Modeling_Katrina_Cleaned

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_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score', 'real_gross'), funs(log = log10(.)))
valid <- valid %>%
  mutate_at(vars('real_budget', 'director_facebook_likes', 'cast_total_facebook_likes',
         'imdb_score', 'real_gross'), 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(aes(sample = lresid)) +
    geom_qq() +
    geom_qq_line() +
    labs(title = 'Residual QQPlot',
       x = 'Theoretical Quantiles', y = 'Sample Quantiles')
}
```

Fit Model with Genre Variables vs Real Revenue

Start with genre because it looks like it has a strong relationship with revenue (different genres have different average revenues) and because this was our original hypothesis.
However, based on our EDA, the average revenue of some genres vs not genre is almost the same. Thus we want to start by determining which genres should be included by fitting a model of only genre variables.

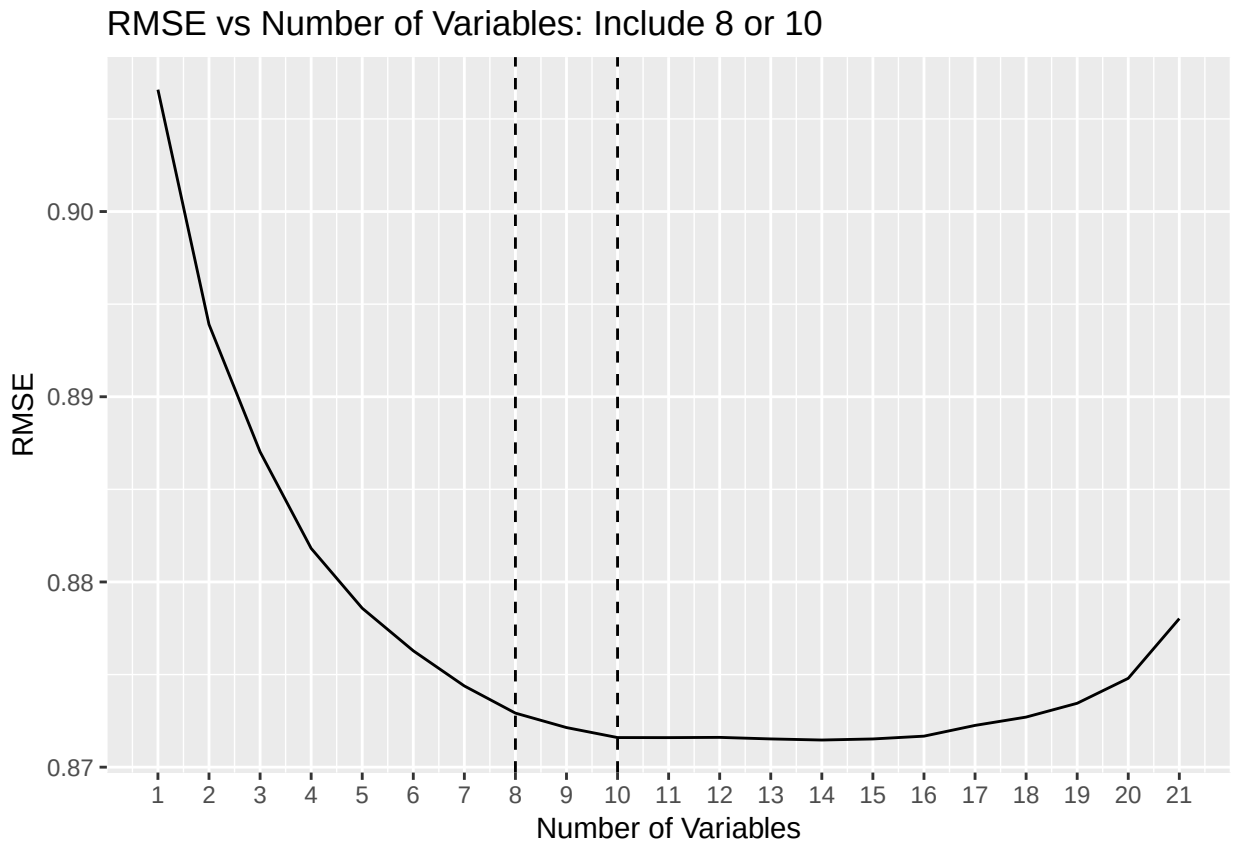
Note this is not as simple as a categorical varible because one movie can have multiple genres (adventure, action, comedy).

Step Wise Selection

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.

```
# version of train set with just genre columns to loop through
all_genre_vars <- c('Action', 'Adventure', 'Animation', 'Biography', 'Comedy', 'Crime', 'Documentary',
                    'Drama', '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)
```

```
# graph RMSE at each step
fit_rmse <- tibble(nvar = 1:length(rmse_lst),
                   rmse = rmse_lst)
ggplot(fit_rmse, aes(x = nvar, y = rmse)) + geom_line() +
  scale_x_continuous(breaks = seq(1, length(rmse_lst), by = 1)) +
  geom_vline(xintercept = 8, linetype = 'dashed') +
  geom_vline(xintercept = 10, linetype = 'dashed') +
  labs(x = 'Number of Variables', y = 'RMSE',
       title = 'RMSE vs Number of Variables: Include 8 or 10')
```




```
# after var 8, decreases too small or increase (debatably 10?)

# model based off of step wise
mod_genre <- lm(real_gross_log ~ Adventure + Action + Family + Mystery + Romance + Drama + History + Do
mod_genre10 <- lm(real_gross_log ~ Adventure + Action + Family + Mystery + Romance + Drama + History + 


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
```

```
# two additional variables Musical and War are insignificant and RMSE goes from .873 to .872
summary(mod_genre10)
```

```
##
## Call:
## lm(formula = real_gross_log ~ Adventure + Action + Family + Mystery +
##     Romance + Drama + History + Documentary + Musical + War,
##     data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0873 -0.3245  0.1659  0.5635  1.7803
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.21185    0.04047 178.215  < 2e-16 ***
## Adventure1    0.25575    0.05958   4.292 1.86e-05 ***
## Action1       0.36445    0.05378   6.776 1.66e-11 ***
## Family1       0.46793    0.06792   6.890 7.65e-12 ***
## Mystery1      0.20093    0.07027   2.859  0.00429 **
## Romance1      0.10540    0.04894   2.154  0.03140 *
## Drama1       -0.23707    0.04420  -5.364 9.18e-08 ***
## History1      0.38804    0.13002   2.984  0.00288 **
## Documentary1 -1.11351    0.13064  -8.524  < 2e-16 ***
## Musical1     -0.06322    0.13019  -0.486  0.62730
## War1          0.19858    0.12895   1.540  0.12375
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8581 on 1840 degrees of freedom
## Multiple R-squared:  0.166,  Adjusted R-squared:  0.1615
## F-statistic: 36.63 on 10 and 1840 DF,  p-value: < 2.2e-16
```

```
rmse(mod_genre10, data = valid)
```

```
## [1] 0.8715984
```

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

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 101 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.

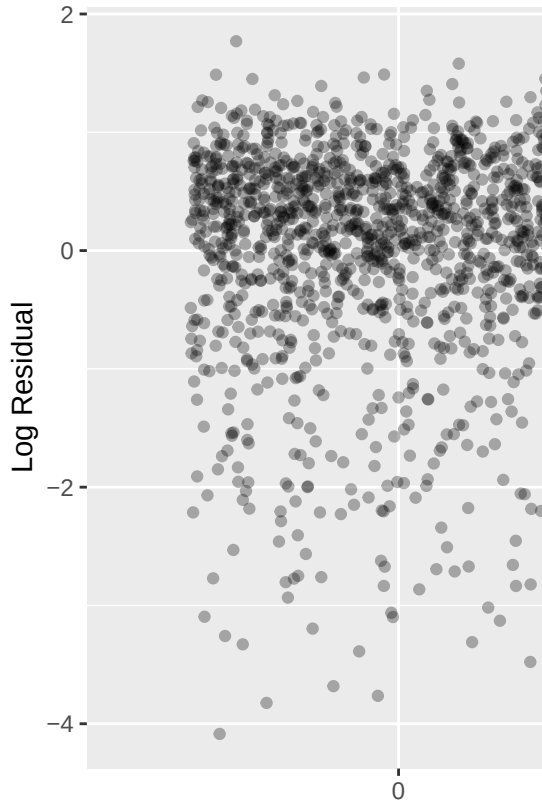
```
train %>% filter(Animation == 1, Family == 1) %>% count() # 93
train %>% filter(Animation == 1) %>% count() # 101

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

Residuals graph

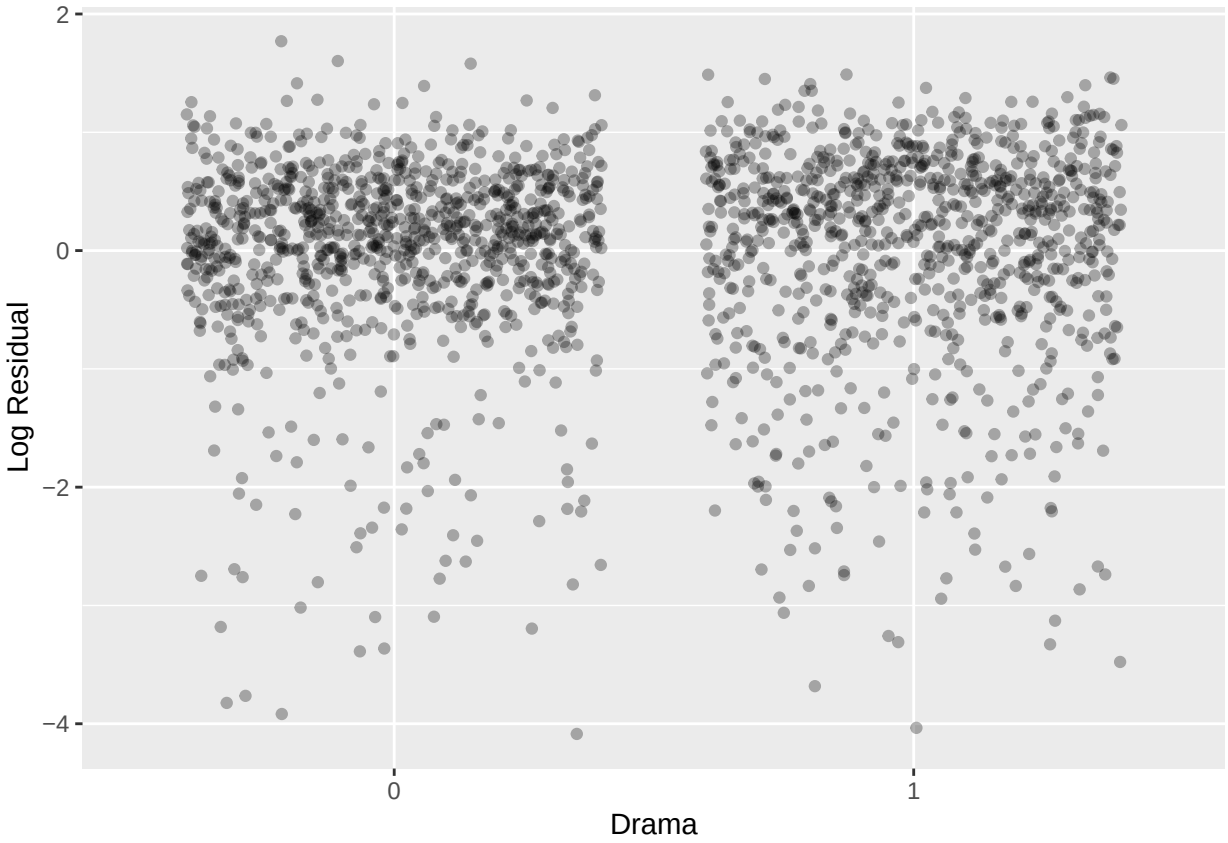
I like this geom jitter view better. Can see individual points. Most movies have some outliers where actual makes less money than predicted based on the included genres. But tricky because movies are multiple genres. So could be because that movie is also another genre that makes less money. Bulk of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

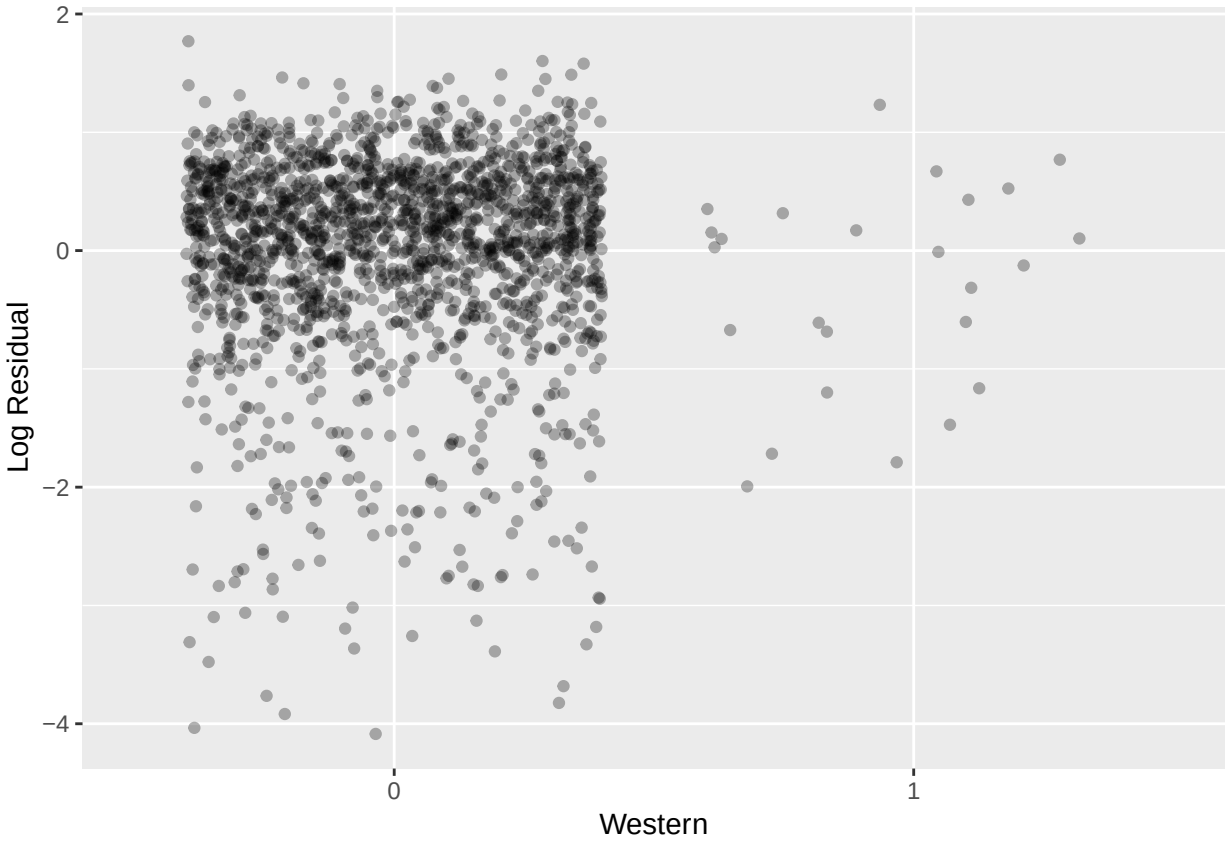
```
# graph residuals against each variable included in the model
# most look random except Adventure
train_resid <- train %>%
  add_residuals(mod_genre, 'lresid')

lapply(genre_xvar, function(var) {
  train_resid %>%
    ggplot() +
    geom_jitter(aes_string(var, y = 'lresid'), alpha = .3)
})
```

```
# 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)
})
```



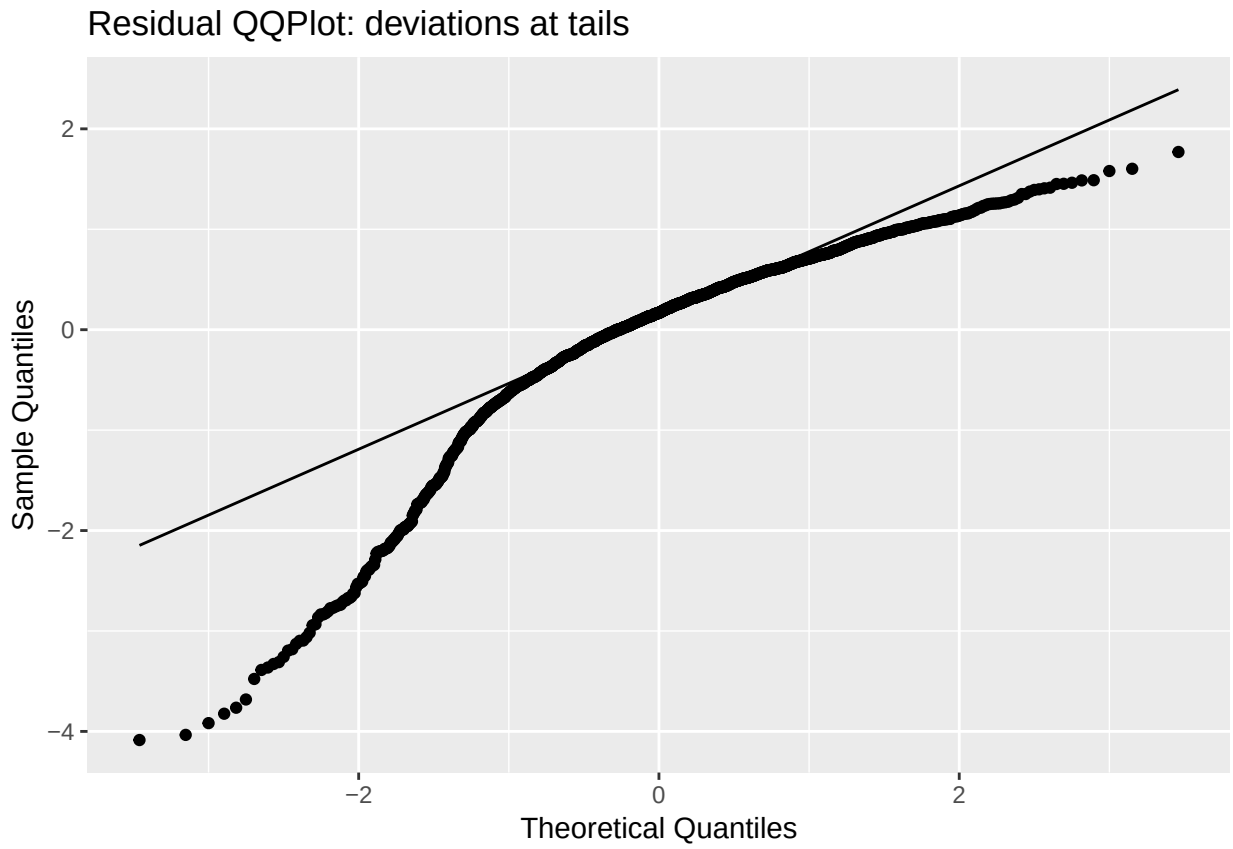
Display a couple of plots for presentation: some diversity. But show all around 0


2
0
−2
−4
Log Residual
0
1
Drama



QQ-Plot Not great.

```
# residuals themselves are NOT normally distributed
# qq plot
train_resid %>% ggplot(aes(sample = lresid)) +
  geom_qq() +
  geom_qq_line() +
  labs(title = 'Residual QQPlot: deviations at tails',
       x = 'Theoretical Quantiles', y = 'Sample Quantiles')
```



Plot Predictions

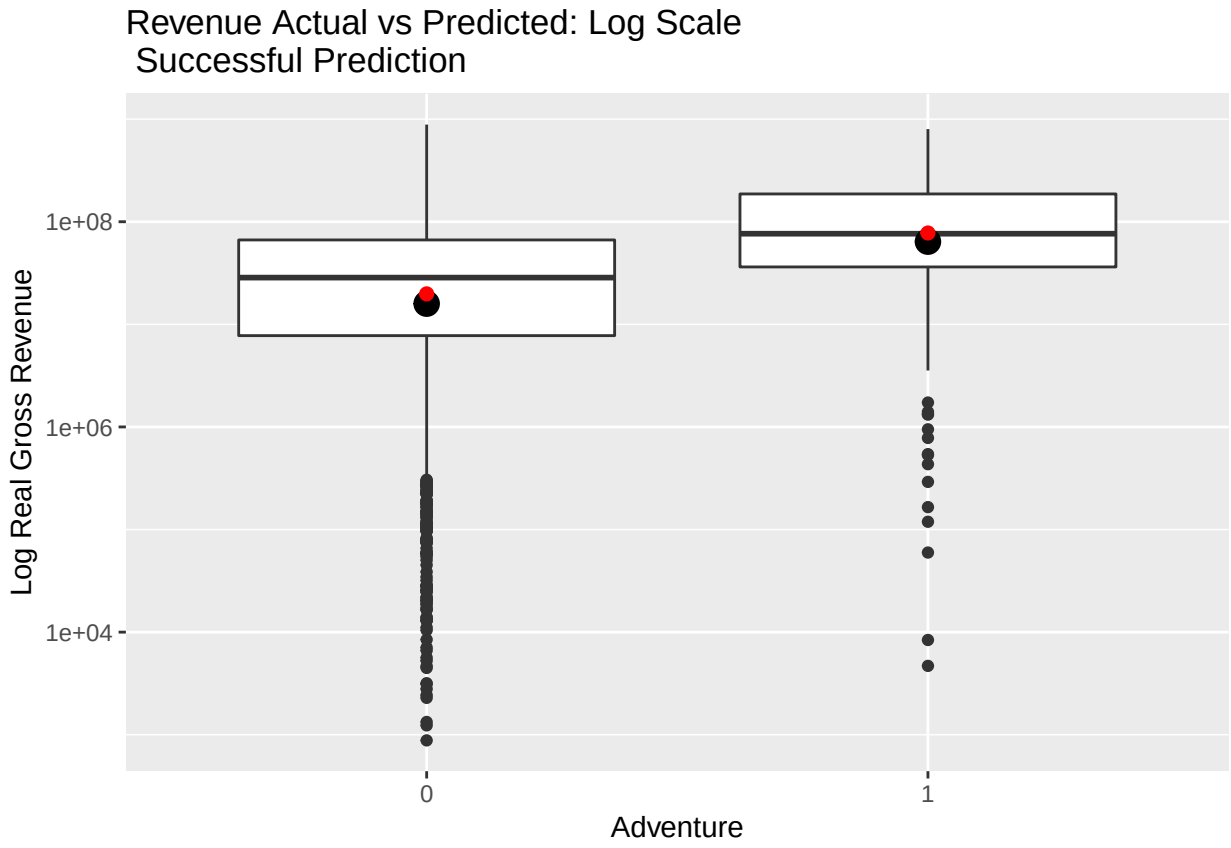
Plot prediction for mean real revenue against each genre included in the model.
AND genres not included in the model: still pretty good with some exceptions. Evidence that these genres are not useful for prediction/are covered by other genres.

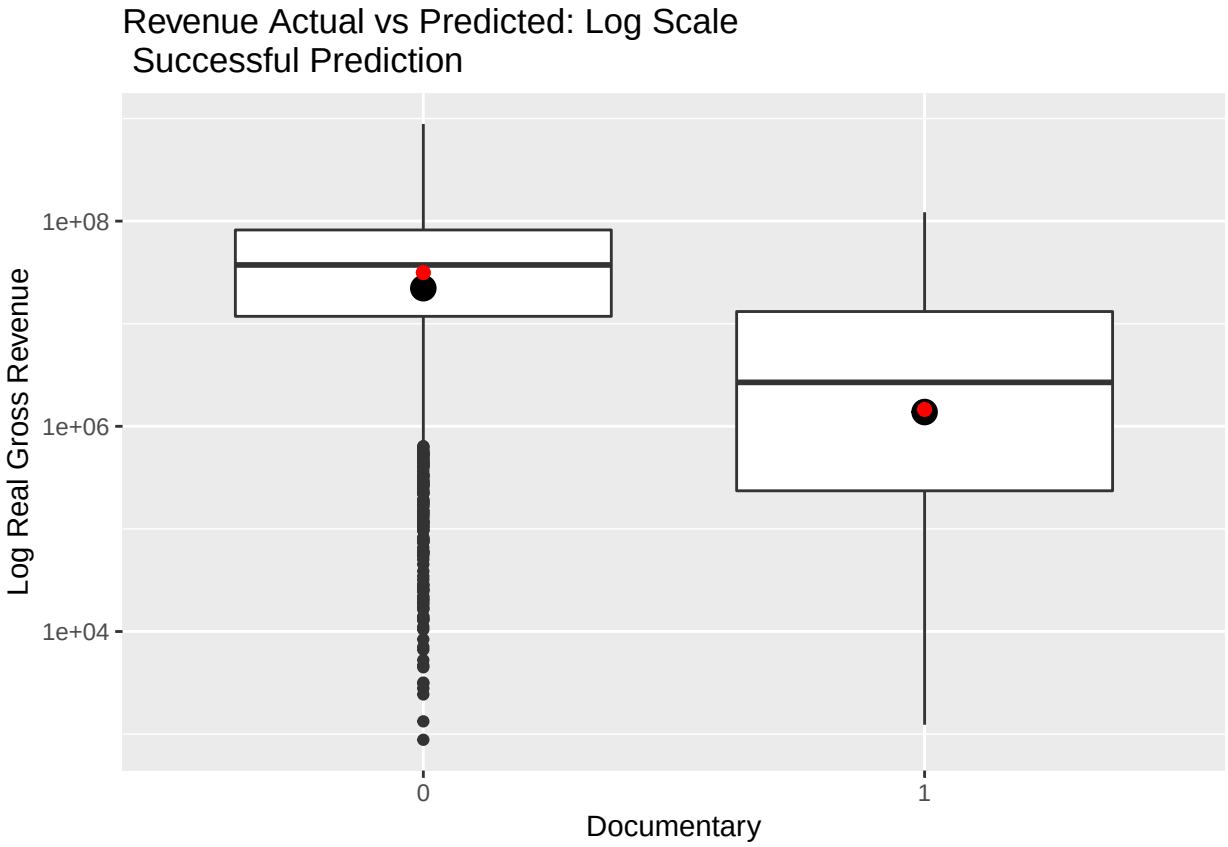
```
train_pred <- train %>%
  add_predictions(mod_genre, 'lpred') %>%
  mutate(pred = 10^lpred)

lapply(genre_xvar, function(var) {
  gr <- train_pred %>%
    ggplot(aes_string(x = var)) +
    geom_boxplot(aes(y = real_gross)) +
    # include mean
    stat_summary(aes(y = real_gross), fun.y = mean, geom = 'point', size = 4) +
    geom_point(data = train_pred %>% group_by(!!rlang::sym(var)) %>% summarize(mean = mean(pred)),
               aes(y = mean), color = 'red', size = 2) +
    scale_y_log10() +
    labs(y = 'Log Real Gross Revenue', title = 'Revenue Actual vs Predicted: Log Scale \n Successful Pr
})
```

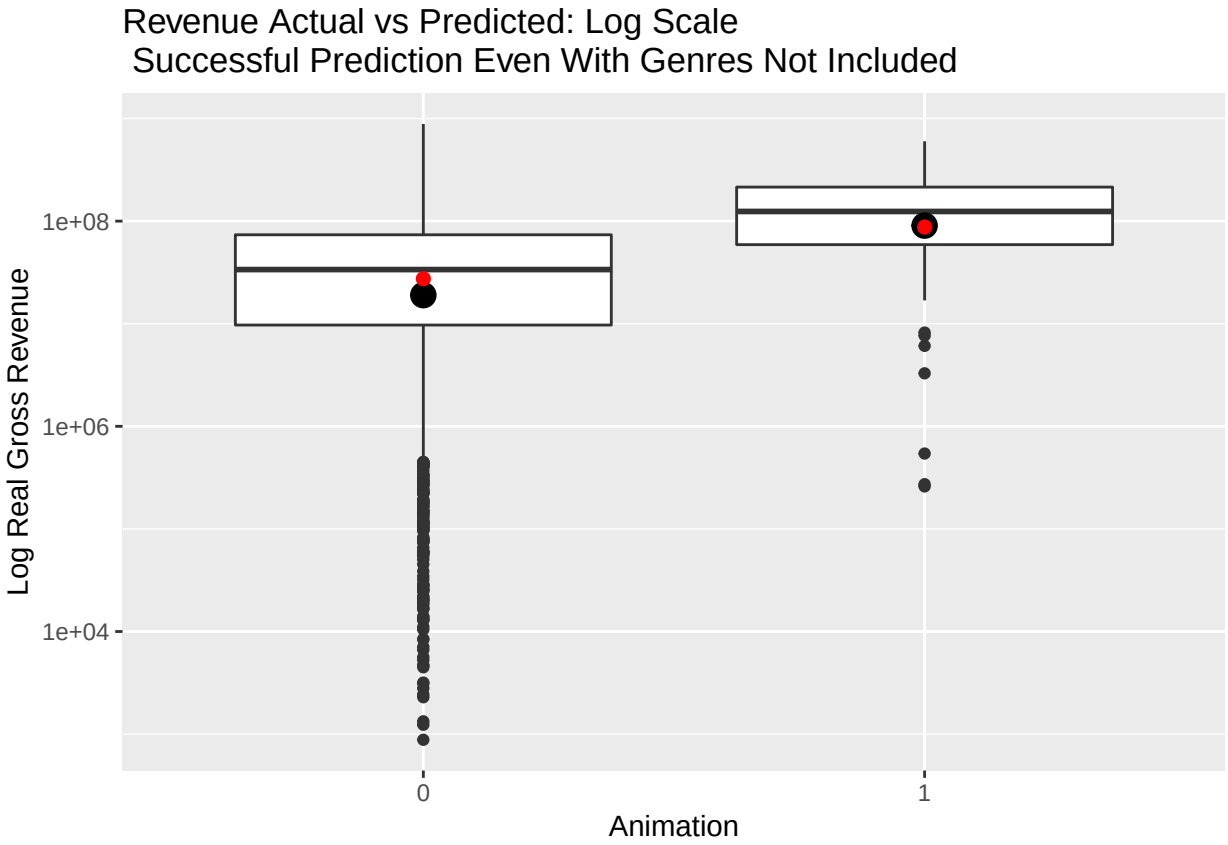
```
# predictions against other genres
lapply(names(train_genre_only %>% select(-genre_xvar)), function(var) {
  train_pred %>%
    ggplot(aes_string(x = var)) +
    geom_boxplot(aes(y = real_gross)) +
```

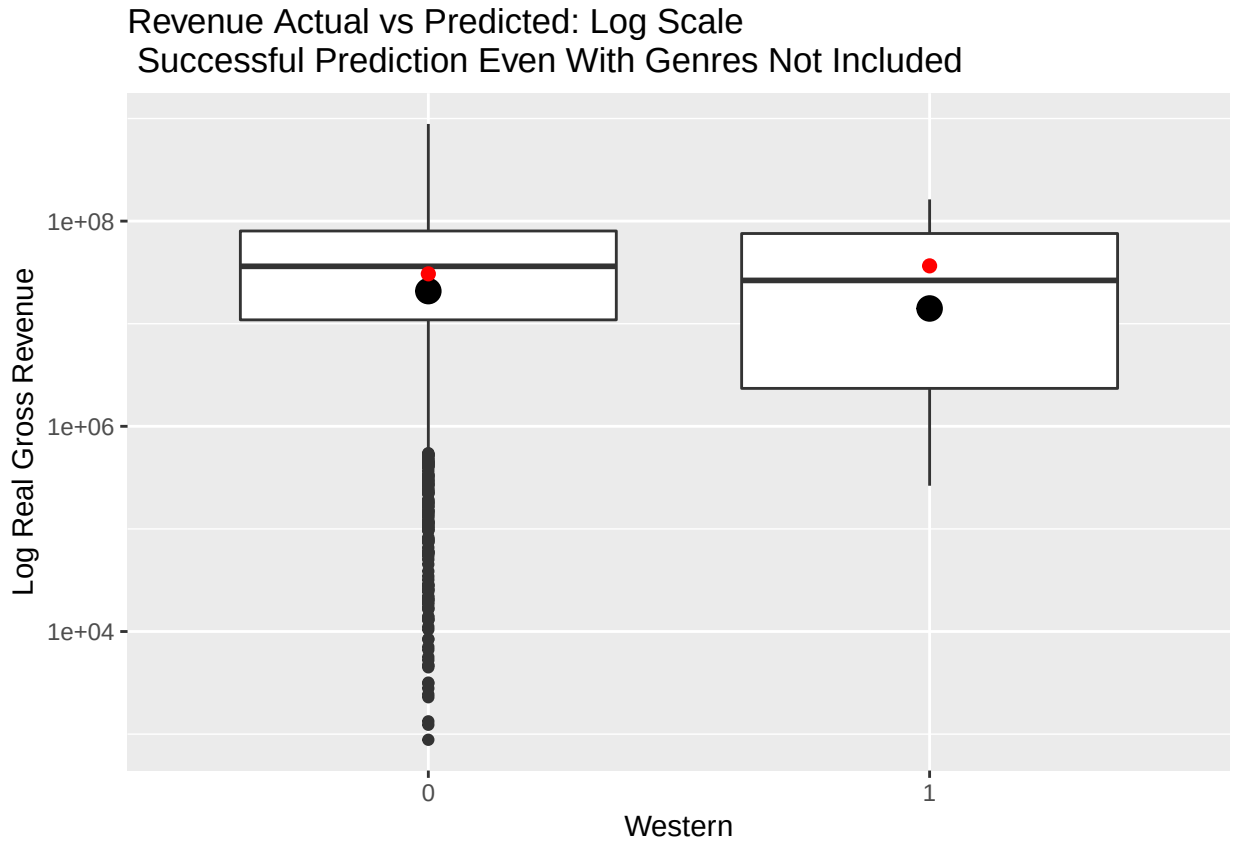
```
    # include mean
    stat_summary(aes(y = real_gross), fun.y = mean, geom = 'point', size = 4) +
    geom_point(data = train_pred %>% group_by(!!rlang::sym(var)) %>% summarize(mean = mean(pred)),
               aes(y = mean), color = 'red', size = 2) +
    scale_y_log10() +
    labs(y = 'Log Real Gross Revenue', title = 'Revenue Actual vs Predicted: Log Scale \n Successful Pr
})
```

Display a couple of plots for presentation. In slides, report: note about black dot is true mean, red dot is predicted




Revenue Actual vs Predicted: Log Scale
Successful Prediction
Log Real Gross Revenue
1e+08
1e+06
1e+04
0
1
Documentary

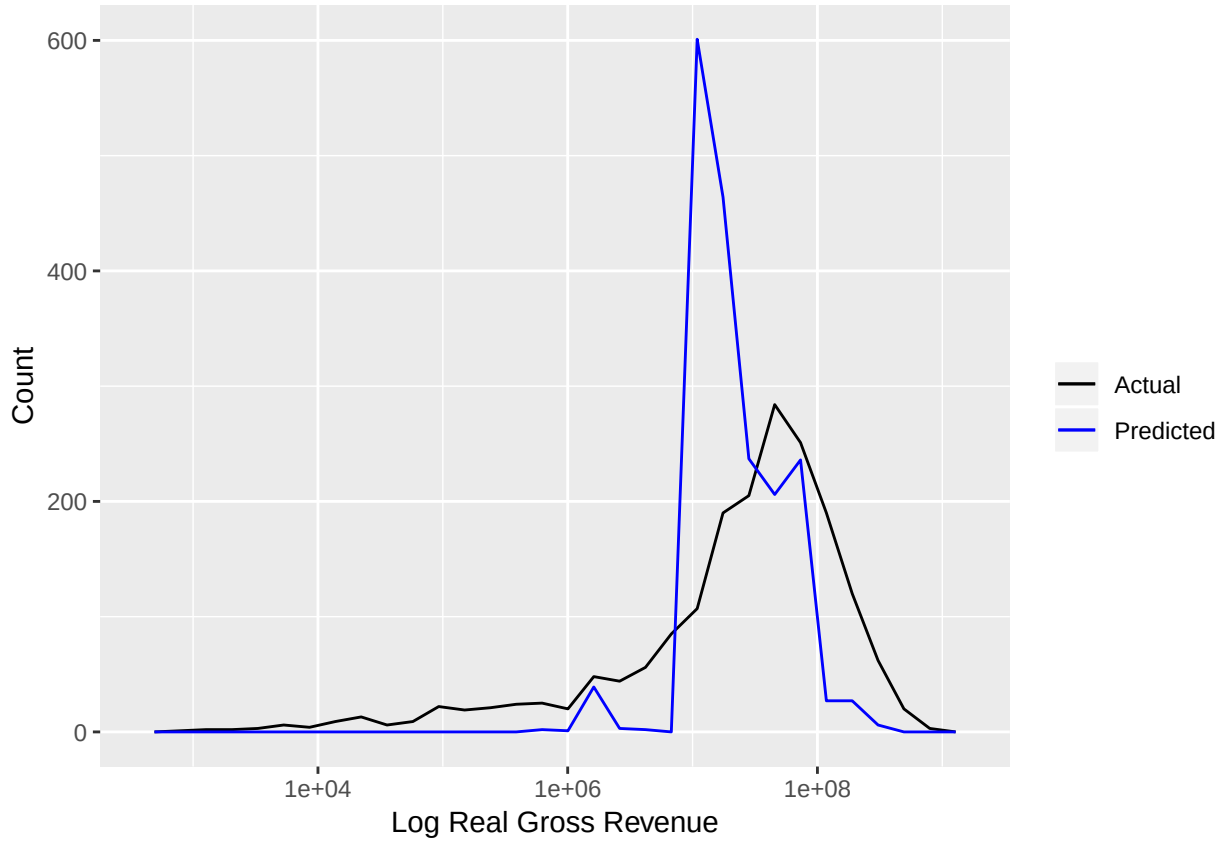

Revenue Actual vs Predicted: Log Scale
Successful Prediction Even With Genres Not Included
Log Real Gross Revenue
1e+08
1e+06
1e+04
0
1
Animation



Overall predictions: clearly not enough to just specify genres

```
train %>%
  add_predictions(mod_genre, 'lpred') %>%
  mutate(pred = 10^lpred) %>%
  ggplot() +
  geom_freqpoly(aes(x = real_gross, color = 'Actual')) +
  geom_freqpoly(aes(x = pred, color = 'Predicted')) +
  scale_x_log10() +
  labs(x = 'Log Real Gross Revenue', y = 'Count') +
  scale_color_manual(name = '', values = c(Actual = 'black', Predicted = 'blue'))
```

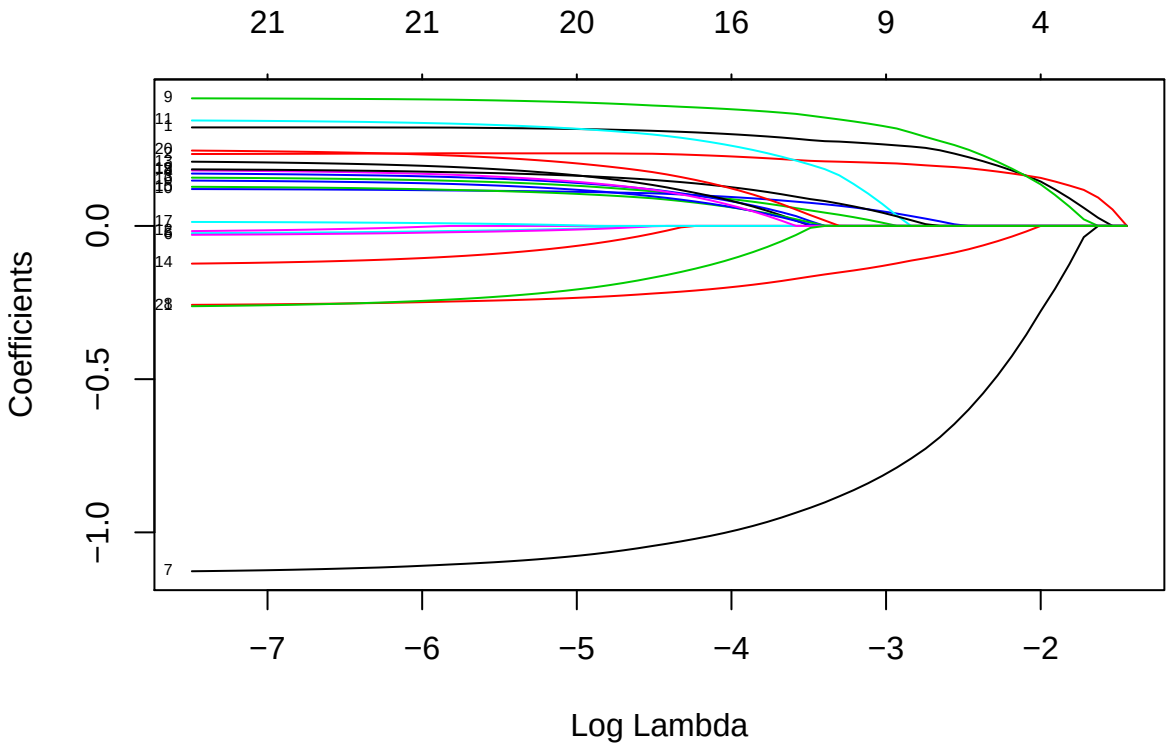
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



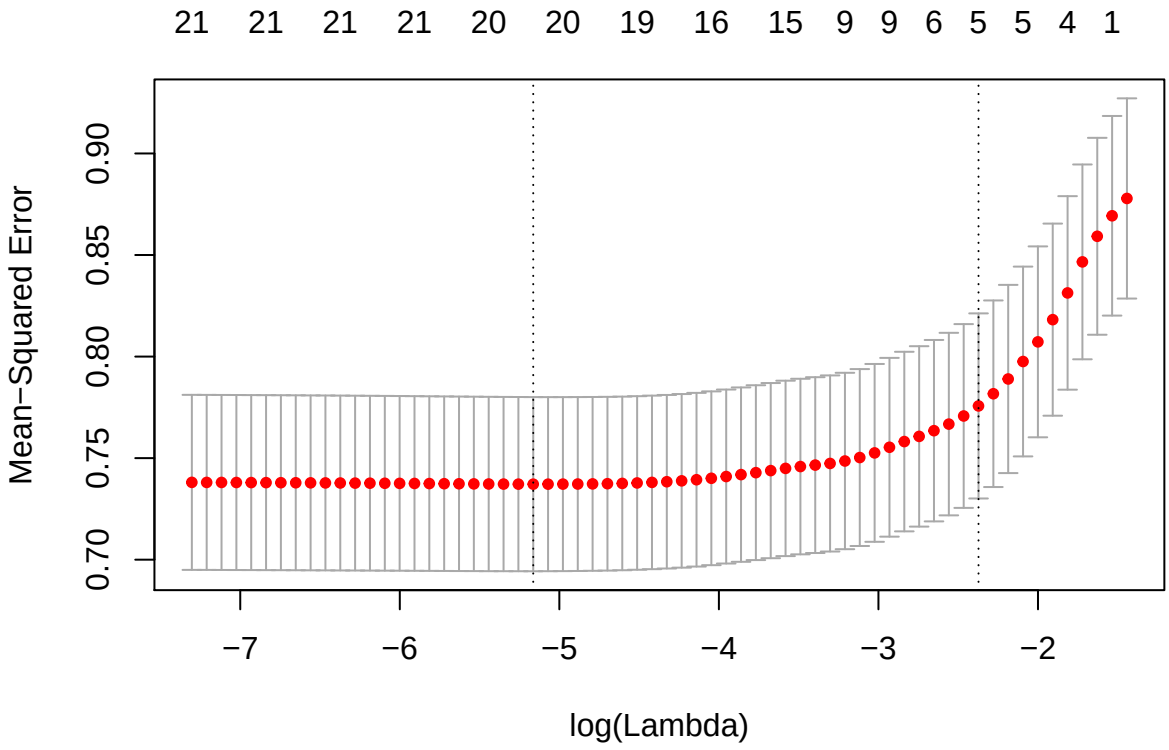
Glmnet: sparse

Try to eliminate most of the genre variables using glmnet and see if results are similar to stepwise. Can't do statistical tests, so not useful for analysis, but can use to aid justification.

```
# 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)  7.157540227
## Action       0.317543067
## Adventure    0.236828217
## Animation    0.135506319
## Biography    0.145534271
## Comedy      -0.010964561
## Crime       -0.014150867
## Documentary -1.084241783
## Drama       -0.237845227
## Family       0.405574661
## Fantasy      0.112498696
## History      0.321743675
## Horror       .
## Music        0.172500588
## Musical     -0.074590785
## Mystery      0.108157227
## Romance      0.123131935
## SciFi        0.003661742
## Sport        0.150296957
## Thriller     0.166927824
## War          0.211110797
## Western     -0.216137862
```

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

```
## 22 x 1 sparse Matrix of class "dgCMatrix"
##                        1
## (Intercept)  7.24955074
## Action       0.21115913
## Adventure    0.18312646
## Animation    .
## Biography    .
## Comedy       .
## Crime        .
## Documentary -0.54632201
## Drama       -0.06351752
## Family       0.22984009
## Fantasy      .
## History      .
## Horror       .
## Music        .
## Musical      .
## Mystery      .
## Romance      .
## SciFi        .
## Sport        .
## Thriller     .
## War          .
## Western      .
```

Use genre model as a base

Next, we started with the genre model and added in additional variables: clear that genre is not sufficient. Several steps:

- Plot other variables (budget, facebook lides, imdb score, number of oscars, content rating, year) against the residuals from the genre model.
  - All of these have somewhat non-random relationships with residuals. Especially budget and IMDB score. Movies with higher budgets make more revenue than predicted by genre (positive residual)
- Based on these non-random relationships, used them all in step wise, but started with the genre variables from the genre model as a base.
- Looked at new residuals of included and excluded relationships. All fairly random.
- Looked at predictions overall. Looks much better.
- Looked at predictions for each individual variable: result??

Overall, this is a pretty good model. Prediction is pretty good. Residuals are more normal. A lot of the variables are not significant, but they are significant when considered together in anova.
However, still not convinced this is the best model. Maybe not a valid assumption that we should start with genre, especially since a lot did come in insignificant. Perhaps these effects are being captured by other variables, so we should not start with the best assumption that we include genre.

```
train_resid <- train %>%
  add_residuals(mod_genre, 'lresid')

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

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

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

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

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

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

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

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

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

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

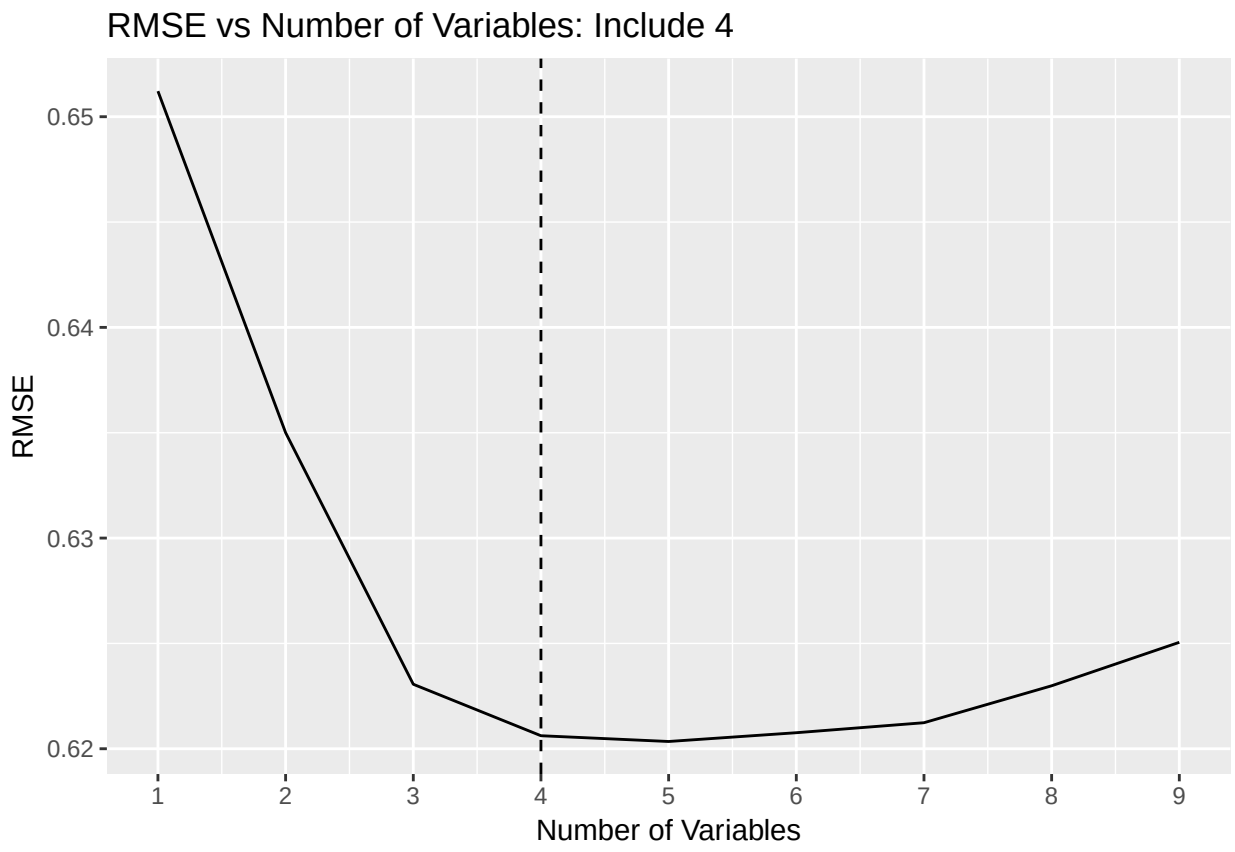
```
##
## [[3]]
```

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

```
# For some log, need to turn -Inf from log(0) to NA
train <- train %>%
  mutate_at(vars(contains('log')), funs(ifelse(is.infinite(.), NA, .)))
valid <- valid %>%
  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_log),
                           starting_vars = genre_xvar,
                           starting_formula = starting_formula)
```

```
# graph RMSE vs number of variables
fit_rmse <- tibble(nvar = 1:length(rmse_lst),
                   rmse = rmse_lst)
ggplot(fit_rmse, aes(x = nvar, y = rmse)) + geom_line() +
```

```
  scale_x_continuous(breaks = seq(1, length(rmse_lst), by = 1)) +
  geom_vline(xintercept = 4, linetype = 'dashed') +
  labs(x = 'Number of Variables', y = 'RMSE',
       title = 'RMSE vs Number of Variables: Include 4')
```




```
# 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
```

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

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

```
# 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
```

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

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

```
gr_resid(mod_all)
```

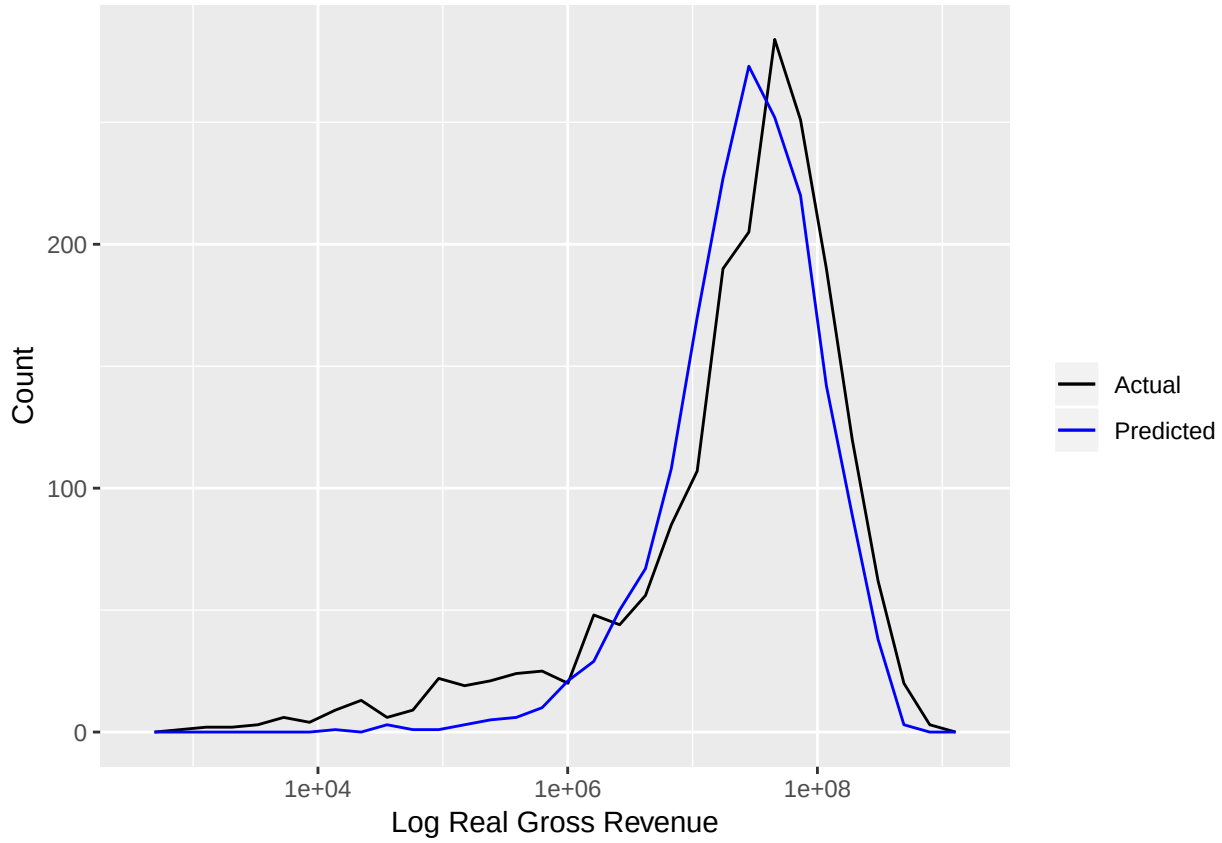
```
# two points with consistently really high residuals (greater than 2) i.e. based on all of their factor
  ### make a graph labeling these and pointing out
# often many points with large negative results less than -2: often get movies that are a flop. High bu
train %>%
  add_residuals(mod_all, 'lresid')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

```
train %>%
  add_predictions(mod_all, 'lpred') %>%
  mutate(pred = 10^lpred) %>%
  ggplot() +
  geom_freqpoly(aes(x = real_gross, color = 'Actual')) +
  geom_freqpoly(aes(x = pred, color = 'Predicted')) +
  scale_x_log10() +
```

```
  labs(x = 'Log Real Gross Revenue', y = 'Count') +
  scale_color_manual(name = '', values = c(Actual = 'black', Predicted = '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).
```



TO DO: * Plot individual predictions for individual variables for mod_all * Add in mod_all2 for starting without genre * Write up justification for not including genre in final model * Fit model on test set (exlucde early years, log variables) * At the end, create those two labeled graphs with the two major outliers in residuals. * Put model results into a table (LaTeX?)