Predicting Success of a Movie - Data Cleaning

Madhusudan

11/2021

```
# Import Required Libraries

library("tidyverse")
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.1.1     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("priceR")
```

```
# Import Required Data sets

movies_df <- as_tibble(read_csv("Raw Inputs/movies_metadata.csv"))
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 45466 Columns: 24

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (14): belongs_to_collection, genres, homepage, imdb_id, original_langua...
## dbl   (7): budget, id, popularity, revenue, runtime, vote_average, vote_count
## lgl   (2): adult, video
## date  (1): release_date

##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
IMDB_movies <- as_tibble(read_csv("Raw Inputs/IMDb movies.csv"))
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
## Rows: 85855 Columns: 22
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (15): imdb_title_id, title, original_title, date_published, genre, count...
## dbl  (7): year, duration, avg_vote, votes, metascore, reviews_from_users, re...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
IMDB_rating <- as_tibble(read_csv("Raw Inputs/IMDb ratings.csv"))
```

```
## Rows: 85855 Columns: 49
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): imdb_title_id
## dbl (48): weighted_average_vote, total_votes, mean_vote, median_vote, votes_...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
lang_codes <- as_tibble(read_csv("Raw Inputs/language_codes.csv"))
```

```
## Rows: 184 Columns: 2
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): alpha2, Language
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(movies_df,5)
```

```
## # A tibble: 5 x 24
##   adult belongs_to_colle~  budget genres homepage    id imdb_id original_langua~
##   <lgl> <chr>               <dbl> <chr>  <chr>    <dbl> <chr>   <chr>
## 1 FALSE {'id': 10194, 'n~   3  e7 [{'id~ http://~   862 tt0114~ en
## 2 FALSE <NA>                6.5e7 [{'id~ <NA>      8844 tt0113~ en
## 3 FALSE {'id': 119050, '~   0     [{'id~ <NA>     15602 tt0113~ en
## 4 FALSE <NA>                1.6e7 [{'id~ <NA>     31357 tt0114~ en
## 5 FALSE {'id': 96871, 'n~   0     [{'id~ <NA>     11862 tt0113~ en
## # ... with 16 more variables: original_title <chr>, overview <chr>,
## #   popularity <dbl>, poster_path <chr>, production_companies <chr>,
## #   production_countries <chr>, release_date <date>, revenue <dbl>,
## #   runtime <dbl>, spoken_languages <chr>, status <chr>, tagline <chr>,
## #   title <chr>, video <lgl>, vote_average <dbl>, vote_count <dbl>
```

```
head(IMDB_movies,5)
```

```
## # A tibble: 5 x 22
##   imdb_title_id title original_title  year date_published genre duration country
##   <chr>         <chr> <chr>          <dbl> <chr>          <chr>    <dbl> <chr>
## 1 tt0000009     Miss~ Miss Jerry      1894 1894-10-09     Roma~       45 USA
## 2 tt0000574     The ~ The Story of ~  1906 1906-12-26     Biog~       70 Austra~
## 3 tt0001892     Den ~ Den sorte drøm  1911 1911-08-19     Drama       53 German~
## 4 tt0002101     Cleo~ Cleopatra       1912 1912-11-13     Dram~      100 USA
## 5 tt0002130     L'In~ L'Inferno       1911 1911-03-06     Adve~       68 Italy
## # ... with 14 more variables: language <chr>, director <chr>, writer <chr>,
## #   production_company <chr>, actors <chr>, description <chr>, avg_vote <dbl>,
## #   votes <dbl>, budget <chr>, usa_gross_income <chr>,
## #   worlwide_gross_income <chr>, metascore <dbl>, reviews_from_users <dbl>,
## #   reviews_from_critics <dbl>
```

```
head(IMDB_rating,5)
```

```
## # A tibble: 5 x 49
##   imdb_title_id weighted_average_vote total_votes mean_vote median_vote votes_10
##   <chr>                         <dbl>       <dbl>     <dbl>       <dbl>    <dbl>
## 1 tt0000009                       5.9         154       5.9           6       12
## 2 tt0000574                       6.1         589       6.3           6       57
## 3 tt0001892                       5.8         188       6             6        6
## 4 tt0002101                       5.2         446       5.3           5       15
## 5 tt0002130                       7          2237       6.9           7      210
## # ... with 43 more variables: votes_9 <dbl>, votes_8 <dbl>, votes_7 <dbl>,
## #   votes_6 <dbl>, votes_5 <dbl>, votes_4 <dbl>, votes_3 <dbl>, votes_2 <dbl>,
## #   votes_1 <dbl>, allgenders_0age_avg_vote <dbl>, allgenders_0age_votes <dbl>,
## #   allgenders_18age_avg_vote <dbl>, allgenders_18age_votes <dbl>,
## #   allgenders_30age_avg_vote <dbl>, allgenders_30age_votes <dbl>,
## #   allgenders_45age_avg_vote <dbl>, allgenders_45age_votes <dbl>,
## #   males_allages_avg_vote <dbl>, males_allages_votes <dbl>, ...
```

```
head(lang_codes,5)
```

```
## # A tibble: 5 x 2
##   alpha2 Language
##   <chr>  <chr>
## 1 aa     Afar
## 2 ab     Abkhazian
## 3 ae     Avestan
## 4 af     Afrikaans
## 5 ak     Akan
```

```
# List of columns to keep and drop

to_keep_columns <- c("adult","genres", "imdb_id","popularity","runtime","vote_count","production_countri
drop_columns <- c("belongs_to_collection","homepage","id","budget","poster_path","video","tagline","proc
                  "release_date","revenue","status","original_title","vote_average")
```

```r
# Helper Function to convert columns containing Dictionaries to List:

getAttribute <- function(vector) {
    vector <- as.vector(str_split(vector, regex("[\\[{'':,}\\]]"))[[1]])
    vector <- vector[!vector == "" & !vector == " " ]
    vector <- as.vector(vector[which(vector =="name")+1])
    return(toString(vector))
}
```

```r
# Helper Function to convert any currency into USD as per today's current exchange rate

convert_currency <- function(currency)
{
  #retrieves a list of currencies seen in currency
  curr_type = unique(str_sub(currency,1,4))

  for (curr in curr_type){
    #Fetches the currency Value using priceR package
    exch_rate = exchange_rate_latest(curr)
    conversion_value = as.double(exch_rate[exch_rate[1] == "USD"])[2]
    # Retrieves values in data with current currency
    sub_currency = currency[str_sub(currency,1,4)==curr]
    for (data in sub_currency)
    {
      ind = which(currency == data)
      value = as.double(str_sub(data,5))
      res = as.integer(value * conversion_value)
      currency[ind] = res
    }
  }
  return(currency)
}
```

```r
# Helper Function to create a 1/0 column for each value in a comma separated text column

create_cols <- function(x,colname,df){

    ncols <- max(stringr::str_count(x,", ")) + 1
    colm <- paste(colname,1:ncols,sep="_")

    df <- tidyr::separate(data = df, col = colname, sep = ", ", into = colm, remove = FALSE)
    unique_val_list <- as.data.frame(matrix(ncol = 1, nrow = 0))

    for(i in colm)
    {
      colnames(unique_val_list) <- i
      tmp <- as.data.frame(df[,i])
      colnames(tmp) <- i
      unique_val_list <- rbind(unique_val_list,tmp)
    }

    unique_val_list <- as.data.frame(unique(unique_val_list))
    unique_val_list <- na.omit(unique_val_list)
```

```
    for(i in 1:length(unique_val_list[,1]))
    {
        df[unique_val_list[i,1]] <- 0
    }

    for(i in 1:nrow(df))
    {
      for(j in colm)
      {
        if(!is.na(df[i,j]))
        {
            k <- as.character(df[i,j])
            df[i,k] = 1
        }
      }
    }

    df <- select(df, -colm)
    return (df)
}
```

```
# Drop irrelevant columns

movies_df <- movies_df[to_keep_columns]
movies_df
```

```
## # A tibble: 45,466 x 9
##    adult genres       imdb_id popularity runtime vote_count production_countries
##    <lgl> <chr>        <chr>        <dbl>   <dbl>      <dbl> <chr>
##  1 FALSE [{'id': 16,~ tt0114~      21.9       81       5415 [{'iso_3166_1': 'US~
##  2 FALSE [{'id': 12,~ tt0113~      17.0      104       2413 [{'iso_3166_1': 'US~
##  3 FALSE [{'id': 107~ tt0113~      11.7      101         92 [{'iso_3166_1': 'US~
##  4 FALSE [{'id': 35,~ tt0114~       3.86     127         34 [{'iso_3166_1': 'US~
##  5 FALSE [{'id': 35,~ tt0113~       8.39     106        173 [{'iso_3166_1': 'US~
##  6 FALSE [{'id': 28,~ tt0113~      17.9      170       1886 [{'iso_3166_1': 'US~
##  7 FALSE [{'id': 35,~ tt0114~       6.68     127        141 [{'iso_3166_1': 'DE~
##  8 FALSE [{'id': 28,~ tt0112~       2.56      97         45 [{'iso_3166_1': 'US~
##  9 FALSE [{'id': 28,~ tt0114~       5.23     106        174 [{'iso_3166_1': 'US~
## 10 FALSE [{'id': 12,~ tt0113~      14.7      130       1194 [{'iso_3166_1': 'GB~
## # ... with 45,456 more rows, and 2 more variables: original_language <chr>,
## #   title <chr>
```

```
# Convert key:value formats into comma separated values

movies_df$genres <- sapply(movies_df$genres,getAttribute, USE.NAMES = FALSE, simplify = "array")
movies_df$production_countries <- sapply(movies_df$production_countries,getAttribute, USE.NAMES = FALSE
movies_df
```

```
## # A tibble: 45,466 x 9
##    adult genres      imdb_id  popularity runtime vote_count production_countries
##    <lgl> <chr>       <chr>         <dbl>   <dbl>      <dbl> <chr>
##  1 FALSE Animation,~ tt01147~      21.9       81       5415 United States of Am~
```

```
##  2 FALSE Adventure,~ tt01134~      17.0      104       2413 United States of Am~
##  3 FALSE Romance, C~ tt01132~      11.7      101         92 United States of Am~
##  4 FALSE Comedy, Dr~ tt01148~       3.86     127         34 United States of Am~
##  5 FALSE Comedy      tt01130~       8.39     106        173 United States of Am~
##  6 FALSE Action, Cr~ tt01132~      17.9      170       1886 United States of Am~
##  7 FALSE Comedy, Ro~ tt01143~       6.68     127        141 Germany, United Sta~
##  8 FALSE Action, Ad~ tt01123~       2.56      97         45 United States of Am~
##  9 FALSE Action, Ad~ tt01145~       5.23     106        174 United States of Am~
## 10 FALSE Adventure,~ tt01131~      14.7      130       1194 United Kingdom, Uni~
## # ... with 45,456 more rows, and 2 more variables: original_language <chr>,
## #   title <chr>
```

```
#Replace blank values with NA

movies_df <- movies_df %>%
  mutate(genres = ifelse(genres == '', NA, genres)) %>%
  mutate(production_countries = ifelse(production_countries == '', NA, production_countries))
```

```
# Join the main movies file and IMDB movies.

movies_df <- dplyr::inner_join(movies_df,
                               select(IMDB_movies,year,imdb_title_id,director,budget,worlwide_gross_incom
                               by = c("imdb_id" = "imdb_title_id"))
movies_df <- dplyr::inner_join(movies_df,
                               select(IMDB_rating,imdb_title_id,weighted_average_vote),
                               by = c("imdb_id" = "imdb_title_id"))
movies_df <- na.omit(movies_df)
movies_df
```

```
## # A tibble: 8,994 x 14
##    adult genres      imdb_id  popularity runtime vote_count production_countries
##    <lgl> <chr>       <chr>         <dbl>   <dbl>      <dbl> <chr>
##  1 FALSE Animation,~ tt01147~      21.9       81       5415 United States of Am~
##  2 FALSE Adventure,~ tt01134~      17.0      104       2413 United States of Am~
##  3 FALSE Romance, C~ tt01132~      11.7      101         92 United States of Am~
##  4 FALSE Comedy, Dr~ tt01148~       3.86     127         34 United States of Am~
##  5 FALSE Comedy      tt01130~       8.39     106        173 United States of Am~
##  6 FALSE Action, Cr~ tt01132~      17.9      170       1886 United States of Am~
##  7 FALSE Comedy, Ro~ tt01143~       6.68     127        141 Germany, United Sta~
##  8 FALSE Action, Ad~ tt01145~       5.23     106        174 United States of Am~
##  9 FALSE Adventure,~ tt01131~      14.7      130       1194 United Kingdom, Uni~
## 10 FALSE Comedy, Dr~ tt01123~       6.32     106        199 United States of Am~
## # ... with 8,984 more rows, and 7 more variables: original_language <chr>,
## #   title <chr>, year <dbl>, director <chr>, budget <chr>,
## #   worlwide_gross_income <chr>, weighted_average_vote <dbl>
```

```
# Convert all currencies to USD

# Currency Conversion
movies_df$budget[!str_detect(movies_df$budget, "^\\$")] = convert_currency(movies_df$budget[!str_detect
```

```
## For full currency exchange rate API documentation visit:
##  https://exchangerate.host/#/#docs
##  (this message will only appear once per session)
```

```
## Daily GBP  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily EUR  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily CAD  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily FRF  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily DEM  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily AUD  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily DKK  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily JPY  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily HKD  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily RUR  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion
```

```
## Daily ITL  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion to integer range

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion to integer range

## Daily ESP  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily BEF  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily SEK  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily INR  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily IEP  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily ATS  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily NOK  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily BRL  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion
```

```
## Daily FIM  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily SGD  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily THB  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily NLG  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily CNY  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily HUF  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily CZK  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily PLN  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily KRW  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily CHF  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion
```

```
## Daily ISK  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily EGP  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily BGL  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily TWD  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily MXN  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily LTL  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily NZD  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily ARS  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily VEB  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily NGN  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion
```

```
## Daily LVL  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily ZAR  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily PKR  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily TRL  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily IDR  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily PHP  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily ILS  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion

## Daily AMD  exchange rate as at end of day 2022-03-01 GMT

## Warning in convert_currency(movies_df$budget[!str_detect(movies_df$budget, : NAs
## introduced by coercion
```

```
movies_df$worlwide_gross_income[!str_detect(movies_df$worlwide_gross_income, "^\\$")] = convert_currency

movies_df = na.omit(movies_df)

# Dollar Sign removal
movies_df$budget[str_detect(movies_df$budget, "^\\$")]= as.numeric(str_sub(movies_df$budget[str_detect(m
movies_df$worlwide_gross_income[str_detect(movies_df$worlwide_gross_income, "^\\$")]= as.numeric(str_sub

movies_df$budget = as.numeric(movies_df$budget)
movies_df$worlwide_gross_income = as.numeric(movies_df$worlwide_gross_income)

movies_df <- na.omit(movies_df)

movies_df
```

```
## # A tibble: 8,992 x 14
##    adult genres      imdb_id  popularity runtime vote_count production_countries
##    <lgl> <chr>       <chr>         <dbl>   <dbl>      <dbl> <chr>
##  1 FALSE Animation,~ tt01147~      21.9       81       5415 United States of Am~
##  2 FALSE Adventure,~ tt01134~      17.0      104       2413 United States of Am~
##  3 FALSE Romance, C~ tt01132~      11.7      101         92 United States of Am~
##  4 FALSE Comedy, Dr~ tt01148~       3.86     127         34 United States of Am~
##  5 FALSE Comedy      tt01130~       8.39     106        173 United States of Am~
##  6 FALSE Action, Cr~ tt01132~      17.9      170       1886 United States of Am~
##  7 FALSE Comedy, Ro~ tt01143~       6.68     127        141 Germany, United Sta~
##  8 FALSE Action, Ad~ tt01145~       5.23     106        174 United States of Am~
##  9 FALSE Adventure,~ tt01131~      14.7      130       1194 United Kingdom, Uni~
## 10 FALSE Comedy, Dr~ tt01123~       6.32     106        199 United States of Am~
## # ... with 8,982 more rows, and 7 more variables: original_language <chr>,
## #   title <chr>, year <dbl>, director <chr>, budget <dbl>,
## #   worlwide_gross_income <dbl>, weighted_average_vote <dbl>
```

```
# Create the Y-Variable, that is, is a movie hit or not-hit

movies_df = movies_df %>%
  mutate("hit" = ifelse(worlwide_gross_income/budget > 1.0, 1, 0))
```

```
# Create column for each production country and genre and create a sparse data base of 1 and 0

movies_df <- as.data.frame(create_cols(movies_df$production_countries, "production_countries",movies_df)
```

```
## Warning: Expected 12 pieces. Missing pieces filled with `NA` in 8991 rows [1, 2,
## 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
## Note: Using an external vector in selections is ambiguous.
## i Use `all_of(colm)` instead of `colm` to silence this message.
## i See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This message is displayed once per session.
```

```
movies_df <- as.data.frame(create_cols(movies_df$genres, "genres",movies_df))
```

```
## Warning: Expected 8 pieces. Missing pieces filled with `NA` in 8991 rows [1, 2,
## 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
colnames(movies_df)
```

```
##   [1] "adult"                    "genres"
##   [3] "imdb_id"                  "popularity"
##   [5] "runtime"                  "vote_count"
##   [7] "production_countries"     "original_language"
##   [9] "title"                    "year"
##  [11] "director"                 "budget"
##  [13] "worlwide_gross_income"    "weighted_average_vote"
##  [15] "hit"                      "United States of America"
##  [17] "Germany"                  "United Kingdom"
##  [19] "France"                   "Italy"
```

```
##  [21] "Australia"                "Belgium"
##  [23] "Canada"                   "Iran"
##  [25] "Netherlands"              "Hong Kong"
##  [27] "Japan"                    "Austria"
##  [29] "New Zealand"              "Mexico"
##  [31] "Taiwan"                   "Peru"
##  [33] "China"                    "South Africa"
##  [35] "Denmark"                  "Spain"
##  [37] "Serbia"                   "Sweden"
##  [39] "Czech Republic"           "Ireland"
##  [41] "Trinidad and Tobago"      "Russia"
##  [43] "India"                    "Brazil"
##  [45] "Aruba"                    "Israel"
##  [47] "Luxembourg"               "Argentina"
##  [49] "Ecuador"                  "Bahamas"
##  [51] "Malaysia"                 "Switzerland"
##  [53] "Bulgaria"                 "Thailand"
##  [55] "Namibia"                  "South Korea"
##  [57] "Norway"                   "Finland"
##  [59] "Afghanistan"              "Iceland"
##  [61] "Romania"                  "Soviet Union"
##  [63] "Hungary"                  "Chile"
##  [65] "Bhutan"                   "Poland"
##  [67] "Palestinian Territory"    "Uruguay"
##  [69] "Turkey"                   "Morocco"
##  [71] "Algeria"                  "Singapore"
##  [73] "Mongolia"                 "Bosnia and Herzegovina"
##  [75] "Mali"                     "Lebanon"
##  [77] "Kazakhstan"               "Greece"
##  [79] "United Arab Emirates"     "Indonesia"
##  [81] "Egypt"                    "Slovenia"
##  [83] "Macedonia"                "Estonia"
##  [85] "Portugal"                 "Mauritania"
##  [87] "Cyprus"                   "Bangladesh"
##  [89] "Vietnam"                  "Lithuania"
##  [91] "Jordan"                   "Nigeria"
##  [93] "Philippines"              "Venezuela"
##  [95] "Pakistan"                 "Burkina Faso"
##  [97] "Latvia"                   "Cuba"
##  [99] "Malta"                    "Qatar"
## [101] "Samoa"                    "Ukraine"
## [103] "Colombia"                 "Cambodia"
## [105] "Panama"                   "Georgia"
## [107] "Dominican Republic"       "Azerbaijan"
## [109] "Armenia"                  "Botswana"
## [111] "Croatia"                  "Costa Rica"
## [113] "Ghana"                    "Tunisia"
## [115] "Rwanda"                   "Angola"
## [117] "Monaco"                   "Puerto Rico"
## [119] " \"Lao People"            "Slovakia"
## [121] "Gibraltar"                "Liechtenstein"
## [123] "Chad"                     "Iraq"
## [125] "Serbia and Montenegro"    "Paraguay"
## [127] "Animation"                "Adventure"
```

```
## [129] "Romance"                 "Comedy"
## [131] "Action"                  "History"
## [133] "Drama"                   "Crime"
## [135] "Fantasy"                 "Science Fiction"
## [137] "Music"                   "Horror"
## [139] "Family"                  "Mystery"
## [141] "Thriller"                "Western"
## [143] "War"                     "Documentary"
## [145] "TV Movie"                "Foreign"
```

```
# Convert coded form of languages to full name of the language

movies_df <- dplyr::left_join(movies_df,lang_codes,
                              by = c("original_language" = "alpha2"),
                             keep = FALSE)
```

```
# Create the final output, to be used for visualization and modelling

movies_df <- relocate(movies_df, hit, .after = last_col())

write.csv(movies_df,"cleaned_movies_database.csv", row.names = FALSE)
```