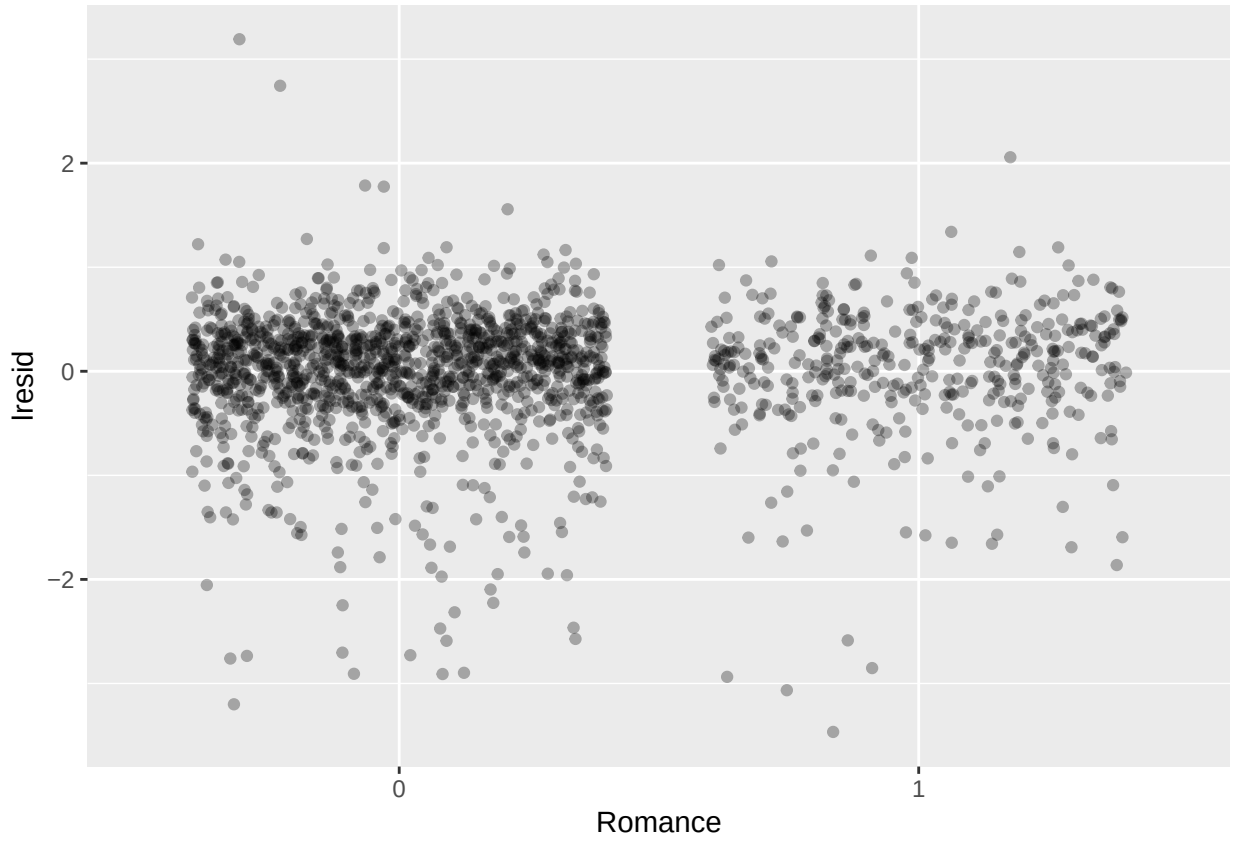

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



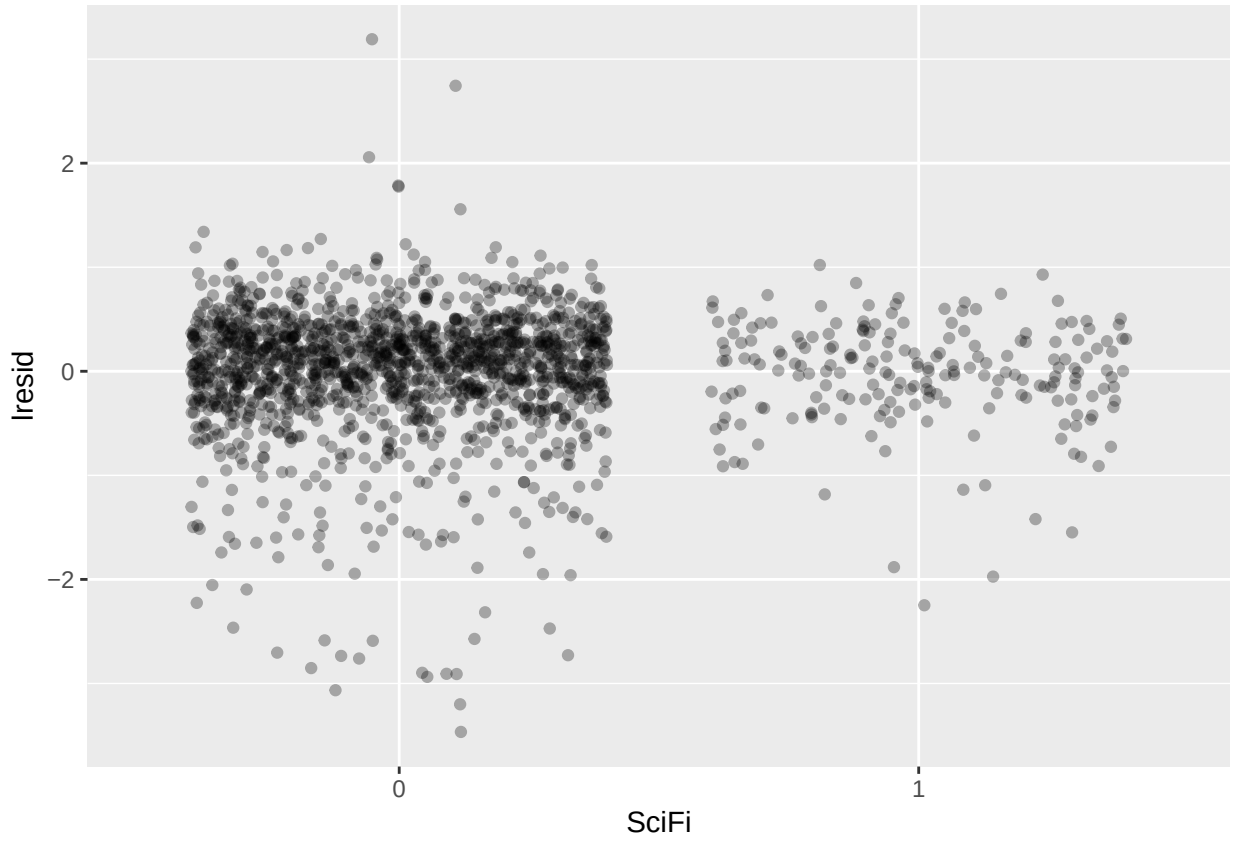

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



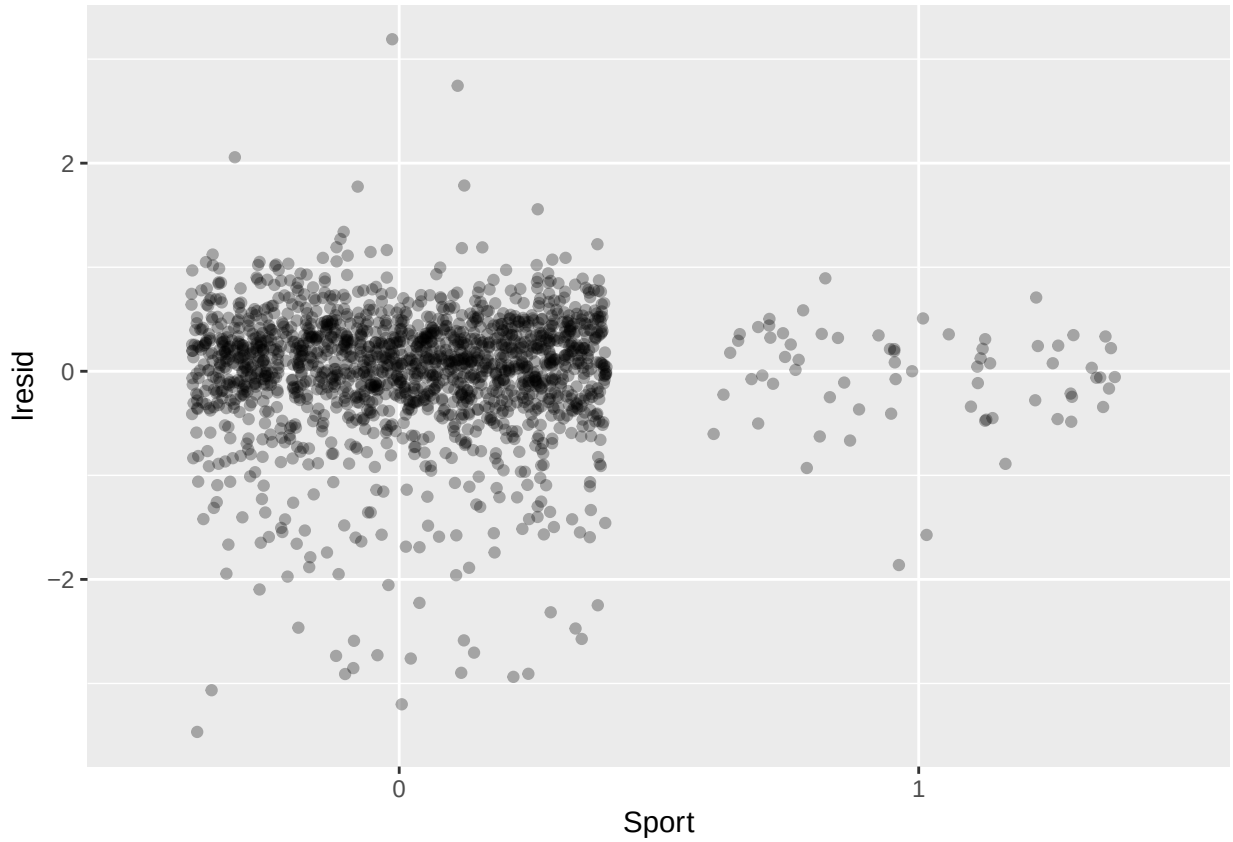

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



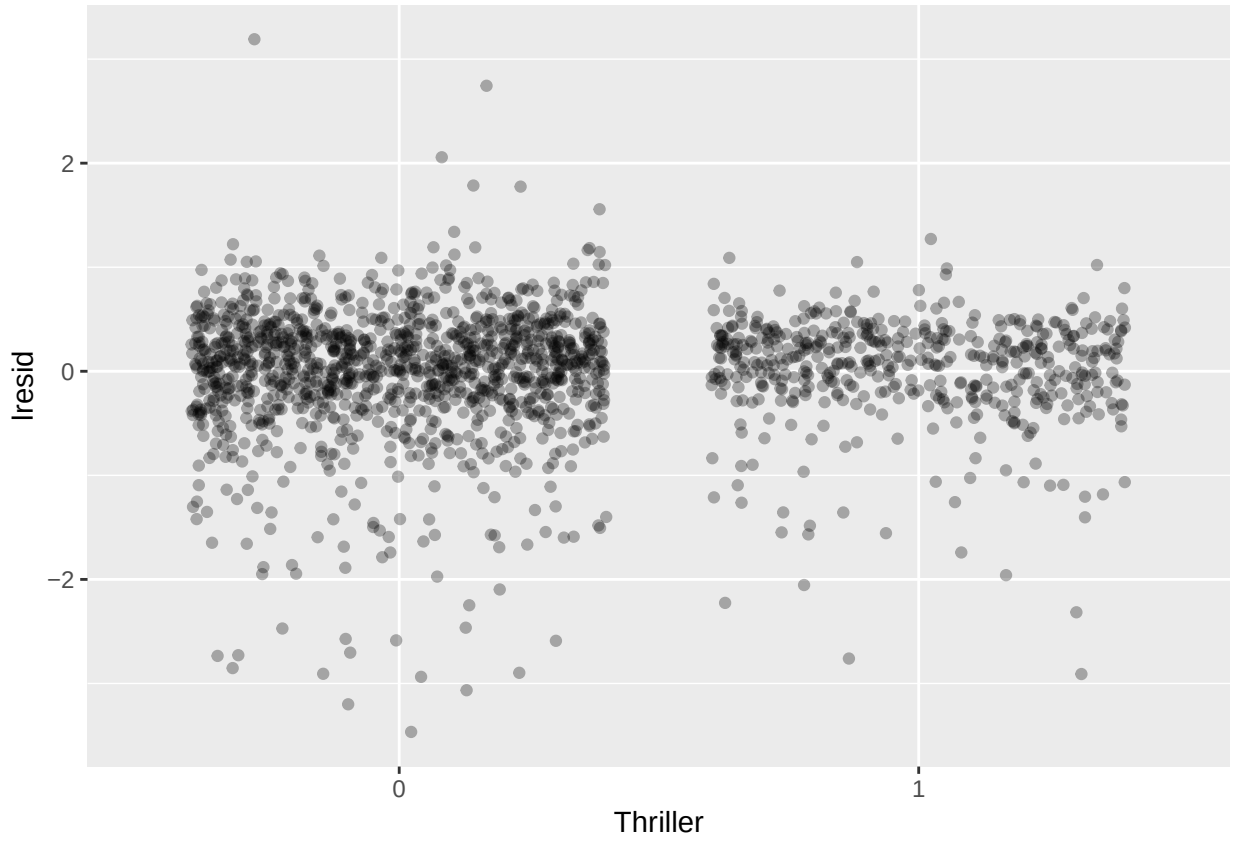

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



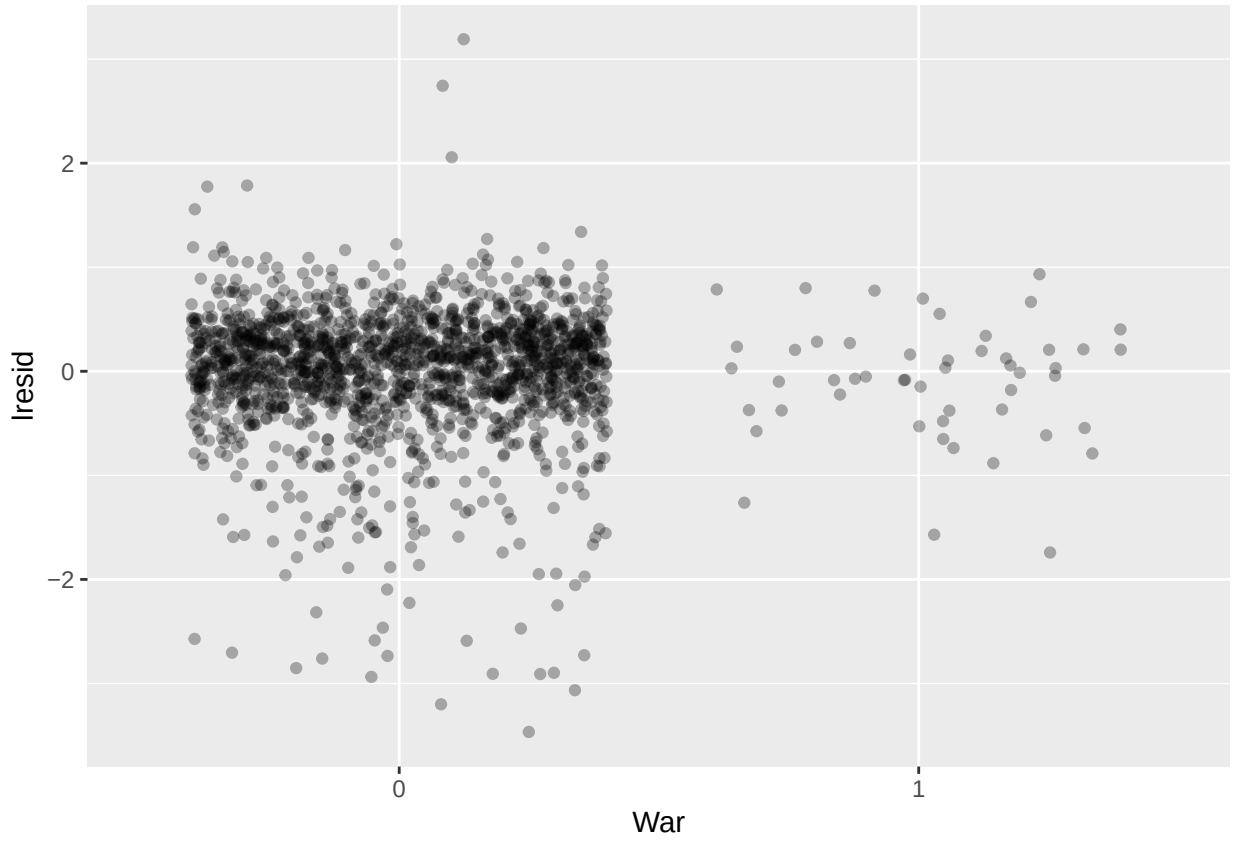

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



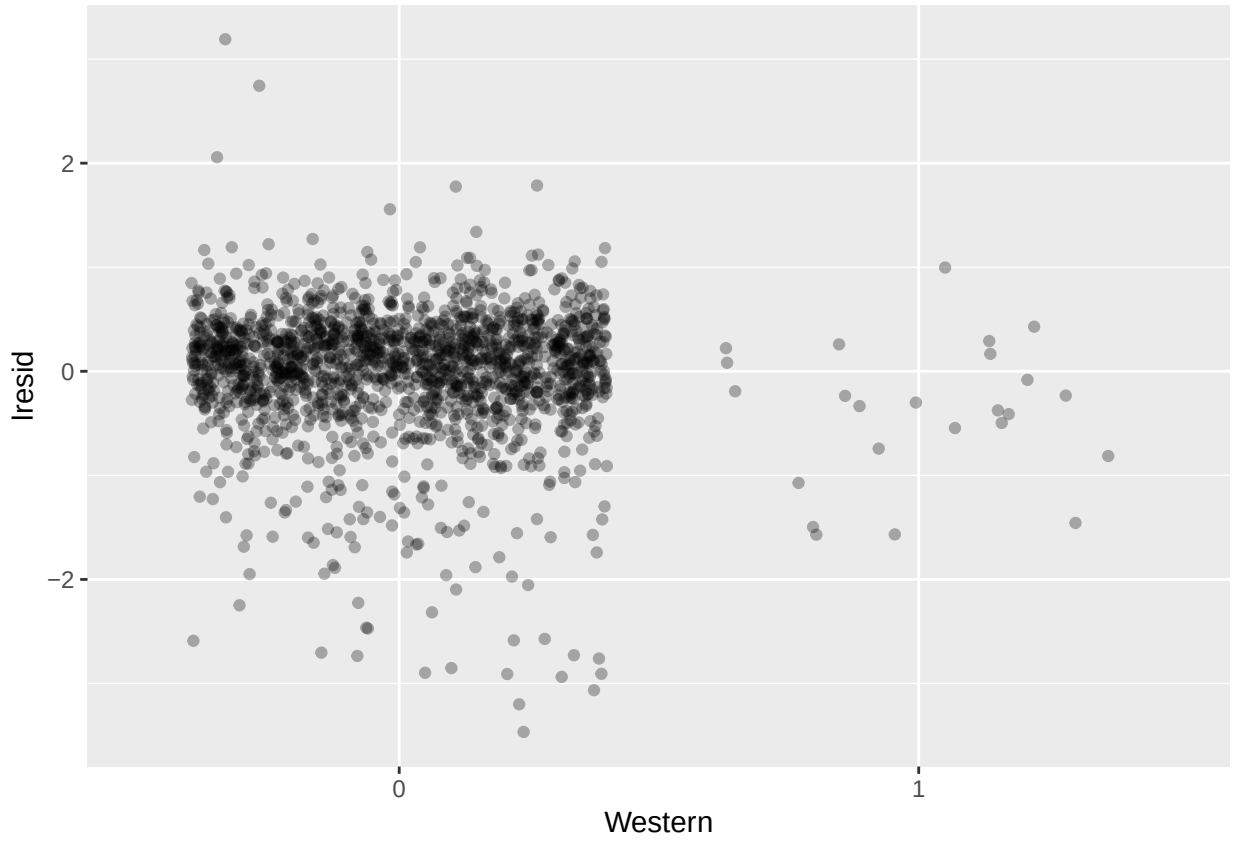

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



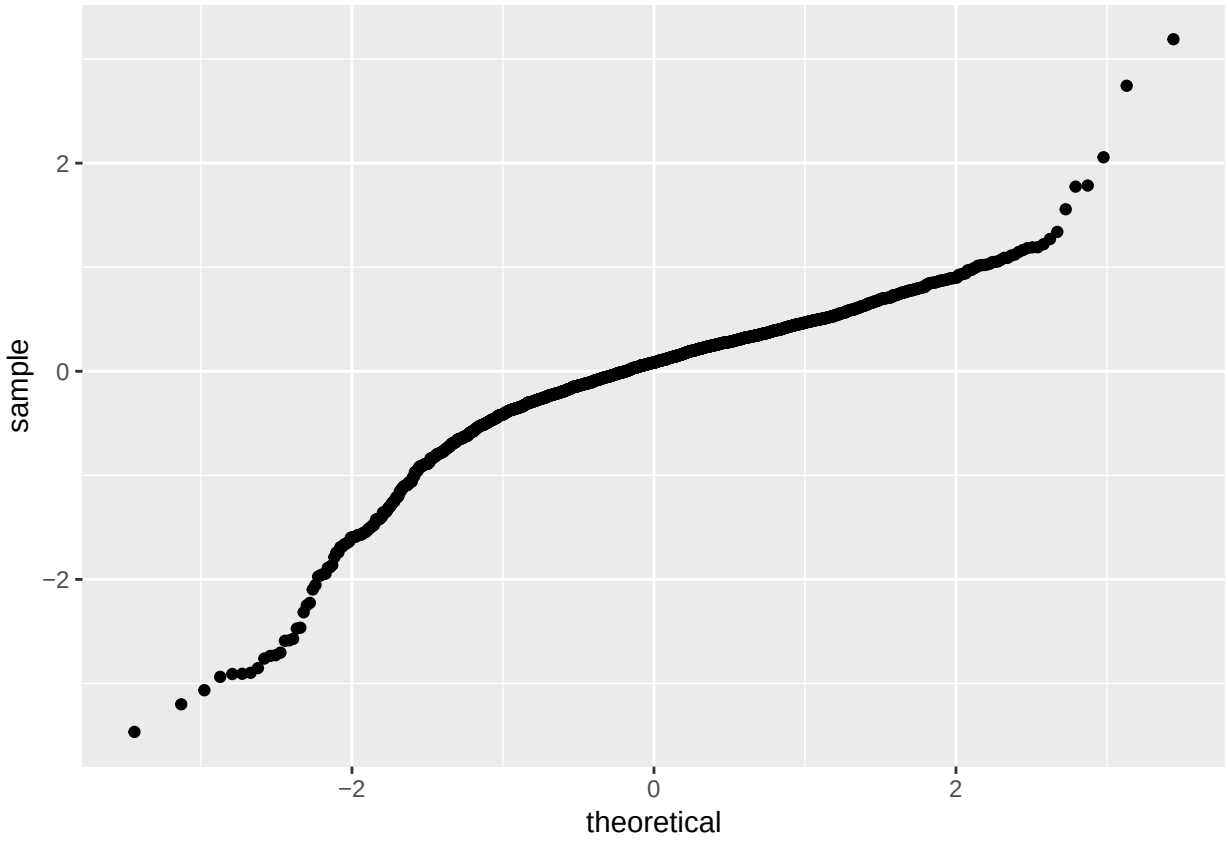

```
## Warning: Removed 132 rows containing missing values (geom_point).
```




```
## Warning: Removed 132 rows containing missing values (geom_point).
```

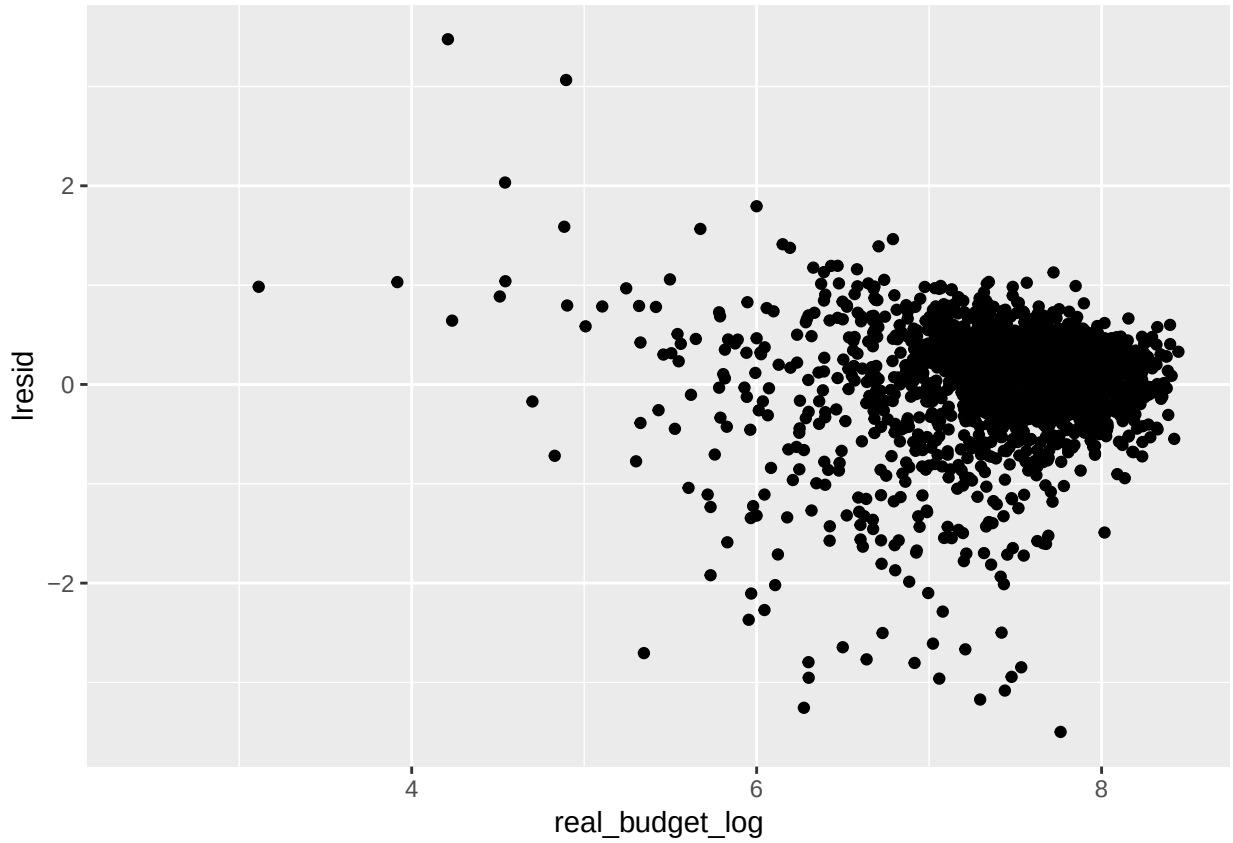



```
## Warning: Removed 132 rows containing non-finite values (stat_qq).
```

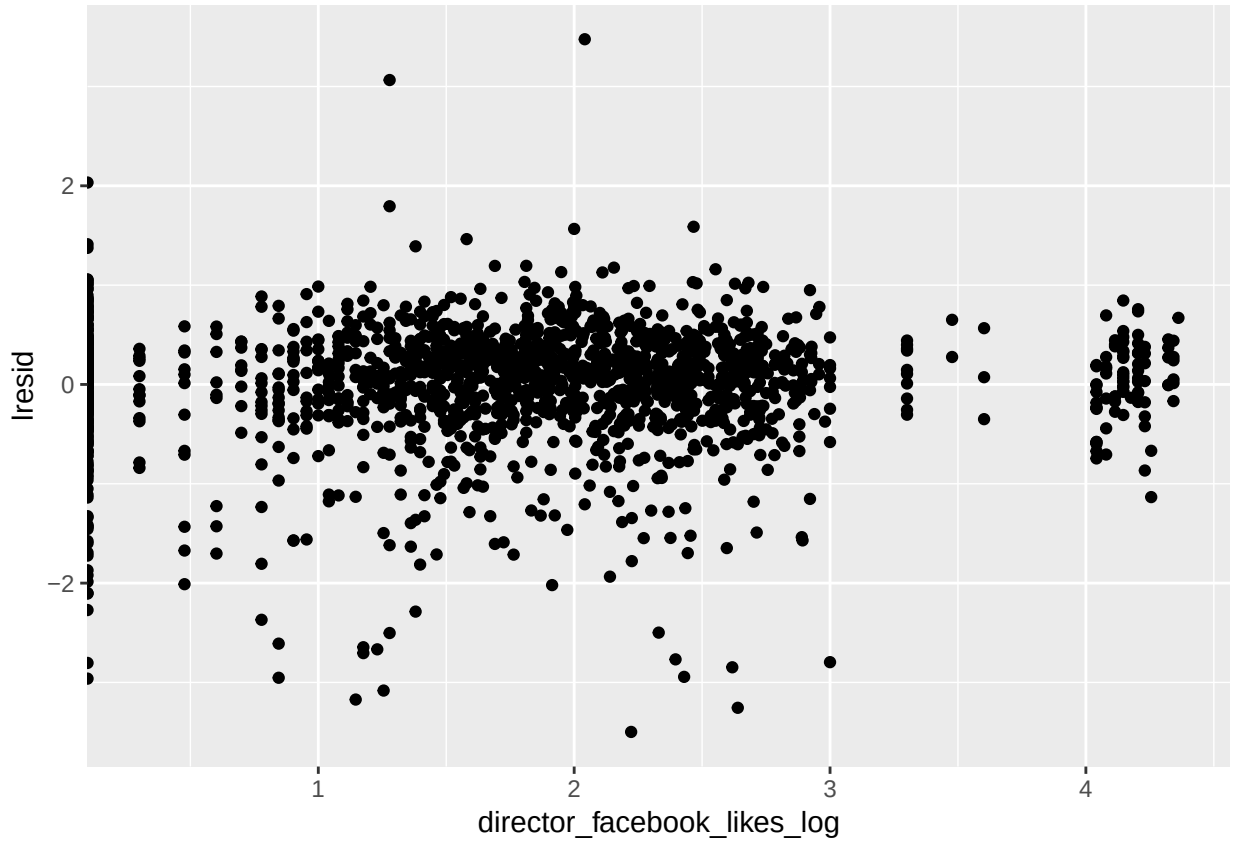



```
gr_resid(mod_simple_plus2)
```

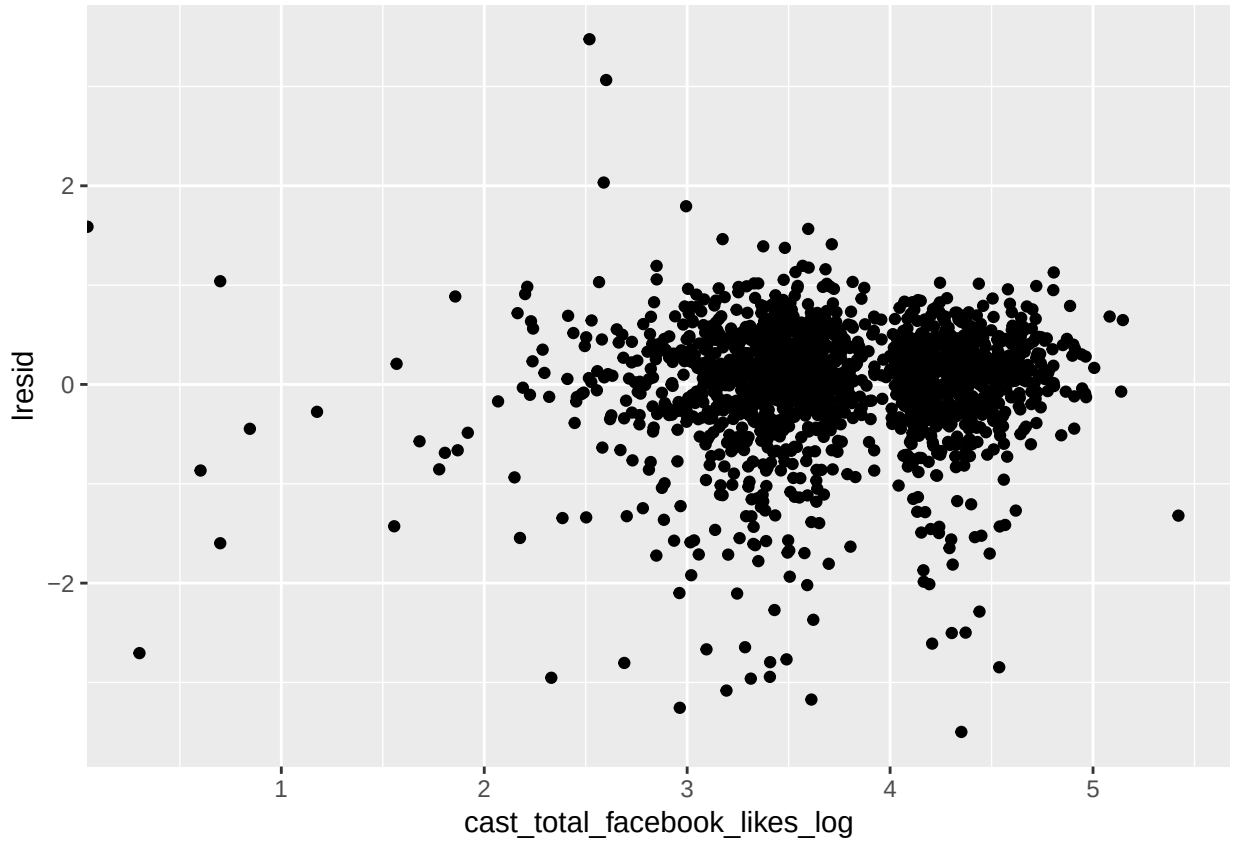
```
## Warning: Removed 132 rows containing missing values (geom_point).
```



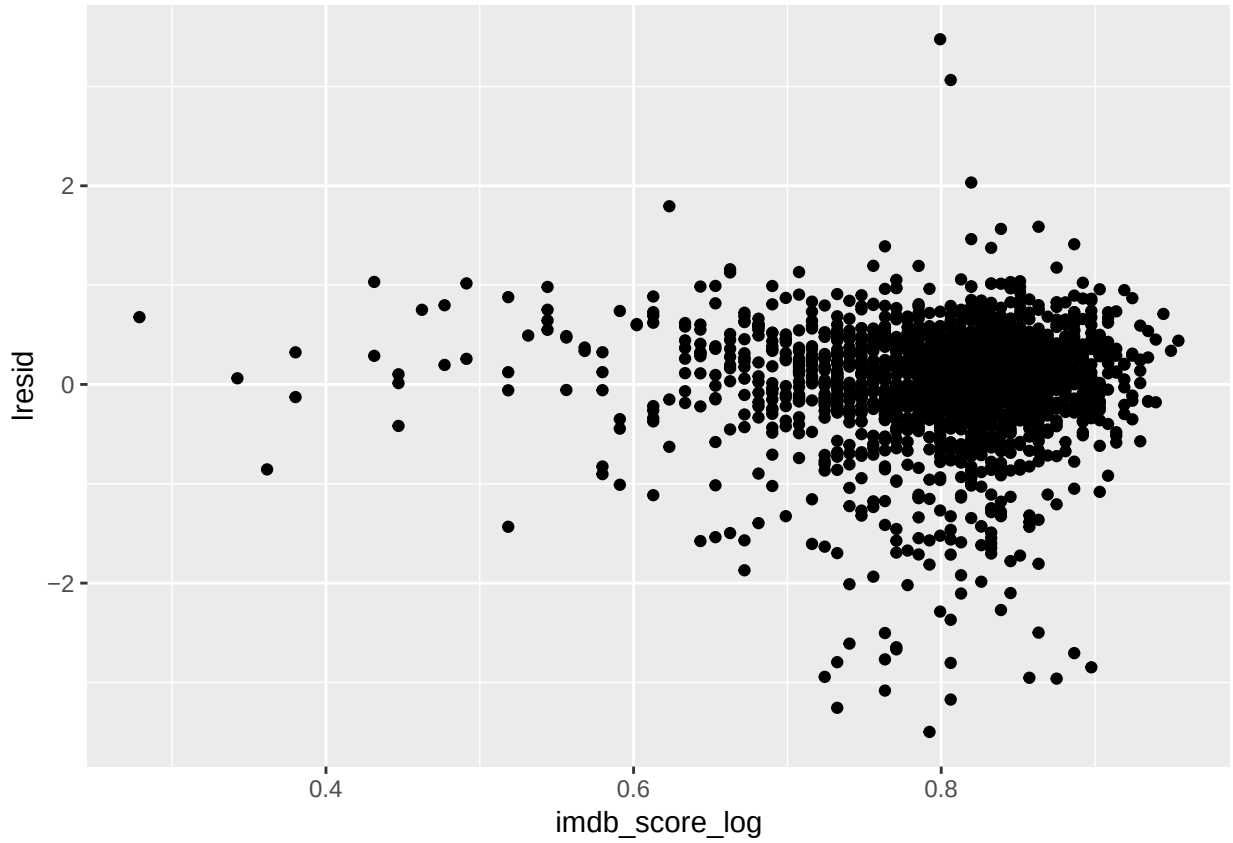

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



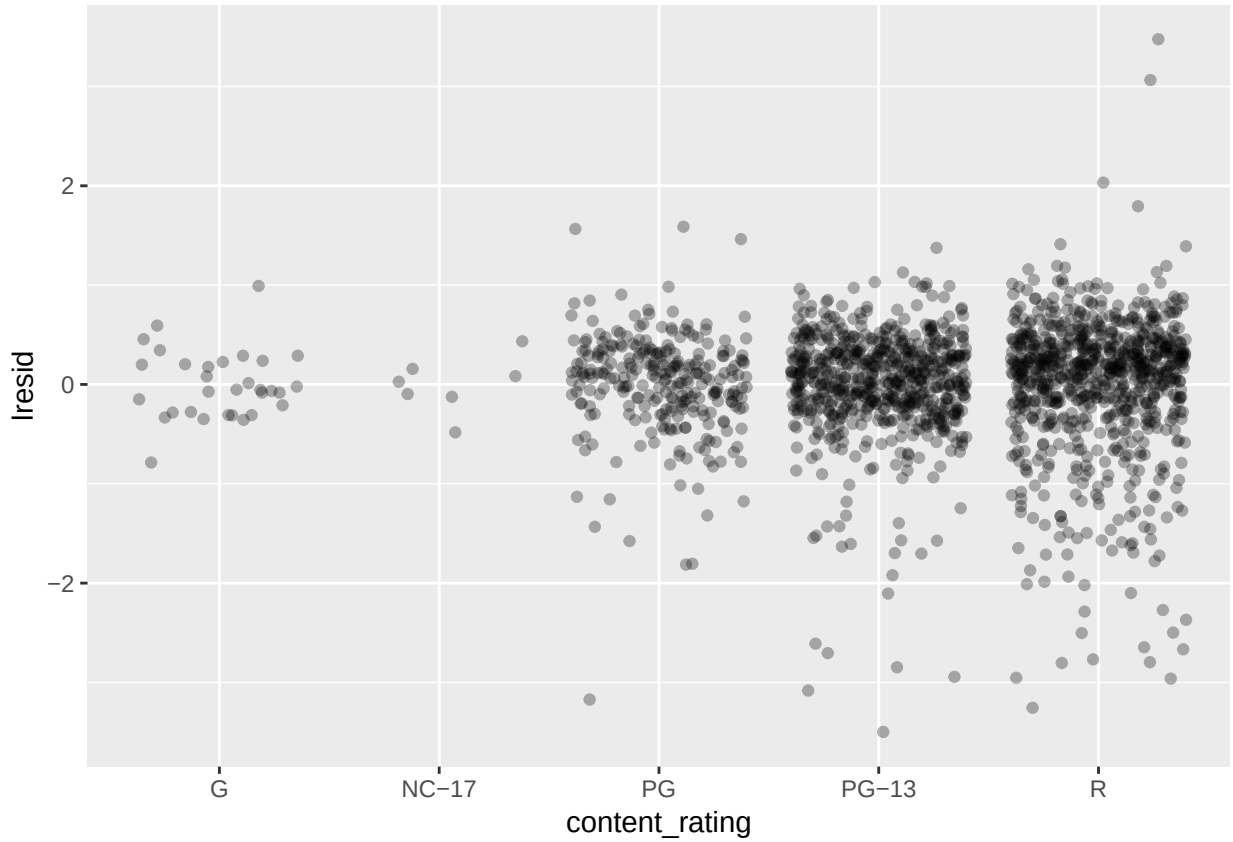

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



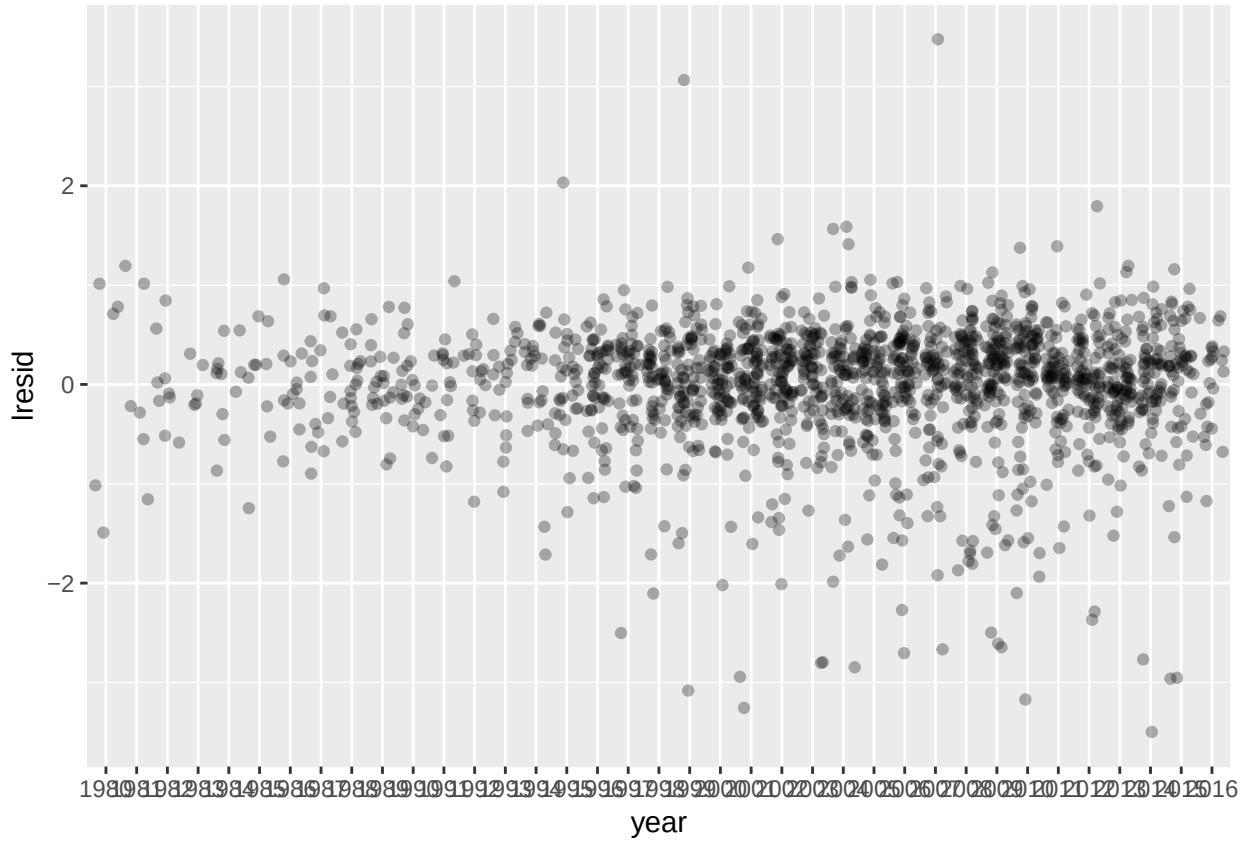

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



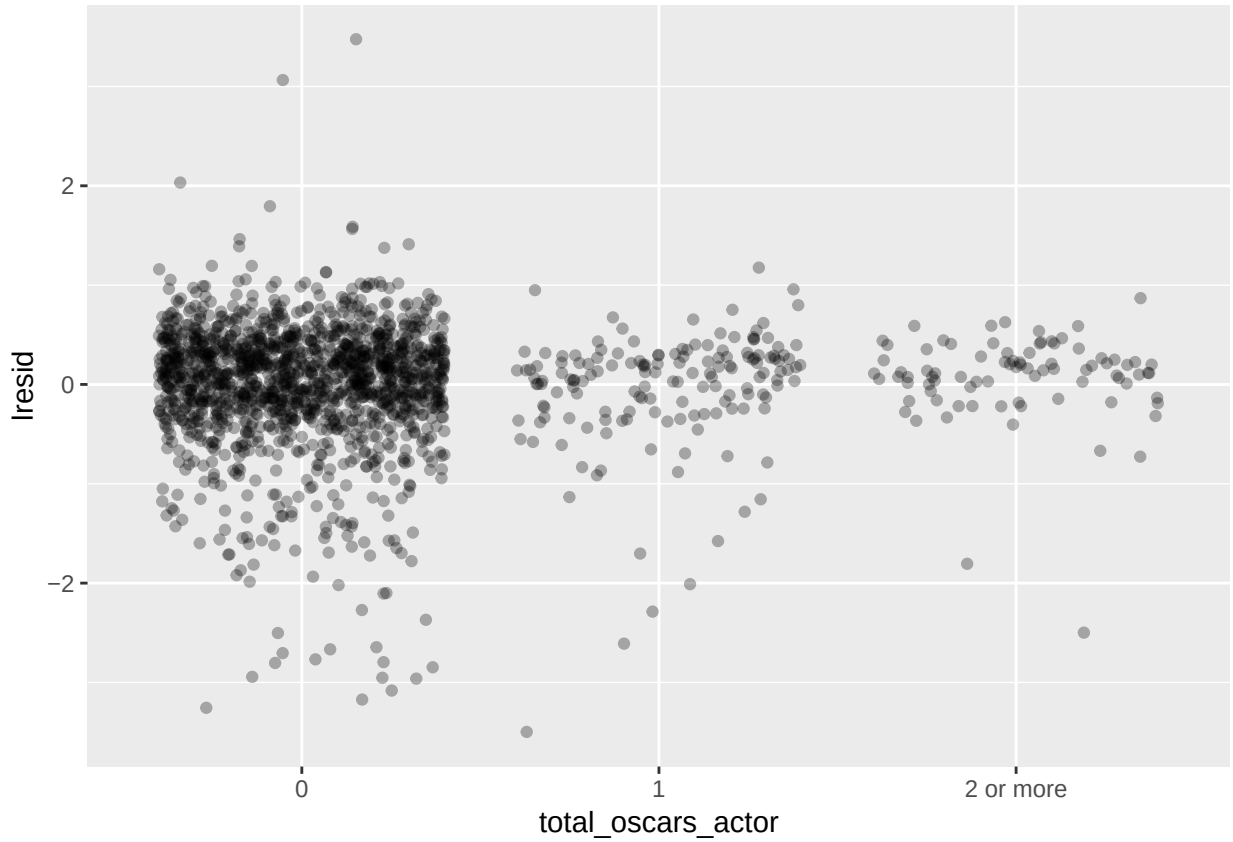

```
## Warning: Removed 94 rows containing missing values (geom_point).
```



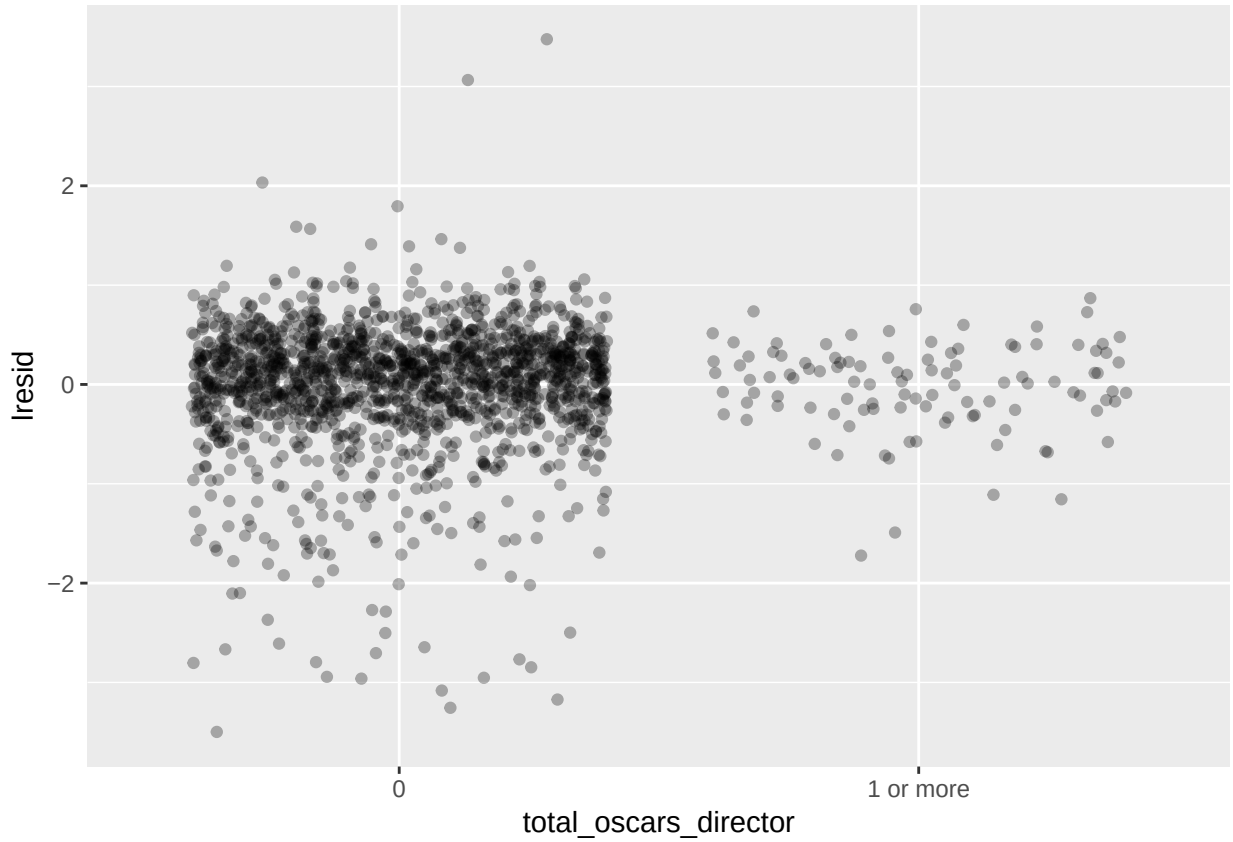

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



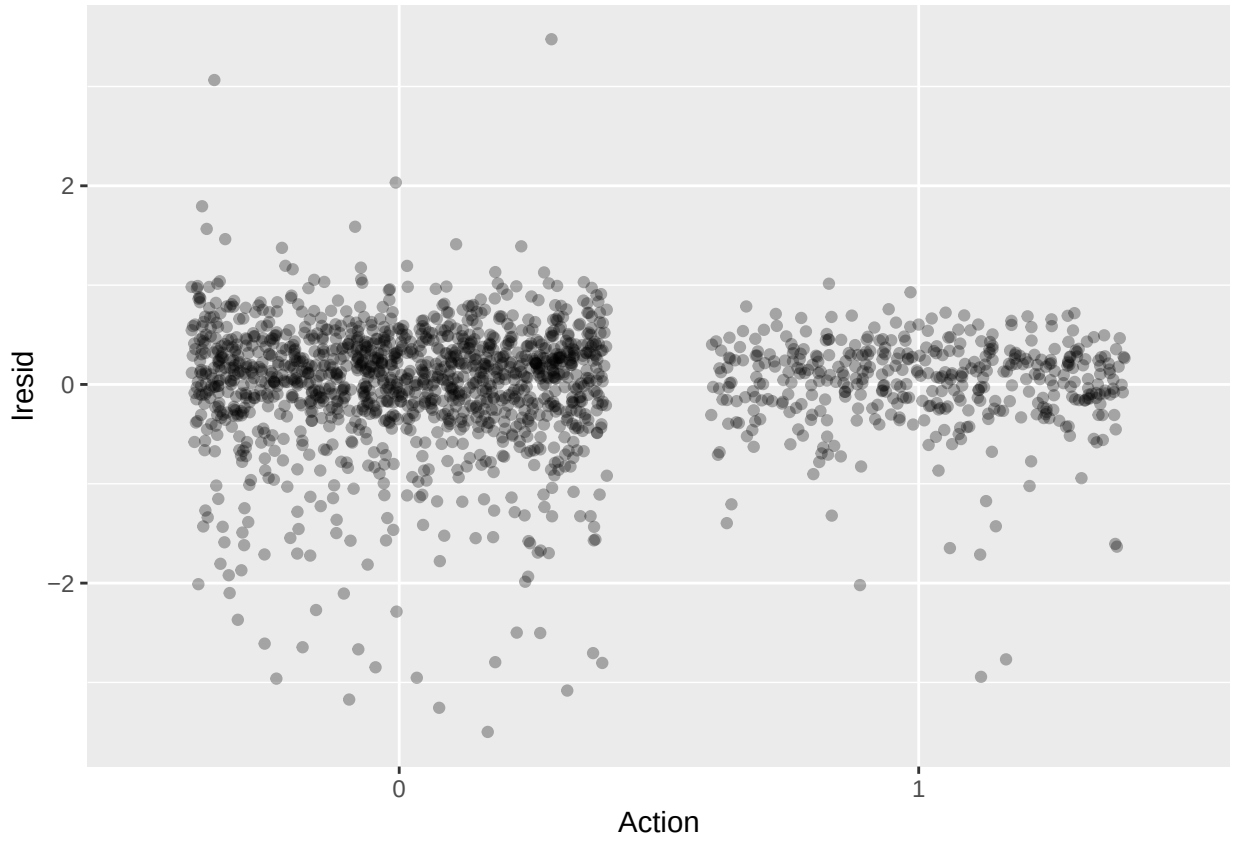

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



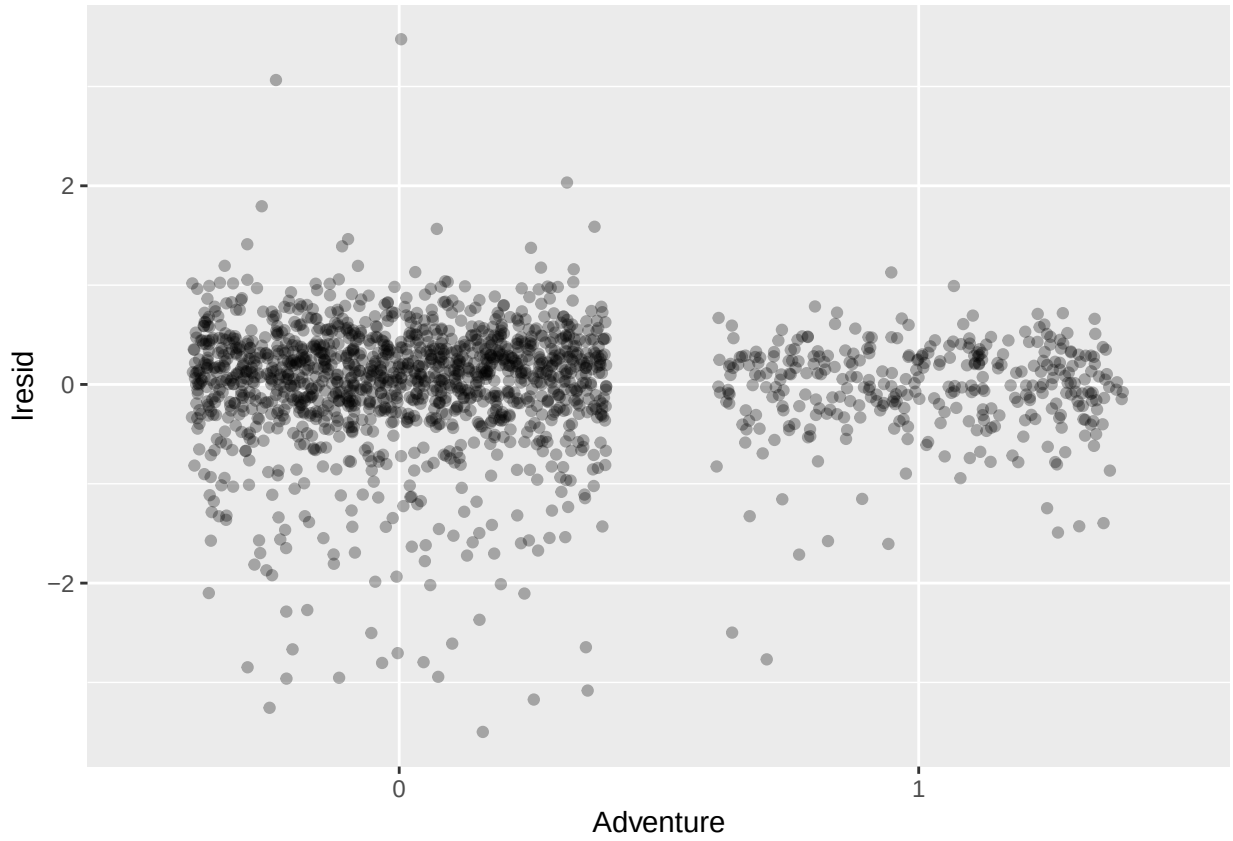

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



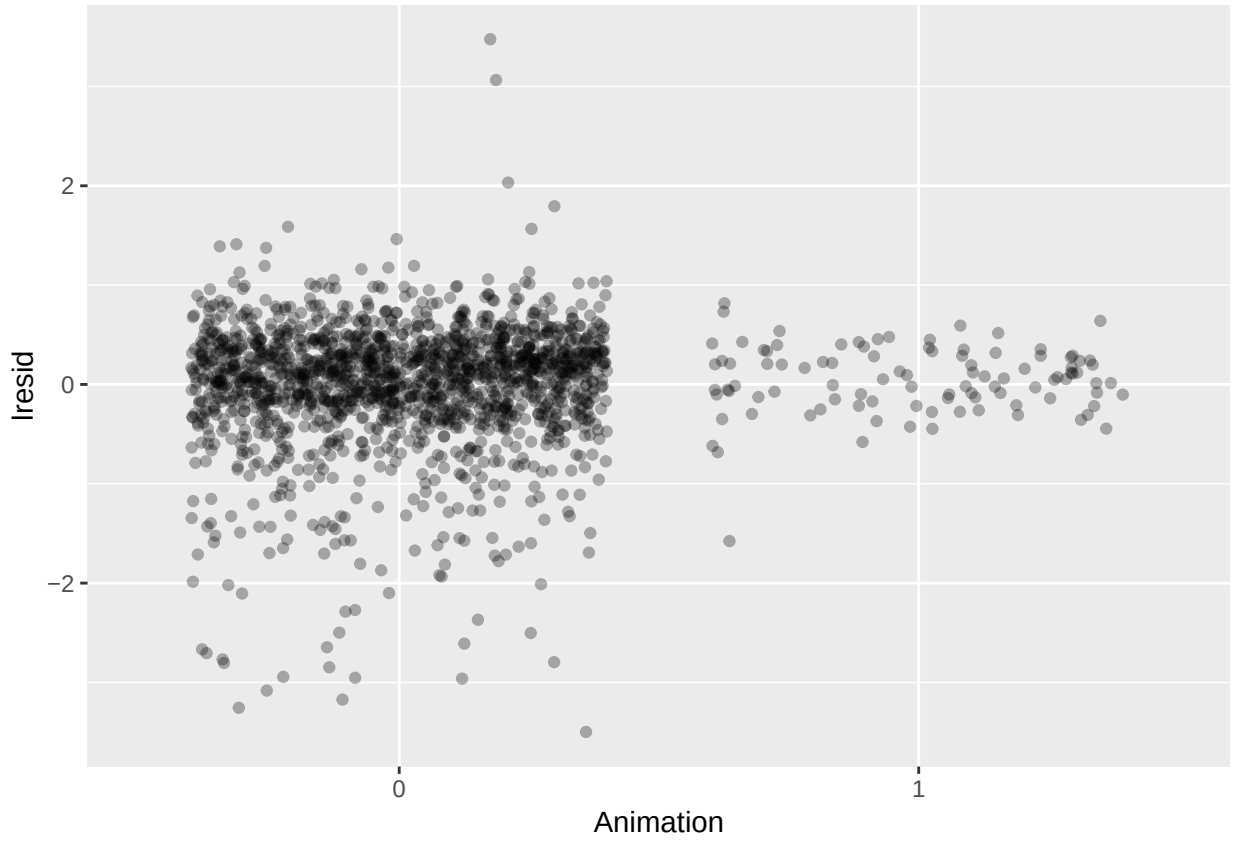

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



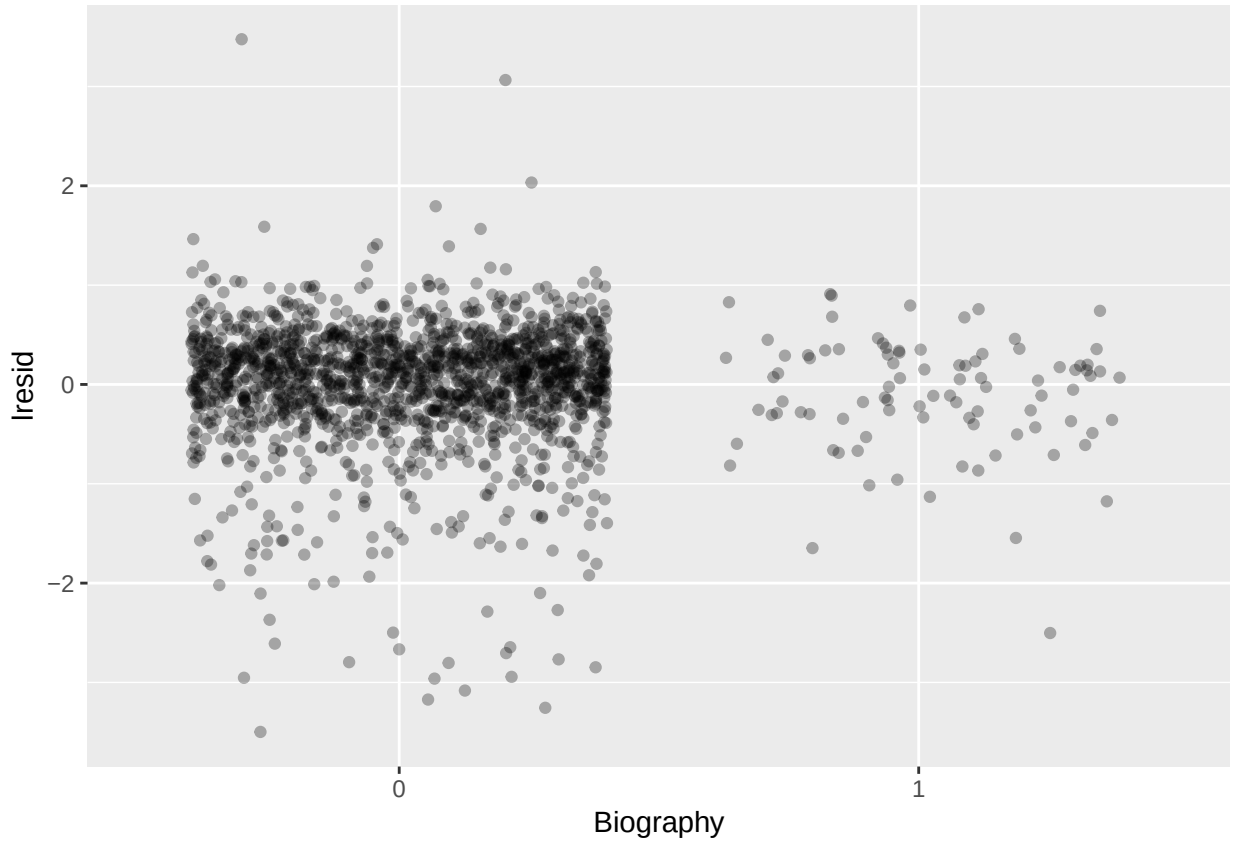

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



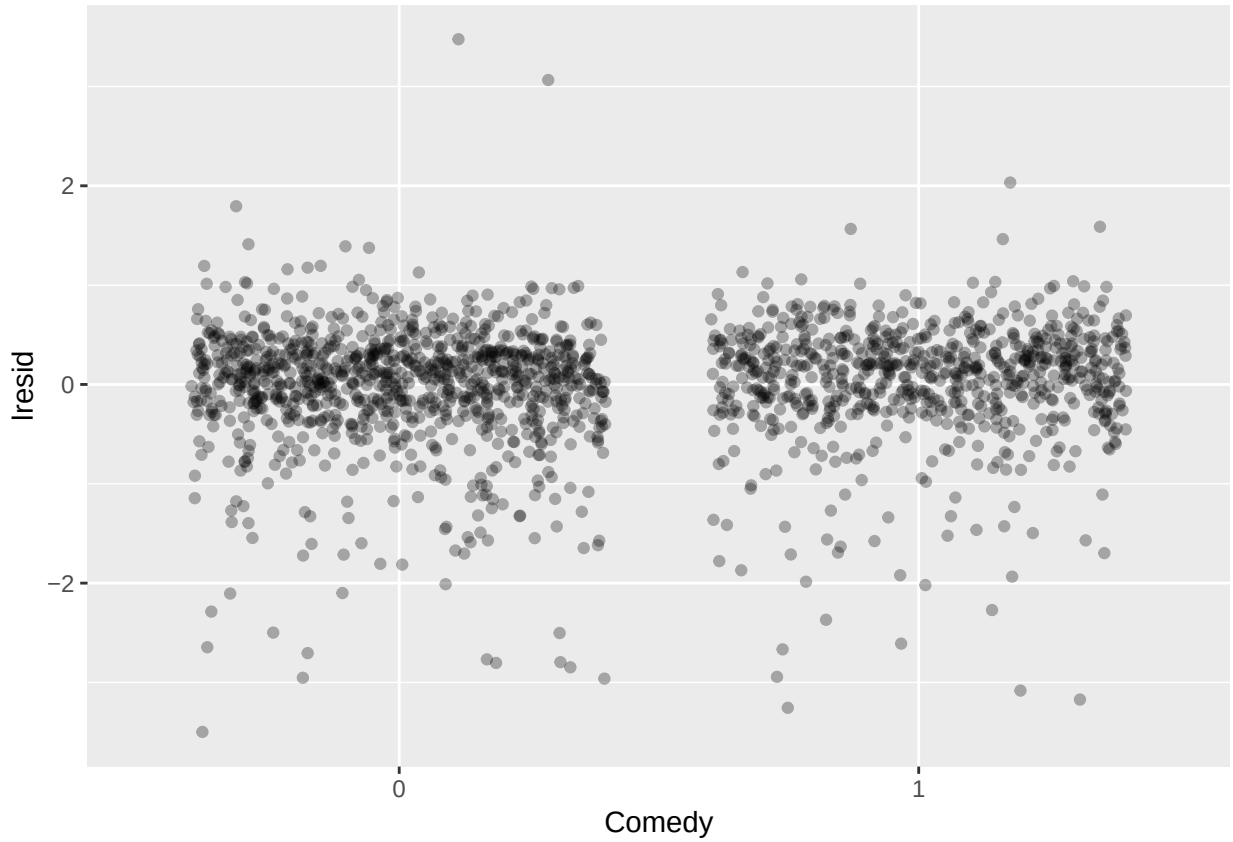

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



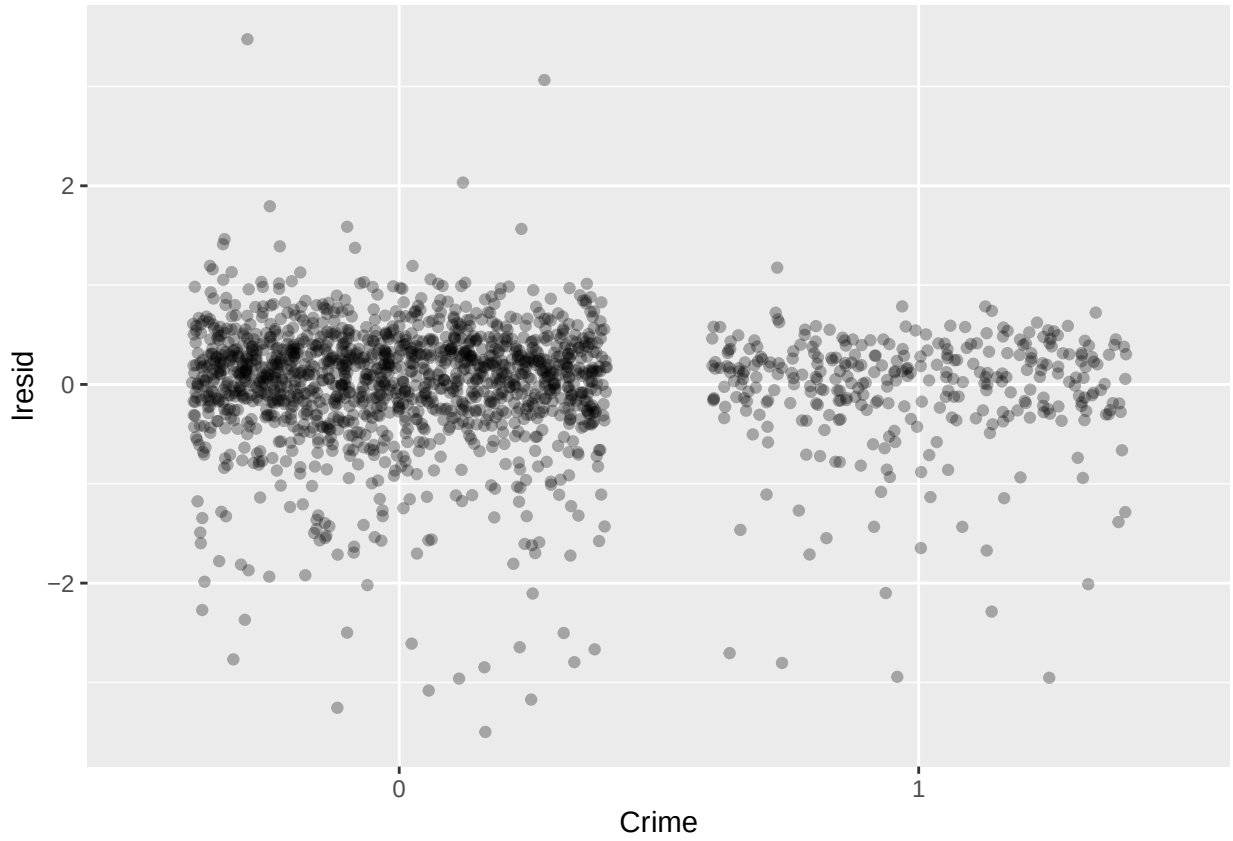

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



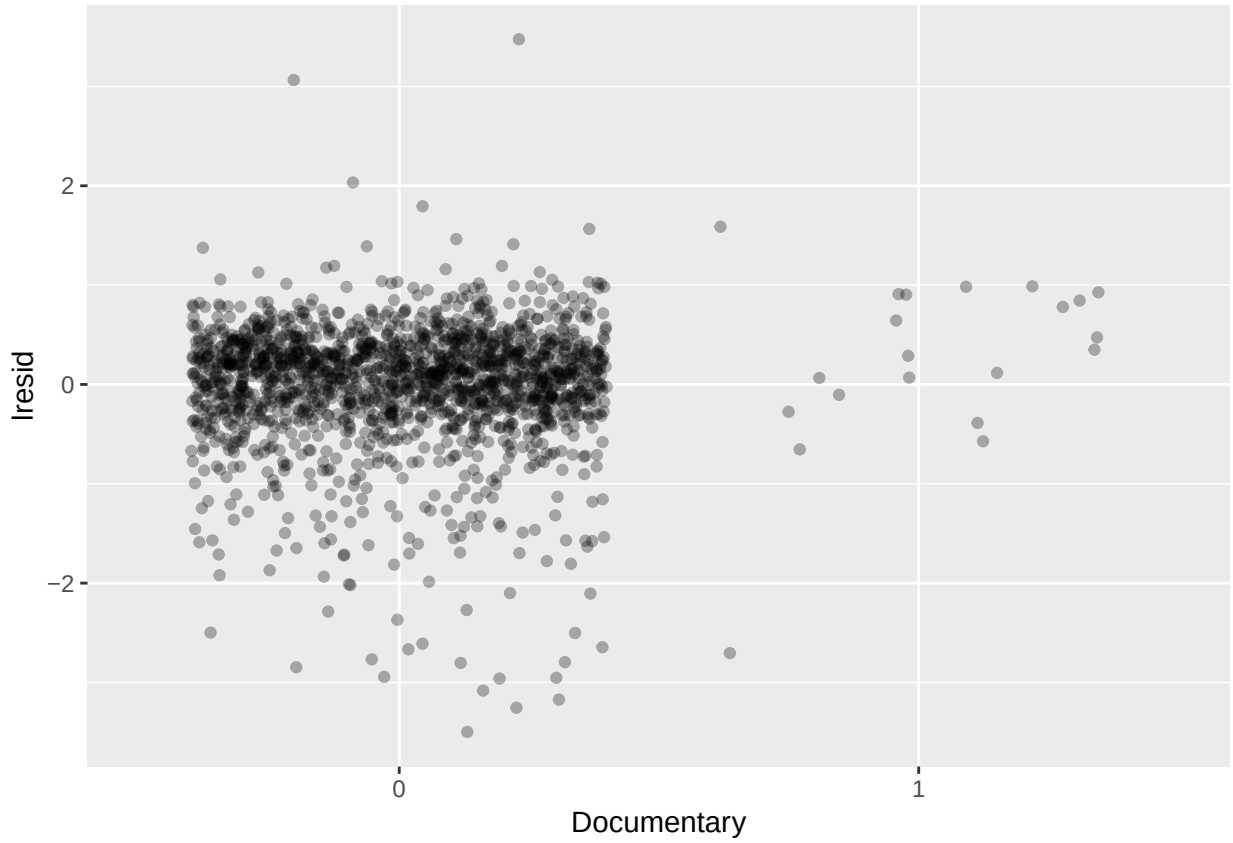

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



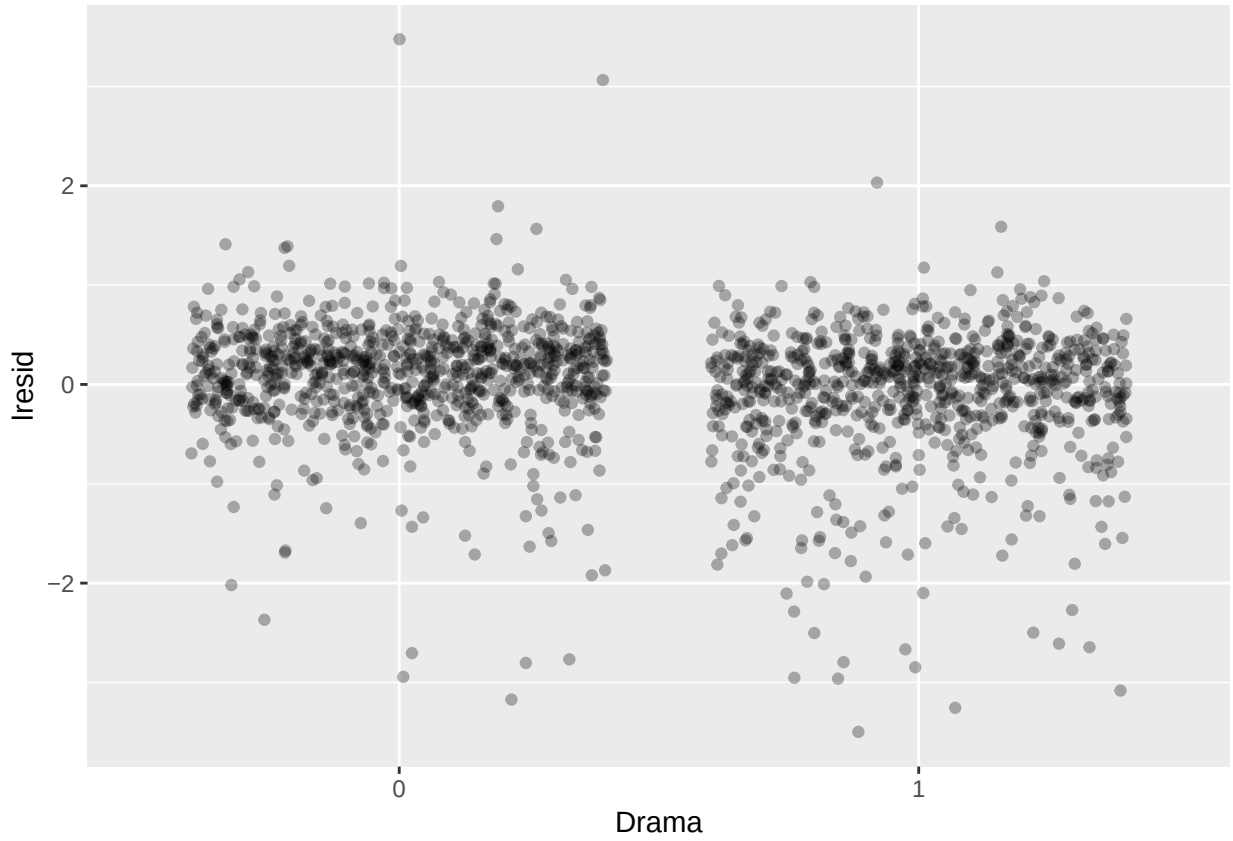

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



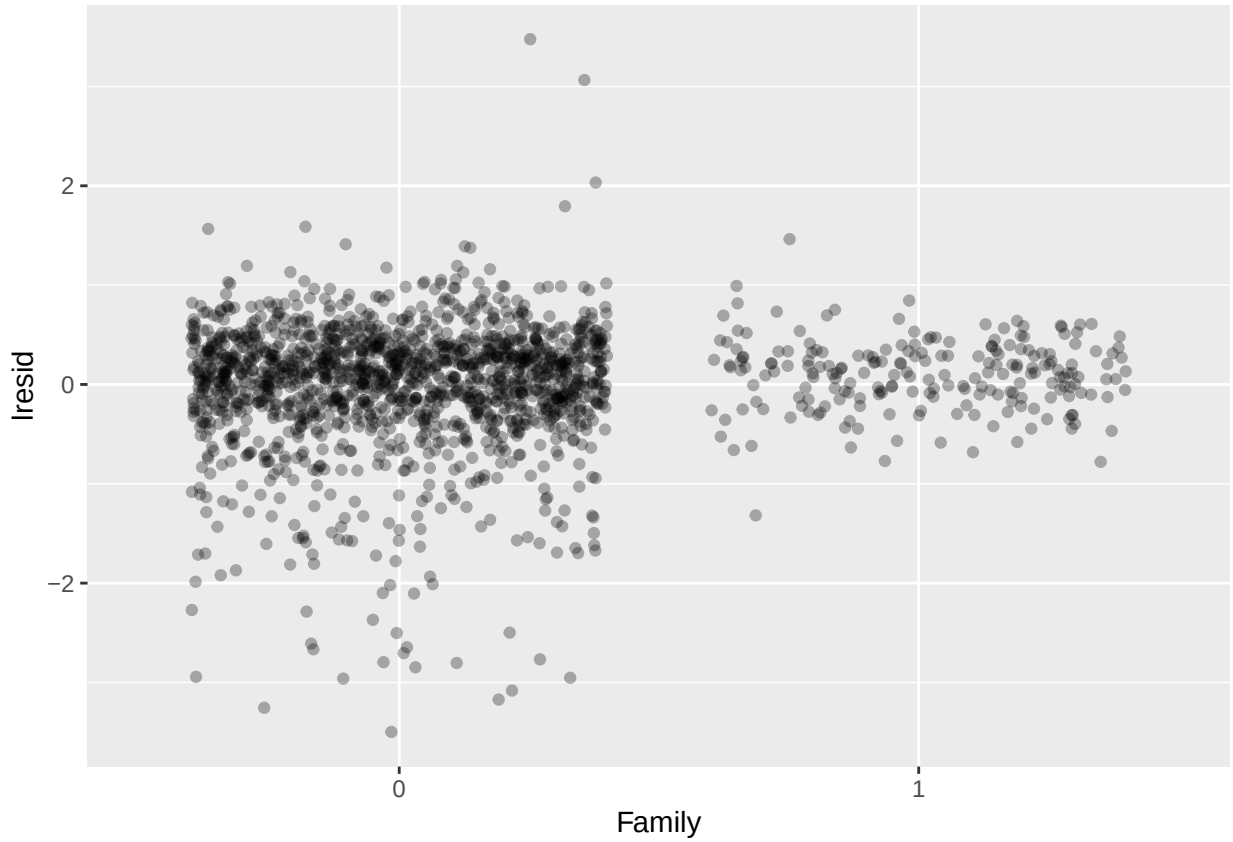

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



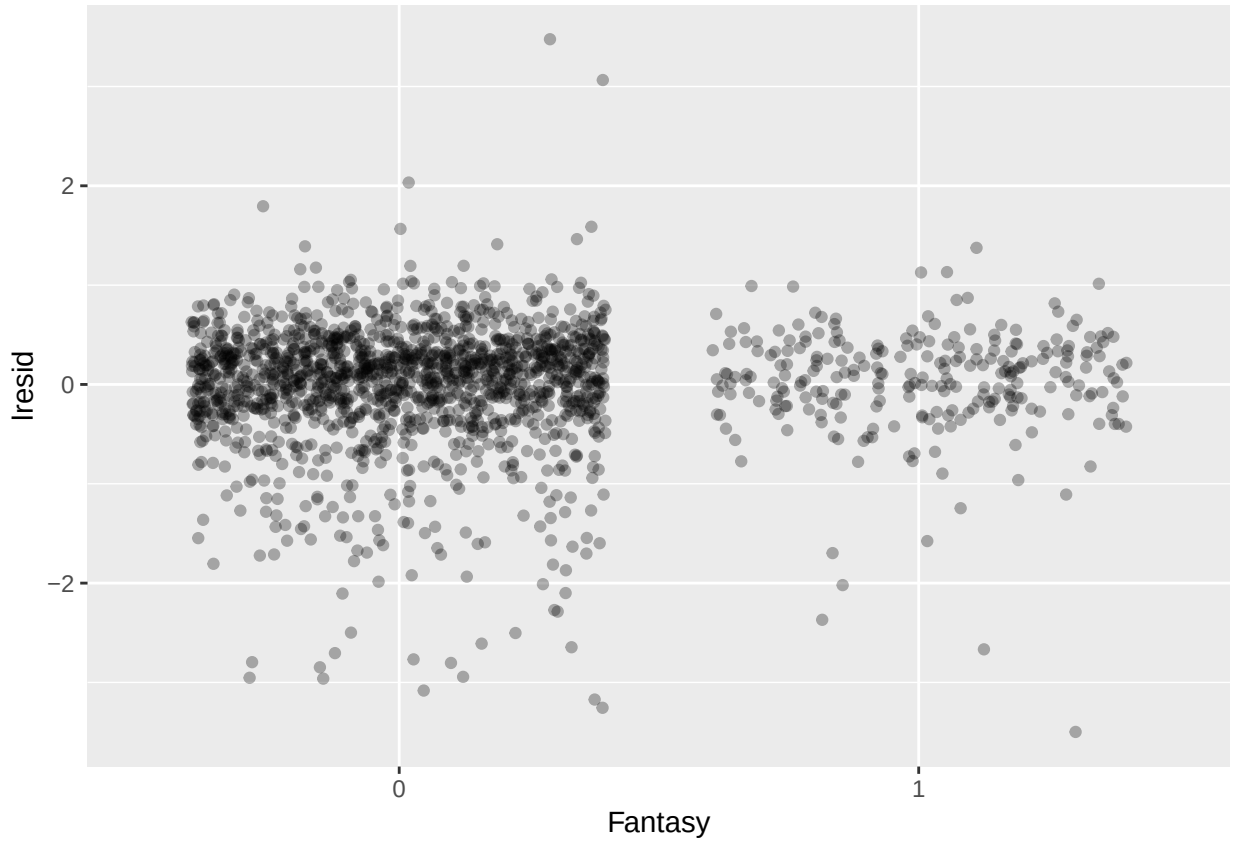

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



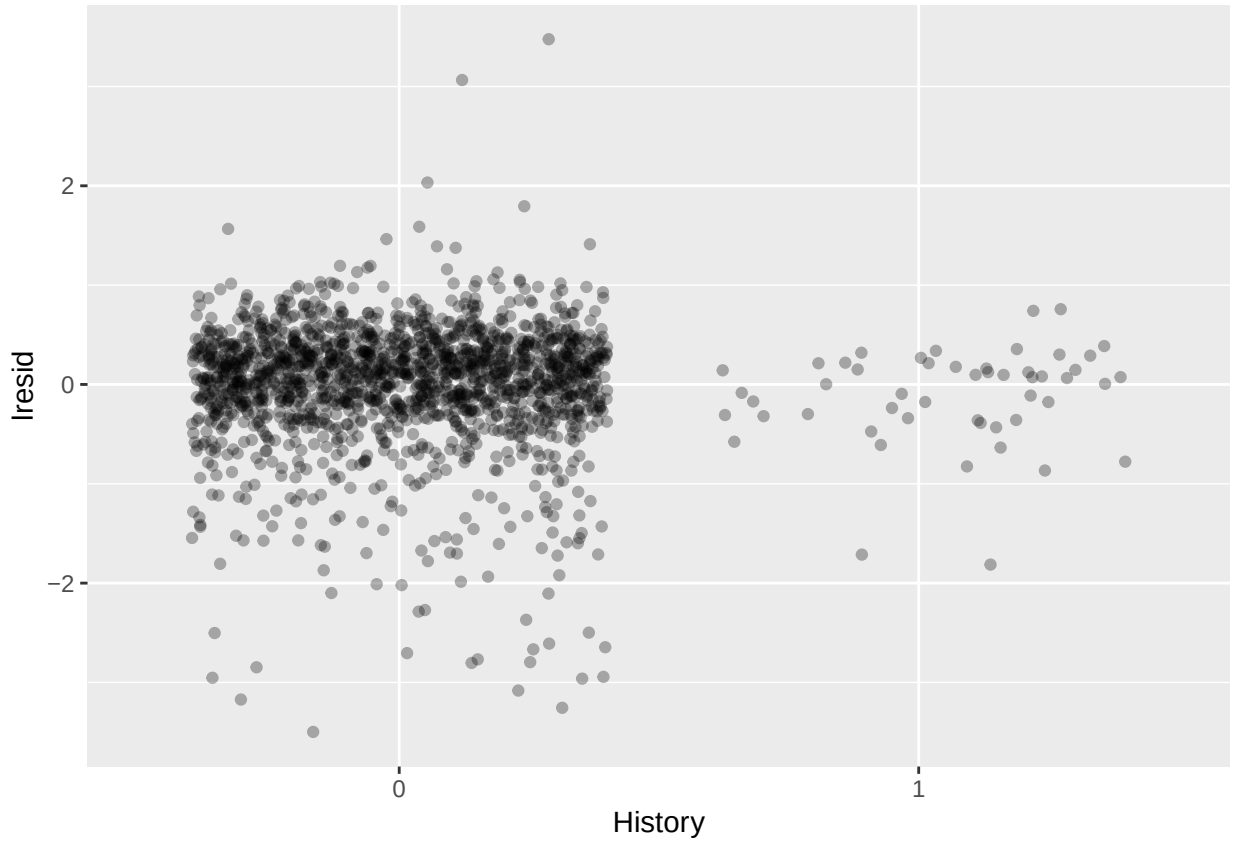

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



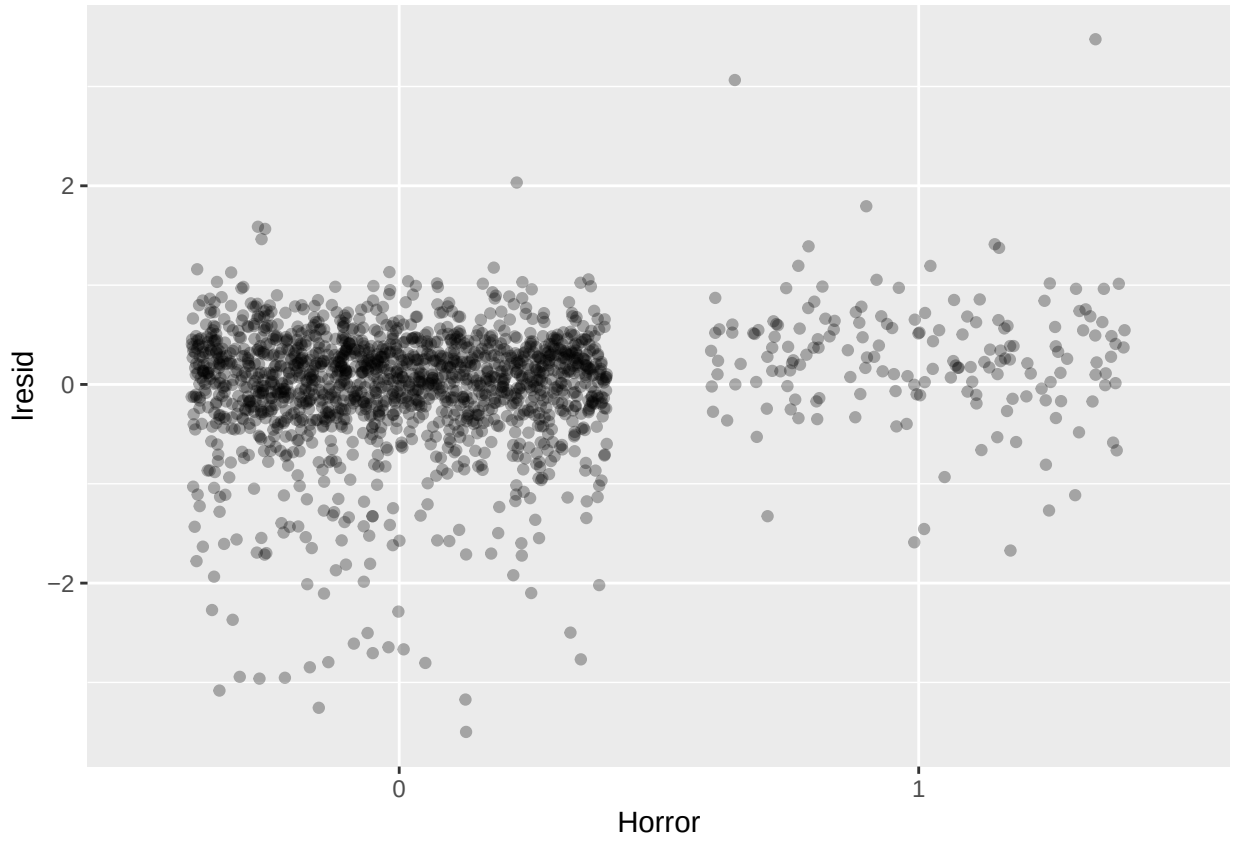

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



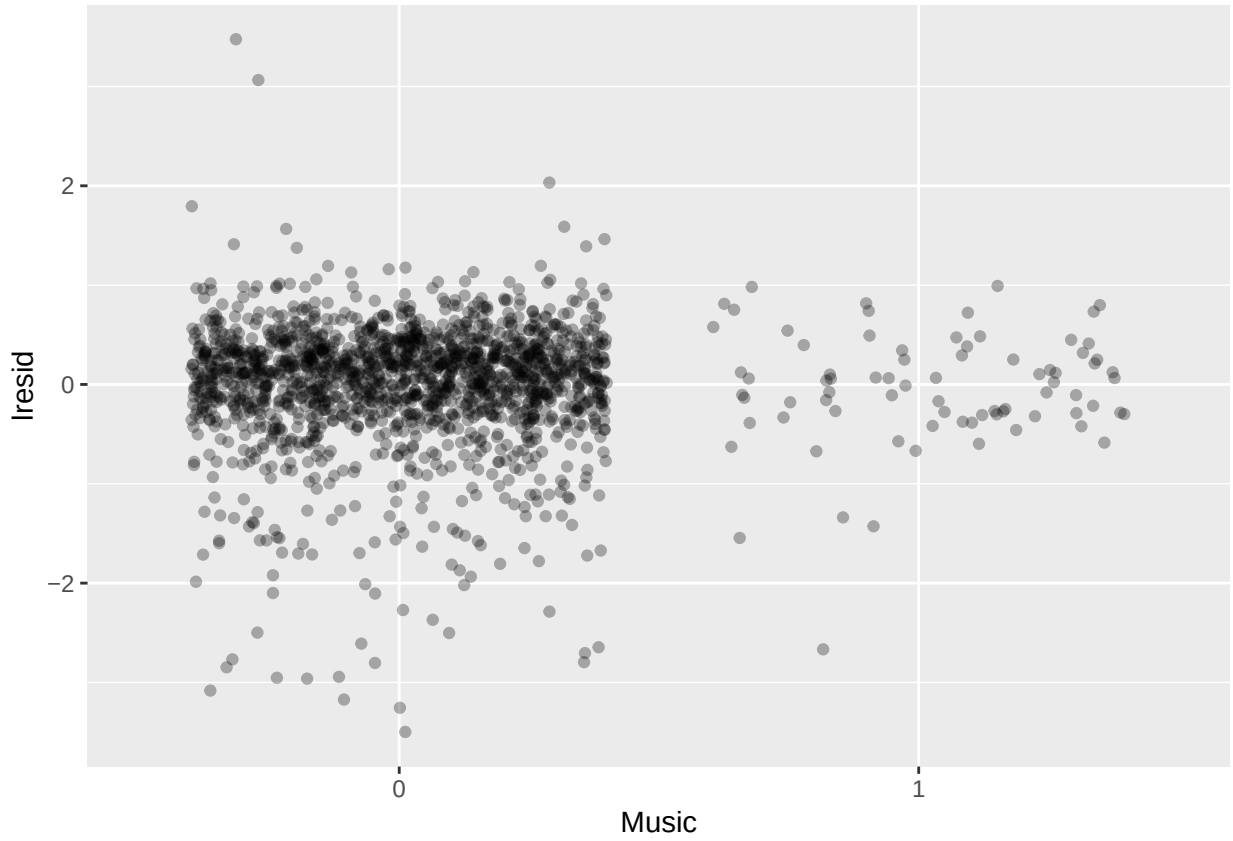

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



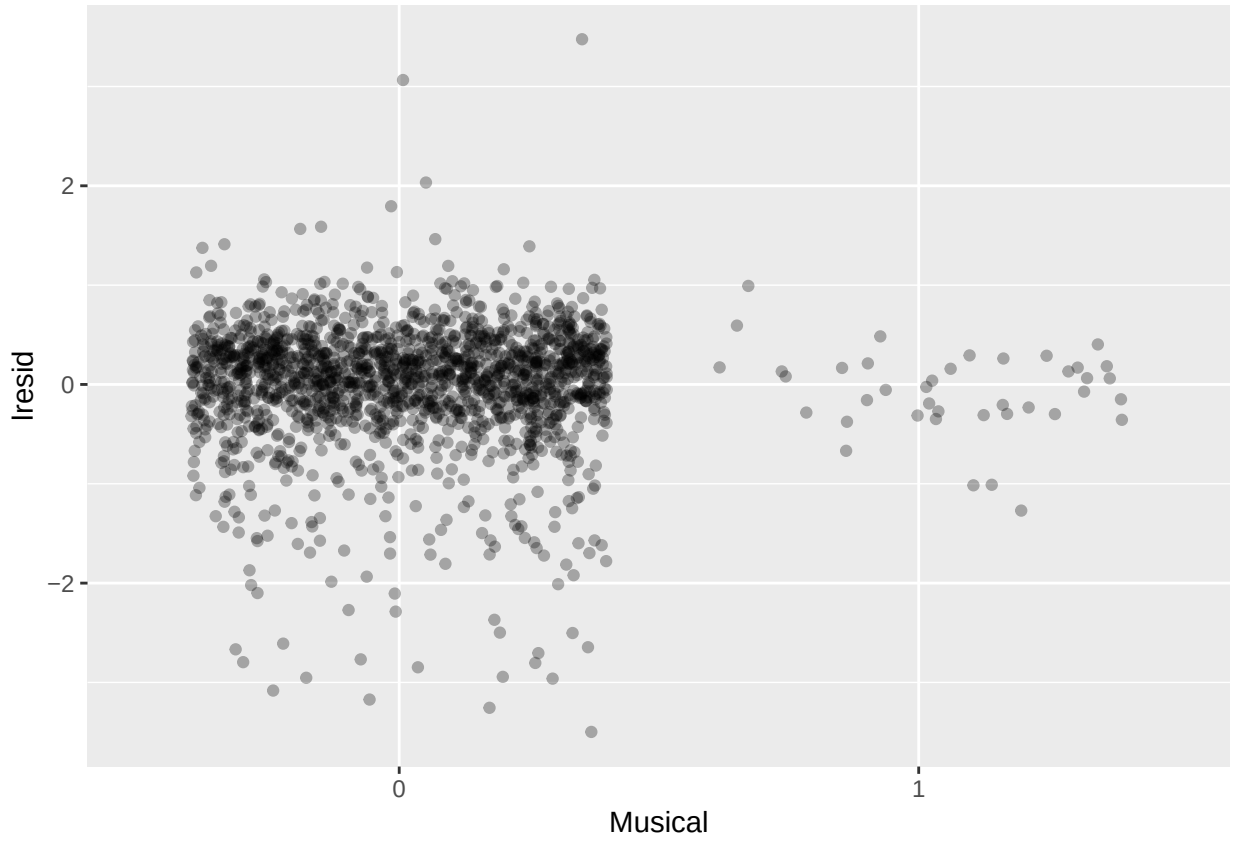

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



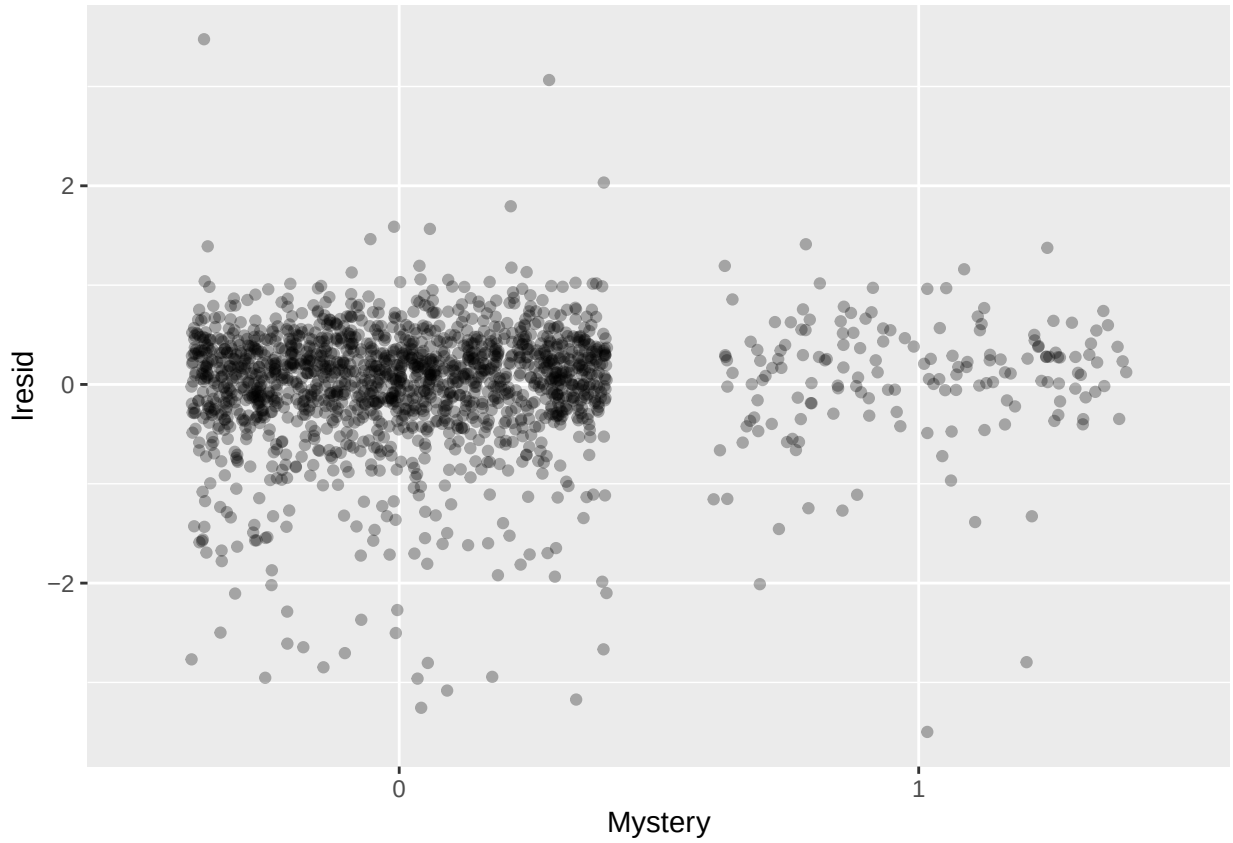

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



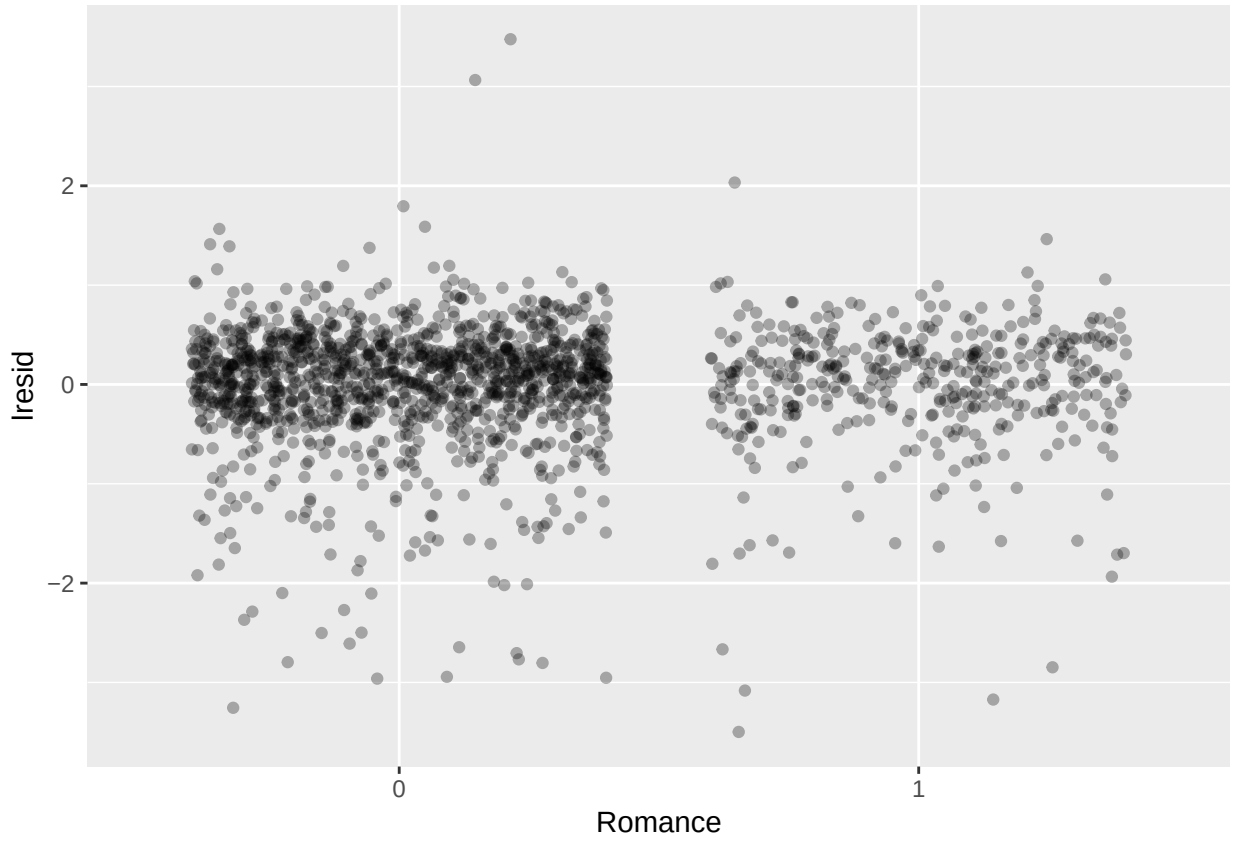

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



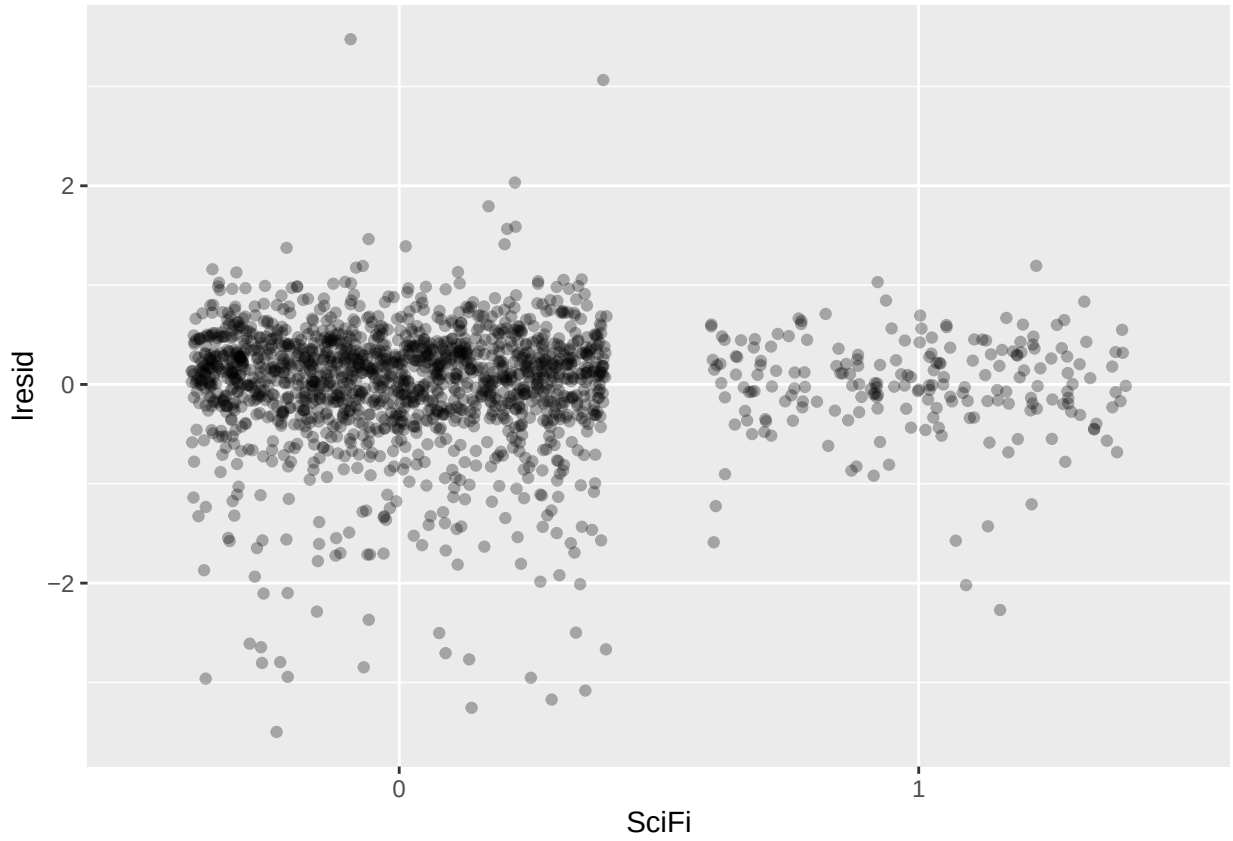

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



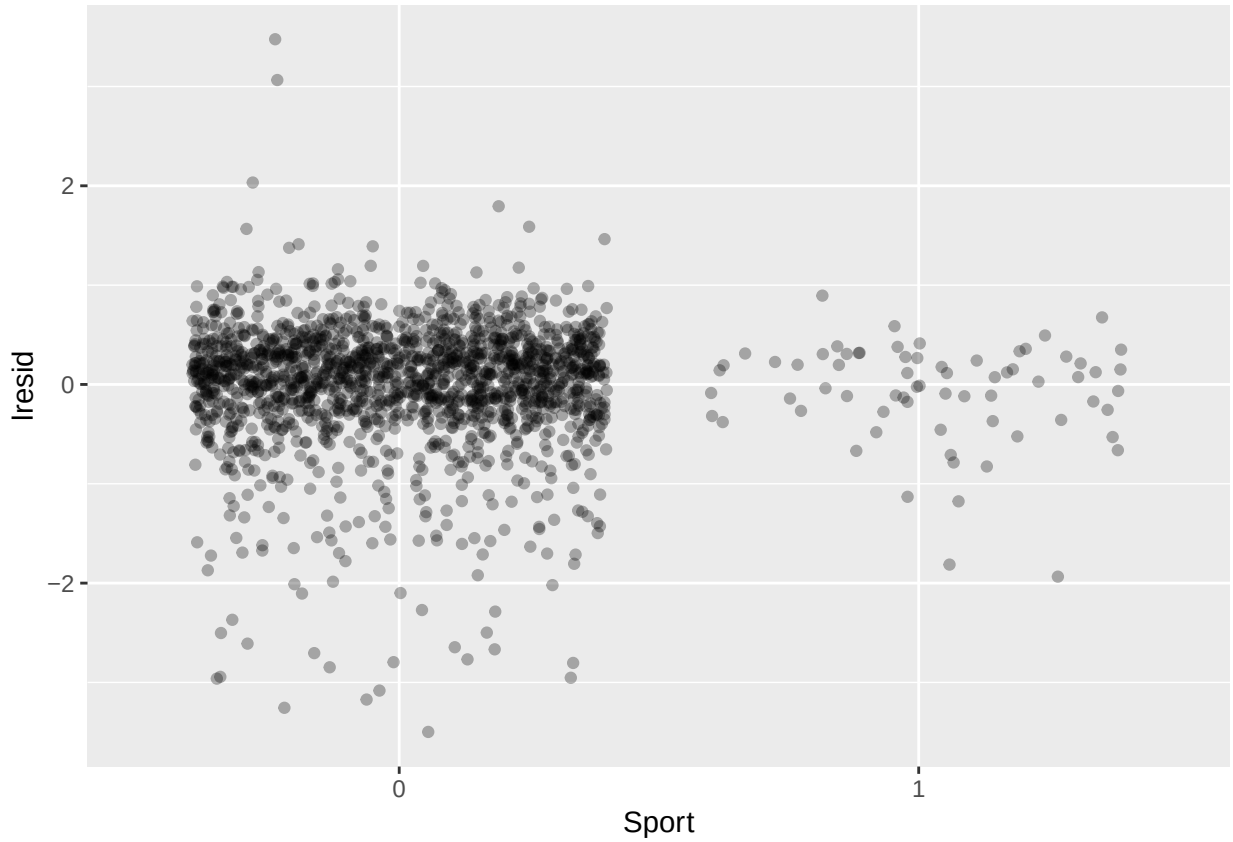

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



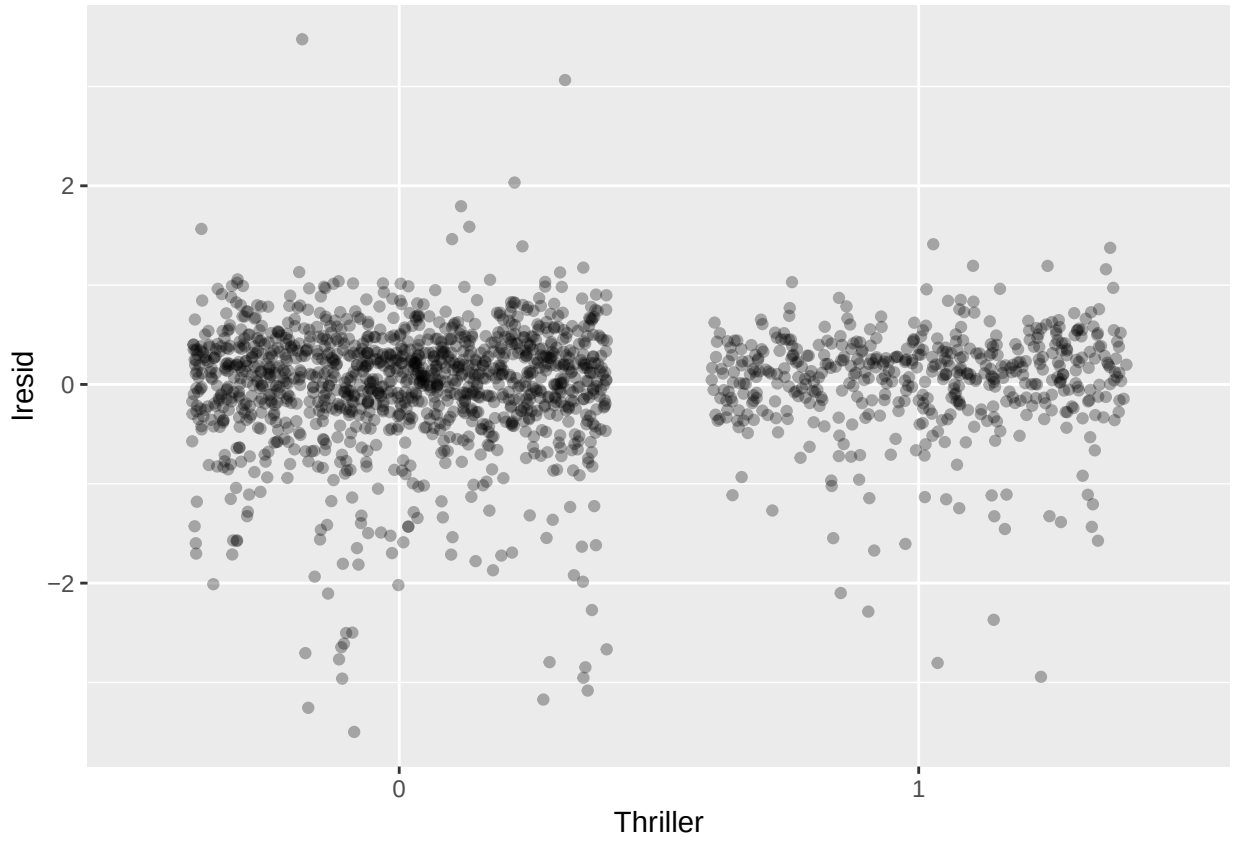

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



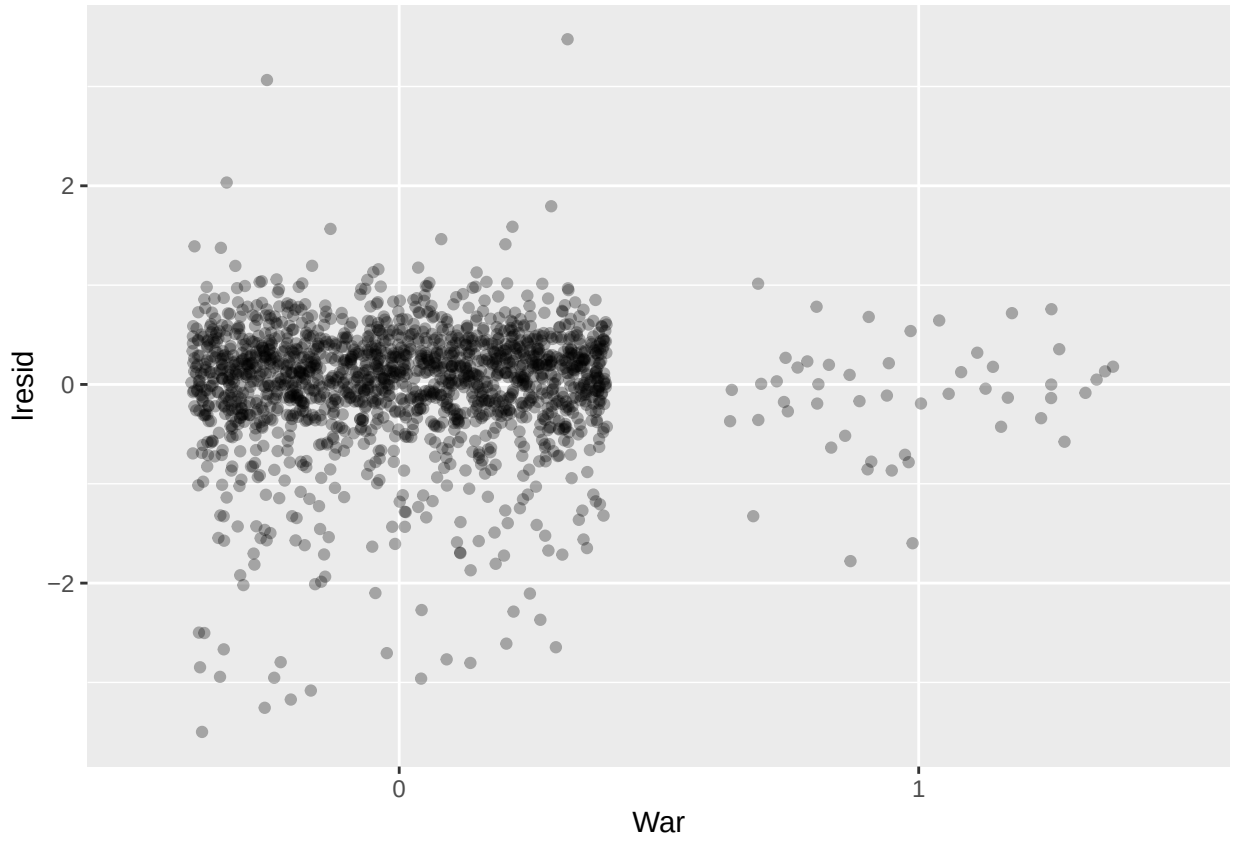

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



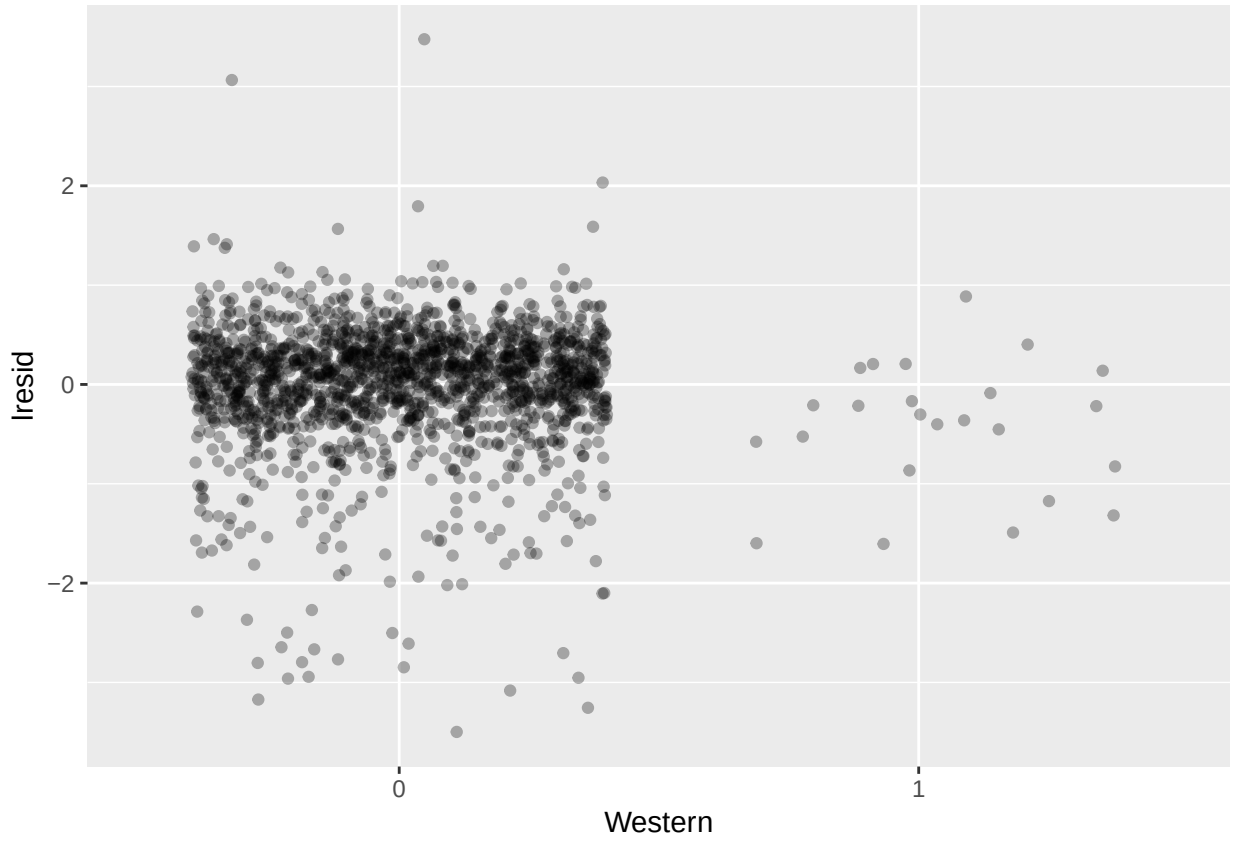

```
## Warning: Removed 132 rows containing missing values (geom_point).
```



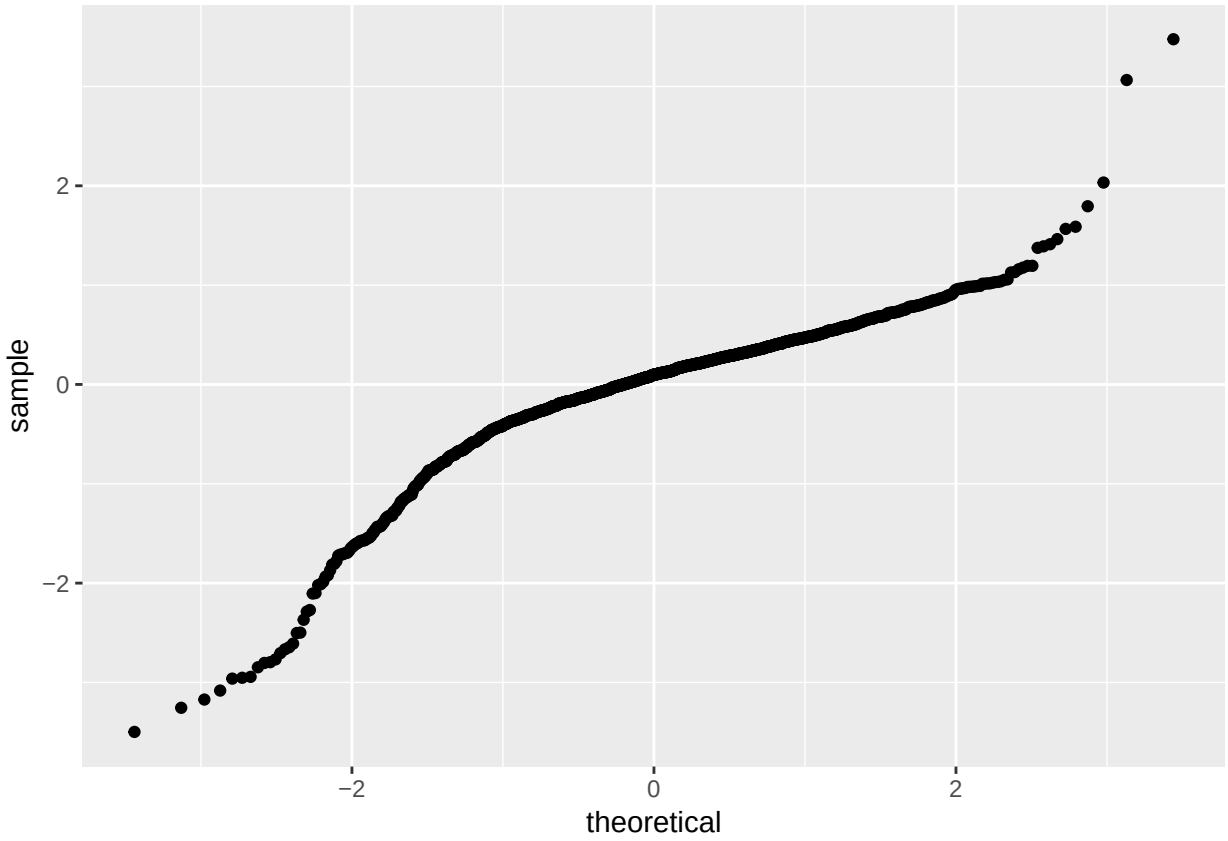

```
## Warning: Removed 132 rows containing missing values (geom_point).
```




```
## Warning: Removed 132 rows containing missing values (geom_point).
```




```
## Warning: Removed 132 rows containing non-finite values (stat_qq).
```




```
# slightly non-random relationships with some of the variables that are included in mod_simple_plus.
# but those variables don't improve model fit enough
```

Try treating genre as one variable: they all need to be added at once

Step wise. Genre should not be included.

```
rmse(lm(real_gross_log ~ Action + Adventure + Animation + Biography + Comedy + Crime + Documentary
        + Drama + Family + Fantasy + History + Horror + Music + Musical + Mystery + Romance +
          SciFi + Sport + Thriller + War + Western, data = train), data = valid)
```

```
## [1] 0.8780216
```

```
rmse(lm(real_gross_log ~ real_budget_log, data = train), data = valid) # .649
```

```
## [1] 0.6497192
```

```
rmse(lm(real_gross_log ~ imdb_score_log, data = train), data = valid)
```

```
## [1] 0.9273344
```

```
rmse(lm(real_gross_log ~ year, data = train), data = valid)
```

```
## [1] 0.9297676
```

```
rmse(lm(real_gross_log ~ content_rating, data = train), data = valid)
```

```
## [1] 0.8789942
```

```
rmse(lm(real_gross_log ~ cast_total_facebook_likes, data = train), data = valid)
```

```
## [1] 0.96301
```

```
rmse(lm(real_gross_log ~ director_facebook_likes, data = train), data = valid)
```

```
## [1] 0.9360115
```

```
rmse(lm(real_gross_log ~ total_oscars_actor, data = train), data = valid)
```

```
## [1] 0.9406675
```

```
rmse(lm(real_gross_log ~ total_oscars_director, data = train), data = valid)
```

```
## [1] 0.9352206
```

```
rmse(lm(real_gross_log ~ real_budget_log + Action + Adventure + Animation + Biography + Comedy + Crime +
        + Drama + Family + Fantasy + History + Horror + Music + Musical + Mystery + Romance +
          SciFi + Sport + Thriller + War + Western, data = train), data = valid)
```

```
## [1] 0.6605224
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log, data = train), data = valid) # .638
```

```
## [1] 0.6382915
```

```
rmse(lm(real_gross_log ~ real_budget_log + year, data = train), data = valid)
```

```
## [1] 0.6395231
```

```
rmse(lm(real_gross_log ~ real_budget_log + content_rating, data = train), data = valid)
```

```
## [1] 0.6445119
```

```
rmse(lm(real_gross_log ~ real_budget_log + cast_total_facebook_likes, data = train), data = valid)
```

```
## [1] 0.6516661
```

```
rmse(lm(real_gross_log ~ real_budget_log + director_facebook_likes, data = train), data = valid)
```

```
## [1] 0.6501299
```

```
rmse(lm(real_gross_log ~ real_budget_log + total_oscars_actor, data = train), data = valid)
```

```
## [1] 0.6510423
```

```
rmse(lm(real_gross_log ~ real_budget_log + total_oscars_director, data = train), data = valid)
```

```
## [1] 0.6494547
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log +  Action + Adventure + Animation + Biography
          SciFi + Sport + Thriller + War + Western, data = train), data = valid)
```

```
## [1] 0.6371965
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year, data = train), data = valid) # .628
```

```
## [1] 0.6281035
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + content_rating, data = train), data = valid
```

```
## [1] 0.631455
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + cast_total_facebook_likes, data = train), da
```

```
## [1] 0.6390821
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + director_facebook_likes, data = train), data
```

```
## [1] 0.6389138
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + total_oscars_actor, data = train), data = va
```

```
## [1] 0.6385564
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + total_oscars_director, data = train), data =
```

```
## [1] 0.6383809
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year +  Action + Adventure + Animation + Bio
          SciFi + Sport + Thriller + War + Western, data = train), data = valid)
```

```
## [1] 0.62748
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating, data = train), data =
```

```
## [1] 0.623201
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + cast_total_facebook_likes, data = tra
```

```
## [1] 0.629357
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + director_facebook_likes, data = train
```

```
## [1] 0.6284716
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + total_oscars_actor, data = train), da
```

```
## [1] 0.6286905
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + total_oscars_director, data = train)
```

```
## [1] 0.6281841
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating + Action + Adventure +
```

```
## [1] 0.6249983
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating + cast_total_facebook_
```

```
## [1] 0.6234161
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating + director_facebook_li
```

```
## [1] 0.6235301
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating + total_oscars_actor,
```

```
## [1] 0.6240377
```

```
rmse(lm(real_gross_log ~ real_budget_log + imdb_score_log + year + content_rating + total_oscars_directo
```

```
## [1] 0.623331
```

```
# RMSE has leveled off. not even including genre yet
```