Movie Franchises Analysis

Shanshan Bradford

```
# The 3rd hypothesis: The movie franchises are the rising money makers,
# and it is important to the profitability of the movie business.

rm(list = ls())
working.path = "/Users/syu/Library/CloudStorage/OneDrive-
St.JudeChildren'sResearchHospital/UDrive/Documents_syu_Backup/Github_deposit/
MoviesFranchises"
setwd(working.path)

#read two data files and order data sets in an invert chronological order
movie.meta = read.csv("movie_metadata_cleaned.csv", header = T, sep = ",",
                      as.is = T, na.strings = c(""))
movie.meta = movie.meta[order(movie.meta$title_year, decreasing = T), ]

movfran.fina = read.csv("MovieFranchise_FinanceInfo.csv", header = T, sep =
",",
                        as.is = T, na.strings = c("", "NA"))
movfran.fina = movfran.fina[order(movfran.fina$Franchise, decreasing = T), ]

# the year and gross profit data will be needed
# remove the null value of title year and gross revenue
movie.gross = as.numeric(as.character(movie.meta$gross))

# check how many records will be removed
length(movie.gross[is.na(movie.gross)])

## [1] 884

movie.meta.clean = movie.meta[!is.na(movie.gross),] # remove records of no
gross revenue

movie.year = as.character(as.factor(movie.meta.clean$title_year))
movie.year[is.na(movie.year)] # check how many records will be removed

## [1] NA NA NA

movie.meta.clean = movie.meta.clean[!is.na(movie.year),]# remove records of
no title_year
dim(movie.meta.clean)

## [1] 4156   28
## Clean error in several text content variables that will be used for
regression analysis
unique(movie.meta.clean$color) # color variable has unnecessary space

## [1] "Color"            " Black and White" NA

movie.meta.clean$color = gsub("^ +", "", movie.meta.clean$color)

# content_rating variable mixed old and new rating system
# replace the old rating records with current USA rating system
unique(movie.meta.clean$content_rating)

##  [1] "PG-13"     "PG"        "R"         NA          "Not Rated" "G"
##  [7] "Unrated"   "NC-17"     "X"         "GP"        "M"
"Approved"
## [13] "Passed"

table(movie.meta.clean$content_rating)

##
##  Approved         G        GP         M     NC-17 Not Rated    Passed
PG
##        18        95         1         2         6        56         3
611
##     PG-13         R   Unrated         X
##      1400      1856        34        10

# If a film has not been submitted for a rating or is an uncut version,
# the labels Not Rated (NR) or Unrated (UR) are often used
movie.meta.clean$content_rating = gsub("Not Rated", "Unrated",
movie.meta.clean$content_rating)

# rating "Approved" is only for Pre-1968 titles, should be equal to "Passed"
# films were approved or disapproved simply based on whether they were deemed
'moral' or 'immoral'
movie.meta.clean$content_rating = gsub("Passed", "Approved",
movie.meta.clean$content_rating)

# "M" was renamed to "GP" in 1970
movie.meta.clean$content_rating = gsub("M", "GP",
movie.meta.clean$content_rating)
# in 1972, "GP" was revised to "PG"
movie.meta.clean$content_rating = gsub("GP", "PG",
movie.meta.clean$content_rating)
# in 1990, "X" replaced by "NC-17"
movie.meta.clean$content_rating = gsub("X", "NC-17",
movie.meta.clean$content_rating)

#########################################################################
# adjust the inflation ratio with CPI data
#########################################################################
```

```
# Before adjustment, change the datatype of gross revenue and title year
movie.meta.clean$movie.gross =
as.numeric(as.character(movie.meta.clean$gross))
movie.meta.clean$movie.year =
as.character(as.factor(movie.meta.clean$title_year))
# Then, remove data of unnecessary data types
movie.meta.clean$gross = NULL
movie.meta.clean$title_year = NULL


# Then, extract gross profit and year information in two vectors
# get ready for inflation adjustment
new.movie.gross = movie.meta.clean$movie.gross
# year information is treated as numeric here to ease the inflation
adjustment
new.movie.year = as.numeric(movie.meta.clean$movie.year)

range(new.movie.year) # this data set almost has a century of movie
information
```

```
## [1] 1920 2016
```

```
# load in historical inflation data
cpi.infla.hist = read.csv("CPIHistoricInflationData.csv", header = T,
                          sep = ",", as.is = T, na.strings = "NA")
# check out the variables and subset the historical inflation data to 2016
colnames(cpi.infla.hist)
```

```
##  [1] "X"    "Year" "Jan"  "Feb"  "Mar"  "Apr"  "May"  "Jun"  "Jul"  "Aug"
## [11] "Sep"  "Oct"  "Nov"  "Dec"  "Ave."
```

```
cpi.infla = cpi.infla.hist[-1, c("X", "Year", "Ave.")]
colnames(cpi.infla) = c("X", "Year", "Ave")

# reassign new row names and index number
row.names(cpi.infla) <- 1:104
cpi.infla$X <- 1:104
head(cpi.infla) # check whether the cpi.infla is updated after change
```

```
##   X Year     Ave
## 1 1 2016 240.008
## 2 2 2015 237.017
## 3 3 2014 236.736
## 4 4 2013 232.957
## 5 5 2012 229.594
## 6 6 2011 224.939
```

```
str(cpi.infla) # check the data type to match the data from two other data
sets
```

```
## 'data.frame':    104 obs. of  3 variables:
##  $ X   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Year: int  2016 2015 2014 2013 2012 2011 2010 2009 2008 2007 ...
##  $ Ave : num  240 237 237 233 230 ...
```

```
# To simplify the code, extract year and annual inflation rate to two vectors
cpi.ave = cpi.infla$Ave
cpi.year = cpi.infla$Year

# calculate how much of the past profit would worth in 2016
# use the year of each movie to find the corresponding inflation ratio
# then multiply the inflation ratio to the recorded gross profit
adjust.gross <- sapply(1:length(new.movie.gross), simplify = T,
  function(i)
  {new.movie.gross[i] * cpi.ave[1]/cpi.ave[grep(new.movie.year[i],
cpi.year)]})

# add back the inflation adjusted gross back to the cleaned data set
movie.meta.clean$adjust.gross = adjust.gross
#############################################################################
#########

# Clean text content in names and franchise title
# Replace non graphical character and punctuation with one space each
movie.names = gsub("[^[:graph:]]", " ", movie.meta.clean$movie_title)
movie.names = gsub("[[:punct:]]{1,20}", " ", movie.names)
Fran.title = gsub("[^[:graph:]]", " ", movfran.fina$Franchise)
Fran.title = gsub("[[:punct:]]{1,20}", " ", Fran.title)

# Replace tab and extra space introduced early with one space
movie.names = gsub("[ |\t]{2,}", " ", movie.names)
movie.names = gsub("\\s+", " ", movie.names)
Fran.title = gsub("[ |\t]{2,}", " ", Fran.title)
Fran.title = gsub("\\s+", " ", Fran.title)

# Remove extra blank space at the beginning and the end
movie.names = gsub("^ +", "", movie.names)
movie.names = gsub(" $+", "", movie.names)
Fran.title = gsub("^ +", " ", Fran.title)
Fran.title = gsub(" $+", " ", Fran.title)

# add cleaned title and names back to the data frames loaded from csv files
movie.meta.clean$movie.names.clean = movie.names
movfran.fina$fran.title.clean = Fran.title

# find whether there are franchise titles were recorded more than once
which(table(Fran.title) >1) # expected result is none
```

```
## named integer(0)
```

```r
#change names to lower cases for further analysis
movie.names = tolower(movie.names)
Fran.title = tolower(Fran.title)

# use franchise names to find the franchise movie names
fran.mov.list = sapply(1:length(Fran.title), simplify = T,
            function(i){grep(Fran.title[i], movie.names, ignore.case = T)})

fran.movname.list = sapply(1:length(Fran.title), simplify = T,
  function(i){grep(Fran.title[i], movie.names, ignore.case = T, value = T)})


# check franchise title with numeric title cause mismatches
fran.movname.list[c(751:760)] # it looks like franchise 300 grab other movies

## [[1]]
## [1] "300 rise of an empireξ"   "300ξ"
## [3] "mr 3000ξ"                 "3000 miles to gracelandξ"
##
## [[2]]
## [1] "3 ninjas kick backξ"
##
## [[3]]
## character(0)
##
## [[4]]
## [1] "28 days later ξ"
##
## [[5]]
## [1] "21 jump streetξ"
##
## [[6]]
## [1] "2001 a space odysseyξ"
##
## [[7]]
## [1] "12 roundsξ"
##
## [[8]]
## character(0)
##
## [[9]]
## character(0)
##
## [[10]]
## character(0)

fran.mov.list[[751]] # the wrong movie numbers are 2235 and 2746

## [1]  239 1771 2235 2746
```

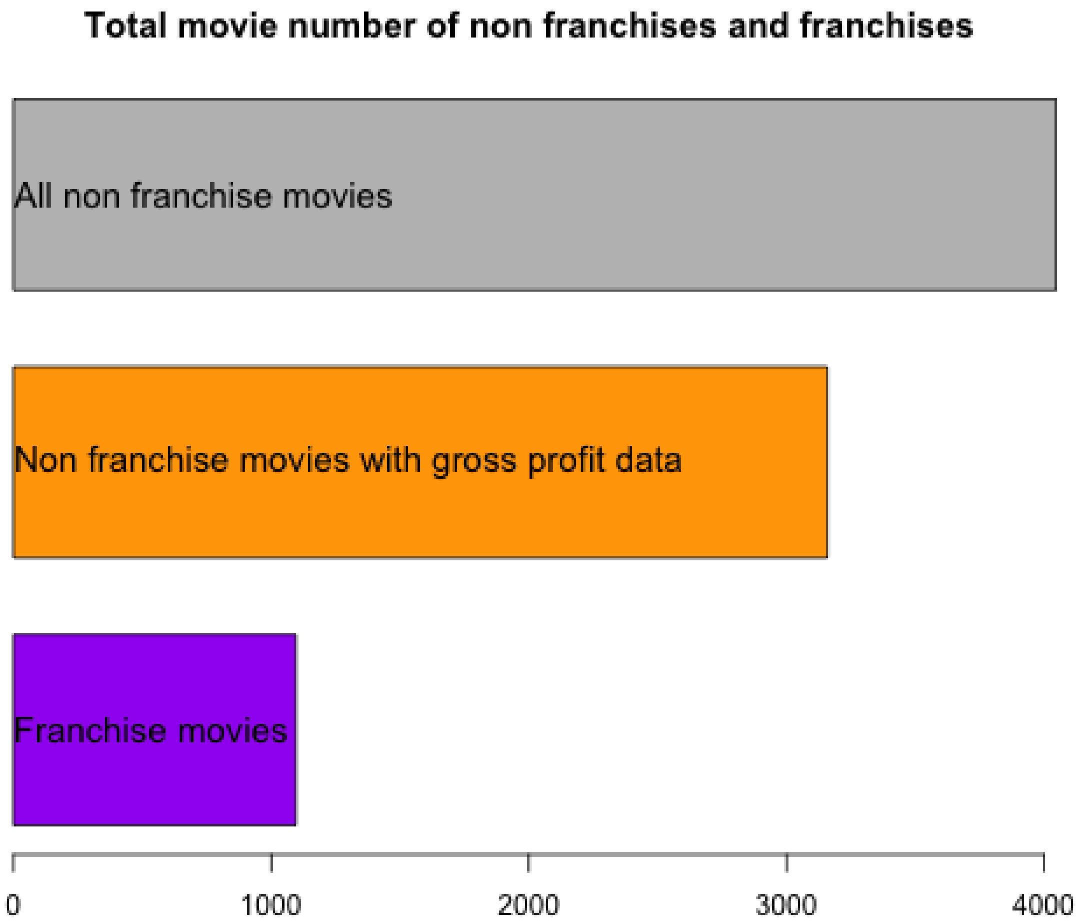
```
# unlist the franchise movie
fran.mov.index = unlist(fran.mov.list, recursive = T)
two.wrongmov = c(grep("2235", fran.mov.index), grep("2746", fran.mov.index))

# check whether wrong movies is removed
length(fran.mov.index) -length(fran.mov.index[-two.wrongmov])
```

```
## [1] 2
```

```
fran.mov.index = fran.mov.index[-two.wrongmov]

# create a subset data of franchise movies and non franchise movies
fran.movie = movie.meta.clean[fran.mov.index,]
other.movie = movie.meta.clean[-fran.mov.index,]
other.movie.nogross = movie.meta[-fran.mov.index,]
write.csv(fran.movie, file = "FranchiseMovieDetails.csv", eol = "\r\n")

fran.movie$movie_title = NULL #remove original (pre-cleaned) movie title
other.movie$movie_title = NULL #remove original (pre-cleaned) movie title

# compare these two subsets length and visualize the comparison
movieNo.compare = c(length(row.names(fran.movie)),
  length(row.names(other.movie)), length(row.names(other.movie.nogross)))

#generate an image file
png("barplot_MovieNumberComparison.png")
par(mar=c(5, 1, 5, 1))
bargra1 = barplot(movieNo.compare, horiz = T,
        col = c("purple", "orange","gray"), beside = FALSE, space = 0.4,
      width = c(0.01, 0.01, 0.01))
title(main = "Total movie number of non franchises and franchises",
      cex.main = 1.25, line = 1, adj = 0.5)
text(bargra1, adj = c(0, NA), cex = 1.25,
     labels = c("Franchise movies",
                "Non franchise movies with gross profit data",
                "All non franchise movies") )
#save the image plot
dev.off()
```

```
## quartz_off_screen
##                 2
```

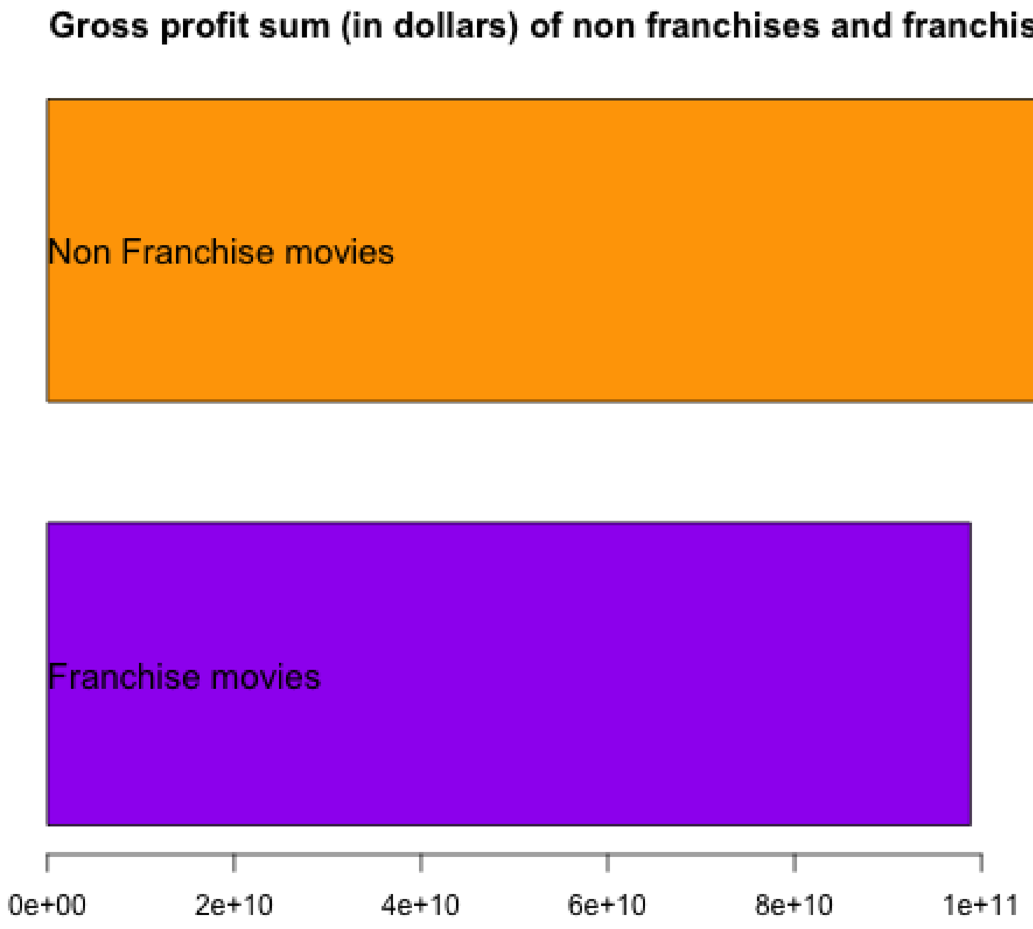
```
# knit the generated image file into the report
knitr::include_graphics(paste(working.path,
"barplot_MovieNumberComparison.png", sep = "/"))
```



```
names(movieNo.compare) = c("Franchise movies",
                           "Non franchise movies with gross profit data",
                           "All non franchise movies")
movieNo.compare

##                            Franchise movies
##                                        1096
## Non franchise movies with gross profit data
##                                        3158
##                    All non franchise movies
##                                        4045

#calculate the total and mean profit of two types of movies
avgprof.fran = mean(fran.movie$movie.gross, na.rm = T)
avgprof.other = mean(other.movie$movie.gross, na.rm = T)

gross.compare = c(sum(fran.movie$movie.gross, na.rm = T),
                  sum(other.movie$movie.gross, na.rm = T))

png("barplot_ProfitSumComparison.png")
par(mar=c(5, 1, 5, 1))
bargra2 = barplot(gross.compare, horiz = T,
        col = c("purple", "orange"), beside = FALSE, las = 1,
        space = 0.4, width = c(0.01, 0.01, 0.01))
title(main = "Gross profit sum (in dollars) of non franchises and
franchises",
      cex.main = 1.25, line = 1, adj = 0.5)
text(bargra2, adj = c(0, NA), cex = 1.25,
     labels = c("Franchise movies", "Non Franchise movies") )
dev.off()
```

```
## quartz_off_screen
##                  2
```

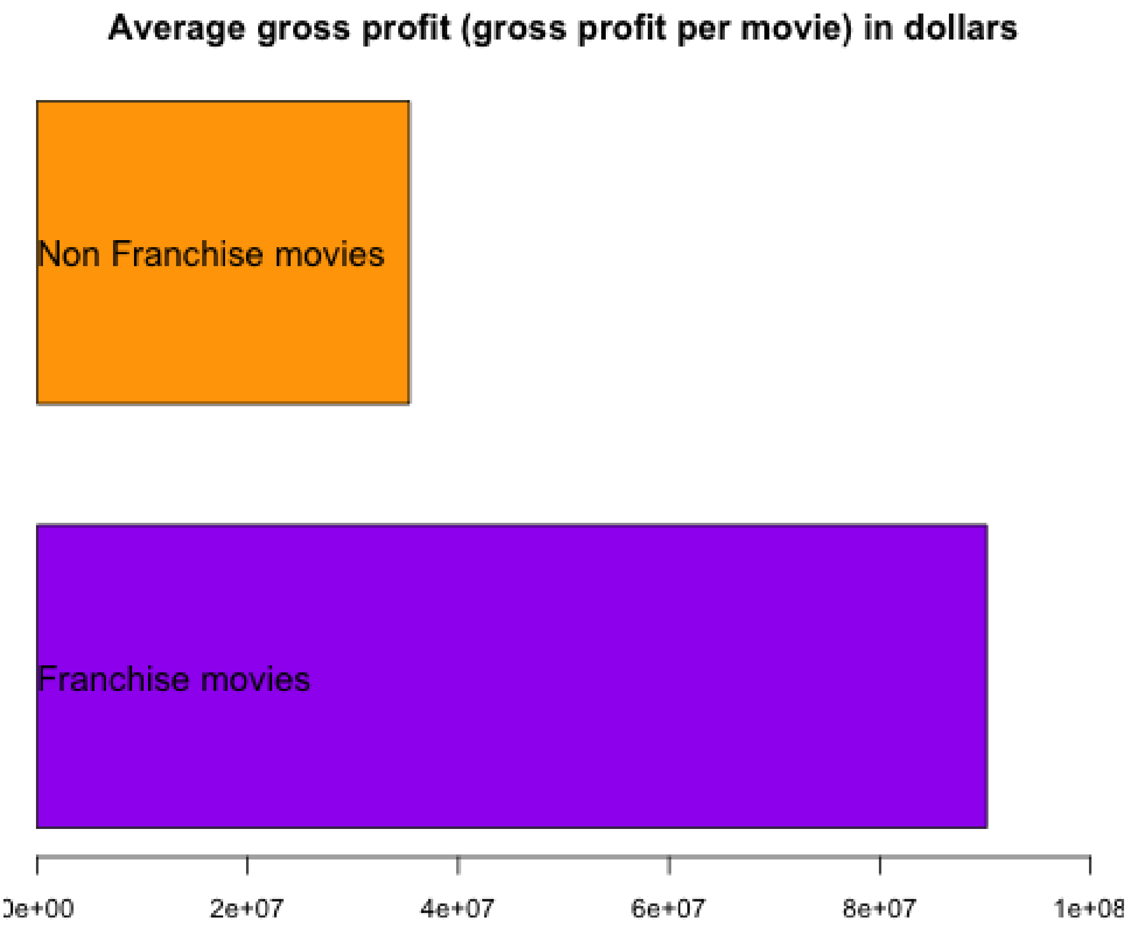
```
knitr::include_graphics(paste(working.path,
"barplot_ProfitSumComparison.png", sep = "/"))
```



```
names(gross.compare) = c("Franchise movies", "Non Franchise movies")
gross.compare

##     Franchise movies Non Franchise movies
##          98819051354         111622020100

png("barplot_AvgfitComparison.png")
par(mar=c(5, 1, 5, 1))
bargra3 =barplot(c(avgprof.fran, avgprof.other), horiz = T,
        col = c("purple", "orange"), beside = FALSE, space = 0.4,
        width = c(0.01, 0.01, 0.01), xlim = c(0, 1e+08), cex.axis = 0.95)
title(main = "Average gross profit (gross profit per movie) in dollars",
      cex.main = 1.25, line = 1, adj = 0.5)
text(bargra3, adj = c(0, NA), cex = 1.25,
     labels = c("Franchise movies", "Non Franchise movies"))
dev.off()
```

```
## quartz_off_screen
##                  2
```

```
knitr::include_graphics(paste(working.path, "barplot_AvgfitComparison.png",
sep = "/"))
```



```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)

#############################################################################
#              K-means clustering and linear regression modeling            #
#############################################################################

#############################################################################
#      K-means clustering, to explore the franchise movie financial data    #
#############################################################################

# Check the datatype of the franchise finance data
```

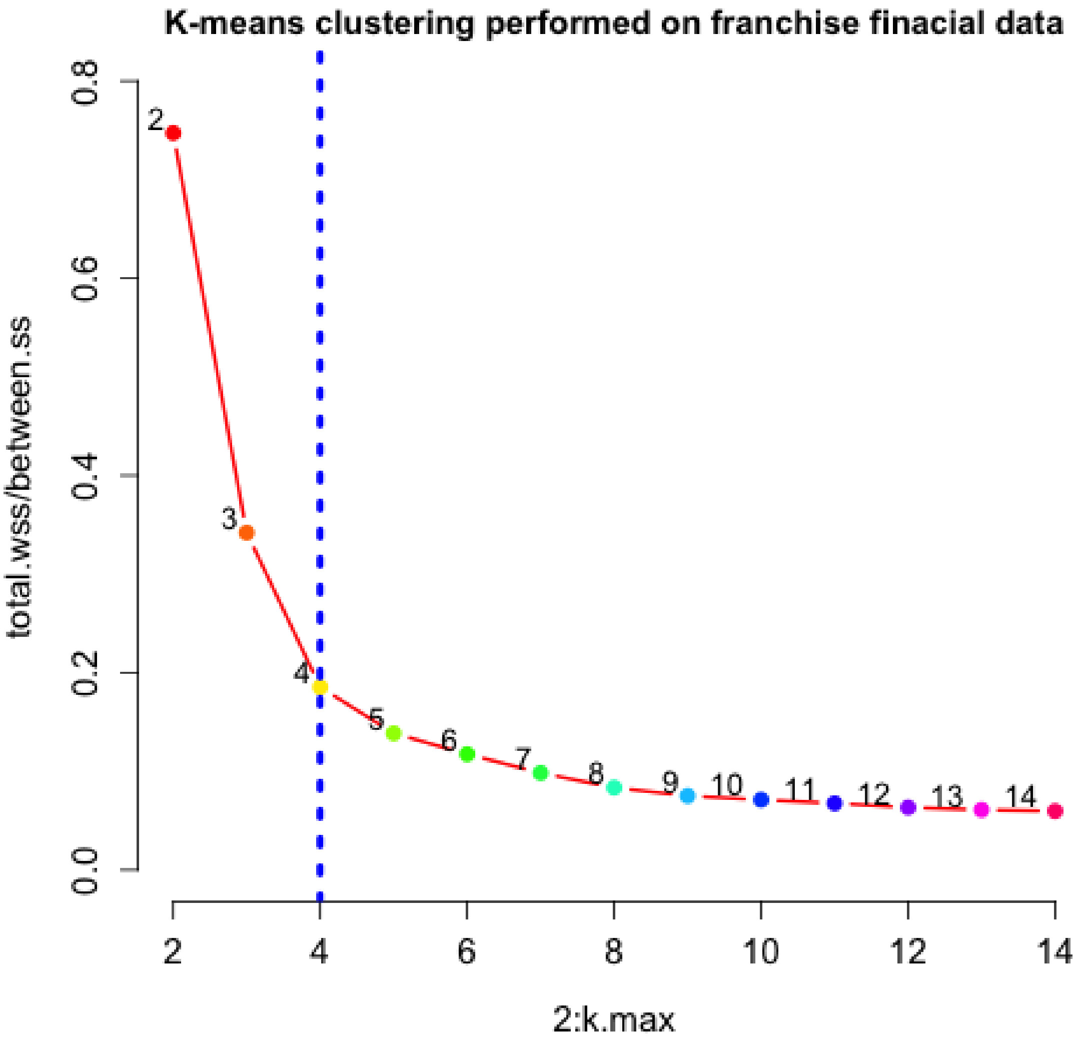
```r
str(movfran.fina) # all the box office records are character because of the
"$"

## 'data.frame':    760 obs. of  10 variables:
##  $ X                         : int  215 372 388 270 447 755 350 185 9 271
...
##  $ Franchise                 : chr  "Zorro" "Zoolander" "Zombieland"
"Young Guns" ...
##  $ No..of.Movies             : int  2 2 2 2 2 1 2 3 14 2 ...
##  $ Domestic.Box.Office       : chr  "$139,404,081 " "$74,020,943 "
"$75,590,286 " "$88,870,054 " ...
##  $ Infl..Adj..Dom..Box.Office: chr  "$241,333,859 " "$100,712,640 "
"$89,700,466 " "$189,951,971 " ...
##  $ Worldwide.Box.Office      : chr  "$375,175,336 " "$116,129,674 "
"$102,236,596 " "$88,870,054 " ...
##  $ First.Year                : int  1998 2001 2009 1988 2004 2016 2010
2002 2000 1998 ...
##  $ Last.Year                 : int  2005 2016 2011 1990 2011 2016 2014
2017 2019 2008 ...
##  $ No..of.Years              : int  7 15 2 2 7 NA 4 15 19 10 ...
##  $ fran.title.clean          : chr  "Zorro" "Zoolander" "Zombieland"
"Young Guns" ...

# replace the dollar sign, comma, and extra space of box office records
# coerce the data to numeric data type after clean the number records
infl.adj.dobo = gsub("\\$", "", movfran.fina$Infl..Adj..Dom..Box.Office)
infl.adj.dobo = gsub(",", "", infl.adj.dobo)
infl.adj.dobo = gsub("^ +", "", infl.adj.dobo)
infl.adj.dobo = gsub(" $+", "", infl.adj.dobo)
infl.adj.dobo = as.numeric(infl.adj.dobo)

world.dobo = gsub("\\$", "", movfran.fina$Worldwide.Box.Office)
world.dobo = gsub(",", "", world.dobo)
world.dobo = gsub("^ +", "", world.dobo)
world.dobo = gsub(" $+", "", world.dobo)
world.dobo = as.numeric(world.dobo)

do.bo = gsub("\\$", "", movfran.fina$Domestic.Box.Office)
do.bo = gsub(",", "", do.bo)
do.bo = gsub("^ +", "", do.bo)
do.bo = gsub(" $+", "", do.bo)
do.bo = as.numeric(do.bo)

# add new numeric data type to the data frame movfan.fina
movfran.fina$infl.adj.dom.boxoffice = infl.adj.dobo
movfran.fina$glob.boxoffice = world.dobo
movfran.fina$dome.boxoffice = do.bo
movfran.fina$other.boxoffice = world.dobo - do.bo

# replace the old character box office record with the NULL
```

```
movfran.fina$Infl..Adj..Dom..Box.Office = NULL
movfran.fina$Worldwide.Box.Office = NULL
movfran.fina$Domestic.Box.Office = NULL
# change the long variable names to short ones
names(movfran.fina) = c("X", "Franchise", "tot.movies", "First.Year",
"Last.Year",
                        "tot.years","fran.title.clean",
"infl.adj.dom.boxoffice",
                        "glob.boxoffice","dome.boxoffice", "other.boxoffice")

# Then, check whether and where NA value in the franchise movie data are
which(is.na(movfran.fina[, -c(2,7)]) == T, arr.ind = T)[1:30,] # all NA in
No.of.Years

##     row col
## 755   6   5
## 759  18   5
## 728  24   5
## 738  33   5
## 751  52   5
## 745  59   5
## 733  73   5
## 639  78   5
## 760  89   5
## 732 105   5
## 716 108   5
## 721 111   5
## 253 114   5
## 251 125   5
## 613 146   5
## 593 148   5
## 651 161   5
## 521 176   5
## 744 192   5
## 612 194   5
## 624 197   5
## 315 203   5
## 594 205   5
## 641 208   5
## 604 232   5
## 475 235   5
## 756 236   5
## 741 237   5
## 754 270   5
## 616 276   5

# Find out row numbers/index for NA value in franchise financial data
franyear.nv = which(is.na(movfran.fina$tot.years) == T)
length(franyear.nv)
```

```
## [1] 93

# All these NA value movies have the same "First.Year" and "Last.Year"
identical(movfran.fina$First.Year[franyear.nv],
movfran.fina$Last.Year[franyear.nv])

## [1] TRUE

# This means these movie franchise were in theater for a year

# majority of movie franchises of one in-theater year have only one movie
table(movfran.fina[franyear.nv, ]$tot.movies)

##
##  1  2  3  4
## 80 10  2  1

# Given most popular movies running in theaters less than one year,
# the in-theater year info for franchises of one movie is probably left
censored
# Possibly it is one reason that these data is missing (measurement unit is
year)

# subset the franchises of one movie that is left censored
franyear.left = which(movfran.fina[franyear.nv,]$tot.movies > 1)
# use R code movfran.fina[franyear.nv[franyear.left],1:5] to make sure index
vectors are right
# and knit the data with a table format
knitr::kable(head(movfran.fina[franyear.nv[franyear.left],2:6], 30),
"simple")
```

	Franchise	tot.movies	First.Year	Last.Year	tot.years
651	St. Trinian’s	2	2009	2009	NA
521	Smoke	2	1995	1995	NA
641	San Francisco Opera Cinemacasts 2007	4	2008	2008	NA
604	Red Cliff	2	2009	2009	NA
628	On the Run	2	2004	2004	NA
693	MSG The Messenger of God	2	2015	2015	NA
707	Kiseijuu	2	2015	2015	NA
537	Jean de Florette	2	1987	1987	NA
690	Gangster Ka	2	2015	2015	NA
653	Donald Strachey	2	2008	2008	NA
370	Dollar Trilogy	3	1967	1967	NA
330	Breakin’	2	1984	1984	NA
649	As Mil e Uma Noites	3	2015	2015	NA

```
# Replace the tot.year NA value with 0.5 to the left censored data
# Use 0.5 year (6 months) as an estimated average of movie running time
movfran.fina[franyear.nv[-franyear.left],]$tot.years = 0.5
# Use 1 year for franchises having more than one movie but only lasting for a
year
movfran.fina[franyear.nv[franyear.left],]$tot.years = 1

# remove the text data out of franchise financial data for k-means clustering
str(movfran.fina[, -c(1,2,7,9,10)]) # check the data; use inflation adjusted
domestic data

## 'data.frame':    760 obs. of  6 variables:
##  $ tot.movies            : int  2 2 2 2 2 1 2 3 14 2 ...
##  $ First.Year            : int  1998 2001 2009 1988 2004 2016 2010 2002
2000 1998 ...
##  $ Last.Year             : int  2005 2016 2011 1990 2011 2016 2014 2017
2019 2008 ...
##  $ tot.years             : num  7 15 2 2 7 0.5 4 15 19 10 ...
##  $ infl.adj.dom.boxoffice: num  2.41e+08 1.01e+08 8.97e+07 1.90e+08
5.74e+07 ...
##  $ other.boxoffice       : num  2.36e+08 4.21e+07 2.66e+07 0.00 1.20e+07
...

# search optimal k with elbow plot for k-means clustering
library(cluster)
library(ggplot2)

set.seed(12345)
k.max = 14
total.wss = sapply(2:k.max, simplify = T,
                  function(k){ kmeans(movfran.fina[, -c(1,2,7,9,10)], k,
nstart = 50,
                                     iter.max = 100)$tot.withinss })

between.ss = sapply(2:k.max, simplify = T,
                 function(k){ kmeans(movfran.fina[, -c(1,2,7,9,10)], k,
nstart = 50,
                                    iter.max = 100)$betweenss })

total.wss/between.ss

##  [1] 0.74732116 0.34201513 0.18536201 0.13852969 0.11737684 0.09824969
##  [7] 0.08336627 0.07504648 0.07106465 0.06749613 0.06324600 0.06086837
## [13] 0.05959745

png("plot_KmeansElbow.png")
par(mar=c(4, 4, 4, 1))
plot(2:k.max, total.wss/between.ss, type = "b", pch = 19, col =
rainbow(c(k.max - 1)),
     frame.plot = F, lwd = 2, cex.lab = 1.25, cex.axis = 1.25,
     xlim = c(2,14), ylim = c(0,0.8))
title(main = "K-means clustering performed on franchise finacial data", adj =
0.5,
      line = 0.5)
text(2:k.max, total.wss/between.ss, labels = 2:k.max, cex = 1.1, adj = c(1.5,
-0.2))
abline(v = 4, lwd = 4, lty = 3, col = "blue")
dev.off()

## quartz_off_screen
##                 2

knitr::include_graphics(paste(working.path, "plot_KmeansElbow.png", sep =
"/"))
```



```
# set k = 4 for k-means
set.seed(12345)
```

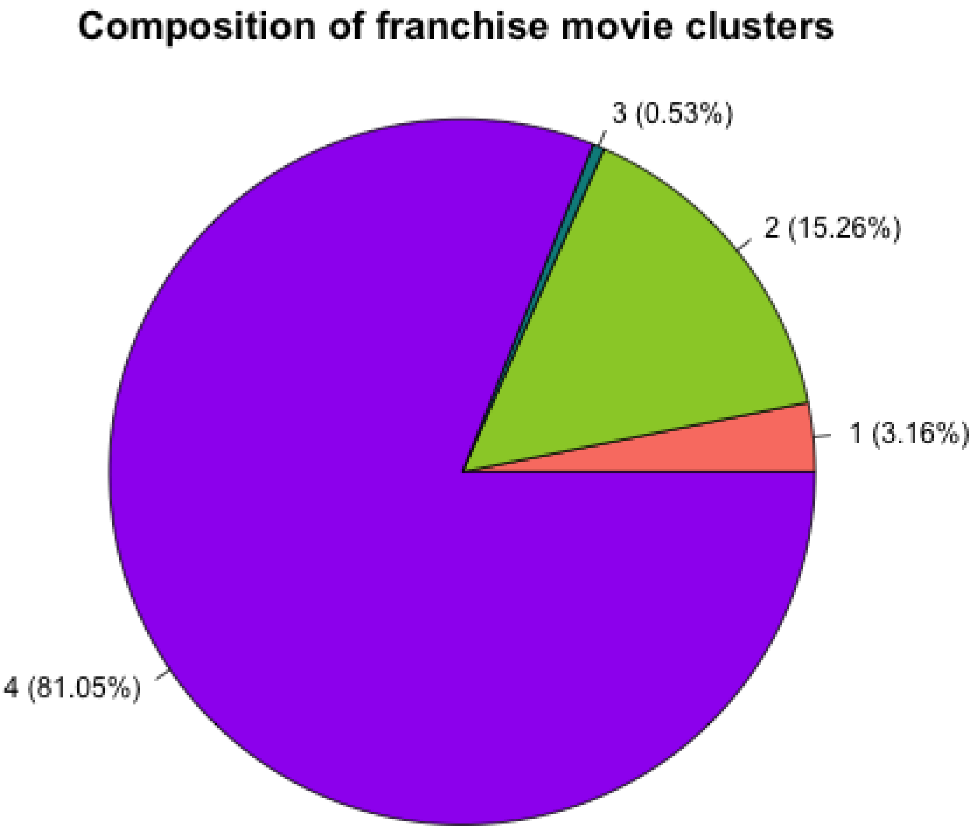
```r
kcluster.movfran = kmeans(movfran.fina[, -c(1,2,7,9,10)], 4, nstart =
50,iter.max = 100)

# add the clusters to the movie franchise financial data
movfran.fina$K.cluster = kcluster.movfran$cluster
# select the variables needed for result discussion and further analysis
colnames(movfran.fina)
```

```
##  [1] "X"                    "Franchise"            "tot.movies"
##  [4] "First.Year"           "Last.Year"            "tot.years"
##  [7] "fran.title.clean"     "infl.adj.dom.boxoffice" "glob.boxoffice"
## [10] "dome.boxoffice"       "other.boxoffice"      "K.cluster"
```

```r
movfran.result = cbind(movfran.fina[, c(1,7,12)], movfran.fina[, c(8,11)],
                       movfran.fina[,c(3,6)], movfran.fina[,c(4,5)])

#organize the result by k-means clusters
movfran.result = movfran.result[order(movfran.result$K.cluster, decreasing =
F), ]
per.kcluster = prop.table(table(movfran.result$K.cluster))
per.kcluster = round(per.kcluster*100, 2)
per.kcluster = paste(per.kcluster, "%", sep = "")
kluster.label = paste(c("1","2","3","4"), " ", "(", per.kcluster, ")", sep =
"")

png("pie_K-clusterComposition.png")
par(mar = c(4,1,1,1))
kcluster.pie = pie(table(movfran.result$K.cluster), clockwise = F,
                   labels = kluster.label, cex.main = 1.4, line = -1.25,
                   main = "Composition of franchise movie clusters ",
                   col = c("salmon", "yellow green", "dark cyan","purple"))
dev.off()
```

```
## quartz_off_screen
##                 2
```

```r
knitr::include_graphics(paste(working.path, "pie_K-clusterComposition.png",
sep = "/"),
                         auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



```
# list movie names in clusters 1, 3, and 4
movfran.result[movfran.result$K.cluster == 3,]$fran.title.clean

## [1] "Star Wars"                  "Marvel Cinematic Universe"
## [3] "James Bond"                 "Harry Potter"

movfran.result[movfran.result$K.cluster == 4,]$fran.title.clean[1:50]

##  [1] "Zorro"                      "Zoolander"
##  [3] "Zombieland"                 "Young Guns"
##  [5] "You Got Served"             "Yokai Watch"
##  [7] "Yogi Bear"                  "X Files"
##  [9] "Wrong Turn"                 "World War Z"
## [11] "Work and the Glory"         "Wolf Creek"
## [13] "Without a Paddle"           "Winx Club"
## [15] "Winnie the Pooh"            "Willard"
## [17] "Wilden Kerle"               "Wild Things"
## [19] "Wild Orchid"               "Wild Geese"
## [21] "Why Did I Get Married"     "Whole Nine Yards"
## [23] "White Noise"               "White Fang"
## [25] "When Love Happens"         "When Calls the Heart"
## [27] "What Would Jesus Do "      "What the Bleep"
## [29] "Weiner Dog"                "Weekend at Bernie s"
## [31] "Wayne s World"             "Warlock"
## [33] "Wallace and Gromit"        "Wall Street"
## [35] "Waiting"                   "Viva Pedro Box"
## [37] "Visiteurs"                 "Vengeance Trilogy"
## [39] "VeggieTales"               "Van Wilder"
## [41] "Vacanze"                   "Vacancy"
## [43] "V H S"                     "USA Land of Opportunities"
## [45] "Urban Legend"              "Untouchables"
## [47] "Universal Soldier"         "Undisputed"
## [49] "Underworld"                "Under Siege"
```

```
movfran.result[movfran.result$K.cluster == 1,]$fran.title.clean
```

```
##  [1] "X Men"                             "Twilight"
##  [3] "Transformers"                      "The Hobbit"
##  [5] "Superman"                          "Star Trek"
##  [7] "Spider Man"                        "Shrek"
##  [9] "Planet of the Apes"                "Pirates of the Caribbean"
## [11] "Peter Jackson s Lord of the Rings" "Mission Impossible"
## [13] "Madagascar"                        "Jurassic Park"
## [15] "Iron Man"                          "Indiana Jones"
## [17] "Ice Age"                           "Hunger Games"
## [19] "Fast and the Furious"              "Despicable Me"
## [21] "DC Extended Universe"              "Dark Knight Trilogy"
## [23] "Batman"                            "Avatar"
```

```
# use following codes to show clusters results, and knit them into tables
# movfran.result[movfran.result$K.cluster == 3, -1]
knitr::kable(movfran.result[movfran.result$K.cluster == 3, 2:6], "simple")
```

	fran.title.clean	K.cluster	infl.adj.dom.boxoffice	other.boxoffice	tot.movies
1	Star Wars	3	6529365840	3874592228	12
3	Marvel Cinematic Universe	3	5390016938	7803549152	23
2	James Bond	3	5625743524	4964007386	26
4	Harry Potter	3	3399078859	5906843667	12

```
# movfran.result[movfran.result$K.cluster == 4, -1][1:30,]
knitr::kable(head(movfran.result[movfran.result$K.cluster == 4, 2:6], 20),
"simple")
```

	fran.title.clean	K.cluster	infl.adj.dom.boxoffice	other.boxoffice	tot.movies
215	Zorro	4	241333859	235771255	2

	fran.title.clean	K.cluster	infl.adj.dom.boxoffice	other.boxoffice	tot.movies
372	Zoolander	4	100712640	42108731	2
388	Zombieland	4	89700466	26646310	2
270	Young Guns	4	189951971	0	2
447	You Got Served	4	57422186	11975903	2
755	Yokai Watch	4	0	5786581	1
350	Yogi Bear	4	112882278	104528679	2
271	X Files	4	185218941	152466424	2
510	Wrong Turn	4	22755912	13231785	6
236	World War Z	4	221525388	329154939	2
543	Work and the Glory	4	9266493	0	3
512	Wolf Creek	4	22354424	12984045	2
403	Without a Paddle	4	83346008	6964845	2
759	Winx Club	4	0	18523991	1
331	Winnie the Pooh	4	134428350	127960901	6
415	Willard	4	78459917	0	2
703	Wilden Kerle	4	0	29700000	2
448	Wild Things	4	56541184	25781400	2
506	Wild Orchid	4	24429833	0	2
728	Wild Geese	4	0	0	1

```
# movfran.result[movfran.result$K.cluster == 1, -1]
knitr::kable(head(movfran.result[movfran.result$K.cluster == 1, 2:6], 20),
"simple")
```

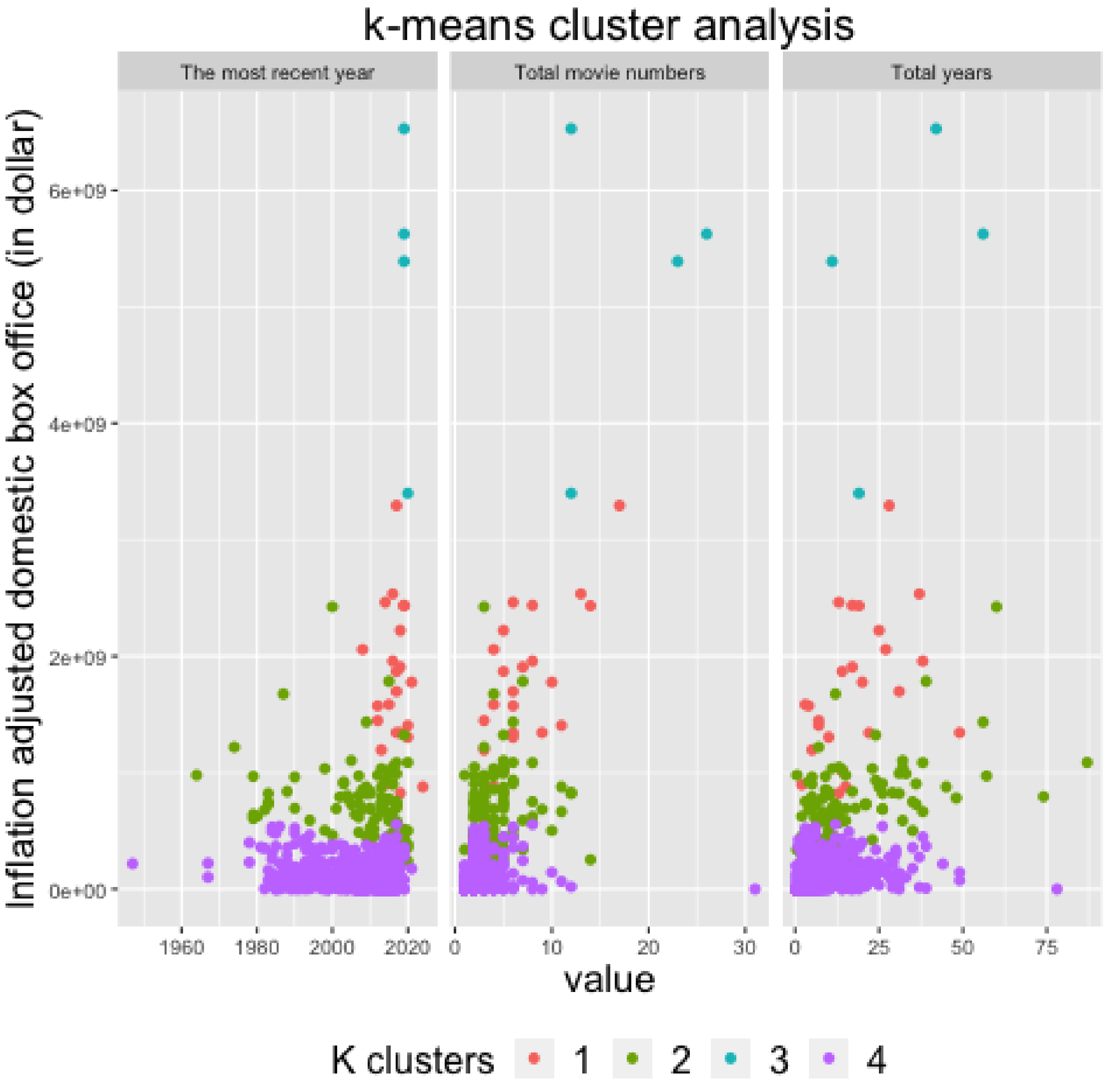
	fran.title.clean	K.cluster	infl.adj.dom.boxoffice	other.boxoffice	tot.movies
9	X Men	1	2432925375	2972739360	14
21	Twilight	1	1573729675	1951548393	6
18	Transformers	1	1697217057	2927234228	6
52	The Hobbit	1	897304916	2116000000	3
13	Superman	1	1958183722	1254701047	8
6	Star Trek	1	2534934135	865520289	13
8	Spider Man	1	2436949137	2945378747	8
14	Shrek	1	1907496906	2127785519	7
25	Planet of the Apes	1	1344102393	1333798977	9
15	Pirates of the Caribbean	1	1869342701	3043094095	5
7	Peter Jackson s Lord of the Rings	1	2462373441	4043374332	6

	fran.title.clean	K.cluster	infl.adj.dom.boxoffice	other.boxoffice	tot.movies
26	Mission Impossible	1	1342145009	1866036048	6
60	Madagascar	1	824536103	1597299728	5
11	Jurassic Park	1	2220851105	2236833067	5
30	Iron Man	1	1195061980	1381690479	3
12	Indiana Jones	1	2057228548	1041500294	4
39	Ice Age	1	1015160281	2387647674	5
20	Hunger Games	1	1585025298	1508428615	4
17	Fast and the Furious	1	1777326927	3622617438	10
28	Despicable Me	1	1304970563	2495486726	6

```
# create side by side visualization for comparison
library(reshape2) # to create with ggplot2, need melt of reshape2 to remold
data
library(plyr)

# subset the result data needed to be melted
plot1.subset = movfran.result[, c(3:4, 6:7, 9)]

# choose ID variables from the subset that are not going to be melted
# facet_wrap will use these ID to create graph panel(layout)
plot1.id1 = names(movfran.result)[3:4]
plot1.subset = melt(plot1.subset, id = plot1.id1)
plot1.subset$variable = gsub("tot.movies", "Total movie numbers",
plot1.subset$variable)
plot1.subset$variable = gsub("tot.years", "Total years",
plot1.subset$variable)
plot1.subset$variable = gsub("Last.Year", "The most recent year",
plot1.subset$variable)

str(plot1.subset) # check the data types of the melted subset

## 'data.frame':    2280 obs. of  4 variables:
##  $ K.cluster             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ infl.adj.dom.boxoffice: num  2.43e+09 1.57e+09 1.70e+09 8.97e+08
1.96e+09 ...
##  $ variable              : chr  "Total movie numbers" "Total movie
numbers" "Total movie numbers" "Total movie numbers" ...
##  $ value                 : num  14 6 6 3 8 13 8 7 9 5 ...

# This panel will plot the relationship between tot.movies, tot.years,
Last.years
# and inflation adjusted domestic box office record

png("ggplot_KmeansAnalysis.png")
ggplot(plot1.subset, aes(value, infl.adj.dom.boxoffice,
      col = as.factor(plot1.subset$K.cluster)) +
      geom_point(shape = 16, size = 2) +
      facet_wrap( ~ variable, nrow = 1, ncol = 3, scales = "free_x") +
      theme(legend.position = "bottom",
      legend.text = element_text(size = 16),
      plot.title = element_text(size = rel(1.75), hjust = 0.5, vjust=0),
      axis.title.y = element_text(size = rel(1.5), angle = 90),
      axis.title.x = element_text(size = rel(1.5), angle = 0)) +
      labs(title = "k-means cluster analysis",  par(adj = 1)) +
      ylab("Inflation adjusted domestic box office (in dollar)") +
            guides(color = guide_legend(title = "K clusters",
                   title.theme = element_text(size = 16,
                   colour = "black", face = "plain", angle = 0)))
dev.off()
```

```
## quartz_off_screen
##                  2
```

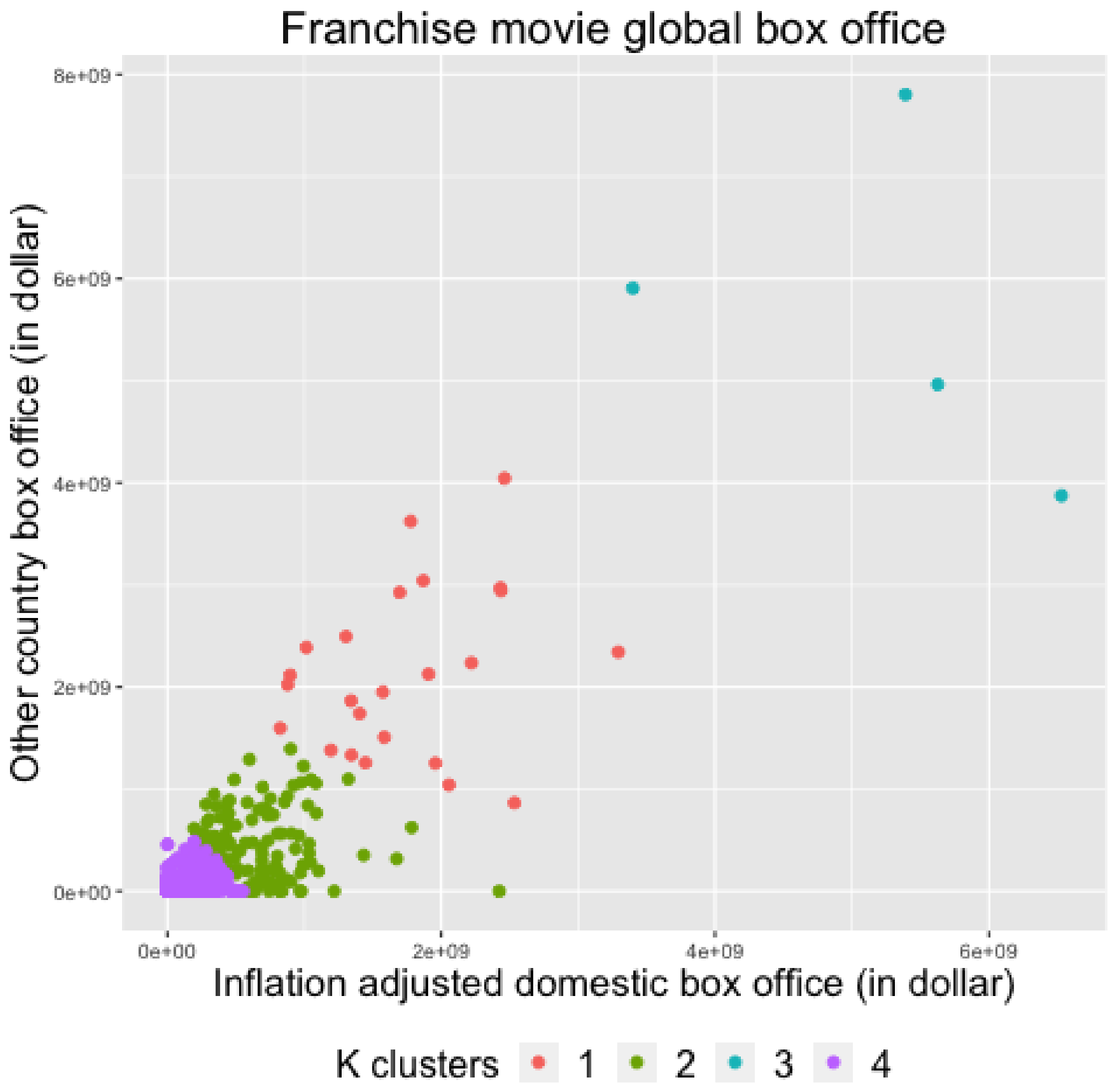
```
knitr::include_graphics(paste(working.path, "ggplot_KmeansAnalysis.png", sep
= "/"),
                         auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



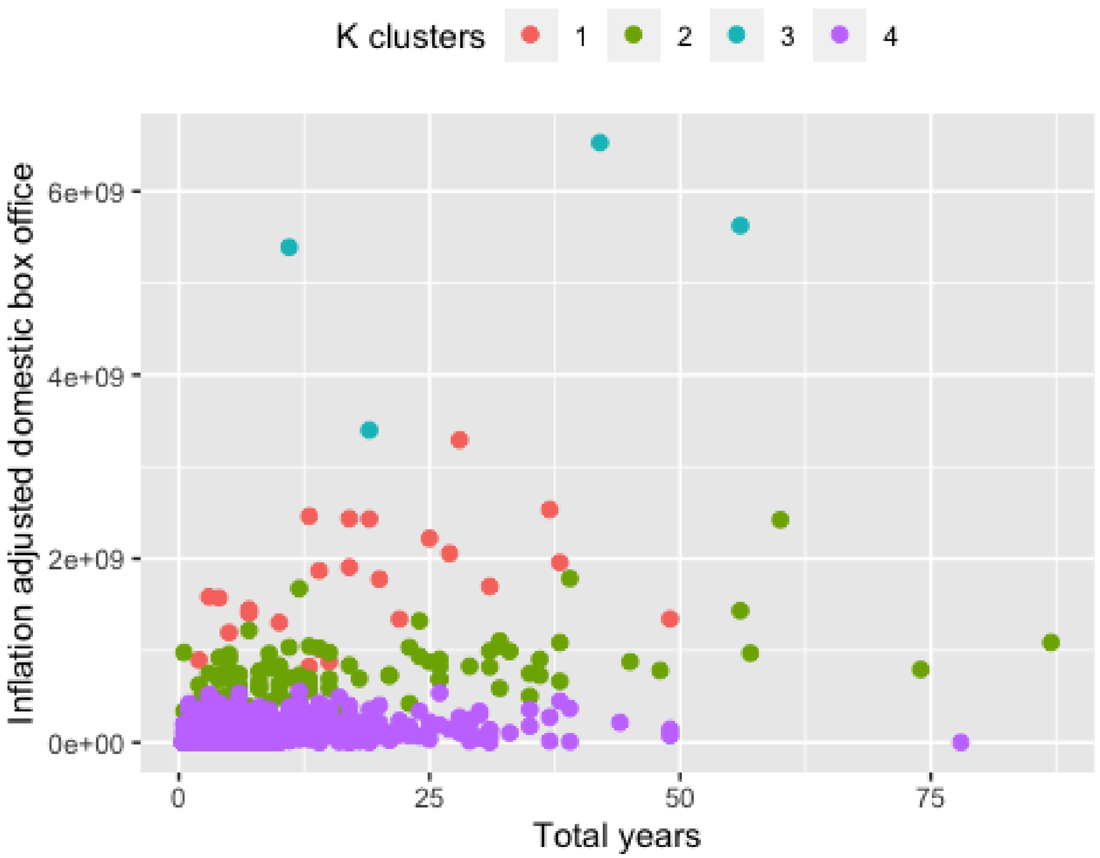
```
png("ggplot_FranchiseGlobalBox.png")
ggplot(movfran.result, aes(movfran.result$infl.adj.dom.boxoffice,
                           movfran.result$other.boxoffice,
                           col = as.factor(movfran.result$K.cluster))) +
        geom_point(shape = 19, size = 2) +
        theme(legend.position="bottom",
              legend.text = element_text(size = 16),
              plot.title = element_text(size = rel(1.75), hjust = 0.5, vjust
= 0),
              axis.title.y = element_text(size = rel(1.5), angle = 90),
              axis.title.x = element_text(size = rel(1.5), angle = 0)) +
        labs(title = "Franchise movie global box office",  par(adj = 1)) +
        xlab("Inflation adjusted domestic box office (in dollar)") +
        ylab("Other country box office (in dollar)") +
        guides(color = guide_legend(title = "K clusters",
              title.theme = element_text(size = 16,
              colour = "black", face = "plain", angle = 0)))
dev.off()
```

```
## quartz_off_screen
##                  2
```

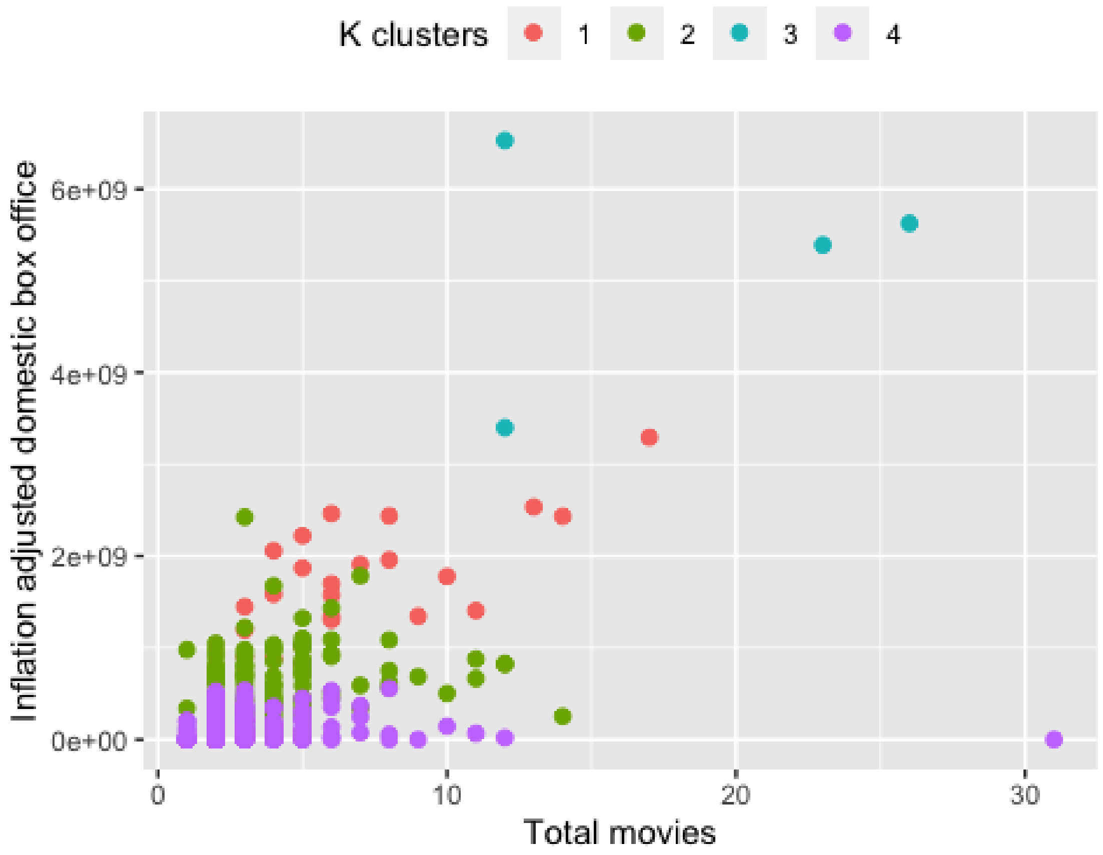
```
knitr::include_graphics(paste(working.path, "ggplot_FranchiseGlobalBox.png",
sep = "/"),
                         auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



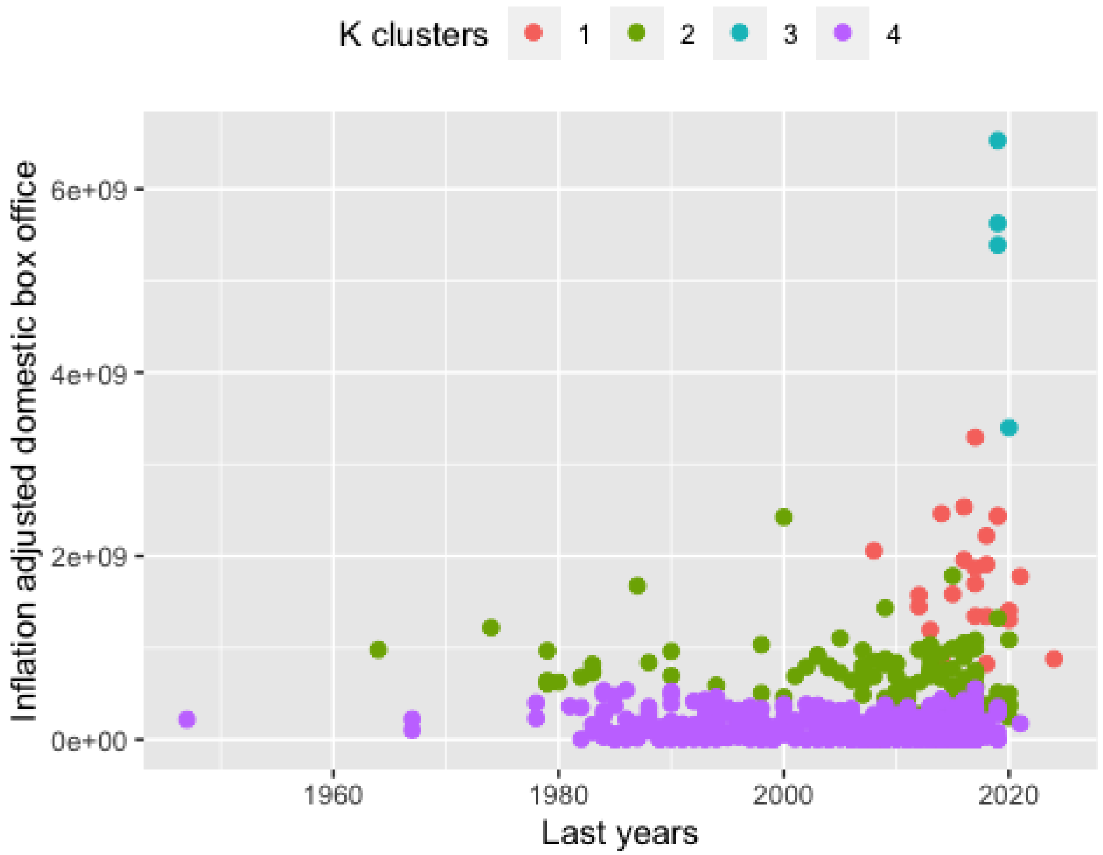
```
# Visualize the individual graphs
ggplot(movfran.result, aes(movfran.result$tot.years,
                           movfran.result$infl.adj.dom.boxoffice,
                           col = as.factor(movfran.result$K.cluster))) +
  geom_point(shape = 19, size = 2) +
  ylab("Inflation adjusted domestic box office") +
  xlab("Total years") +
  theme(legend.position="top") +
  guides(color = guide_legend(title = "K clusters" ))
```



```
ggplot(movfran.result, aes(movfran.result$tot.movies,
                           movfran.result$infl.adj.dom.boxoffice,
                           col = as.factor(movfran.result$K.cluster))) +
  geom_point(shape = 19, size = 2) +
  ylab("Inflation adjusted domestic box office") +
  xlab("Total movies") +
  theme(legend.position="top") +
  guides(color = guide_legend(title = "K clusters" ))
```



```
ggplot(movfran.result, aes(movfran.result$Last.Year,
                           movfran.result$infl.adj.dom.boxoffice,
                           col = as.factor(movfran.result$K.cluster))) +
  geom_point(shape = 19, size = 2) +
  ylab("Inflation adjusted domestic box office") +
  xlab("Last years") +
  theme(legend.position="top") +
  guides(color = guide_legend(title = "K clusters" ))
```



```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)

options(width = 350)
################################################################################
#
#              Regression modeling, to explore movie meta data
#
################################################################################
#

## Check and remove NA values left the cleaned movie meta data set
# check all column names, leave out text-content variables to examine NA
values
names(movie.meta.clean) # check all column names

##  [1] "color"                        "director_name"
"num_critic_for_reviews"    "duration"
"director_facebook_likes"   "actor_3_facebook_likes"    "actor_2_name"
"actor_1_facebook_likes"    "genres"                    "actor_1_name"
"movie_title"               "num_voted_users"
## [13] "cast_total_facebook_likes" "actor_3_name"
"facenumber_in_poster"      "plot_keywords"             "movie_imdb_link"
"num_user_for_reviews"      "language"                  "country"
```

```
"content_rating"            "budget"
"actor_2_facebook_likes"    "imdb_score"
## [25] "aspect_ratio"              "movie_facebook_likes"      "movie.gross"
"movie.year"                "adjust.gross"              "movie.names.clean"

# further subset the cleaned data set for regression
# leave out repeat and unnecessary variables
movie.regrset= movie.meta.clean[, -c(11,27)]

# display the col names for movie.regrset
names(movie.regrset)

##  [1] "color"                     "director_name"
"num_critic_for_reviews"    "duration"
"director_facebook_likes"   "actor_3_facebook_likes"    "actor_2_name"
"actor_1_facebook_likes"    "genres"                    "actor_1_name"
"num_voted_users"           "cast_total_facebook_likes"
## [13] "actor_3_name"              "facenumber_in_poster"
"plot_keywords"             "movie_imdb_link"
"num_user_for_reviews"      "language"                  "country"
"content_rating"            "budget"
"actor_2_facebook_likes"    "imdb_score"                "aspect_ratio"
## [25] "movie_facebook_likes"      "movie.year"
"adjust.gross"              "movie.names.clean"

# create an index vector for character type columns of movie.regrset
chrcol.index = c(1:2,7,9:10,13,15:16,18:20,26,28) # movie year should be
character data
str(movie.regrset[, chrcol.index]) # check whether the index vector is
correct

## 'data.frame':    4156 obs. of  13 variables:
##  $ color            : chr  "Color" "Color" "Color" "Color" ...
##  $ director_name    : chr  "Zack Snyder" "Anthony Russo" "Justin Lin"
"David Yates" ...
##  $ actor_2_name     : chr  "Lauren Cohan" "Scarlett Johansson" "Melissa
Roxburgh" "Alexander Skarsg\x92\xc7rd" ...
##  $ genres           : chr  "Action|Adventure|Sci-Fi"
"Action|Adventure|Sci-Fi" "Action|Adventure|Sci-Fi|Thriller"
"Action|Adventure|Drama|Romance" ...
##  $ actor_1_name     : chr  "Henry Cavill" "Robert Downey Jr." "Sofia
Boutella" "Christoph Waltz" ...
##  $ actor_3_name     : chr  "Alan D. Purwin" "Chris Evans" "Lydia Wilson"
"Casper Crump" ...
##  $ plot_keywords    : chr  "based on comic book|batman|sequel to a
reboot|superhero|superman" "based on comic book|knife|marvel cinematic
universe|returning character killed off|superhero" "hatred|sequel|space
opera|star trek|third part" "africa|capture|jungle|male
objectification|tarzan" ...
##  $ movie_imdb_link  : chr
"http://www.imdb.com/title/tt2975590/?ref_=fn_tt_tt_1"
```

```
"http://www.imdb.com/title/tt3498820/?ref_=fn_tt_tt_1"
"http://www.imdb.com/title/tt2660888/?ref_=fn_tt_tt_1"
"http://www.imdb.com/title/tt0918940/?ref_=fn_tt_tt_1" ...
##  $ language         : chr  "English" "English" "English" "English" ...
##  $ country          : chr  "USA" "USA" "USA" "USA" ...
##  $ content_rating   : chr  "PG-13" "PG-13" "PG-13" "PG-13" ...
##  $ movie.year       : chr  "2016" "2016" "2016" "2016" ...
##  $ movie.names.clean: chr  "Batman v Superman Dawn of Justiceξ" "Captain
America Civil Warξ" "Star Trek Beyondξ" "The Legend of Tarzanξ" ...
```

```
str(movie.regrset[, -chrcol.index])
```

```
## 'data.frame':    4156 obs. of  15 variables:
##  $ num_critic_for_reviews   : int  673 516 322 248 396 418 370 286 218 275
...
##  $ duration                 : int  183 147 122 110 144 123 106 120 113 123
...
##  $ director_facebook_likes  : int  0 94 681 282 0 452 4000 776 33 0 ...
##  $ actor_3_facebook_likes   : int  2000 11000 105 103 1000 329 591 535
11000 648 ...
##  $ actor_1_facebook_likes   : int  15000 21000 998 11000 34000 10000 19000
890 40000 3000 ...
##  $ num_voted_users          : int  371639 272670 53607 42372 148379 118992
106072 58137 21352 111609 ...
##  $ cast_total_facebook_likes: int  24450 64798 1327 21175 49684 11287
32921 3233 80806 5505 ...
##  $ facenumber_in_poster     : int  0 0 4 2 6 8 0 0 1 0 ...
##  $ num_user_for_reviews     : int  3018 1022 432 239 622 971 398 520 131
781 ...
##  $ budget                   : num  2.50e+08 2.50e+08 1.85e+08 1.80e+08
1.78e+08 1.75e+08 1.75e+08 1.65e+08 1.70e+08 1.60e+08 ...
##  $ actor_2_facebook_likes   : int  4000 19000 119 10000 13000 336 13000
812 25000 716 ...
##  $ imdb_score               : num  6.9 8.2 7.5 6.6 7.3 6.9 7.8 5.5 6.4 7.3
...
##  $ aspect_ratio             : num  2.35 2.35 2.35 2.35 2.35 2.35 1.85 2.35
1.85 2.35 ...
##  $ movie_facebook_likes     : int  197000 72000 30000 29000 54000 80000
65000 67000 30000 89000 ...
##  $ adjust.gross             : num  3.30e+08 4.07e+08 1.30e+08 1.24e+08
1.55e+08 ...
```

```
# create column length vectors for both subsets of character or numeric data
chr.col.length = length(colnames(movie.regrset[, chrcol.index]))
num.col.length = length(colnames(movie.regrset[, -chrcol.index]))

# coerce data types back and forward to change character "NA" to Null value
movie.regrset.chrcol = sapply(1:chr.col.length, simplify = T, function(j){
  as.character(as.factor(movie.regrset[, chrcol.index][,j]))})
```

```
movie.regrset.numcol = sapply(1:num.col.length, simplify = T, function(i){
  as.numeric(as.character(movie.regrset[, -chrcol.index][,i]))})

# check out new generated subsets: matrixs
head(movie.regrset.chrcol,2)
```

```
##      [,1]    [,2]            [,3]                 [,4]
[,5]                [,6]             [,7]
[,8]                                                   [,9]      [,10] [,11]
[,12]  [,13]
## [1,] "Color" "Zack Snyder"   "Lauren Cohan"       "Action|Adventure|Sci-
Fi" "Henry Cavill"      "Alan D. Purwin" "based on comic book|batman|sequel
to a reboot|superhero|superman"
"http://www.imdb.com/title/tt2975590/?ref_=fn_tt_tt_1" "English" "USA" "PG-
13" "2016" "Batman v Superman Dawn of Justiceξ"
## [2,] "Color" "Anthony Russo" "Scarlett Johansson" "Action|Adventure|Sci-
Fi" "Robert Downey Jr." "Chris Evans"    "based on comic book|knife|marvel
cinematic universe|returning character killed off|superhero"
"http://www.imdb.com/title/tt3498820/?ref_=fn_tt_tt_1" "English" "USA" "PG-
13" "2016" "Captain America Civil Warξ"
```

```
head(movie.regrset.numcol,2)
```

```
##      [,1] [,2] [,3]  [,4]  [,5]   [,6]  [,7] [,8] [,9]   [,10] [,11] [,12]
[,13]  [,14]     [,15]
## [1,]  673  183    0  2000 15000 371639 24450    0 3018 2.5e+08  4000   6.9
2.35 197000 330249062
## [2,]  516  147   94 11000 21000 272670 64798    0 1022 2.5e+08 19000   8.2
2.35  72000 407197282
```

```
# add column names to matrixs
colnames(movie.regrset.chrcol) = colnames(movie.regrset[, chrcol.index])
colnames(movie.regrset.numcol) = colnames(movie.regrset[, -chrcol.index])

# switch the cleaned movie names to the left first column and fix the column
name
movie.regrset.chrcol = cbind(movie.regrset.chrcol[, "movie.names.clean"],
                             movie.regrset.chrcol[, c(1:chr.col.length-1)])
colnames(movie.regrset.chrcol)[1] = "movie.names.clean"

# change the matrixs to data frame and combine two data frames
movie.regrset.numcol = data.frame(movie.regrset.numcol, stringsAsFactors = F)
movie.regrset.chrcol = data.frame(movie.regrset.chrcol, stringsAsFactors = T)
movie.regrset = cbind(movie.regrset.chrcol, movie.regrset.numcol)

# check out the column names again and choose column for regression
colnames(movie.regrset)
```

```
##  [1] "movie.names.clean"         "color"
"director_name"             "actor_2_name"              "genres"
"actor_1_name"              "actor_3_name"              "plot_keywords"
```

```
"movie_imdb_link"            "language"                   "country"
"content_rating"
## [13] "movie.year"                "num_critic_for_reviews"    "duration"
"director_facebook_likes"    "actor_3_facebook_likes"
"actor_1_facebook_likes"     "num_voted_users"
"cast_total_facebook_likes" "facenumber_in_poster"
"num_user_for_reviews"       "budget"
"actor_2_facebook_likes"
## [25] "imdb_score"                "aspect_ratio"
"movie_facebook_likes"       "adjust.gross"

# For regression model, keep movie names as ID
# use numeric data +/- character data of color, language, country, content
rating, year
movie.regr.01 = movie.regrset[, c(1,14:28)]
movie.regr.02 = movie.regrset[, c(1:2,10:13,14:28)]


# create matrix of NA value indices
# use non-repeat row number of the matrix to remove records with NA value
narec.regr01 = which(is.na(movie.regr.01) == T, arr.ind = T)
movie.regr.01 = movie.regr.01[-unique(narec.regr01[, "row"]), ]
narec.regr02 = which(is.na(movie.regr.02) == T, arr.ind = T)
movie.regr.02 = movie.regr.02[-unique(narec.regr02[, "row"]), ]

# 31 records that NA values are only in the text-content variables
dim(movie.regr.01)[1] - dim(movie.regr.02)[1]

## [1] 31

# regression modeling
library("stats")
#create length variables for training data sets
set.seed(3456)
train.length.01 = floor(0.7*dim(movie.regr.01)[1])
train.length.02 = floor(0.7*dim(movie.regr.02)[1])

# generate random index with sample function to create training and test data
sets
# exclude the movie names for both training and test data sets
train.ind.01 = sample(1:dim(movie.regr.01)[1], train.length.01)
train.reg.01 = movie.regr.01[train.ind.01, -1]
test.reg.01 = movie.regr.01[-train.ind.01, -1]

train.ind.02 = sample(1:dim(movie.regr.02)[1], train.length.02)
train.reg.02 = movie.regr.02[train.ind.02, -1]
test.reg.02 = movie.regr.02[-train.ind.02, -1]

#create a vector for storing the original value of adjust.gross variable
#adjust.gross is a dependent variable
```

```
adjgross.testreg01 = test.reg.01$adjust.gross
adjgross.testreg02 = test.reg.02$adjust.gross

#Then removed the existing variables for prediction
test.reg.01$adjust.gross = NULL
test.reg.02$adjust.gross = NULL

# run logistic model to the subset with only numeric variables only
# adjust.gross is the dependent variable
# "." means include everything except the dependent variable
rgmodel.01 = glm(adjust.gross ~ ., family = gaussian, data = train.reg.01)
summary.glm(rgmodel.01)
```

```
##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = train.reg.01)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -645303864   -37825008   -18934583    13718763  3216406456
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -8.375e+06  2.317e+07  -0.361 0.717824
## num_critic_for_reviews    -2.296e+04  3.288e+04  -0.698 0.484930
## duration                   6.727e+05  1.218e+05   5.524 3.63e-08 ***
## director_facebook_likes   -8.535e+02  8.912e+02  -0.958 0.338288
## actor_3_facebook_likes    -1.180e+04  3.840e+03  -3.072 0.002149 **
## actor_1_facebook_likes    -1.055e+04  2.314e+03  -4.561 5.32e-06 ***
## num_voted_users            3.926e+02  3.071e+01  12.783  < 2e-16 ***
## cast_total_facebook_likes  1.031e+04  2.309e+03   4.467 8.28e-06 ***
## facenumber_in_poster      -1.640e+06  1.286e+06  -1.275 0.202457
## num_user_for_reviews       2.079e+03  1.046e+04   0.199 0.842464
## budget                     4.965e-02  2.419e-02   2.052 0.040233 *
## actor_2_facebook_likes    -1.028e+04  2.429e+03  -4.231 2.41e-05 ***
## imdb_score                 2.534e+06  2.886e+06   0.878 0.379902
## aspect_ratio              -2.192e+07  6.646e+06  -3.298 0.000986 ***
## movie_facebook_likes      -5.329e+02  1.818e+02  -2.931 0.003409 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.637995e+16)
##
##     Null deviance: 5.4709e+19  on 2659  degrees of freedom
## Residual deviance: 4.3325e+19  on 2645  degrees of freedom
## AIC: 106876
##
## Number of Fisher Scoring iterations: 2
```

```
names(train.reg.01)
```

```
##  [1] "num_critic_for_reviews"    "duration"
"director_facebook_likes"   "actor_3_facebook_likes"
"actor_1_facebook_likes"    "num_voted_users"
"cast_total_facebook_likes" "facenumber_in_poster"
"num_user_for_reviews"      "budget"
"actor_2_facebook_likes"    "imdb_score"
## [13] "aspect_ratio"              "movie_facebook_likes"
"adjust.gross"

# Observation to model 01
# imdb_score appear to be insignificant
# num_user_for_reviews and num_critic_for_reviews appear to be less
significant

# remove imdb_score and num_user_for_reviews
rgmodel.01mo = glm(adjust.gross ~ ., family = gaussian, data =
train.reg.01[,-c(1,9,12)])
summary.glm(rgmodel.01mo)

##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = train.reg.01[,
##     -c(1, 9, 12)])
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -634531928   -37693481   -19145774    12779490  3216991818
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                3.485e+06  1.760e+07   0.198 0.843034
## duration                   7.056e+05  1.152e+05   6.126 1.03e-09 ***
## director_facebook_likes   -8.237e+02  8.887e+02  -0.927 0.354088
## actor_3_facebook_likes    -1.153e+04  3.826e+03  -3.013 0.002609 **
## actor_1_facebook_likes    -1.031e+04  2.293e+03  -4.494 7.29e-06 ***
## num_voted_users            3.983e+02  2.077e+01  19.182  < 2e-16 ***
## cast_total_facebook_likes  1.006e+04  2.288e+03   4.397 1.14e-05 ***
## facenumber_in_poster      -1.696e+06  1.276e+06  -1.329 0.183893
## budget                     4.693e-02  2.402e-02   1.954 0.050805 .
## actor_2_facebook_likes    -1.005e+04  2.409e+03  -4.173 3.10e-05 ***
## aspect_ratio              -2.273e+07  6.584e+06  -3.453 0.000563 ***
## movie_facebook_likes      -6.027e+02  1.435e+02  -4.199 2.77e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.636843e+16)
##
##     Null deviance: 5.4709e+19  on 2659  degrees of freedom
## Residual deviance: 4.3344e+19  on 2648  degrees of freedom
## AIC: 106872
```

```
##
## Number of Fisher Scoring iterations: 2

#predict response variable, the predicted values are probabilities
pred.reg.01 <- predict(rgmodel.01mo, newdata = test.reg.01[,-c(1,9,12)], type
= "response")


# run logistic model to the subset of combined numeric & several categorical
variables
rgmodel.02 = glm(adjust.gross ~ ., family = gaussian, data = train.reg.02)
# create an index of sorted Estimate/P-value, to list important variables on
top
rgmodel.02.PvalueSort = order(summary.glm(rgmodel.02)[12]$coefficients[,4],
decreasing = FALSE)

#list logistic linear regression modeling statistics
summary.glm(rgmodel.02)[c(1,3:4)]

## $call
## glm(formula = adjust.gross ~ ., family = gaussian, data = train.reg.02)
##
## $family
##
## Family: gaussian
## Link function: identity
##
##
## $deviance
## [1] 1.463139e+19

rgmodel.02.coefficient =
summary.glm(rgmodel.02)[12]$coefficients[rgmodel.02.PvalueSort[1:65],]
knitr::kable(rgmodel.02.coefficient, digits = 4, format.args =
list(scientific = TRUE), "simple")
```

	Estimate	Std. Error	t value	Pr(>|t|)
movie.year1939	2.147252e+09	1.049004e+08	2.04694e+01	0.000e+00
num_voted_users	3.444402e+02	2.048800e+01	1.68118e+01	0.000e+00
content_ratingG	8.078037e+08	4.963908e+07	1.62735e+01	0.000e+00
content_ratingPG	7.705069e+08	4.963585e+07	1.55232e+01	0.000e+00
content_ratingPG-13	7.433225e+08	4.964663e+07	1.49723e+01	0.000e+00
content_ratingUnrated	7.434255e+08	5.166848e+07	1.43884e+01	0.000e+00
content_ratingR	7.074366e+08	4.962365e+07	1.42560e+01	0.000e+00
content_ratingNC-17	7.005515e+08	5.543222e+07	1.26380e+01	0.000e+00
movie.year1965	9.571191e+08	9.801735e+07	9.76480e+00	0.000e+00

	Estimate	Std. Error	t value	Pr(>|t|)
movie.year1963	1.038986e+09	1.138135e+08	9.12880e+00	0.000e+00
movie.year1973	1.005161e+09	1.152786e+08	8.71940e+00	0.000e+00
movie.year1967	1.034741e+09	1.258872e+08	8.21960e+00	0.000e+00
movie.year1964	8.342897e+08	1.074443e+08	7.76490e+00	0.000e+00
actor_1_facebook_likes	-1.118205e+04	1.481768e+03	-7.54640e+00	0.000e+00
cast_total_facebook_likes	1.111970e+04	1.481339e+03	7.50650e+00	0.000e+00
movie_facebook_likes	-8.101617e+02	1.144682e+02	-7.07760e+00	0.000e+00
actor_2_facebook_likes	-1.076712e+04	1.602624e+03	-6.71840e+00	0.000e+00
movie.year1929	8.403409e+08	1.253792e+08	6.70240e+00	0.000e+00
movie.year1948	7.791357e+08	1.257064e+08	6.19810e+00	0.000e+00
movie.year1966	7.960300e+08	1.340164e+08	5.93980e+00	0.000e+00
num_critic_for_reviews	1.518952e+05	2.590140e+04	5.86440e+00	0.000e+00
actor_3_facebook_likes	-1.340300e+04	2.473241e+03	-5.41920e+00	0.000e+00
(Intercept)	-7.954990e+08	1.577246e+08	-5.04360e+00	0.000e+00
movie.year1977	4.528441e+08	9.436295e+07	4.79900e+00	0.000e+00
colorColor	4.325911e+07	9.180939e+06	4.71180e+00	0.000e+00
duration	3.466479e+05	8.355978e+04	4.14850e+00	0.000e+00
movie.year1974	3.671283e+08	9.681493e+07	3.79210e+00	2.000e-04
movie.year1962	3.952299e+08	1.051388e+08	3.75910e+00	2.000e-04
director_facebook_likes	-1.912640e+03	5.266796e+02	-3.63150e+00	3.000e-04
movie.year1969	3.382613e+08	1.019394e+08	3.31830e+00	9.000e-04
movie.year1953	3.107981e+08	1.144674e+08	2.71520e+00	6.700e-03
movie.year1946	2.947734e+08	1.141923e+08	2.58140e+00	9.900e-03
budget	1.610000e-02	7.600000e-03	2.11140e+00	3.480e-02
movie.year1978	1.874586e+08	9.436508e+07	1.98650e+00	4.710e-02
movie.year1971	2.184432e+08	1.155273e+08	1.89080e+00	5.880e-02
imdb_score	-3.471006e+06	1.923764e+06	-1.80430e+00	7.130e-02
countryIran	-2.341014e+08	1.387202e+08	-1.68760e+00	9.160e-02
movie.year1960	1.881850e+08	1.152326e+08	1.63310e+00	1.026e-01
num_user_for_reviews	1.187363e+04	7.319340e+03	1.62220e+00	1.049e-01
languageItalian	-1.249864e+08	7.712370e+07	-1.62060e+00	1.052e-01
movie.year1981	1.426888e+08	8.960739e+07	1.59240e+00	1.114e-01
facenumber_in_poster	-1.204400e+06	7.627791e+05	-1.57900e+00	1.145e-01
movie.year1959	1.775279e+08	1.140975e+08	1.55590e+00	1.199e-01
movie.year1980	1.374307e+08	8.923598e+07	1.54010e+00	1.237e-01

	Estimate	Std. Error	t value	Pr(>|t|)
movie.year1990	1.345950e+08	8.859018e+07	1.51930e+00	1.288e-01
countryWest Germany	-2.085584e+08	1.412611e+08	-1.47640e+00	1.400e-01
languagePersian	1.642709e+08	1.113792e+08	1.47490e+00	1.404e-01
movie.year1984	1.232050e+08	8.827015e+07	1.39580e+00	1.629e-01
movie.year1957	1.612754e+08	1.156204e+08	1.39490e+00	1.632e-01
movie.year1985	1.185082e+08	8.906648e+07	1.33060e+00	1.835e-01
movie.year1986	1.061760e+08	8.854881e+07	1.19910e+00	2.306e-01
movie.year1983	1.024320e+08	8.930303e+07	1.14700e+00	2.515e-01
movie.year1968	1.077962e+08	1.025911e+08	1.05070e+00	2.935e-01
movie.year1992	9.031499e+07	8.755830e+07	1.03150e+00	3.024e-01
languageJapanese	-6.627992e+07	6.766928e+07	-9.79500e-01	3.274e-01
countryGreece	-1.204367e+08	1.338718e+08	-8.99600e-01	3.684e-01
movie.year1975	9.165692e+07	1.021704e+08	8.97100e-01	3.698e-01
movie.year1989	7.411740e+07	8.760823e+07	8.46000e-01	3.976e-01
movie.year1982	7.406720e+07	8.877235e+07	8.34300e-01	4.042e-01
movie.year1979	7.915515e+07	9.702253e+07	8.15800e-01	4.147e-01
movie.year1991	6.822476e+07	8.758902e+07	7.78900e-01	4.361e-01
languageKorean	-6.573648e+07	8.649465e+07	-7.60000e-01	4.473e-01
countryItaly	-8.600700e+07	1.158626e+08	-7.42300e-01	4.580e-01
movie.year1996	5.722028e+07	8.672720e+07	6.59800e-01	5.095e-01
movie.year1997	5.710557e+07	8.670908e+07	6.58600e-01	5.102e-01

```
# Observation to model 02:
# 1.Country Brazil,Hungary,India,Indonesia,Iran,Netherlands,New Zealand,
Norway has NA coefficient
# 2.Except Japan and Greece has P-value around 0.2, other countries have P-
value over 0.5
# 3. aspect_ratio appears to be very irrelevant
# 4. the reference selected by glm make data interpretation difficult
#    (due to too many levels in languages and years)

# Improvement: remove irrelevant variables and reference the categorical
variables
# To remove irrelevant variables
train.reg.02 = movie.regr.02[train.ind.02, -c(1,4,19)]
test.reg.02 = movie.regr.02[-train.ind.02, -c(1,4,19)]

# Specify reference levels for each categorical variables
train.reg.02$color = relevel(train.reg.02$color, ref = "Color")
train.reg.02$language = relevel(train.reg.02$language, ref = "English")
train.reg.02$content_rating = relevel(train.reg.02$content_rating, ref = "R")
```

```
# calculate sum of adj.gross for specifying movie.year reference
gros.ansum = aggregate(train.reg.02$adjust.gross, by =
list(train.reg.02$movie.year), FUN = sum)
gros.ansum[order(gros.ansum$x, decreasing = T),] # 2009 is the most
profitable year
```

```
##    Group.1            x
## 62    2012 8999595415.9
## 59    2009 8720961267.2
## 63    2013 7713742865.4
## 65    2015 7464209226.8
## 51    2001 7393579650.5
## 50    2000 7384407875.3
## 53    2003 7377413160.6
## 52    2002 7346128197.0
## 64    2014 7340105823.8
## 55    2005 7117368182.1
## 60    2010 7038461735.6
## 61    2011 6652025593.3
## 54    2004 6651489694.5
## 58    2008 6630872810.5
## 57    2007 6584979465.4
## 49    1999 6063072644.1
## 47    1997 6037057291.3
## 56    2006 5719723160.6
## 48    1998 4942624415.4
## 46    1996 4224140067.3
## 4     1939 3813500752.7
## 44    1994 3407924246.3
## 66    2016 3221301704.0
## 43    1993 3066106128.2
## 42    1992 2991769105.1
## 34    1984 2988447879.6
## 40    1990 2685569473.8
## 15    1965 2579523383.2
## 45    1995 2451176552.0
## 41    1991 2256385506.4
## 27    1977 2216056697.8
## 30    1980 2058589979.5
## 39    1989 1961000126.4
## 33    1983 1813856626.1
## 36    1986 1539921966.2
## 32    1982 1534686701.0
## 31    1981 1478247784.2
## 35    1985 1432593320.3
## 37    1987 1413698844.3
## 24    1974 1194454093.7
## 38    1988 1147789848.6
## 23    1973 1105793615.3
## 28    1978 1097485884.7
## 14    1964  819124077.4
## 19    1969  818178116.4
## 13    1963  647472562.1
## 18    1968  507019458.7
## 25    1975  505128496.5
## 29    1979  477701873.3
## 7     1953  323606292.1
## 17    1967  309710922.2
## 5     1946  291086625.6
## 21    1971  259564207.4
## 11    1960  259468108.1
## 9     1957  232320911.0
## 10    1959  206192439.9
## 20    1970  202893360.8
## 12    1962  175373011.1
## 26    1976  105451669.6
## 8     1954   88054036.9
## 16    1966   45186691.4
## 2     1929   39411840.0
## 6     1948   29438325.6
## 3     1936    2818712.7
## 22    1972    1036300.6
## 1     1927     364632.8
```

```
train.reg.02$movie.year = relevel(train.reg.02$movie.year, ref = "2009")

# re-run logistic model to the subset of combined numeric & several
categorical variables
rgmodel.02 = glm(adjust.gross ~ ., family = gaussian, data = train.reg.02)

# create an index of sorted Estimate/P-value, to list important variables on
top
rgmodel.02.Rerun.PvalueSort =
order(summary.glm(rgmodel.02)[12]$coefficients[,4], decreasing = FALSE)

#list logistic linear regression modeling statistics
summary.glm(rgmodel.02)[c(1,3:4)]
```

```
## $call
## glm(formula = adjust.gross ~ ., family = gaussian, data = train.reg.02)
##
## $family
##
## Family: gaussian
## Link function: identity
##
##
## $deviance
## [1] 1.494226e+19
```

```
rgmodel.02.Rerun.coefficient =
summary.glm(rgmodel.02)[12]$coefficients[rgmodel.02.Rerun.PvalueSort[1:47],]
knitr::kable(rgmodel.02.Rerun.coefficient, digits = 4, format.args =
list(scientific = TRUE), "simple")
```

	Estimate	Std. Error	t value	Pr(>|t|)
movie.year1939	2.143408e+09	6.108340e+07	3.50899e+01	0.0e+00
movie.year1965	9.470390e+08	4.709384e+07	2.01096e+01	0.0e+00
num_voted_users	3.524513e+02	2.041460e+01	1.72646e+01	0.0e+00
content_ratingApproved	-7.146629e+08	4.977762e+07	-1.43571e+01	0.0e+00
movie.year1963	1.009462e+09	7.470329e+07	1.35130e+01	0.0e+00
content_ratingPG	6.267118e+07	4.748339e+06	1.31985e+01	0.0e+00
movie.year1973	9.968493e+08	7.736001e+07	1.28858e+01	0.0e+00
movie.year1964	8.245938e+08	6.422512e+07	1.28391e+01	0.0e+00
movie.year1967	1.003204e+09	9.193176e+07	1.09125e+01	0.0e+00
movie.year1977	4.295242e+08	3.957374e+07	1.08538e+01	0.0e+00
content_ratingPG-13	3.635361e+07	3.552456e+06	1.02334e+01	0.0e+00
content_ratingG	1.006592e+08	1.044897e+07	9.63340e+00	0.0e+00
movie.year1929	8.376460e+08	9.226935e+07	9.07830e+00	0.0e+00
movie.year1948	7.768180e+08	9.197448e+07	8.44600e+00	0.0e+00
actor_1_facebook_likes	-1.197316e+04	1.478543e+03	-8.09790e+00	0.0e+00
cast_total_facebook_likes	1.192047e+04	1.477846e+03	8.06610e+00	0.0e+00
movie.year1974	3.582020e+08	4.531533e+07	7.90470e+00	0.0e+00
movie.year1966	7.847977e+08	1.024027e+08	7.66380e+00	0.0e+00
movie_facebook_likes	-8.274478e+02	1.140317e+02	-7.25630e+00	0.0e+00
actor_2_facebook_likes	-1.146408e+04	1.601838e+03	-7.15680e+00	0.0e+00
movie.year1962	3.619827e+08	6.077438e+07	5.95620e+00	0.0e+00
actor_3_facebook_likes	-1.456276e+04	2.465999e+03	-5.90540e+00	0.0e+00
movie.year1969	3.161384e+08	5.504850e+07	5.74290e+00	0.0e+00
movie.year1990	1.252637e+08	2.191415e+07	5.71610e+00	0.0e+00
num_critic_for_reviews	1.451276e+05	2.566488e+04	5.65470e+00	0.0e+00
movie.year1980	1.288969e+08	2.453976e+07	5.25260e+00	0.0e+00
movie.year1984	1.030836e+08	2.031611e+07	5.07400e+00	0.0e+00
colorBlack and White	-4.408316e+07	9.141498e+06	-4.82230e+00	0.0e+00
languageItalian	-1.812079e+08	3.773432e+07	-4.80220e+00	0.0e+00
movie.year1992	7.615584e+07	1.697636e+07	4.48600e+00	0.0e+00
movie.year1981	1.088960e+08	2.470299e+07	4.40820e+00	0.0e+00
movie.year1985	1.026716e+08	2.442090e+07	4.20430e+00	0.0e+00

	Estimate	Std. Error	t value	Pr(>|t|)
movie.year1986	8.800750e+07	2.183255e+07	4.03100e+00	1.0e-04
movie.year1953	3.113686e+08	7.842012e+07	3.97050e+00	1.0e-04
movie.year1978	1.370777e+08	3.537578e+07	3.87490e+00	1.0e-04
duration	3.072542e+05	8.120626e+04	3.78360e+00	2.0e-04
movie.year1946	2.969423e+08	7.907484e+07	3.75520e+00	2.0e-04
movie.year1997	4.371039e+07	1.185793e+07	3.68620e+00	2.0e-04
movie.year1996	4.416808e+07	1.210084e+07	3.65000e+00	3.0e-04
movie.year1989	6.216852e+07	1.752133e+07	3.54820e+00	4.0e-04
movie.year1983	8.507105e+07	2.455023e+07	3.46520e+00	5.0e-04
movie.year1991	5.770478e+07	1.743860e+07	3.30900e+00	9.0e-04
director_facebook_likes	-1.674105e+03	5.256587e+02	-3.18480e+00	1.5e-03
movie.year1982	6.558257e+07	2.249233e+07	2.91580e+00	3.6e-03
movie.year1995	4.161556e+07	1.496468e+07	2.78090e+00	5.5e-03
(Intercept)	-4.022286e+07	1.457270e+07	-2.76020e+00	5.8e-03
languageJapanese	-7.306219e+07	2.680394e+07	-2.72580e+00	6.5e-03

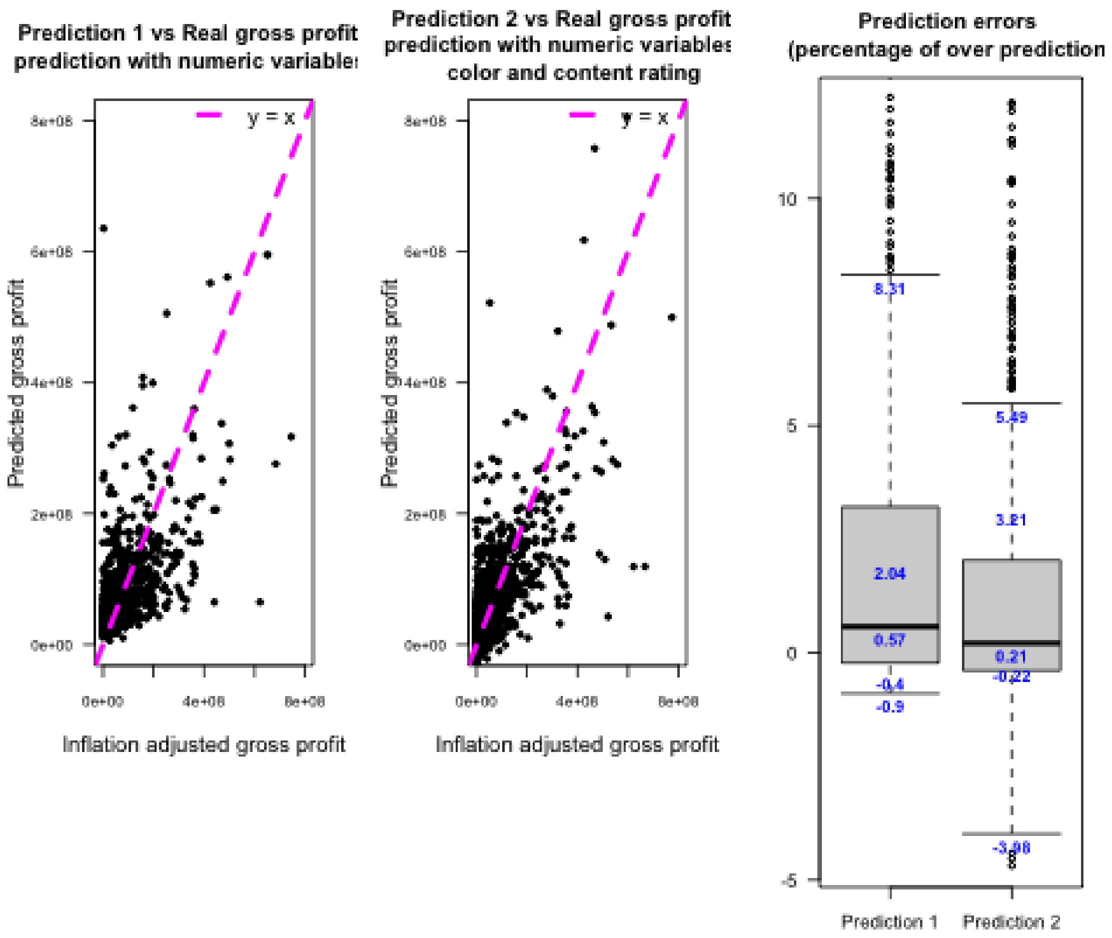
```
# Observation to 2nd round model 02:
# 1. Not enough data for language levels and movie.years
# 2. budget appears insignificant in this model
# 3. num_critic_for_reviews appears less significant
# not enough data records for language levels and movie years
# re-run logistic model to the subset of numeric data with "color" and
"content_rating"
rgmodel.02mo = glm(adjust.gross ~ ., family = gaussian, data = train.reg.02[,
-c(2,4,14,5)])
summary.glm(rgmodel.02mo)

##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = train.reg.02[,
##     -c(2, 4, 14, 5)])
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -628777779   -33867064    -7219643    18853844  3093227218
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -9.201e+07  1.604e+07  -5.736 1.08e-08 ***
## colorBlack and White      -2.669e+07  1.130e+07  -2.362  0.01826 *
## content_ratingApproved     1.947e+08  3.094e+07   6.291 3.69e-10 ***
## content_ratingG            1.500e+08  1.299e+07  11.544  < 2e-16 ***
## content_ratingNC-17        2.364e+07  2.965e+07   0.797  0.42541
## content_ratingPG           7.379e+07  5.970e+06  12.360  < 2e-16 ***
## content_ratingPG-13        3.389e+07  4.600e+06   7.369 2.30e-13 ***
## content_ratingUnrated      1.663e+07  1.592e+07   1.045  0.29626
## duration                   8.315e+05  9.851e+04   8.441  < 2e-16 ***
## director_facebook_likes   -1.853e+03  6.797e+02  -2.727  0.00644 **
## actor_3_facebook_likes    -1.476e+04  3.234e+03  -4.563 5.28e-06 ***
## actor_1_facebook_likes    -1.219e+04  1.935e+03  -6.300 3.47e-10 ***
## num_voted_users            3.696e+02  2.555e+01  14.467  < 2e-16 ***
## cast_total_facebook_likes  1.205e+04  1.933e+03   6.232 5.33e-10 ***
## facenumber_in_poster      -9.896e+05  9.794e+05  -1.010  0.31238
## num_user_for_reviews       1.704e+04  8.370e+03   2.036  0.04181 *
## actor_2_facebook_likes    -1.195e+04  2.096e+03  -5.700 1.33e-08 ***
## imdb_score                 2.063e+05  2.355e+06   0.088  0.93020
## movie_facebook_likes      -5.674e+02  1.094e+02  -5.185 2.32e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.032267e+16)
##
##     Null deviance: 4.1571e+19  on 2638  degrees of freedom
## Residual deviance: 2.7045e+19  on 2620  degrees of freedom
## AIC: 104818
##
## Number of Fisher Scoring iterations: 2
```

```r
pred.reg.02 <- predict(rgmodel.02mo, newdata = test.reg.02[, -c(2,4,14,5)],
type = "response")

# visualize prediction vs actual data
png("plot_MovieMetaProfitPrediction.png")
par(mfrow = c(1,3))
par(mar = c(20, 5, 5, 2))
plot(adjgross.testreg01, pred.reg.01, type = "p", pch = 20,
     xlim = c(0, 8e+8), ylim = c(0, 8e+8), las = 1,
     xlab = "Inflation adjusted gross profit",
     ylab = "Predicted gross profit", cex.axis = 0.8, cex.lab = 1.2)
title(main = "Prediction 1 vs Real gross profit \nprediction with numeric
variables",
      line = 1.5, adj = 0.6, cex.main = 1.2)
abline(a = 0, b= 1, col = "magenta", lty = 2, lwd = 3)
legend(5.5e+08, 8.5e+08, "y = x", xjust = 0.5, col = "magenta",
       lty = 2, lwd = 3, bty = "n", x.intersp = 0.5, cex = 1.2)

plot(adjgross.testreg02, pred.reg.02, type = "p", pch = 20,
     xlim = c(0, 8e+8), ylim = c(0, 8e+8), las = 1,
     xlab = "Inflation adjusted gross profit",
     ylab = "Predicted gross profit", cex.axis = 0.8, cex.lab = 1.2)
title(main = "Prediction 2 vs Real gross profit \nprediction with numeric
variables,\ncolor and content rating",
      line = 1, adj = 0.6, cex.main = 1.2)
abline(a = 0, b= 1, col = "magenta", lty = 2, lwd = 3)
```

```
legend(5.5e+08, 8.5e+08, "y = x", xjust = 0.5, col = "magenta",
       lty = 2, lwd = 3, bty = "n", x.intersp = 0.5, cex = 1.2)

# standard errors from two prediction
pred.error01 = pred.reg.01/adjgross.testreg01 - 1
pred.error02 = pred.reg.02/adjgross.testreg02 - 1

pred.erorlist = list(pred.error01, pred.error02)
bop01 = boxplot(pred.erorlist, range = 1.5, ylim = c(-4.5, 12), horizontal =
F,
        axes = T,staplewex = 1, las = 1, par(mar = c(10, 4, 4, 1)),
        names = c("Prediction 1", "Prediction 2"), cex.lab = 1.2, cex.main =
1.2,
        main = "Prediction errors \n(percentage of over prediction)")
text(unique(bop01$group),  bop01$stats, pos = 1, offset = 0.4,
     labels = round(bop01$stats, 2), col = "blue", cex = 0.9, font = 2)
dev.off()
```

```
## quartz_off_screen
##                  2
```

```
knitr::include_graphics(paste(working.path,
"plot_MovieMetaProfitPrediction.png", sep = "/"))
```



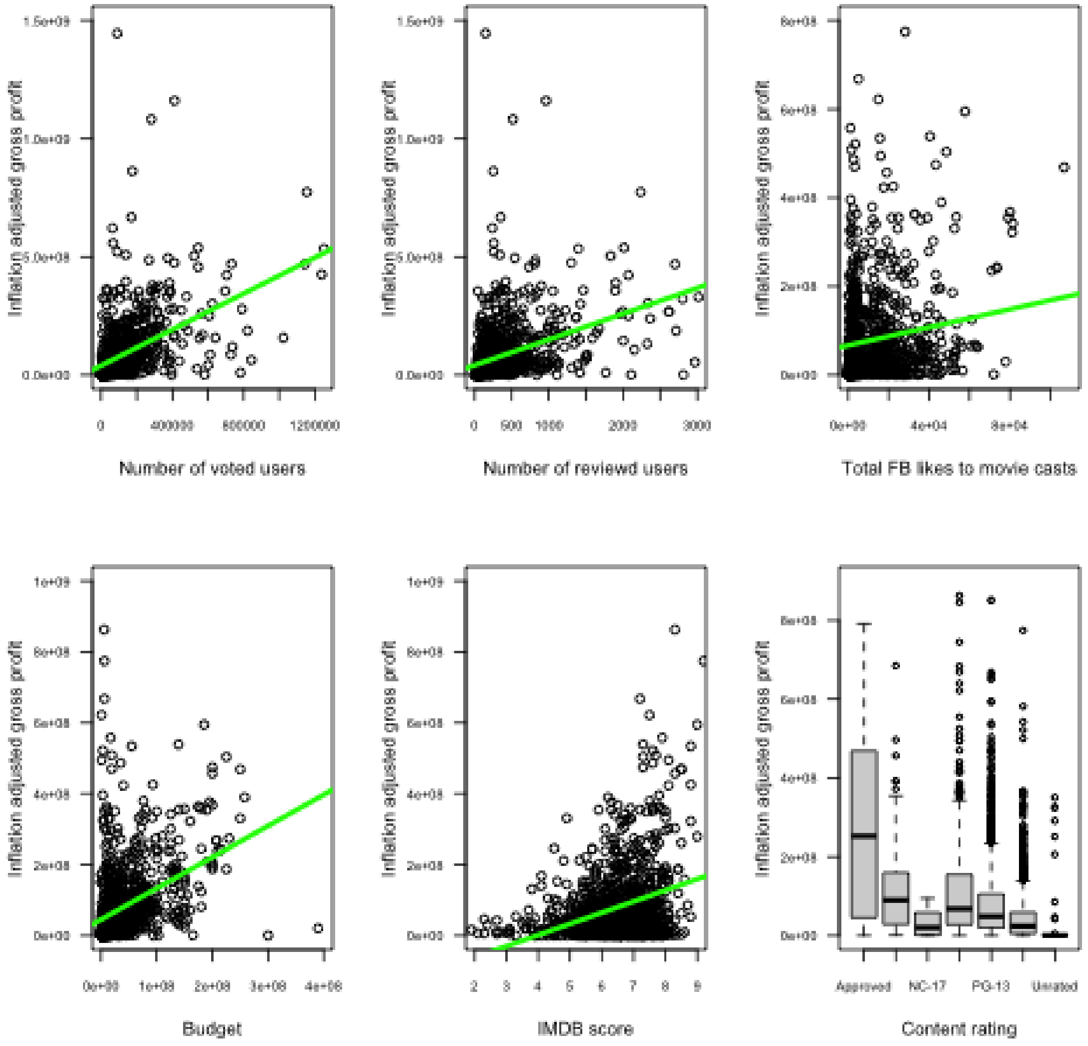
```
colnames(bop01$stats) = c("Prediction 1", "Prediction 2")
bop01$stats  # show the errors box plot statistics for both predictions
```

```
##      Prediction 1 Prediction 2
## [1,]   -0.8955211   -3.9836087
## [2,]   -0.2195871   -0.3961334
## [3,]    0.5739881    0.2108951
## [4,]    3.2128661    2.0436879
## [5,]    8.3136637    5.4874706
```

```
# visualize the relationship between individual predictors and movie gross
profit
png("plot_MovieMetaData_ProfitContributors.png")
par(mfrow=c(2,3))
par(mar = c(4,4,4,2))
plot(test.reg.02$num_voted_users, adjgross.testreg02,
     xlim = c(0,1250000), ylim = c(0,1.5e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Number of voted users")
abline(lm(adjgross.testreg02 ~ test.reg.02$num_voted_users),
       col = "green", lwd = 3)

plot(test.reg.02$num_user_for_reviews, adjgross.testreg02,
     xlim = c(0,3000), ylim = c(0,1.5e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Number of reviewd users")
abline(lm(adjgross.testreg02 ~ test.reg.02$num_user_for_reviews),
       col = "green", lwd = 3)


plot(test.reg.02$cast_total_facebook_likes, adjgross.testreg02,
     xlim = c(0,110000), ylim = c(0,8e+08),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Total FB likes to movie casts")
abline(lm(adjgross.testreg02 ~ test.reg.02$cast_total_facebook_likes),
       col = "green", lwd = 3)


plot(test.reg.02$budget, adjgross.testreg02,
     xlim = c(0,4e+08), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Budget")
abline(lm(adjgross.testreg02 ~ test.reg.02$budget),
       col = "green", lwd = 3)


plot(test.reg.02$imdb_score, adjgross.testreg02,
     xlim = c(2,9), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "IMDB score")
abline(lm(adjgross.testreg02 ~ test.reg.02$imdb_score),
       col = "green", lwd = 3)


plot(movie.regr.02$content_rating, movie.regr.02$adjust.gross,
     ylim = c(0,9e+08),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Content rating")
dev.off()
```

```
## quartz_off_screen
##                 2
```

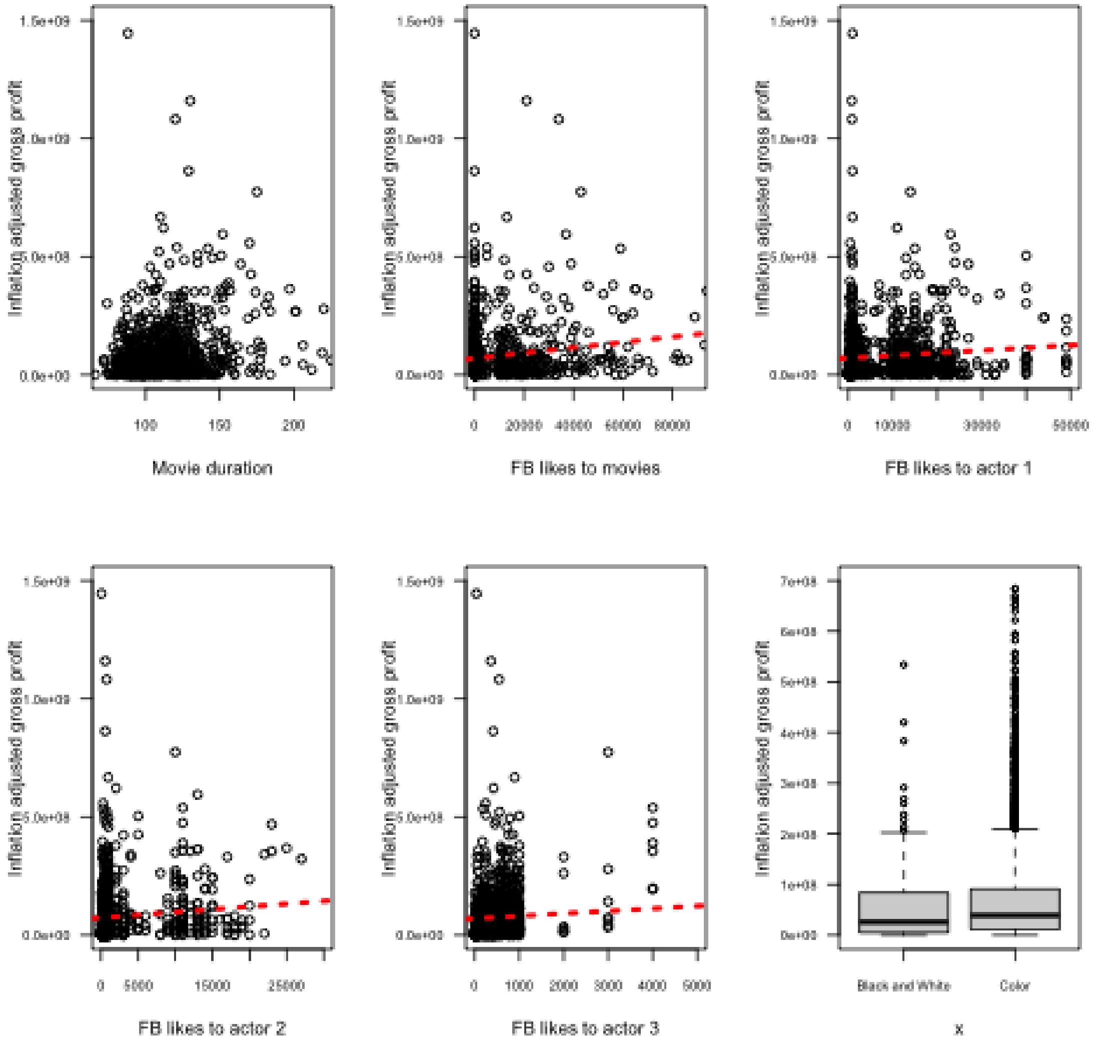
```
knitr::include_graphics(paste(working.path,
"plot_MovieMetaData_ProfitContributors.png", sep = "/"))
```



```
png("plot_MovieMetaData_NotRealContributors.png")
par(mfrow = c(2,3))
par(mar = c(4,4,4,2))

plot(test.reg.02$duration, adjgross.testreg02,
     xlim = c(70,220), ylim = c(0,1.5e+09),
     las = 1, cex.axis = 0.75, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Movie duration")

plot(test.reg.02$movie_facebook_likes, adjgross.testreg02,
     xlim = c(0,90000), ylim = c(0,1.5e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "FB likes to movies")
abline(lm(adjgross.testreg02 ~ test.reg.02$movie_facebook_likes),
       col = "red", lwd = 3, lty = 3)

plot(test.reg.02$actor_1_facebook_likes, adjgross.testreg02,
     xlim = c(0,50000), ylim = c(0,1.5e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "FB likes to actor 1")
abline(lm(adjgross.testreg02 ~ test.reg.02$actor_1_facebook_likes),
       col = "red", lwd = 3, lty = 3)

plot(test.reg.02$actor_2_facebook_likes, adjgross.testreg02,
     xlim = c(0,30000), ylim = c(0,1.5e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "FB likes to actor 2")
abline(lm(adjgross.testreg02 ~ test.reg.02$actor_2_facebook_likes),
       col = "red", lwd = 3, lty = 3)

plot(test.reg.02$actor_3_facebook_likes, adjgross.testreg02,
    xlim = c(0,5000), ylim = c(0,1.5e+09),
    las = 1, cex.axis = 0.7, cex.lab = 1,
    ylab = "Inflation adjusted gross profit",
    xlab = "FB likes to actor 3")
abline(lm(adjgross.testreg02 ~ test.reg.02$actor_3_facebook_likes),
       col = "red", lwd = 3, lty = 3)

plot(movie.regr.02$color, movie.regr.02$adjust.gross,
     ylim = c(0,7e+08), las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit")
dev.off()
```

```
## quartz_off_screen
##                  2
```

```
knitr::include_graphics(paste(working.path,
"plot_MovieMetaData_NotRealContributors.png", sep = "/"))
```



```
##############################################################################
#
#       Regression modeling, to explore the franchise movie financial data
#
##############################################################################
#

# check out the variables in the data set of franchise movie detail
names(fran.movie)

##  [1] "color"                      "director_name"
"num_critic_for_reviews"    "duration"
"director_facebook_likes"   "actor_3_facebook_likes"    "actor_2_name"
"actor_1_facebook_likes"    "genres"                    "actor_1_name"
"num_voted_users"           "cast_total_facebook_likes"
## [13] "actor_3_name"               "facenumber_in_poster"
"plot_keywords"             "movie_imdb_link"
```

```
"num_user_for_reviews"      "language"                  "country"
"content_rating"            "budget"
"actor_2_facebook_likes"    "imdb_score"                "aspect_ratio"
## [25] "movie_facebook_likes"      "movie.gross"               "movie.year"
"adjust.gross"              "movie.names.clean"
```

```
# leave out pre-clean, repeat, unsuitable (to regression) text-content
variables
# and create a subset for regression modeling
franmovie.regrset= fran.movie[, -c(9, 15:16, 26)]
names(franmovie.regrset)
```

```
##  [1] "color"                     "director_name"
"num_critic_for_reviews"    "duration"
"director_facebook_likes"   "actor_3_facebook_likes"    "actor_2_name"
"actor_1_facebook_likes"    "actor_1_name"              "num_voted_users"
"cast_total_facebook_likes" "actor_3_name"
## [13] "facenumber_in_poster"      "num_user_for_reviews"      "language"
"country"                   "content_rating"            "budget"
"actor_2_facebook_likes"    "imdb_score"                "aspect_ratio"
"movie_facebook_likes"      "movie.year"                "adjust.gross"
## [25] "movie.names.clean"
```

```
str(franmovie.regrset)
```

```
## 'data.frame':    1096 obs. of  25 variables:
##  $ color                    : chr  "Color" "Color" "Color" "Color" ...
##  $ director_name            : chr  "Martin Campbell" "Martin Campbell"
"Ben Stiller" "Ben Stiller" ...
##  $ num_critic_for_reviews   : int  137 156 226 135 445 50 58 143 77 191
...
##  $ duration                 : int  129 136 102 90 88 107 95 80 101 132 ...
##  $ director_facebook_likes  : int  258 258 0 0 181 58 548 40 93 357 ...
##  $ actor_3_facebook_likes   : int  163 94 1000 8000 11 316 360 375 218 212
...
##  $ actor_2_name             : chr  "Nick Chinlund" "Tony Amendola" "Will
Ferrell" "Alexander Skarsg\x92\xc7rd" ...
##  $ actor_1_facebook_likes   : int  2000 12000 14000 14000 15000 549 389
3000 287 14000 ...
##  $ actor_1_name             : chr  "Michael Emerson" "Anthony Hopkins"
"Milla Jovovich" "Milla Jovovich" ...
##  $ num_voted_users          : int  71574 135404 34964 201084 386217 42614
23671 16385 51349 142569 ...
##  $ cast_total_facebook_likes: int  2864 12396 24107 34565 28011 1747 1792
4394 993 14790 ...
##  $ actor_3_name             : chr  "Adrian Alonso" "Stuart Wilson" "Justin
Theroux" "Will Ferrell" ...
##  $ facenumber_in_poster     : int  1 0 4 0 4 5 0 0 2 0 ...
##  $ num_user_for_reviews     : int  244 318 150 523 553 120 247 100 213 737
...
##  $ language                 : chr  "Spanish" "English" "English" "English"
```

```
...
##  $ country                    : chr  "USA" "USA" "USA" "Germany" ...
##  $ content_rating             : chr  "PG" "PG-13" "PG-13" "PG-13" ...
##  $ budget                     : num  75000000 65000000 50000000 28000000
23600000 13000000 8000000 80000000 87000000 70000000 ...
##  $ actor_2_facebook_likes    : int  277 174 8000 10000 13000 439 363 642
233 223 ...
##  $ imdb_score                 : num  5.9 6.7 4.8 6.6 7.7 6.8 3.5 4.6 4.3 5.8
...
##  $ aspect_ratio               : num  2.35 2.35 2.35 2.35 2.35 1.85 1.85 1.85
2.35 2.35 ...
##  $ movie_facebook_likes     : int  951 0 28000 0 26000 0 0 0 0 10000 ...
##  $ movie.year                 : chr  "2005" "1998" "2016" "2001" ...
##  $ adjust.gross               : num  5.57e+07 1.38e+08 2.88e+07 6.12e+07
8.46e+07 ...
##  $ movie.names.clean         : chr  "The Legend of Zorroξ" "The Mask of
Zorroξ" "Zoolander 2ξ" "Zoolanderξ" ...

# create a vector indexing categorical variable of fran.regrest
fran.chrcol.index = c(1:2,7,9,12,15:17,23) # movie year should be character
data

# use cleaned movie names as ID, move to the 1st left column
franmovie.regrset= data.frame(franmovie.regrset[,25],
franmovie.regrset[,fran.chrcol.index],
      franmovie.regrset[, -c(fran.chrcol.index,25)])
names(franmovie.regrset)[1] = "movie.names.clean"

# store franchise movie names in a vector
fran.movie.name = names(franmovie.regrset)
# coerce data types back and forward to change character "NA" to Null value
franmovie.regchr =  sapply(1: (1+length(fran.chrcol.index)), simplify = T,
function(j){
                    as.character(as.factor(franmovie.regrset[, 1:10][,j]))})

franmovie.regnum = sapply(1:(dim(franmovie.regrset)[2]-1-
length(fran.chrcol.index)),
                    simplify = T, function(i){
                    as.numeric(as.character(franmovie.regrset[,
11:25][,i]))})

franmovie.regchr = data.frame(franmovie.regchr, stringsAsFactors = T)
franmovie.regnum = data.frame(franmovie.regnum, stringsAsFactors = F)
franmovie.regrset = cbind(franmovie.regchr, franmovie.regnum)
names(franmovie.regrset) = fran.movie.name
str(franmovie.regrset)

## 'data.frame':    1096 obs. of  25 variables:
##  $ movie.names.clean         : Factor w/ 962 levels "10 Cloverfield
Laneξ",..: 803 814 961 962 960 958 957 956 954 955 ...
```

```
##  $ color                   : Factor w/ 2 levels "Black and White",..: 2 2
2 2 2 2 2 2 2 2 ...
##  $ director_name           : Factor w/ 604 levels "Adam Marcus",..: 354
354 42 42 478 82 77 155 329 452 ...
##  $ actor_2_name            : Factor w/ 763 levels "A. Michael
Baldwin",..: 527 714 749 28 77 547 467 709 535 227 ...
##  $ actor_1_name            : Factor w/ 553 levels "Abbie Cornish",..: 371
36 387 387 158 213 241 284 502 540 ...
##  $ actor_3_name            : Factor w/ 835 levels "A.J. Buckley",..: 5
750 416 814 196 90 746 406 824 475 ...
##  $ language                : Factor w/ 16 levels
"Bosnian","Cantonese",..: 14 4 4 4 4 4 4 4 4 4 ...
##  $ country                 : Factor w/ 22 levels
"Australia","Belgium",..: 22 22 22 8 22 22 22 22 22 22 ...
##  $ content_rating          : Factor w/ 7 levels "Approved","G",..: 4 5 5
5 6 6 5 4 5 5 ...
##  $ movie.year              : Factor w/ 52 levels "1920","1939",..: 41 34
52 37 45 24 40 46 41 38 ...
##  $ num_critic_for_reviews   : num  137 156 226 135 445 50 58 143 77 191
...
##  $ duration                : num  129 136 102 90 88 107 95 80 101 132 ...
##  $ director_facebook_likes : num  258 258 0 0 181 58 548 40 93 357 ...
##  $ actor_3_facebook_likes   : num  163 94 1000 8000 11 316 360 375 218 212
...
##  $ actor_1_facebook_likes   : num  2000 12000 14000 14000 15000 549 389
3000 287 14000 ...
##  $ num_voted_users         : num  71574 135404 34964 201084 386217 ...
##  $ cast_total_facebook_likes: num  2864 12396 24107 34565 28011 ...
##  $ facenumber_in_poster     : num  1 0 4 0 4 5 0 0 2 0 ...
##  $ num_user_for_reviews     : num  244 318 150 523 553 120 247 100 213 737
...
##  $ budget                  : num  75000000 65000000 50000000 28000000
23600000 13000000 8000000 80000000 87000000 70000000 ...
##  $ actor_2_facebook_likes   : num  277 174 8000 10000 13000 439 363 642
233 223 ...
##  $ imdb_score              : num  5.9 6.7 4.8 6.6 7.7 6.8 3.5 4.6 4.3 5.8
...
##  $ aspect_ratio            : num  2.35 2.35 2.35 2.35 2.35 1.85 1.85 1.85
2.35 2.35 ...
##  $ movie_facebook_likes     : num  951 0 28000 0 26000 0 0 0 0 10000 ...
##  $ adjust.gross            : num  5.57e+07 1.38e+08 2.88e+07 6.12e+07
8.46e+07 ...

dim(franmovie.regrset)

## [1] 1096   25

# There are several hundreds of levels for directors and actors,
# but the franchise movies data set does not have enough data points for so
many levels
```

```
director_name.count = table(franmovie.regrset$director_name)[1:30]
actor_2_name.count = table(franmovie.regrset$actor_2_name)[1:30]
actor_1_name.count = table(franmovie.regrset$actor_1_name)[1:30]
actor_3_name.count = table(franmovie.regrset$actor_3_name)[1:30]

knitr::kable(director_name.count[-c(9)], col.names = c("Director's Name",
"Frequency"), valign = 't')
```

Director's Name	Frequency
Adam Marcus	1
Adam McKay	2
Adam Shankman	3
Agnieszka Holland	1
Alan Metter	1
Alan Parker	1
Alan Taylor	2
Alejandro Agresti	1
Alessandro Carloni	1
Alex Craig Mann	1
Alex Gibney	1
Alex Proyas	1
Alexander Witt	1
Alexandre Aja	1
Alfonso Cuar<92>_n	1
Alfred Hitchcock	1
Andrew Adamson	7
Andrew Davis	1
Andrew Douglas	1
Andrew Morahan	1
Andrew Stanton	1
Andrzej Bartkowiak	1
Andy Fickman	1
Andy Tennant	2
Ang Lee	2
Angela Robinson	1
Angelina Jolie Pitt	1
Annabel Jankel	1
Anne Fletcher	1

```r
knitr::kable(actor_1_name.count[-c(28)], col.names = c("First Actor Name",
"Frequency"), valign = 't')
```

First Actor Name	Frequency
Abbie Cornish	1
Adam Baldwin	3
Adam Goldberg	2
Adam Scott	1
Aidan Turner	4
Al Pacino	2
Alan Rickman	2
Albert Brooks	1
Alex Gibney	1
Alexa PenaVega	3
Alexander Gould	1
Alfre Woodard	2
Alice Braga	2
Alice Greczyn	1
Alice Krige	1
Alicia Witt	3
Alison Brie	1
Alison Lohman	1
Alyson Hannigan	2
Alyson Stoner	1
Amanda Schull	1
Amber Stevens West	2
America Ferrera	2
Ami Ayalon	1
Amos Oz	1
Amy Poehler	5
Andrew Fiscella	2
Andrew Robinson	1
Angelina Jolie Pitt	7

```r
knitr::kable(actor_2_name.count, col.names = c("Second Actor Name",
"Frequency"), valign = 't')

## Error in nchar(x, "chars"): invalid multibyte string, element 30
```

```
knitr::kable(actor_3_name.count, col.names = c("Third Actor Name",
"Frequency"), valign = 't')
```

Third Actor Name	Frequency
A.J. Buckley	1
Aaron Stanford	1
Adam Copeland	1
Adam Trese	1
Adrian Alonso	1
Agnes Bruckner	1
Al Leong	2
Al Roker	2
Alan D. Purwin	2
Alan David	1
Alanna Ubach	1
Albert Finney	1
Alessandro Nivola	1
Alex Borstein	1
Alex Winter	2
Alexa Davalos	2
Alexa PenaVega	1
Alexander Gould	2
Alexandra Callas	1
Alexandre Rodrigues	1
Ali Hillis	1
Alice Krige	1
Alisha Boe	1
Alison Brie	1
Alison Doody	1
Alissa Dean	1
Allen Covert	1
Alyson Stoner	1
Amanda Wyss	2
Amber Armstrong	1

```
# There are 7 to 52 levels for other categorical variables
# Values of color, language and country are dominated by "color", "English"
and "USA"
regrset.color.count = table(franmovie.regrset$color)
regrset.lang.count = table(franmovie.regrset$language)
```

```r
regrset.country.count =table(franmovie.regrset$country)
regrset.rating.count =table(franmovie.regrset$content_rating)
regrset.year.count =table(franmovie.regrset$movie.year)

knitr::kable(regrset.color.count, col.names = c("Color", "Count"), valign =
't')
```

Color	Count
Black and White	25
Color	1071

```r
knitr::kable(regrset.lang.count, col.names = c("Language", "Count"), valign =
't')
```

Language	Count
Bosnian	1
Cantonese	1
Danish	1
English	1062
French	7
Hebrew	2
Hindi	2
Indonesian	2
Japanese	2
Kazakh	1
Mandarin	7
Portuguese	1
Russian	1
Spanish	3
Telugu	1
Thai	1

```r
knitr::kable(regrset.country.count, col.names = c("Country", "Count"), valign
= 't')
```

Country	Count
Australia	19
Belgium	1
Brazil	1
Canada	20
China	7
Denmark	2

Country	Count
France	22
Germany	19
Hong Kong	1
India	3
Indonesia	1
Israel	2
Japan	4
New Zealand	6
Official site	1
Russia	2
South Korea	1
Spain	2
Taiwan	1
Thailand	1
UK	60
USA	920

```
knitr::kable(regrset.rating.count, col.names = c("Content Rating", "Count"),
valign = 't')
```

Content Rating	Count
Approved	2
G	42
NC-17	9
PG	232
PG-13	390
R	406
Unrated	6

```
knitr::kable(regrset.year.count[1:30,], col.names = c("Movie Years",
"Count"), valign = 't')

## Error in `[.default`(regrset.year.count, 1:30, ): incorrect number of
dimensions

# create two subsets of franchise movies
# the first one only has numeric data,
# the second one has numeric data with variables of color, language, country
and content rating
franmovie.regr01 = franmovie.regrset[,-c(2:10)]
franmovie.regr02 = franmovie.regrset[,-c(3:6)]
```

```r
# create matrix of NA value indices
# use non-repeat row number of the matrix to remove records with NA value
fran.narec.01 = which(is.na(franmovie.regr01) == T, arr.ind = T)
franmovie.regr01 = franmovie.regr01[-unique(fran.narec.01[, "row"]), ]
fran.narec.02 = which(is.na(franmovie.regr02) == T, arr.ind = T)
franmovie.regr02  = franmovie.regr02[-unique(fran.narec.02[, "row"]), ]

# 4 records that NA values are only in the text-content variables
dim(franmovie.regr01)[1] - dim(franmovie.regr02)[1]
```

```
## [1] 4
```

```r
# regression modeling
#create length variables for training data sets
library("stats")
set.seed(3456)
fran.train01.lgth = floor(0.6*dim(franmovie.regr01)[1])
fran.train02.lgth = floor(0.6*dim(franmovie.regr02)[1])

# generate random index with sample function to create training and test data
sets
# exclude the movie names for both training and test data sets
fran.train01.ind = sample(1:dim(franmovie.regr01)[1], fran.train01.lgth)
fran.trn01.reg = franmovie.regr01[fran.train01.ind, -1]
fran.tst01.reg  = franmovie.regr01[-fran.train01.ind, -1]

fran.train02.ind = sample(1:dim(franmovie.regr02)[1], fran.train02.lgth)
fran.trn02.reg = franmovie.regr02[fran.train02.ind, -1]
fran.tst02.reg  = franmovie.regr02[-fran.train02.ind, -1]

#create a vector for storing the original value of adjust.gross variable
#adjust.gross is a dependent variable
fran.adjgross.tstreg01 = fran.tst01.reg$adjust.gross
fran.adjgross.tstreg02 = fran.tst02.reg$adjust.gross

#Then removed the existing variables for prediction
fran.tst01.reg$adjust.gross = NULL
fran.tst02.reg$adjust.gross = NULL

# run logistic model to the subset with only numeric variables only
# adjust.gross is the dependent variable
# "." means include everything except the dependent variable
fran.model01 = glm(adjust.gross ~ ., family = gaussian, data =
fran.trn01.reg)

# create an index of sorted Estimate/P-value, to list important variables on
top
fran.model01.PvalueSort =
order(summary.glm(fran.model01)[12]$coefficients[,4], decreasing = FALSE)
```

```
#list logistic linear regression modeling statistics
summary.glm(fran.model01)[c(1,3:4)]
```

```
## $call
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn01.reg)
##
## $family
##
## Family: gaussian
## Link function: identity
##
##
## $deviance
## [1] 8.524655e+18
```

```
fran.model01.coefficient =
summary.glm(fran.model01)[12]$coefficients[fran.model01.PvalueSort,]
knitr::kable(fran.model01.coefficient, digits = 4, format.args =
list(scientific = TRUE), "simple")
```

	Estimate	Std. Error	t value	Pr(>|t|)
num_voted_users	3.848227e+02	5.019300e+01	7.6669e+00	0.000e+00
budget	6.605000e-01	1.115000e-01	5.9260e+00	0.000e+00
movie_facebook_likes	-7.828239e+02	2.851721e+02	-2.7451e+00	6.200e-03
imdb_score	1.461757e+07	5.375184e+06	2.7195e+00	6.700e-03
aspect_ratio	-3.639075e+07	1.968925e+07	-1.8483e+00	6.500e-02
actor_1_facebook_likes	-5.783765e+03	3.600543e+03	-1.6064e+00	1.087e-01
cast_total_facebook_likes	5.562579e+03	3.612477e+03	1.5398e+00	1.241e-01
num_user_for_reviews	-2.317176e+04	1.543091e+04	-1.5016e+00	1.337e-01
actor_2_facebook_likes	-5.504218e+03	3.759100e+03	-1.4642e+00	1.436e-01
director_facebook_likes	1.983345e+03	1.754351e+03	1.1305e+00	2.587e-01
actor_3_facebook_likes	-6.588551e+03	6.214261e+03	-1.0602e+00	2.895e-01
duration	9.293342e+04	2.421431e+05	3.8380e-01	7.013e-01
num_critic_for_reviews	1.911349e+04	6.711947e+04	2.8480e-01	7.759e-01
(Intercept)	1.249990e+07	5.120908e+07	2.4410e-01	8.072e-01
facenumber_in_poster	-6.108535e+04	2.466203e+06	-2.4800e-02	9.802e-01

```
names(fran.trn01.reg)
```

```
##  [1] "num_critic_for_reviews"    "duration"
"director_facebook_likes"   "actor_3_facebook_likes"
"actor_1_facebook_likes"    "num_voted_users"
"cast_total_facebook_likes" "facenumber_in_poster"
"num_user_for_reviews"      "budget"
"actor_2_facebook_likes"    "imdb_score"
```

```
## [13] "aspect_ratio"              "movie_facebook_likes"
"adjust.gross"
```

```
#Observation to fran.model01
# movie_facebook_likes, aspect_ratio appears to be insignificant
# director_facebook_likes, facenumber_in_poster appears to be insignificant
# duration, num_user_for_reviews, imdb_score appears to be less significant
fran.model01mo = glm(adjust.gross ~ ., family = gaussian, data =
fran.trn01.reg[,-c(2,3,8,9,12,13,14)])

# create an index of sorted Estimate/P-value, to list important variables on
top
fran.model01mo.PvalueSort =
order(summary.glm(fran.model01mo)[12]$coefficients[,4], decreasing = FALSE)

#list logistic linear regression modeling statistics
summary.glm(fran.model01mo)[c(1,3:4)]
```

```
## $call
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn01.reg[,
##     -c(2, 3, 8, 9, 12, 13, 14)])
##
## $family
##
## Family: gaussian
## Link function: identity
##
##
## $deviance
## [1] 8.856684e+18
```

```
fran.model01mo.coefficient =
summary.glm(fran.model01mo)[12]$coefficients[fran.model01mo.PvalueSort,]
knitr::kable(fran.model01mo.coefficient, digits = 4, format.args =
list(scientific = TRUE), "simple")
```

	Estimate	Std. Error	t value	Pr(>|t|)
num_voted_users	3.921673e+02	3.120260e+01	1.25684e+01	0.000e+00
budget	6.434000e-01	1.073000e-01	5.99910e+00	0.000e+00
(Intercept)	4.446744e+07	8.623156e+06	5.15670e+00	0.000e+00
num_critic_for_reviews	-9.848207e+04	4.931770e+04	-1.99690e+00	4.630e-02
actor_1_facebook_likes	-4.754276e+03	3.634101e+03	-1.30820e+00	1.913e-01
cast_total_facebook_likes	4.562908e+03	3.646881e+03	1.25120e+00	2.113e-01
actor_2_facebook_likes	-4.644623e+03	3.794927e+03	-1.22390e+00	2.215e-01
actor_3_facebook_likes	-5.837317e+03	6.261997e+03	-9.32200e-01	3.516e-01

```
#predict response variable, the predicted values are probabilities
pred.fran.reg01 <- predict(fran.model01mo, newdata = fran.tst01.reg, type =
"response")

# run logistic model to the subset of combined numeric & several categorical
variables
fran.model02 = glm(adjust.gross ~ ., family = gaussian, data =
fran.trn02.reg)
# create an index of sorted Estimate/P-value, to list important variables on
top
fran.model02.PvalueSort =
order(summary.glm(fran.model02)[12]$coefficients[,4], decreasing = FALSE)

#list logistic linear regression modeling statistics
summary.glm(fran.model02)[c(1,3:4)]

## $call
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn02.reg)
##
## $family
##
## Family: gaussian
## Link function: identity
##
##
## $deviance
## [1] 2.930291e+18

fran.model02.coefficient =
summary.glm(fran.model02)[12]$coefficients[fran.model02.PvalueSort[1:35],]
knitr::kable(fran.model02.coefficient, digits = 4, format.args =
list(scientific = TRUE), "simple")
```

	Estimate	Std. Error	t value	Pr(>|t|)
movie.year1975	9.376143e+08	7.920679e+07	1.18375e+01	0.00e+00
movie.year1973	6.865898e+08	7.780576e+07	8.82440e+00	0.00e+00
budget	7.232000e-01	8.280000e-02	8.73520e+00	0.00e+00
num_voted_users	2.672344e+02	3.361730e+01	7.94930e+00	0.00e+00
movie.year1978	2.586649e+08	3.990798e+07	6.48150e+00	0.00e+00
movie.year1972	4.234522e+08	8.208341e+07	5.15880e+00	0.00e+00
movie_facebook_likes	-9.615697e+02	2.224710e+02	-4.32220e+00	0.00e+00
movie.year1980	2.059477e+08	4.921504e+07	4.18470e+00	0.00e+00
movie.year2008	-1.010376e+08	2.663360e+07	-3.79360e+00	2.00e-04
movie.year2011	-9.541526e+07	2.589193e+07	-3.68510e+00	3.00e-04
content_ratingNC-17	-3.082235e+08	8.794022e+07	-3.50490e+00	5.00e-04
movie.year2004	-9.221588e+07	2.666191e+07	-3.45870e+00	6.00e-04
movie.year1983	1.432599e+08	4.409531e+07	3.24890e+00	1.20e-03

	Estimate	Std. Error	t value	Pr(>|t|)
movie.year1976	2.569953e+08	7.932672e+07	3.23970e+00	1.30e-03
content_ratingR	-2.595487e+08	8.027559e+07	-3.23320e+00	1.30e-03
movie.year2009	-8.006683e+07	2.644468e+07	-3.02770e+00	2.60e-03
movie.year2005	-7.613832e+07	2.651438e+07	-2.87160e+00	4.20e-03
movie.year2010	-7.550207e+07	2.688155e+07	-2.80870e+00	5.20e-03
content_ratingUnrated	-2.532773e+08	9.147634e+07	-2.76880e+00	5.80e-03
movie.year2013	-6.559921e+07	2.462536e+07	-2.66390e+00	8.00e-03
content_ratingPG-13	-2.143847e+08	8.065543e+07	-2.65800e+00	8.10e-03
movie.year2007	-7.573666e+07	2.874486e+07	-2.63480e+00	8.70e-03
actor_1_facebook_likes	-6.644260e+03	2.537243e+03	-2.61870e+00	9.10e-03
cast_total_facebook_likes	6.580709e+03	2.549970e+03	2.58070e+00	1.01e-02
movie.year2014	-6.393032e+07	2.556520e+07	-2.50070e+00	1.27e-02
movie.year2006	-6.514237e+07	2.643844e+07	-2.46390e+00	1.40e-02
content_ratingG	-2.024424e+08	8.239956e+07	-2.45680e+00	1.43e-02
content_ratingPG	-1.976061e+08	8.074510e+07	-2.44730e+00	1.47e-02
(Intercept)	2.918864e+08	1.194352e+08	2.44390e+00	1.48e-02
actor_2_facebook_likes	-6.025795e+03	2.646216e+03	-2.27710e+00	2.32e-02
movie.year1994	-6.883672e+07	3.085954e+07	-2.23060e+00	2.61e-02
movie.year2012	-5.853090e+07	2.633530e+07	-2.22250e+00	2.67e-02
movie.year1968	1.795686e+08	8.128698e+07	2.20910e+00	2.76e-02
num_critic_for_reviews	1.249659e+05	5.702738e+04	2.19130e+00	2.88e-02
movie.year1984	7.692255e+07	3.666441e+07	2.09800e+00	3.64e-02

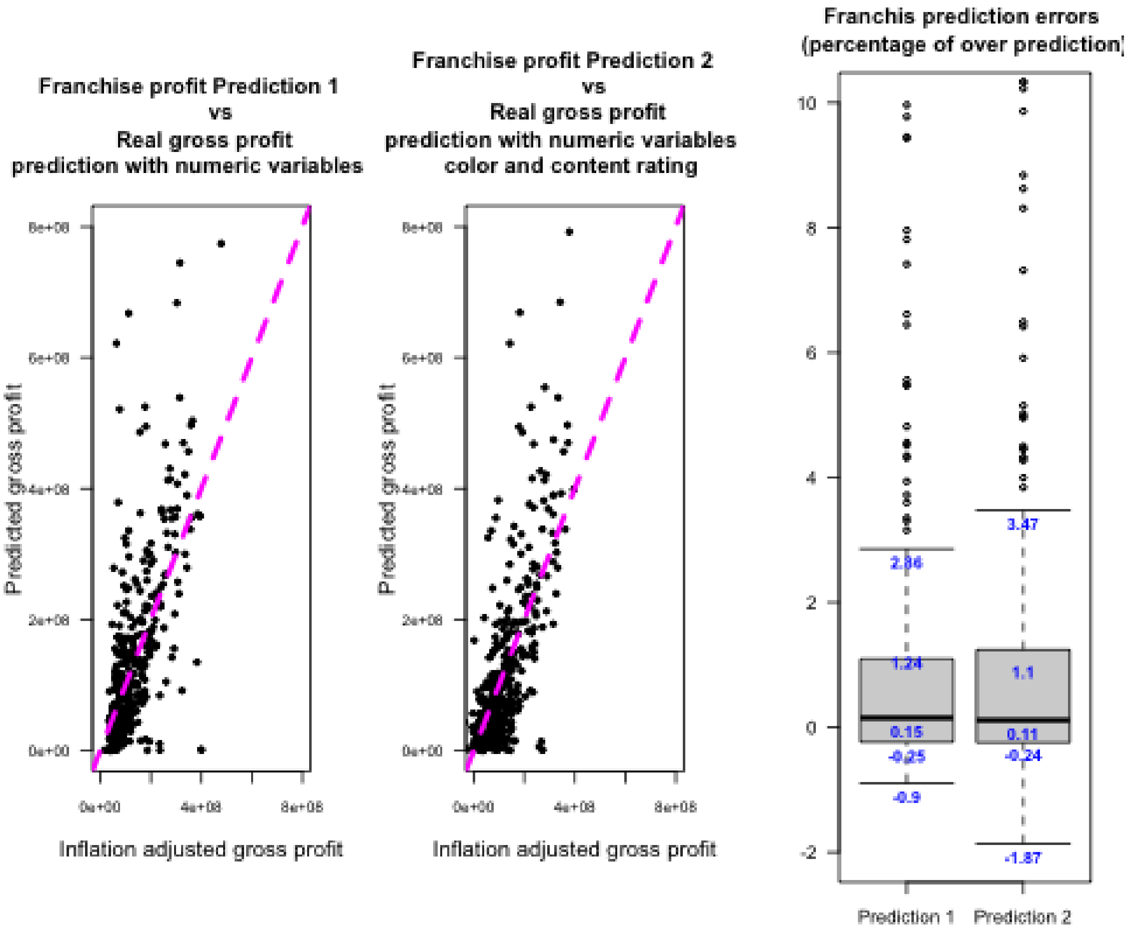
```
# Observation:
# 1.Country Brazil,Denmark, Hongkong, Taiwan and Thailand has NA coefficient
# 2. Except Canada and USA has P-value less than 0.2, other countries have P-
value no less than 0.4
# 3. Not enough data for language, country, movie years

# specify references for color and content_rating variables
fran.trn02.reg$color = relevel(fran.trn02.reg$color, ref = "Color")
table(fran.trn02.reg$content_rating)

##
## Approved        G    NC-17       PG    PG-13        R  Unrated
##        1       19        7      146      209      242        5

fran.trn02.reg$content_rating = relevel(fran.trn02.reg$content_rating, ref =
"R")

# remove variables: language, country, movie years
fran.model02 = glm(adjust.gross ~ ., family = gaussian, data =
fran.trn02.reg[,-c(2:3,5)])
summary.glm(fran.model02)

## 
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn02.reg[, 
##     -c(2:3, 5)])
## 
## Deviance Residuals: 
##        Min          1Q      Median          3Q         Max  
## -479689049   -50734985    -6058274    30993533   898392754  
## 
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)               -7.472e+07  4.417e+07  -1.692 0.091175 .  
## colorBlack and White      -5.175e+07  2.610e+07  -1.983 0.047829 *  
## content_ratingApproved     2.907e+08  1.019e+08   2.853 0.004472 ** 
## content_ratingG            6.940e+07  2.451e+07   2.831 0.004789 ** 
## content_ratingNC-17        4.956e+06  3.791e+07   0.131 0.896042    
## content_ratingPG           8.834e+07  1.143e+07   7.727 4.59e-14 ***
## content_ratingPG-13        5.166e+07  1.054e+07   4.904 1.21e-06 ***
## content_ratingUnrated     -2.281e+07  4.495e+07  -0.507 0.612034    
## num_critic_for_reviews     2.516e+03  5.713e+04   0.044 0.964892    
## duration                   3.023e+05  2.373e+05   1.274 0.203061    
## director_facebook_likes    9.032e+02  1.422e+03   0.635 0.525667    
## actor_3_facebook_likes    -8.730e+03  5.654e+03  -1.544 0.123092    
## actor_1_facebook_likes    -8.626e+03  3.254e+03  -2.651 0.008233 ** 
## num_voted_users            2.611e+02  3.994e+01   6.536 1.34e-10 ***
## cast_total_facebook_likes  8.441e+03  3.268e+03   2.583 0.010022 *  
## facenumber_in_poster      -1.758e+06  2.206e+06  -0.797 0.425688    
## num_user_for_reviews       1.067e+04  1.154e+04   0.925 0.355408    
## budget                     3.448e-01  9.588e-02   3.596 0.000350 ***
## actor_2_facebook_likes    -7.607e+03  3.393e+03  -2.242 0.025344 *  
## imdb_score                 1.765e+07  4.781e+06   3.692 0.000243 ***
## aspect_ratio              -2.424e+07  1.626e+07  -1.491 0.136568    
## movie_facebook_likes      -6.086e+02  2.305e+02  -2.640 0.008494 ** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## (Dispersion parameter for gaussian family taken to be 9.542446e+15)
## 
##     Null deviance: 1.1236e+19  on 628  degrees of freedom
## Residual deviance: 5.7923e+18  on 607  degrees of freedom
## AIC: 24952
## 
## Number of Fisher Scoring iterations: 2

pred.fran.reg02 <- predict(fran.model02, type = "response",
                           newdata = fran.tst02.reg[,-c(2:3,5)])
```

```
# Observation to 2nd round model02
# 1. num_user_for_reviews, num_critic_for_reviews, director_facebook_likes
appear to be insignificant
# 2. duration director_facebook_likes, facenumber_in_poster appear to be
insignificant
# 3. actor_3_facebook_likes, actor_2_facebook_likes appear to be less
significant

fran.model02mo = glm(adjust.gross ~ ., family = gaussian, data =
fran.trn02.reg[,-c(2:3,5:9,13:14,16)])
summary.glm(fran.model02mo)

##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn02.reg[,
##     -c(2:3, 5:9, 13:14, 16)])
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -489433361   -52440735    -7552657    31995972   913770333
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -5.040e+07  4.161e+07  -1.211 0.226247
## colorBlack and White    -5.267e+07  2.607e+07  -2.021 0.043746 *
## content_ratingApproved   2.850e+08  1.020e+08   2.795 0.005358 **
## content_ratingG          6.461e+07  2.394e+07   2.699 0.007153 **
## content_ratingNC-17      1.464e+06  3.781e+07   0.039 0.969122
## content_ratingPG         8.517e+07  1.101e+07   7.739 4.15e-14 ***
## content_ratingPG-13      5.324e+07  1.035e+07   5.146 3.59e-07 ***
## content_ratingUnrated   -3.446e+07  4.456e+07  -0.773 0.439591
## actor_1_facebook_likes  -1.946e+03  5.296e+02  -3.675 0.000259 ***
## num_voted_users          3.006e+02  2.859e+01  10.515  < 2e-16 ***
## cast_total_facebook_likes  1.680e+03  4.661e+02   3.604 0.000339 ***
## budget                   4.059e-01  8.529e-02   4.759 2.43e-06 ***
## imdb_score               1.827e+07  4.583e+06   3.986 7.52e-05 ***
## aspect_ratio            -2.199e+07  1.585e+07  -1.388 0.165790
## movie_facebook_likes    -5.824e+02  1.740e+02  -3.348 0.000864 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 9.598199e+15)
##
##     Null deviance: 1.1236e+19  on 628  degrees of freedom
## Residual deviance: 5.8933e+18  on 614  degrees of freedom
## AIC: 24949
##
## Number of Fisher Scoring iterations: 2
```

```
pred.fran.reg02 <- predict(fran.model02mo, newdata = fran.tst02.reg[,-
c(2:3,5:9,13:14,16)],
                           type = "response")

# visualize prediction vs actual data
png("plot_FranchiseMovieProfitPrediction.png")
par(mfrow = c(1,3))
par(mar = c(15, 4, 10, 3))
plot(pred.fran.reg01, fran.adjgross.tstreg01,  type = "p", pch = 20,
     xlim = c(0, 8e+8), ylim = c(0, 8e+8), las = 1,
     xlab = "Inflation adjusted gross profit",
     ylab = "Predicted gross profit", cex.axis = 0.8, cex.lab = 1.2)
title(main = "Franchise profit Prediction 1 \nvs\n Real gross profit
\nprediction with numeric variables",
      line = 1.5, adj = 0.6, cex.main = 1.2)
abline(a = 0, b= 1, col = "magenta", lty = 2, lwd = 3)

plot(pred.fran.reg02, fran.adjgross.tstreg02,  type = "p", pch = 20,
     xlim = c(0, 8e+8), ylim = c(0, 8e+8), las = 1,
     xlab = "Inflation adjusted gross profit",
     ylab = "Predicted gross profit", cex.axis = 0.8, cex.lab = 1.2)
title(main = "Franchise profit Prediction 2 \nvs\n Real gross profit
\nprediction with numeric variables\ncolor and content rating",
      line = 1.5, adj = 0.6, cex.main = 1.2)
abline(a = 0, b= 1, col = "magenta", lty = 2, lwd = 3)

# standard errors from two prediction of franchise movies
fran.pred.error01 = pred.fran.reg01/fran.adjgross.tstreg01 - 1
fran.pred.error02 = pred.fran.reg02/fran.adjgross.tstreg02 - 1

fran.pred.erorlist = list(fran.pred.error01, fran.pred.error02)
bop02 = boxplot(fran.pred.erorlist, range = 1.5, ylim = c(-2, 10),
                axes = T, staplewex = 1, las = 1, par(mar = c(10, 4, 4,
1.5)),
                names = c("Prediction 1", "Prediction 2"),
                cex.lab = 1.2, cex.main = 1.2,
                main = "Franchis prediction errors \n(percentage of over
prediction)")
text(unique(bop02$group),  bop02$stats, pos = 1, offset = 0.4,
     labels = round(bop02$stats, 2), col = "blue", cex = 0.9, font = 2)
dev.off()
```

```
## quartz_off_screen
##                 2
```

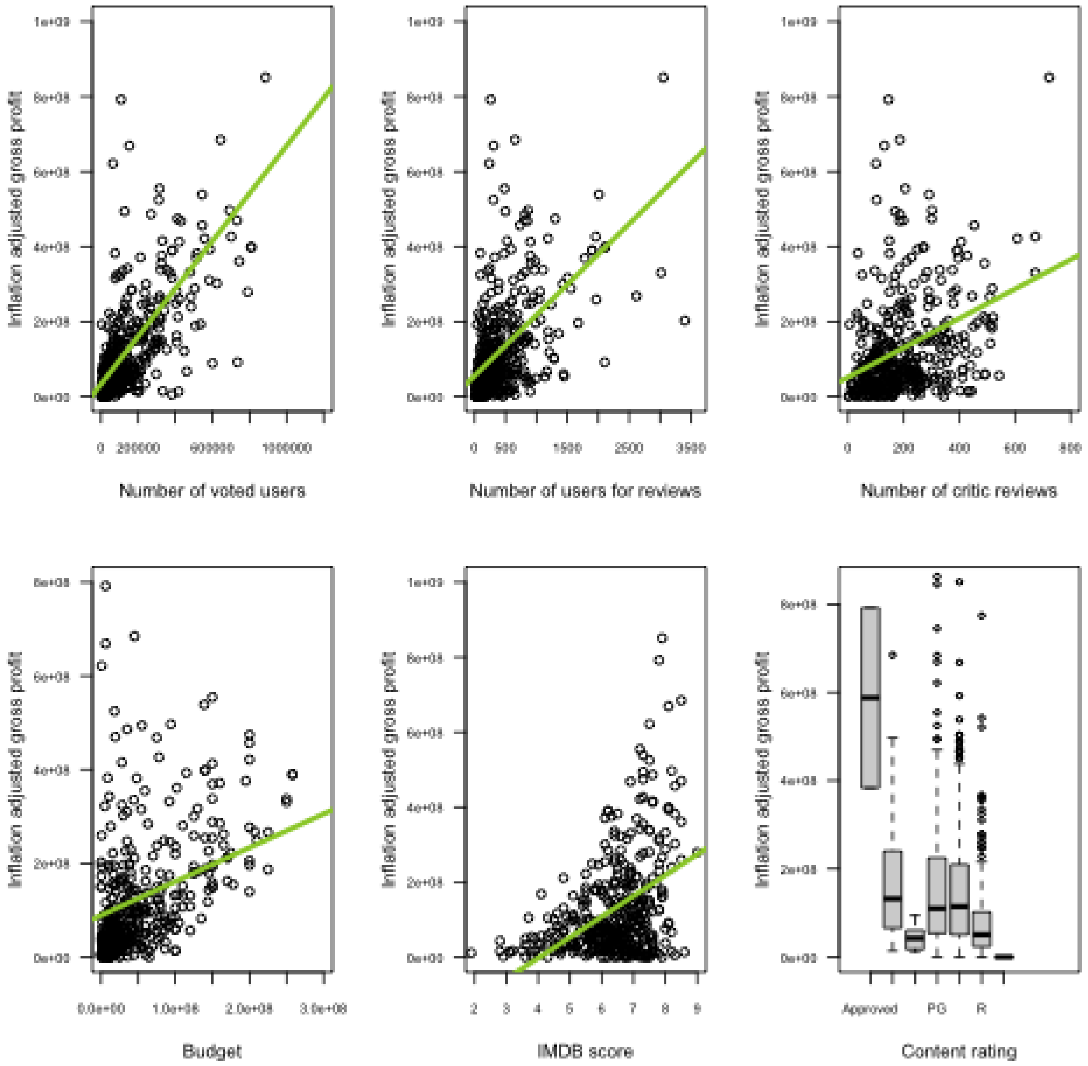
```
knitr::include_graphics(paste(working.path,
"plot_FranchiseMovieProfitPrediction.png", sep = "/"),
                         auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



```
colnames(bop02$stats) = c("Franchise prediction 1", "Franchise prediction 2")
bop02$stats  # show the errors box plot statistics for both predictions

##      Franchise prediction 1 Franchise prediction 2
## [1,]             -0.8965300             -1.8665564
## [2,]             -0.2363189             -0.2473451
## [3,]              0.1518477              0.1125718
## [4,]              1.0951961              1.2418691
## [5,]              2.8556699              3.4725812

# visualize the relationship between individual predictors and movie gross
profit
png("plot_FranchiseMovie_ProfitContributors.png")
par(mfrow=c(2,3))
par(mar = c(4,4,3,2))

plot(fran.tst02.reg$num_voted_users, fran.adjgross.tstreg02,
     xlim = c(0,1200000), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Number of voted users")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$num_voted_users),
       col = "yellow green", lwd = 3)

plot(fran.tst02.reg$num_user_for_reviews, fran.adjgross.tstreg02,
     xlim = c(0,3600), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Number of users for reviews")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$num_user_for_reviews),
       col = "yellow green", lwd = 3)

plot(fran.tst02.reg$num_critic_for_reviews, fran.adjgross.tstreg02,
     xlim = c(0,800), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Number of critic reviews")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$num_critic_for_reviews),
       col = "yellow green", lwd = 3)

plot(fran.tst02.reg$budget, fran.adjgross.tstreg02,
     xlim = c(0,3e+08), ylim = c(0,8e+08),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Budget")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$budget),
       col = "yellow green", lwd = 3)

plot(fran.tst02.reg$imdb_score, fran.adjgross.tstreg02,
     xlim = c(2,9), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "IMDB score")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$imdb_score),
       col = "yellow green", lwd = 3)

plot(franmovie.regr02$content_rating, franmovie.regr02$adjust.gross,
     xlim = c(0,10), ylim = c(0,8.5e+08),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Content rating")
dev.off()
```

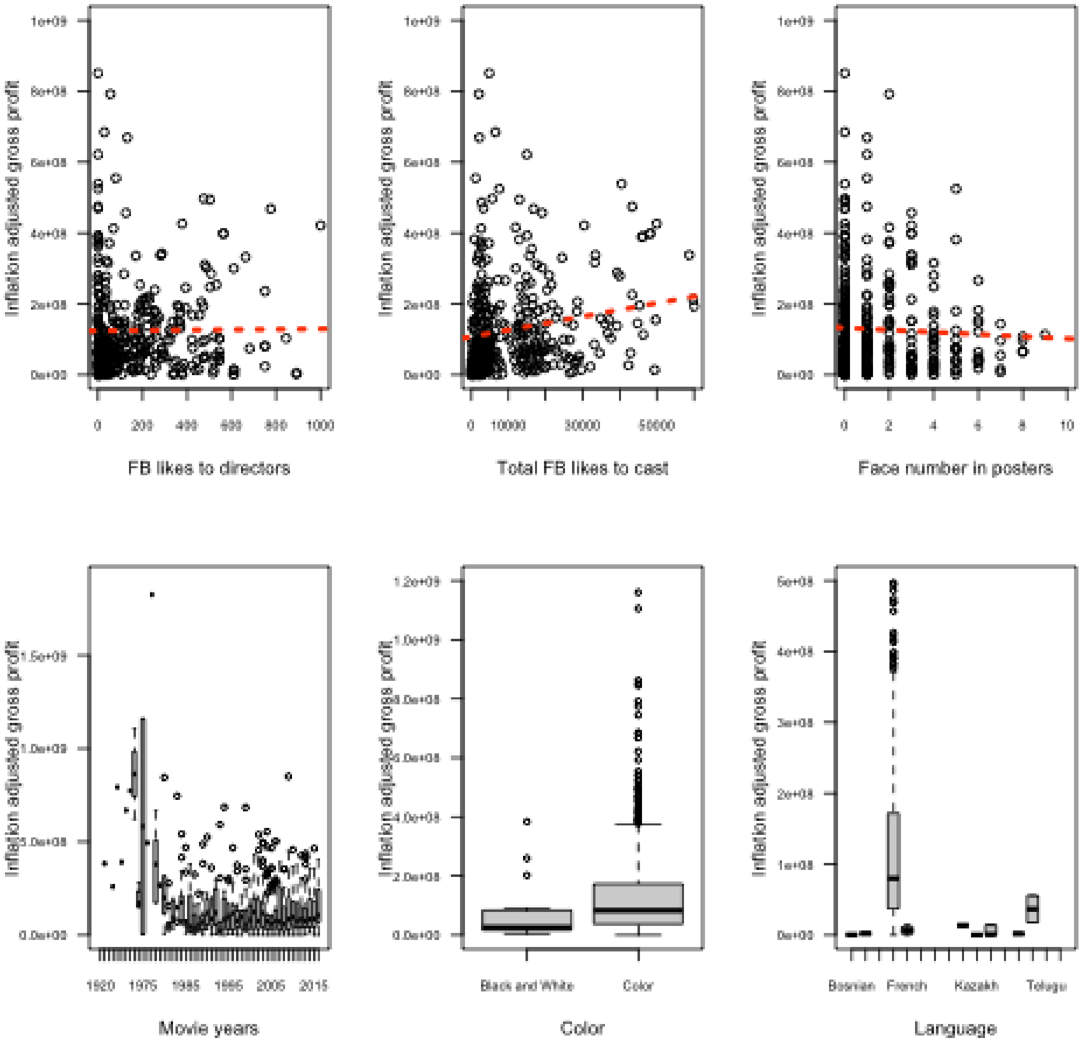
```
## quartz_off_screen
##                 2
```

```
knitr::include_graphics(paste(working.path,
"plot_FranchiseMovie_ProfitContributors.png", sep = "/"),
                         auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



```
png("plot_FranchiseMovie_NotRealContributors.png")
par(mfrow=c(2,3))
par(mar = c(4,4,4,2))

plot(fran.tst02.reg$director_facebook_likes, fran.adjgross.tstreg02,
     xlim = c(0,1000), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "FB likes to directors")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$director_facebook_likes),
       col = "orange red", lwd = 3, lty = 3)
```

```
plot(fran.tst02.reg$cast_total_facebook_likes, fran.adjgross.tstreg02,
     xlim = c(0,60000), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Total FB likes to cast")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$cast_total_facebook_likes),
       col = "orange red", lwd = 3, lty = 3)

plot(fran.tst02.reg$facenumber_in_poster, fran.adjgross.tstreg02,
     xlim = c(0,10), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Face number in posters")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$facenumber_in_poster),
       col = "orange red", lwd = 3, lty = 3)

plot(franmovie.regr02$movie.year, franmovie.regr02$adjust.gross,
     ylim = c(0,1.9e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Movie years")

plot(franmovie.regr02$color, franmovie.regr02$adjust.gross,
     ylim = c(0,1.2e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Color")

plot(fran.tst02.reg$language, fran.adjgross.tstreg02,
     ylim = c(0,5e+08),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Language")

dev.off()

## quartz_off_screen
##                 2

knitr::include_graphics(paste(working.path,
"plot_FranchiseMovie_NotRealContributors.png", sep = "/"),
                         auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



```
knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
```