Practical 6: Latent Dirichlet Allocation

In this practical, we will process the text data that includes the abstracts of all papers in the Journal of Statistical Software (JSS), up to 08/05/2010.
The JSS data is available as a list matrix in the package `corpus.JSS.papers` which can be installed and loaded by:

```
### Download  the collection of abstracts of the Journal of Statistical Software (JSS)

if(!require(corpus.JSS.papers))
install.packages("corpus.JSS.papers", repos =
                   "http://datacube.wu.ac.at/", type = "source")

data("JSS_papers", package = "corpus.JSS.papers")
View(JSS_papers[1:2,])
```

	title	creator	subject	description
1	A Diagnostic to Assess the Fit of a Variogram M...	Barry, Ronald	c("", "", "")	The fit of a variogram
2	Homogeneity Analysis in Xlisp-Stat	c("Bond, Jason", "Michailides, George")	c("", "", "")	In this paper a highly

I. Processing text data

In this section, we use the `tm` and `XML` packages to process the JSS dataset.

1. Install the `tm` and `XML` packages.

```
#I. Processing text data
# 1. Install the tm and XML packages
if(!require(tm))
  install.packages("tm")
if(!require(XML))
  install.packages("XML")
```

2. We use only abstracts published up to 2010-08-05 and omit those containing non-ASCII characters in the abstracts.

```
#2. We use only abstracts published up to 2010-08-05
JSS_papers <- JSS_papers[JSS_papers[,"date"] < "2010-08-05",]
# and omit those containing non-ASCII characters in the abstracts.
JSS_papers <- JSS_papers[sapply(JSS_papers[, "description"]
                                ,Encoding) == "unknown",]

dim(JSS_papers)
```

```
> dim(JSS_papers)
[1] 342  15
```

Install SnowballC

```
if(!require(SnowballC))
  install.packages("SnowballC")
# Load libraries
library("tm")
library("XML")
library("SnowballC")
```

We may want to create a function to remove any html markup. Following examples show how to do use XML to extract text only. (This part is only for demonstrative purposes and does not form part of the practical)

```
### Suppose you think some abstracts include html markups
## (e.g. sentence 1 <a>text</a> sentence 2 <div>another text</div>)
## you can use XML package to extract only text

dummyExample <- "sentence 1 <a>text</a> sentence 2 <div>another text</div>"
doc <-  htmlTreeParse(dummyExample, asText = TRUE, trim = FALSE)
doc <- xmlValue(xmlRoot(doc))
doc
```

```
> doc
[1] "sentence 1 text sentence 2 another text"
```

```
# note: if text include math symbol <, it will be confused by html markups
# causing a potential lost of text

dummyExample <- "number a < number b"
doc <-  htmlTreeParse(dummyExample, asText = TRUE, trim = FALSE)
```

```
doc <- xmlValue(xmlRoot(doc))
doc
```

```
> doc
[1] "number a "
```

3. The final data set contains 342 documents. Before analysis we transform it to a "**Corpus**" using package `tm`. HTML markup in the abstracts for greek letters, subscripting, etc., is removed using package `XML`. Install `tm`, `XML` and `SnowballC` packages to perform this task.

```
# following function remove any html markup in a character object
remove_HTML_markup <- function(s) tryCatch({
  doc <- htmlTreeParse(paste("", s), asText = TRUE, trim = FALSE)
  xmlValue(xmlRoot(doc))
}, error = function(s) s)

# create a corpus (def. A text corpus is a large and unstructured set of texts)
corpus <- Corpus(VectorSource(sapply(JSS_papers[, "description"],
                                     remove_HTML_markup)))
```

4. The corpus is exported to a document-term matrix using function `DocumentTermMatrix()` from package tm. The terms are stemmed and the stop words, punctuation, numbers and terms of length less than 3 are removed using the control argument

```
# Create a Document Term Matrix (def. A document-term matrix is a
# mathematical matrix that describes the frequency of terms that
# occur in a collection of documents.)

Sys.setlocale("LC_COLLATE", "C")
#this is just to make sure we will have the same results

JSS_dtm <-
  DocumentTermMatrix(corpus
                     , control = list(stemming = TRUE
                                      , stopwords = TRUE
                                      , minWordLength = 3
                                      , removeNumbers = TRUE
                                      , removePunctuation = TRUE))
```

5. The mean term frequency-inverse document frequency (tf-idf) over documents containing this term is used to select the vocabulary. This measure allows to omit terms which have low frequency as well as those occurring in many documents. In this step, we need to install the package `stam`.

```
# Use the mean term frequency-inverse document frequency (tf-idf)
# to select the vocabulary
# In this step, we may need to install the package stam.
library("slam")
summary(col_sums(JSS_dtm))
term_tfidf <- tapply(JSS_dtm$v/row_sums(JSS_dtm)[JSS_dtm$i],
                     JSS_dtm$j, mean) * log2(nDocs(JSS_dtm)/col_sums(JSS_dtm > 0))
summary(term_tfidf)

# We only include terms which have a tf-idf value of at least 0.1
# which is a bit less than the median and ensures that the very
# frequent terms are omitted

JSS_dtm <- JSS_dtm[,term_tfidf >= 0.1]
JSS_dtm <- JSS_dtm[row_sums(JSS_dtm) > 0,]
summary(col_sums(JSS_dtm))

dim(JSS_dtm)
```

Now we have 342 documents and 1690 terms

```
> dim(JSS_dtm)
[1]  342 1690
```

II. Fitting the Latent Dirichlet Allocation (LDA) model

In this section, we fit an LDA model with 30 unknown topics to the dataset using the topicmodels package. We need to install the topicmodels package to perform this step.

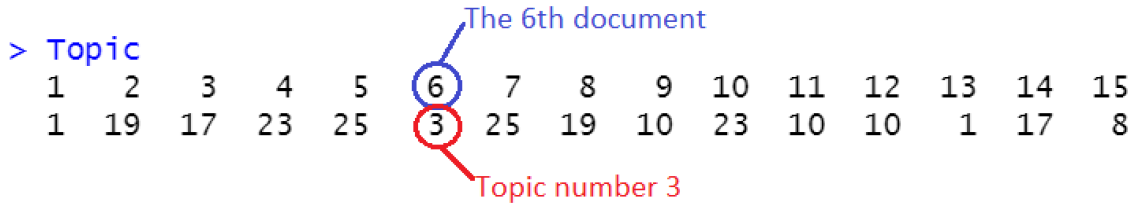
```
#II. Fitting the Latent Dirichlet Allocation (LDA) model
#  fit an LDA model with 30 unknown topics to the dataset using the
#  topicmodels package

if(!require(topicmodels))
  install.packages("topicmodels")
library("topicmodels")
k <- 30
SEED <- 2010
jss_TM <- LDA(JSS_dtm, k = k, method = "Gibbs", control =
                list(seed = SEED, burnin = 1000, thin = 100, iter = 1000))
```

1. The most likely topic for each document is obtained by:

```
#The most likely topic for each document is obtained by:
Topic <- topics(jss_TM,1)
Topic
```

In this case, the topic number 3 is the most likely topic for document number 6 (this can change due to randomness in the process)



2. The five most frequent terms of each topic

```
#The five most frequent terms of each topic
Terms <- terms(jss_TM, 5)
Terms[,1:5] #list the frequent terms of the first 5 topics
```

```
> Terms[,1:5] #list the frequent terms of the first 5 topics
     Topic 1     Topic 2    Topic 3   Topic 4     Topic 5
[1,] "confid"    "tabl"     "control" "lispstat"  "popul"
[2,] "multinomi" "socr"     "excel"   "graph"     "ecolog"
[3,] "survey"    "conting"  "gene"    "loglinear" "speci"
[4,] "probit"    "haplotyp" "chart"   "vista"     "captur"
[5,] "monoton"   "learn"    "draw"    "aspect"    "period"
```