Homework 6

Jingyi Guo (jingyig1), Pittsburgh Campus

12/11/2017

```
setwd("~/Desktop/ML2/Homework 6")
```

1

(a)

```
library(tm)
```

```
## Loading required package: NLP
```

```
# Load the data. This will create: stories: a character vector with one
# article in each element class: a character vector specifying whether each
# article is about 'music' or 'art'
load("./articles.RData")
### Stop here, take a look inside both variables to see what you have.
# Convert that vector into a 'corpus' that the tm package will recognize
corpus = VCorpus(VectorSource(stories))
# Compute a document term matrix. Set this to remove numbers and
# punctuation, and to change all words to lowercase.
dtm = DocumentTermMatrix(corpus, control = list(tolower = T, removeNumbers = T,
removePunctuation = T))
# Make a matrix, so that it's easier to work with as numbers
dtm_mat = as.matrix(dtm)
print(nrow(dtm_mat))
```

```
## [1] 102
```

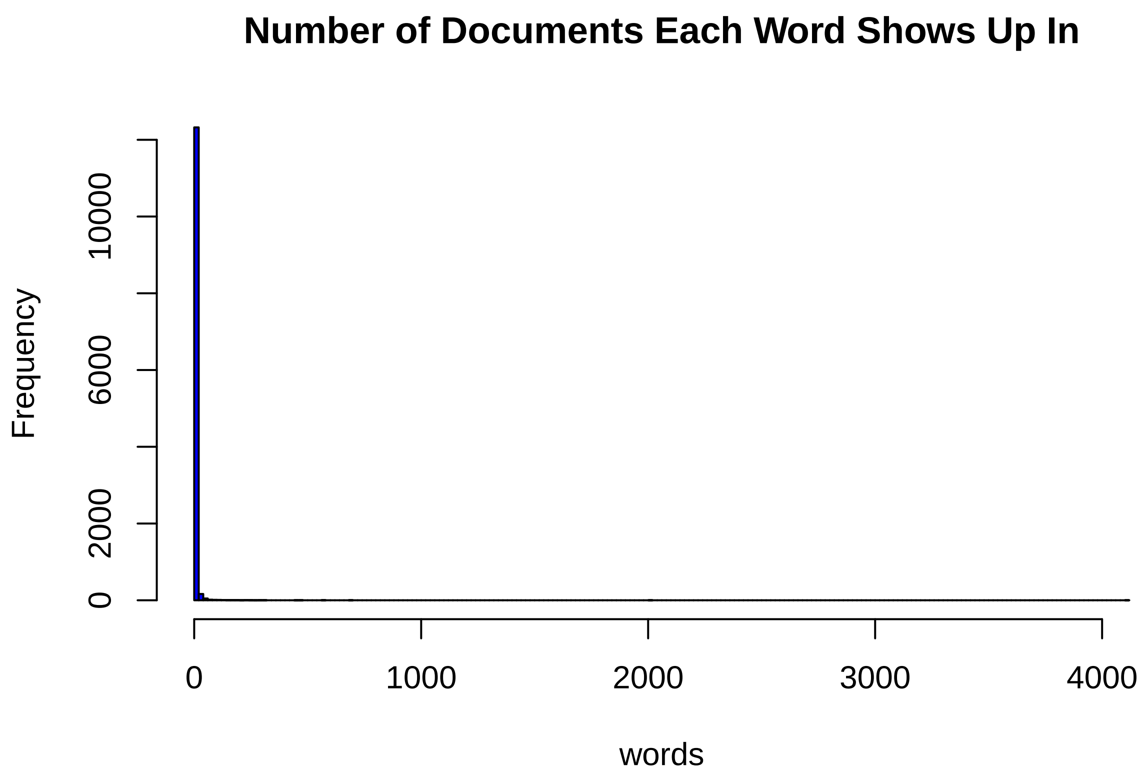
```
print(ncol(dtm_mat))
```

```
## [1] 12612
```

So there are 102 documents in data set and 12612 words in the document-term matrix.

(b)

```
num_doc = colSums(dtm_mat)
hist(num_doc, breaks = 200, main = "Number of Documents Each Word Shows Up In",xlab = "words", col = "bl
```




```
print(colnames(dtm_mat)[which(num_doc ==max(num_doc))])
```

```
## [1] "the"
```

```
print(max(num_doc))
```

```
## [1] 4110
```

The word that appears in the most documents is "the" and it appears 4110 times.

(c)

```
dtm_new = DocumentTermMatrix(corpus, control = list(tolower = T, removeNumbers = T, removePunctuation =
dtm_new_mat = as.matrix(dtm_new)
print(nrow(dtm_new_mat))
```

```
## [1] 102
```

```
print(ncol(dtm_new_mat))
```

```
## [1] 2336
```

So there are 102 documents in data set and 2336 words in the new document-term matrix.

```
num_new_doc = colSums(dtm_new_mat)
print(colnames(dtm_new_mat)[which(num_new_doc ==max(num_new_doc))])
```

```
## [1] "said"
```

```
print(max(num_new_doc))
```

```
## [1] 294
```

The word that appears in the most documents is "said" and it appears 294 times.

(d)

```
colnames(dtm_new_mat)[1:10]
```

```
##  [1] "abc"           "ability"       "able"          "absorbed"
##  [5] "abstract"      "abstraction"   "abstractions"  "academic"
##  [9] "academy"       "access"
```

We can see that some words with different forms have similar meanings, such as "able" and "ability". If we do not consider this stemming problem, we may mistakenly count the number of words.

```
dtm_stem = DocumentTermMatrix(corpus, control = list(tolower = T, removeNumbers = T, removePunctuation =
dtm_stem_mat = as.matrix(dtm_stem)
colnames(dtm_stem_mat)[1:10]
```

```
##  [1] "abandon"  "abc"      "abil"     "abl"      "absorb"   "abstract"
##  [7] "abund"    "academ"   "academi"  "accept"
```

We can see the stemming problem is partially solved. There are still (fragments of) words with similar meanings, such as "academ" and "academi"

```
print(nrow(dtm_stem_mat))
```

```
## [1] 102
```

```
print(ncol(dtm_stem_mat))
```

```
## [1] 2170
```

So there are 102 documents in data set and 2170 words in the new document-term matrix.

```
num_stem_doc = colSums(dtm_stem_mat)
print(colnames(dtm_stem_mat)[which(num_stem_doc ==max(num_stem_doc))])
```

```
## [1] "art"
```

```
print(max(num_stem_doc))
```

```
## [1] 305
```

The word that appears in the most documents is "art" and it appears 305 times.

2

(a)

```
library(topicmodels)
# These are just some decent default parameters. For now, set them for your
# homework and don't worry about them too much
burnin <- 4000
iter <- 2000
thin <- 500
seed <- list(2003, 5, 63, 100001, 765)
nstart <- 5
best <- TRUE
```

```r
# Number of topics (This is the interesting parameter)
k <- 2
# Run LDA using Gibbs sampling
lda_out <- LDA(dtm_stem, k, method = "Gibbs", control = list(nstart = nstart,
seed = seed, best = best, burnin = burnin, iter = iter, thin = thin))
# docs to topics
lda_topics <- as.matrix(topics(lda_out))
# top 10 terms in each topic
lda_terms <- as.matrix(terms(lda_out, 20)) # probabilities associated with each topic assignment
topic_probabilities <- as.data.frame(lda_out@gamma)
print(lda_terms[1:10, ])
```

```
##       Topic 1  Topic 2
##  [1,] "art"    "said"
##  [2,] "work"   "will"
##  [3,] "artist" "music"
##  [4,] "paint"  "show"
##  [5,] "museum" "new"
##  [6,] "one"    "year"
##  [7,] "like"   "time"
##  [8,] "also"   "first"
##  [9,] "new"    "one"
## [10,] "use"    "play"
```

(b)

```r
class_id = factor(class, labels = c(1, 2))
err = sum(lda_topics != class_id)/length(lda_topics)
print(round(err,6))
```

```
## [1] 0.117647
```

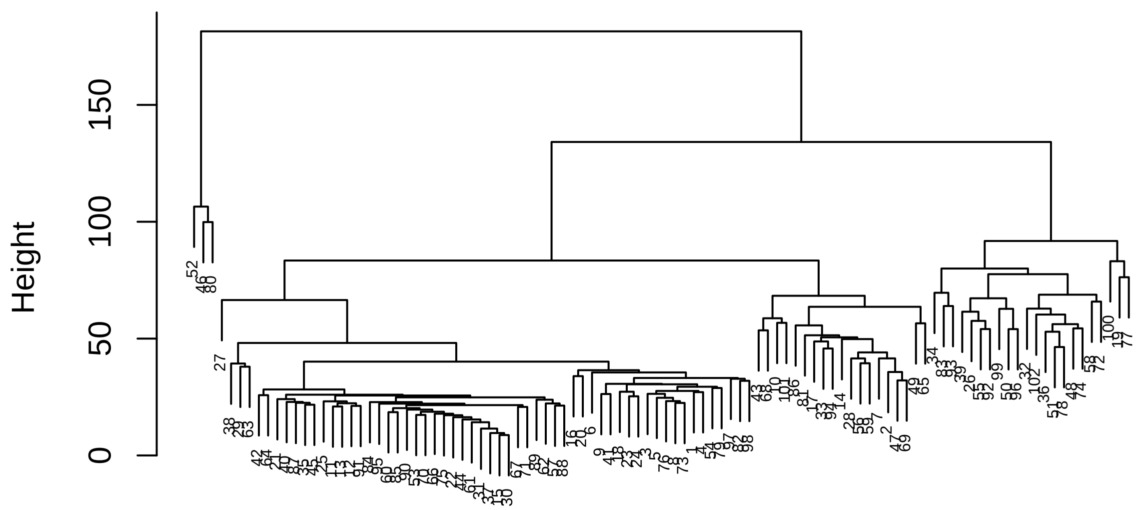
The misclassification error is 0.117647.

3

(a)

```r
X = dtm_mat
dtm_dist = dist(X)
dtm_tree = hclust(dtm_dist, method = "complete")
plot(dtm_tree, main = "Complete Method Cluster Dendrogram", cex = 0.5, xlab = "")
```

Complete Method Cluster Dendrogram



hclust (*, "complete")

```r
# mean of length of story
mean(nchar(stories))
```

```
## [1] 4107.206
```

```r
# median of length of story
median(nchar(stories))
```

```
## [1] 2926
```

```r
# story length of point #46
nchar(stories)[46]
```

```
## [1] 12210
```

```r
# story length of point #52
nchar(stories)[52]
```

```
## [1] 17014
```

```r
# story lengh of point #80
nchar(stories)[80]
```

```
## [1] 10774
```

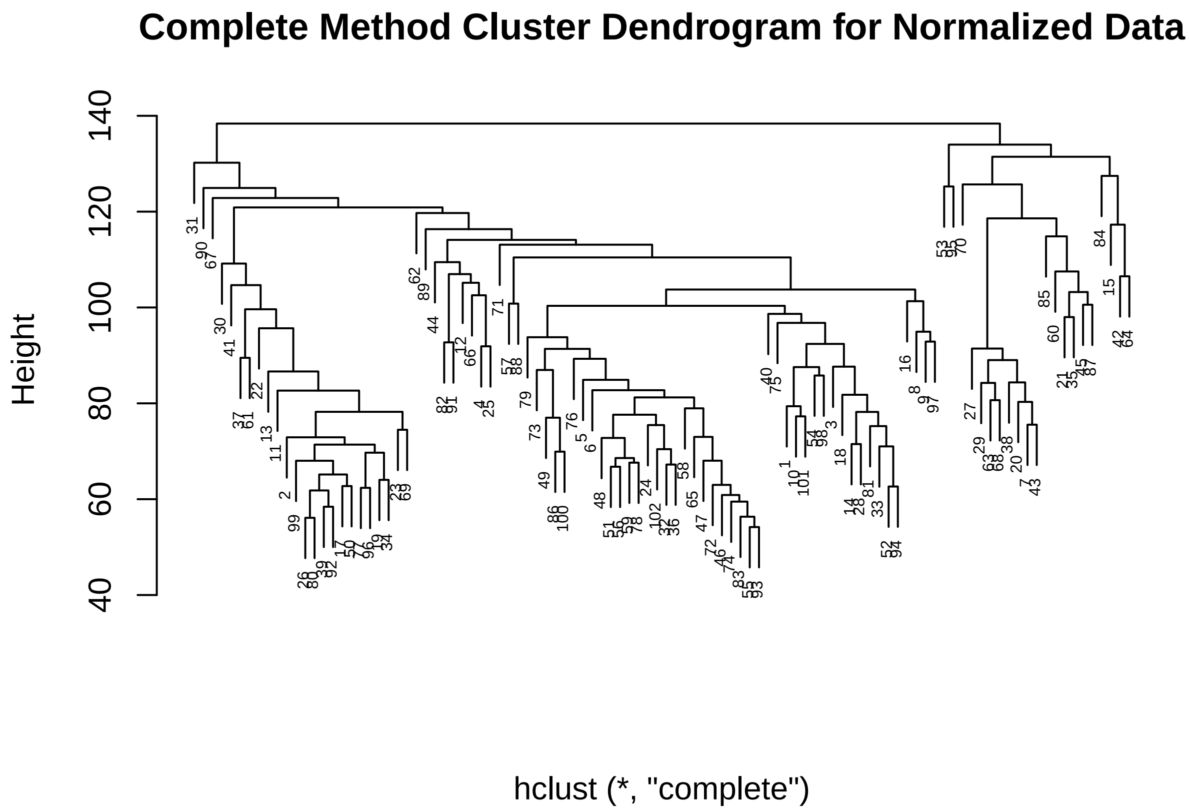
The dendrogram tree is unbalanced on the left part with indices of 46, 52 and 80.

The mean of length of story is 4107.206 and median of length of story is 2926.

Story length is 12210 for point #46, 17014 for point #52 and 10774 for point #80.

(b)

```r
X2 = t(scale(t(X), scale = T))
dtm_norm_dist = dist(X2)
dtm_norm_tree = hclust(dtm_norm_dist, method = "complete")
plot(dtm_norm_tree, main = "Complete Method Cluster Dendrogram for Normalized Data",
cex = 0.5, xlab = "")
```



The dendrogram looks more balanced and more like a reasonable clustering, mostly evenly divided into two subgroups and the outliers disappeared.

(c)

```r
labels = cutree(dtm_norm_tree, k = 6)
cut_id = ifelse(labels == 1, 1, 2)
err = sum(class_id != cut_id)/length(class_id)
print(round(err, 6))
```

```
## [1] 0.421569
```

The misclassification error is 0.421569.

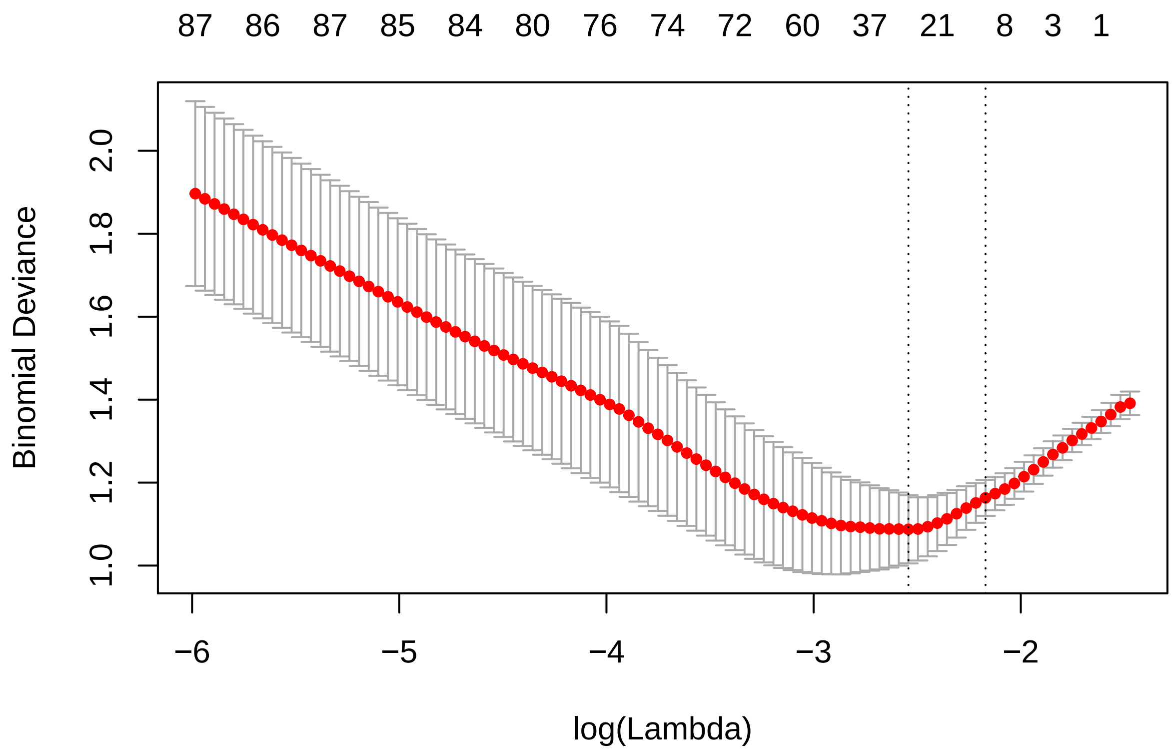
4

(a)

```
set.seed(0)
library(glmnet)
```

```
## Loading required package: Matrix
## Loading required package: foreach
## Loaded glmnet 2.0-13
```

```
cv_lasso = cv.glmnet(x = X, y = class_id, family = "binomial", alpha = 1)
plot(cv_lasso)
```




```
opt_lambda_pos = which(cv_lasso$glmnet.fit$lambda == cv_lasso$lambda.1se)
opt_lambda = cv_lasso$lambda[opt_lambda_pos]
cv_error = cv_lasso$cvm[opt_lambda_pos]
print(opt_lambda)
```

```
## [1] 0.1141329
```

```
print(cv_error)
```

```
## [1] 1.163134
```

We choose the optimal lambda as 0.1141329, and the cross validation error is 1.163134.

```
set.seed(0)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

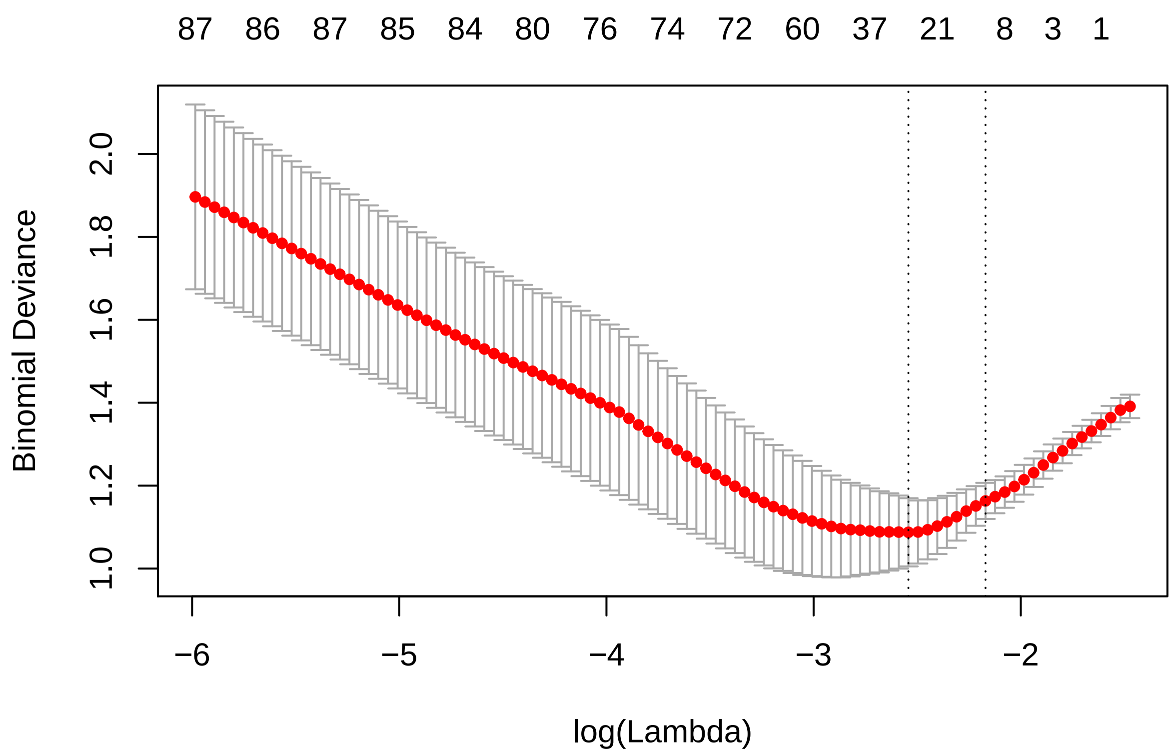
```
rf = randomForest(x = X, y = factor(class_id))
print(rf$err.rate[nrow(rf$err.rate),1])
```

```
##       OOB
## 0.1764706
```

The OOB error of random forest is 0.1764706.

(b)

```
set.seed(0)
X_clustering = cbind(X, cut_id)
cv_lasso = cv.glmnet(x = X_clustering, y = class_id, family = "binomial", alpha = 1)
plot(cv_lasso)
```




```
opt_lambda_pos = which(cv_lasso$glmnet.fit$lambda == cv_lasso$lambda.1se)
opt_lambda = cv_lasso$lambda[opt_lambda_pos]
cv_error = cv_lasso$cvm[opt_lambda_pos]
print(opt_lambda)
```

```
## [1] 0.1141329
```

```
print(cv_error)
```

```
## [1] 1.163134
```

The optimal lambda is 0.1141329 and the cross validation error is 1.163134.

```
set.seed(0)
rf = randomForest(x = X_clustering, y = factor(class_id))
print(rf$err.rate[nrow(rf$err.rate), 1])
```

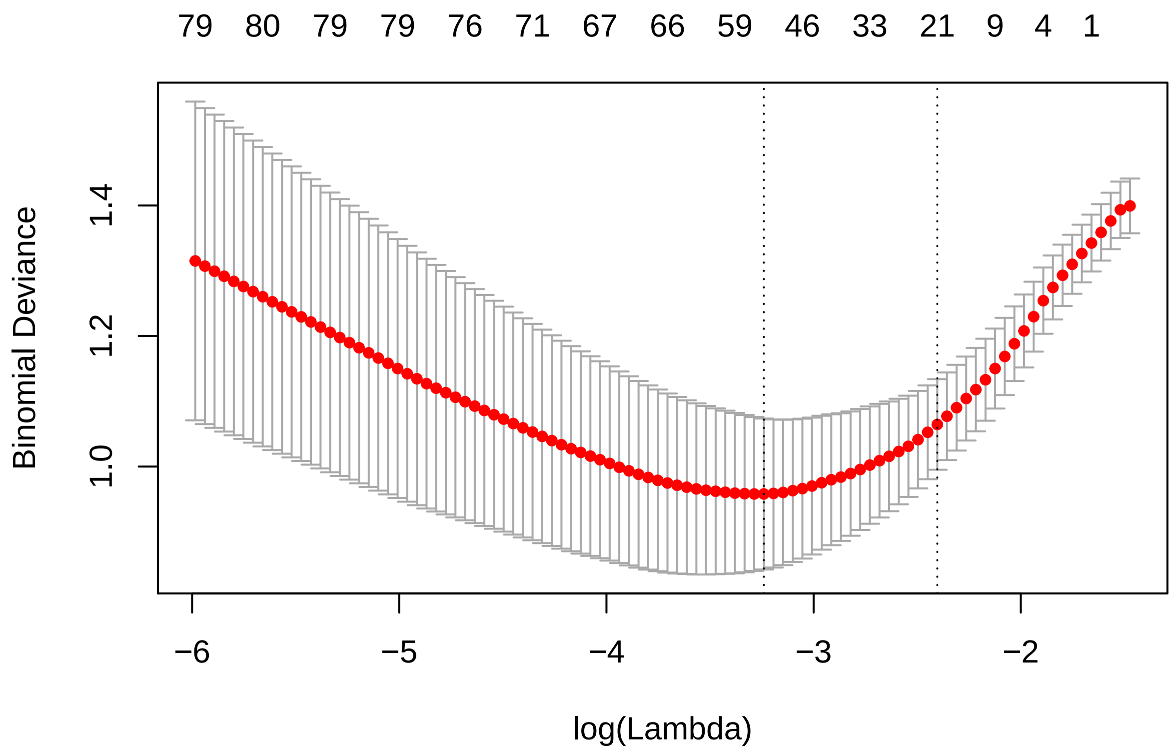
```
##       OOB
## 0.254902
```

The OOB error is 0.254902.

Adding additional feature of clustering does not help with the logistic lasso classification result and it does not improve the OOB error of random forest.

(c)

```
k <- 5
# Run LDA using Gibbs sampling
lda_out <- LDA(dtm_stem, k, method = "Gibbs", control = list(nstart = nstart,
seed = seed, best = best, burnin = burnin, iter = iter, thin = thin))
# docs to topics
lda_topics <- as.matrix(topics(lda_out))
# top 10 terms in each topic
lda_terms <- as.matrix(terms(lda_out, 20)) # probabilities associated with each topic assignment
topic_probabilities <- as.data.frame(lda_out@gamma)
X_lda = cbind(X, lda_topics)
cv_lasso = cv.glmnet(x = X_lda, y = class_id, family = "binomial", alpha = 1)
plot(cv_lasso)
```




```
opt_lambda_pos = which(cv_lasso$glmnet.fit$lambda == cv_lasso$lambda.1se)
opt_lambda = cv_lasso$lambda[opt_lambda_pos]
cv_error = cv_lasso$cvm[opt_lambda_pos]
print(opt_lambda)
```

```
## [1] 0.09044837
```

```r
print(cv_error)
```

```
## [1] 1.064487
```

The optimal lambda is 0.09044837, and the cross validation error is 1.064487.

```r
set.seed(0)
rf = randomForest(x = X_lda, y = factor(class_id))
print(rf$err.rate[nrow(rf$err.rate), 1])
```

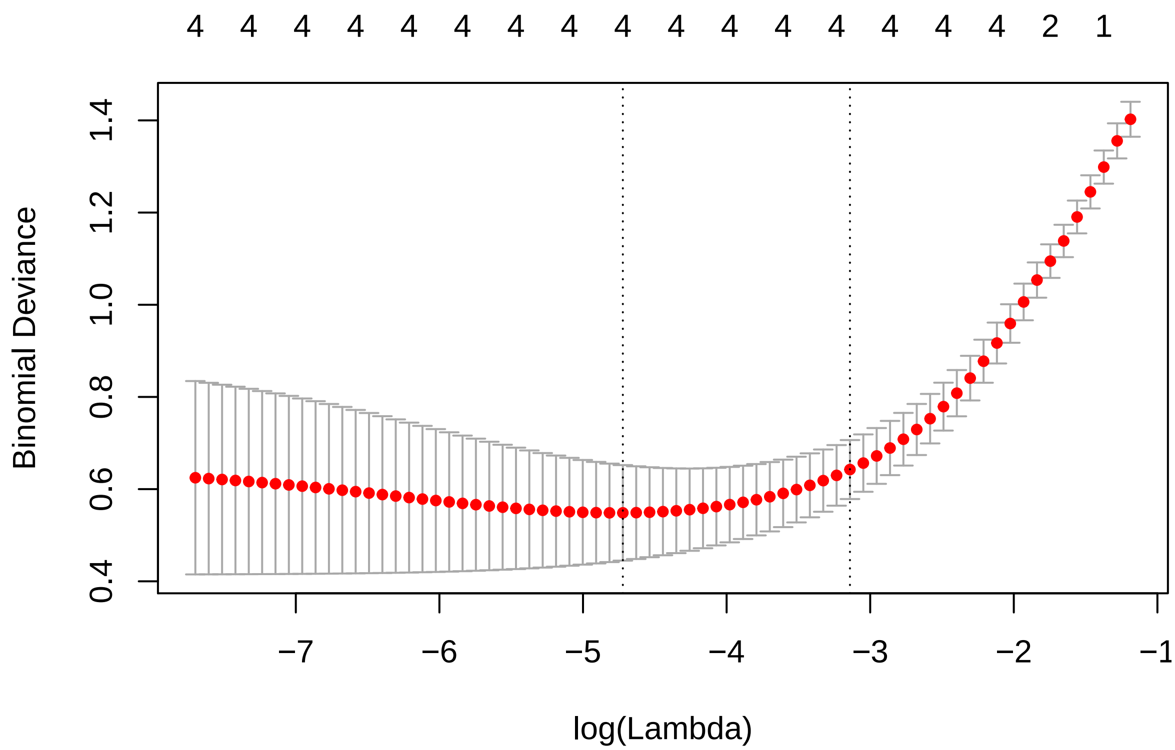
```
##       OOB
## 0.1960784
```

The OOB error is 0.1960784.

The cross validation error of the logistic lasso is slightly lower, but it doesn't improve the OOB error result of random forest.

(d)

```r
X = as.matrix(topic_probabilities)
cv_lasso = cv.glmnet(x = X, y = class_id, family = "binomial", alpha = 1)
plot(cv_lasso)
```




```r
opt_lambda_pos = which(cv_lasso$glmnet.fit$lambda == cv_lasso$lambda.1se)
opt_lambda = cv_lasso$lambda[opt_lambda_pos]
cv_error = cv_lasso$cvm[opt_lambda_pos]
print(opt_lambda)
```

```
## [1] 0.04324195
```

```r
print(cv_error)
```

```
## [1] 0.642471
```

The optimal lambda is 0.04324195, and the cross validation error is 0.642471.

```r
set.seed(0)
rf = randomForest(x = X, y = factor(class_id))
print(rf$err.rate[nrow(rf$err.rate), 1])
```

```
##       OOB
## 0.1176471
```

The OOB error is 0.1176471.

If we just use the five features from the LDA alone, we can have better result in both logistic lasso classification and in random forest.