Milestone 2

Varuni Gang, Andrew Greene, Scott Shaffer, Liang-Liang Zhang

April 2, 2017

NB: Many of these topics we dealt with in the "Additonal Material" section of Milestone 1

Discussion about the imbalanced nature of the data and how you want to address it

The various genres have different base rates of occurance, none of which is particularly large. By modeling each genre separately, we can mostly ignore the differences between genres in base rates. For each model, we can provide class weights or use resampling to balance the in-genre and out-of-genre populations for those families of models where such balancing is needed.

Description of your data

Column	*Source*	*Description*
tmdb_id	TMDb	Our master key; all sources use this to join on
release_date	TMDb	When the movie was first released
release_month	TMDb	Derived from *release_date*, an integer between 1 and 12
release_year	TMDb	Derived from *release_date*, the year portion of it
budget	TMDb	Production budget - not trustworthy? Unclear whether it's in absolute or inflation-adjusted dollars
original_language	TMDb	Lots of very small languages, so we will replace this with dummy variables for the 5 most common values
popularity	TMDb	unclear what this means or if it's useful
vote_average	TMdb	The average of the votes for ratings of this movie on TMDb. Not useful
vote_count	TMDb	The number of votes for ratings of this movie on TMDb. Not useful
runtime	TMDb	In minutes.
revenue	TMDb	Also not necessarily trustworth. Unclear whether it's in absolute or inflation-adjusted dollars, or how they handle re-releases
cast_score_genre_...	TMDb + Team	For each genre, our computed "cast-genre affinity score" (see our discussion in Milestone 1)
director_score_genre_...	TMDb + Team	For each genre, our computed "director-genre affinity score" (see our discussion in Milestone 1)
*{RGBHSV}*Intensity*n*	TMDb + Team	For the pixels in the poster, for the channel R, G, B, H, S, or V, what are the top 5 values (0-255)?
*{RGBHSV}*Count*n*	TMDb + Team	For the pixels in the poster, for the channel R, G, B, H, S, or V, for the top 5 values, how many times does each occur in the image?
imdb_id	IMDb + TMDb	Key to query IMDb to obtain movie data and merge key for IMDb to TMDb
votes_imdb	IMDb	Number of votes. It could also be seen as the number of users who voted for the movie.
rating_imdb	IMDb	This is a numeric between 0 and 10, providing a "weighted average of the votes" given by users. The statistical method used to calculate rating has not been disclosed by IMDb.
year_imdb	IMDb	This is the year of release. Not useful, as we already have the same data from TMDb.

Column	*Source*	*Description*
canonical_title_imdb	IMDb	This is the title of the movie. There is a lot we can do with this information (length of title, key words in title... etc.)
kind_imdb	IMDb	This serves as a categorization by one of the following: 'movie', 'tv series', 'tv mini series', 'video game', 'video movie', 'tv movie', 'episode'
score_genre_... _imdb_director	IMDb + Team	For each genre, our computed "director-genre affinity score" (see our discussion in Milestone 1)
score_genre_... _imdb_cast	IMDb + Team	For each genre, our computed "cast-genre affinity score" (see our discussion in Milestone 1)

Use your application knowledge and the insight you gathered from your genre pair analysis and additional EDA to design Y. Do you want to include all genres? Are there genres that you assume to be easier to separate than others? Are there genres that could be grouped together? There is no one right answer here. We are looking for your insight, so be sure to describe your decision process in your notebook.

What does your choice of Y look like?

We addressed this in Milestone 1:

We have decided to treat each genre label as an independent outcome. Thus, a romantic comedy would have `TRUE` for the `genre_Romance` and `genre_Comedy` columns. In particular, for each of the 19 genres in the TMDB data, we will create an independent model and use 0-1 loss.

This approach has several advantages.

First and second, it makes scoring our loss more fair *and* more productive. For example, if we successfully predict that the romantic comedy is a "Romance" but fail to predict that it is a "Comedy", we want our accuracy to reflect that we were half correct – and we want the feedback into our model to reflect *which* half.

Third, it eliminates the risk of creating a hand-curated list of hybrid genres that will fail to predict some new fusion film that might occur in the future.

Fourth, it allows us to use our "genre affinity" approach, as described below, to affiliate actors and directors with the genres in which they appear most often.

Which features do you choose for X and why?

Based on our initial decision to build separate models for each genre, we applied a generalized linear model to each genre separately using the features mentioned above. We've identified that almost all of the features are valuable predictors for at least one genre. We identified a feature's predictive value based on the respective p-values for a given genre.

For example, we have identified that the following features are highly predictive for the genre Action: year, day, budget, vote_average, vote_count, runtime, revenue and rating_imdb. In addition, year, month, budget, vote_average, runtime, rating_imdb and votes_imdb are highly predictive features for the genre Adventure. We have mapped a different set of base features for each genre that we will use for further analysis. In the image below, blue p-values are less than .05 meaning that we can reject the null hypothesis.

	Action	Adventure	Animation	Comedy	Crime	Documentar	Drama	Family	Fantasy	Foreign	History	Horror	Music	Mystery	Romance	Science_Fict	TV_Movie	Thriller	War	Western
Intercept	0.98359127	1.75E-13	0.28494543	0.01673483	3.17E-20	0	0.00524213	6.87E-18	3.28E-08	2.40E-22	0.22469218	2.00E-70	0.15473927	2.37E-07	0.01077349	3.94E-16	2.63E-10	2.21E-74	4.36E-17	1.51E-192
year	0.02349321	1.98E-22	0.26346474	1.30E-06	2.21E-32	2.31E-302	0.86014539	4.29E-11	0.00015825	1.90E-14	0.30738204	2.90E-60	0.20621471	1.24E-13	3.29E-08	4.95E-11	4.21E-07	1.37E-60	2.23E-26	1.52E-217
month	0.20973611	0.00019657	0.00121537	3.32E-07	0.97076794	0.00299739	3.34E-10	1.19E-10	0.01834606	0.60747728	0.00016596	0.6177357	4.01E-05	0.01360473	6.03E-05	0.67080229	3.94E-06	0.25552848	0.22943181	0.00057475
day	0.00335944	0.15677771	0.30935974	1.19E-15	0.00180988	0.03447077	3.03E-13	0.01618574	0.37705648	0.44315954	0.12346676	0.98845237	0.29933942	0.06762203	9.08E-06	0.60628499	0.39321715	0.00486149	0.46108193	0.03775075
budget	1.63E-49	4.36E-53	4.39E-11	0.02239135	9.01E-12	3.86E-07	0.66793843	3.31E-11	7.74E-22	0.15590512	7.74E-10	0.00249198	0.03382595	1.21E-05	0.61099997	1.04E-14	0.27880134	2.64E-17	5.71E-11	0.00437026
popularity	0.15130747	0.63183361	0.77333855	0.65392696	0.08580879	5.48E-64	0.00242771	0.19731165	0.22940522	1.42E-07	0.41955405	0.01496773	0.07856603	0.18115803	0.07357711	0.06005007	0.00378075	0.08377849	0.16752473	0.18520015
vote_averag	1.51E-103	1.96E-36	7.02E-123	2.39E-151	4.12E-53	9.54E-05	1.88E-84	5.75E-36	1.10E-26	9.91E-82	3.02E-08	8.63E-56	3.20E-06	2.02E-47	8.96E-55	4.95E-36	1.41E-06	2.66E-118	9.05E-11	0.9360079
vote_count	0.00410607	0.07373629	0.16436717	0.07210654	0.03341468	0.00558439	2.23E-11	0.68822672	0.60783961	1.11E-08	0.00192192	0.02388958	0.36019997	0.03298797	0.03813369	0.00023454	0.67454821	0.07924762	0.00624998	0.96162462
runtime	3.92E-107	9.97E-24	3.29E-304	1.82E-17	6.19E-49	5.04E-08	0	0.35286094	0.06461226	2.66E-38	2.26E-16	7.85E-15	1.91E-07	2.04E-11	2.94E-124	0.00031104	0.00073028	1.16E-77	9.14E-11	1.79E-06
revenue	7.44E-11	0.15602285	0.00113001	0.10919855	5.23E-06	0.91352843	6.57E-10	3.14E-08	0.44436422	0.05914571	0.11045462	0.01400169	0.01204716	0.02140889	0.77429912	7.63E-05	0.32328455	1.20E-08	0.0082534	0.05886476
rating_imdb	1.04E-55	5.09E-05	1.09E-21	0.06285678	0.33280331	2.25E-120	8.22E-17	0.00755424	0.00915858	7.33E-10	1.74E-16	2.00E-161	3.73E-40	0.00221309	0.03018023	6.83E-28	0.00961335	3.73E-57	3.78E-07	0.05026974
votes_imdb	0.89575338	0.01423909	0.0001419	0.75763685	2.49E-08	0.023522	4.91E-19	7.19E-07	0.5691502	1.50E-08	0.00133537	0.97018202	0.24211362	0.00010413	0.03521004	0.89181892	0.00474605	7.92E-10	0.00052682	0.91798852

Figure 1: Tablular representation of p-values of features for each genre based on glm model

What we have done so far is just the first step, and as we start building models, we'll follow a *mixed selection* approach as discussed on page 79 of the textbook. We will also use regularization techniques such as LASSO (and the equivalent for CNN models) to identify which of the remaining features can be omitted.

How do you sample your data, how many samples, and why? We addressed this in Milestone 1:

We are separating our data as follows: We take the last digit of the movie's TMDB id. (We have verified that this seems to be evenly spread across years and genres.)

- If the last digit is 0-4, then this movie is training data from the get-go
- If it is 5, then this movie was test data for Milestone 1, and is added to the training data as of Milestone 2
- If it is 6, then this movie is test data for Milestone 2, and is added to the training data after that
- If it is 7, then this movie is test data for Milestone 3, and is added to the training data after that
- If it is 8, then this movie is test data for Milestone 4, and is added to the training data after that
- If it is 9, then this movie is test data for Milestone 5 (the final deliverable)

We will allow ourselves to "peek" ahead only to download related data (e.g., to make sure we have everything from IMDb early in the process).

Appendix A - Visualizations

This appendix contains visualizations of the features under consideration for use in our models.

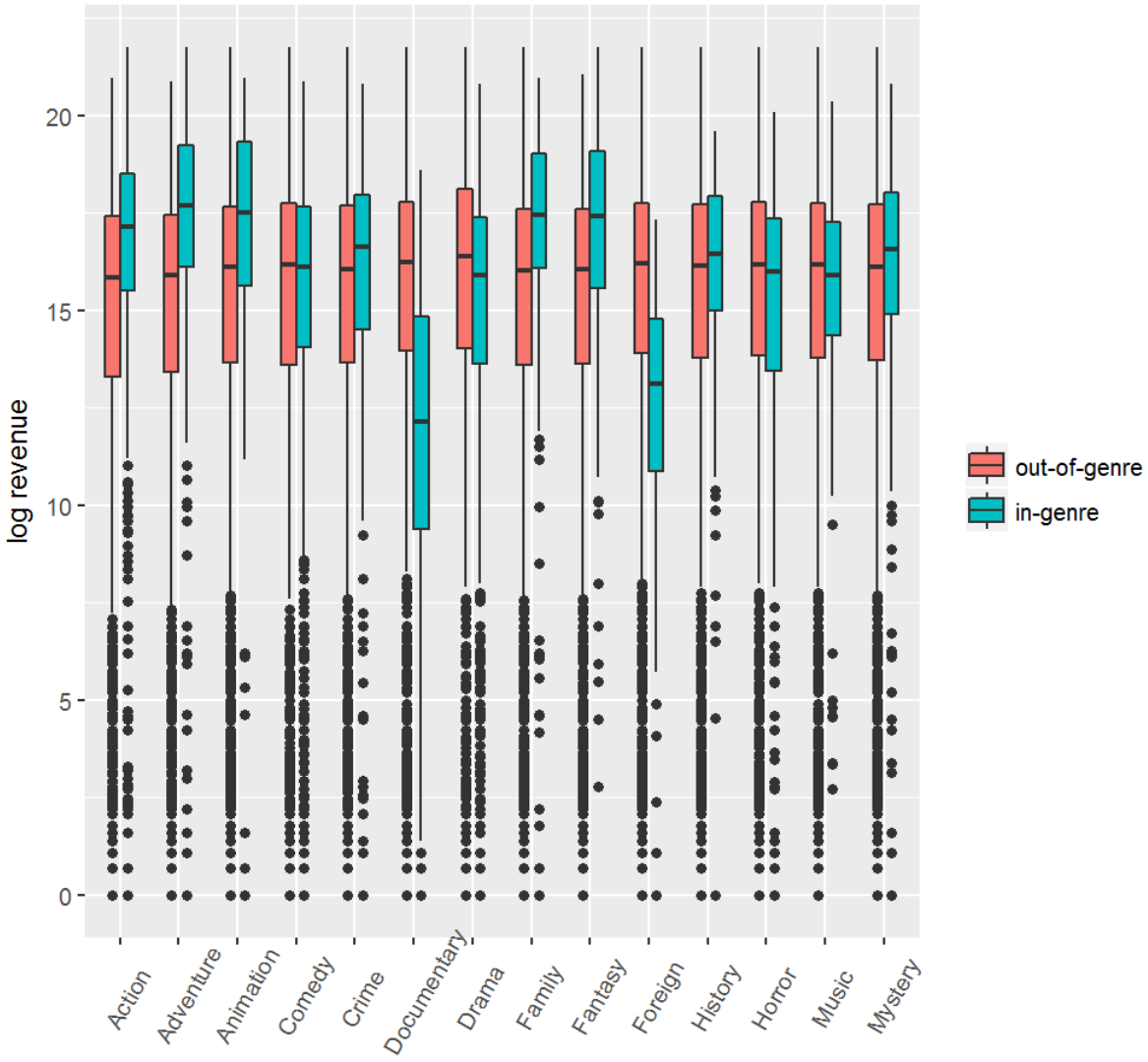


Figure 2: Visualization Boxplot_for_revenue

We can see here certain genres for which `revenue` appears promising as a predictor, such as Adventure (where the median of in-genre is higher that the 75%ile of out-of-genre) and Documentary (where the median of in-genre is lower than the 25%ile of out-of-genre). There are also some genres (such as Comedy) where the IQRs for in-genre and out-of-genre are comparable, and so for the models for those genres we do not expect `revenue` to be a useful predictor.

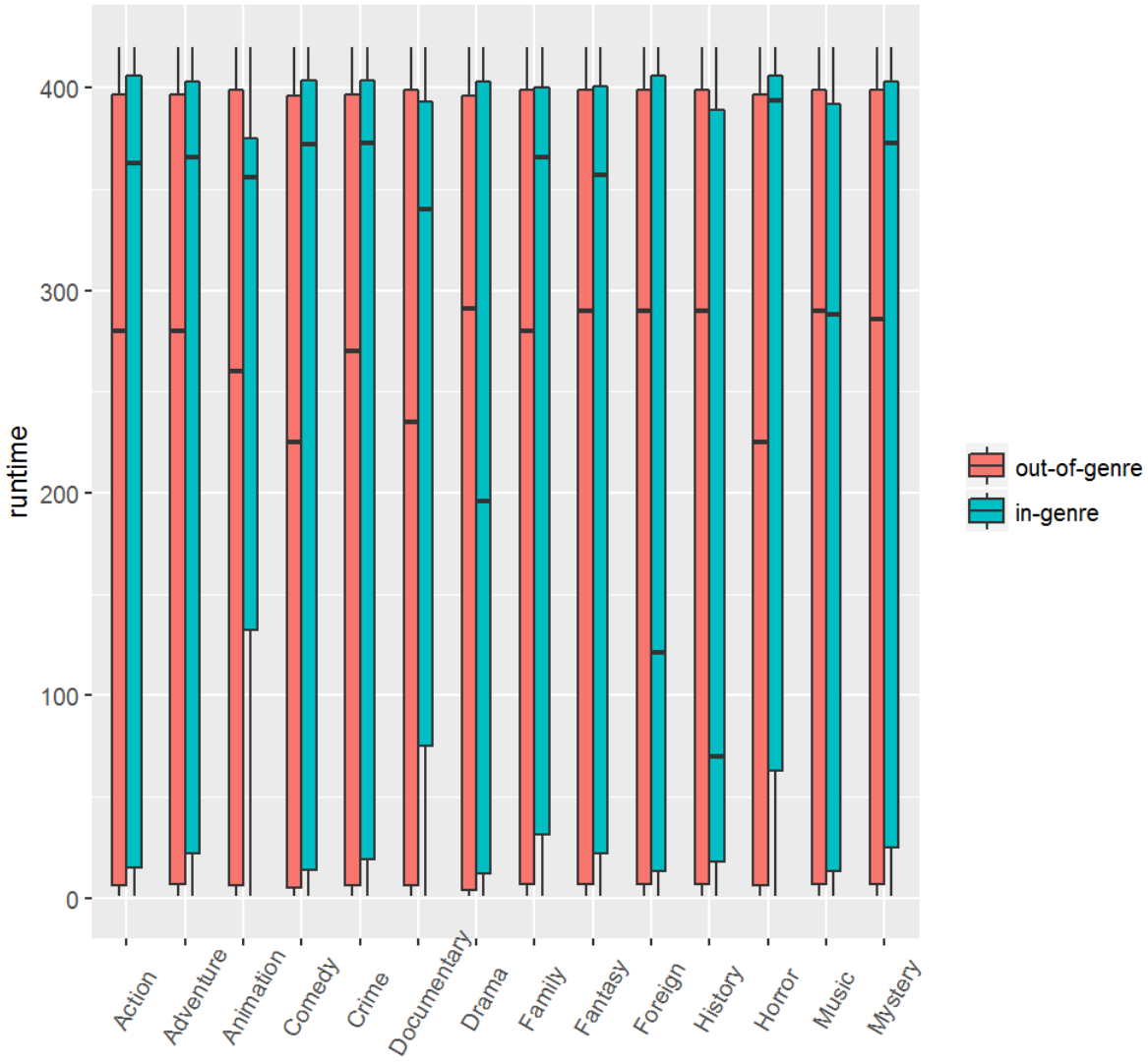
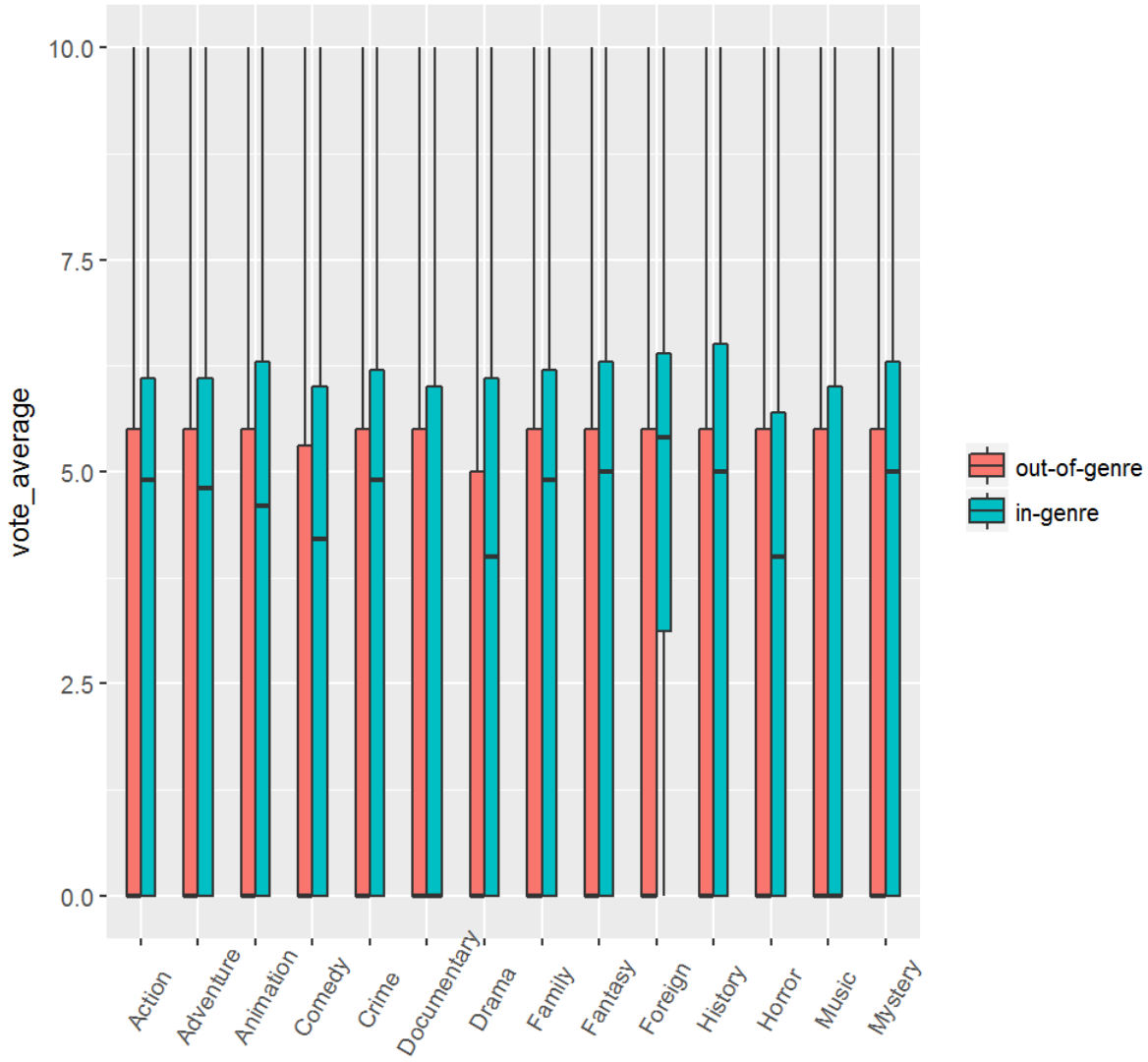
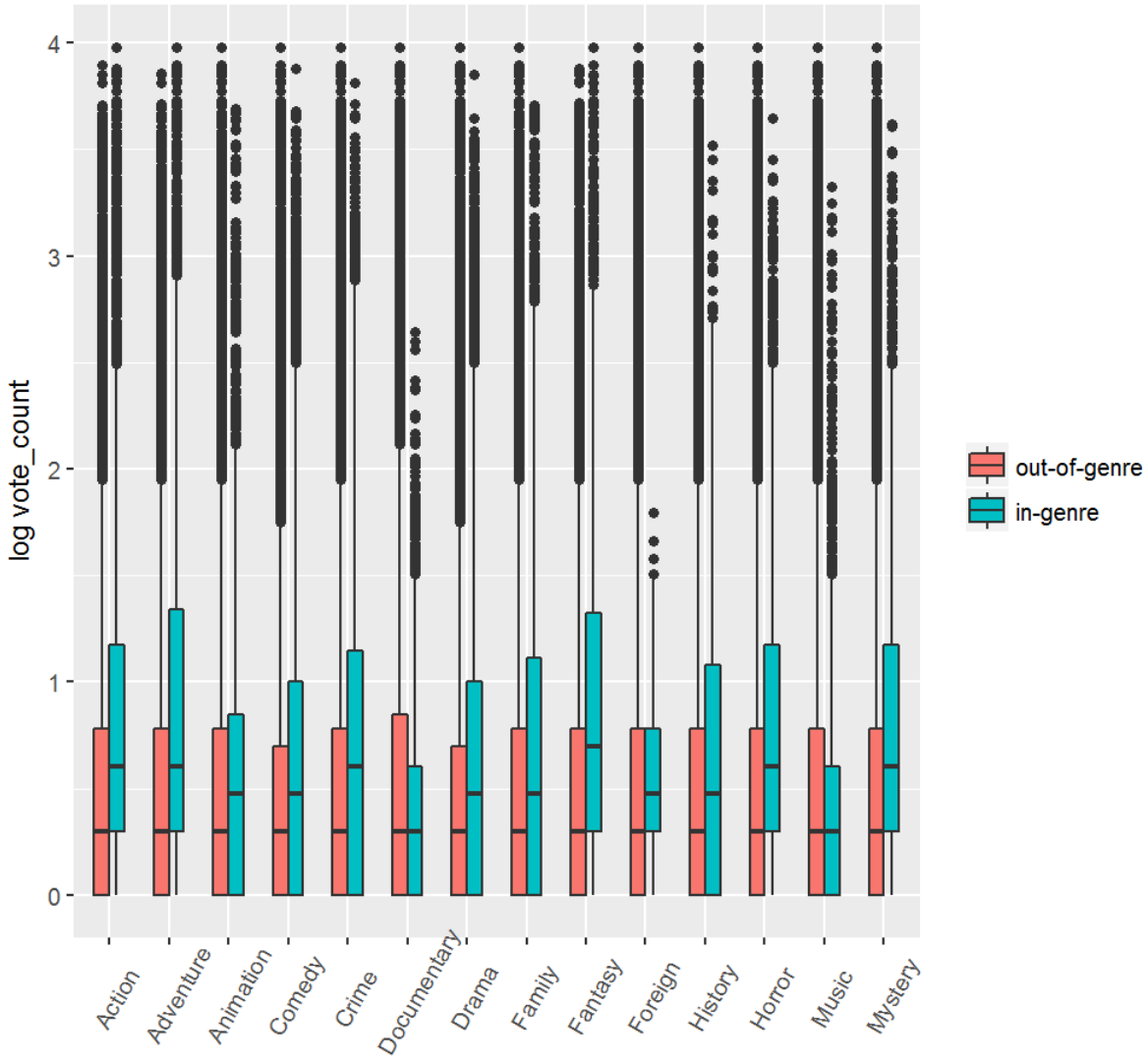


Figure 3: Visualization Boxplot_for_runtime

Here is an example of a visualization that indicates that `runtime` is not likely to be a useful predictor for most genres.


vote_average
10.0
7.5
5.0
2.5
0.0
Action
Adventure
Animation
Comedy
Crime
Documentary
Drama
Family
Fantasy
Foreign
History
Horror
Music
Mystery
out-of-genre
in-genre


log vote_count
4
3
2
1
0
Action
Adventure
Animation
Comedy
Crime
Documentary
Drama
Family
Fantasy
Foreign
History
Horror
Music
Mystery
out-of-genre
in-genre

Code for the preceding plots:

```
library(ggplot2)
library(scales)
```

```
tmdb <- read.delim('tmdb.tsv')
```

budget

```
png('Boxplot for "budget".png', 1000, 1000, res=160)
# isolate data
tmdb_budget <- data.frame(budget = tmdb[,5], tmdb[,12:25])
colnames(tmdb_budget) <- sub('genre_', '', colnames(tmdb_budget))

#reshape data
tmdb_budget_melt <- melt(tmdb_budget, id.vars = "budget")

#plot
ggplot(tmdb_budget_melt, aes(x = factor(variable),  y= log(budget), fill=factor(value)))+
  geom_boxplot(position=position_dodge(width=.50),  width = 0.5)+
  theme(axis.text.x=element_text(angle=60, hjust = 0.4, vjust = 0.6)) +
  labs(title = "",
       x = '',
       y = 'log Budget') +
  scale_fill_discrete(name="",
                      breaks=c("0", "1"),
                      labels=c("out-of-genre", "in-genre"))
graphics.off()
```

popularity

```
png('Boxplot for "popularity".png', 1000, 1000, res=160)
# isolate data
tmdb_popularity <- data.frame(popularity = tmdb[,7], tmdb[,12:25])
colnames(tmdb_popularity) <- sub('genre_', '', colnames(tmdb_popularity))

#reshape data
tmdb_popularity_melt <- melt(tmdb_popularity, id.vars = "popularity")

#plot
ggplot(tmdb_popularity_melt, aes(x = factor(variable),  y = log(popularity), fill=factor(value)))+
  geom_boxplot(position=position_dodge(width=.50),  width = 0.5)+
  theme(axis.text.x=element_text(angle=60, hjust = 0.4, vjust = 0.6)) +
  labs(title = "",
       x = '',
       y = 'log popularity') +
  scale_fill_discrete(name="",
                      breaks=c("0", "1"),
                      labels=c("out-of-genre", "in-genre"))
graphics.off()
```

vote_average

```
png('Boxplot for "vote_average".png', 1000, 1000, res=160)
# isolate data
tmdb_voteaverage <- data.frame(vote_average = tmdb[,8], tmdb[,12:25])
colnames(tmdb_voteaverage) <- sub('genre_', '', colnames(tmdb_voteaverage))

#reshape data
tmdb_voteaverage_melt <- melt(tmdb_voteaverage, id.vars = "vote_average")

#plot
ggplot(tmdb_voteaverage_melt, aes(x = factor(variable),  y= vote_average, fill=factor(value)))+
  geom_boxplot(position=position_dodge(width=.50),  width = 0.5)+
  theme(axis.text.x=element_text(angle=60, hjust = 0.4, vjust = 0.6)) +
  labs(title = "",
       x = '',
       y = 'vote_average') +
  scale_fill_discrete(name="",
                      breaks=c("0", "1"),
                      labels=c("out-of-genre", "in-genre"))
graphics.off()
```

vote_count

```
png('Boxplot for "vote_count".png', 1000, 1000, res=160)
# isolate data
tmdb_votecount <- data.frame(vote_count = tmdb[,9], tmdb[,12:25])
colnames(tmdb_votecount) <- sub('genre_', '', colnames(tmdb_votecount))

#reshape data
tmdb_votecount_melt <- melt(tmdb_votecount, id.vars = "vote_count")

#plot
ggplot(tmdb_votecount_melt, aes(x = factor(variable),  y= log(vote_count, base= 10), fill=factor(value)))
  geom_boxplot(position=position_dodge(width=.50),  width = 0.5)+
  theme(axis.text.x=element_text(angle=60, hjust = 0.4, vjust = 0.6)) +
  labs(title = "",
       x = '',
       y = 'log vote_count') +
  scale_fill_discrete(name="",
                      breaks=c("0", "1"),
                      labels=c("out-of-genre", "in-genre"))
graphics.off()
```

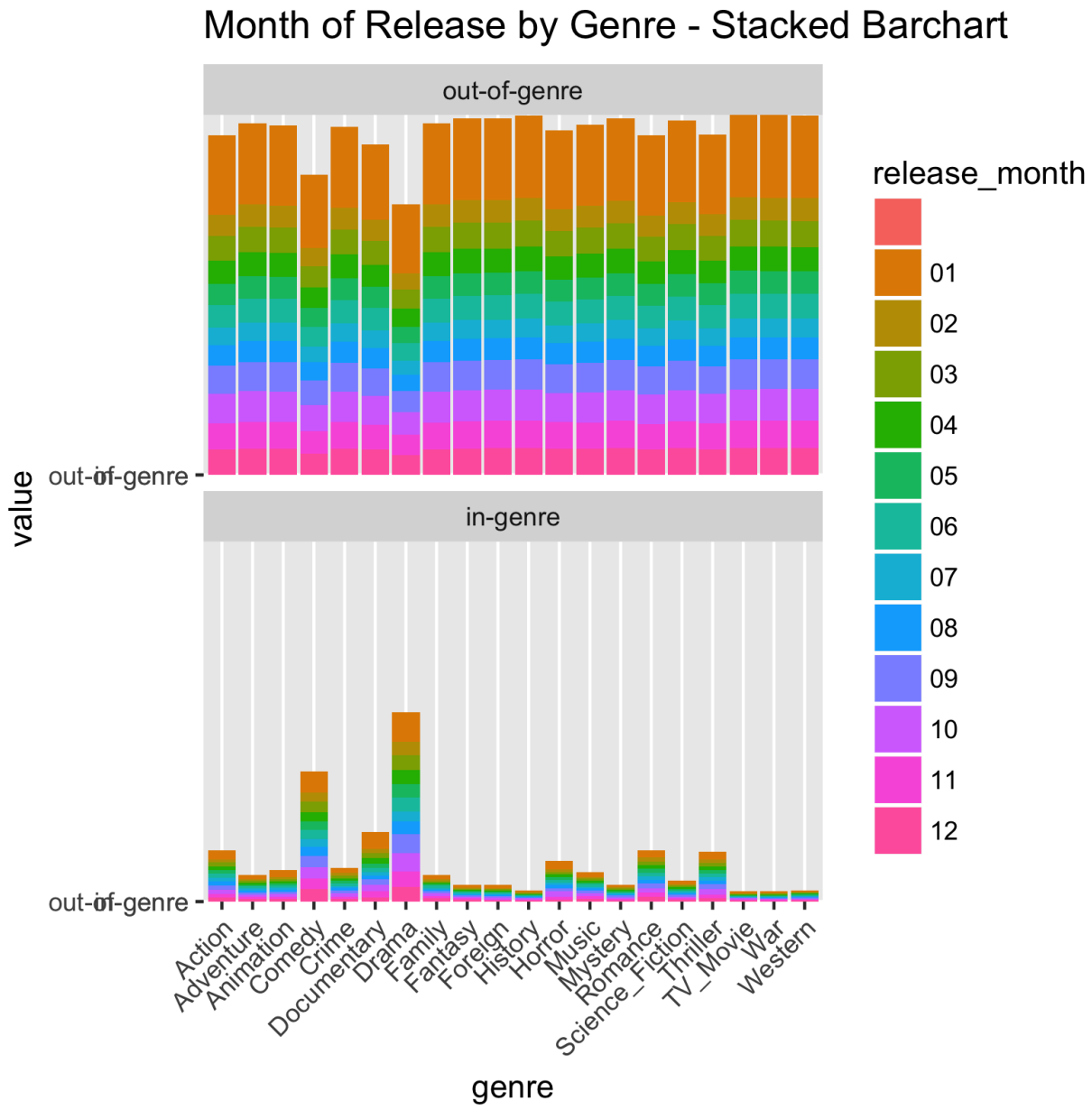
runtime

```
png('Boxplot for "runtime".png', 1000, 1000, res=160)
# isolate data
tmdb_runtime <- data.frame(runtime = tmdb[,10], tmdb[,12:25])
# remove movies with runtime = None
```

```
tmdb_runtime_removeNone <- tmdb_runtime[!tmdb_runtime$runtime=='None', ]
colnames(tmdb_runtime_removeNone) <- sub('genre_', '', colnames(tmdb_runtime_removeNone))

#reshape data
tmdb_runtime_melt <- melt(tmdb_runtime_removeNone, id.vars = "runtime")

#plot
ggplot(tmdb_runtime_melt, aes(x = factor(variable),  y = as.numeric(runtime), fill=factor(value)))+
  geom_boxplot(position=position_dodge(width=.50),  width = 0.5)+
  theme(axis.text.x=element_text(angle=60, hjust = 0.4, vjust = 0.6)) +
  labs(title = "",
       x = '',
       y = 'runtime') +
  scale_fill_discrete(name="",
                      breaks=c("0", "1"),
                      labels=c("out-of-genre", "in-genre"))
graphics.off()
```

revenue

```
png('Boxplot for "revenue".png', 1000, 1000, res=160)
# isolate data
tmdb_revenue <- data.frame(revenue = tmdb[,11], tmdb[,12:25])
colnames(tmdb_revenue) <- sub('genre_', '', colnames(tmdb_revenue))

#reshape data
tmdb_revenue_melt <- melt(tmdb_revenue, id.vars = "revenue")

#plot
ggplot(tmdb_revenue_melt, aes(x = factor(variable),  y= log(revenue), fill=factor(value)))+
  geom_boxplot(position=position_dodge(width=.50),  width = 0.5)+
  theme(axis.text.x=element_text(angle=60, hjust = 0.4, vjust = 0.6)) +
  labs(title = "",
       x = '',
       y = 'log revenue') +
  scale_fill_discrete(name="",
                      breaks=c("0", "1"),
                      labels=c("out-of-genre", "in-genre"))
graphics.off()
```


Month of Release by Genre - Stacked Barchart
out-of-genre
in-genre
value
out-of-genre
out-of-genre
release_month
01
02
03
04
05
06
07
08
09
10
11
12
Action
Adventure
Animation
Comedy
Crime
Documentary
Drama
Family
Fantasy
Foreign
History
Horror
Music
Mystery
Romance
Science_Fiction
Thriller
TV_Movie
War
Western
genre

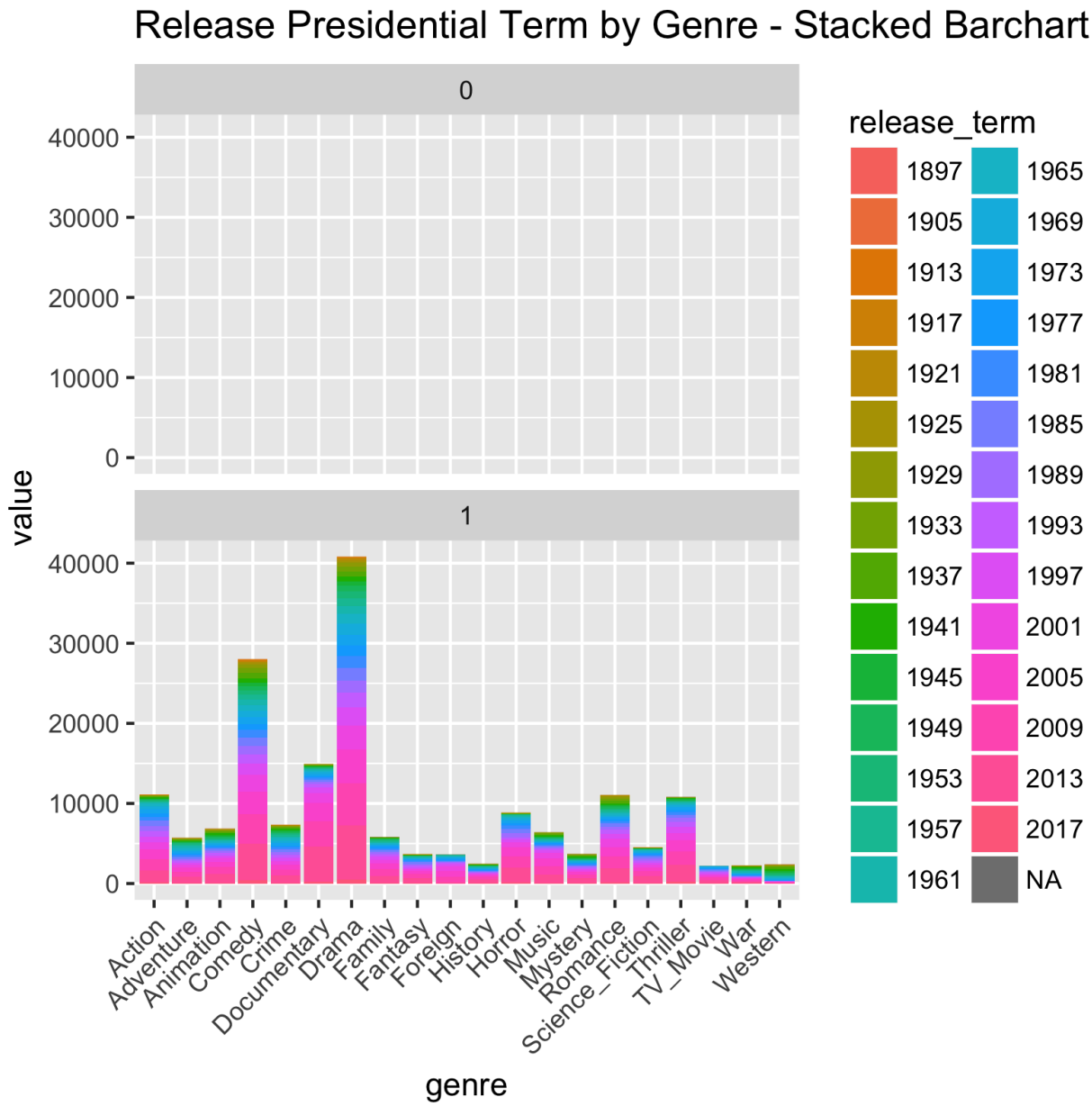

Release Presidential Term by Genre - Stacked Barchart
0
1
value
40000
30000
20000
10000
0
Action
Adventure
Animation
Comedy
Crime
Documentary
Drama
Family
Fantasy
Foreign
History
Horror
Music
Mystery
Romance
Science_Fiction
Thriller
TV_Movie
War
Western
genre
release_term
1897
1905
1913
1917
1921
1925
1929
1933
1937
1941
1945
1949
1953
1957
1961
1965
1969
1973
1977
1981
1985
1989
1993
1997
2001
2005
2009
2013
2017
NA

Figure 4: Visualization Release_Year_4_by_Genre_Stacked_Barchart

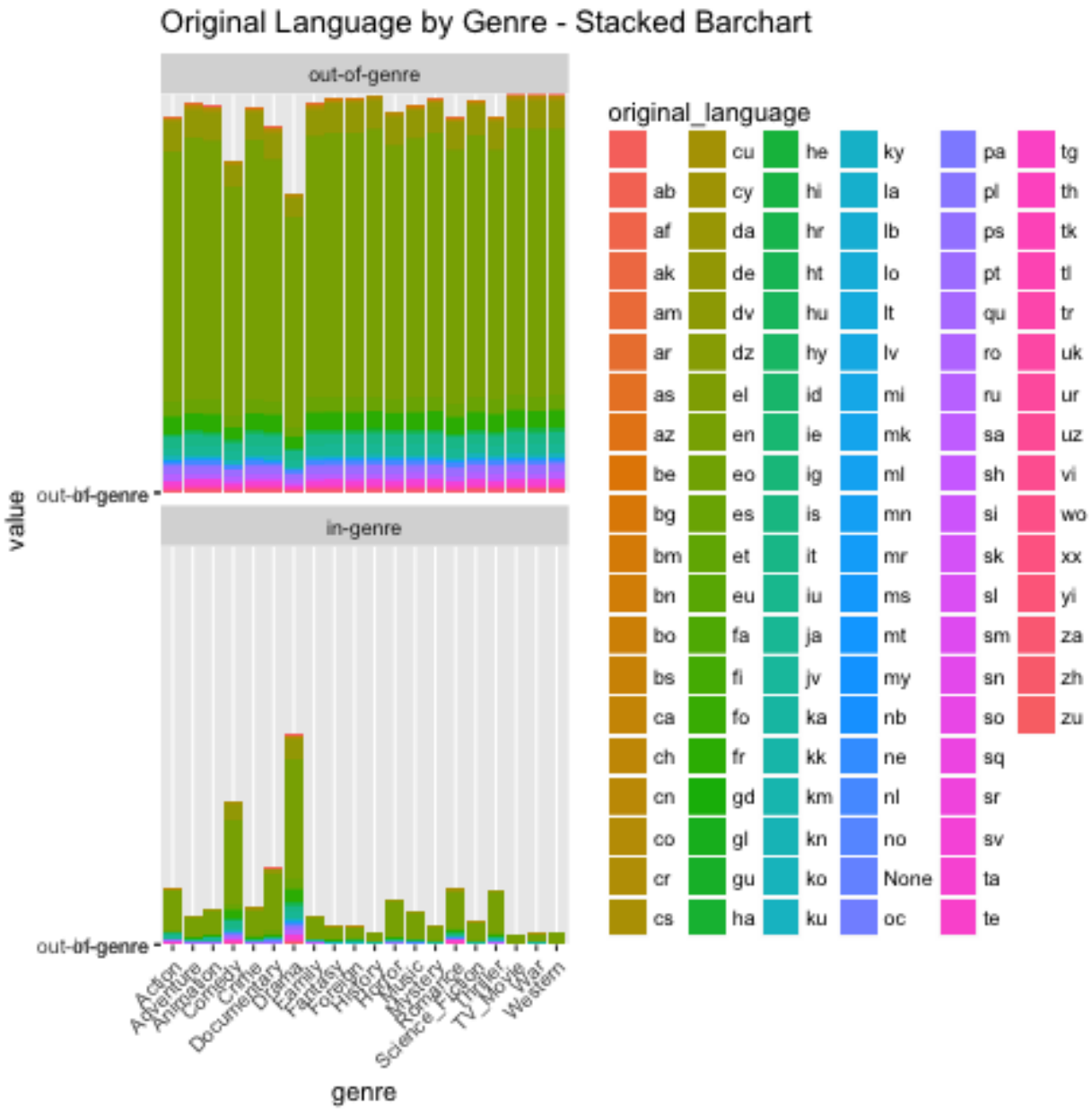


Figure 5: Visualization Original_Language_by_Genre_Stacked_Barchart