Movie Analysis Report

Malay Raj

2023-05-27

## MOVIES DATASET ANALYSIS:

Movies Dataset (made available by Rounak Banik under a CC0 license): Metadata about 45,000 movies, with data points including cast, crew, plot keywords, budget, revenue, ratings, release dates, languages, production companies, and more.

## Loading Required Packages.

library(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.2.3

## Warning: package 'ggplot2' was built under R version 4.2.3

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.0 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.1.8  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the ]8;;http://conflicted.r-lib.org/conflicted package]8;; to force all conflicts to become errors

library(lubridate)  
library(ggplot2)  
library(plyr)

## ------------------------------------------------------------------------------  
## You have loaded plyr after dplyr - this is likely to cause problems.  
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:  
## library(plyr); library(dplyr)  
## ------------------------------------------------------------------------------  
##   
## Attaching package: 'plyr'  
##   
## The following objects are masked from 'package:dplyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize  
##   
## The following object is masked from 'package:purrr':  
##   
## compact

library(skimr)

## Warning: package 'skimr' was built under R version 4.2.3

library(readxl)

## Displaying my current working directory.

getwd()

## [1] "C:/Movies Analysis"

## Collecting the data.

movies = read\_excel("C:/Movies Analysis/movies\_data.xlsx")

## Warning: Expecting numeric in A1400 / R1400C1: got a date

## Warning: Expecting numeric in A3200 / R3200C1: got a date

## Warning: Expecting numeric in A4799 / R4799C1: got a date

## Warning: Expecting numeric in A7745 / R7745C1: got a date

## Warning: Expecting numeric in A8100 / R8100C1: got a date

## Warning: Expecting numeric in A8858 / R8858C1: got a date

## Warning: Expecting numeric in A9743 / R9743C1: got a date

## Warning: Expecting numeric in A10790 / R10790C1: got a date

## Warning: Expecting numeric in A11366 / R11366C1: got a date

## Warning: Expecting numeric in A12308 / R12308C1: got a date

## Warning: Expecting numeric in A13013 / R13013C1: got a date

## Warning: Expecting numeric in A13072 / R13072C1: got a date

## Warning: Expecting numeric in A16110 / R16110C1: got a date

## Warning: Expecting numeric in A20553 / R20553C1: got a date

## Warning: Expecting numeric in A20933 / R20933C1: got a date

## Warning: Expecting numeric in A22315 / R22315C1: got a date

## Warning: Expecting numeric in A26310 / R26310C1: got a date

## Warning: Expecting numeric in A32149 / R32149C1: got a date

## Warning: Expecting numeric in A33255 / R33255C1: got a date

## Warning: Expecting numeric in A39526 / R39526C1: got a date

## Warning: Expecting numeric in A41909 / R41909C1: got a date

## Warning: Expecting numeric in A43117 / R43117C1: got a date

## Warning: Expecting numeric in A44162 / R44162C1: got a date

## Warning: Expecting numeric in A44382 / R44382C1: got a date

## Warning: Expecting numeric in A44907 / R44907C1: got a date

## New names:  
## • `genres` -> `genres...4`  
## • `genres` -> `genres...5`

## Displaying the first six rows of the dataset.

head(movies)

## # A tibble: 6 × 17  
## id title budget genre…¹ genre…² imdb\_id origi…³ popul…⁴ produ…⁵ relea…⁶  
## <dbl> <chr> <dbl> <chr> <chr> <chr> <chr> <dbl> <chr> <chr>   
## 1 2 Ariel 0 [{'id'… Drama tt0094… fi 3.86 [{'iso… 32437.0  
## 2 3 Shadows … 0 [{'id'… Drama tt0092… fi 2.29 [{'iso… 31701.0  
## 3 5 Four Roo… 4 e6 [{'id'… Crime tt0113… en 9.03 [{'iso… 35042.0  
## 4 6 Judgment… 0 [{'id'… Actio tt0107… en 5.54 [{'iso… 34257.0  
## 5 11 Star Wars 1.1e7 [{'id'… Adven tt0076… en 42.1 [{'iso… 28270.0  
## 6 12 Finding … 9.4e7 [{'id'… Anima tt0266… en 25.5 [{'iso… 37771.0  
## # … with 7 more variables: release\_year <chr>, countries <chr>, revenue <dbl>,  
## # runtime <dbl>, status <chr>, vote\_average <dbl>, vote\_count <dbl>, and  
## # abbreviated variable names ¹​genres...4, ²​genres...5, ³​original\_language,  
## # ⁴​popularity, ⁵​production\_countries, ⁶​release\_date

## Checking the number of rows and dimension of the dataset.

nrow(movies)

## [1] 45460

dim(movies)

## [1] 45460 17

## Renaming few columns.

movies=plyr::rename(movies, replace=c( "genres...5" = "Genres",  
 "title"="movies\_title",  
 "id"="movies\_id"))

## Viewing the new column names.

colnames(movies)

## [1] "movies\_id" "movies\_title" "budget"   
## [4] "genres...4" "Genres" "imdb\_id"   
## [7] "original\_language" "popularity" "production\_countries"  
## [10] "release\_date" "release\_year" "countries"   
## [13] "revenue" "runtime" "status"   
## [16] "vote\_average" "vote\_count"

## Inspecting the dataframe and look for inconguencies.

str(movies)

## tibble [45,460 × 17] (S3: tbl\_df/tbl/data.frame)  
## $ movies\_id : num [1:45460] 2 3 5 6 11 12 13 14 15 16 ...  
## $ movies\_title : chr [1:45460] "Ariel" "Shadows in Paradise" "Four Rooms" "Judgment Night" ...  
## $ budget : num [1:45460] 0.0 0.0 4.0e+06 0.0 1.1e+07 ...  
## $ genres...4 : chr [1:45460] "[{'id': 18, 'name': 'Drama'}, {'id': 80, 'name': 'Crime'}]" "[{'id': 18, 'name': 'Drama'}, {'id': 35, 'name': 'Comedy'}]" "[{'id': 80, 'name': 'Crime'}, {'id': 35, 'name': 'Comedy'}]" "[{'id': 28, 'name': 'Action'}, {'id': 53, 'name': 'Thriller'}, {'id': 80, 'name': 'Crime'}]" ...  
## $ Genres : chr [1:45460] "Drama" "Drama" "Crime" "Actio" ...  
## $ imdb\_id : chr [1:45460] "tt0094675" "tt0092149" "tt0113101" "tt0107286" ...  
## $ original\_language : chr [1:45460] "fi" "fi" "en" "en" ...  
## $ popularity : num [1:45460] 3.86 2.29 9.03 5.54 42.15 ...  
## $ production\_countries: chr [1:45460] "[{'iso\_3166\_1': 'FI', 'name': 'Finland'}]" "[{'iso\_3166\_1': 'FI', 'name': 'Finland'}]" "[{'iso\_3166\_1': 'US', 'name': 'United States of America'}]" "[{'iso\_3166\_1': 'JP', 'name': 'Japan'}, {'iso\_3166\_1': 'US', 'name': 'United States of America'}]" ...  
## $ release\_date : chr [1:45460] "32437.0" "31701.0" "35042.0" "34257.0" ...  
## $ release\_year : chr [1:45460] "1988" "1986" "1995" "1993" ...  
## $ countries : chr [1:45460] "FI" "FI" "US" "JP" ...  
## $ revenue : num [1:45460] 0.00 0.00 4.30e+06 1.21e+07 7.75e+08 ...  
## $ runtime : num [1:45460] 69 76 98 110 121 100 142 122 119 140 ...  
## $ status : chr [1:45460] "Released" "Released" "Released" "Released" ...  
## $ vote\_average : num [1:45460] 7.1 7.1 6.5 6.4 8.1 7.6 8.2 7.9 8 7.7 ...  
## $ vote\_count : num [1:45460] 44 35 539 79 6778 ...

# Convert release\_year to numeric datatype so that they can stack correctly.

movies$release\_year = as.numeric(movies$release\_year)

## Inspecting the dataframe after changing the datatypes.

str(movies)

## tibble [45,460 × 17] (S3: tbl\_df/tbl/data.frame)  
## $ movies\_id : num [1:45460] 2 3 5 6 11 12 13 14 15 16 ...  
## $ movies\_title : chr [1:45460] "Ariel" "Shadows in Paradise" "Four Rooms" "Judgment Night" ...  
## $ budget : num [1:45460] 0.0 0.0 4.0e+06 0.0 1.1e+07 ...  
## $ genres...4 : chr [1:45460] "[{'id': 18, 'name': 'Drama'}, {'id': 80, 'name': 'Crime'}]" "[{'id': 18, 'name': 'Drama'}, {'id': 35, 'name': 'Comedy'}]" "[{'id': 80, 'name': 'Crime'}, {'id': 35, 'name': 'Comedy'}]" "[{'id': 28, 'name': 'Action'}, {'id': 53, 'name': 'Thriller'}, {'id': 80, 'name': 'Crime'}]" ...  
## $ Genres : chr [1:45460] "Drama" "Drama" "Crime" "Actio" ...  
## $ imdb\_id : chr [1:45460] "tt0094675" "tt0092149" "tt0113101" "tt0107286" ...  
## $ original\_language : chr [1:45460] "fi" "fi" "en" "en" ...  
## $ popularity : num [1:45460] 3.86 2.29 9.03 5.54 42.15 ...  
## $ production\_countries: chr [1:45460] "[{'iso\_3166\_1': 'FI', 'name': 'Finland'}]" "[{'iso\_3166\_1': 'FI', 'name': 'Finland'}]" "[{'iso\_3166\_1': 'US', 'name': 'United States of America'}]" "[{'iso\_3166\_1': 'JP', 'name': 'Japan'}, {'iso\_3166\_1': 'US', 'name': 'United States of America'}]" ...  
## $ release\_date : chr [1:45460] "32437.0" "31701.0" "35042.0" "34257.0" ...  
## $ release\_year : num [1:45460] 1988 1986 1995 1993 1977 ...  
## $ countries : chr [1:45460] "FI" "FI" "US" "JP" ...  
## $ revenue : num [1:45460] 0.00 0.00 4.30e+06 1.21e+07 7.75e+08 ...  
## $ runtime : num [1:45460] 69 76 98 110 121 100 142 122 119 140 ...  
## $ status : chr [1:45460] "Released" "Released" "Released" "Released" ...  
## $ vote\_average : num [1:45460] 7.1 7.1 6.5 6.4 8.1 7.6 8.2 7.9 8 7.7 ...  
## $ vote\_count : num [1:45460] 44 35 539 79 6778 ...

## Removing few columns because we don’t need it in our analysis.

movies = subset(movies, select = -c(genres...4, production\_countries, release\_date))

## Viewing all the column left after removing the columns.

colnames(movies)

## [1] "movies\_id" "movies\_title" "budget"   
## [4] "Genres" "imdb\_id" "original\_language"  
## [7] "popularity" "release\_year" "countries"   
## [10] "revenue" "runtime" "status"   
## [13] "vote\_average" "vote\_count"

## Checking for null values in our dataset.

sum(is.null(movies))

## [1] 0

## Checking for any duplicate values in our dataset.

anyDuplicated(movies)

## [1] 2298

Removing the duplicate values from our dataset.

movies1 <- unique(movies)

Now checking for the duplicate values again.

anyDuplicated(movies1)

## [1] 0

## Seeing how many observations fall under each of these columns.

table(movies$Genres)

##   
## 'Mys 'Scie : 'Fa : 'Fo : 'Mu : 'Ro : 'TV : 'Wa Actio Adven Anima   
## 554 647 524 118 487 1191 389 379 4488 1514 1123   
## Comed Crime Docum Drama Fanta Histo Horro Others Thril Weste   
## 8820 1685 3415 11966 704 279 2619 2442 1665 451

table(movies$countries)

##   
## AE AF AL AM AO AR AT AU AW AZ BA   
## 13 4 3 7 2 211 151 506 4 1 22   
## BD BE BF BG BM BN BO BR BS BT BW   
## 2 299 8 36 1 1 8 262 4 3 2   
## BY CA CD CH CI CL CM CN CO CR CU   
## 5 1498 3 99 2 50 4 300 19 4 16   
## CY CZ DE DK DO DZ EC EE EG ES ET   
## 2 163 1418 297 6 8 5 46 19 601 3   
## FI FR GB GE GH GR GT HK HR HU ID   
## 324 2705 3070 21 2 131 1 468 31 119 28   
## IE IL IN IQ IR IS IT JM JO JP KG   
## 135 96 783 2 92 41 1471 4 3 1493 5   
## KH KP KR KZ LA LB LI LK LR LT LU   
## 6 1 457 12 1 6 1 3 2 16 27   
## LV LY MA MC ME MK ML MM MN MQ MR   
## 19 3 13 1 1 7 1 1 3 1 3   
## MT MX MY NA NG NI NL NO NP NZ Others   
## 2 236 6 1 4 1 226 124 2 94 6282   
## PA PE PG PH PK PL PR PS PT PY QA   
## 3 15 1 70 14 245 4 7 74 1 8   
## RO RS RU RW SA SE SG SI SK SN SU   
## 87 65 800 2 1 396 17 15 7 10 17   
## SV SY TD TH TJ TN TR TT TW TZ UA   
## 1 1 2 90 2 3 134 2 88 1 33   
## UG UM US UY UZ VE VN WS XC XG YU   
## 2 1 18425 9 4 11 8 1 3 5 4   
## ZA   
## 65

table(movies$original\_language)

##   
## ab af am ar ay bg bm bn bo bs ca cn cs   
## 10 2 2 39 1 10 3 29 2 14 12 313 130   
## cy da de el en eo es et eu fa fi fr fy   
## 1 225 1080 113 32267 1 994 24 3 101 297 2438 1   
## gl he hi hr hu hy id is it iu ja jv ka   
## 1 67 508 29 100 1 20 24 1529 2 1349 1 18   
## kk kn ko ku ky la lb lo lt lv mk ml mn   
## 3 3 444 3 3 1 1 2 9 18 5 36 2   
## mr ms mt nb ne nl no ot pa pl ps pt qu   
## 25 5 1 6 2 248 106 11 2 219 2 316 1   
## ro ru rw sh si sk sl sm sq sr sv ta te   
## 57 826 1 5 1 9 17 1 5 63 384 78 45   
## tg th tl tr uk ur uz vi wo xx zh zu   
## 1 76 23 150 16 8 1 10 5 33 409 1

table(movies$release\_year)

##   
## 0 1874 1878 1883 1887 1888 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899   
## 84 1 1 1 1 2 5 6 3 1 13 7 14 8 13 5   
## 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912 1913 1914 1915   
## 14 10 2 18 8 5 7 7 6 10 7 6 14 13 19 29   
## 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931   
## 32 17 16 23 29 34 34 23 39 40 50 41 66 76 81 100   
## 1932 1933 1934 1935 1936 1937 1938 1939 1940 1941 1942 1943 1944 1945 1946 1947   
## 136 147 141 144 152 155 127 134 152 147 146 145 142 147 128 156   
## 1948 1949 1950 1951 1952 1953 1954 1955 1956 1957 1958 1959 1960 1961 1962 1963   
## 157 174 184 196 194 200 191 209 218 244 220 224 218 203 229 233   
## 1964 1965 1966 1967 1968 1969 1970 1971 1972 1973 1974 1975 1976 1977 1978 1979   
## 254 247 306 289 338 305 351 378 381 356 348 332 333 334 321 338   
## 1980 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995   
## 361 360 368 353 362 368 391 462 467 439 427 426 453 489 544 599   
## 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011   
## 633 661 722 723 789 865 905 882 992 1125 1270 1320 1473 1586 1501 1667   
## 2012 2013 2014 2015 2016 2017 2018 2020   
## 1722 1889 1974 1905 1604 532 5 1

## CONDUCTNG DESCRIPTIVE ANALYSIS:

Descriptive analysis on few columns.

summary(movies1$budget)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0 0 0 4226417 0 380000000

summary(movies1$revenue)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000e+00 0.000e+00 0.000e+00 1.121e+07 0.000e+00 2.788e+09

summary(movies1$vote\_average)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000 5.000 6.000 5.618 6.800 10.000

summary(movies1$vote\_count)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.0 3.0 10.0 109.9 34.0 14075.0

Comparing vote\_count, vote\_average with different Genres.

aggregate(movies1$vote\_count ~ movies1$Genres, FUN = mean)

## movies1$Genres movies1$vote\_count  
## 1 'Mys 99.272563  
## 2 'Scie 279.471406  
## 3 : 'Fa 150.053435  
## 4 : 'Fo 6.576271  
## 5 : 'Mu 32.611910  
## 6 : 'Ro 76.774139  
## 7 : 'TV 17.020566  
## 8 : 'Wa 62.704485  
## 9 Actio 235.051271  
## 10 Adven 404.205552  
## 11 Anima 215.309884  
## 12 Comed 83.434955  
## 13 Crime 119.746437  
## 14 Docum 12.195079  
## 15 Drama 79.589214  
## 16 Fanta 238.691323  
## 17 Histo 72.032258  
## 18 Horro 102.788851  
## 19 Others 2.944308  
## 20 Thril 129.337342  
## 21 Weste 53.824834

aggregate(movies1$vote\_count ~ movies1$Genres, FUN = median)

## movies1$Genres movies1$vote\_count  
## 1 'Mys 10  
## 2 'Scie 13  
## 3 : 'Fa 9  
## 4 : 'Fo 5  
## 5 : 'Mu 5  
## 6 : 'Ro 7  
## 7 : 'TV 7  
## 8 : 'Wa 7  
## 9 Actio 19  
## 10 Adven 23  
## 11 Anima 18  
## 12 Comed 12  
## 13 Crime 13  
## 14 Docum 4  
## 15 Drama 9  
## 16 Fanta 21  
## 17 Histo 7  
## 18 Horro 18  
## 19 Others 1  
## 20 Thril 13  
## 21 Weste 4

aggregate(movies1$vote\_count ~ movies1$Genres, FUN = max)

## movies1$Genres movies1$vote\_count  
## 1 'Mys 6023  
## 2 'Scie 12000  
## 3 : 'Fa 6141  
## 4 : 'Fo 31  
## 5 : 'Mu 1092  
## 6 : 'Ro 4215  
## 7 : 'TV 484  
## 8 : 'Wa 4600  
## 9 Actio 14075  
## 10 Adven 11187  
## 11 Anima 7048  
## 12 Comed 8147  
## 13 Crime 6768  
## 14 Docum 520  
## 15 Drama 12269  
## 16 Fanta 5398  
## 17 Histo 5895  
## 18 Horro 4564  
## 19 Others 403  
## 20 Thril 8670  
## 21 Weste 6558

aggregate(movies1$vote\_count ~ movies1$Genres, FUN = min)

## movies1$Genres movies1$vote\_count  
## 1 'Mys 0  
## 2 'Scie 0  
## 3 : 'Fa 0  
## 4 : 'Fo 0  
## 5 : 'Mu 0  
## 6 : 'Ro 0  
## 7 : 'TV 0  
## 8 : 'Wa 0  
## 9 Actio 0  
## 10 Adven 0  
## 11 Anima 0  
## 12 Comed 0  
## 13 Crime 0  
## 14 Docum 0  
## 15 Drama 0  
## 16 Fanta 0  
## 17 Histo 0  
## 18 Horro 0  
## 19 Others 0  
## 20 Thril 0  
## 21 Weste 0

aggregate(movies1$vote\_average ~ movies1$Genres, FUN = mean)

## movies1$Genres movies1$vote\_average  
## 1 'Mys 5.641877  
## 2 'Scie 5.211437  
## 3 : 'Fa 5.589313  
## 4 : 'Fo 5.781356  
## 5 : 'Mu 5.828337  
## 6 : 'Ro 5.615449  
## 7 : 'TV 5.088432  
## 8 : 'Wa 5.928760  
## 9 Actio 5.582189  
## 10 Adven 5.745605  
## 11 Anima 6.305877  
## 12 Comed 5.685982  
## 13 Crime 5.863717  
## 14 Docum 5.805624  
## 15 Drama 5.881129  
## 16 Fanta 5.653201  
## 17 Histo 5.787814  
## 18 Horro 5.107980  
## 19 Others 4.239435  
## 20 Thril 5.397835  
## 21 Weste 4.817738

aggregate(movies1$vote\_average ~ movies1$Genres, FUN = median)

## movies1$Genres movies1$vote\_average  
## 1 'Mys 6.00  
## 2 'Scie 5.50  
## 3 : 'Fa 6.00  
## 4 : 'Fo 6.15  
## 5 : 'Mu 6.40  
## 6 : 'Ro 6.00  
## 7 : 'TV 5.50  
## 8 : 'Wa 6.30  
## 9 Actio 5.90  
## 10 Adven 6.00  
## 11 Anima 6.50  
## 12 Comed 6.00  
## 13 Crime 6.20  
## 14 Docum 6.60  
## 15 Drama 6.20  
## 16 Fanta 5.90  
## 17 Histo 6.30  
## 18 Horro 5.30  
## 19 Others 5.00  
## 20 Thril 5.70  
## 21 Weste 5.50

aggregate(movies1$vote\_average ~ movies1$Genres, FUN = max)

## movies1$Genres movies1$vote\_average  
## 1 'Mys 10.0  
## 2 'Scie 10.0  
## 3 : 'Fa 9.5  
## 4 : 'Fo 9.0  
## 5 : 'Mu 10.0  
## 6 : 'Ro 10.0  
## 7 : 'TV 10.0  
## 8 : 'Wa 10.0  
## 9 Actio 10.0  
## 10 Adven 10.0  
## 11 Anima 10.0  
## 12 Comed 10.0  
## 13 Crime 10.0  
## 14 Docum 10.0  
## 15 Drama 10.0  
## 16 Fanta 10.0  
## 17 Histo 10.0  
## 18 Horro 10.0  
## 19 Others 10.0  
## 20 Thril 10.0  
## 21 Weste 10.0

aggregate(movies1$vote\_average ~ movies1$Genres, FUN = min)

## movies1$Genres movies1$vote\_average  
## 1 'Mys 0  
## 2 'Scie 0  
## 3 : 'Fa 0  
## 4 : 'Fo 0  
## 5 : 'Mu 0  
## 6 : 'Ro 0  
## 7 : 'TV 0  
## 8 : 'Wa 0  
## 9 Actio 0  
## 10 Adven 0  
## 11 Anima 0  
## 12 Comed 0  
## 13 Crime 0  
## 14 Docum 0  
## 15 Drama 0  
## 16 Fanta 0  
## 17 Histo 0  
## 18 Horro 0  
## 19 Others 0  
## 20 Thril 0  
## 21 Weste 0

## SUMMARY STATISTICS.

skimr::skim\_without\_charts(movies1)

Data summary

|  |  |
| --- | --- |
| Name | movies1 |
| Number of rows | 45443 |
| Number of columns | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 6 |
| numeric | 8 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| --- | --- | --- | --- | --- | --- | --- | --- |
| movies\_title | 0 | 1 | 1 | 105 | 0 | 42275 | 0 |
| Genres | 0 | 1 | 4 | 6 | 0 | 21 | 0 |
| imdb\_id | 0 | 1 | 3 | 9 | 0 | 45414 | 0 |
| original\_language | 0 | 1 | 2 | 2 | 0 | 90 | 0 |
| countries | 0 | 1 | 2 | 6 | 0 | 144 | 0 |
| status | 0 | 1 | 6 | 15 | 0 | 7 | 0 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| movies\_id | 0 | 1.00 | 108356.29 | 112471.90 | 2 | 26449.50 | 59993.00 | 157328.00 | 4.691720e+05 |
| budget | 0 | 1.00 | 4226416.53 | 17427745.75 | 0 | 0.00 | 0.00 | 0.00 | 3.800000e+08 |
| popularity | 0 | 1.00 | 2.92 | 6.01 | 0 | 0.39 | 1.13 | 3.68 | 5.474900e+02 |
| release\_year | 0 | 1.00 | 1988.20 | 88.87 | 0 | 1978.00 | 2001.00 | 2010.00 | 2.020000e+03 |
| revenue | 0 | 1.00 | 11213501.08 | 64343920.42 | 0 | 0.00 | 0.00 | 0.00 | 2.787965e+09 |
| runtime | 257 | 0.99 | 94.13 | 38.41 | 0 | 85.00 | 95.00 | 107.00 | 1.256000e+03 |
| vote\_average | 0 | 1.00 | 5.62 | 1.92 | 0 | 5.00 | 6.00 | 6.80 | 1.000000e+01 |
| vote\_count | 0 | 1.00 | 109.92 | 491.40 | 0 | 3.00 | 10.00 | 34.00 | 1.407500e+04 |

As we can see there are few missing values in runtime column.

movies1 <- movies1 %>% drop\_na(runtime)

Again using the same code to check if the missing values are dropped or not.

skimr::skim\_without\_charts(movies1)

Data summary

|  |  |
| --- | --- |
| Name | movies1 |
| Number of rows | 45186 |
| Number of columns | 14 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| character | 6 |
| numeric | 8 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: character**

| skim\_variable | n\_missing | complete\_rate | min | max | empty | n\_unique | whitespace |
| --- | --- | --- | --- | --- | --- | --- | --- |
| movies\_title | 0 | 1 | 1 | 105 | 0 | 42025 | 0 |
| Genres | 0 | 1 | 4 | 6 | 0 | 21 | 0 |
| imdb\_id | 0 | 1 | 3 | 9 | 0 | 45159 | 0 |
| original\_language | 0 | 1 | 2 | 2 | 0 | 90 | 0 |
| countries | 0 | 1 | 2 | 6 | 0 | 144 | 0 |
| status | 0 | 1 | 6 | 15 | 0 | 7 | 0 |

**Variable type: numeric**

| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| movies\_id | 0 | 1 | 107469.46 | 111831.61 | 2 | 26301.25 | 59423.50 | 154425.75 | 4.691720e+05 |
| budget | 0 | 1 | 4249999.20 | 17474380.76 | 0 | 0.00 | 0.00 | 0.00 | 3.800000e+08 |
| popularity | 0 | 1 | 2.94 | 6.02 | 0 | 0.39 | 1.14 | 3.72 | 5.474900e+02 |
| release\_year | 0 | 1 | 1988.64 | 83.53 | 0 | 1978.00 | 2001.00 | 2010.00 | 2.020000e+03 |
| revenue | 0 | 1 | 11277197.87 | 64521085.78 | 0 | 0.00 | 0.00 | 0.00 | 2.787965e+09 |
| runtime | 0 | 1 | 94.13 | 38.41 | 0 | 85.00 | 95.00 | 107.00 | 1.256000e+03 |
| vote\_average | 0 | 1 | 5.63 | 1.91 | 0 | 5.00 | 6.00 | 6.80 | 1.000000e+01 |
| vote\_count | 0 | 1 | 110.53 | 492.73 | 0 | 3.00 | 10.00 | 34.00 | 1.407500e+04 |

## Checking the dimension of the dataframe after deleting columns and dropping missing values.

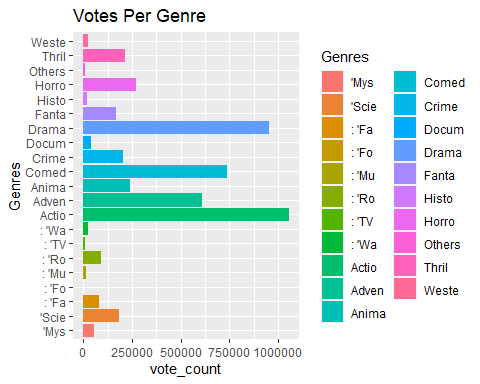
dim(movies1)

## [1] 45186 14

## DATA VISUALIZATION:

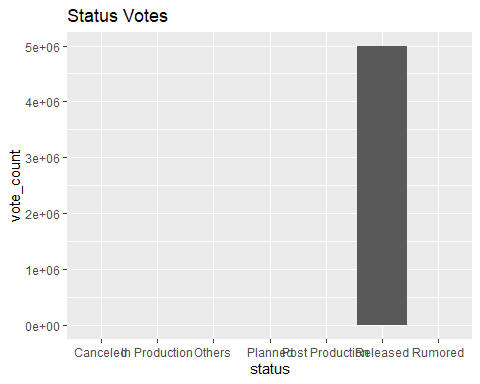
Checking the number of votes per Genre.

ggplot(movies1) +   
 geom\_col(mapping = aes(x=Genres, y=vote\_count, fill=Genres)) +   
 coord\_flip() +  
 labs(title="Votes Per Genre")



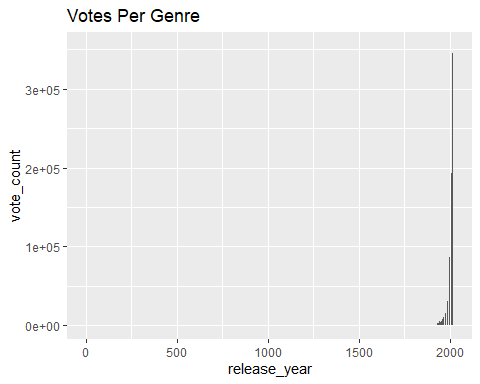
Checking number of votes with different status.

ggplot(movies1) +   
 geom\_col(mapping = aes(x=status, y=vote\_count)) +   
 labs(title="Status Votes")



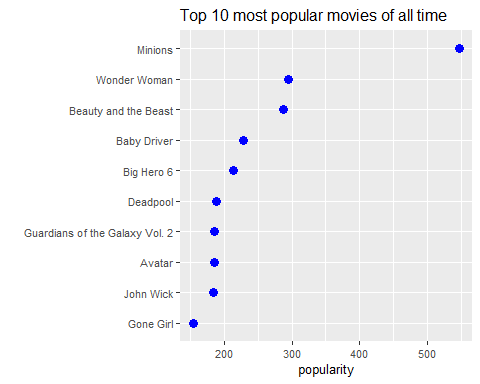
Checking for the number of votes per Genre.

ggplot(movies1) +   
 geom\_col(mapping = aes(x=release\_year, y=vote\_count)) +   
 labs(title="Votes Per Genre")



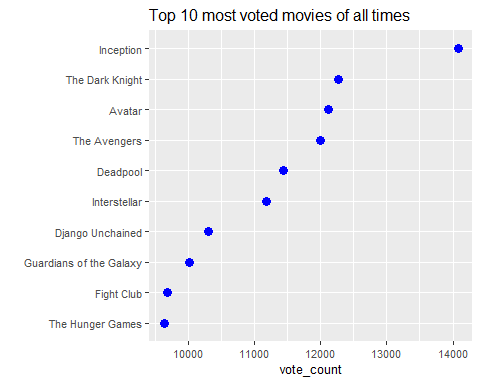
Checking the Top 10 movies popularity vise.

movies1 %>%  
 arrange(-popularity) %>%  
 slice\_head(n=10) %>%  
 ggplot(aes(reorder(movies\_title, popularity), popularity)) +  
 geom\_point(colour = "blue", size = 3) +  
 theme(text=element\_text(size=10)) +   
 coord\_flip() +  
 labs(title = "Top 10 most popular movies of all time", x = "")



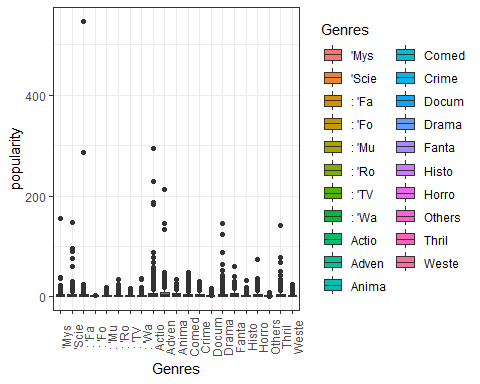
Checking for the Top 10 most voted movies.

movies1 %>%  
 arrange(-vote\_count) %>%  
 slice\_head(n=10) %>%  
 ggplot(aes(reorder(movies\_title, vote\_count), vote\_count)) +  
 geom\_point(colour = "blue", size = 3) +  
 theme(text=element\_text(size=10)) +   
 coord\_flip() +  
 labs(title = "Top 10 most voted movies of all times", x = "")



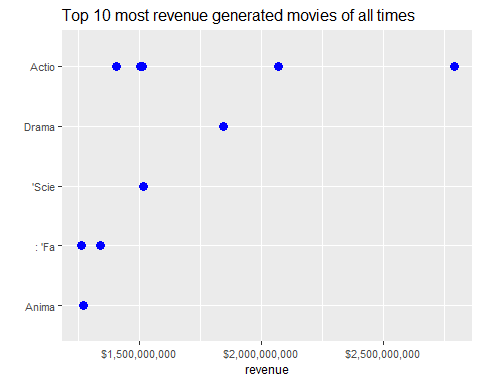
Checking the most popular Genres.

ggplot(movies1 , aes(x = Genres, y = popularity)) +  
 geom\_boxplot(aes(fill=Genres)) + theme\_bw() +  
 theme(axis.text.x = element\_text (angle = 90))



Checking for the Top 10 highest revenue generated movies of all time.

movies1 %>%  
 arrange(-revenue) %>%  
 slice\_head(n=10) %>%  
 ggplot(aes(reorder(Genres, revenue), revenue)) +  
 geom\_point(colour = "blue", size = 3) +  
 theme(text=element\_text(size=10)) +   
 scale\_y\_continuous(labels=scales::dollar\_format()) +  
 coord\_flip() +  
 labs(title = "Top 10 most revenue generated movies of all times", x = "")



Checking for the most revenue generated Genres.

movies1 %>%  
 ggplot(aes(x=Genres, y=revenue)) +  
 geom\_col(fill='blue') +  
 coord\_flip() +  
 scale\_y\_continuous(labels=scales::dollar\_format()) +  
 labs(title="Most revenue generated genres")

