Análise de Dados Textuais

Pedro J. Fernandez, Paulo C. Marques F. e Hedibert F. Lopes

Sábado, 2 de Dezembro de 2017

Insper

Dados textuais

Insper

Nesta aula continuamos a estudar problemas de aprendizagem não supervisionada, mas com um tipo de dado especial: os dados textuais (*text data*), ou dados em forma de textos.

O advento das redes sociais deu certa proeminência a este tipo de dado e sua modelagem. De fato, a incorporação efetiva da informação de dados textuais em análises mais tradicionais é um problema com muito espaço para desenvolvimento e inovação.

Há muitas variantes de dados textuais. Por exemplo:

1. Pequenas mensagens do *Twitter* (*tweets*), com no máximo 240 caracteres.
2. Textos de tamanho intermediário, como avaliações de consumidores sobre livros, restaurantes, filmes etc.
3. Textos mais longos, como críticas profissionais de livros e filmes, editoriais e notícias em jornais ou *sites*, comunicações para acionistas, registros médicos, discursos políticos e decisões judiciais.

Terminologia

Em um conjunto de dados textuais cada unidade de informação é denominada *documento*; e um conjunto de certos documentos é denominado *corpus*.

No tipo de análise que iremos considerar, cada documento é reduzido a um mero conjunto de palavras, sem levar-se em conta a posição relativa das palavras dentro do documento. Tal representação é denominada *sacola-de-palavras* (*bag-of-words*).

De fato, cada documento é representado por uma lista de *termos* e suas respectivas frequências de ocorrência no documento. Tais termos não são formados necessariamente por palavras individuais.

Na preparação de um conjunto de dados textuais, podemos identificar *tokens* (símbolos) que formam termos com mais de uma palavra. Por exemplo, a expressão "muito ruim" pode ser considerada um único *token* (um *bigrama*) formado pelas palavras "muito" e "ruim", tendo um sentido diferente das palavras "muito" e "ruim" consideradas separadamente. Do mesmo modo, podemos tratar "Ministério da Justiça" como um único *token* (um *trigrama*). Um *token* formado por n palavras é denominado *n-grama*.

Dados do Twitter (1)

Insper

Uma boa maneira de explorar os conceitos básicos relacionados a dados textuais é minerar algumas mensagens do *Twitter* utilizando a biblioteca `twitteR` do `R`.

Além de possuir uma conta regular do *Twitter*, para rodar os exemplos a seguir é necessário criar uma conta (gratuita) de desenvolvedor (informações disponíveis com o Prof. Tiago). Nossa conta de exemplo está acessível em:

https://twitter.com/BayesianFactory

O primeiro passo é se conectar e autenticar. Abaixo, por uma questão de privacidade, ocultamos os valores das chaves de acesso. Utilize suas próprias chaves quando reproduzir os exemplos.

```
library(twitteR)
library(ROAuth)

setup_twitter_oauth(consumer_key, consumer_secret, access_token, access_secret)
```

```
## [1] "Using direct authentication"
```

Dados do Twitter (2)

Insper

Uma funcionalidade básica da biblioteca `twitteR` é enviar *Tweets* utilizando a função `tweet()`.

```
tweet("Thomas Bayes!")
```

Também podemos verificar as *Twitter trends* de uma certa localidade, identificada pelo `woeid` ("Where On Earth IDentifier").

Para descobrir o `woeid` da localidade mais próxima a partir de uma latitude e uma longitude (por exemplo, as da Vila Olímpia), utilizamos a função `closestTrendLocations()`:

```
closestTrendLocations(lat = -23.53, long = -46.67)
```

```
##        name country  woeid
## 1 São Paulo  Brazil 455827
```

Dados do Twitter (3)

Insper

Estas são as 10 principais *Twitter trends* de São Paulo:

```
trends <- getTrends(woeid = 455827)

trends$name[1:10]
```

```
##  [1] "#ExNaMTV"              "#ExathlonBrasil"       "WhatsApp"
##  [4] "Lucas Lima"            "#QuintaDetremuraSdv"   "Lanus"
##  [7] "Renato Gaúcho"         "Flamengo"              "#OOutroLadoDoParaiso"
## [10] "Marcos"
```

Dados do Twitter (4)

Insper

Utilizando a função `searchTwitter()` podemos fazer uma busca das mensagens mais recentes escritas em Português que contém a *hashtag* `ReformaDaPrevidencia`.

```
results <- searchTwitter("#ReformaDaPrevidencia", n = 1000, lang = "pt")
tweets <- twListToDF(results)
dim(tweets)
```

```
## [1] 533  16
```

O objeto `tweets` é um `data.frame` com as colunas:

```
names(tweets)
```

```
##  [1] "text"          "favorited"     "favoriteCount" "replyToSN"
##  [5] "created"       "truncated"     "replyToSID"    "id"
##  [9] "replyToUID"    "statusSource"  "screenName"    "retweetCount"
## [13] "isRetweet"     "retweeted"     "longitude"     "latitude"
```

Este é o conteúdo do quinto *tweet*:

```
writeLines(strwrap(tweets$text[5], 80))
```

```
## E aí, a reforma da Previdência sai ou não sai? #reformadaprevidencia
## #crisepolitica #congressonacional https://t.co/LaJSJL6DOH
```

Dados do Twitter (5)

Insper

A partir dos textos dos *tweets* criamos o *corpus* com as funções da biblioteca `tm` (*Text Mining*).

```
library(tm)

txt <- sapply(tweets$text, function(row) iconv(row, "UTF-8", "ASCII//TRANSLIT", sub = ""))

corpus <- Corpus(VectorSource(txt))
```

Antes de examinar o *corpus* criado, iremos tranformá-lo, deixando todas as palavras em caixa baixa e removendo: espaços em branco dobrados, números, palavras irrelevantes, pontuações e *stop words* (artigos, preposições etc).

```
corpus <- tm_map(corpus, content_transformer(tolower))

corpus <- tm_map(corpus, stripWhitespace)

corpus <- tm_map(corpus, removeNumbers)

corpus <- tm_map(corpus, removeWords, c("https", "rt", "via", "reformadaprevidencia", "sobre"))

corpus <- tm_map(corpus, removePunctuation)

corpus <- tm_map(corpus, removeWords, iconv(stopwords("portuguese"), from = "UTF-8", to = "ASCII//TRANSLIT"))
```

Dados do Twitter (6)

Insper

Na fase de preparação do *corpus*, uma transformação importante é o processo de *stemming*, pelo qual palavras derivadas de um radical comum são reduzidas a um único termo.

A própria função `tm_map()` nos permite fazer o *stemming* do *corpus* da seguinte maneira:

```
corpus <- tm_map(corpus, stemDocument, language = "portuguese")
```

No entanto, não ficamos satisfeitos com os resultados deste algoritmo quando aplicado ao nosso idioma. Portanto, na *nuvem de palavras* que vemos a seguir temos o *corpus* antes do processo de *stemming*.

Nesta nuvem de palavras, o tamanho da fonte de um termo é proporcional à sua frequência de ocorrência no *corpus*.

Nuvem de palavras

Insper

```
library(wordcloud)

wordcloud(corpus, max.words = 40, random.order = FALSE, rot.per = 0, colors = brewer.pal(8, "Dark2"))
```

Matriz documento-termo

Insper

O *corpus* pode ser resumido em uma *matriz documento-termo*, cujas linhas e colunas correspondem a documentos e termos, respectivamente, e as entradas da matriz são as frequências observadas. De posse desta matriz, podemos procurar os termos com ocorrência mais frequente no *corpus*, bem como examinar as associações entre os termos.

```
dtm <- DocumentTermMatrix(corpus, control = c(weighting = weightTf))

findFreqTerms(dtm, lowfreq = 40)
```

```
## [1] "previdencia"   "reforma"       "aposentadoria" "governo"
## [5] "temer"         "vai"
```

```
findAssocs(dtm, terms = "temer", corlimit = 0.3)
```

```
## $temer
##      enganosa         ajuda      encontro especialistas     deputados
##          0.38          0.38          0.36          0.36          0.33
##          tcoz          eita        mandou  publicitaria          nela
##          0.33          0.30          0.30          0.30          0.30
```

Um *corpus* da BBC (1)

Insper

Agora vamos analisar um *corpus* de documentos com notícias da BBC. Os documentos estão pré-classificados por tópicos (assuntos). Para examinar o potencial de uma aprendizagem não supervisionada com estes dados, iremos "esquecer" os tópicos originais e tentar recuperá-los.

Dizemos que uma matriz é *esparsa* quando esta possui muitos elementos nulos; tenham estes elementos o valor 0, ou sejam apenas valores ausentes (`NA`). Uma matriz esparsa é representada de maneira que tais valores nulos não sejam armazenados.

O armazenamento dos documentos da BBC está estruturado em torno de uma matriz esparsa, no formato `MatrixMarket`, que conecta as listas de termos e documentos do *corpus*. Para ler esta matriz esparsa utilizamos a função `readMM()` da biblioteca `Matrix`.

```
library(Matrix)

mtx <- readMM("./data/bbc.mtx")
```

Um *corpus* da BBC (2)

Insper

A partir desta matriz esparsa `mtx`, construímos a matriz documento-termo utilizando as funções da biblioteca `tm`.

```
library(tm)

tdm <- as.TermDocumentMatrix(mtx, weighting = weightTf)
dtm <- t(tdm)

dtm$dimnames$Terms <- scan("./data/bbc.terms", what = "character")
dtm$dimnames$Docs <- scan("./data/bbc.docs", what = "character")
```

Um *corpus* da BBC (3)

Insper

Cada linha da matriz `dtm` corresponde a um documento do *corpus* e cada coluna corresponde a um termo. Esta matriz armazena as frequências de ocorrências dos termos nos documentos. Podemos examinar a classe de `dtm` e seus atributos.

```
class(dtm)
```

```
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
```

```
attributes(dtm)
```

```
## $names
## [1] "i"        "j"        "v"        "nrow"     "ncol"     "dimnames"
##
## $class
## [1] "DocumentTermMatrix"    "simple_triplet_matrix"
##
## $weighting
## [1] "term frequency" "tf"
```

Um *corpus* da BBC (4)

Insper

As dimensões de `dtm` quantificam o tamanho do *corpus* que estamos examinando: 2225 documentos e 9635 termos (um *Big Data* legítimo).

Não podemos inspecionar os valores da matriz `dtm` diretamente, devido à forma como esta matriz é armazenada. A biblioteca `tm` fornece as facilidades para acessarmos os elementos de `dtm`. Por exemplo, estes são os primeiros 5 documentos e as frequências de 10 de seus termos.

```
inspect(dtm[1:5, 1:10])
```

```
## <<DocumentTermMatrix (documents: 5, terms: 10)>>
## Non-/sparse entries: 19/31
## Sparsity           : 62%
## Maximal term length: 9
## Weighting          : term frequency (tf)
## Sample             :
##                Terms
## Docs            ad boost giant jump media profit quarterli sale time warner
##   business.001  1     2     1    1     1     10          1    5    3      4
##   business.002  0     1     0    0     0      0          0    0    2      0
##   business.003  0     0     1    0     0      0          0    4    0      0
##   business.004  0     0     0    0     0      4          1    1    0      0
##   business.005  0     0     1    0     0      0          0    0    1      0
```

Um *corpus* da BBC (5)

Insper

Usando as conversões de tipos adequadas, podemos listar todos os termos de um documento, por exemplo, o documento `business.007`.

```
doc_terms <- as.matrix(dtm["business.007",])

print(sort(dtm$dimnames$Terms[doc_terms > 0]), quote = FALSE)
```

```
##  [1] 000        133        146        157        190        2001
##  [7] 2004       52         ad         administr  albeit     amount
## [13] analyst    boost      bush       celebr     chief      clearview
## [19] condit     continu    creat      creation   decemb     depart
## [25] deputi     dollar     econom     economi    economist  elect
## [31] employ     end        environ    expand     expect     fall
## [37] favour     fewer      figur      financi    fine       firm
## [43] first      gain       get        given      got        group
## [49] growth     herbert    interest   issu       januari    job
## [55] kei        ken        labor      level      limit      low
## [61] lowest     margin     market     mayland    mean       moder
## [67] net        novemb     number     offic      opportun   pace
## [73] payrol     posit      presid     presidenti produc     push
## [79] rate       record     rel        result     revis      rick
## [85] satisfi    septemb    slow       strong     suggest    term
## [91] territori  three      unemploy   valu       worker     year
```

Um *corpus* da BBC (6)

Insper

Observando a convenção utilizada para nomear os documentos, podemos extrair os tópicos correspondentes e calcular a quantidade de documentos existentes em cada tópico.

```
document_topic <- sapply(strsplit(rownames(dtm), "[.]"), function(x) x[1])

table(document_topic)
```

```
## document_topic
##      business entertainment      politics         sport          tech
##           510           386           417           511           401
```

Um *corpus* da BBC (7)

Insper

Explorando o *corpus*, podemos procurar, por exemplo, todos os termos que ocorrem mais do que 1300 vezes.

```
print(findFreqTerms(dtm, 1300), quote = FALSE)
```

```
## [1] time  on    year  peopl game
```

Um *corpus* da BBC (8)

Insper

Estes são os 7 termos com ocorrência mais frequente no *corpus*.

```
freq <- colSums(as.matrix(dtm))

print(format(sort(freq, decreasing = TRUE)[1:7], width = 9), quote = FALSE)
```

```
##      year     peopl        on      game      time     first    govern
##      2830      2044      1838      1640      1487      1283      1246
```

Podemos representar graficamente as frequências de ocorrências dos termos neste *corpus* da BBC por uma nuvem de palavras, lembrando que nesta nuvem o tamanho da fonte de cada termo é proporcional à sua frequência de ocorrência no *corpus*.

Um *corpus* da BBC (9)

Insper

```
library(wordcloud)

wordcloud(names(freq), freq, max.words = 40, random.order = FALSE,
          rot.per = 0, colors = brewer.pal(8, "Dark2"), main = "Title")
```

Um *corpus* da BBC (10)

Insper

Para cada tópico, estes são os termos mais frequentes.

```
for (topic in unique(document_topic)) {
    cat(topic, ":\n", sep = "")
    print(format(sort(colSums(as.matrix(dtm)[grepl(topic, rownames(dtm)),]),
                      decreasing = TRUE)[1:7], width = 9), quote = FALSE)
    cat("\n")
}
```

```
## business:
##      year   compani      firm    market      bank      sale     price
##       884       627       557       539       459       414       393
##
## entertainment:
##      film      year      best     music     award      star      show
##       964       594       590       540       522       429       424
##
## politics:
##    labour    govern     parti     elect     peopl     blair    minist
##       760       759       709       670       623       575       565
##
## sport:
##      game      plai       win    player   england   against      year
##       648       624       590       474       459       454       444
##
## tech:
##     peopl      game technolog     mobil     phone        on    servic
##       960       875       631       595       540       519       512
```

Suponha que você tem um *corpus* cuja extensão torna a leitura de todos os documentos proibitiva.

Por exemplo, você tem acesso a milhares de decisões judiciais de ações trabalhistas relacionadas a um certo setor da indústria, ou centenas de milhares de *e-mails* de uma grande corporação envolvida em esquemas de corrupção.

Um *modelo de tópicos probabilístico* é uma ferramenta que permite agrupar os documentos do *corpus* de acordo com os seus conteúdos. Ou seja, criar clusters dos documentos levando em conta suas similaridades e dissimilaridades.

A ideia é que há um certo número de *tópicos*, que são variáveis aleatórias latentes (não observadas), e que as palavras dos documentos têm uma distribuição de probabilidades que depende do tópico específico do documento ao qual ela pertence.

Modelo de Tópicos (1)

Insper

O objetivo desta modelagem é obter a probabilidade *a posteriori* dos tópicos de cada documento, dados os conteúdos de todos os documentos do *corpus*.

O modelo probabilístico dominante na área de modelagem de tópicos é a *Alocação Dirichlet Latente* (*Latent Dirichlet Allocation*, ou LDA) de Blei, Ng e Jordan (paper disponível no Blackboard).

Ao final do treinamento do modelo LDA, temos as palavras mais prováveis de cada tópico, bem como o tópico mais provável de cada documento do *corpus*.

Modelo de Tópicos (2)

Insper

Suponha que "perdemos" a informação sobre os tópicos originais a que pertencem os documentos deste *corpus* da BBC.

Utilizando a biblioteca `topicmodels`, vamos treinar um modelo LDA com 5 tópicos.

```
library(topicmodels)

model <- LDA(dtm, k = 5, method = "Gibbs",
             control = list(seed = 4321, burnin = 250, thin = 2, iter = 1000))
```

Resultados (1)

Insper

Estes são os 10 termos com maior probabilidade de ocorrência em cada tópico formado pelo modelo LDA.

```
print(terms(model, 10), quote = FALSE)
```

```
##       Topic 1 Topic 2 Topic 3   Topic 4 Topic 5
##  [1,] year    film    peopl     plai    govern
##  [2,] compani year    game      game    labour
##  [3,] market  best    servic    win     parti
##  [4,] sale    show    technolog england elect
##  [5,] firm    award   mobil     against peopl
##  [6,] share   includ  phone     first   plan
##  [7,] expect  on      on        back    minist
##  [8,] bank    music   get       player  sai
##  [9,] month   top     work      two     told
## [10,] price   star    user      time    blair
```

Resultados (2)

Insper

Estes são os tópicos mais prováveis para os cinco primeiros documentos do *corpus*.

```
topics(model, 3)[, 1:5]
```

```
##      business.001 business.002 business.003 business.004 business.005
## [1,]            1            1            1            1            1
## [2,]            3            5            5            4            5
## [3,]            2            3            4            3            2
```

Quantos documentos foram alocados corretamente nos tópicos originais?

Resultados (3)

Insper

```
true_topic <- as.factor(document_topic)
names(true_topic) <- rownames(dtm)

predicted_topic <- factor(topics(model),
                          levels = c(1, 2, 5, 4, 3),
                          labels = c("business", "entertainment", "politics", "sport", "tech"),
                          ordered = TRUE)

confusion <- table(predicted_topic, true_topic, dnn = c("Predicted Topic", "True Topic"))
print(confusion)
```

```
##                 True Topic
## Predicted Topic business entertainment politics sport tech
##   business           475             4        9     2    7
##   entertainment        3           363        5     2   12
##   politics            24            14      400     1    9
##   sport                0             0        2   506    3
##   tech                 8             5        1     0  370
```

```
(sprintf("Correct alocation: %.2f%%\n", 100 * (sum(diag(confusion)) / sum(confusion))))
```

```
## [1] "Correct alocation: 95.01%\n"
```

Quantos tópicos? (1)

Insper

Em qualquer análise de clusters, descobrir o número adequado de clusters é sempre uma questão delicada.

Em geral, é necessário examinar os clusters obtidos para decidir se o número é adequado.

Em certos problemas, o número de clusters é definido por questões de natureza prática: por exemplo, o número de especialistas que irão analisar os documentos dos clusters criados.

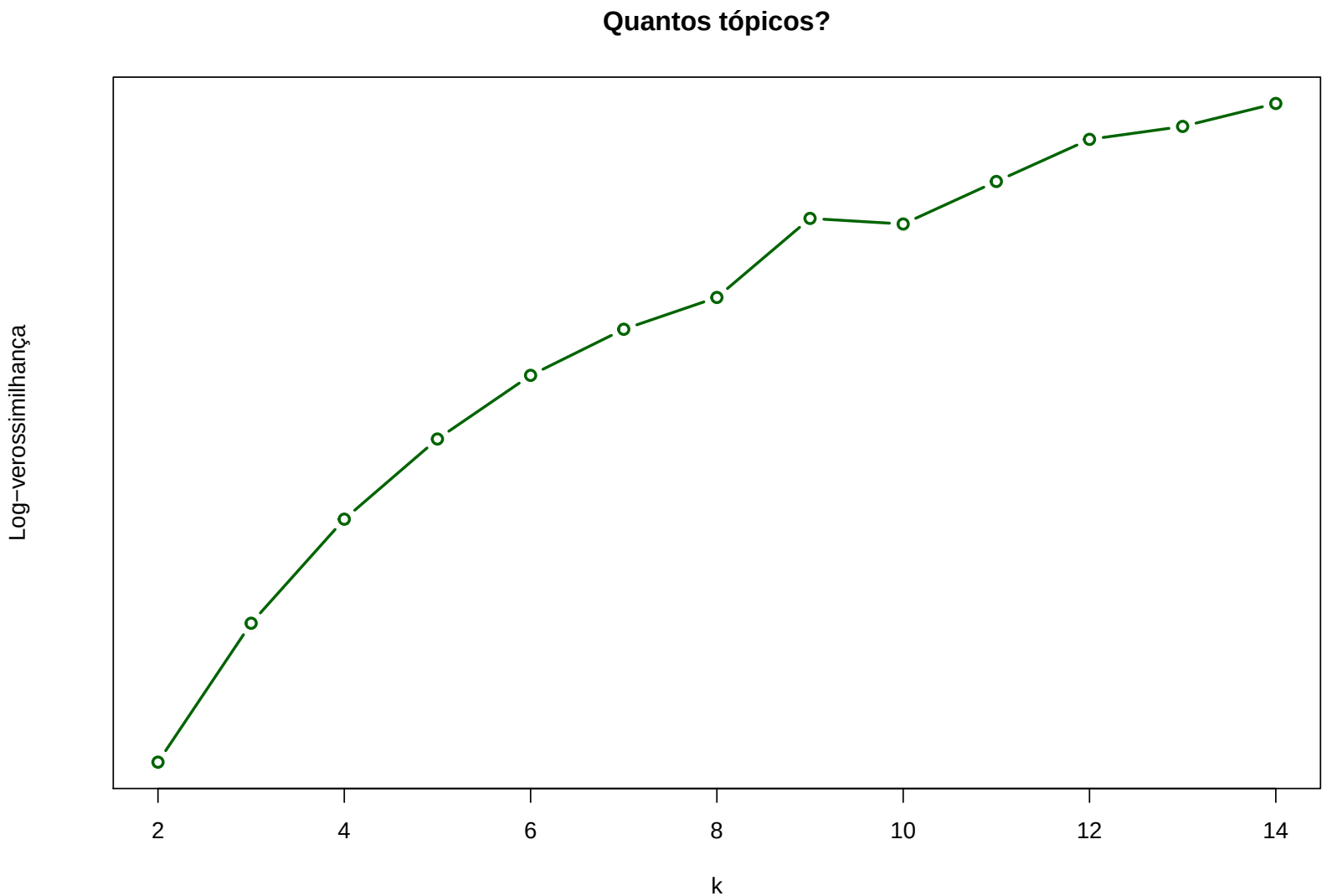
Uma técnica comum que ampara a decisão sobre o número de cluster é treinar vários modelos LDA com números de tópicos distintos e procurar um "cotovelo" na curva definida pelas log-verossimilhanças dos modelos LDA treinados.

```
k_range <- 2:14

log_L <- numeric()

for (k in k_range) {
    mdl <- LDA(dtm, k = k, method = "Gibbs", control = list(seed = 4321, burnin = 250, thin = 2, iter = 1000))
    log_L <- c(log_L, mdl@loglikelihood)
}
```

Quantos tópicos? (2)

Insper

```
plot(k_range, log_L, type = "b", lwd = 2, col = "dark green", yaxt = "n",
     xlab = "k", ylab = "Log-verossimilhança", main = "Quantos tópicos?")
```



Créditos

O material desta aula é baseado no capítulo de *Topic Models* escrito originalmente pelo professor Pedro J. Fernandez.

Agradecemos ao professor Pedro por sua gentileza e pela parceria na elaboração deste material.