# Author attribution

```r
library(tm)
```

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

```r
library(foreach)
library(randomForest)
```

```
## randomForest 4.6-14
```

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

```r
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 2.0-16
```

```r
library(tidyverse)
```

```
## -- Attaching packages ------------------------------- tidyverse 1.2.1 --
```

```
## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
```

```
## -- Conflicts ---------------------------------- tidyverse_conflicts() --
## x purrr::accumulate() masks foreach::accumulate()
## x ggplot2::annotate() masks NLP::annotate()
## x dplyr::combine()    masks randomForest::combine()
## x tidyr::expand()     masks Matrix::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::lag()        masks stats::lag()
## x ggplot2::margin()   masks randomForest::margin()
## x purrr::when()       masks foreach::when()
```

```r
set.seed("6")

readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }
#read in train and test
train_list = Sys.glob('ReutersC50/C50train/*/*.txt')
test_list = Sys.glob('ReutersC50/C50test/*/*.txt')
file_list = c(train_list,test_list)

alldata = lapply(file_list, readerPlain)

filename = file_list %>%
{ strsplit(., '/', fixed=TRUE) } %>%
{ lapply(., tail, n=2) } %>%
{ lapply(., paste0, collapse = '') } %>%
  unlist

authorname = file_list %>%
{ strsplit(., '/', fixed=TRUE) } %>%
{ lapply(., tail, n=2) } %>%
{ lapply(., head, n=1) } %>%
  unlist

names(alldata) = filename

documents_raw = Corpus(VectorSource(alldata))
#tokenization process
my_documents = documents_raw
my_documents = tm_map(my_documents, content_transformer(tolower))
```

```
## Warning in tm_map.SimpleCorpus(my_documents, content_transformer(tolower)):
## transformation drops documents
```

```r
my_documents = tm_map(my_documents, content_transformer(removeNumbers))
```

```
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removeNumbers)): transformation drops documents
```

```r
my_documents = tm_map(my_documents, content_transformer(removePunctuation))
```

```
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removePunctuation)): transformation drops documents
```

```r
my_documents = tm_map(my_documents, content_transformer(stripWhitespace))
```

```
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(stripWhitespace)): transformation drops documents
```

```r
my_documents = tm_map(my_documents, content_transformer(removeWords), stopwords("en"))
```

```
## Warning in tm_map.SimpleCorpus(my_documents,
## content_transformer(removeWords), : transformation drops documents
```

```
#create document term matrix and remove sparcity
DTM_all = DocumentTermMatrix(my_documents)
DTM_all = removeSparseTerms(DTM_all, 0.95)

tfidf_all = weightTfIdf(DTM_all)

X = as.matrix(tfidf_all)
summary(colSums(X))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    9.07   12.03   13.49   16.22   64.05
```

```
scrub_cols = which(colSums(X) == 0)
X = X[,-scrub_cols]
#conduct principle component analysis
pca_train = prcomp(X, scale=TRUE)

X = pca_train$x[1:2500,1:100]
y = authorname[1:2500]
```

```
#Model1: lasso
out1 = cv.glmnet(X, y, family='multinomial', type.measure="class")

lambda_hat = out1$lambda.min

paste("The chosen lambda is",lambda_hat)
```

```
## [1] "The chosen lambda is 0.000562636650667486"
```

```
predict.lasso = predict.cv.glmnet(out1,pca_train$x[2501:5000,1:100],s=lambda_hat)

predict.name = vector()
for (i in 1:2500){
  a=which(predict.lasso[i,,1] == max(predict.lasso[i,,1]))
  predict.name = c(predict.name,colnames(predict.lasso)[a])

}

result = predict.name == authorname[2501:5000]
paste("The correction ratio is",sum(result) / length(result))
```

```
## [1] "The correction ratio is 0.5832"
```

```
a=table(predict.name, authorname[2501:5000])
a[1:10,1:10]
```

```
##
## predict.name       AaronPressman AlanCrosby AlexanderSmith BenjaminKangLim
##   AaronPressman               34          0              2               0
##   AlanCrosby                   0         25              0               0
##   AlexanderSmith               0          0             25               0
##   BenjaminKangLim              0          0              0              17
##   BernardHickey                0          0              0               0
##   BradDorfman                  1          0              0               0
##   DarrenSchuettler             0          0              0               0
##   DavidLawder                  0          0              0               0
##   EdnaFernandes                0          0              1               0
##   EricAuchard                  0          0              0               0
##
## predict.name       BernardHickey BradDorfman DarrenSchuettler DavidLawder
##   AaronPressman                0           0                1           0
##   AlanCrosby                   0           0                0           0
##   AlexanderSmith               0           0                0           0
##   BenjaminKangLim              0           0                0           0
##   BernardHickey               36           0                0           0
##   BradDorfman                  0          32                0           9
##   DarrenSchuettler             0           0                9           0
##   DavidLawder                  0           2                0           6
##   EdnaFernandes                0           0                0           0
##   EricAuchard                  0           0                0           0
##
## predict.name       EdnaFernandes EricAuchard
##   AaronPressman                0           0
##   AlanCrosby                   0           0
##   AlexanderSmith               4           0
##   BenjaminKangLim              0           0
##   BernardHickey                0           0
##   BradDorfman                  0           2
##   DarrenSchuettler             0           0
##   DavidLawder                  0           0
##   EdnaFernandes               15           0
##   EricAuchard                  0          22
```

```
############
p=2500-1
mtryv = sqrt(p)
ntreev = 600

a=data.frame(authorname[1:2500],pca_train$x[1:2500,1:50])
colnames(a)[1]="author"

temprf = randomForest(author ~., data=a,mtry=mtryv,ntree=ntreev)
predict.name = predict(temprf,newdata=pca_train$x[2501:5000,1:50])
result = predict.name == authorname[2501:5000]
paste("The correction ratio is", sum(result) / length(result))
```

```
## [1] "The correction ratio is 0.4628"
```

```
t=table(predict.name, authorname[2501:5000])
t[1:10,1:10]
```

```
##
## predict.name       AaronPressman AlanCrosby AlexanderSmith BenjaminKangLim
##   AaronPressman               31          0              0               0
##   AlanCrosby                   0         18              0               0
##   AlexanderSmith               0          0             22               0
##   BenjaminKangLim              0          0              0              12
##   BernardHickey                0          0              0               0
##   BradDorfman                  1          0              0               0
##   DarrenSchuettler             2          0              2               0
##   DavidLawder                  0          0              0               0
##   EdnaFernandes                0          0              1               0
##   EricAuchard                  0          0              0               0
##
## predict.name       BernardHickey BradDorfman DarrenSchuettler DavidLawder
##   AaronPressman                2           1                0           0
##   AlanCrosby                   0           0                0           0
##   AlexanderSmith               0           0                0           0
##   BenjaminKangLim              0           0                0           0
##   BernardHickey               22           0                0           0
##   BradDorfman                  0          27                1           1
##   DarrenSchuettler             0           0                8           1
##   DavidLawder                  0           0                0           7
##   EdnaFernandes                0           0                0           0
##   EricAuchard                  0           0                0           1
##
## predict.name       EdnaFernandes EricAuchard
##   AaronPressman                0           0
##   AlanCrosby                   0           0
##   AlexanderSmith               1           0
##   BenjaminKangLim              0           0
##   BernardHickey                0           0
##   BradDorfman                  0           1
##   DarrenSchuettler             0           0
##   DavidLawder                  0           0
##   EdnaFernandes               14           0
##   EricAuchard                  0          12
```

```
k=0
i1=0
j1=0
for (i in 1:50) {
  for (j in 1:50) {
      if(i != j){
        if(t[i,j]>k){
          k=t[i,j]
          i1=i
          j1=j
        }
      }
  }
}

t[12:14,6:8]
```

```
##
## predict.name       BradDorfman DarrenSchuettler DavidLawder
##   GrahamEarnshaw             0                0           0
##   HeatherScoffield           0               35           0
##   JaneMacartney              0                0           0
```

From the result we can see that the Lasso regression model is better than random forest. DarrenSchuettler and HeatherScoffield are most easily be distinguished.