W4201 Final Report

Group Members:
Cheng Lian (cl3279)
Yan Liu (yl3174)
Junting Ma (jm4130)
Xiaolun Peng (xp2138)
Minyang Su (sm3894)
Wanru Shao (ws2427)
Wanda Wu (ww2371)
Yong Wang (yw2647)
Yang Yan (yy2554)
Tianxing Zhou (tz2234)

Contents

I. Introduction

Forecasting movie box office results has always been a challenging problem that intrigues many scholars and industry leaders. The unpredictability of box office results came from the ever-changing taste of consumers, market conditions and marketing strategies.

For consumers, information about a movie beforehand plays a pivotal role in decision-making process of whether to see this particular movie or not. Although trailers and marketing campaigns are informative and easily accessible, consumers may require something more objective to tell them which movies will be worth their money. This is where the movie critics come in.

For the movie industry, if there is a clear link between critics' reviews of a movie prior to its release and its final box office result, this could help with the strategy adjustment and distribution decision of major film studios and theaters. For example, if professional reviewers are amazed by a movie's test screening and write positive reviews on major magazines and websites, theaters can decide to prolong this movie's schedule and play it in more theaters in hope of bringing in more revenue.

In this paper, we focused on applying text regression techniques to predict final movie box office results based on critics' reviews. We also conducted sentimental words analysis and critical words analysis to identify particular words and phrases that may have potential impacts on the box office results.

II. Objective

- Apply Lasso and Bayesian Lasso to estimate movie box office results based on critics' reviews prior to movie release
- Identify particular words and phrases that may have potential impacts on the box office results for marketing purpose

III. Material & Methods

- Data Source

Our sample consists of 2,000 reviews for 400 movies (top 100 of year 2011, 2012, 2013 and 2014 ranked on Rotten Tomato) and our goal is to analyze the impacts of reviews and other non-text variables on the box office results. As there is no handy csv or txt format data available and it is too cumbersome to manually copy and paste the data online, we employed web crawler to obtain text and non-text variables automatically.

Web Crawler Steps	
Step 1	Get a list of movies from Rotten Tomato and save it to an array
Step 2	Figure out the URL structure of searching engines like Google that could lead us to the searching page of movies automatically
Step 3	Apply regular expression by package *re* to find the URL of the desired movies from searching result page
Step 4	Download the page behind the URL we found using package *httplib2* and obtain the structure of the page using package *BeautifulSoup*.
Step 5	Find the patterns of text and non-text variables on Metacritic, thereby getting and saving these data in csv and txt files
Data Collected	
Non-text Variables	- Numerical: Metascore, userscore, runtime, box office result - Categorical: film rating, release time, genre, country
Text Variables	- Critics' reviews before release date

After collecting the desired data, we observed two major problems with our data set. Firstly, data like Metascore, userscore and country were occasionally missed for some movies. Since the number of movies with missing information was quite small, we chose to delete them. Secondly, there were some garbled characters in the txt files due to the limitation that foreign words (e.g. French and Spanish) cannot be recognized correctly. We wanted to remove all of these garbled characters, as they could cause problems in the construction of document term matrix in R. When we loaded these problematic files into Python, we analyzed the pattern and observed that most of the garbled characters were shown as '\xa1\xaa'. Having known this pattern, we then simply used package *re* to remove all garbled characters

- Analysis Methods (Dimension Reduction, Primary Analysis)

  Dimension Reduction

  The problem arising naturally here is that we still have too many words (subsequently too many predictors) in the reviews even after automatic word screening. Among so many words in the reviews, which of them is making a difference? How can we decide that a certain word, say "excellent", did encourage audience attendance at a theatre or not? In our project, three steps of dimension reduction were conducted one by one.

  First of all, word frequency screening. We believe that a word should appear at least a certain number of times in all the reviews in order to be considered into our dictionary. After setting the critical value of frequency to be 30 and 10 for critic and sentimental words respectively, 2931 critics and 1637 sentimental words survived.

  Secondly, it is also reasonable to delete those words that appear in reviews of only one or two movies. This step is called movie frequency screening. After applying this rule of thumb, 2874 critics and 1637 sentimental words are left.

  And these two steps above also provided solid basic assumptions for our third step: the two-sample t test.

  According to whether a certain word appears in the reviews of a film, we divided the movies into two groups. Then we conducted two-sample t test to compare the mean of the box office of the two groups.

  Let $X_1, X_2, X_3, ..., X_n$ and $Y_1, Y_2, Y_3, ..., Y_m$ be independent random samples from normal distributions with respective means μ_1 and μ_2, and unknown but equal variance σ^2.

$$H_0 : \mu_1 = \mu_2 \text{ vs } H_1 : \mu_1 \neq \mu_2.$$

$$T = \frac{(\bar{X} - \bar{Y})}{\sqrt{S_p^2(\frac{1}{n} + \frac{1}{m})}}, \text{ where } S_p^2 = \frac{(n-1)S_X^2 + (m-1)S_Y^2}{n+m-2}$$

  Under the null hypothesis, t has t distribution with n+m-2 degrees of freedom.

  If the null hypothesis, which suggests equal mean, is not rejected, then we can conclude that this word does not have a significant impact on the box office. So we would throw the word out of our predictors. Otherwise, it should be included.

  With alpha level 0.1% and 1% for critics and sentimental words respectively, we finally have a comparatively satisfying result with 187 critics and 212 sentimental words left.

Primary Analysis: Bayesian Lasso

The Lasso of Tibshirani (1996) is a shrinkage and selection method for linear regression. It is most commonly applied to the model: $\boldsymbol{y} = \mu\mathbf{1}_n + \boldsymbol{X\beta} + \boldsymbol{\epsilon}$, (1)

where $\boldsymbol{y}$ is the $n\times1$ vector of responses, μ is the overall mean, $\boldsymbol{X}$ is the $n\times p$ matrix of standardized predictors, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is the vector of regression coefficients to be estimated, and $\boldsymbol{\epsilon}$ is the $n\times1$ vector of independent and identically distributed normal errors with mean 0 and unknown variance σ^2. The Lasso estimates can be viewed as L_1 penalized least squares estimates. They achieve: $\min_{\boldsymbol{\beta}}(\tilde{\boldsymbol{y}} - \boldsymbol{X\beta})^T(\tilde{\boldsymbol{y}} - \boldsymbol{X\beta}) + \lambda\sum_{j=1}^{p}|\beta_j|$ (2)
for some $\lambda \geq 0$, where $\tilde{\boldsymbol{y}} = \boldsymbol{y} - \bar{y}\mathbf{1}_n$.

Noting the form of the penalty term in (2), Lasso estimates can be interpreted as a Bayesian posterior mode estimate when the parameters β_j have independent and identical double exponential priors. Indeed, with the prior

$$\pi(\beta) = \prod_{j=1}^{p}\frac{\lambda}{2}e^{-\lambda|\beta_j|}, \quad (3)$$

and an independent prior $\pi(\sigma^2)$ on $\sigma^2 > 0$. By Bayes' theorem, the posterior distribution, conditional on $\tilde{\boldsymbol{y}}$, can be expressed as

$$p(\boldsymbol{\beta}|\tilde{\boldsymbol{y}}, \sigma^2, \lambda) \propto \left[\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\left(-\frac{(\tilde{\boldsymbol{y}}-\boldsymbol{X\beta})^T(\tilde{\boldsymbol{y}}-\boldsymbol{X\beta})}{2\sigma^2}\right)}\right]\times\prod_{j=1}^{p}\frac{\lambda}{2}e^{-\lambda|\beta_j|}\times\pi(\sigma^2)$$

$$p(\boldsymbol{\beta}|\tilde{\boldsymbol{y}}, \sigma^2, \lambda) \propto \pi(\sigma^2)(\sigma^2)^{-(n-1)/2}e^{\left(-\frac{(\tilde{\boldsymbol{y}}-\boldsymbol{X\beta})^T(\tilde{\boldsymbol{y}}-\boldsymbol{X\beta})}{2\sigma^2} - \lambda\sum_{j=1}^{p}|\beta_j|\right)}$$

For any fixed value of $\sigma^2 > 0$, the maximizing $\boldsymbol{\beta}$ is a Lasso estimate, and hence the posterior mode estimate, if it exists, will be a Lasso estimate.

In Bayesian Lasso, we shall use conditional priors on $\boldsymbol{\beta}$ of the form

$$\pi(\beta|\sigma^2) = \prod_{j=1}^{p}\frac{\lambda}{2\sqrt{\sigma^2}}e^{-\lambda|\beta_j|/\sqrt{\sigma^2}}, \quad (4)$$

instead of prior (3) and the scale invariant prior $\pi(\sigma^2) = 1/\sigma^2$ on σ^2.Then the joint distribution of $\tilde{\boldsymbol{y}}$, $\boldsymbol{\beta}$, c and λ is

$$f(\tilde{\boldsymbol{y}}, \boldsymbol{\beta}, \sigma^2, \lambda) = \left[\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{\left(-\frac{(\tilde{\boldsymbol{y}}-\boldsymbol{X\beta})^T(\tilde{\boldsymbol{y}}-\boldsymbol{X\beta})}{2\sigma^2}\right)}\right]\times\prod_{j=1}^{p}\frac{\lambda}{2\sqrt{\sigma^2}}e^{-\lambda|\beta_j|/\sqrt{\sigma^2}}\times\pi(\sigma^2) \quad (5)$$

To get the Bayesian Lasso estimate, we use Gibbs sampler in two variable case. We consider $\boldsymbol{\beta}$ and σ^2 are a pair of random variables. For fixed λ, the joint distribution of $\boldsymbol{\beta}$ and σ^2 is given by (5). The initial value $\sigma^2 = \sigma^{2(0)}$ is specified(may be chosen from a uniform distribution). Then we can generate the subsequent $\boldsymbol{\beta}$ and σ^2 values using:

$$\boldsymbol{\beta}^{(j)} \sim f\left(\boldsymbol{\beta} \middle| \sigma^2 = \sigma^{2(j)}\right)$$
$$\sigma^{2(j)} \sim f\left(\sigma^2 \middle| \boldsymbol{\beta} = \boldsymbol{\beta}^{(j-1)}\right)$$

Under reasonably general conditions, the distribution of $\boldsymbol{\beta}^{(k)}$ converges to $f(\boldsymbol{\beta})$ as $k \to \infty$.
Specifically, the steps of Bayesian Lasso are:

(a) Let $k = 0$ and choose initial $\lambda^{(0)}$.
(b) Generate a sample from the posterior distribution of β, σ^2 using the Gibbs sampler with λ set to $\lambda^{(k)}$.
(c) Approximate the expected "completed-data" log likelihood for λ by substituting averages based on the Gibbs sample of the previous step for any terms involving expected values of β, σ^2
(d) Let $\lambda^{(k+1)}$ be the value of λ that maximizes the expected log likelihood of the previous step
(e) Return to the second step, and iterate until desired level of convergence.

Sufficient Conditions for Convergence in Gibbs Sampler:

It was already mentioned that the sequence generated by Gibbs Sampler converges to the target density as the number of iterations become large. To give a formal statement, let $K(x, x')$ be the transition density of the Gibbs Algorithm and $K^{(M)}(x_0, x')$ be the density of the draw x' after M iterations, given that the starting value was x_0. Then, the following property

$\left\|K^{(M)}(x_0, x') - f_X(x')\right\| \to 0$, as $M \to \infty$ is satisfied under following conditions:

1. $f_X(x) > 0$ implies that there exists an open neighborhood N_x containing x and $\epsilon > 0$ such that for all $x' \in N_x$, $f_X(x') > \epsilon > 0$;
2. $\int f(x) dx_k$ is locally bounded for all k, where x_k is the k^{th} parameter in the sequence;
3. The support of x is arc connected.

IV. Results

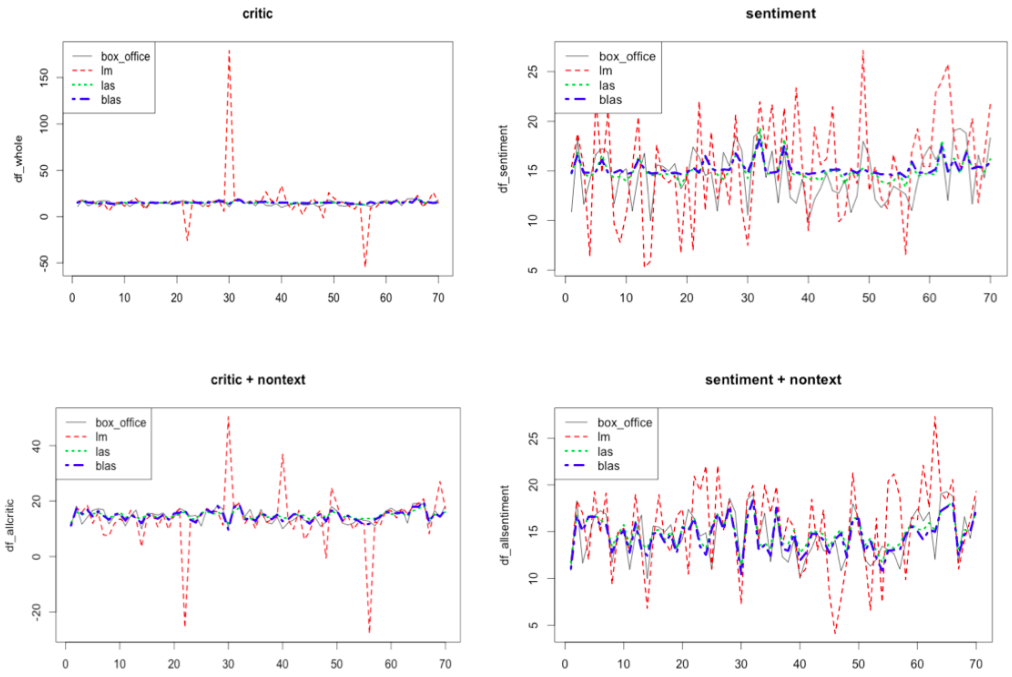
We applied our models separately on four groups of covariate (sentimental words, critical words, sentimental and non-text words, critical words and non-text words) to predict the log box office result. Lasso and Bayesian Lasso picked the ten most influential words. Intuitively we would expect words like "advocate" and "entertain" to have positive effects on box office results and expect words like "foolish" and "mundane" to have negative ones. And indeed, the coefficient of "advocate" is 0.9213, indicating that the log box-office result will increase by 0.9123 unit if the review contains the word "advocate". Similarly, the word "foolish" with a coefficient of -0.634 means the log box-office result will decrease by 0.634 unit due to the occurrence of the word "foolish".

Sentimental				
Advocate 0.9123025	Harass -0.8686897	Foolish -0.6342833	Knife -0.5336194	Unwilling -0.5112143
Prolife -0.5033499	Entertain 0.4582444	Mundane -0.3973151	Horrify -0.3971504	Woe -0.3609907
Critical				
Franchise 0.376119896	India -0.372164724	Fun 0.343699573	Tape -0.334158036	Shown -0.276224758
Documentary -0.178296814	Countryside -0.093153350	Sundance -0.065916003	Doc -0.036117086	Canny -0.002039419
Sentimental and Non-text				
PG-13 1.4900877	USA 1.4258562	PG 1.3317707	Documentary -1.2438425	Advocate 0.9073130
Action 0.7206865	Harass -0.5276461	Profile -0.4917763	R 0.4075140	User Score 0.3855165
Critical and Non-text				
Documentary -1.48362765	USA 1.45278521	PG-13 0.89399460	Action 0.62522747	PG 0.52551407
India -0.37009372	User Score 0.24776005	Fun 0.11079206	Franchise 0.09421666	Shown -0.06729551

To validate our models and avoid over-fitting, we divided the box office result data into a training set (first 300 movies chronologically) and a testing set (latter 70 movies). From the MSE table below, we can see that MSE of Lasso and Bayesian Lasso appear to be much smaller than MSE of simple linear regression.

Test MSE		Models		
		lm	Lasso	Bayesian Lasso
Predictors	Critical	526.833833	5.267974	5.592131
	Sentimental	24.851895	5.060926	5.615744
	Non-text & critical	94.961584	3.793435	2.887212
	Non-text & sentimental	16.654420	2.937351	2.778231

Additionally, we also constructed graphical displays of comparison between real box office results and the ones predicted by our models. The closer the line fits the grey line (real result), the better the model predicted. As expected, the red line (simple linear regression) differs significantly from the grey line, whereas Lasso and Bayesian Lasso fit closely.



V. Discussion & Conclusions

Based on the results we obtained, following conclusions can be drawn:

- Text data does have substantial predicting power as text words survived the two-sample t-test with a significance level of 0.01 and 0.001.
- Comparison of test MSEs of simple linear regression, Lasso and Bayesian Lasso proves the power of *Bayesian Lasso* in variable selection.
- Sentimental words have *lower MSE* than critical words among all three methods. We would expect sentimental words to have better predicting power as review readers tend to focus more on these sentimental words that express the reviewer's subjective feeling towards a movie.
- The fact that seven predictors of the ten most important chosen variables are *non-text* reminds us not to overstate the significance of text data.

VI. References

Casella, George, and Edward George. "Explaining the Gibbs Sampler." The American Statistician 46.3 (1992): 67-174.

Joshi, Mahesh, et al. "Movie reviews and revenues: An experiment in text regression." Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. Association for Computational Linguistics (2010).

Mandhare, Aditya. "Application of Decision Tree to Predict Gross Income of a Movie." International Journal of Computer Applications 106.5 (2014).

Park, Trevor, and George Casella. "The bayesian lasso." Journal of the American Statistical Association 103.482 (2008): 681-686.

Ramsey, Fred L., and Daniel W. Schafer. The Statistical Sleuth: A Course in Methods of Data Analysis. Boston: Brooks/Cole, Cengage Learning. 2013. 211-212.

Tibshirani, Robert. "Regression Shrinkage and Selection Via the Lasso." Journal of the Royal Statistical Society, Series B 58.1 (1996): 267-288.

VII. Appendix

- *EDA*

We conducted the Exploratory Data Analysis (EDA) to analyze the main characteristics of our data set. EDA is very useful for detecting outliers, testing underlying assumptions and analyzing the relationship between the explanatory variables and the dependent variables.

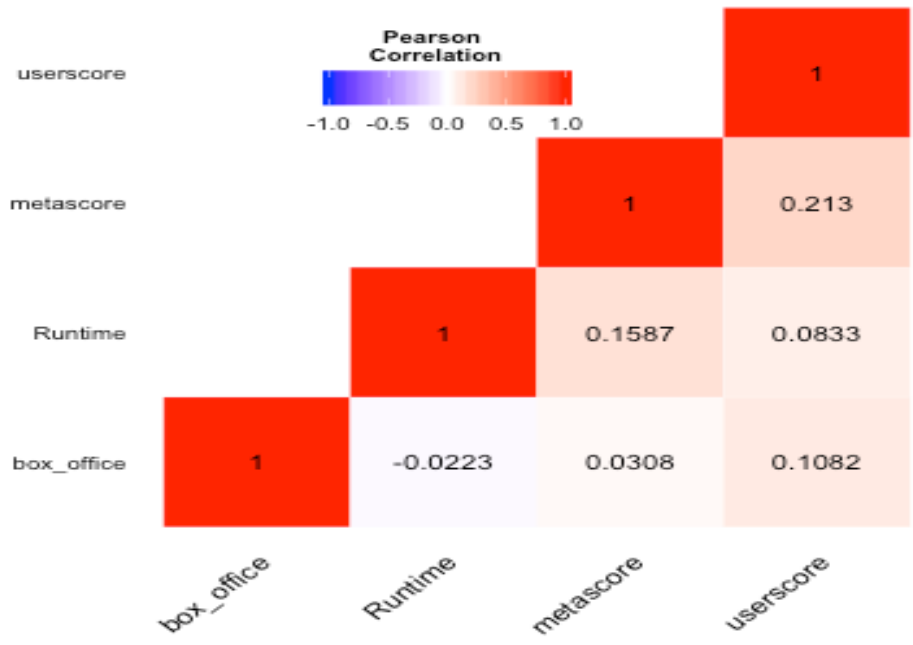
For non-text quantitative variables, a combination of quantitative and graphical techniques of EDA should be performed to reveal several interesting features of the data set.

Table 1.1 is a simple tabulation of the basic descriptive statistics for the numerical variables.

	Mean	Median	Variance	Sd	Range
Metascore	76.59	760	71.31	8.44	54
Userscore	7.50	7.60	0.64	0.80	5.90
Runtime	107.80	104	458.40	21.41	203
Log Result	14.78	14.38	6.92	2.63	12.37

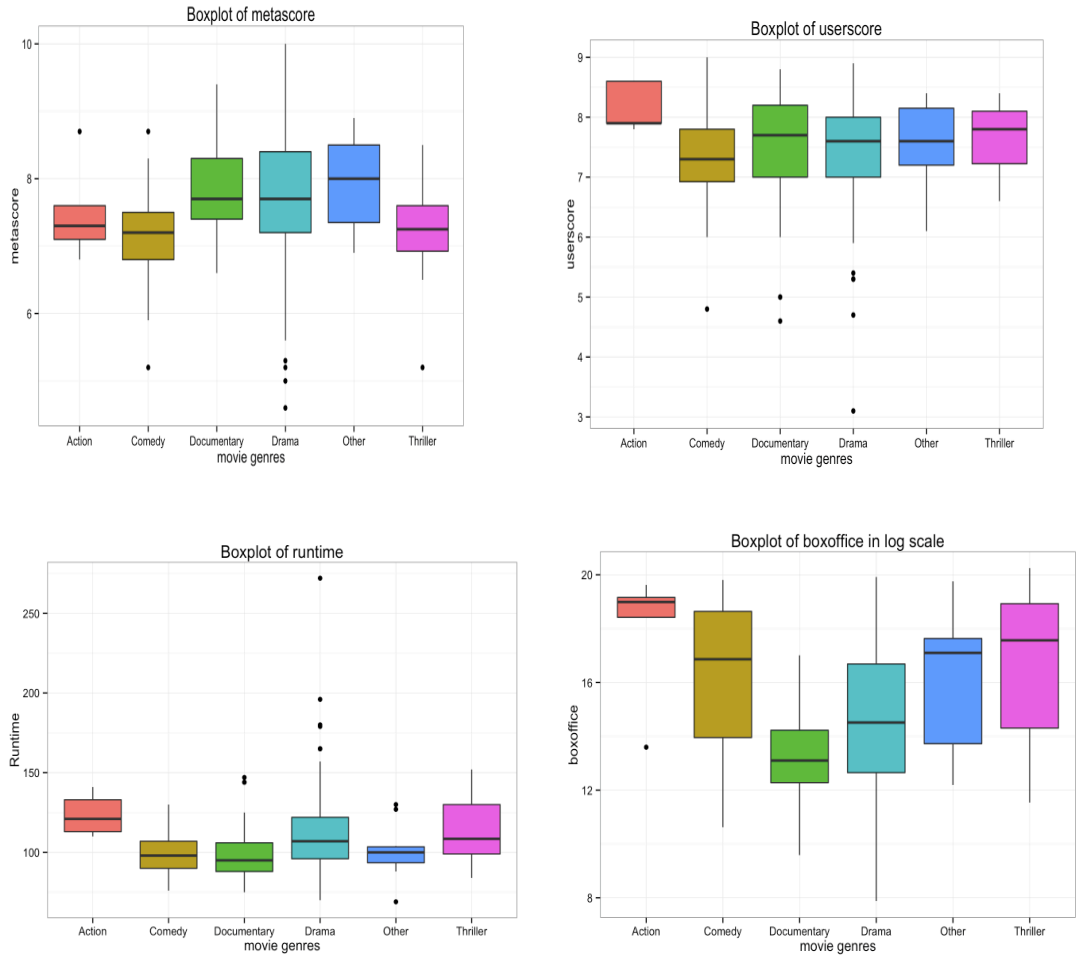
1.1 Table of Basic Descriptive Statistics

The correlation heat map is exhibited below to check for multicollinearity. According to the map, there is no multicollinearity and no strong association among the numerical variables.



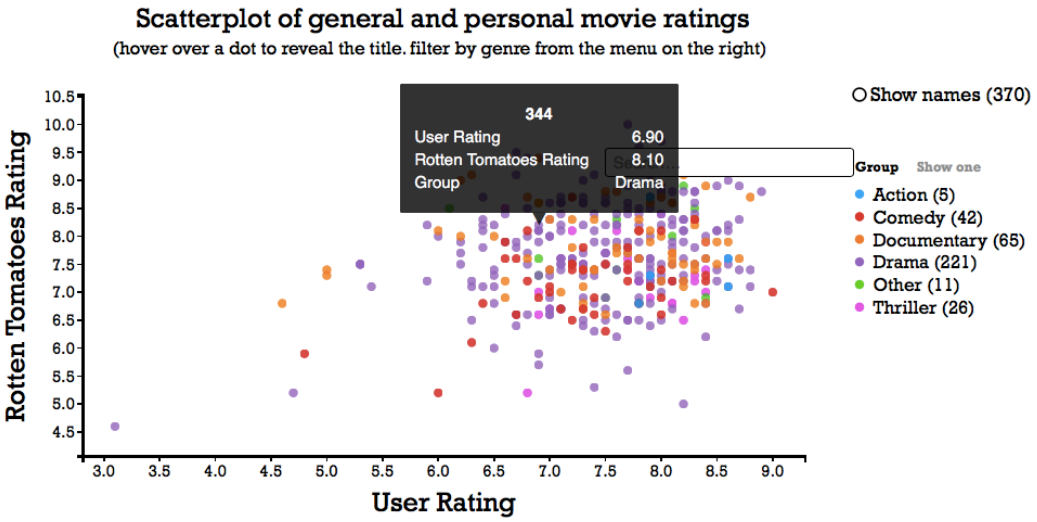
1.2 Correlation Heat Map

We created four individual boxplots for every numerical variable for outlier detection. Several outliers were detected and then removed for further analyses.



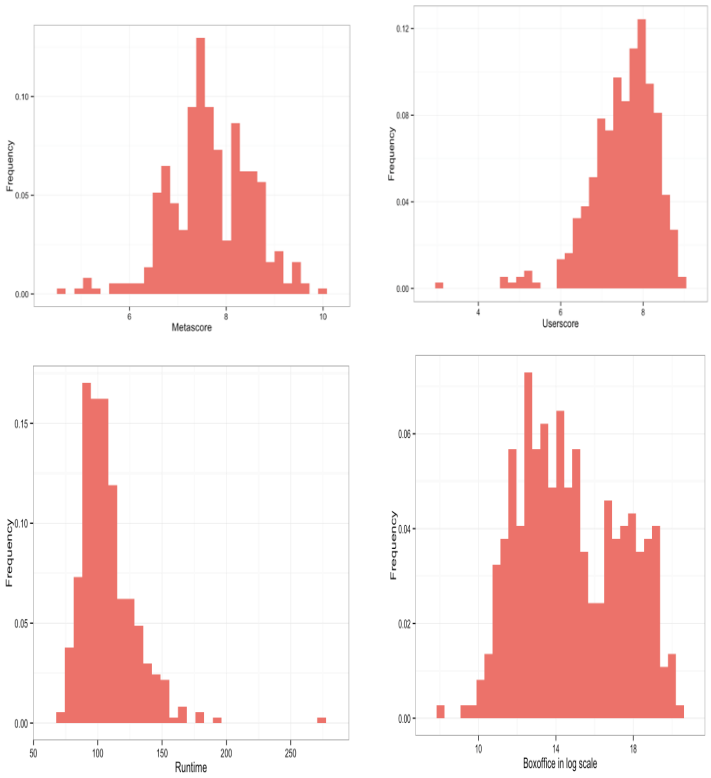
1.3 Boxplots of Metascore, Userscore, Runtime and Box-office Result in log-scale

An interactive scatter plot of Metascore and Userscore was also constructed that allows the users to interact actively with all the 370 observations by a simple hover over the dots.

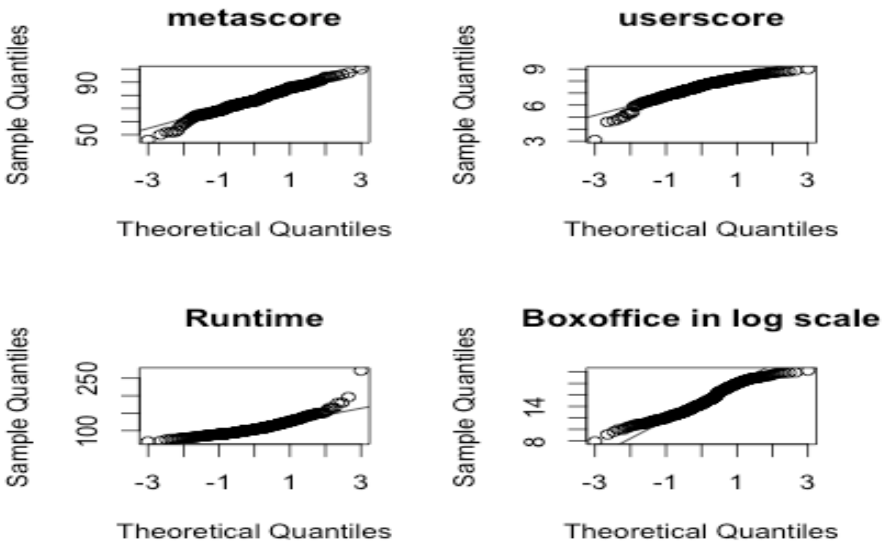


1.4 Scatter plot of Userscore and Metascore

Histogram and QQ plot below test whether our data set is normally distributed. Although normality does not seem to be obvious here, the symmetry of our data can compensate for the weak normality.

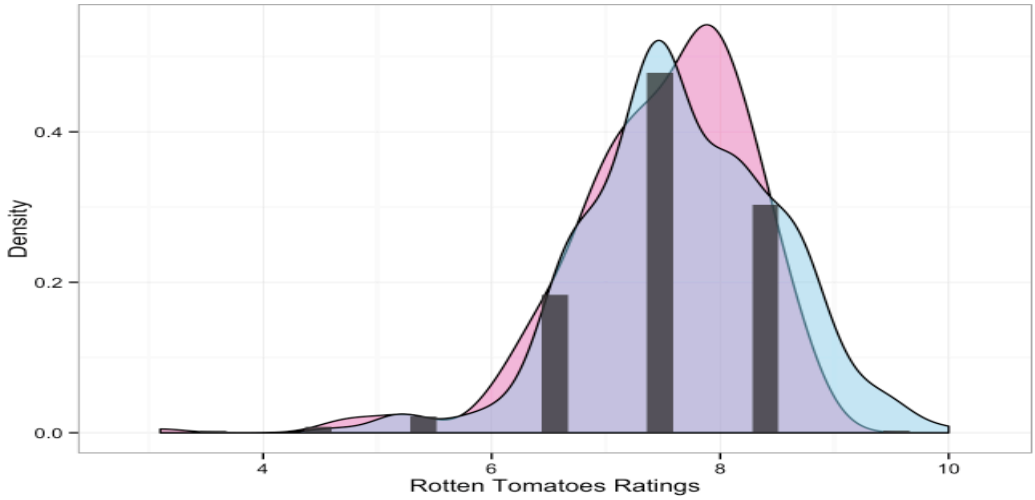


1.5 Histogram of Metascore, Userscore, Runtime and Boxoffice in log-scale



1.6 QQ plot of Metascore, Userscore, Runtime and Box-office Result in log-scale

This figure below plots the frequency distribution (black bars) and density (red area) of user ratings and the density of the Rotten Tomatoes scores (in blue) for the 370 observations in the data.



1.7 Frequency distribution and density of Metascore and Userscore

For non-text categorical variables, we were merely interested in the frequency of occurrence of each level and their association with the dependent variables. Following two tables show the frequency of each level for movie genres and quarters.

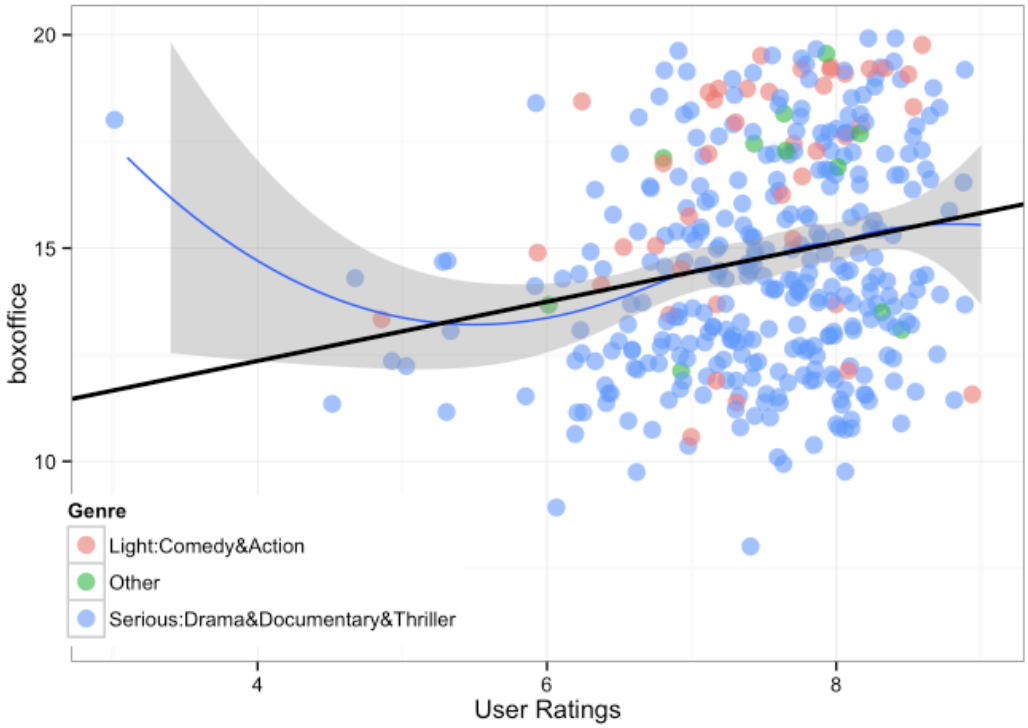
MPAA Rating	G	NC-17	Not Rated	PG	PG-13	R
Count	5	2	121	34	74	134

1.8 Table of frequency count of MPAA rating

Quarter	1	2	3	4
Count	76	96	91	107

1.9 Table of frequency count of Quarter

The scatter plot below demonstrates the relationship between the box-office result and user rating, categorized by different movie genres. The solid black line is the regression fit, the blue one shows a nonparametric loess smoothing that suggests some non-linearity in the relationship that we will explore later.



1.10 Scatter plot of Boxoffice and Userscore categorized by movie genres

For text variables, which are the critics' review before releasing data we scraped online, we use a Word Cloud to generate a visual representation of the word frequency for our data.

Word Cloud is extremely useful for us to recognize the most prominent single words as the importance of each tag is shown with font size or color. The Bigger font size and darker color the tags are, the more important the words are. The graph can help us to perceive directly which words may have the most predicting power for our dependent variable.

1.11 Sentimental Word Cloud

We can find the most important words are Sneakily, Unafford, Realize, Impoverish.

1.12 Critics Word Cloud

Again, judging by the font size and color of the font, we can find the most important words are Say, Emotion, Drive, Version.

To conclude, EDA approach provides us with a maximized insight of the data.

Regression assumptions:

There are four principal underlying assumptions that justify the use of the linear regression models. (Ramsey/Schafer 2013,211-212)

- ✓ **Linearity**: The plot of response means against the explanatory variable is a straight line.
- ✓ **Normality**: The subpopulations of responses at the different values of the explanatory variable all have normal distributions.
- ✓ **Homoscedasticity**: The spread of the responses around the straight line is the same at all levels of the explanatory variables.
- ✓ **Uncorrelated error**: The location of any response in relation to its mean cannot be predicted, either fully or partially, from knowledge of where other responses are in relation to their means. Furthermore, the location of any responses in relation to its mean cannot be predicted from knowledge of the explanatory variable values.

- *Related Researches & Improvements*

Related Researches:

1. Park, Trevor, and George Casella. "The bayesian lasso." Journal of the American Statistical Association 103.482 (2008): 681-686.

2. Mandhare, Aditya. "Application of Decision Tree to Predict Gross Income of a Movie." International Journal of Computer Applications 106.5 (2014).

3. Joshi, Mahesh, et al. "Movie reviews and revenues: An experiment in text regression." *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010.

Improvements: Bayesian Lasso and Sentiment Analysis

Bayesian Lasso: Although both Lasso and Bayesian Lasso are effective when dealing with high dimensional data, Lasso only shrinks the parameters down to zero, whereas the Bayesian Lasso provides interval estimates (Bayesian credible intervals) for variable selection.

Sentiment Analysis: Using sentimental words instead of all other single words for text mining analysis not only can give us some keywords for box office prediction, but also provides us with reviewers' attitude toward each movie.

- *Codes*

```
setwd("~/Dropbox/fall2015/4201-ada/ada-project")
ada<-load("~/Dropbox/fall2015/4201-ada/ada-
 project/ada.project.RData",verbose=FALSE)
require('wordcloud')
require('biclust')
require('cluster')
 require('igraph')
 require('dplyr')
 require('scales')
 require('SnowballC')
 require('RColorBrewer')
 require('ggplot2')
 require('tm')
 require('Rgraphviz')
 require('fpc')
 library('NLP')
 library("ngram")
 library("RWeka")

 source("http://bioconductor.org/biocLite.R")
 biocLite("Rgraphviz")

 ####2.Explorary data analysis
 # Data processing
 boxoffice<-log(y)
 X$metascore<-X$metascore/10
 X$userscore_level<-ifelse( X$userscore<1, 0.5,ifelse( X$userscore<2,
  1.5,ifelse( X$userscore<3, 2.5,ifelse( X$userscore<4, 3.5,ifelse( X$userscore<5,
  4.5,ifelse( X$userscore<6, 5.5,ifelse( X$userscore<7, 6.5,ifelse( X$userscore<8,
  7.5,ifelse( X$userscore<9, 8.5,ifelse( X$userscore<10, 9.5,NA)))))))))
 X$boxoffice<-boxoffice
 X$genre<-ifelse(X$Drama==1,'Drama',ifelse(X$Comedy==1,'Comedy',
  ifelse(X$Documentary==1,'Documentary', ifelse(X$Thriller==1,'Thriller',
  ifelse(X$Action==1,'Action','Other')))))

 for (i in 1:nrow(X)){
   if (X$Drama[i]==1|X$Documentary[i]==1|X$Thriller[i]==1)
     X$new.genre[i]<-'Serious:Drama&Documentary&Thriller'
   else
     if (X$Comedy[i]==1|X$Action[i]==1)
       X$new.genre[i]<-'Light:Comedy&Action'
     else
       X$new.genre[i]<-'Other'
 }

 # Animated scatterplot
 # Get the visualization library
 library(devtools)
```

```
install_github("clickme", "nachocab")

library(clickme)

# Big plot with title (genre)
c1<-clickme("points", out_height = 400, out_width = 800,
        x = X$userscore, y = X$metascore,
        title = "Scatterplot of general and personal movie ratings", subtitle = "(hover over
 a dot to reveal the title. filter by genre from the menu on the right)",
        formats = list(x = ".1f", y = ".0f"),
        opacity = .8, jitter = 0.4, radius = 5,
        height = 400, width = 900,
        formats = list(x = ".0f", y = ".0f"),
        color_groups = X$genre,
        names = X$Title,
        x_title = "User Rating", y_title = "Rotten Tomatoes Rating",
        file_name = "animatedscatter.html")
c1

p <- ggplot(X, aes(factor(genre), userscore))+
  geom_boxplot(aes(fill = factor(genre)))+
  theme_bw()+theme(legend.position="none")+
  labs(x='movie genres')+
  labs(title='Boxplot of userscore')
p

p <- ggplot(X, aes(factor(genre), metascore))+
  geom_boxplot(aes(fill = factor(genre)))+
  theme_bw()+theme(legend.position="none")+
  labs(x='movie genres')+
  labs(title='Boxplot of metascore')
p

p <- ggplot(X, aes(factor(genre), Runtime))+
  geom_boxplot(aes(fill = factor(genre)))+
  theme_bw()+theme(legend.position="none")+
  labs(x='movie genres')+
  labs(title='Boxplot of runtime')
p


p <- ggplot(X, aes(factor(genre), boxoffice))+
  geom_boxplot(aes(fill = factor(genre)))+
  theme_bw()+theme(legend.position="none")+
  labs(x='movie genres')+
  labs(title='Boxplot of boxoffice in log scale')
p

# normality
# 1. density
```

```
p<-ggplot(X, aes(x=userscore_level))+
  geom_density(alpha=0.5,aes(x=userscore, y = ..density..),fill='hotpink')+
  geom_density(alpha=0.5,aes(x=metascore, y = ..density..),fill='skyblue')+
  geom_histogram(alpha=0.8,aes(x=userscore_level,y=..count../sum(..count..)))+
  scale_x_continuous('Rotten Tomatoes Ratings')+
  scale_y_continuous('Density')+
  theme_bw()
p
# 2. histogram of metascore, userscore, runtime, and box office
p1<-ggplot(X,aes(x=userscore))+
  geom_histogram(aes(y=..count../sum(..count..),fill='red'))+
  scale_x_continuous('Userscore')+
  scale_y_continuous('Frequency')+
  theme_bw()+theme(legend.position="none")
p1

# 2. histogram of metascore, userscore, runtime, and box office
p1<-ggplot(X,aes(x=userscore))+
  geom_histogram(aes(y=..count../sum(..count..),fill='red'))+
  scale_x_continuous('Userscore')+
  scale_y_continuous('Frequency')+
  theme_bw()+theme(legend.position="none")
p1

p1<-ggplot(X,aes(x=metascore))+
  geom_histogram(aes(y=..count../sum(..count..),fill='red'))+
  scale_x_continuous('Metascore')+
  scale_y_continuous('Frequency')+
  theme_bw()+theme(legend.position="none")
p1

p1<-ggplot(X,aes(x=Runtime))+
  geom_histogram(aes(y=..count../sum(..count..),fill='red'))+
  scale_x_continuous('Runtime')+
  scale_y_continuous('Frequency')+
  theme_bw()+theme(legend.position="none")
p1

p1<-ggplot(X,aes(x=boxoffice))+
  geom_histogram(aes(y=..count../sum(..count..),fill='red'))+
  scale_x_continuous('Boxoffice in log scale')+
  scale_y_continuous('Frequency')+
  theme_bw()+theme(legend.position="none")
p1



# figure 2
summary(m1<-lm(boxoffice~userscore, data=X))
sqrt(mean(residuals(m1)^2)) #root mean squared error: 2.57
```

```
p <- ggplot(X, aes(userscore, boxoffice))+
  geom_point(position=position_jitter(width=0.1,height=.25),shape=16, size=4,alpha=0.6,
             aes(colour = new.genre, ))+
  stat_smooth(se = TRUE)+
  scale_x_continuous('User Ratings')+
  scale_y_continuous('Box Office')+
  theme_bw()+
  scale_colour_discrete(name="Genre")+
  scale_size_continuous(guide=FALSE)+
  theme(legend.position=c(0.2, 0.14))+
  geom_abline(size=1, aes(intercept=9.5768, slope=0.6945))+
  ylim(6,20)

# figure 3: Adding predictors
summary(m2<-lm(boxoffice~userscore+Documentary+Action,data=X))
sqrt(mean(residuals(m2)^2)) #root mean squared error: 2.25

box_office = log(y)
#find mean,median,var,sd,range,IQR
nu_variables = cbind(metascore,userscore,Runtime,box_office)
apply(nu_variables,2,mean)
apply(nu_variables,2,median)
apply(nu_variables,2,var)
apply(nu_variables,2,sd)
apply(nu_variables,2,range)[2,]-apply(nu_variables,2,range)[1,]
apply(nu_variables,2,IQR)

#find correlation
cor = cor(nu_variables)
cormat <- round(cor,4)
head(cormat)

library(reshape2)
# Get upper triangle of the correlation matrix
get_upper_tri <- function(cormat){
  cormat[lower.tri(cormat)]<- NA
  return(cormat)
}

# Heatmap
library(ggplot2)
reorder_cormat <- function(cormat){
  # Use correlation between variables as distance
  dd <- as.dist((1-cormat)/2)
  hc <- hclust(dd)
  cormat <-cormat[hc$order, hc$order]
}

# Reorder the correlation matrix
```

```
cormat <- reorder_cormat(cormat)
upper_tri <- get_upper_tri(cormat)

# Melt the correlation matrix
melted_cormat <- melt(upper_tri, na.rm = TRUE)

# Create a ggheatmap
ggheatmap <- ggplot(melted_cormat, aes(Var2, Var1, fill = value))+
  geom_tile(color = "white")+
  scale_fill_gradient2(low = "blue", high = "red", mid = "white",
                       midpoint = 0, limit = c(-1,1), space = "Lab",
                       name="Pearson\nCorrelation") +
  theme_minimal()+ # minimal theme
  theme(axis.text.x = element_text(angle = 45, vjust = 1,
                                   size = 12, hjust = 1))+
  coord_fixed()

# Print the heatmap
print(ggheatmap)

ggheatmap +
  geom_text(aes(Var2, Var1, label = value), color = "black", size = 4) +
  theme(
    axis.title.x = element_blank(),
    axis.title.y = element_blank(),
    panel.grid.major = element_blank(),
    panel.border = element_blank(),
    panel.background = element_blank(),
    axis.ticks = element_blank(),
    legend.justification = c(1, 0),
    legend.position = c(0.6, 0.7),
    legend.direction = "horizontal")+
  guides(fill = guide_colorbar(barwidth = 5, barheight = 1,
                               title.position = "top", title.hjust = 0.4))


par(mfrow=c(2,2))

qqnorm(metascore,main = "metascore")
qqline(metascore)
qqnorm(userscore,main = "userscore")
qqline(userscore)
qqnorm(Runtime,main = "Runtime")
qqline(Runtime)
qqnorm(box_office,main = "Boxoffice in log scale")
qqline(box_office)


#non-text varaibles
summary(model_data[,3])
```

```
summary(as.factor(model_data[,6]))
genre_list = c(7:34)
sort(apply(model_data[,genre_list],2,sum))

####3. Making DTM matrix
document=c("movies_cleaned") #the document contains all movies


partialmatrix_ngram=function(document){

 ##################
 # This function is used to show the content of each file
 show.content=function(row_number,dirname){
  ds = DirSource(dirname)
  txt=read.table(ds$filelist[row_number],header=F)
  #print(ds$filelist[row_number])
  #print(txt)
  id=ds$filelist[row_number]
  return(list(ID=id,content=txt))
 }
 ##################

 ##########################################
 # This code uses tm to preprocess the papers into a format useful for NB
 preprocess.directory = function(dirname){

  # the directory must have all the relevant text files
  ds = DirSource(dirname)
  # Corpus will make a tm document corpus from this directory
  fp = Corpus( ds )
  # inspect to verify
  # inspect(fp[1])
  # another useful command
  # identical(fp[[1]], fp[["Federalist01.txt"]])
  # now let us iterate through and clean this up using tm functionality
  #for (i in 1:length(fp)){
  # make all words lower case
  fp = tm_map( fp , content_transformer(tolower));
  # remove all punctuation
  fp = tm_map( fp , removePunctuation);
  # remove stopwords like the, a, and so on.
  fp = tm_map( fp, removeWords, stopwords("english"));
  # remove stems like suffixes
  fp = tm_map( fp, stemDocument)
  # remove extra whitespace
  fp = tm_map( fp, stripWhitespace)
  # }
  # now write the corpus out to the files for our future use.
  # MAKE SURE THE _CLEAN DIRECTORY EXISTS
  writeCorpus( fp , sprintf('%s_clean',dirname) )
```

```
  }
  ##########################################
  #dir_list = c( 'fp_hamilton_train' , 'fp_hamilton_test' , 'fp_madison_train',
 'fp_madison_test')
  #ham.train=preprocess.directory("fp_hamilton_train")

  title=preprocess.directory(document)
  #title.part=preprocess.directory(partial)
  #################
  # Problem 1b
  #################

  ##########################################
  # To read in data from the directories:
  # Partially based on code from C. Shalizi

  clean=paste(document,"_clean",sep="")

  ds = DirSource(clean)
  title=Corpus( ds )


  options(mc.cores=1)
  tdm <- DocumentTermMatrix(title)

  DF <- as.matrix(tdm)
  return(DF)
}
#filenames=dir(partial)
#part_matrix=partialmatrix(document,partial,1,directory)
whole_word_matrix=partialmatrix_ngram(document)
rownames(whole_word_matrix)=rownames(dtm_sentiment)
reduced_matrix=whole_word_matrix
remove_name=c("a_place_at_the_table.txt","at_berkeley.txt","Bobby_Fischer_Against_
 The_World.txt","brooklyn_castle.txt","Corman's_World_Exploits_Of_A_Hollywood_R
 ebel.txt","Declaration_of_War.txt","Diplomacy.txt","everyday-sunshine-the-story-of-
 fishbone.txt","king-of-devils-
 island.txt","la_grande_illusion_(grand_illusion).txt","Manakamana.txt","mea_maxima_c
 ulpa_silence_in_the_house_of_god.txt","Mistaken for
 Strangers.txt","the_mother_of_george.txt","my_brother_the_devil.txt","national_gallery.
 txt","our_children.txt","silent-
 souls.txt","sister.txt","Tabu.txt","The_Crash_Reel.txt","The Dog.txt","The Kill
 Team.txt","the_last_of_the_unjust.txt","the_loved_ones.txt","the_patience_stone.txt","T
 o Be Takei.txt","Viva_Riva!.txt","The Normal Heart.txt","Behind_the_Candelabra.txt")
for(i in 1:length(remove_name)){
 reduced_matrix=reduced_matrix[-
 which(rownames(reduced_matrix)==remove_name[i]),]
 print(dim(reduced_matrix))
}
reduced_word_matrix=reduced_matrix
```

```
reudced_emot_matrix=dtm_sentiment
for(i in 1:length(remove_name)){
  reudced_emot_matrix=reudced_emot_matrix[-
 which(rownames(reudced_emot_matrix)==remove_name[i]),]
  print(dim(reudced_emot_matrix))
}
name=rownames(reudced_emot_matrix)
for(i in 1:nrow(reduced_word_matrix)){
  name[i]=gsub("_"," ",name[i])
  name[i]=gsub("-"," ",name[i])
  name[i]=tolower(name[i])
  name[i]=gsub('.txt',"",name[i])
}
sorted_name=sort(name)
ordered_emot_matrix=reudced_emot_matrix
ordered_word_matrix=reduced_word_matrix
for(i in 1:nrow(reudced_emot_matrix)){
  ordered_emot_matrix[i,]=reudced_emot_matrix[which(sorted_name[i]==name),]
  ordered_word_matrix[i,]=reduced_word_matrix[which(sorted_name[i]==name),]
}
rownames(ordered_emot_matrix)=sorted_name
rownames(ordered_word_matrix)=sorted_name



####4.Dimension Reduction and word cloud after dimension reduction

set.seed(100)
wordcloud(colnames(dtm_sentiment_final),dtm_sentiment_final,
          min.freq=15, # plot words apprear 10+ times
          scale=c(4,0.5), # make it bigger with argument "scale"
          colors=brewer.pal(8, "Dark2"), # use color palettes
          random.color=FALSE,
          random.order=FALSE)

set.seed(100)
wordcloud(colnames(whole_word_temp),whole_word_temp,
          min.freq=30, # plot words apprear 30+ times
          scale=c(4,0.5), # make it bigger with argument "scale"
          colors=brewer.pal(8, "Dark2"), # use color palettes
          random.color=FALSE,
          random.order=FALSE)

dtm_sentiment_temp = dtm_sentiment_final[,which(colSums(dtm_sentiment_final) >
 10)]

temp_number_vector = vector(length = dim(dtm_sentiment_temp)[2])
for (i in 1:dim(dtm_sentiment_temp)[2]){
  temp_number = length(y[which ( dtm_sentiment_temp[,i] != 0)])
  temp_number_vector[i] = temp_number
```

```
}

p_vector = vector(length = length(temp_number_vector))
for (i in 1:length(p_vector)){
  temp_test = t.test(y[which ( dtm_sentiment_temp[,i] == 0)], y[which
 ( dtm_sentiment_temp[,i] != 0)])
  p_vector[i] = temp_test$p.value
}

dtm_sentiment_lasso = dtm_sentiment_temp[,which(p_vector < 0.01)]

whole_word_temp = whole_word_final[,which(colSums(whole_word_final) > 30)]

temp_number_vector = vector(length = dim(whole_word_temp)[2])
for (i in 1:dim(whole_word_temp)[2]){
  temp_number = length(y[which ( whole_word_temp[,i] != 0)])
  temp_number_vector[i] = temp_number
}

whole_word_temp2 = whole_word_temp[,which(temp_number_vector>2)]

p_vector = vector(length = dim(whole_word_temp2)[2])
for (i in 1:length(p_vector)){
  temp_test = t.test(y[which ( whole_word_temp2[,i] == 0)], y[which
 ( whole_word_temp2[,i] != 0)])
  p_vector[i] = temp_test$p.value
}


whole_word_lasso = whole_word_temp2[,which(p_vector < 0.001)]

####5.Modelling and results
# Using nontext data to predict the box office
reg.ols.nontext <- regress(nontext[1:300,], log(y[1:300]))
MSE_nontext_lm = mean((log(y)[301:370] -
 cbind(rep(1,70),nontext[301:370,])%*%reg.ols.nontext$b)^2)
r_nontext_lm = cbind(rep(1,70),nontext[301:370,])%*%reg.ols.nontext$b
## Lasso regression

reg.las.nontext <- cv.glmnet(nontext[1:300,],log(y)[1:300], alpha = 1)
plot(reg.las.nontext)
coef.opt.lasso <- coef(reg.las.nontext, s="lambda.min")
coef.opt.lasso[order(abs(coef.opt.lasso[,1]), decreasing=T),][1:10          # top 10
MSE_nontext_las = mean((log(y)[301:370] -
 cbind(rep(1,70),nontext[301:370,])%*%coef.opt.lasso)^2)
r_nontext_las = cbind(rep(1,70),nontext[301:370,])%*%coef.opt.lasso
## Bayesian Lasso regression

reg.blas.nontext <- blasso(cbind(nontext[1:300,]), log(y[1:300]))
```

```
plot(reg.blas.nontext, burnin=200, which="m")

s.nontext <- summary(reg.blas.nontext, burnin=200)

MSE_nontext_blas = mean((log(y)[301:370] -
 cbind(rep(1,70),nontext[301:370,])%*%c(median(reg.blas.nontext$mu[801:1000]),apply
 (reg.blas.nontext$beta[801:1000,],2,median)))^2)

r_nontext_blas =
 cbind(rep(1,70),nontext[301:370,])%*%c(median(reg.blas.nontext$mu[801:1000]),apply
 (reg.blas.nontext$beta[801:1000,],2,median))



# Using nontext data to predict the box office
reg.ols.sentiment <- regress(dtm_sentiment_lasso[1:300,], log(y[1:300]))
MSE_sentiment_lm = mean((log(y)[301:370] -
 cbind(rep(1,70),dtm_sentiment_lasso[301:370,])%*%reg.ols.sentiment$b)^2)
r_sentiment_lm =
 cbind(rep(1,70),dtm_sentiment_lasso[301:370,])%*%reg.ols.sentiment$b
## Lasso regression

reg.las.sentiment <- cv.glmnet(dtm_sentiment_lasso[1:300,],log(y)[1:300], alpha = 1)
plot(reg.las.sentiment)
coef.opt.lasso <- coef(reg.las.sentiment, s="lambda.min")
coef.opt.lasso[order(abs(coef.opt.lasso[,1]), decreasing=T),][1:11          # top 10
MSE_sentiment_las = mean((log(y)[301:370] -
 cbind(rep(1,70),dtm_sentiment_lasso[301:370,])%*%coef.opt.lasso)^2)
r_sentiment_las = cbind(rep(1,70),dtm_sentiment_lasso[301:370,])%*%coef.opt.lasso
## Bayesian Lasso regression

reg.blas.sentiment <- blasso(dtm_sentiment_lasso[1:300,], log(y[1:300]))

plot(reg.blas.sentiment, burnin=200, which="m")

s.sentiment <- summary(reg.blas.sentiment, burnin=200)

MSE_sentiment_blas = mean((log(y)[301:370] -
 cbind(rep(1,70),dtm_sentiment_lasso[301:370,])%*%c(median(reg.blas.sentiment$mu[8
 01:1000]),apply(reg.blas.sentiment$beta[801:1000,],2,median)))^2)

r_sentiment_blas =
 cbind(rep(1,70),dtm_sentiment_lasso[301:370,])%*%c(median(reg.blas.sentiment$mu[8
 01:1000]),apply(reg.blas.sentiment$beta[801:1000,],2,median))
```

```
# Using nontext data to predict the box office
reg.ols.whole <- regress(whole_word_lasso[1:300,], log(y[1:300]))
MSE_whole_lm = mean((log(y)[301:370] -
 cbind(rep(1,70),whole_word_lasso[301:370,])%*%reg.ols.whole$b)^2)
r_whole_lm = cbind(rep(1,70),whole_word_lasso[301:370,])%*%reg.ols.whole$b
## Lasso regression

reg.las.whole <- cv.glmnet(whole_word_lasso[1:300,],log(y)[1:300], alpha = 1)
plot(reg.las.whole)
coef.opt.lasso <- coef(reg.las.whole, s="lambda.min")
coef.opt.lasso[order(abs(coef.opt.lasso[,1]), decreasing=T),][1:11          # top 10
MSE_whole_las = mean((log(y)[301:370] -
 cbind(rep(1,70),whole_word_lasso[301:370,])%*%coef.opt.lasso)^2)
r_whole_las = cbind(rep(1,70),whole_word_lasso[301:370,])%*%coef.opt.lasso
## Bayesian Lasso regression

reg.blas.whole <- blasso(whole_word_lasso[1:300,], log(y[1:300]))

plot(reg.blas.whole, burnin=200, which="m")

s.whole <- summary(reg.blas.whole, burnin=200)

MSE_whole_blas = mean((log(y)[301:370] -
 cbind(rep(1,70),whole_word_lasso[301:370,])%*%c(median(reg.blas.whole$mu[801:10
 00]),apply(reg.blas.whole$beta[801:1000,],2,median)))^2)

r_whole_blas =
 cbind(rep(1,70),whole_word_lasso[301:370,])%*%c(median(reg.blas.whole$mu[801:10
 00]),apply(reg.blas.whole$beta[801:1000,],2,median))
```

```
# Using nontext data to predict the box office
reg.ols.allcritic <- regress(all_critic[1:300,], log(y[1:300]))
MSE_allcritic_lm = mean((log(y)[301:370] -
 cbind(rep(1,70),all_critic[301:370,])%*%reg.ols.allcritic$b)^2)
r_allcritic_lm = cbind(rep(1,70),all_critic[301:370,])%*%reg.ols.allcritic$b
## Lasso regression

reg.las.allcritic <- cv.glmnet(all_critic[1:300,],log(y)[1:300], alpha = 1)
plot(reg.las.allcritic)
coef.opt.lasso <- coef(reg.las.allcritic, s="lambda.min")
coef.opt.lasso[order(abs(coef.opt.lasso[,1]), decreasing=T),][1:11          # top 10
MSE_allcritic_las = mean((log(y)[301:370] -
 cbind(rep(1,70),all_critic[301:370,])%*%coef.opt.lasso)^2)
```

```
r_allcritic_las = cbind(rep(1,70),all_critic[301:370,])%*%coef.opt.lasso

## Bayesian Lasso regression

reg.blas.allcritic <- blasso(all_critic[1:300,], log(y[1:300]))

plot(reg.blas.allcritic, burnin=200, which="m")

s.allcritic <- summary(reg.blas.allcritic, burnin=200)

MSE_allcritic_blas = mean((log(y)[301:370] -
 cbind(rep(1,70),all_critic[301:370,])%*%c(median(reg.blas.allcritic$mu[801:1000]),app
 ly(reg.blas.allcritic$beta[801:1000,],2,median)))^2)

r_allcritic_blas =
 cbind(rep(1,70),all_critic[301:370,])%*%c(median(reg.blas.allcritic$mu[801:1000]),app
 ly(reg.blas.allcritic$beta[801:1000,],2,median))



# Using nontext data to predict the box office
reg.ols.allsentiment <- regress(all_sentiment[1:300,], log(y[1:300]))
MSE_allsentiment_lm = mean((log(y)[301:370] -
 cbind(rep(1,70),all_sentiment[301:370,])%*%reg.ols.allsentiment$b)^2)
r_allsentiment_lm = cbind(rep(1,70),all_sentiment[301:370,])%*%reg.ols.allsentiment$b
## Lasso regression

reg.las.allsentiment <- cv.glmnet(all_sentiment[1:300,],log(y)[1:300], alpha = 1)
plot(reg.las.allsentiment)
coef.opt.lasso <- coef(reg.las.allsentiment, s="lambda.min")
coef.opt.lasso[order(abs(coef.opt.lasso[,1]), decreasing=T),][1:11          # top 10
MSE_allsentiment_las = mean((log(y)[301:370] -
 cbind(rep(1,70),all_sentiment[301:370,])%*%coef.opt.lasso)^2)
r_allsentiment_las = cbind(rep(1,70),all_sentiment[301:370,])%*%coef.opt.lasso
## Bayesian Lasso regression

reg.blas.allsentiment <- blasso(all_sentiment[1:300,], log(y[1:300]))

plot(reg.blas.allsentiment, burnin=200, which="m")

s.allsentiment <- summary(reg.blas.allsentiment, burnin=200)

MSE_allsentiment_blas = mean((log(y)[301:370] -
 cbind(rep(1,70),all_sentiment[301:370,])%*%c(median(reg.blas.allsentiment$mu[801:1
 000]),apply(reg.blas.allsentiment$beta[801:1000,],2,median)))^2)

r_allsentiment_blas =
 cbind(rep(1,70),all_sentiment[301:370,])%*%c(median(reg.blas.allsentiment$mu[801:1
 000]),apply(reg.blas.allsentiment$beta[801:1000,],2,median))
```

```
df_nontext =
 data.frame(box_office[301:370],r_nontext_lm,r_nontext_las,r_nontext_blas)
names (df_nontext) = c('box_office','lm','las','blas')

df_whole =
 data.frame(box_office[301:370],r_whole_lm,as.matrix(r_whole_las),r_whole_blas)
names (df_whole) = c('box_office','lm','las','blas')

df_sentiment =
 data.frame(box_office[301:370],r_sentiment_lm,as.matrix(r_sentiment_las),r_sentiment
 _blas)
names (df_sentiment) = c('box_office','lm','las','blas')

df_allcritic =
 data.frame(box_office[301:370],r_allcritic_lm,as.matrix(r_allcritic_las),r_allcritic_blas)
names (df_allcritic) = c('box_office','lm','las','blas')

df_allsentiment =
 data.frame(box_office[301:370],r_allsentiment_lm,as.matrix(r_allsentiment_las),r_allse
 ntiment_blas)
names (df_allsentiment) = c('box_office','lm','las','blas')

matplot(df_allcritic, type = c("l"), pch=1:4,lwd = 1:4, lty = 1:4,col = 1:4,main = 'critic +
 nontext') #plot
legend('topleft',legend = c('box_office','lm','las','blas'),lwd = 1:4,lty = 1:4,col=1:4) #
 optional legend

matplot(df_allsentiment, type = c("l"), pch=1:4,lwd = 1:4, lty = 1:4,col = 1:4,main =
 'sentiment + nontext') #plot
legend('topleft',legend = c('box_office','lm','las','blas'),lwd = 1:4,lty = 1:4,col=1:4) #
 optional legend

matplot(df_nontext, type = c("l"), pch=1:4,lwd = 1:4, lty = 1:4,col = 1:4, main = 'nontext')
 #plot
legend('topleft',legend = c('box_office','lm','las','blas'),lwd = 1:4,lty = 1:4,col=1:4) #
 optional legend

matplot(df_whole, type = c("l"), pch=1:4,lwd = 1:4, lty = 1:4,col = 1:4, main = 'critic')
 #plot
legend('topleft',legend = c('box_office','lm','las','blas'),lwd = 1:4,lty = 1:4,col=1:4) #
 optional legend

matplot(df_sentiment, type = c("l"), pch=1:4,lwd = 1:4, lty = 1:4,col = 1:4,main =
 'sentiment') #plot
legend('topleft',legend = c('box_office','lm','las','blas'),lwd = 1:4,lty = 1:4,col=1:4) #
 optional legend
```