```
#################################
##第十五回「トピックモデル：Latent Dirichlet Allocation」
#################################

library("Matrix")
library("slam")
library("topicmodels")

# source("sprase2ldacformat.R")

x <- readMM('jawiki1_short_bow.mm') #単語頻度行列の読み込み
titles <- read.delim2('jawiki1_short_titles_tmp.txt') #記事タイトル一覧を読み込
み
#文字コードをRの環境に合わせるため，shift-jisにしてある．各自の環境に合わせて変
更しておくと良い．
tts[titles[[1]]+1] <- as.character(titles[[2]])
voclist <- read.delim2('jawiki1_short_wordids_tmp.txt') #単語一覧を読み込み
vvs <- c(1:length(voclist[[1]]))
vvs[voclist[[1]]+1] <- as.character(voclist$aa)

dd = dim(x)

#そのままのデータは大きいので，適切なサイズに縮小
vocsize = 50000  #単語数
docsize = 10000  #記事数

set.seed(101)
rsum <- rowSums(x)
csum <- colSums(x)

# 上位vocsize分の頻出単語を採用
res <- sort.int(csum,decreasing = TRUE,index.return = TRUE)
sc <- res$x
sc.ind <- res$ix[1:vocsize]
sx <- x[,sc.ind]

# 単語が50個以上使われている記事からランダムにdocsize分選ぶ
sxrsum <- rowSums(sx)
tmpind = 1:dd[1]
sr.ind = sample(tmpind[sxrsum>50],docsize)
sx <- sx[sr.ind,]

# サイズ変更に伴い単語，記事のインデックスを修正
svoc <- vvs[sc.ind] #単語ベクトル
stit <- tts[sr.ind] #記事タイトルベクトル

# LDA()用にsimple_triplet_matrixフォーマットに変更
# 詳しくはslamパッケージのドキュメントを参照
ssx <- as.simple_triplet_matrix(sx)
ssx$dim = dim(ssx)
ssx$dimnames[[1]] <- stit #記事タイトル
ssx$dimnames[[2]] <- svoc #単語

# LDAを実行．推定方法はGibbsサンプリングを使う．
K = 20 #トピック数
wiki.lda <- LDA(ssx,K,method='Gibbs',control = list(burnin=2000,iter = 5000))
# 実行は小一時間かかる．


######################
## 結果の表示
# wiki.lda@beta:各トピックの単語生成確率のlog
# wiki.lda@gamma:各文章のトピックの割合

# トピックごとに主要単語の列挙
for(i in 1:K){
    print(paste('Topic',i,':'))
    res <- sort.int(wiki.lda@beta[i,],decreasing=TRUE,index.return = TRUE) #トピ
ックiからの生成確率が高い順に単語を取ってくる
    print(paste0(svoc[res$ix[1:20]],collapse =" | "))
}
```

```
#  トピックごとに主要記事タイトルの列挙
for(i in 1:K){
    print(paste('Topic',i,':'))
    res <- sort.int(wiki.lda@gamma[,i],decreasing=TRUE,index.return = TRUE) #ト
ピックiに属している確率が高い順に記事を取ってくる
    print(paste0(stit[res$ix[1:20]],collapse =" | "))
}


res <- terms(wiki.lda,10) #各トピックごとの主要単語を10個列挙
res

library(wordcloud)
##単語のワードクラウド
par(mfrow=c(2,5))
for(i in 1:10){
    res <- sort.int(wiki.lda@beta[i,],decreasing=TRUE,index.return = TRUE)
    wordcloud(svoc[res$ix[1:15]],exp(res$x[1:15]))
}
par(mfrow=c(1,1))

par(mfrow=c(2,5))
for(i in 11:20){
    res <- sort.int(wiki.lda@beta[i,],decreasing=TRUE,index.return = TRUE)
    wordcloud(svoc[res$ix[1:15]],exp(res$x[1:15]))
}
par(mfrow=c(1,1))
```