ECON 424 - Final Project (Winter 2023)

Submitted by Raphael Shawn Gozali (20805288) and Jason Tedjosoesilo (20801292)

Questions

For this project, these are the questions that we will be answering:

1. Without any interactions, which covariate(s) affect revenue the most? Are there any certain genres, actors, or keywords that affect revenue the most?

2. If we do interactions between genres, actors, and keywords, are there any particular combination(s) that affect revenue the most?

3. Do people's preferences on movie genres and actors change over the decades?

Data

In this project, we will be using the Movies dataset from Kaggle: https://www.kaggle.com/datasets/akshaypawar7/millions-of-movies?resource=download. We downloaded this dataset on April 2, 2023 on 2pm EST and continue to work with it locally. This dataset consists of around 723 thousand rows of movies, with their descriptions and details separated into columns. The response variable that we are focusing on from this dataset is the "revenue" column, which shows us the revenue of the movie. The covariates that might be useful from this dataset includes: date published, budget, genres, casts, keywords, and runtime.

```r
data <- read.csv("./movies.csv")
```

Before we do some analysis, we will clean the data. First, we will remove unnecessary columns and also data with zero revenues. We are only keeping released movies above 40 minutes as we are not including short movies in the data. This is based on the definition from the Academy of Motion Picture Arts and Sciences, which define a short film as "an original motion picture that has a running time of 40 minutes or less, including all credits". We also do not include movies with a runtime of value 999. There might also be duplicates in the data, so we need to remove duplicates as well. Since people can add random movies to this database, we try as best as we can to filter those out. To make sure that a movie is legitimate, we filter the movies that do not have any genres, production companies, and credits (all three are empty values).

```r
# Only movies that has revenue, is released, and more than
# 40 minutes long Also remove if runtime = 999 and remove
# the movies with missing genres, production companies, and
# credits
clean_data <- data[data$revenue > 0 & data$status == "Released" &
    data$runtime > 40 & data$runtime != 999 & !(data$genres ==
    "" & data$production_companies == "" & data$credits == ""),
    !names(data) %in% c("overview", "popularity", "status", "tagline",
        "vote_average", "vote_count", "poster_path", "backdrop_path",
        "recommendations")]
```

```r
# removing empty ID rows
clean_data <- clean_data[!is.na(clean_data$id), ]

# resetting row.names
row.names(clean_data) <- NULL

# remove duplicate data
clean_data <- clean_data[!duplicated(clean_data[, 1]), ]

write.csv(clean_data, file = "./movies_filtered.csv", row.names = FALSE)

n <- length(clean_data[, 1])
```

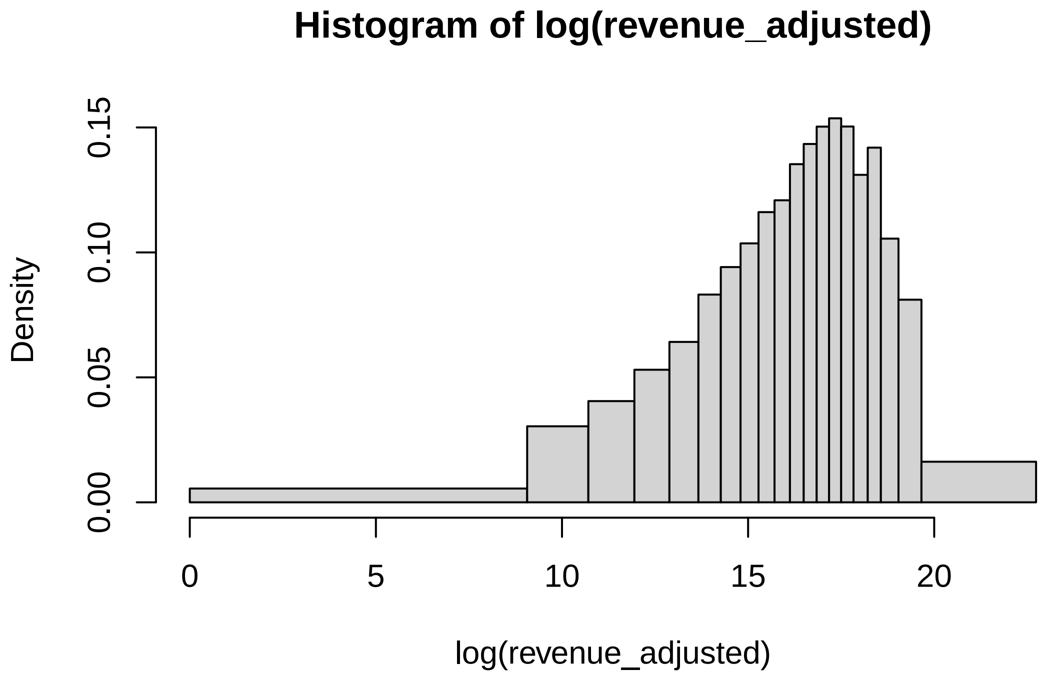
We took this data and use a Python API to calculate the adjusted revenues by adding inflation factors. Here are the Python code below.

```python
# import libraries
import pandas as pd
import matplotlib.pyplot as plt
import requests
import json
from tqdm import tqdm
from pathlib import Path

# get movies from movies_filtered.csv
movies = pd.read_csv(r'movies_filtered.csv')
n = len(movies)

# Function to get the inflation rate from the API
# Input: start_time (based on release_date)
# Output: inflation rate (inflated to April 3, 2023)
def get_dollar(start_time):
    api_url = "https://www.statbureau.org/calculate-inflation-price-jsonp?jsoncallback=?"

    headers = {'Content-type': 'application/json'}

    payload = {
        "country": "united-states",
        "start": str(start_time),
        "end": "2023/04/03",
        "amount": "1",
        "format": True
    }
    response = requests.post(api_url,  data=json.dumps(payload), headers=headers)
    my_bytes = response.content

    amount_s = my_bytes.decode('utf8').replace("'", '"')
    amount_s = amount_s[4:-2]
    return float(amount_s)

# getting the inflation rate of each movie
inflation = [0] * n
```

```
failed = []
for i in tqdm(range(n)):
    try:
        inflation[i] = get_dollar(movies.iloc[i]['release_date'])
    except:
        failed.append(i)

# data with no release_date will be given inflation = 1 (no inflation)
for fail in failed:
    inflation[fail] = 1

# getting the adjusted revenue for each movie
revenue_adjusted_c = movies['revenue'] * np.array(inflation)
df2 = movies.assign(revenue_adjusted=revenue_adjusted_c)

# exporting the file movies_adjusted.csv
filepath = Path('movies_adjusted.csv')
filepath.parent.mkdir(parents=True, exist_ok=True)
df2.to_csv(filepath)
```

We calculated all the revenues inflated to April 3, 2023 and include them in the data as a new column called "revenue_adjusted". The data that do not have release_date will have their adjusted revenue be the same as their revenue.

```
clean_data <- read.csv("./movies_adjusted.csv")
n <- length(clean_data[, 1])

hist(log(clean_data$revenue_adjusted), breaks = quantile(log(clean_data$revenue_adjusted),
    p = seq(0, 1, length.out = 21)), freq = FALSE, xlab = "log(revenue_adjusted)",
    main = "Histogram of log(revenue_adjusted)")
```



From the histogram above, we can see that when we split the dataset into 20 bins of 5% quantiles each, most of them have high values on revenues after being adjusted to inflation.

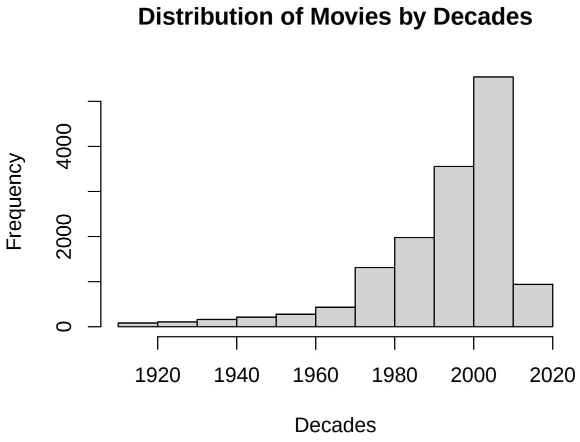
After cleaning the data, we have a total of 14,678 rows of movies.

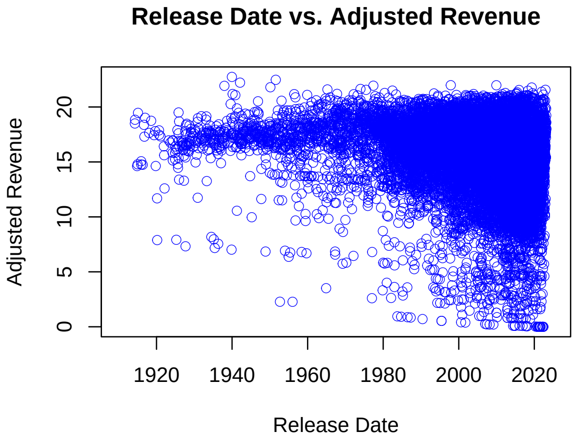
Limitations

Before we start with our methods, we would like to address the limitation of the methods that we are using here.

```
clean_data$year <- format(as.Date(clean_data$release_date), format = "%Y")
clean_data$decades <- floor(as.numeric(clean_data$year)/10) *
    10
df_list <- split(clean_data, clean_data$decades)
```

```
par(mfrow = c(1, 2))
hist(clean_data$decades, xlab = "Decades", main = "Distribution of Movies by Decades")
plot(as.Date(clean_data$release_date), log(clean_data$revenue_adjusted),
    xlim = c(as.Date("1910-01-01"), as.Date("2025-01-01")), col = "blue",
    lwd = 0.5, xlab = "Release Date", ylab = "Adjusted Revenue",
    main = "Release Date vs. Adjusted Revenue")
```





Based on the histogram above, we can see that the distribution of the movies is not equal across time. There are more recent movies in the dataset than older movies. Also, from the scatterplot above, it is visible that the variance of the data changes over time. This shows that our data is non-stationary. In addition, we do not see a lot of data points in the early decades, and not a lot of them have low revenues. Since this is a problem, we do not use time as a variable in our model. We do not change the distribution of the data since it is better to have more data on recent decades. This gives us a notion of weights on the data, with more movies closer to the present.

Other than that, note that the data cleaning process is not perfect. Since the data source is publicly available to be added, there are some made-up data points in the raw data. We cleaned it as best as we could, but there is no guarantee on whether there are still some made-up data in this dataset.

Methods & Results

To answer the questions above, we will create some models using LASSO method via cv.glmnet function. We use LASSO since we will have a lot of variables coming from the text columns. Thus, we use LASSO to do variable reduction to obtain a balanced model.

Model without interactions

To build the model, we will have to do a text analysis on the texts first. There are five columns which have text variables: production_companies, genres, credits, keywords, and original_language. The terms that we are using from these columns are separated by dashes ("-"). Using this information, we will separate these text columns into a sparse matrix full of terms. For each document, the term will be equal to 1 if it appears on that document, and 0 if it does not appear. After getting each term separated, we added single-letter prefixes with "$" on the terms to be able to tell which column the terms come from. Then, we filter the sparse matrix so that only terms that appear in at least 50 movies are included to be in the model.

```
library(pdftools)
```

```
## Using poppler version 22.04.0
```

```
library(tm)
```

```
## Loading required package: NLP
```

```r
library(SnowballC)

to_another <- content_transformer(function(x, y, z) gsub(y, z,
    x))

add_pre <- function(x, pre) {
    ifelse(!is.na(x) & x != "" & nchar(x) > 0 & x != " ", return(paste(pre,
        x, sep = "")), return(x))
}

add_prefix <- content_transformer(add_pre)
```

For terms coming from the production_companies column, we have to change "Metro-Goldwyn-Mayer" to "Metro_Goldwyn_Mayer" due to the name having dashes in it. This is the only company that appears in at least 50 movies that contain "-" in its name. We added "p$" to indicate production_companies terms. From here, we got a result of 71 production companies.

```r
#### PRODUCTION COMPANIES
prod_comp <- clean_data$production_companies
docs <- Corpus(VectorSource(prod_comp))
# Remove '-' from the name since it is the separator of the
# data
docs <- tm_map(docs, to_another, "Metro-Goldwyn-Mayer", "Metro_Goldwyn_Mayer")
docs <- tm_map(docs, to_another, "no_production_companies", "")
docs <- tm_map(docs, to_another, " ", "_")
docs <- tm_map(docs, to_another, "-", " ")
docs <- tm_map(docs, stripWhitespace)
# adding prefix p$ for production_companies
docs <- tm_map(docs, add_prefix, "p$")
docs <- tm_map(docs, to_another, " ", " p$")
dtm <- DocumentTermMatrix(docs)

# get production companies in at least 50 movies
key_pc <- sort(findFreqTerms(dtm, 50))
key_pc
```

```
##  [1] "p$20th_century_fox"              "p$amblin_entertainment"
##  [3] "p$arte_france_cinéma"            "p$atresmedia"
##  [5] "p$bbc_film"                      "p$blumhouse_productions"
##  [7] "p$canal+"                        "p$canal+_españa"
##  [9] "p$cannon_group"                  "p$castle_rock_entertainment"
## [11] "p$ciné+"                         "p$cj_entertainment"
## [13] "p$cnc"                           "p$columbia_pictures"
## [15] "p$constantin_film"               "p$dentsu"
## [17] "p$dimension_films"               "p$dreamworks_pictures"
## [19] "p$dune_entertainment"            "p$europacorp"
## [21] "p$film_i_väst"                   "p$film4_productions"
## [23] "p$filmnation_entertainment"      "p$focus_features"
## [25] "p$fox_2000_pictures"             "p$fox_searchlight_pictures"
## [27] "p$france_2_cinéma"               "p$france_3_cinéma"
## [29] "p$gaumont"                       "p$hollywood_pictures"
## [31] "p$imagine_entertainment"         "p$ingenious_media"
## [33] "p$lakeshore_entertainment"       "p$lionsgate"
## [35] "p$malpaso_productions"           "p$metro_goldwyn_mayer"
## [37] "p$millennium_films"              "p$miramax"
## [39] "p$morgan_creek_productions"      "p$new_line_cinema"
## [41] "p$new_regency_pictures"          "p$orion_pictures"
## [43] "p$paramount"                     "p$participant"
## [45] "p$pathé"                         "p$polygram_filmed_entertainment"
## [47] "p$regency_enterprises"           "p$relativity_media"
## [49] "p$scott_free_productions"        "p$scott_rudin_productions"
## [51] "p$screen_gems"                   "p$silver_pictures"
## [53] "p$sony_pictures"                 "p$studiocanal"
## [55] "p$summit_entertainment"          "p$téléfilm_canada"
## [57] "p$tf1_films_production"          "p$the_weinstein_company"
## [59] "p$toei_company"                  "p$toho"
## [61] "p$touchstone_pictures"           "p$tristar_pictures"
## [63] "p$tsg_entertainment"             "p$united_artists"
## [65] "p$universal_pictures"            "p$village_roadshow_pictures"
## [67] "p$walt_disney_pictures"          "p$warner_bros._pictures"
## [69] "p$wild_bunch"                    "p$working_title_films"
## [71] "p$zdf"
```

For terms coming from genres column, we added “g$” to indicate genre terms. We decided to use all genres since there are only 19 of them. The only genre with less than 50 movies is “tv_movie”, but we decided to include it as a term.

```
#### GENRES
genres <- clean_data$genres
docs2 <- Corpus(VectorSource(genres))
docs2 <- tm_map(docs2, to_another, " ", "_")
docs2 <- tm_map(docs2, to_another, "-", " ")
docs2 <- tm_map(docs2, stripWhitespace)
# adding prefix g$ for genres
docs2 <- tm_map(docs2, add_prefix, "g$")
docs2 <- tm_map(docs2, to_another, " ", " g$")
dtm2 <- DocumentTermMatrix(docs2)

# get all genres
key_gen <- sort(findFreqTerms(dtm2, 0))
key_gen
```

```
##  [1] "g$action"          "g$adventure"       "g$animation"
##  [4] "g$comedy"          "g$crime"           "g$documentary"
##  [7] "g$drama"           "g$family"          "g$fantasy"
## [10] "g$history"         "g$horror"          "g$music"
## [13] "g$mystery"         "g$romance"         "g$science_fiction"
## [16] "g$thriller"        "g$tv_movie"        "g$war"
## [19] "g$western"
```

For terms coming from the credits column, there are a lot of names that contain a dash symbol. The names of Korean casts can be dealt with since they follow a regex pattern, as seen in the code. However, this does not cover all names. It is hard to clean this, so we decided to remove one-word names from the results. We added “c$” to indicate cast terms. From this column, we managed to get 67 terms.

```
#### CREDITS
casts <- clean_data$credits
docs3 <- Corpus(VectorSource(casts))
docs3 <- tm_map(docs3, to_another, " ", "_")
# deal with Korean names
docs3 <- tm_map(docs3, to_another, "_([[:alpha:]]+)-([[:lower:]]+)$",
    "_\\1_\\2")
docs3 <- tm_map(docs3, to_another, "_([[:alpha:]]+)-([[:lower:]]+)-",
    "_\\1_\\2-")
docs3 <- tm_map(docs3, to_another, "-", " ")
docs3 <- tm_map(docs3, stripWhitespace)
# adding prefix c$ for casts
docs3 <- tm_map(docs3, add_prefix, "c$")
docs3 <- tm_map(docs3, to_another, " ", " c$")
dtm3 <- DocumentTermMatrix(docs3)

# get all casts in at least 50 movies
key_cast <- sort(findFreqTerms(dtm3, 50))

# hard to remove '-' from two-worded names separated by '-'
# so we remove single-word names that comes from them
key_cast <- key_cast[grepl("_", key_cast)]
key_cast
```

```
##  [1] "c$alec_baldwin"        "c$alfred_molina"       "c$anthony_hopkins"
##  [4] "c$antonio_banderas"    "c$ben_kingsley"        "c$ben_stiller"
##  [7] "c$bess_flowers"        "c$bill_murray"         "c$brad_pitt"
## [10] "c$brian_cox"           "c$bruce_willis"        "c$cate_blanchett"
## [13] "c$christopher_plummer" "c$christopher_walken"  "c$clint_eastwood"
## [16] "c$danny_glover"        "c$danny_trejo"         "c$dennis_quaid"
## [19] "c$donald_sutherland"   "c$ethan_hawke"         "c$forest_whitaker"
## [22] "c$frank_welker"        "c$gene_hackman"        "c$harrison_ford"
## [25] "c$harry_dean_stanton"  "c$harvey_keitel"       "c$j.k._simmons"
## [28] "c$james_franco"        "c$joe_chrest"          "c$john_cusack"
## [31] "c$john_goodman"        "c$john_hurt"           "c$john_leguizamo"
## [34] "c$john_turturro"       "c$johnny_depp"         "c$julianne_moore"
## [37] "c$keanu_reeves"        "c$keith_david"         "c$kevin_bacon"
## [40] "c$liam_neeson"         "c$m._emmet_walsh"      "c$matt_damon"
## [43] "c$meryl_streep"        "c$michael_caine"       "c$michael_papajohn"
## [46] "c$morgan_freeman"      "c$nicolas_cage"        "c$nicole_kidman"
## [49] "c$owen_wilson"         "c$paul_giamatti"       "c$richard_jenkins"
## [52] "c$robert_de_niro"      "c$robert_downey_jr."   "c$robert_duvall"
## [55] "c$robin_williams"      "c$samuel_l._jackson"   "c$sigourney_weaver"
## [58] "c$stanley_tucci"       "c$stephen_root"        "c$stephen_tobolowsky"
## [61] "c$steve_buscemi"       "c$susan_sarandon"      "c$sylvester_stallone"
## [64] "c$thomas_rosales_jr."  "c$tom_hanks"           "c$willem_dafoe"
## [67] "c$woody_harrelson"
```

For terms coming from the keywords column, we added “k$” to indicate keyword terms. From this column, we managed to get 312 terms.

```
#### KEYWORDS
keywords <- clean_data$keywords
docs4 <- Corpus(VectorSource(keywords))
docs4 <- tm_map(docs4, to_another, " ", "_")
docs4 <- tm_map(docs4, to_another, "-", " ")
docs4 <- tm_map(docs4, stripWhitespace)
# adding prefix k$ for keywords
docs4 <- tm_map(docs4, add_prefix, "k$")
docs4 <- tm_map(docs4, to_another, " ", " k$")
dtm4 <- DocumentTermMatrix(docs4)

# get all keywords in at least 50 movies
key_keys <- sort(findFreqTerms(dtm4, 50))
key_keys
```

```
##   [1] "k$1920s"                        "k$1930s"
##   [3] "k$1940s"                        "k$1950s"
##   [5] "k$1960s"                        "k$1970s"
##   [7] "k$1980s"                        "k$1990s"
##   [9] "k$19th_century"                 "k$action_hero"
##  [11] "k$adultery"                     "k$africa"
##  [13] "k$aftercreditsstinger"          "k$airplane"
##  [15] "k$alcohol"                      "k$alcoholic"
##  [17] "k$alcoholism"                   "k$alien"
##  [19] "k$alien_invasion"               "k$amnesia"
##  [21] "k$animal"                       "k$anime"
##  [23] "k$anthropomorphism"             "k$anti_hero"
##  [25] "k$apocalyptic_future"           "k$army"
##  [27] "k$artificial_intelligence"      "k$assassin"
##  [29] "k$assassination"                "k$australia"
##  [31] "k$author"                       "k$baby"
##  [33] "k$bank_robbery"                 "k$baseball"
##  [35] "k$based_on_children’s_book"     "k$based_on_comic"
##  [37] "k$based_on_manga"               "k$based_on_novel_or_book"
##  [39] "k$based_on_play_or_musical"     "k$based_on_short_story"
##  [41] "k$based_on_true_story"          "k$based_on_video_game"
##  [43] "k$based_on_young_adult_novel"   "k$battle"
##  [45] "k$beach"                        "k$best_friend"
##  [47] "k$betrayal"                     "k$biography"
##  [49] "k$black_and_white"              "k$blackmail"
##  [51] "k$bomb"                         "k$brother"
##  [53] "k$brutality"                    "k$buddy_cop"
##  [55] "k$bullying"                     "k$california"
##  [57] "k$cancer"                       "k$car_crash"
##  [59] "k$castle"                       "k$cat"
##  [61] "k$chase"                        "k$chicago_illinois"
##  [63] "k$child_abuse"                  "k$china"
##  [65] "k$christmas"                    "k$church"
##  [67] "k$cia"                          "k$code"
##  [69] "k$college"                      "k$coming_of_age"
##  [71] "k$competition"                  "k$concert"
##  [73] "k$conspiracy"                   "k$cop"
##  [75] "k$corruption"                   "k$creature"
##  [77] "k$criminal"                     "k$cult_film"
##  [79] "k$dance"                        "k$dark_comedy"
##  [81] "k$daughter"                     "k$death"
##  [83] "k$demon"                        "k$depression"
##  [85] "k$desert"                       "k$detective"
##  [87] "k$disaster"                     "k$divorce"
##  [89] "k$doctor"                       "k$dog"
##  [91] "k$dragon"                       "k$dream"
##  [93] "k$drug_addiction"               "k$drug_dealer"
##  [95] "k$drugs"                        "k$duringcreditsstinger"
##  [97] "k$dying_and_death"              "k$dysfunctional_family"
##  [99] "k$dystopia"                     "k$england"
## [101] "k$epic"                         "k$escape"
## [103] "k$ex"                           "k$experiment"
## [105] "k$explosion"                    "k$extramarital_affair"
## [107] "k$fairy_tale"                   "k$faith"
## [109] "k$falling_in_love"              "k$family"
## [111] "k$family_relationships"         "k$father"
## [113] "k$father_daughter_relationship" "k$father_son_relationship"
## [115] "k$fbi"                          "k$female_friendship"
## [117] "k$female_protagonist"           "k$female_wrestler"
## [119] "k$fight"                        "k$film_noir"
## [121] "k$fire"                         "k$flashback"
## [123] "k$florida"                      "k$forest"
## [125] "k$found_footage"                "k$france"
## [127] "k$friends"                      "k$friendship"
## [129] "k$funeral"                      "k$future"
## [131] "k$gambling"                     "k$gang"
## [133] "k$gangster"                     "k$gay"
## [135] "k$gay_interest"                 "k$ghost"
## [137] "k$giant_monster"                "k$good_versus_evil"
## [139] "k$gore"                         "k$grief"
## [141] "k$gun"                          "k$gunfight"
## [143] "k$hallucination"                "k$haunted_house"
## [145] "k$heist"                        "k$helicopter"
## [147] "k$hero"                         "k$high_school"
## [149] "k$hitman"                       "k$holiday"
## [151] "k$hollywood"                    "k$horror"
## [153] "k$horse"                        "k$hospital"
## [155] "k$hostage"                      "k$hotel"
## [157] "k$husband_wife_relationship"    "k$in"
## [159] "k$infidelity"                   "k$investigation"
## [161] "k$island"                       "k$japan"
## [163] "k$jealousy"                     "k$journalist"
## [165] "k$jungle"                       "k$kidnapping"
## [167] "k$killer"                       "k$kung_fu"
## [169] "k$las_vegas"                    "k$lawyer"
## [171] "k$lgbt"                         "k$lgbt_interest"
## [173] "k$live_action_and_animation"    "k$london_england"
## [175] "k$los_angeles_california"       "k$loss_of_loved_one"
## [177] "k$love"                         "k$love_of_one’s_life"
## [179] "k$love_triangle"                "k$mafia"
## [181] "k$magic"                        "k$male_friendship"
## [183] "k$male_homosexuality"           "k$manhattan_new_york_city"
## [185] "k$marijuana"                    "k$marriage"
## [187] "k$martial_arts"                 "k$mental_illness"
## [189] "k$mexico"                       "k$military"
## [191] "k$money"                        "k$monster"
## [193] "k$mother_daughter_relationship" "k$mother_son_relationship"
## [195] "k$motorcycle"                   "k$movie_business"
## [197] "k$murder"                       "k$musical"
## [199] "k$musician"                     "k$nazi"
## [201] "k$neighbor"                     "k$neo"
## [203] "k$new_love"                     "k$new_york_city"
## [205] "k$nightclub"                    "k$nightmare"
## [207] "k$noir"                         "k$obsession"
## [209] "k$organized_crime"              "k$orphan"
## [211] "k$paranoia"                     "k$parent_child_relationship"
## [213] "k$paris_france"                 "k$parody"
## [215] "k$period_drama"                 "k$pets"
## [217] "k$police"                       "k$police_officer"
## [219] "k$politics"                     "k$post"
## [221] "k$pregnancy"                    "k$priest"
## [223] "k$princess"                     "k$prison"
## [225] "k$pro_wrestling"                "k$prostitute"
## [227] "k$psychological_thriller"       "k$psychopath"
## [229] "k$racism"                       "k$rape"
## [231] "k$relationship"                 "k$religion"
## [233] "k$remake"                       "k$rescue"
## [235] "k$restaurant"                   "k$revenge"
## [237] "k$rivalry"                      "k$road_trip"
## [239] "k$robbery"                      "k$robot"
## [241] "k$romance"                      "k$romantic_comedy"
## [243] "k$rural_area"                   "k$sadism"
## [245] "k$san_francisco_california"     "k$satire"
## [247] "k$school"                       "k$scientist"
## [249] "k$secret_agent"                 "k$secret_identity"
## [251] "k$seduction"                    "k$self"
## [253] "k$sequel"                       "k$serial_killer"
## [255] "k$sheriff"                      "k$ship"
## [257] "k$shootout"                     "k$showdown"
## [259] "k$sibling_relationship"         "k$silent_film"
## [261] "k$singer"                       "k$single_mother"
## [263] "k$slasher"                      "k$small_town"
## [265] "k$snow"                         "k$soldier"
## [267] "k$space"                        "k$space_travel"
## [269] "k$spacecraft"                   "k$spoof"
## [271] "k$sports"                       "k$spy"
## [273] "k$street_gang"                  "k$suicide"
## [275] "k$suicide_attempt"              "k$summer"
## [277] "k$super_power"                  "k$superhero"
## [279] "k$supernatural"                 "k$surrealism"
## [281] "k$survival"                     "k$sword_fight"
## [283] "k$teacher"                      "k$teen_movie"
## [285] "k$teenage_girl"                 "k$teenager"
## [287] "k$terrorism"                    "k$terrorist"
## [289] "k$texas"                        "k$thief"
## [291] "k$time_travel"                  "k$torture"
## [293] "k$train"                        "k$transformation"
## [295] "k$travel"                       "k$undercover"
## [297] "k$up"                           "k$usa_president"
## [299] "k$vampire"                      "k$vigilante"
## [301] "k$village"                      "k$villain"
## [303] "k$wedding"                      "k$whodunit"
## [305] "k$widow"                        "k$winter"
## [307] "k$witch"                        "k$woman_director"
## [309] "k$world_war_ii"                 "k$wrestling"
## [311] "k$writer"                       "k$zombie"
```

For terms coming from the original_language column, we added “l$” to indicate language terms. From this column, we managed to get 20 terms.

```
#### ORIGINAL LANGUAGE
og_lng <- clean_data$original_language
docs5 <- Corpus(VectorSource(og_lng))
# adding prefix l$ for language
docs5 <- tm_map(docs5, add_prefix, "l$")
docs5 <- tm_map(docs5, to_another, " ", " l$")
dtm5 <- DocumentTermMatrix(docs5)

# get all languages in at least 50 movies
key_lang <- sort(findFreqTerms(dtm5, 50))
key_lang
```

```
##  [1] "l$ar" "l$cn" "l$de" "l$en" "l$es" "l$fa" "l$fr" "l$hi" "l$it" "l$ja"
## [11] "l$ko" "l$ml" "l$pt" "l$ru" "l$sv" "l$ta" "l$te" "l$tr" "l$ur" "l$zh"
```

With all of the terms above combined, we managed to get a total of 489 terms. We used these terms as variables in our model, and include the budget_adjusted as a variable as well. Due to the large values of budgets and revenues, we decided to use log transformation on both of these variables and do modeling with them.

```
X <- cbind(dtm[, key_pc], dtm2[, key_gen], dtm3[, key_cast],
    dtm4[, key_keys], dtm5[, key_lang])
y_adj <- log(clean_data$revenue_adjusted)
```

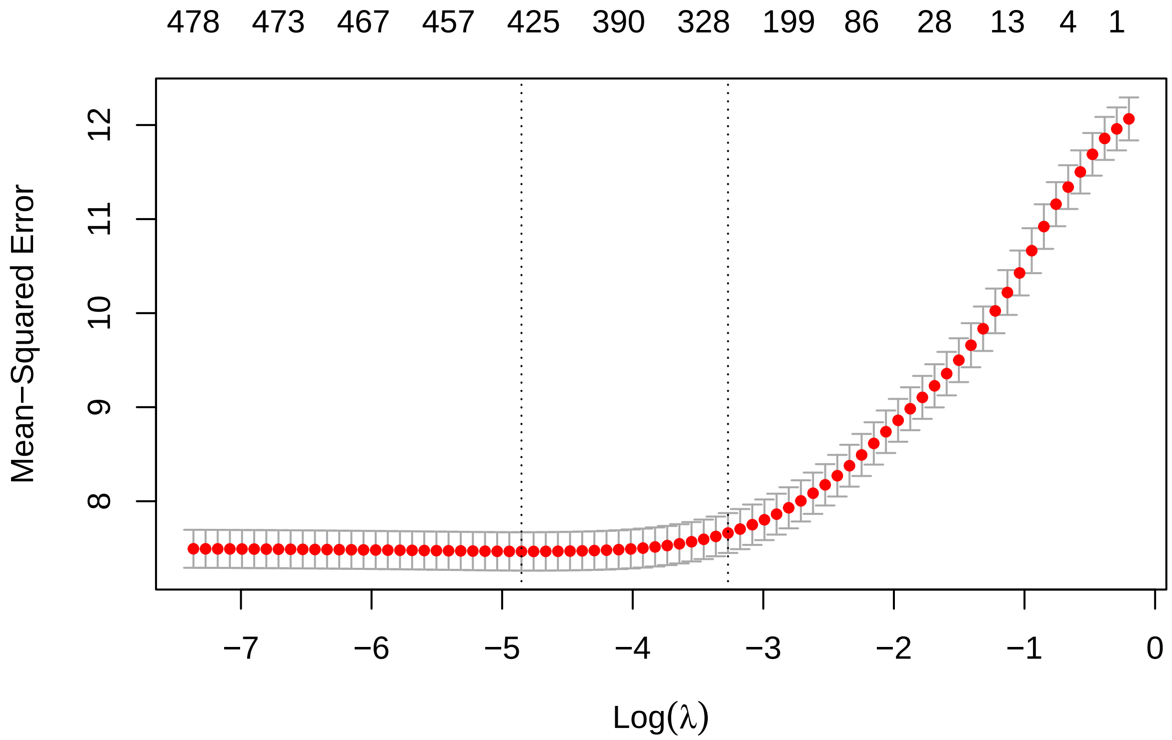
Originally, we would have liked to include the budget column in the model. However, it turns out that 5,178 rows have either zero budget or an unspecified amount. So, we decided to not include this column in the model.

```
library(glmnet)
```

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

```
## Loaded glmnet 4.1-6
```

```
set.seed(424)
model1 <- cv.glmnet(as.matrix(X), y_adj)
plot(model1, xvar = "lambda")
```



```
log(model1$lambda.min)
```

```
## [1] -4.851949
```

From the plot above, we can see that based on the cross-validation, the lowest MSE is reached when log lambda is -4.851949.

```
coef1 <- coef(model1, s = "lambda.min")
length(coef1[which(coef1 != 0), ][-1])  # -1 to exclude intercept
```

```
## [1] 429
```

```
# to see all non-zero coefficients sort(coef1[which(coef1
# != 0),1], decreasing = TRUE)
```

From 489 variables, we have 429 variables in our model. Below is the intercept of the model, the top 10 variables that positively affect revenue, and the top 10 variables that negatively affect revenue.

```
ic1 <- coef1[c("(Intercept)"), 1]
paste("The intercept is ", ic1)
```

```
## [1] "The intercept is  12.6124353194507"
```

```
coef1_sort <- sort(coef1[, 1], decreasing = TRUE)[-1]
# Top 10 variables that positively affect the revenue
head(coef1_sort, 10)
```

```
##            p$screen_gems              p$paramount     p$columbia_pictures
##                 2.691183                 2.568606                2.443534
##  p$walt_disney_pictures        p$20th_century_fox    p$touchstone_pictures
##                 2.411203                 2.385321                2.285198
##    p$universal_pictures         p$new_line_cinema p$warner_bros._pictures
##                 2.222275                 2.183599                2.139672
##        p$united_artists
##                 2.124229
```

```
# Top 10 variables that negatively affect the revenue
tail(coef1_sort, 10)
```

```
##   k$dark_comedy k$pro_wrestling            l$ur         k$grief          k$lgbt
##      -0.3957772      -0.5246407      -0.5824556      -0.5990844      -0.7805669
##            l$pt            l$fa   g$documentary     k$wrestling      g$tv_movie
##      -1.0285757      -1.1534591      -1.5529399      -2.6921577      -2.9715127
```

From the result above, we can see that the top variables with a positive correlation toward revenue are all production companies. These production companies are all big companies or subsidiaries of them. For example, Walt Disney Studios is one of the biggest animation production companies. Screen Gems and Columbia Pictures are both subsidiaries of Sony Pictures, which is a really big production company.

On the other hand, the top variables with negative correlation towards the revenue come from diverse terms. For instance, one of the genres there is 'documentary'. The number of people who are interested in watching documentary movies is probably not large. This might be why documentary movies might not get a lot of revenue. There are also several language terms (Urdu, Farsi, and Estonian). It looks like the movies with these languages are not popular and thus the revenue is low.

In order to interpret this model, we need to transform the revenue back from the log transformation. Thus, all coefficients will become exponents with base e, and these variables will become multipliers to the intercept in the model. For example, the movies with Screen Gems will generate $e^{2.691183} = 14.74911$ times more than the movies without Screen Gems, with all the other variables unchanged. If the movie does not contain any of the terms in the model, the movie is expected to have a revenue based on the intercept, which is $e^{12.61243353194507} = 300269.4$ US dollars.

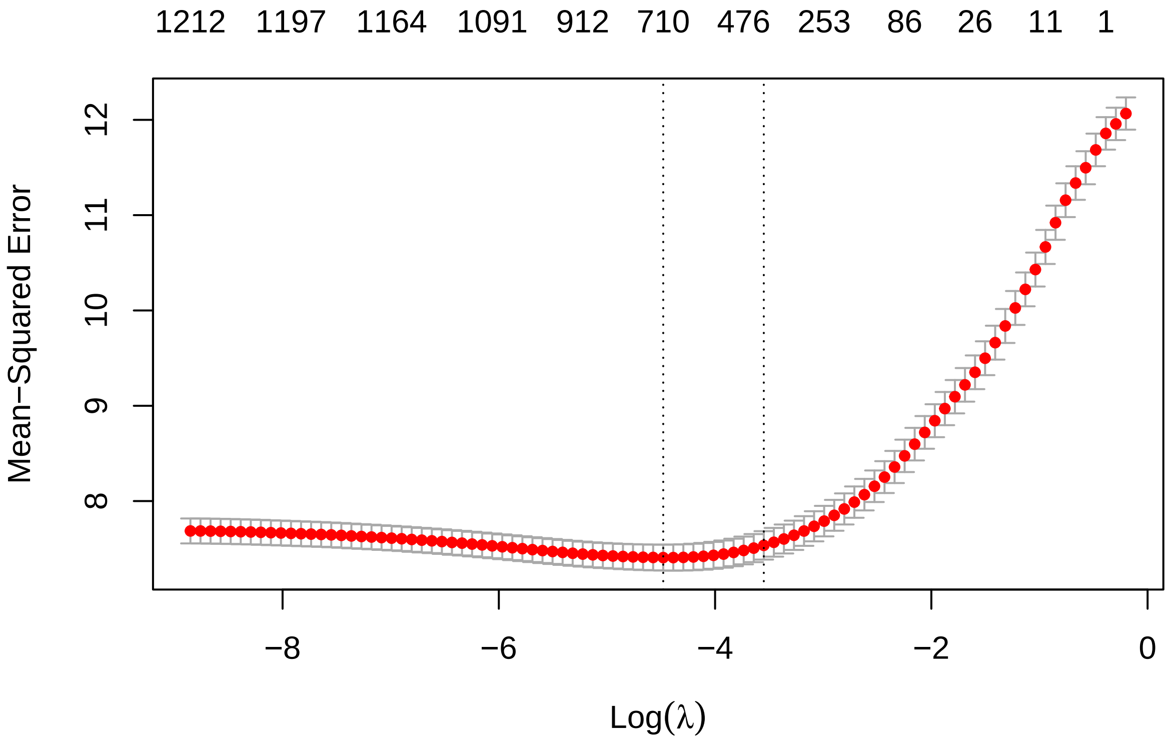
Model with interactions

To improve the model more, we want to include interactions between terms in the model. Since it is too much to interact with all of the terms that we have, we will get the top 10 terms from each column and interact with them with each other to make a pair of terms. We do not include the original_language column in the interaction since most of the movies' original language is English. So, we have 40 terms to be paired with each other, giving us 780 interaction variables.

```
# top 10 of each
key_pc2 <- names(findMostFreqTerms(dtm, 10, INDEX = rep(1, each = n))[[1]])
key_gen2 <- names(findMostFreqTerms(dtm2, 10, INDEX = rep(1,
    each = n))[[1]])
key_cast2 <- names(findMostFreqTerms(dtm3, 14, INDEX = rep(1,
    each = n))[[1]])
key_cast2 <- key_cast2[grepl("_", key_cast2)]  # 4 of them are single names
key_keys2 <- names(findMostFreqTerms(dtm4, 10, INDEX = rep(1,
    each = n))[[1]])
int_vars <- c(key_pc2, key_gen2, key_cast2, key_keys2)
inact <- c()
inact_name <- c()
for (i in 1:(length(int_vars) - 1)) {
    for (j in (i + 1):length(int_vars)) {
        a = as.matrix(X[, int_vars[i]])
        b = as.matrix(X[, int_vars[j]])
        var_name = paste(int_vars[i], ".", int_vars[j])
        v = a * b
        inact <- cbind(inact, v)
        inact_name <- c(inact_name, var_name)
    }
}

df_inact = data.frame(inact)
colnames(df_inact) <- inact_name
```

We add these 780 variables on top of the initial 489 variables in the first model, giving us 1269 variables for the second model.

```
X2 <- cbind(dtm[, key_pc], dtm2[, key_gen], dtm3[, key_cast],
    dtm4[, key_keys], dtm5[, key_lang], df_inact)
model2 <- cv.glmnet(as.matrix(X2), y_adj)
plot(model2, xvar = "lambda")
```



```
log(model2$lambda.min)
```

```
## [1] -4.479814
```

From the plot above, we can see that based on the cross-validation, the lowest MSE is reached when log lambda is -4.479814.

```
coef2 <- coef(model2, s = "lambda.min")
length(coef2[which(coef2 != 0), ][-1])  # -1 to exclude intercept
```

```
## [1] 710
```

```
# to see all the non-zero coefficients
# sort(coef2[which(coef2 != 0),1], decreasing = TRUE)
```

From 1269 variables, we have 710 variables in our model with interactions. Below is the intercept of the model, the top 10 variables that positively affect revenue, and the top 10 variables that negatively affect revenue.

```
ic2 <- coef2[c("(Intercept)"), 1]
paste("The intercept is ", ic2)
```

```
## [1] "The intercept is  12.4991157439431"
```

```r
paste("Top 10 variables that positively affect the revenue:")
```

```
## [1] "Top 10 variables that positively affect the revenue:"
```

```r
coef2_sort <- sort(coef2[, 1], decreasing = TRUE)[-1]
head(coef2_sort, 10)
```

```
##  c$robert_de_niro . k$woman_director                 p$columbia_pictures
##                             3.121845                            2.643334
##                         p$screen_gems                         p$paramount
##                             2.615547                            2.595986
##                 p$touchstone_pictures                 p$20th_century_fox
##                             2.561390                            2.514055
##                p$walt_disney_pictures             p$warner_bros._pictures
##                             2.420880                            2.343409
##                  p$universal_pictures g$science_fiction . c$robert_de_niro
##                             2.313728                            2.312593
```

```r
paste("Top 10 variables that negatively affects the revenue:")
```

```
## [1] "Top 10 variables that negatively affects the revenue:"
```

```r
tail(coef2_sort, 10)
```

```
##      p$metro_goldwyn_mayer . c$morgan_freeman
##                                     -2.266051
## p$metro_goldwyn_mayer . k$duringcreditsstinger
##                                     -2.316966
##                      c$steve_buscemi . k$sequel
##                                     -2.371802
##                          g$horror . k$biography
##                                     -2.507689
##                                    k$wrestling
##                                     -2.681321
##                p$paramount . p$20th_century_fox
##                                     -2.687947
##                 c$nicolas_cage . c$willem_dafoe
##                                     -2.692918
##         c$bruce_willis . k$based_on_true_story
##                                     -2.817947
##                                     g$tv_movie
##                                     -2.837188
##     p$20th_century_fox . p$metro_goldwyn_mayer
##                                     -3.132617
```

From the results above, we notice that the top variables have changed from the first model. The top variables with positive correlation towards the revenue are mostly still production companies, except for one. This interaction between a cast “Robert De Niro” and a keyword “Woman Director” is really interesting. From our research on the internet, turns out that Robert De Niro have only worked with 4 female directors before, producing a total of 4 movies. The most popular movie from this interaction being “The Intern”, with a revenue of nearly 200 million dollars. This model suggests that Robert De Niro should be a part of more projects with female directors.

On the other hand, there are a lot of interaction variables among the top variables with a negative correlation to the revenue. One of the interactions is the cast “Nicolas Cage” and “Willem Dafoe”. This is surprising because both actors are really popular. One of the movies that they played together is “Dog Eat Dog” which only gathers $80 of revenue based on the dataset. This suggests that the movies that they were both cast in performed relatively badly compared to movies where they are performing without each other.

Models with data separated into decades

Next, we will try to see which terms are included in the model (without interactions) when we split the data set into decades.

```r
for (df in df_list) {
    #### PRODUCTION COMPANIES
    sub_prod_comp <- df$production_companies
    sub_docs <- Corpus(VectorSource(sub_prod_comp))
    # Remove '-' from the name since it is the splitter
    # symbol of the data
    sub_docs <- tm_map(sub_docs, to_another, "Metro-Goldwyn-Mayer",
        "Metro_Goldwyn_Mayer")
    sub_docs <- tm_map(sub_docs, to_another, "no_production_companies",
        "")
    sub_docs <- tm_map(sub_docs, to_another, " ", "_")
    sub_docs <- tm_map(sub_docs, to_another, "-", " ")
    sub_docs <- tm_map(sub_docs, stripWhitespace)
    # adding prefix p$ for production_companies
    sub_docs <- tm_map(sub_docs, add_prefix, "p$")
    sub_docs <- tm_map(sub_docs, to_another, " ", " p$")
    sub_dtm <- DocumentTermMatrix(sub_docs)

    # get top 10 production companies
    sub_key_pc <- findMostFreqTerms(sub_dtm, 10, INDEX = rep(1,
        each = length(df[, 1])))
    # print(sub_key_pc)

    #### GENRES
    sub_genres <- df$genres
    sub_docs2 <- Corpus(VectorSource(sub_genres))
    sub_docs2 <- tm_map(sub_docs2, to_another, " ", "_")
    sub_docs2 <- tm_map(sub_docs2, to_another, "-", " ")
    sub_docs2 <- tm_map(sub_docs2, stripWhitespace)
    # adding prefix g$ for genres
    sub_docs2 <- tm_map(sub_docs2, add_prefix, "g$")
    sub_docs2 <- tm_map(sub_docs2, to_another, " ", " g$")
    sub_dtm2 <- DocumentTermMatrix(sub_docs2)

    # get top 10 genres
    sub_key_gen <- findMostFreqTerms(sub_dtm2, 10, INDEX = rep(1,
        each = length(df[, 1])))
    # sub_key_gen

    #### CREDITS
```

```r
sub_casts <- df$credits
sub_docs3 <- Corpus(VectorSource(sub_casts))
sub_docs3 <- tm_map(sub_docs3, to_another, " ", "_")
# deal with Korean names
sub_docs3 <- tm_map(sub_docs3, to_another, "_([[:alpha:]]+)-([[:lower:]]+)$",
    "_\\1_\\2")
sub_docs3 <- tm_map(sub_docs3, to_another, "_([[:alpha:]]+)-([[:lower:]]+)-",
    "_\\1_\\2-")
sub_docs3 <- tm_map(sub_docs3, to_another, "-", " ")
sub_docs3 <- tm_map(sub_docs3, stripWhitespace)
# adding prefix c$ for casts
sub_docs3 <- tm_map(sub_docs3, add_prefix, "c$")
sub_docs3 <- tm_map(sub_docs3, to_another, " ", " c$")
sub_dtm3 <- DocumentTermMatrix(sub_docs3)

# get top 10 casts
sub_key_cast <- findMostFreqTerms(sub_dtm3, 10, INDEX = rep(1,
    each = length(df[, 1])))

#### KEYWORDS
sub_keywords <- df$keywords
sub_docs4 <- Corpus(VectorSource(sub_keywords))
sub_docs4 <- tm_map(sub_docs4, to_another, " ", "_")
sub_docs4 <- tm_map(sub_docs4, to_another, "-", " ")
sub_docs4 <- tm_map(sub_docs4, stripWhitespace)
# adding prefix k$ for keywords
sub_docs4 <- tm_map(sub_docs4, add_prefix, "k$")
sub_docs4 <- tm_map(sub_docs4, to_another, " ", " k$")
sub_dtm4 <- DocumentTermMatrix(sub_docs4)

# get top 10 keywords
sub_key_keys <- findMostFreqTerms(sub_dtm4, 10, INDEX = rep(1,
    each = length(df[, 1])))
# sub_key_keys

#### ORIGINAL LANGUAGE
sub_og_lng <- df$original_language
sub_docs5 <- Corpus(VectorSource(sub_og_lng))
# adding prefix l$ for language
sub_docs5 <- tm_map(sub_docs5, add_prefix, "l$")
sub_docs5 <- tm_map(sub_docs5, to_another, " ", " l$")
sub_dtm5 <- DocumentTermMatrix(sub_docs5)

# get all languages in at least 50 movies
sub_key_lang <- findMostFreqTerms(sub_dtm5, 10, INDEX = rep(1,
    each = length(df[, 1])))
# sub_key_lang

sub_X <- cbind(sub_dtm[, names(sub_key_pc[[1]])], sub_dtm2[,
    names(sub_key_gen[[1]])], sub_dtm3[, names(sub_key_cast[[1]])],
    sub_dtm4[, names(sub_key_keys[[1]])], sub_dtm5[, names(sub_key_lang[[1]])])
sub_y <- log(df$revenue_adjusted)
sub_model1 <- cv.glmnet(cbind(as.matrix(sub_X), budget = log(df$budget)),
        sub_y)
    sub_coef1 <- coef(sub_model1, s = "lambda.min")
    sub_coef_sort <- sort(sub_coef1[which(sub_coef1 != 0), 1],
        decreasing = TRUE)

    print(df$decades[1])
    print(sub_coef_sort)

}
```

```
## [1] 1910
##                           (Intercept) p$jesse_l._lasky_feature_play_company
##                             17.619188                             -2.037566
## [1] 1920
##                   (Intercept)                          l$en
##                  12.872678334                   2.515355239
##                       g$music                 k$world_war_i
##                   1.572596046                   1.560638871
##      k$based_on_novel_or_book p$charles_chaplin_productions
##                   0.822351799                   0.700643326
##                c$harold_lloyd                 k$silent_film
##                   0.698528915                   0.544411996
##              p$united_artists                     g$romance
##                   0.428811161                   0.391860588
##                        k$code   p$the_vitaphone_corporation
##                   0.347126199                   0.281911891
##                      g$action                c$john_gilbert
##                   0.242734866                   0.219200601
##                   g$adventure                       g$drama
##                   0.212686060                   0.070313929
##               c$noble_johnson       p$warner_bros._pictures
##                   0.057078012                   0.051760059
##            c$nigel_de_brulier                         k$pre
##                   0.047115097                   0.008536016
##                          l$zh
##                  -5.169422687
## [1] 1930
##               (Intercept)                      l$en               g$adventure
##               13.35492875                3.82657050                0.39509310
##            c$irving_bacon  k$based_on_novel_or_book                 k$musical
##                0.39501978                0.27075923                0.23081056
##         k$black_and_white             c$clark_gable               c$ward_bond
##                0.04833436                0.03865860                0.01243344
##   p$warner_bros._pictures                   g$drama      p$rko_radio_pictures
##               -0.02852272               -0.05415548               -0.66325381
## p$first_national_pictures                      l$zh                      l$sv
##               -1.80483687               -3.43722103               -4.02346834
## [1] 1940
##               (Intercept)                      l$en                  g$family
##                15.5739766                 1.7904567                 0.6221444
## p$walt_disney_productions          c$bert_moorhouse         k$black_and_white
##                 0.3285450                 0.3108720                 0.2751412
##                   g$drama                      l$fr
##                 0.0116140                -3.1339220
## [1] 1950
##               (Intercept)                      l$en p$walt_disney_productions
##               14.01406765                2.73212189                1.83971429
##                    k$epic            c$bess_flowers                 g$romance
##                0.74312169                0.47505151                0.32310133
##                g$thriller                 k$musical  k$based_on_novel_or_book
##                0.23444109                0.22173996                0.19897367
##          c$franklyn_farnum                      l$it                      l$zh
##                0.09930969               -0.30331202               -4.32828693
## [1] 1960
##                (Intercept)                       l$el
##                13.79014960                 2.93332389
##                       l$en                       l$ja
##                 2.91435762                 2.12048806
## k$based_on_play_or_musical                       l$it
##                 1.36996240                 1.25700882
##              c$frank_baker                     k$epic
##                 1.06703617                 0.83283244
##           p$united_artists                g$adventure
##                 0.79462128                 0.78341680
##              c$paul_newman    p$warner_bros._pictures
##                 0.68694503                 0.66826751
##                  k$musical                c$john_wayne
##                 0.63191755                 0.60100324
##            k$new_york_city                   g$action
##                 0.49003242                 0.48393764
##         p$20th_century_fox   k$based_on_novel_or_book
##                 0.39508430                 0.38180402
##      p$metro_goldwyn_mayer        p$columbia_pictures
##                 0.36525624                 0.30889991
##                 g$thriller                      g$war
##                 0.26071218                 0.24500781
##                  c$al_bain             c$arthur_tovey
##                 0.20794112                 0.18202296
##                    g$drama                    g$crime
##                 0.14596997                 0.12853139
##             c$bert_stevens                  g$western
##                 0.08529632                 0.07598935
##                       l$tr                       l$es
##                -0.06728581                -0.08449426
##                     c$jean                k$cult_film
##                -0.18116967                -0.30000364
##                       l$fa                       l$sv
##                -2.03212288                -2.15777742
## [1] 1970
##               (Intercept)                      l$en                      l$fr
##               13.87015674                3.00268601                2.26708756
##                      l$it                      l$ja                      l$sv
##                2.18202972                1.94060889                1.55480931
##               p$paramount   p$warner_bros._pictures p$walt_disney_productions
##                1.27936799                1.07138399                1.04815586
##            c$arthur_tovey       p$columbia_pictures        p$20th_century_fox
##                0.92969578                0.87519520                0.75186122
##      p$universal_pictures          p$united_artists           k$new_york_city
##                0.54677529                0.53072478                0.48793989
##                 k$musical         g$science_fiction                g$thriller
##                0.43787811                0.39385770                0.38601106
##                  g$action                  g$comedy                  k$police
##                0.34800193                0.32897615                0.31015823
##           c$burt_reynolds     p$metro_goldwyn_mayer     p$malpaso_productions
##                0.21870918                0.18933499                0.08099298
##                   g$crime           c$robert_duvall               g$adventure
##                0.05604740                0.04842614                0.04752269
##                 g$romance              c$ned_beatty                      l$tr
##                0.02824088                0.02440882               -0.21166567
##                  k$sports          c$m._emmet_walsh                 g$mystery
##               -0.35207672               -0.44310853               -0.48344011
## [1] 1980
##              (Intercept)                     l$en              p$paramount
##              13.89905010               1.82778249               1.81906516
##     c$sylvester_stallone       p$20th_century_fox                     l$sv
##               1.48692788               1.46321868               1.44100156
##     p$universal_pictures                     l$ja           c$frank_welker
##               1.32452977               1.24234903               1.23232011
##  p$warner_bros._pictures    p$metro_goldwyn_mayer                 k$sequel
##               1.15589328               1.06691026               0.91995750
##       p$tristar_pictures             c$dan_aykroyd      p$columbia_pictures
##               0.85515911               0.81154316               0.77466050
##          k$new_york_city         p$orion_pictures         c$m._emmet_walsh
##               0.71814985               0.71626304               0.70287922
##                     l$it              g$adventure  k$based_on_novel_or_book
##               0.61340999               0.58225244               0.52375062
##           k$martial_arts                k$revenge             c$john_candy
##               0.50697243               0.50348480               0.44735945
##                   c$jean            c$peter_jason k$los_angeles_california
##               0.44558948               0.34885817               0.30536553
##                 g$comedy                 k$police          c$robert_loggia
##               0.28300566               0.20666617               0.15690247
##                     l$cn               g$thriller                g$fantasy
##               0.14726297               0.13083631               0.10518175
##                g$romance        g$science_fiction                  g$crime
##               0.10202353               0.09583170               0.06829004
##           p$cannon_group                     l$es                 g$horror
##              -0.04652501              -0.16066176              -0.21423347
##                  g$drama            c$dick_miller         k$woman_director
##              -0.31123069              -0.49666933              -0.53212941
##                     l$tr                     l$zh
##              -0.81733132              -1.18439981
## [1] 1990
##              (Intercept)       p$20th_century_fox              p$paramount
##             13.720757127              2.206850882              2.087479002
##      p$columbia_pictures     p$universal_pictures    p$touchstone_pictures
##              1.886018123              1.881050193              1.738760563
##       p$tristar_pictures  p$warner_bros._pictures                     l$ja
##              1.644264077              1.613253595              1.610805786
##           c$frank_welker         c$robin_williams     p$hollywood_pictures
##              1.560736571              1.511392863              1.474940512
##        p$new_line_cinema           c$bruce_willis         c$robert_de_niro
##              1.382403003              0.987984202              0.955454456
## k$based_on_novel_or_book           k$martial_arts                     l$en
##              0.889702102              0.836977454              0.828841150
##      c$samuel_l._jackson              g$adventure                p$miramax
##              0.823513621              0.785618621              0.692722691
##               g$thriller          k$new_york_city                     l$it
##              0.685003161              0.621321141              0.591272377
## k$los_angeles_california     c$thomas_rosales_jr.             c$dan_hedaya
##              0.553431907              0.489165535              0.471287020
##                     l$hi                 g$family                 g$comedy
##              0.435066306              0.398755244              0.367024631
##                g$romance                 g$action                  c$jones
##              0.345123882              0.314035654              0.157788389
##                k$revenge                 k$police                 g$horror
##              0.140494122              0.136421170              0.133583726
##                     l$ta        g$science_fiction                   c$jean
##              0.132756870              0.125371487              0.088701763
##                 k$murder                     l$de                  g$drama
##              0.027585913             -0.004287399             -0.057595978
##         k$woman_director                     l$zh
##             -0.279046763             -1.837899989
## [1] 2000
##                 (Intercept)         p$columbia_pictures
##                 12.61244355                  2.74771847
##     p$warner_bros._pictures        p$universal_pictures
##                  2.66962573                  2.60278671
##           p$new_line_cinema       p$walt_disney_pictures
##                  2.57037434                  2.33777122
##       p$dreamworks_pictures           p$20th_century_fox
##                  2.31671042                  2.29660207
##                 p$paramount                        l$ja
##                  2.23948373                  1.99134851
##                        l$hi                        l$ko
##                  1.93195968                  1.92529145
##                        l$de                        l$it
##                  1.82036494                  1.78232946
##                   p$miramax      k$duringcreditsstinger
##                  1.69680067                  1.52386840
##         c$samuel_l._jackson                        l$en
##                  1.38103387                  1.36010170
##                     c$jones               c$bruce_willis
##                  1.34930001                  1.34829750
##                    p$canal+                c$keith_david
##                  1.20485638                  1.15624653
## k$parent_child_relationship           k$loss_of_loved_one
##                  1.02933395                  0.96657352
##                        l$fr                        l$zh
##                  0.95346062                  0.94644874
##                        l$es    k$based_on_novel_or_book
##                  0.92159115                  0.88467866
##                k$friendship                   g$fantasy
##                  0.86685655                  0.77882279
##              c$willem_dafoe                    g$action
##                   0.76114084                  0.74616265
##              k$new_york_city                 g$adventure
##                   0.71088120                  0.67138589
##             c$morgan_freeman                    g$family
##                   0.55706324                  0.53629208
##                   g$thriller                   g$romance
##                   0.52535440                  0.49146181
##                         l$ru                    g$comedy
##                   0.48953322                  0.41782800
##                       c$jean            c$david_koechner
##                   0.39149510                  0.34113609
##                      c$marie                    k$murder
##                   0.33485901                  0.33380671
##                     k$sports                     g$crime
##                   0.32726149                  0.19847428
##                     g$horror            k$woman_director
##                   0.19144594                  0.05502081
##                      g$drama                   k$revenge
##                  -0.09068283                 -0.09601916
##                c$justin_long
##                  -0.10545322
## [1] 2010
##              (Intercept)  p$warner_bros._pictures      p$columbia_pictures
##             12.322133772              3.401782560              3.216685469
##       p$20th_century_fox              p$paramount   p$walt_disney_pictures
##              3.131444226              3.100950661              3.029579880
##     p$universal_pictures              p$lionsgate   k$duringcreditsstinger
##              2.881218723              2.275124953              2.134907953
##      c$samuel_l._jackson            c$liam_neeson                 k$sequel
##              1.756017270              1.669748150              1.666026750
##                     l$ja       p$relativity_media                     l$ko
##              1.480024703              1.474050135              1.403209998
## k$based_on_novel_or_book                     l$hi              k$biography
##              1.397301271              1.377900301              1.293368335
##    k$based_on_true_story                     l$zh              g$adventure
##              1.236522516              1.201010232              1.167805572
##             k$friendship             c$joe_chrest                g$fantasy
##              0.932092714              0.924104306              0.876887070
##                 p$canal+             p$studiocanal                 g$comedy
##              0.780160044              0.776022433              0.767181449
##               g$thriller              c$bill_hader                 g$action
##              0.724151839              0.663429052              0.661057856
##                 g$family                g$romance                   k$love
##              0.647299240              0.625278838              0.529420113
##                   c$jean                k$revenge                  c$smith
##              0.519376715              0.492722602              0.487087961
##                  c$jones                     l$en                  c$marie
##              0.398056765              0.384382114              0.337754379
##                     l$fr                  g$crime                     l$ml
##              0.330218603              0.289608133              0.288904415
##           c$j.k._simmons                 k$murder                  g$drama
##              0.139690495              0.129353692              0.113813158
##                     l$ru                 g$horror         k$woman_director
##              0.008290509             -0.036884321             -0.087694833
##                     l$de           c$james_franco                     l$es 
##             -0.139944734             -0.217332030             -0.363613470 
## [1] 2020
##              (Intercept)                     l$zh              p$paramount 
##             12.087304591              3.921624687              3.180876603 
##     p$universal_pictures  p$warner_bros._pictures      p$columbia_pictures 
##              2.738808140              2.673959836              2.294189751 
##                 k$sequel         p$focus_features   k$duringcreditsstinger 
##              2.110304894              1.797544324              1.697885299 
##                     l$ko                    c$joy           p$bron_studios 
##              1.693729477              1.524905682              1.523702809 
##         k$based_on_comic        c$anthony_molinari                 k$murder 
##              1.422460670              1.363106498              1.306277112 
##                 g$action              p$lionsgate                     l$ja 
##              1.289708784              1.192233090              1.143895503 
##                     l$fr              g$adventure     k$based_on_true_story 
##              1.068539163              1.047272717              0.991453298 
##                  k$anime k$based_on_novel_or_book     k$aftercreditsstinger 
##              0.953351926              0.844944935              0.831584008 
##                 p$canal+                  g$drama                g$fantasy 
##              0.803874310              0.517732182              0.454619018 
##                 g$comedy              k$superhero                   c$jean 
##              0.384640959              0.338885732              0.333984681 
##               g$thriller                 g$family          c$michelle_yeoh 
##              0.288610630              0.265129662              0.249500713 
##  p$blumhouse_productions              g$animation        p$ingenious_media 
##              0.221367373              0.198826393              0.118964231 
##                g$romance                  c$jones         k$woman_director 
##              0.118189999              0.116224343              0.027057797 
##                 g$horror                     l$de                     l$ru 
##             -0.006439627             -0.092382844             -0.193508780 
##                     l$es 
##             -0.542501300
```

1920s We see that in the 1920s, the strongest indicators that a movie is successful are 'Music', 'World War 1', 'Based on novel or book' 'Charles Chaplin Productions'. Movies about World War 1 is a particularly interesting one because it would be then that movies were being made about the war that just ended about a decade prior, and it was a popular sub-genre.

1930s We see that in the 1930s, movies with Irving Bacon were very popular. Irving Bacon appeared in 500 movies throughout his life but he was most active in the 1930s and 1940s. The model seems to suggest that he was in a lot of movies that generated high revenue.

1940s We see that in the 1940s, it seems that family movies were particularly favored in this decade. We also have our first notice of Disney as an indicator. The 1940s were a part of the Golden Age of Disney and their movies were performing well.

1950s Once again we see that Disney movies continued to be popular. Movies that told epic stories were also popular. We noticed the movie Ben-Hur which was released in 1959 as a particularly strong example of this. Adjusted for inflation, the movie has made approximately 180 million USD at the box office.

1960s We can see varying popular indicators in the 1960s. We note a few notable ones here. Western movies were popular and John Wayne who was essentially an icon of the genre was also in movies that were very successful. Japanese movies were also increasing in popularity in that decade.

1970s In the 1970s, most of the top indicators of a movie's success were the production companies producing the movies.

1980s The 1980s continued the popularity of big production companies. It was also a good decade for Sylvester Stallone as there were several Rocky movies that performed particularly well.

1990s - 2020s We see that the dominating indicator for these decades were mostly production companies.

Summary

From our project, we see that the model without interactions shows that most production companies affect the revenue positively, while some of the non-popular languages and genres affect the revenue negatively. On the other hand, in our analysis of the model with interactions, we could not find a satisfactory generalized conclusion. However, we did find a few interesting interactions which suggested some aspects that are open for moviemakers to interpret and discuss further.

From the analysis of the different decades above, we see that people's preferences mostly change over time. However, in recent times we see that big production companies have been consistently performing well at the box office. Another interesting indicator we notice is that movies that were based on a novel or book have consistently been a positive indicator even since the 1920s.