# FranchisesProject

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
#############################################################################
```

```r
# 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
```

```r
# 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."
```

```r
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
```

```r
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)
```

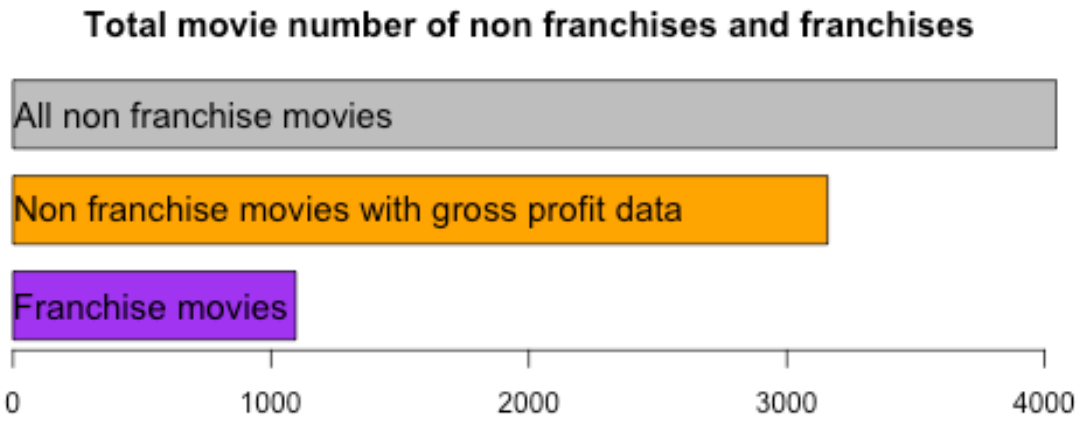
```r
#change names to lower cases for further analysis
movie.names = tolower(movie.names)
Fran.title = tolower(Fran.title)

# 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)
```

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

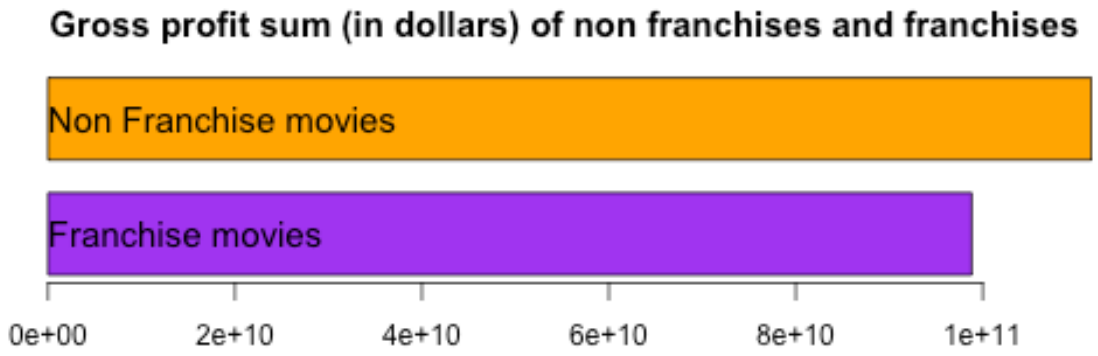
```
## [1]  239 1771 2235 2746
```

```r
# 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(22, 1, 3, 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 = "/"),
                        auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```




```
#cat('\n!["Movie number
compare"](paste(working.path,"barplot_MovieNumberComparison.png", sep =
"/"))\n')

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
```

```r
#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(24, 1, 3, 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 = "/"))
```



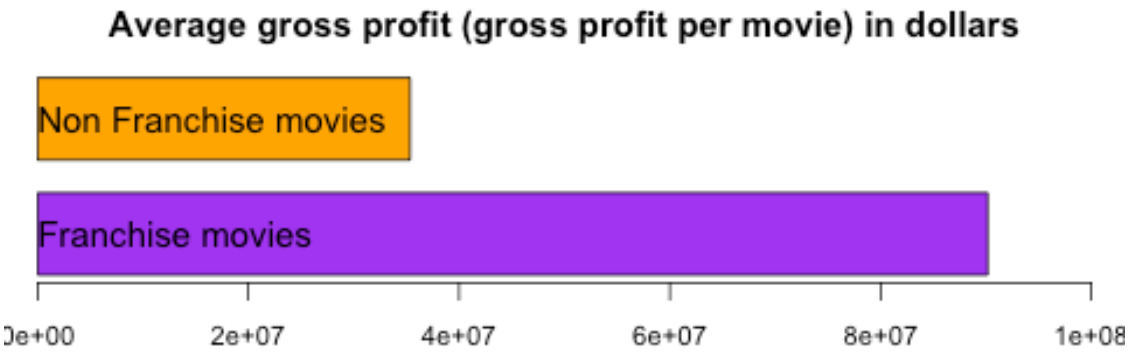

```
names(gross.compare) = c("Franchise movies", "Non Franchise movies")
gross.compare
```

```
##     Franchise movies Non Franchise movies
##          98819051354         111622020100
```

```
png("barplot_AvgfitComparison.png")
par(mar=c(24, 1, 3, 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 = "/"),
                        auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```




```
knitr::opts_chunk$set(echo = TRUE)

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

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

```r
# Check the datatype of the franchise finance data
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" ...
```

```r
# 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
```

```r
# 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) # 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
## 628 280   5
## 610 281   5
## 358 285   5
```

```
## 654 290   5
## 629 312   5
## 693 315   5
## 743 333   5
## 42  340   5
## 752 342   5
## 165 345   5
## 717 357   5
## 757 358   5
## 549 363   5
## 722 364   5
## 749 369   5
## 731 374   5
## 742 376   5
## 707 377   5
## 725 379   5
## 264 381   5
## 734 401   5
## 537 403   5
## 726 404   5
## 727 413   5
## 740 414   5
## 340 419   5
## 719 429   5
## 747 445   5
## 387 447   5
## 656 456   5
## 427 461   5
## 638 467   5
## 739 474   5
## 614 477   5
## 723 488   5
## 690 497   5
## 635 530   5
## 595 535   5
## 558 536   5
## 554 537   5
## 758 547   5
## 748 549   5
## 458 550   5
## 653 553   5
## 370 555   5
## 578 579   5
## 735 582   5
## 600 585   5
## 560 586   5
## 724 623   5
## 330 638   5
## 750 642   5
## 718 656   5
```

```
## 736 691   5
## 649 697   5
## 561 702   5
## 632 707   5
## 720 717   5
## 737 736   5
## 730 738   5
## 753 745   5
## 729 748   5
## 746 759   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)
movfran.fina[franyear.nv[franyear.left],] # To make sure index vectors are
right
```

```
##       X                           Franchise tot.movies First.Year
Last.Year
## 651 651                       St. Trinian's          2       2009
2009
## 521 521                               Smoke          2       1995
1995
## 641 641 San Francisco Opera Cinemacasts 2007          4       2008
2008
```

```
## 604 604                           Red Cliff          2       2009
2009
## 628 628                          On the Run          2       2004
2004
## 693 693             MSG The Messenger of God          2       2015
2015
## 707 707                            Kiseijuu          2       2015
2015
## 537 537                      Jean de Florette          2       1987
1987
## 690 690                          Gangster Ka          2       2015
2015
## 653 653                       Donald Strachey          2       2008
2008
## 370 370                        Dollar Trilogy          3       1967
1967
## 330 330                            Breakin'          2       1984
1984
## 649 649                   As Mil e Uma Noites          3       2015
2015
##     tot.years                      fran.title.clean infl.adj.dom.boxoffice
## 651        NA                          St Trinian s                  17800
## 521        NA                                 Smoke                19699243
## 641        NA San Francisco Opera Cinemacasts 2007                    60840
## 604        NA                               Red Cliff                738211
## 628        NA                              On the Run                148426
## 693        NA                MSG The Messenger of God                     0
## 707        NA                                  Kiseijuu                   0
## 537        NA                        Jean de Florette                12065811
## 690        NA                             Gangster Ka                   0
## 653        NA                         Donald Strachey                5278
## 370        NA                          Dollar Trilogy               103091655
## 330        NA                                 Breakin               134694700
## 649        NA                     As Mil e Uma Noites                   24191
##     glob.boxoffice dome.boxoffice other.boxoffice
## 651       29830239          15000        29815239
## 521       15128284        9628284         5500000
## 641          49088          49088               0
## 604      150127047         627047       149500000
## 628         303715         103569          200146
## 693          27053              0           27053
## 707        7078834              0         7078834
## 537        5504613        5504613               0
## 690        1588476              0         1588476
## 653           4269           4269               0
## 370       13900000       13900000               0
## 330       51100000       51100000               0
## 649          50537          23160           27377
```

```r
# 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
```

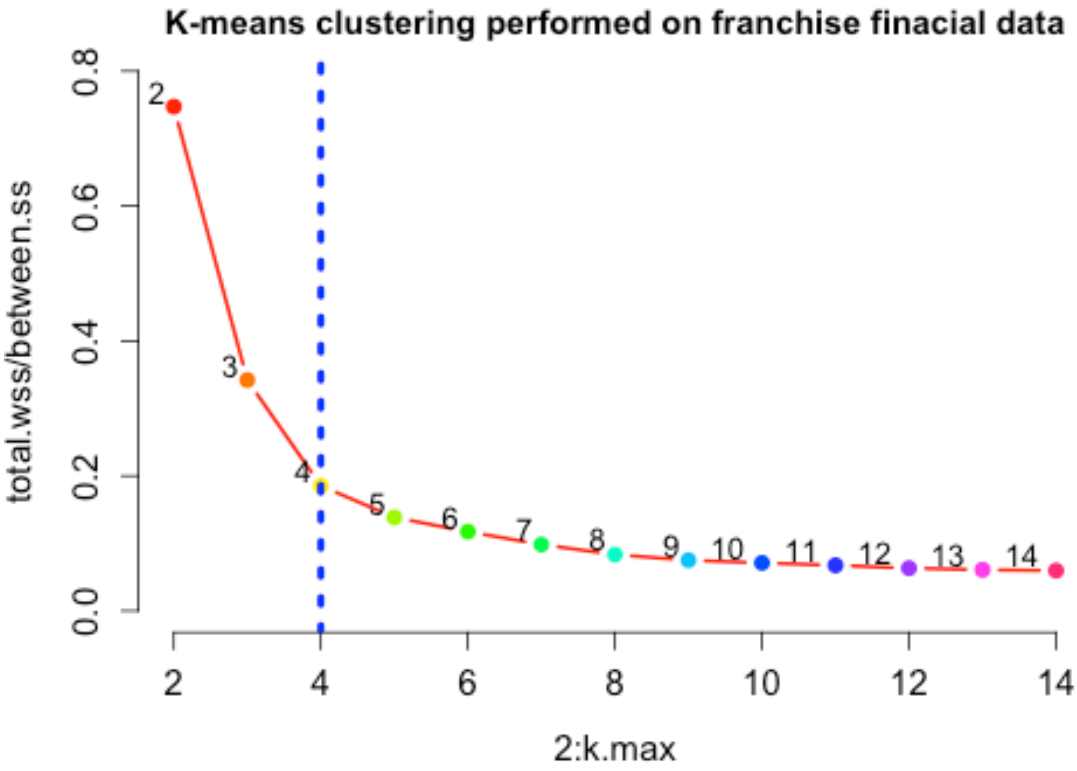
```
## 'data.frame':    760 obs. of  6 variables:
##  $ tot.movies            : int  2 2 2 2 2 1 2 3 14 2 ...
##  $ First.Year            : int  1998 2001 2009 1988 2004 2016 2010 2002
2000 1998 ...
##  $ Last.Year             : int  2005 2016 2011 1990 2011 2016 2014 2017
2019 2008 ...
##  $ tot.years             : 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
...
```

```r
# 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
```

```r
png("plot_KmeansElbow.png")
par(mar=c(12, 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 =
"/"),
                        auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



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

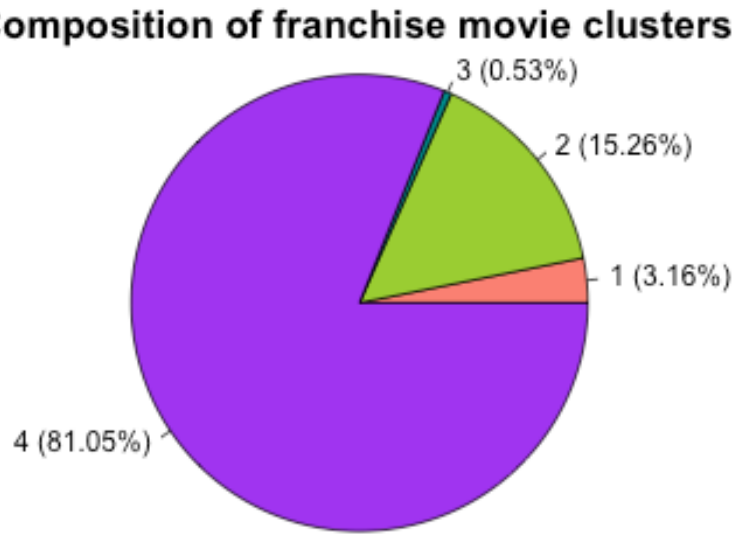
```r
movfran.result = cbind(movfran.fina[, c(1,7,12)], movfran.fina[, c(8,11)],
                       movfran.fina[,c(3,6)], movfran.fina[,c(4,5)])

#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(14,1,1,1))
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"))
```

```
## Warning in text.default(1.1 * P$x, 1.1 * P$y, labels[i], xpd = TRUE, adj =
## ifelse(P$x < : "line" is not a graphical parameter

## Warning in text.default(1.1 * P$x, 1.1 * P$y, labels[i], xpd = TRUE, adj =
## ifelse(P$x < : "line" is not a graphical parameter

## Warning in text.default(1.1 * P$x, 1.1 * P$y, labels[i], xpd = TRUE, adj =
## ifelse(P$x < : "line" is not a graphical parameter

## Warning in text.default(1.1 * P$x, 1.1 * P$y, labels[i], xpd = TRUE, adj =
## ifelse(P$x < : "line" is not a graphical parameter
```

```r
dev.off()
```

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

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




```
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"

movfran.result[movfran.result$K.cluster == 3, -1]

##            fran.title.clean K.cluster infl.adj.dom.boxoffice
other.boxoffice
## 1                 Star Wars         3             6529365840
3874592228
## 3 Marvel Cinematic Universe         3             5390016938
7803549152
## 2                James Bond         3             5625743524
4964007386
## 4              Harry Potter         3             3399078859
5906843667
##   tot.movies tot.years First.Year Last.Year
## 1         12        42       1977      2019
## 3         23        11       2008      2019
```

```
## 2         26        56       1963      2019
## 4         12        19       2001      2020

movfran.result[movfran.result$K.cluster == 4, -1][1:50,]

##              fran.title.clean K.cluster infl.adj.dom.boxoffice
other.boxoffice
## 215                     Zorro         4              241333859
235771255
## 372                  Zoolander         4              100712640
42108731
## 388                 Zombieland         4               89700466
26646310
## 270                 Young Guns         4              189951971
0
## 447            You Got Served         4               57422186
11975903
## 755                Yokai Watch         4                     0
5786581
## 350                  Yogi Bear         4              112882278
104528679
## 271                    X Files         4              185218941
152466424
## 510                 Wrong Turn         4               22755912
13231785
## 236                World War Z         4              221525388
329154939
## 543        Work and the Glory         4                9266493
0
## 512                 Wolf Creek         4               22354424
12984045
## 403          Without a Paddle         4               83346008
6964845
## 759                  Winx Club         4                     0
18523991
## 331           Winnie the Pooh         4              134428350
127960901
## 415                    Willard         4               78459917
0
## 703               Wilden Kerle         4                     0
29700000
## 448                Wild Things         4               56541184
25781400
## 506                Wild Orchid         4               24429833
0
## 728                 Wild Geese         4                     0
0
## 321     Why Did I Get Married         4              139201554
1464868
## 342          Whole Nine Yards         4              117947214
```

```
38265375
## 417               White Noise         4               77884523
44243567
## 384                White Fang         4               92510027
0
## 700         When Love Happens         4                      0
7573
## 681      When Calls the Heart         4                      0
0
## 705      What Would Jesus Do          4                      0
0
## 528            What the Bleep         4               15827867
0
## 738                Weiner Dog         4                      0
0
## 378       Weekend at Bernie s         4               94796401
0
## 148             Wayne s World         4              364264362
61400000
## 499                   Warlock         4               27045044
0
## 411        Wallace and Gromit         4               79861658
141524605
## 300               Wall Street         4              157738821
84957003
## 511                   Waiting         4               22388208
2548731
## 451            Viva Pedro Box         4               55454280
136184015
## 588                 Visiteurs         4                1604137
124095000
## 592         Vengeance Trilogy         4                1332579
28792536
## 452               VeggieTales         4               55253113
304603
## 478                Van Wilder         4               38425874
17956607
## 710                   Vacanze         4                      0
7901030
## 505                   Vacancy         4               24659987
5300000
## 630                     V H S         4                 139089
1882209
## 579 USA Land of Opportunities         4                2294126
13790369
## 361              Urban Legend         4              107753884
51560712
## 280              Untouchables         4              173607930
0
## 377         Universal Soldier         4               96157612
```

```
1809728
## 751                Undisputed         4                      0
0
## 177                 Underworld         4              315818775
286886720
## 196               Under Siege         4              281591505
127300000
##     tot.movies tot.years First.Year Last.Year
## 215          2       7.0       1998      2005
## 372          2      15.0       2001      2016
## 388          2       2.0       2009      2011
## 270          2       2.0       1988      1990
## 447          2       7.0       2004      2011
## 755          1       0.5       2016      2016
## 350          2       4.0       2010      2014
## 271          2      10.0       1998      2008
## 510          6      11.0       2003      2014
## 236          2       4.0       2013      2017
## 543          3       2.0       2004      2006
## 512          2       9.0       2005      2014
## 403          2       5.0       2004      2009
## 759          1       0.5       2012      2012
## 331          6      31.0       1977      2008
## 415          2      19.0       1971      1990
## 703          2       1.0       2007      2008
## 448          2      12.0       1998      2010
## 506          2       2.0       1990      1992
## 728          1       0.5       1985      1985
## 321          3       4.0       2006      2010
## 342          2       4.0       2000      2004
## 417          2       3.0       2005      2008
## 384          2       3.0       1991      1994
## 700          2       1.0       2015      2016
## 681          2       2.0       2013      2015
## 705          2       5.0       2010      2015
## 528          2       2.0       2004      2006
## 738          1       0.5       2013      2013
## 378          2       4.0       1989      1993
## 148          2       1.0       1992      1993
## 499          2       2.0       1991      1993
## 411          4      14.0       1995      2009
## 300          2      23.0       1987      2010
## 511          2       4.0       2005      2009
## 451          8      18.0       1986      2004
## 588          2       2.0       1996      1998
## 592          3       1.0       2005      2006
## 452          2       6.0       2002      2008
## 478          3       7.0       2002      2009
## 710          2       4.0       2011      2015
## 505          2       2.0       2007      2009
```

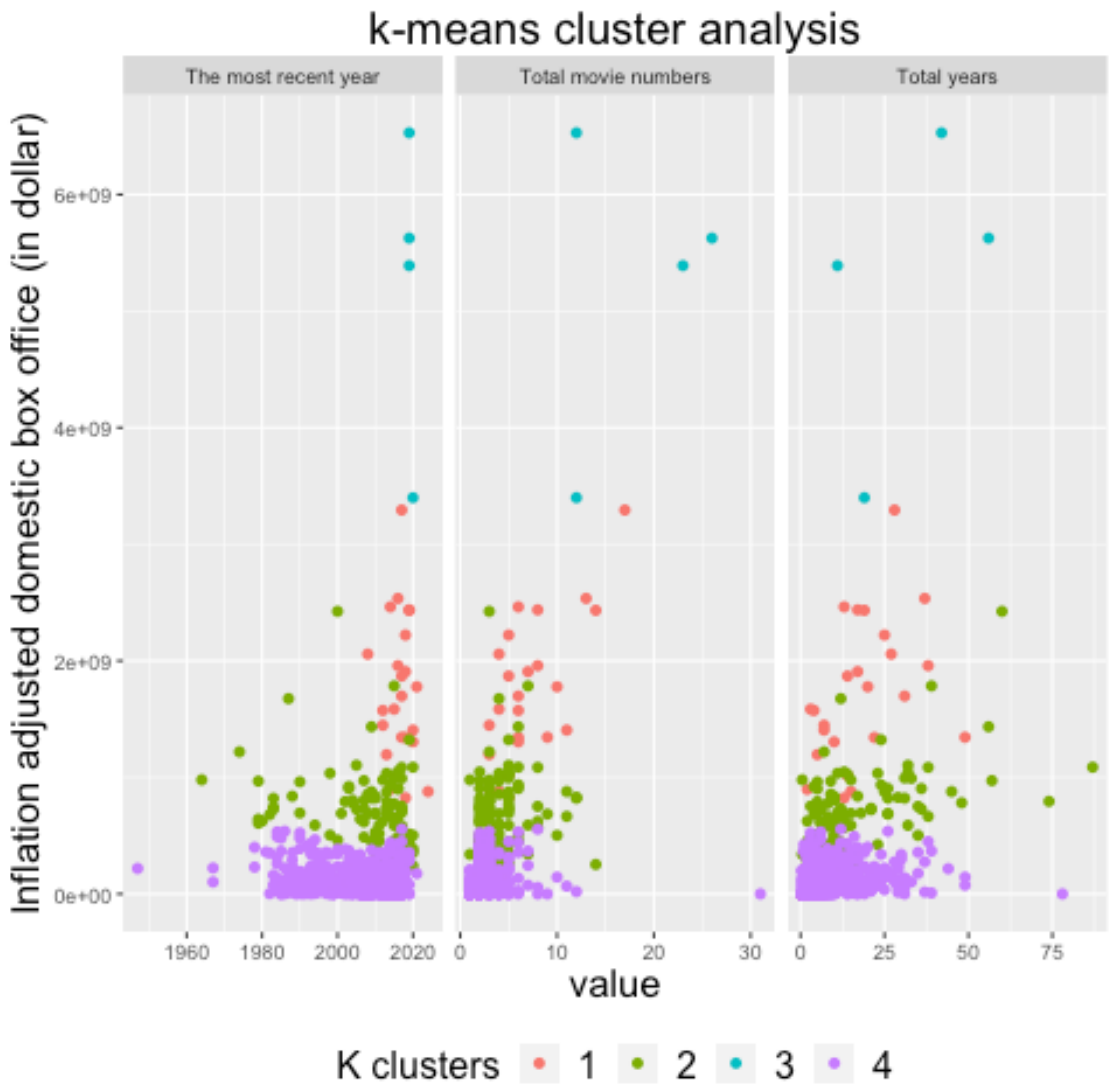
```
## 630          3       2.0       2012      2014
## 579          3       5.0       2004      2009
## 361          2       2.0       1998      2000
## 280          2      23.0       1987      2010
## 377          4      20.0       1992      2012
## 751          1       0.5       2007      2007
## 177          5      13.0       2003      2016
## 196          2       3.0       1992      1995
```

```
movfran.result[movfran.result$K.cluster == 1, -1]
```

```
##                     fran.title.clean K.cluster infl.adj.dom.boxoffice
## 9                              X Men         1             2432925375
## 21                          Twilight         1             1573729675
## 18                      Transformers         1             1697217057
## 52                        The Hobbit         1              897304916
## 13                          Superman         1             1958183722
## 6                           Star Trek         1             2534934135
## 8                          Spider Man         1             2436949137
## 14                              Shrek         1             1907496906
## 25                Planet of the Apes         1             1344102393
## 15          Pirates of the Caribbean         1             1869342701
## 7  Peter Jackson s Lord of the Rings         1             2462373441
## 26                Mission Impossible         1             1342145009
## 60                        Madagascar         1              824536103
## 11                     Jurassic Park         1             2220851105
## 30                          Iron Man         1             1195061980
## 12                     Indiana Jones         1             2057228548
## 39                           Ice Age         1             1015160281
## 20                      Hunger Games         1             1585025298
## 17              Fast and the Furious         1             1777326927
## 28                    Despicable Me         1             1304970563
## 24              DC Extended Universe         1             1405565825
## 22               Dark Knight Trilogy         1             1447157328
## 5                              Batman         1             3293294047
## 53                             Avatar         1              878514150
##    other.boxoffice tot.movies tot.years First.Year Last.Year
## 9       2972739360         14        19       2000      2019
## 21      1951548393          6         4       2008      2012
## 18      2927234228          6        31       1986      2017
## 52      2116000000          3         2       2012      2014
## 13      1254701047          8        38       1978      2016
## 6        865520289         13        37       1979      2016
## 8       2945378747          8        17       2002      2019
## 14      2127785519          7        17       2001      2018
## 25      1333798977          9        49       1968      2017
## 15      3043094095          5        14       2003      2017
## 7       4043374332          6        13       2001      2014
## 26      1866036048          6        22       1996      2018
## 60      1597299728          5        13       2005      2018
```

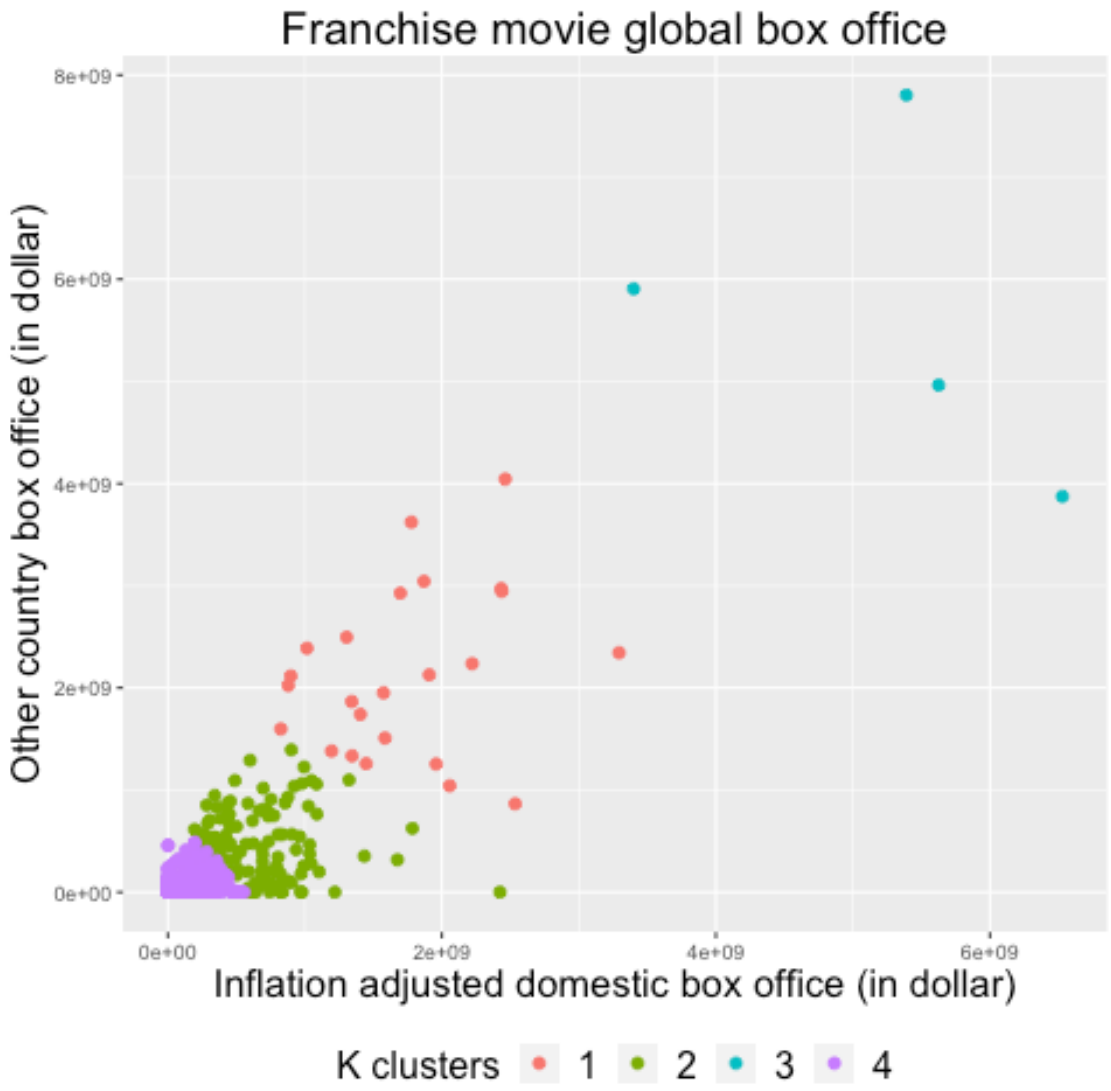
```
## 11      2236833067          5        25       1993      2018
## 30      1381690479          3         5       2008      2013
## 12      1041500294          4        27       1981      2008
## 39      2387647674          5        14       2002      2016
## 20      1508428615          4         3       2012      2015
## 17      3622617438         10        20       2001      2021
## 28      2495486726          6        10       2010      2020
## 24      1740203407         11         7       2013      2020
## 22      1258374208          3         7       2005      2012
## 5       2343815515         17        28       1989      2017
## 53      2023411357          4        15       2009      2024

# 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)))

## Warning: Use of `plot1.subset$K.cluster` is discouraged. Use `K.cluster`
## instead.

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)))
```

```
## Warning: Use of `movfran.result$infl.adj.dom.boxoffice` is discouraged.
Use
## `infl.adj.dom.boxoffice` instead.

## Warning: Use of `movfran.result$other.boxoffice` is discouraged. Use
## `other.boxoffice` instead.

## Warning: Use of `movfran.result$K.cluster` is discouraged. Use `K.cluster`
## instead.

dev.off()

## quartz_off_screen
##                 2

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

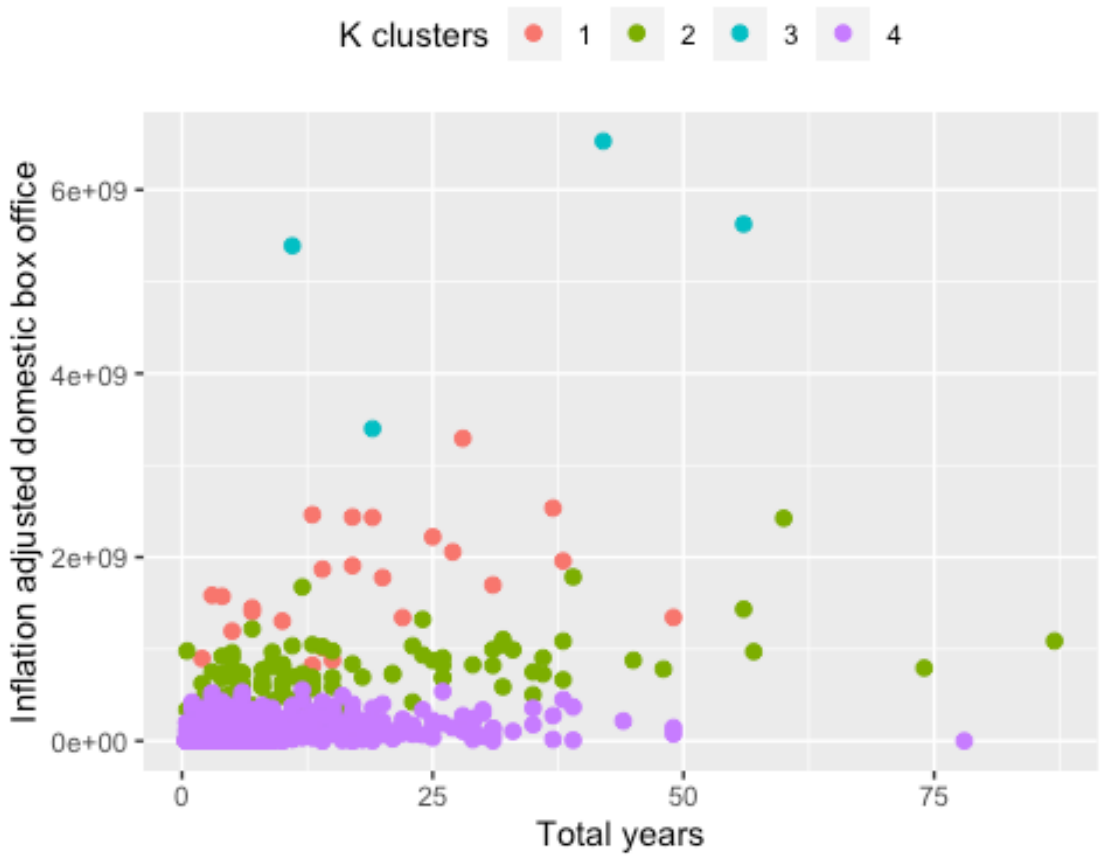



```
# Visualize the individual graphs
ggplot(movfran.result, aes(movfran.result$tot.years,
                           movfran.result$infl.adj.dom.boxoffice,
                           col = as.factor(movfran.result$K.cluster))) +
  geom_point(shape = 19, size = 2) +
  ylab("Inflation adjusted domestic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

```
## Warning: Use of `movfran.result$tot.years` is discouraged. Use `tot.years`
## instead.
```

```
## Warning: Use of `movfran.result$infl.adj.dom.boxoffice` is discouraged.
Use
## `infl.adj.dom.boxoffice` instead.
```

```
## Warning: Use of `movfran.result$K.cluster` is discouraged. Use `K.cluster`
## instead.
```

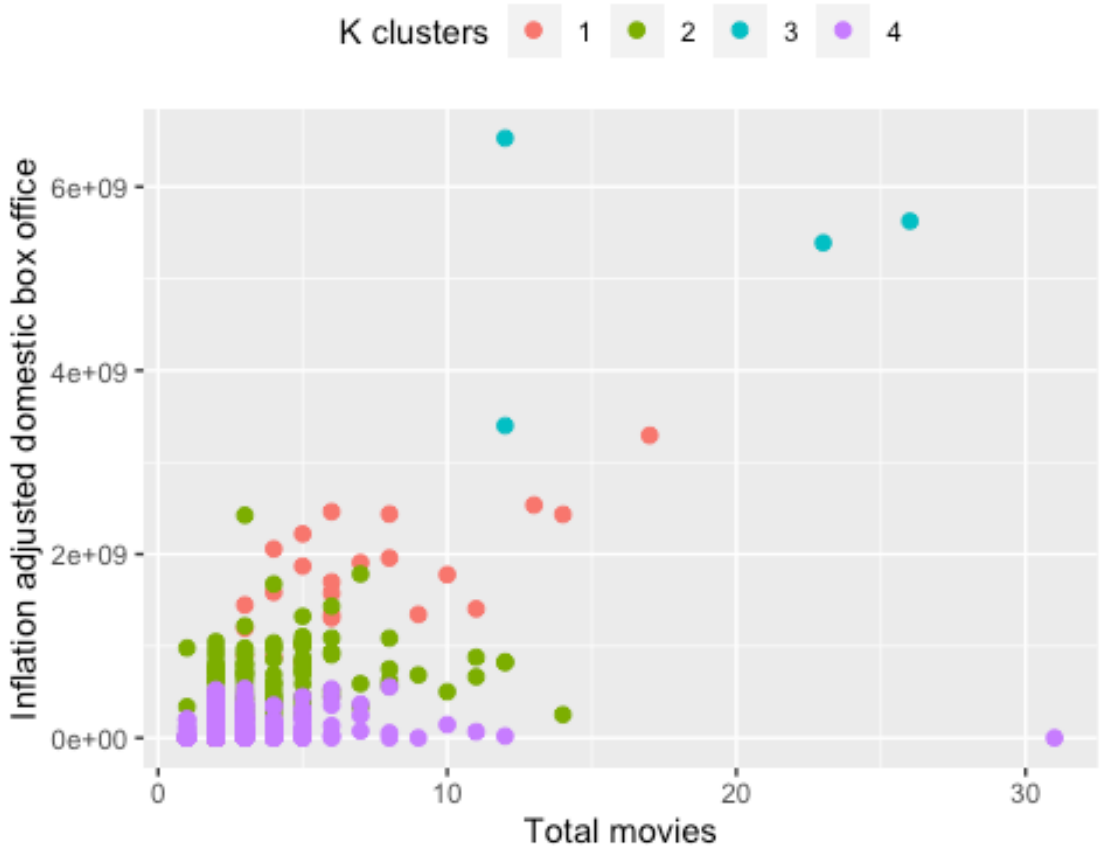



```
ggplot(movfran.result, aes(movfran.result$tot.movies,
                           movfran.result$infl.adj.dom.boxoffice,
                           col = as.factor(movfran.result$K.cluster))) +
  geom_point(shape = 19, size = 2) +
  ylab("Inflation adjusted domestic box office") +
  xlab("Total movies")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

```
## Warning: Use of `movfran.result$tot.movies` is discouraged. Use
`tot.movies`
## instead.
```

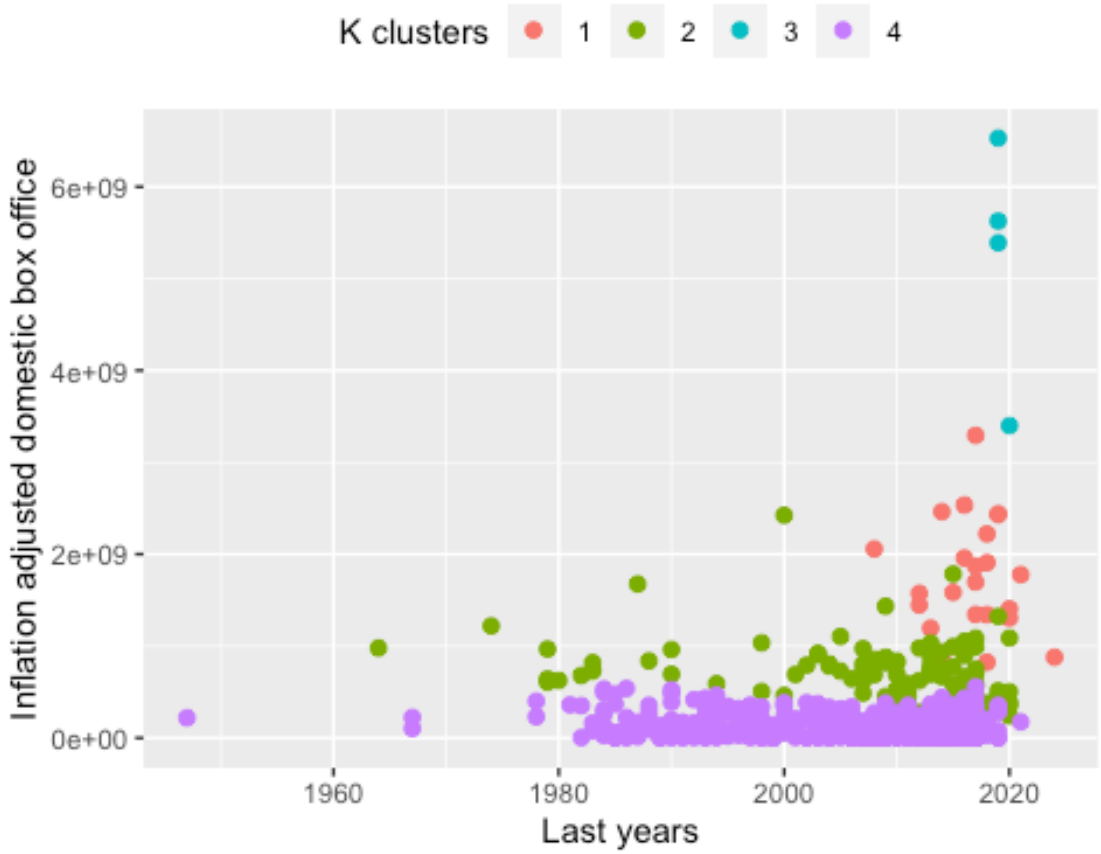
```
## Warning: Use of `movfran.result$infl.adj.dom.boxoffice` is discouraged.
Use
## `infl.adj.dom.boxoffice` instead.
```

```
## Warning: Use of `movfran.result$K.cluster` is discouraged. Use `K.cluster`
## instead.
```




```
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" ))
```

```
## Warning: Use of `movfran.result$Last.Year` is discouraged. Use `Last.Year`
## instead.

## Warning: Use of `movfran.result$infl.adj.dom.boxoffice` is discouraged.
Use
## `infl.adj.dom.boxoffice` instead.

## Warning: Use of `movfran.result$K.cluster` is discouraged. Use `K.cluster`
## instead.
```




```
knitr::opts_chunk$set(echo = TRUE)

################################################################################
#
#              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"
##  [3] "num_critic_for_reviews"     "duration"
##  [5] "director_facebook_likes"    "actor_3_facebook_likes"
##  [7] "actor_2_name"               "actor_1_facebook_likes"
##  [9] "genres"                     "actor_1_name"
## [11] "movie_title"                "num_voted_users"
## [13] "cast_total_facebook_likes" "actor_3_name"
## [15] "facenumber_in_poster"       "plot_keywords"
## [17] "movie_imdb_link"            "num_user_for_reviews"
```

```
## [19] "language"                  "country"
## [21] "content_rating"            "budget"
## [23] "actor_2_facebook_likes"    "imdb_score"
## [25] "aspect_ratio"              "movie_facebook_likes"
## [27] "movie.gross"               "movie.year"
## [29] "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"
##  [3] "num_critic_for_reviews"    "duration"
##  [5] "director_facebook_likes"   "actor_3_facebook_likes"
##  [7] "actor_2_name"              "actor_1_facebook_likes"
##  [9] "genres"                    "actor_1_name"
## [11] "num_voted_users"           "cast_total_facebook_likes"
## [13] "actor_3_name"              "facenumber_in_poster"
## [15] "plot_keywords"             "movie_imdb_link"
## [17] "num_user_for_reviews"      "language"
## [19] "country"                   "content_rating"
## [21] "budget"                    "actor_2_facebook_likes"
## [23] "imdb_score"                "aspect_ratio"
## [25] "movie_facebook_likes"      "movie.year"
## [27] "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]))
```

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

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

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

```r
# 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"
```

```r
# 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"
##  [3] "director_name"             "actor_2_name"
##  [5] "genres"                    "actor_1_name"
##  [7] "actor_3_name"              "plot_keywords"
##  [9] "movie_imdb_link"           "language"
## [11] "country"                   "content_rating"
## [13] "movie.year"                "num_critic_for_reviews"
## [15] "duration"                  "director_facebook_likes"
## [17] "actor_3_facebook_likes"    "actor_1_facebook_likes"
## [19] "num_voted_users"           "cast_total_facebook_likes"
## [21] "facenumber_in_poster"      "num_user_for_reviews"
## [23] "budget"                    "actor_2_facebook_likes"
## [25] "imdb_score"                "aspect_ratio"
## [27] "movie_facebook_likes"      "adjust.gross"
```

```r
# 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
```

```r
# 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])
```

```r
# 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(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"
##  [3] "director_facebook_likes"   "actor_3_facebook_likes"
##  [5] "actor_1_facebook_likes"    "num_voted_users"
##  [7] "cast_total_facebook_likes" "facenumber_in_poster"
##  [9] "num_user_for_reviews"      "budget"
## [11] "actor_2_facebook_likes"    "imdb_score"
## [13] "aspect_ratio"              "movie_facebook_likes"
## [15] "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(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)
summary(rgmodel.02)
```

```
##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = train.reg.02)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.094e+09  -2.857e+07  -3.381e+06   2.196e+07   1.094e+09
##
## Coefficients: (7 not defined because of singularities)
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -7.955e+08  1.577e+08  -5.044 4.90e-07 ***
## colorColor                 4.326e+07  9.181e+06   4.712 2.59e-06 ***
## languageBosnian            1.936e+07  9.503e+07   0.204 0.838610
## languageCantonese          2.149e+07  8.814e+07   0.244 0.807389
## languageCzech              4.816e+07  1.221e+08   0.394 0.693316
## languageDanish             2.198e+07  1.027e+08   0.214 0.830509
## languageDari              -4.952e+07  9.467e+07  -0.523 0.601010
## languageDutch             -3.943e+07  1.338e+08  -0.295 0.768189
## languageEnglish            2.182e+06  5.500e+07   0.040 0.968354
## languageFilipino           5.731e+07  9.493e+07   0.604 0.546109
## languageFrench             8.738e+06  5.812e+07   0.150 0.880505
## languageGerman            -4.102e+06  6.608e+07  -0.062 0.950499
## languageHebrew            -1.360e+07  1.443e+08  -0.094 0.924907
## languageHindi             -4.076e+07  1.300e+08  -0.314 0.753922
```

```
## languageHungarian          -6.585e+06  1.235e+08  -0.053 0.957469
## languageIndonesian          1.092e+07  9.565e+07   0.114 0.909136
## languageItalian            -1.250e+08  7.712e+07  -1.621 0.105231
## languageJapanese           -6.628e+07  6.767e+07  -0.979 0.327444
## languageKazakh              5.235e+07  9.503e+07   0.551 0.581763
## languageKorean             -6.574e+07  8.649e+07  -0.760 0.447323
## languageMandarin           -6.422e+05  8.204e+07  -0.008 0.993755
## languageMaya               -2.845e+07  9.477e+07  -0.300 0.764027
## languageMongolian          -2.738e+07  1.448e+08  -0.189 0.850027
## languageNorwegian          -4.183e+07  1.300e+08  -0.322 0.747668
## languagePersian             1.643e+08  1.114e+08   1.475 0.140371
## languagePortuguese         -6.289e+07  1.270e+08  -0.495 0.620429
## languageRomanian           -6.329e+07  1.454e+08  -0.435 0.663496
## languageSpanish            -3.257e+06  6.138e+07  -0.053 0.957679
## languageThai               -3.956e+07  1.096e+08  -0.361 0.718097
## languageVietnamese          3.781e+07  9.479e+07   0.399 0.690009
## languageZulu                4.543e+07  9.439e+07   0.481 0.630377
## countryArgentina           -1.342e+07  1.208e+08  -0.111 0.911608
## countryAustralia           -3.832e+07  1.099e+08  -0.348 0.727504
## countryBrazil                      NA         NA      NA       NA
## countryCanada              -2.533e+07  1.097e+08  -0.231 0.817405
## countryChina               -2.091e+07  1.242e+08  -0.168 0.866389
## countryColombia            -2.363e+06  1.362e+08  -0.017 0.986155
## countryCzech Republic      -4.038e+07  1.335e+08  -0.302 0.762361
## countryDenmark             -5.920e+07  1.158e+08  -0.511 0.609188
## countryFinland             -4.753e+07  1.355e+08  -0.351 0.725755
## countryFrance              -4.514e+07  1.096e+08  -0.412 0.680359
## countryGeorgia              1.081e+07  1.336e+08   0.081 0.935545
## countryGermany             -3.828e+07  1.096e+08  -0.349 0.727006
## countryGreece              -1.204e+08  1.339e+08  -0.900 0.368398
## countryHong Kong           -4.303e+07  1.169e+08  -0.368 0.712886
## countryHungary             -3.795e+07  1.335e+08  -0.284 0.776312
## countryIceland              7.314e+06  1.335e+08   0.055 0.956307
## countryIndia                       NA         NA      NA       NA
## countryIndonesia           -6.288e+07  1.549e+08  -0.406 0.684762
## countryIran                -2.341e+08  1.387e+08  -1.688 0.091617 .
## countryIreland             -3.589e+07  1.216e+08  -0.295 0.767889
## countryIsrael                      NA         NA      NA       NA
## countryItaly               -8.601e+07  1.159e+08  -0.742 0.457964
## countryJapan               -3.253e+07  1.139e+08  -0.286 0.775159
## countryMexico              -2.767e+07  1.144e+08  -0.242 0.808917
## countryNetherlands                 NA         NA      NA       NA
## countryNew Line            -2.862e+07  1.335e+08  -0.214 0.830285
## countryNew Zealand         -1.550e+07  1.136e+08  -0.136 0.891458
## countryNorway                      NA         NA      NA       NA
## countryPeru                 4.638e+07  1.339e+08   0.346 0.729152
## countryRomania                     NA         NA      NA       NA
## countryRussia                      NA         NA      NA       NA
## countrySouth Africa        -3.646e+07  1.220e+08  -0.299 0.765060
## countrySouth Korea         -6.091e+07  1.180e+08  -0.516 0.605646
```

```
## countrySpain               -4.665e+07  1.114e+08  -0.419 0.675501
## countryTaiwan              -7.620e+07  1.476e+08  -0.516 0.605628
## countryThailand            -2.399e+06  1.336e+08  -0.018 0.985668
## countryUK                  -5.197e+07  1.092e+08  -0.476 0.634150
## countryUSA                 -2.106e+07  1.090e+08  -0.193 0.846844
## countryWest Germany        -2.086e+08  1.413e+08  -1.476 0.139962
## content_ratingG             8.078e+08  4.964e+07  16.274  < 2e-16 ***
## content_ratingNC-17         7.006e+08  5.543e+07  12.638  < 2e-16 ***
## content_ratingPG            7.705e+08  4.964e+07  15.523  < 2e-16 ***
## content_ratingPG-13         7.433e+08  4.965e+07  14.972  < 2e-16 ***
## content_ratingR             7.074e+08  4.962e+07  14.256  < 2e-16 ***
## content_ratingUnrated       7.434e+08  5.167e+07  14.388  < 2e-16 ***
## movie.year1929              8.403e+08  1.254e+08   6.702 2.53e-11 ***
## movie.year1936             -6.257e+07  1.155e+08  -0.542 0.588061
## movie.year1939              2.147e+09  1.049e+08  20.469  < 2e-16 ***
## movie.year1946              2.948e+08  1.142e+08   2.581 0.009897 **
## movie.year1948              7.791e+08  1.257e+08   6.198 6.67e-10 ***
## movie.year1953              3.108e+08  1.145e+08   2.715 0.006670 **
## movie.year1954              5.175e+07  1.013e+08   0.511 0.609614
## movie.year1957              1.613e+08  1.156e+08   1.395 0.163179
## movie.year1959              1.775e+08  1.141e+08   1.556 0.119852
## movie.year1960              1.882e+08  1.152e+08   1.633 0.102577
## movie.year1962              3.952e+08  1.051e+08   3.759 0.000174 ***
## movie.year1963              1.039e+09  1.138e+08   9.129  < 2e-16 ***
## movie.year1964              8.343e+08  1.074e+08   7.765 1.18e-14 ***
## movie.year1965              9.571e+08  9.802e+07   9.765  < 2e-16 ***
## movie.year1966              7.960e+08  1.340e+08   5.940 3.25e-09 ***
## movie.year1967              1.035e+09  1.259e+08   8.220 3.25e-16 ***
## movie.year1968              1.078e+08  1.026e+08   1.051 0.293482
## movie.year1969              3.383e+08  1.019e+08   3.318 0.000919 ***
## movie.year1970              1.502e+07  9.806e+07   0.153 0.878253
## movie.year1971              2.184e+08  1.155e+08   1.891 0.058762 .
## movie.year1972              3.581e+07  1.180e+08   0.304 0.761482
## movie.year1973              1.005e+09  1.153e+08   8.719  < 2e-16 ***
## movie.year1974              3.671e+08  9.681e+07   3.792 0.000153 ***
## movie.year1975              9.166e+07  1.022e+08   0.897 0.369753
## movie.year1976              4.290e+07  1.154e+08   0.372 0.710167
## movie.year1977              4.528e+08  9.436e+07   4.799 1.69e-06 ***
## movie.year1978              1.875e+08  9.437e+07   1.987 0.047084 *
## movie.year1979              7.916e+07  9.702e+07   0.816 0.414668
## movie.year1980              1.374e+08  8.924e+07   1.540 0.123668
## movie.year1981              1.427e+08  8.961e+07   1.592 0.111427
## movie.year1982              7.407e+07  8.877e+07   0.834 0.404164
## movie.year1983              1.024e+08  8.930e+07   1.147 0.251485
## movie.year1984              1.232e+08  8.827e+07   1.396 0.162907
## movie.year1985              1.185e+08  8.907e+07   1.331 0.183456
## movie.year1986              1.062e+08  8.855e+07   1.199 0.230616
## movie.year1987              5.634e+07  8.776e+07   0.642 0.520919
## movie.year1988              5.774e+07  8.820e+07   0.655 0.512728
## movie.year1989              7.412e+07  8.761e+07   0.846 0.397629
```

```
## movie.year1990              1.346e+08  8.859e+07   1.519 0.128814
## movie.year1991              6.822e+07  8.759e+07   0.779 0.436101
## movie.year1992              9.031e+07  8.756e+07   1.031 0.302414
## movie.year1993              5.273e+07  8.707e+07   0.606 0.544790
## movie.year1994              4.407e+07  8.720e+07   0.505 0.613320
## movie.year1995              5.538e+07  8.715e+07   0.635 0.525232
## movie.year1996              5.722e+07  8.673e+07   0.660 0.509460
## movie.year1997              5.711e+07  8.671e+07   0.659 0.510221
## movie.year1998              3.781e+07  8.661e+07   0.437 0.662452
## movie.year1999              3.314e+07  8.649e+07   0.383 0.701610
## movie.year2000              3.040e+07  8.636e+07   0.352 0.724831
## movie.year2001              3.116e+07  8.639e+07   0.361 0.718403
## movie.year2002              3.041e+07  8.638e+07   0.352 0.724856
## movie.year2003              3.513e+07  8.639e+07   0.407 0.684324
## movie.year2004              1.009e+07  8.609e+07   0.117 0.906751
## movie.year2005              1.019e+07  8.636e+07   0.118 0.906085
## movie.year2006              3.935e+06  8.615e+07   0.046 0.963567
## movie.year2007              1.662e+07  8.638e+07   0.192 0.847477
## movie.year2008              1.350e+06  8.635e+07   0.016 0.987533
## movie.year2009              1.187e+07  8.614e+07   0.138 0.890371
## movie.year2010              6.065e+06  8.637e+07   0.070 0.944027
## movie.year2011              2.414e+06  8.637e+07   0.028 0.977702
## movie.year2012              1.246e+07  8.630e+07   0.144 0.885227
## movie.year2013              1.826e+07  8.633e+07   0.212 0.832458
## movie.year2014              2.002e+07  8.641e+07   0.232 0.816781
## movie.year2015              3.537e+07  8.652e+07   0.409 0.682673
## movie.year2016              5.189e+07  8.695e+07   0.597 0.550728
## num_critic_for_reviews      1.519e+05  2.590e+04   5.864 5.11e-09 ***
## duration                    3.466e+05  8.356e+04   4.149 3.46e-05 ***
## director_facebook_likes    -1.913e+03  5.267e+02  -3.632 0.000287 ***
## actor_3_facebook_likes     -1.340e+04  2.473e+03  -5.419 6.56e-08 ***
## actor_1_facebook_likes     -1.118e+04  1.482e+03  -7.546 6.23e-14 ***
## num_voted_users             3.444e+02  2.049e+01  16.812  < 2e-16 ***
## cast_total_facebook_likes   1.112e+04  1.481e+03   7.507 8.40e-14 ***
## facenumber_in_poster       -1.204e+06  7.628e+05  -1.579 0.114471
## num_user_for_reviews        1.187e+04  7.319e+03   1.622 0.104881
## budget                      1.610e-02  7.623e-03   2.111 0.034840 *
## actor_2_facebook_likes     -1.077e+04  1.603e+03  -6.718 2.27e-11 ***
## imdb_score                 -3.471e+06  1.924e+06  -1.804 0.071308 .
## aspect_ratio                1.859e+06  6.410e+06   0.290 0.771892
## movie_facebook_likes       -8.102e+02  1.145e+02  -7.078 1.90e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5.873702e+15)
##
##     Null deviance: 4.1571e+19  on 2638  degrees of freedom
## Residual deviance: 1.4631e+19  on 2491  degrees of freedom
## AIC: 103455
```

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

# 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
```

```r
categorical variables
rgmodel.02 = glm(adjust.gross ~ ., family = gaussian, data = train.reg.02)
summary(rgmodel.02)
```

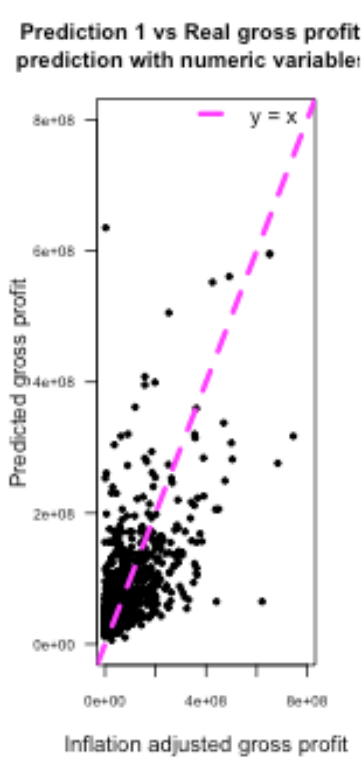
```
##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = train.reg.02)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -1.093e+09  -2.837e+07  -4.192e+06   2.141e+07   1.093e+09
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -4.022e+07  1.457e+07  -2.760 0.005819 **
## colorBlack and White      -4.408e+07  9.141e+06  -4.822 1.50e-06 ***
## languageAboriginal        -1.904e+07  5.468e+07  -0.348 0.727705
## languageBosnian            1.829e+07  7.768e+07   0.235 0.813881
## languageCantonese          3.640e+06  5.507e+07   0.066 0.947302
## languageCzech              3.367e+07  7.734e+07   0.435 0.663374
## languageDanish            -1.330e+07  7.750e+07  -0.172 0.863720
## languageDari              -3.511e+07  5.482e+07  -0.640 0.521951
## languageDutch             -1.251e+07  5.485e+07  -0.228 0.819551
## languageFilipino           6.339e+07  7.785e+07   0.814 0.415617
## languageFrench            -8.958e+06  1.619e+07  -0.553 0.580019
## languageGerman            -3.872e+07  3.226e+07  -1.200 0.230235
## languageHebrew             1.154e+07  7.737e+07   0.149 0.881461
## languageHindi             -1.499e+07  4.479e+07  -0.335 0.737929
## languageHungarian         -1.860e+07  7.972e+07  -0.233 0.815549
## languageIndonesian        -1.702e+07  5.515e+07  -0.309 0.757649
## languageItalian           -1.812e+08  3.773e+07  -4.802 1.66e-06 ***
## languageJapanese          -7.306e+07  2.680e+07  -2.726 0.006459 **
## languageKazakh             3.256e+07  7.735e+07   0.421 0.673807
## languageKorean            -1.026e+08  4.982e+07  -2.059 0.039637 *
## languageMandarin          -5.047e+06  2.757e+07  -0.183 0.854773
## languageMaya              -2.290e+07  7.737e+07  -0.296 0.767260
## languageMongolian          9.030e+05  7.737e+07   0.012 0.990688
## languageNorwegian         -1.493e+07  4.490e+07  -0.333 0.739478
## languagePersian           -4.190e+07  4.505e+07  -0.930 0.352484
## languagePortuguese        -3.607e+07  3.484e+07  -1.035 0.300588
## languageRomanian          -2.727e+07  7.850e+07  -0.347 0.728294
## languageSpanish           -7.816e+06  1.797e+07  -0.435 0.663589
## languageThai              -1.409e+07  5.534e+07  -0.255 0.799006
## languageVietnamese         4.301e+07  7.743e+07   0.555 0.578640
## languageZulu               1.977e+07  7.730e+07   0.256 0.798155
## content_ratingApproved    -7.147e+08  4.978e+07 -14.357  < 2e-16 ***
## content_ratingG            1.007e+08  1.045e+07   9.633  < 2e-16 ***
## content_ratingNC-17       -5.859e+06  2.444e+07  -0.240 0.810551
## content_ratingPG           6.267e+07  4.748e+06  13.199  < 2e-16 ***
## content_ratingPG-13        3.635e+07  3.552e+06  10.233  < 2e-16 ***
```

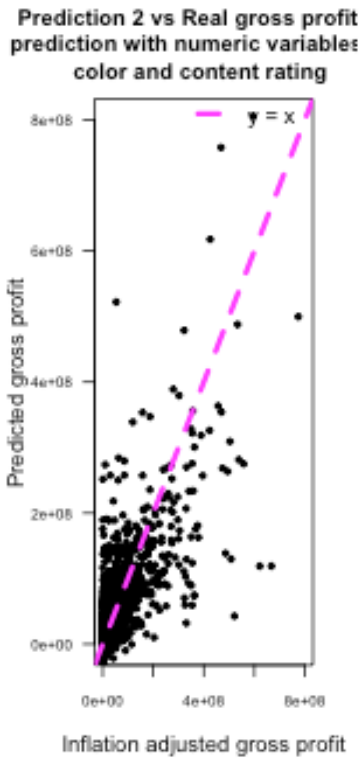
```
## content_ratingUnrated      2.806e+07  1.364e+07   2.058 0.039719 *
## movie.year1927             1.777e+07  8.492e+07   0.209 0.834307
## movie.year1929             8.376e+08  9.227e+07   9.078  < 2e-16 ***
## movie.year1936            -7.129e+07  7.850e+07  -0.908 0.363880
## movie.year1939             2.143e+09  6.108e+07  35.090  < 2e-16 ***
## movie.year1946             2.969e+08  7.907e+07   3.755 0.000177 ***
## movie.year1948             7.768e+08  9.197e+07   8.446  < 2e-16 ***
## movie.year1953             3.114e+08  7.842e+07   3.971 7.37e-05 ***
## movie.year1954             5.040e+07  5.861e+07   0.860 0.389889
## movie.year1957             1.255e+08  7.754e+07   1.618 0.105786
## movie.year1959             1.787e+08  7.888e+07   2.265 0.023595 *
## movie.year1960             1.767e+08  7.802e+07   2.265 0.023622 *
## movie.year1962             3.620e+08  6.077e+07   5.956 2.94e-09 ***
## movie.year1963             1.009e+09  7.470e+07  13.513  < 2e-16 ***
## movie.year1964             8.246e+08  6.423e+07  12.839  < 2e-16 ***
## movie.year1965             9.470e+08  4.709e+07  20.110  < 2e-16 ***
## movie.year1966             7.848e+08  1.024e+08   7.664 2.56e-14 ***
## movie.year1967             1.003e+09  9.193e+07  10.912  < 2e-16 ***
## movie.year1968             7.040e+07  5.614e+07   1.254 0.209907
## movie.year1969             3.161e+08  5.505e+07   5.743 1.04e-08 ***
## movie.year1970             5.948e+06  4.599e+07   0.129 0.897110
## movie.year1971             1.800e+08  7.736e+07   2.327 0.020068 *
## movie.year1972             2.449e+07  8.113e+07   0.302 0.762734
## movie.year1973             9.968e+08  7.736e+07  12.886  < 2e-16 ***
## movie.year1974             3.582e+08  4.532e+07   7.905 3.98e-15 ***
## movie.year1975             6.575e+07  5.544e+07   1.186 0.235729
## movie.year1976             3.547e+07  7.737e+07   0.458 0.646653
## movie.year1977             4.295e+08  3.957e+07  10.854  < 2e-16 ***
## movie.year1978             1.371e+08  3.538e+07   3.875 0.000109 ***
## movie.year1979             7.290e+07  4.541e+07   1.605 0.108510
## movie.year1980             1.289e+08  2.454e+07   5.253 1.63e-07 ***
## movie.year1981             1.089e+08  2.470e+07   4.408 1.09e-05 ***
## movie.year1982             6.558e+07  2.249e+07   2.916 0.003579 **
## movie.year1983             8.507e+07  2.455e+07   3.465 0.000539 ***
## movie.year1984             1.031e+08  2.032e+07   5.074 4.18e-07 ***
## movie.year1985             1.027e+08  2.442e+07   4.204 2.71e-05 ***
## movie.year1986             8.801e+07  2.183e+07   4.031 5.72e-05 ***
## movie.year1987             3.902e+07  1.811e+07   2.154 0.031338 *
## movie.year1988             4.687e+07  2.018e+07   2.323 0.020264 *
## movie.year1989             6.217e+07  1.752e+07   3.548 0.000395 ***
## movie.year1990             1.253e+08  2.191e+07   5.716 1.22e-08 ***
## movie.year1991             5.770e+07  1.744e+07   3.309 0.000949 ***
## movie.year1992             7.616e+07  1.698e+07   4.486 7.58e-06 ***
## movie.year1993             3.830e+07  1.458e+07   2.628 0.008649 **
## movie.year1994             3.389e+07  1.500e+07   2.259 0.023971 *
## movie.year1995             4.162e+07  1.496e+07   2.781 0.005461 **
## movie.year1996             4.417e+07  1.210e+07   3.650 0.000268 ***
## movie.year1997             4.371e+07  1.186e+07   3.686 0.000232 ***
## movie.year1998             2.404e+07  1.128e+07   2.131 0.033212 *
## movie.year1999             1.797e+07  1.055e+07   1.704 0.088540 .
```

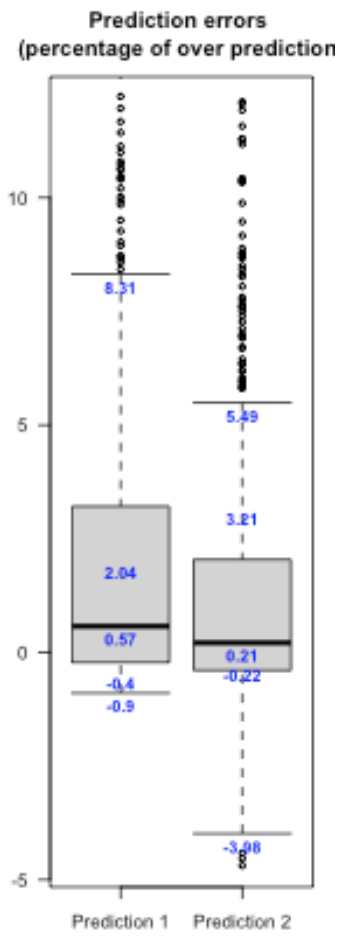
```
## movie.year2000              1.615e+07  9.983e+06   1.618 0.105879
## movie.year2001              1.784e+07  9.921e+06   1.798 0.072325 .
## movie.year2002              1.515e+07  9.741e+06   1.555 0.120020
## movie.year2003              2.295e+07  1.048e+07   2.190 0.028590 *
## movie.year2004             -3.335e+06  9.903e+06  -0.337 0.736312
## movie.year2005             -4.624e+06  9.694e+06  -0.477 0.633446
## movie.year2006             -1.065e+07  9.739e+06  -1.093 0.274469
## movie.year2007              1.160e+06  1.003e+07   0.116 0.907938
## movie.year2008             -1.217e+07  9.563e+06  -1.273 0.203302
## movie.year2010             -6.447e+06  9.907e+06  -0.651 0.515280
## movie.year2011             -1.079e+07  9.796e+06  -1.101 0.270893
## movie.year2012              5.420e+05  1.001e+07   0.054 0.956830
## movie.year2013              5.708e+06  1.027e+07   0.556 0.578355
## movie.year2014              8.082e+06  1.027e+07   0.787 0.431529
## movie.year2015              2.346e+07  1.056e+07   2.222 0.026341 *
## movie.year2016              3.590e+07  1.358e+07   2.643 0.008257 **
## num_critic_for_reviews      1.451e+05  2.566e+04   5.655 1.74e-08 ***
## duration                    3.073e+05  8.121e+04   3.784 0.000158 ***
## director_facebook_likes    -1.674e+03  5.257e+02  -3.185 0.001466 **
## actor_3_facebook_likes     -1.456e+04  2.466e+03  -5.905 3.99e-09 ***
## actor_1_facebook_likes     -1.197e+04  1.479e+03  -8.098 8.60e-16 ***
## num_voted_users             3.525e+02  2.041e+01  17.265  < 2e-16 ***
## cast_total_facebook_likes   1.192e+04  1.478e+03   8.066 1.11e-15 ***
## facenumber_in_poster       -1.163e+06  7.609e+05  -1.528 0.126678
## num_user_for_reviews        1.226e+04  7.313e+03   1.677 0.093682 .
## budget                      1.672e-02  7.649e-03   2.186 0.028898 *
## actor_2_facebook_likes     -1.146e+04  1.602e+03  -7.157 1.08e-12 ***
## imdb_score                 -4.484e+06  1.900e+06  -2.360 0.018334 *
## movie_facebook_likes       -8.274e+02  1.140e+02  -7.256 5.27e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5.920072e+15)
##
##     Null deviance: 4.1571e+19  on 2638  degrees of freedom
## Residual deviance: 1.4942e+19  on 2524  degrees of freedom
## AIC: 103444
##
## Number of Fisher Scoring iterations: 2

# 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(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

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 = "/"),
                        auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```



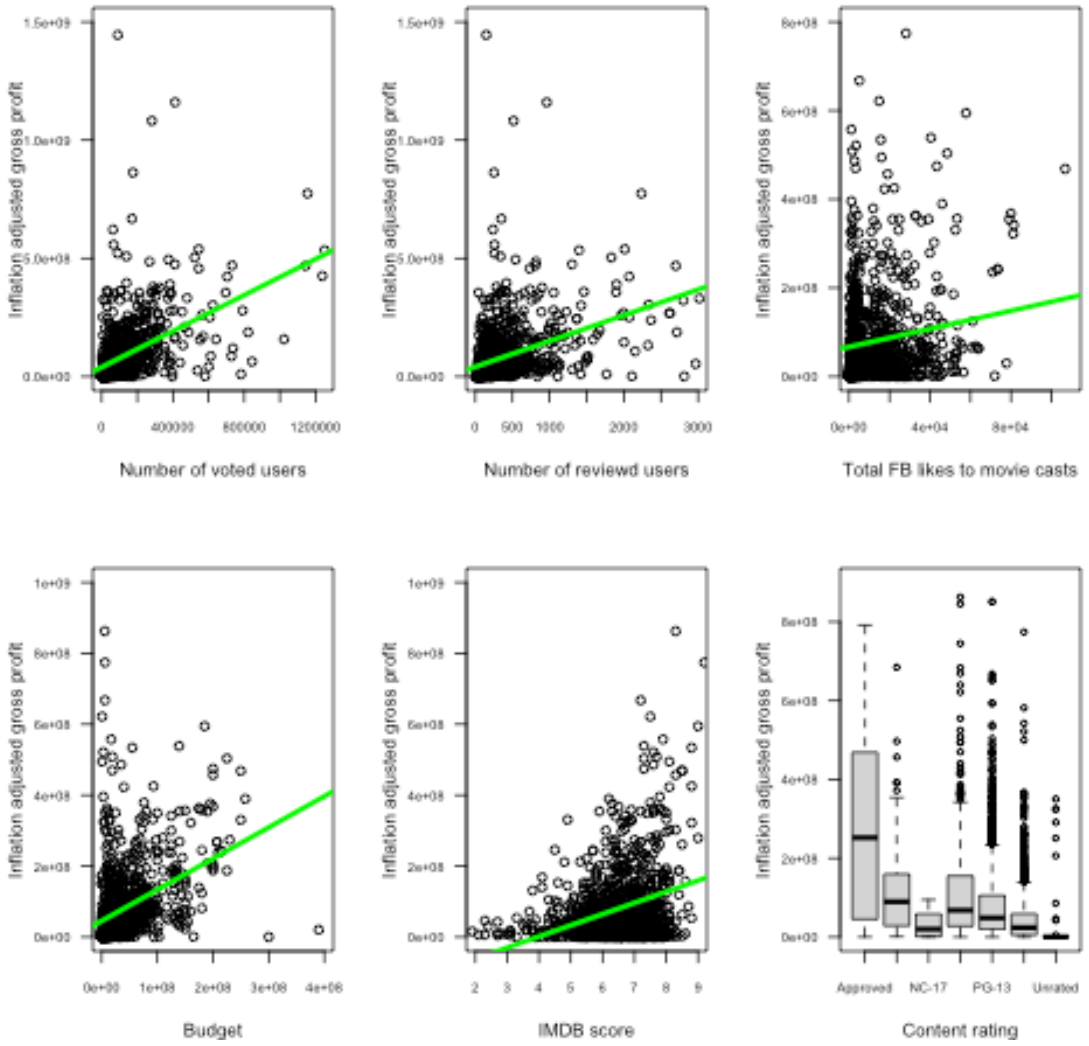

```
colnames(bop01$stats) = c("Prediction 1", "Prediction 2")
bop01$stats  # show the errors box plot statistics for both predictions
```

```
##      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),
```

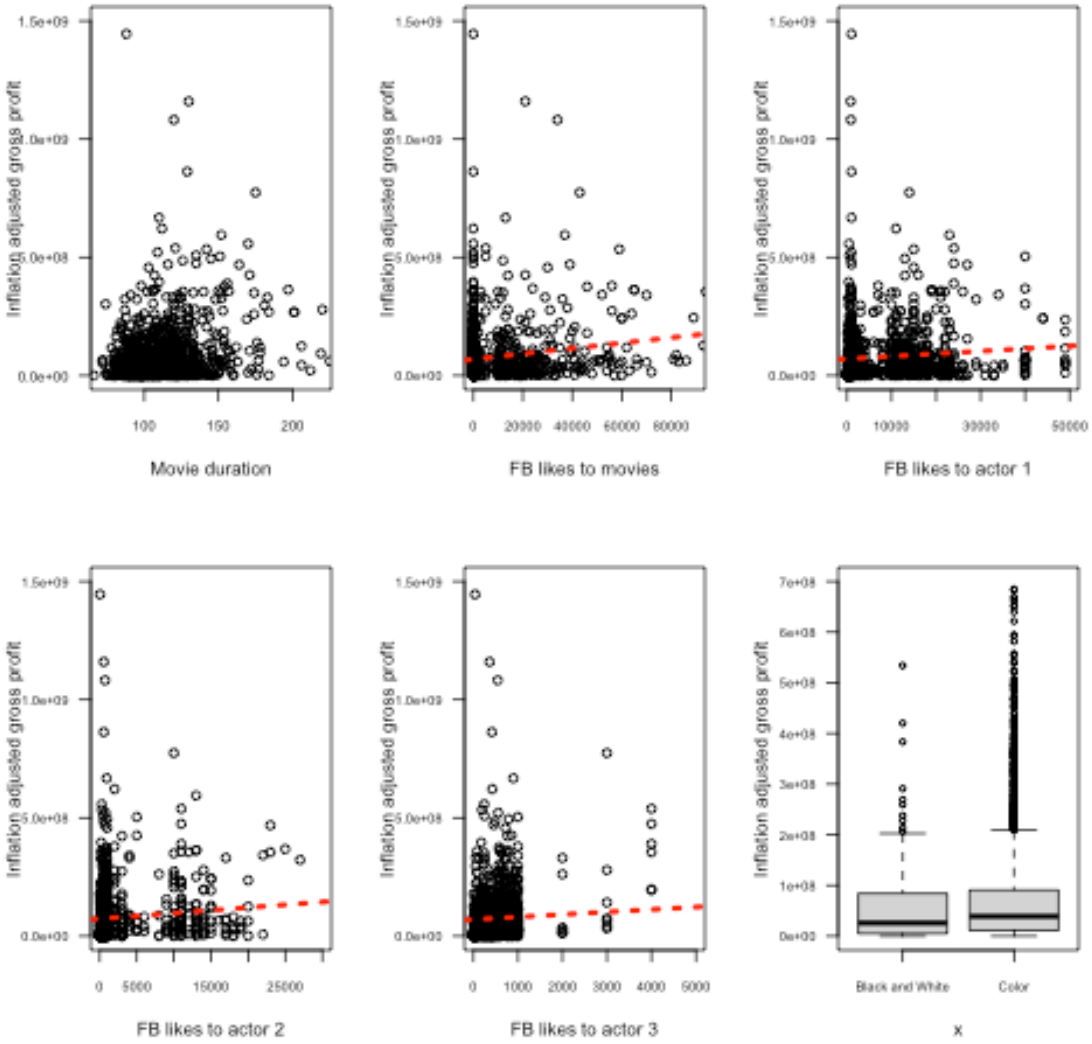
```
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Number of voted users")
abline(lm(adjgross.testreg02 ~ test.reg.02$num_voted_users),
       col = "green", lwd = 3)

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 = "/"),
```

```
                        auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```




```
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 = "/"),
                        auto_pdf = getOption("knitr.graphics.auto_pdf",
TRUE))
```




```
##############################################################################
#
#      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"
##  [3] "num_critic_for_reviews"    "duration"
##  [5] "director_facebook_likes"   "actor_3_facebook_likes"
##  [7] "actor_2_name"              "actor_1_facebook_likes"
##  [9] "genres"                    "actor_1_name"
## [11] "num_voted_users"           "cast_total_facebook_likes"
## [13] "actor_3_name"              "facenumber_in_poster"
```

```
## [15] "plot_keywords"             "movie_imdb_link"
## [17] "num_user_for_reviews"      "language"
## [19] "country"                   "content_rating"
## [21] "budget"                    "actor_2_facebook_likes"
## [23] "imdb_score"                "aspect_ratio"
## [25] "movie_facebook_likes"      "movie.gross"
## [27] "movie.year"                "adjust.gross"
## [29] "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"
##  [3] "num_critic_for_reviews"    "duration"
##  [5] "director_facebook_likes"   "actor_3_facebook_likes"
##  [7] "actor_2_name"              "actor_1_facebook_likes"
##  [9] "actor_1_name"              "num_voted_users"
## [11] "cast_total_facebook_likes" "actor_3_name"
## [13] "facenumber_in_poster"      "num_user_for_reviews"
## [15] "language"                  "country"
## [17] "content_rating"            "budget"
## [19] "actor_2_facebook_likes"    "imdb_score"
## [21] "aspect_ratio"              "movie_facebook_likes"
## [23] "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
table(franmovie.regrset$director_name)[1:50]

##
##                          Adam Marcus                           Adam McKay
##                                    1                                    2
##                        Adam Shankman                     Agnieszka Holland
##                                    3                                    1
##                          Alan Metter                          Alan Parker
##                                    1                                    1
##                          Alan Taylor                     Alejandro Agresti
##                                    2                                    1
## Alejandro G. I\x92\xb1\x92\x8drritu                    Alessandro Carloni
##                                    2                                    1
##                       Alex Craig Mann                          Alex Gibney
##                                    1                                    1
##                          Alex Proyas                       Alexander Witt
##                                    1                                    1
##                       Alexandre Aja                    Alfonso Cuar\x92_n
##                                    1                                    1
##                     Alfred Hitchcock                       Andrew Adamson
##                                    1                                    7
##                         Andrew Davis                       Andrew Douglas
##                                    1                                    1
##                       Andrew Morahan                       Andrew Stanton
##                                    1                                    1
##                    Andrzej Bartkowiak                        Andy Fickman
##                                    1                                    1
##                         Andy Tennant                             Ang Lee
##                                    2                                    2
##                      Angela Robinson                   Angelina Jolie Pitt
##                                    1                                    1
##                       Annabel Jankel                         Anne Fletcher
##                                    1                                    1
##                         Anthony Bell                     Anthony Hemingway
##                                    1                                    1
##                       Anthony Hickox                     Anthony Minghella
##                                    1                                    2
##                        Anthony Russo                        Antoine Fuqua
##                                    2                                    2
##                        Anton Corbijn                       Antony Hoffman
##                                    1                                    1
##                       Barry Levinson                      Barry Sonnenfeld
##                                    1                                    5
##                        Beeban Kidron                         Ben Stiller
##                                    1                                    2
```

```
##                         Betty Thomas                          Bibo Bergeron
##                                    2                                      1
##                          Bill Condon                         Bille Woodruff
##                                    3                                      2
##                   Billy Bob Thornton                              Billy Ray
##                                    1                                      1
##                           Boaz Yakin                              Bob Clark
##                                    1                                      3
```

```
table(franmovie.regrset$actor_2_name)[1:50]
```

```
##
##          A. Michael Baldwin                Abbie Cornish
##                           1                            3
##                Adam Baldwin                   Adam Brown
##                           2                            4
##                 Adam Garcia                 Adam LeFevre
##                           1                            1
##                Adam Sandler                Adelaide Kane
##                           4                            1
##                Adhir Kalyan                  Adrian Paul
##                           1                            1
##             Adrienne Barbeau                Adrienne King
##                           1                            2
##              Aimee Teegarden                  Aisha Tyler
##                           1                            1
##                     Al Gore                     Al Leong
##                           1                            1
##                   Al Pacino                   Alan Ford
##                           1                            1
##                Alan Rickman                   Alan Ruck
##                           2                            2
##                Alanna Ubach                  Aldis Hodge
##                           3                            1
##               Aleksey Chadov                Alex Pettyfer
##                           1                            1
##                Alex Vincent                Alexa Davalos
##                           1                            1
##               Alexa PenaVega Alexander Skarsg\x92\xc7rd
##                           1                            1
##                Alfre Woodard                  Alice Braga
##                           1                            2
##              Alyson Hannigan                Alyson Stoner
##                           1                            2
##               Amber Valletta              America Ferrera
##                           1                            2
##                    Aml Ameen                  Amrish Puri
##                           1                            1
##                  Amy Poehler                  Amy Yasbeck
##                           1                            1
```

```
##             Andre Braugher             Andrew Fiscella
##                          2                           1
##            Andrew Garfield                    Andy Lau
##                          2                           2
##        Angelina Jolie Pitt           Annabelle Wallis
##                          2                           1
##             Anne Parillaud             Annet Mahendru
##                          1                           1
##            Anthony Hopkins           Anthony LaPaglia
##                          2                           1
##             Anushka Shetty             Aretha Franklin
##                          1                           1
```

```
table(franmovie.regrset$actor_1_name)[1:50]
```

```
##
##        Abbie Cornish         Adam Baldwin        Adam Goldberg
##                    1                    3                    2
##           Adam Scott         Aidan Turner            Al Pacino
##                    1                    4                    2
##         Alan Rickman        Albert Brooks          Alex Gibney
##                    2                    1                    1
##       Alexa PenaVega      Alexander Gould        Alfre Woodard
##                    3                    1                    2
##          Alice Braga        Alice Greczyn          Alice Krige
##                    2                    1                    1
##          Alicia Witt          Alison Brie        Alison Lohman
##                    3                    1                    1
##      Alyson Hannigan        Alyson Stoner        Amanda Schull
##                    2                    1                    1
##   Amber Stevens West      America Ferrera           Ami Ayalon
##                    2                    2                    1
##              Amos Oz          Amy Poehler      Andrew Fiscella
##                    1                    5                    2
##      Andrew Garfield      Andrew Robinson  Angelina Jolie Pitt
##                    1                    1                    7
##         Angus Scrimm       Angus T. Jones      Anjelica Huston
##                    1                    2                    4
##        Anna Kendrick        Anne Hathaway      Anthony Hopkins
##                    2                    5                    2
##        Antoni Corone         Antony Starr          Anwar Congo
##                    1                    1                    1
##      Ariana Richards         Arjun Rampal           Art Hindle
##                    1                    1                    1
##      Ashley Rickards          Bam Margera         Barry Watson
##                    3                    3                    1
##        Beau Mirchoff         Bella Thorne          Ben Gazzara
##                    1                    1                    1
## Benedict Cumberbatch  Benjamin A. Onyango
##                    2                    1
```

```
table(franmovie.regrset$actor_3_name)[1:50]

##
##        A.J. Buckley      Aaron Stanford       Adam Copeland          Adam
Trese
##                   1                   1                   1
1
##       Adrian Alonso      Agnes Bruckner            Al Leong            Al
Roker
##                   1                   1                   2
2
##      Alan D. Purwin          Alan David        Alanna Ubach       Albert
Finney
##                   2                   1                   1
1
##   Alessandro Nivola       Alex Borstein         Alex Winter       Alexa
Davalos
##                   1                   1                   2
2
##      Alexa PenaVega     Alexander Gould    Alexandra Callas Alexandre
Rodrigues
##                   1                   2                   1
1
##          Ali Hillis         Alice Krige          Alisha Boe         Alison
Brie
##                   1                   1                   1
1
##        Alison Doody         Alissa Dean        Allen Covert       Alyson
Stoner
##                   1                   1                   1
1
##         Amanda Wyss     Amber Armstrong     America Ferrera           Amy
Steel
##                   2                   1                   1
2
##       Andrea Martin   Andrew Bryniarski        Andy Richter     Angelo
Rossitto
##                   1                   1                   3
2
##       Angus Barnett     Angus Macfadyen         Anita Briem       Anna
Kendrick
##                   1                   2                   1
3
##         Anne Archer       Anne Hathaway          Anne Heche       Annie
Parisse
##                   1                   3                   1
1
##     Anthony Daniels     Anthony Hopkins    Anthony LaPaglia    Anthony
Reynolds
##                   1                   2                   1
```

```
1
## Armin Mueller-Stahl         Ashley Bell
##                   1                   1

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

##
## Black and White           Color
##              25            1071

table(franmovie.regrset$language)

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

table(franmovie.regrset$country)

##
##     Australia       Belgium        Brazil        Canada         China
##            19             1             1            20             7
##       Denmark        France       Germany     Hong Kong         India
##             2            22            19             1             3
##     Indonesia        Israel         Japan   New Zealand Official site
##             1             2             4             6             1
##        Russia   South Korea         Spain        Taiwan      Thailand
##             2             1             2             1             1
##            UK           USA
##            60           920

table(franmovie.regrset$content_rating)

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

table(franmovie.regrset$movie.year)

##
## 1920 1939 1940 1960 1964 1968 1969 1972 1973 1974 1975 1976 1977 1978 1979
1980
```

```
##    1    1    1    1    1    1    1    1    3    4    2    1    1    7    2
6
## 1981 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992 1993 1994 1995
1996
##   10   12    5   13    9    8   17   14   16   12   16   10   11   19   17
14
## 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011
2012
##   16   24   32   41   45   40   52   51   46   59   36   49   48   49   47
52
## 2013 2014 2015 2016
##   53   55   35   29
```

```
# 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)]

# 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
```

```
# 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]
```

```r
#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)
summary(fran.model01)
```

```
##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn01.reg)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -542733447   -50807213   -16108086    27813533  1453902520
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 1.250e+07  5.121e+07   0.244  0.80724
## num_critic_for_reviews      1.911e+04  6.712e+04   0.285  0.77592
## duration                    9.293e+04  2.421e+05   0.384  0.70126
## director_facebook_likes     1.983e+03  1.754e+03   1.131  0.25869
## actor_3_facebook_likes     -6.589e+03  6.214e+03  -1.060  0.28946
## actor_1_facebook_likes     -5.784e+03  3.601e+03  -1.606  0.10871
## num_voted_users             3.848e+02  5.019e+01   7.667 6.90e-14 ***
## cast_total_facebook_likes   5.563e+03  3.612e+03   1.540  0.12412
## facenumber_in_poster       -6.109e+04  2.466e+06  -0.025  0.98025
## num_user_for_reviews       -2.317e+04  1.543e+04  -1.502  0.13370
## budget                      6.605e-01  1.115e-01   5.926 5.17e-09 ***
## actor_2_facebook_likes     -5.504e+03  3.759e+03  -1.464  0.14364
## imdb_score                  1.462e+07  5.375e+06   2.719  0.00672 **
## aspect_ratio               -3.639e+07  1.969e+07  -1.848  0.06504 .
## movie_facebook_likes       -7.828e+02  2.852e+02  -2.745  0.00623 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.383873e+16)
##
##     Null deviance: 1.3796e+19  on 630  degrees of freedom
## Residual deviance: 8.5247e+18  on 616  degrees of freedom
## AIC: 25259
```

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

names(fran.trn01.reg)

##  [1] "num_critic_for_reviews"    "duration"
##  [3] "director_facebook_likes"   "actor_3_facebook_likes"
##  [5] "actor_1_facebook_likes"    "num_voted_users"
##  [7] "cast_total_facebook_likes" "facenumber_in_poster"
##  [9] "num_user_for_reviews"      "budget"
## [11] "actor_2_facebook_likes"    "imdb_score"
## [13] "aspect_ratio"              "movie_facebook_likes"
## [15] "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)])
summary(fran.model01mo)

##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn01.reg[,
##     -c(2, 3, 8, 9, 12, 13, 14)])
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -660855313   -46796292   -19689599    24388852  1442825650
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 4.447e+07  8.623e+06   5.157 3.38e-07 ***
## num_critic_for_reviews     -9.848e+04  4.932e+04  -1.997   0.0463 *
## actor_3_facebook_likes     -5.837e+03  6.262e+03  -0.932   0.3516
## actor_1_facebook_likes     -4.754e+03  3.634e+03  -1.308   0.1913
## num_voted_users             3.922e+02  3.120e+01  12.568  < 2e-16 ***
## cast_total_facebook_likes   4.563e+03  3.647e+03   1.251   0.2113
## budget                      6.434e-01  1.073e-01   5.999 3.37e-09 ***
## actor_2_facebook_likes     -4.645e+03  3.795e+03  -1.224   0.2215
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.421619e+16)
##
##     Null deviance: 1.3796e+19  on 630  degrees of freedom
## Residual deviance: 8.8567e+18  on 623  degrees of freedom
## AIC: 25270
##
## Number of Fisher Scoring iterations: 2
```

```r
#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)
summary(fran.model02)

##
## Call:
## glm(formula = adjust.gross ~ ., family = gaussian, data = fran.trn02.reg)
##
## Deviance Residuals:
##        Min          1Q      Median          3Q         Max
## -338246916   -36051508    -2570595    27227548   350793032
##
## Coefficients: (5 not defined because of singularities)
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.919e+08  1.194e+08   2.444 0.014845 *
## colorColor                 3.592e+07  2.112e+07   1.700 0.089614 .
## languageEnglish           -1.724e+07  8.840e+07  -0.195 0.845491
## languageFrench            -1.371e+08  1.219e+08  -1.124 0.261416
## languageIndonesian        -2.226e+07  1.100e+08  -0.202 0.839675
## languageJapanese          -9.551e+07  1.407e+08  -0.679 0.497520
## languageMandarin           5.536e+07  1.168e+08   0.474 0.635811
## languagePortuguese        -1.209e+08  1.165e+08  -1.038 0.299901
## languageSpanish           -1.115e+08  1.190e+08  -0.937 0.349235
## languageThai              -1.585e+08  1.145e+08  -1.383 0.167103
## countryBrazil                     NA         NA      NA       NA
## countryCanada              1.581e+07  3.148e+07   0.502 0.615836
## countryChina              -1.017e+08  1.064e+08  -0.956 0.339625
## countryDenmark                    NA         NA      NA       NA
## countryFrance              1.915e+07  3.595e+07   0.533 0.594521
## countryGermany             1.543e+07  3.441e+07   0.448 0.654114
## countryIndonesia           3.177e+07  1.187e+08   0.268 0.789072
## countryJapan               2.462e+07  8.151e+07   0.302 0.762693
## countryNew Zealand         9.827e+06  4.272e+07   0.230 0.818172
## countryTaiwan                     NA         NA      NA       NA
## countryThailand                   NA         NA      NA       NA
## countryUK                  1.396e+07  2.749e+07   0.508 0.611921
## countryUSA                 2.951e+07  2.374e+07   1.243 0.214338
## content_ratingG           -2.024e+08  8.240e+07  -2.457 0.014326 *
## content_ratingNC-17       -3.082e+08  8.794e+07  -3.505 0.000494 ***
## content_ratingPG          -1.976e+08  8.075e+07  -2.447 0.014707 *
## content_ratingPG-13       -2.144e+08  8.066e+07  -2.658 0.008090 **
## content_ratingR           -2.595e+08  8.028e+07  -3.233 0.001298 **
## content_ratingUnrated     -2.533e+08  9.148e+07  -2.769 0.005818 **
## movie.year1968             1.796e+08  8.129e+07   2.209 0.027584 *
```

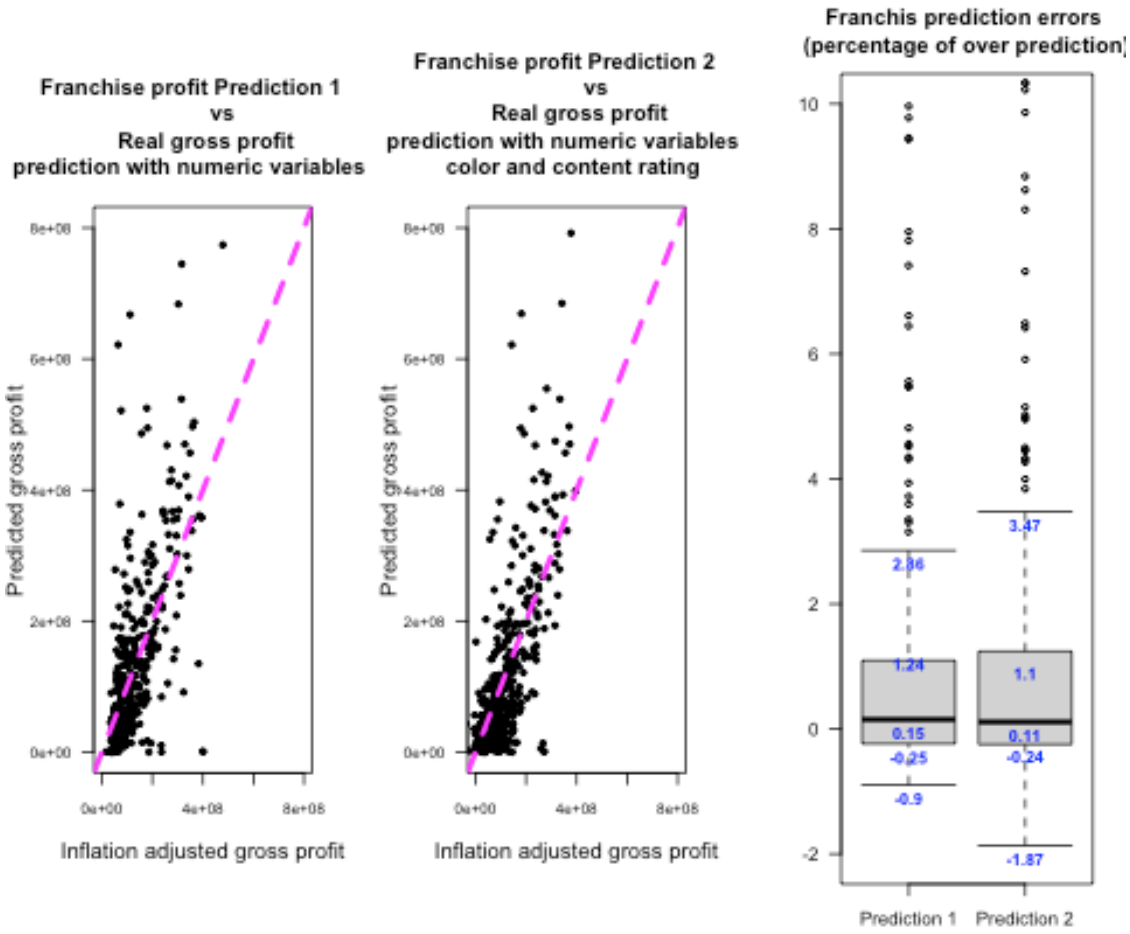
```
## movie.year1972              4.235e+08  8.208e+07   5.159 3.48e-07 ***
## movie.year1973              6.866e+08  7.781e+07   8.824  < 2e-16 ***
## movie.year1975              9.376e+08  7.921e+07  11.838  < 2e-16 ***
## movie.year1976              2.570e+08  7.933e+07   3.240 0.001269 **
## movie.year1978              2.587e+08  3.991e+07   6.482 2.03e-10 ***
## movie.year1979              7.625e+07  5.730e+07   1.331 0.183799
## movie.year1980              2.059e+08  4.922e+07   4.185 3.33e-05 ***
## movie.year1981              1.344e+07  3.728e+07   0.361 0.718514
## movie.year1982             -1.859e+07  3.588e+07  -0.518 0.604529
## movie.year1983              1.433e+08  4.410e+07   3.249 0.001230 **
## movie.year1984              7.692e+07  3.666e+07   2.098 0.036362 *
## movie.year1985             -3.558e+07  4.026e+07  -0.884 0.377112
## movie.year1986              8.411e+06  4.270e+07   0.197 0.843910
## movie.year1987             -1.238e+07  4.583e+07  -0.270 0.787148
## movie.year1988             -3.281e+07  3.288e+07  -0.998 0.318702
## movie.year1989             -3.946e+07  3.360e+07  -1.174 0.240787
## movie.year1990             -2.905e+07  4.060e+07  -0.716 0.474536
## movie.year1991             -4.650e+07  3.267e+07  -1.423 0.155248
## movie.year1992              1.467e+07  3.803e+07   0.386 0.699782
## movie.year1993              4.833e+07  4.130e+07   1.170 0.242359
## movie.year1994             -6.884e+07  3.086e+07  -2.231 0.026109 *
## movie.year1995             -2.489e+07  3.309e+07  -0.752 0.452283
## movie.year1996             -4.752e+07  3.270e+07  -1.453 0.146722
## movie.year1997             -4.312e+07  2.970e+07  -1.452 0.147133
## movie.year1998             -4.481e+07  3.059e+07  -1.465 0.143495
## movie.year1999             -2.236e+07  2.901e+07  -0.771 0.441148
## movie.year2000             -5.459e+07  2.671e+07  -2.044 0.041472 *
## movie.year2001             -3.068e+07  2.700e+07  -1.137 0.256177
## movie.year2002             -4.767e+07  2.737e+07  -1.742 0.082133 .
## movie.year2003             -4.580e+07  2.673e+07  -1.713 0.087280 .
## movie.year2004             -9.222e+07  2.666e+07  -3.459 0.000585 ***
## movie.year2005             -7.614e+07  2.651e+07  -2.872 0.004242 **
## movie.year2006             -6.514e+07  2.644e+07  -2.464 0.014049 *
## movie.year2007             -7.574e+07  2.874e+07  -2.635 0.008658 **
## movie.year2008             -1.010e+08  2.663e+07  -3.794 0.000165 ***
## movie.year2009             -8.007e+07  2.644e+07  -3.028 0.002580 **
## movie.year2010             -7.550e+07  2.688e+07  -2.809 0.005152 **
## movie.year2011             -9.542e+07  2.589e+07  -3.685 0.000251 ***
## movie.year2012             -5.853e+07  2.634e+07  -2.223 0.026656 *
## movie.year2013             -6.560e+07  2.463e+07  -2.664 0.007952 **
## movie.year2014             -6.393e+07  2.557e+07  -2.501 0.012687 *
## movie.year2015             -4.401e+07  2.598e+07  -1.694 0.090920 .
## movie.year2016                     NA         NA      NA       NA
## num_critic_for_reviews      1.250e+05  5.703e+04   2.191 0.028849 *
## duration                   -3.212e+05  1.980e+05  -1.622 0.105398
## director_facebook_likes    -1.691e+03  1.128e+03  -1.499 0.134355
## actor_3_facebook_likes     -5.643e+03  4.398e+03  -1.283 0.200007
## actor_1_facebook_likes     -6.644e+03  2.537e+03  -2.619 0.009071 **
## num_voted_users             2.672e+02  3.362e+01   7.949 1.08e-14 ***
## cast_total_facebook_likes   6.581e+03  2.550e+03   2.581 0.010119 *
```

```
## facenumber_in_poster      -5.949e+05  1.733e+06  -0.343 0.731502
## num_user_for_reviews       2.158e+03  1.032e+04   0.209 0.834523
## budget                     7.232e-01  8.279e-02   8.735  < 2e-16 ***
## actor_2_facebook_likes    -6.026e+03  2.646e+03  -2.277 0.023163 *
## imdb_score                 4.166e+06  3.955e+06   1.054 0.292557
## aspect_ratio              -1.563e+07  1.338e+07  -1.168 0.243177
## movie_facebook_likes      -9.616e+02  2.225e+02  -4.322 1.84e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 5.357023e+15)
##
##     Null deviance: 1.1236e+19  on 628  degrees of freedom
## Residual deviance: 2.9303e+18  on 547  degrees of freedom
## AIC: 24644
##
## Number of Fisher Scoring iterations: 2

# 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(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(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))
```



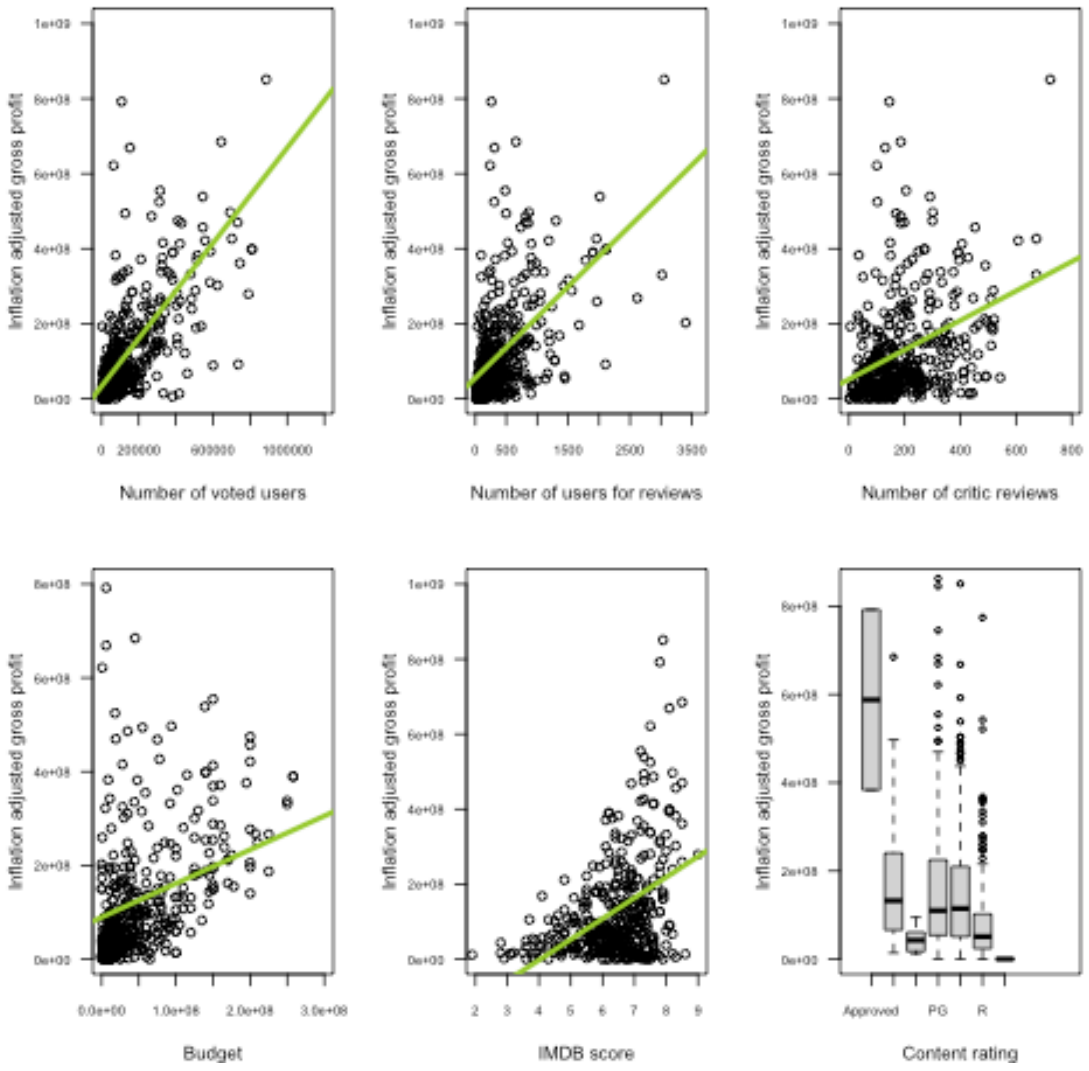

```
colnames(bop02$stats) = c("Franchise prediction 1", "Franchise prediction 2")
bop02$stats  # show the errors box plot statistics for both predictions
```

```
##      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,
```

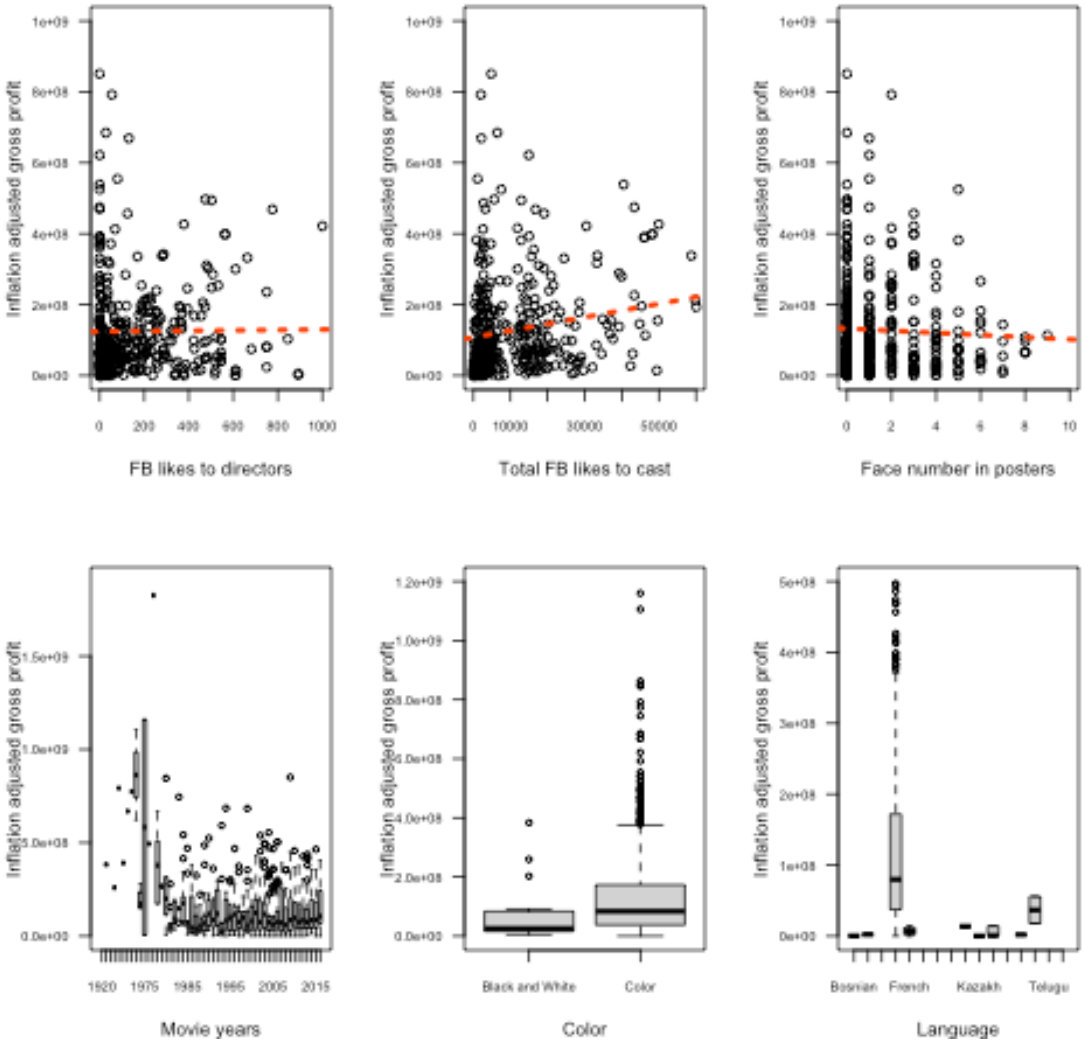
```
     xlim = c(0,1200000), ylim = c(0,1e+09),
     las = 1, cex.axis = 0.7, cex.lab = 1,
     ylab = "Inflation adjusted gross profit",
     xlab = "Number of voted users")
abline(lm(fran.adjgross.tstreg02 ~ fran.tst02.reg$num_voted_users),
       col = "yellow green", lwd = 3)

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)
```