Faculdade de Economia, Administração e Contabilidade

Análise de Dados

Laboratório de Análise de Textos

(dados não-estruturados)

Frequencia e Agrupamento

Linguagem R e RStudio

Prof. Antonio Geraldo da Rocha Vidal

2021

Sumário

Você deve entregar um relatório com os resultados das etapas elaboradas neste laboratório no **e-Disciplinas**, para formatá-lo siga estas orientações:

1. Crie um documento Word e identifique-o com o nome do laboratório, data de elaboração e o seu nome ou do grupo que o elaborou;
2. Crie um tópico para cada resultado que você considerar relevante (manipulação de dados ou resultado de algum processamento) identificando-o com um título e uma breve explicação. Os resultados podem ser imagens de gráficos gerados ou de listas de valores ou dados de resultados obtidos. Não devem ser incluídos os *scripts* ou instruções de processamento utilizados, inclua apenas os resultados que você considerar relevantes.
3. No final do relatório crie um último tópico denominado “Conclusões” e elabore comentários, sugestões e conclusões sobre o que você pode aprender com a elaboração deste laboratório.

Esta apostila introdutória pode conter erros, falhas ou imprecisões. Se você identificar algum problema por favor informe através do e-mail vidal@usp.br para que a correção possa ser providenciada.

Esta apostila não é autoral, tem objetivo estritamente didático como material de apoio para a disciplina. Foi desenvolvida através da compilação dos diversos textos e materiais citados na bibliografia.


Obrigado!

Introdução

Para elaborar este laboratório estamos supondo que você já esteja familiarizado com a utilização básica da Linguagem R e do ambiente de análise RStudio.

Exemplos típicos de aplicações que envolvem análise de textos são o reconhecimento de *spam* em caixas de mensagens, análise de sentimento em redes sociais, o desenvolvimento de sistemas de recomendação e a classificação ou categorização de documentos de qualquer natureza. Outras técnicas de análise de dados também se aplicam à análise de textos, como o agrupamento e a descoberta de regras de associação.

Preparação de Dados Não Estruturados Textuais

Introdução

Enquanto dados estruturados estão organizados e formatados de forma adequada para serem submetidos a análise exploratória de dados e posteriormente passar por procedimentos analíticos, dados não estruturados textuais necessitam obrigatoriamente de uma preparação prévia.

Em base de dados textuais, também conhecidas como ***corpus*** ou ***corpora***, cada exemplar de texto é tratado como um documento. Cada documento em um ***corpus*** pode assumir diferentes características em relação a, por exemplo, tamanho do texto, tipo de conteúdo, idioma e estilo de linguagem (formal, coloquial, poética, técnica etc.).

A transformação de um *corpus* em um conjunto de dados que pode ser submetido à procedimentos de análise é um processo que gera uma representação capaz de descrever cada documento em função de suas características. A lista de todas as palavras que ocorrem em todos os documentos de um *corpus* pode ser chamada de dicionário. Com base no dicionário e na frequência com que as palavras do dicionário aparecem nos documentos e no *corpus* é possível construir uma representação de dados estruturados para o *corpus* viabilizando a utilização de técnicas analíticas de dados.

Um *corpus* já preparado como um conjunto de dados estruturado é definido como um conjunto de *n*-documentos (doc), ou seja, *corpus* = {doc_1, doc_2, ..., doc_n}. Cada um dos documentos (doc) é definido como um conjunto de *d*-termos (radicais, palavras ou conjunto de palavras), de forma que doc_i = (wte_{i1}, wte_{i2}, wte_{i3}, ..., wte_{ij}, ..., wte_{id}), com i = {1, ..., n}. Cada termo wte_{ij} assume os valores 1 ou 0, onde o valor 1 significa que o termo está presente no documento e o valor 0 significa que o termo não está presente. O wte_{ij} também pode assumir valores reais sendo que o valor assumido representa um peso (muitas vezes a frequência de sua ocorrência) que relaciona o termo com o documento, ou relaciona o termo com o documento e o *corpus*. Cada documento é visto como um exemplar ou observação do conjunto de dados, e cada termo é visto como um atributo, dado ou variável descritiva do documento.

Um exemplo de um *corpus* com dez "documentos" é apresentado no quadro a seguir.

Doc1	Mineração de dados é a análise de conjuntos de dados observacionais (sempre grandes) para encontrar relações não explícitas e para resumir os dados em novas formas que são compreensíveis e úteis para o analista de dados
Doc2	Mineração de dados é a descoberta de modelos para aumentar a utilidade dos dados

Doc3	Análise de dados desempenha um papel essencial no processo de descoberta de conhecimento.
Doc4	Na abordagem estatística, análise de dados é a inferência de modelos.
Doc5	Na abordagem de computação, análise de dados é uma forma extrema de processamento analítico, traduzido por consultas especiais que processam grandes quantidades de dados.
Doc6	O gerenciamento e a análise de dados fornecem a infraestrutura para transformar dados brutos em informações de alta qualidade e obter conhecimento.
Doc7	As técnicas de análise de dados são usadas para descobrir relações, padrões e associações existentes entre os dados que descrevem objetos de interesse do negócio (do pesquisador).
Doc8	Os algoritmos analisam um conjunto de dados procurando padrões, regras, anomalias ou tendências de comportamento utilizando técnicas estatísticas e computacionais avançadas.
Doc9	O valor dos modelos e algoritmos de análise de dados é muito influenciado pela visualização de seus resultados.
Doc10	Um processo para obtenção, inspeção, limpeza, transformação e modelagem de dados com o objetivo de descobrir informações úteis, sugerindo conclusões para apoiar a tomada de decisão e o suporte e otimização de ações.

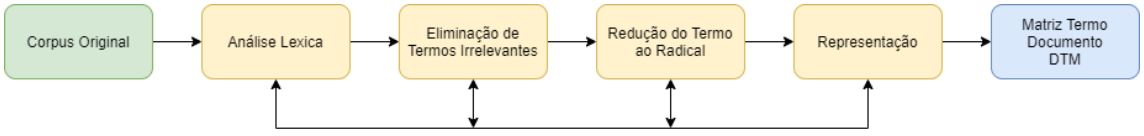
Uma possível aplicação de análise de dados nesses documentos poderia ser a o agrupamento para a descoberta de categorias e perfis de conteúdo. Entretanto, as características referentes ao estilo de escrita, quantidade de caracteres, uso de sinônimos etc., usadas em diferentes textos não seguem uma estrutura comum que permita diretamente avaliar a similaridade entre documentos.

Como o objetivo aqui não é estudar e automatizar a análise de estruturas gramaticais nos documentos, o que se tornaria uma tarefa específica para a área de processamento de linguagem natural, é preciso preparar esses dados textuais de forma a adequá-los às técnicas de análise de dados que estamos estudando, como classificação, agrupamento e associação. Desta forma, é necessário transformar esse *corpus* num conjunto de dados descritos pelos **termos** contidos em cada documento, conforme já definido, isto é, numa matriz de termos de documentos ou DTM (*Document Term Matrix*).

Processo para Geração da Matriz Termo Documento (DTM)

Para transformar o *corpus* original em uma representação que defina o conjunto de dados de uma matriz termo documento ou DTM, uma série de procedimentos deve ser executada, podendo ser entendida como um processo de preparação dos dados.

É importante ressaltar que cada problema de análise de texto está dentro de determinado contexto, domínio ou área de aplicação, e, por isso, informações e conhecimentos inerentes ao domínio ou área de aplicação em estudo devem sempre ser considerados em todas as tomadas de decisão no processo de preparação de dados do *corpus*.



1. **Análise Léxica**: tem como objetivo gerar uma lista de *tokens,* ou seja, a primeira lista de termos, que ao final do processo será modificada e reduzida. Essa fase recebe os documentos do *corpus* original, os converte em uma sequência de caracteres e então procede a geração de *tokens* ou termos. Essa geração ocorre a partir da eliminação de caracteres de pontuação, eliminação de dígitos e até eliminação de caracteres acentuados. Os tokens podem ser gerados a partir da definição de um (ou mais) separadores, como o espaço em branco. Ainda nesta fase, a capitalização de letras pode ser alterada, de forma a que todos os termos sejam transformados para letras maiúsculas ou minúsculas.
2. **Eliminação de Termos Irrelevantes**: um texto contém artigos definidos e indefinidos, preposições, pronomes, numerais, conjunções e advérbios. Palavras que pertencem a essas classes gramaticais são frequentemente irrelevantes em um processo de análise de textos no qual se deseja descobrir padrões, pois não carregam sozinhas um sentido semântico e são, geralmente, muito frequentes em qualquer tipo de texto. Por isso, tais palavras devem fazer parte de uma lista de termos irrelevantes, também conhecida como ***stopwords***, e devem ser removidas dos textos. A remoção de *stopwords* pode reduzir o tamanho de um documento em 30% a 50%, o que é muito importante para reduzir a complexidade do espaço ou dimensionalidade do conjunto de dados submetido à análise. A lista de stopwords, pode ser modificada (ampliada ou reduzida) de acordo com o contexto dos conteúdos dos textos ou de acordo com a tarefa de análise de textos pretendida. Após essa fase, cada documento passa a ser uma lista de palavras formada por termos com maior representatividade no contexto da análise e, portanto, de maior poder discriminatório. É importante destacar que após a execução das duas primeiras fases do processo, os textos dos documentos assumem uma representação que se distancia da linguagem natural, perdendo, em geral, o seu significado semântico.
3. **Redução do termo ao seu radical**: um termo pode sofrer variações, como plural, gerúndio, verbos flexionados, aumentativo, diminutivo etc. Na análise de textos, termos com o mesmo radical (*stem*) não devem ser tratados como diferentes. Assim, neste processo conhecido por ***stemming*** os prefixos ou sufixos são eliminados, uniformizando os termos que possuem o mesmo radical. Para a redução do termo ao seu radical, algoritmos vêm sendo propostos, mas além de dependerem do idioma do texto, podem apresentar várias imprecisões. Em particular, na língua portuguesa, o problema de redução dos termos ao seu radical assume uma complexidade maior do que em línguas como o inglês devido ao grande número de formas para o plural e para flexões verbais, além das diferentes regras para tratamento de aumentativo, diminutivo, masculino e feminino.
4. **Representação**: nesta fase realiza-se a geração do conjunto de dados por meio da construção de uma representação vetorial para cada documento, ou matricial para todos os documentos, onde cada documento passa a ser uma linha da matriz e cada termo uma coluna. Trata-se, na realidade, de um mapeamento das relações entre termos, documentos e *corpus* para dados numéricos. Em termos genéricos, pode-se dizer que é a atribuição de pesos a cada termo presente no dicionário de termos. Um tipo de representação simplificada é obtido atribuindo valores binários que representam a presença ou ausência do termo em um documento. Nesse caso, cada documento do *corpus* original passa a ser representado por um vetor de zeros (ausência do termo) e uns (presença do termo), considerando todos os termos do dicionário de termos. Note que o número de vezes que o termo aparece no documento não é

considerado na representação binária. Porém a frequência de cada termo ou simplesmente **tf** (do inglês, *term frequency*) pode também ser uma forma de representação na qual números inteiros, representando a frequência do termo no documento, podem aparecer nas coordenadas dos vetores de documentos.

Discriminação de Conteúdo

Tanto a representação binária quanto a representação por frequência do termo por documento podem esconder padrões existentes entre a presença das palavras nos documentos e no *corpus*, simplificando artificialmente a representação do *corpus* como um todo. Por exemplo, se um termo **X** é muito frequente em alguns documentos, porém raro no *corpus*, ele deveria ter um peso maior na representação de tais documentos que um termo **A**, frequente nos documentos e também no *corpus*. Neste caso, o termo **X** é uma característica que discrimina um subconjunto de documentos do conjunto total de documentos do *corpus*. Para obter uma medida que pondere melhor os termos dos documentos de forma representativa, é preciso relacionar a frequência dos termos no documento com a frequência dos termos no *corpus*.

A medida de **frequência inversa nos documentos**, ou simplesmente ***idf*** (do inglês, *inverse document frequency*), pode fornecer essa relação da seguinte maneira:

$$idf(te) = \log\left(\frac{n}{nt}\right)$$

Sendo **n** o número total de documentos no *corpus*, e **nt** o número de documentos em que o termo **te** aparece. Observe que essa medida prioriza termos que aparecem em poucos documentos no *corpus*, indicando que existem termos com frequência invertida maior que possuem poder de discriminação mais alto do que termos com frequência invertida menor.

Assim, justifica-se o estabelecimento de uma medida que combine dois fatores importantes: a frequência do termo dentro de um documento, **tf(doc$_i$, te$_j$)**, e a frequência inversa dos termos nos documentos do *corpus* que o contém, **idf(te$_j$)**. Essa medida, conhecida como **tf-idf**, é capaz de estabelecer uma ponderação aos termos presentes em um documento, considerando tanto o próprio documento quanto o seu relacionamento com o *corpus* é dada por:

$$tf - idf\left(doc_i, te_j\right) = tf\left(doc_i, te_j\right) * idf(te_j)$$

Embora a combinação ***tf-idf*** crie boas representações para textos, ainda é preciso considerar que os documentos em um *corpus* podem ter tamanhos diferenciados. Para corrigir distorções, é comum a realização de um procedimento de normalização, dando origem à medida ***tf-idf normalizada***. Para isso, as seguintes equações podem ser aplicadas:

$$tf - idf_{norm}(\mathrm{doc}_i, \mathrm{te}_j) = \frac{tf\left(doc_i, te_j\right) * idf(te_j)}{\sqrt{\sum_{j=1}^{m} tf(doc_i, te_k)^2 * idf(te_k)^2}}$$

ou alternativamente,

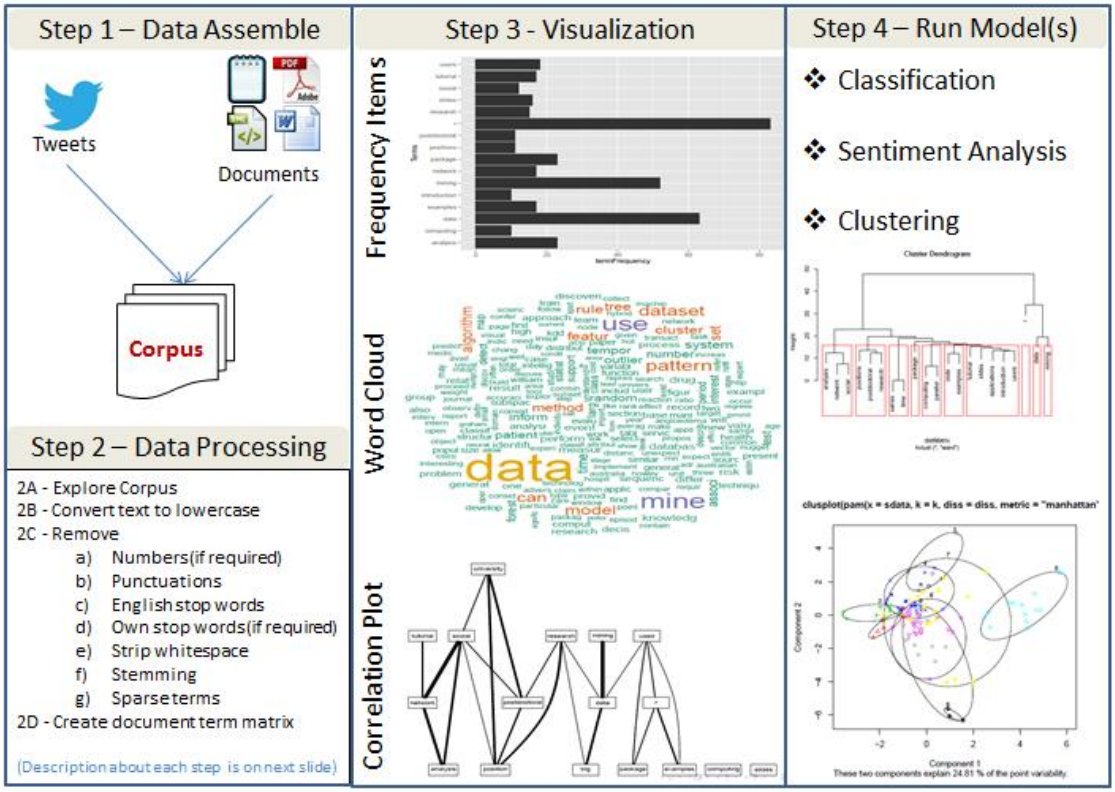
$$tf - idf_{norm}(\mathrm{doc}_i, \mathrm{te}_j) = \frac{tf\left(doc_i, te_j\right) * idf(te_j)}{\sum_{j=1}^{m} tf(doc_i, te_k)}$$

Os quatro tipos de representação vetorial dos documentos e matricial do *corpus* apresentados, permitem que algoritmos para classificação e agrupamento possam ser aplicados.

1. Matriz DTM *tf* binária
2. Matriz DTM *tf* frequência
3. Matriz DTM *tf-idf*
4. Matriz DTM *tf-idf* normalizada

Análise Básica de Textos com R

Introdução ao pacote tm



A implementação da análise de textos na linguagem R é feita com a utilização de funções disponibilizadas no pacote **tm** e, assim é necessário que ele esteja disponível no ambiente R. As instruções para instalação do pacote e para carregamento dos dados dos documentos são:

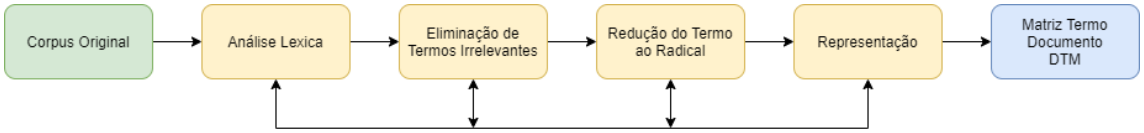
```
install.packages(tm)
library(tm)
corpus <- Corpus(df_textos)
```

A função Corpus() coloca os textos contidos em um dataframe em uma estrutura de dados adequada para que outras funções do pacote **tm** sejam aplicadas. A sintaxe e os parâmetros para uso da função **tm_map()** são:

```
corpus_tf <- tm_map(corpus, operação)
```

- **corpus** é corresponde ao conjunto de textos a ser analisado. Eles podem estar no formato vetorial ou podem ser um corpus de alguma execução precedente.

- **operação** é a indicação de qual processamento deve ser executado sobre os textos – pode ser qualquer um dos processamentos envolvidos nas fases de pré-processamento dos textos conforme ilustrado na figura a seguir.
- **corpus_tf** representa o corpus no formato vetorial ou matricial DTM (documento x termo x frequência).



Para a análise léxica, devem ser consideradas as operações de capitalização, remoção de pontuação, remoção de espaços em branco e remoção de números. Essas ações podem ser executadas conforme exemplificado a seguir. Utilize-se a função **inspect()** para visualizar o conteúdo produzido após a execução de cada operação.

```
corpus tf <- tm map(corpus, contente transformer(tolower))
corpus_tf <- tm_map(corpus_tf, removePunctuation)
corpus_tf <- tm_map(corpus_tf, removeNumbers)
```

Note que a matriz corpus_tf é utilizada como entrada para cada operação. Em muitos casos é prudente manter uma cópia inalterada da matriz corpus_tf antes de cada operação visando facilitar um retorno à operação anterior, caso necessário.

A remoção automática de termos irrelevantes é implementada a partir da escolha de uma lista de palavras (*stopwords*) disponíveis no pacote **tm**, entre elas a lista de ***stopwords*** em português é exemplificada a seguir. Note que a lista de *stopwords* em português pode não estar completa ou ser adequada para os objetivos da análise e, por esse motivo pode ser revisada e substituída ou complementada:

```
corpus_TM <- tm_map(corpus_TM, removeWords, stopwords("portuguese"))
```

É comum ser necessário adicionar novas palavras específicas à lista de *stopwords*. Neste caso a chamada da função **tm_map()** pode incluir um vetor com as novas palavras. Um exemplo de como usar esse recurso é apresentado a seguir:

```
corpus_TM <- tm_map(corpus_TM, removeWords, stopwords("portuguese"), c(palavra01,
palavra02,...,palavraNN)
```

A próxima etapa é a redução dos termos aos seus radicais, conforme exemplificado abaixo. A operação de redução (*steamming*) pode não ser perfeita para a textos em língua portuguesa pois foi originalmente desenvolvida para a língua inglesa. Recomendamos verificar a existência de configurações específicas para a língua portuguesa na versão do pacote **tm** utilizada. Note que por esse motivo geramos o corpus_tfs, de forma que você possa verificar o resultado sem perder o corpus_tf anterior.

```
corpus_tfs <- (corpus_tf, stemDocument)
```

Neste ponto o corpus já está preparado para ser transformado em uma representação vetorial para análise. Para construir uma representação baseada na frequência dos termos nos documentos, a função **TermDocumentMatrix()** deve ser utilizada conforme a sintaxe e parâmetros apresentados a seguir.

```
corpus_tf <- TermDocumentMatrix(corpus_tf, filtro)
```

- **corpus** corresponde ao conjunto de dados textuais a serem analisados.

- **filtro** permite selecionar termos com número mínimo de caracteres e/ou com frequência mínima.
- **TermDocumentMatrix** retorna uma matriz de frequência de termos.

```
corpus_tf <- TermDocumentMatrix(corpus_tf, control = list(minWordLengh = 1, minDocFreq =
1))
```

No pacote **tm** da linguagem R, a métrica ***tf – idf*** (frequência do termo – frequência invertida do termo) é obtida através a com a função **weightTfIdf()** que deve ser aplicada sobre a matriz termo documento gerada pela função **TermDocumentMatrix()**.

```
corpus_tf_idf <- weightTfIdf(corpus_tf, tipo)
```

- **corpus_tf** é a matriz de frequências de termos do corpus.
- **tipo** permite escolher entre gerar uma medida normalizada (TRUE) ou não (FALSE).

  corpus_tf_idf <- weightTfIdf(corpus_tf, normalize=TRUE)

A partir dos conjuntos de dados representados pelas matrizes corpus_tf e corpus_tf_idf, diversos algoritmos de análise de dados podem ser aplicados para a obtenção de diferentes tipos de análise, como classificação ou agrupamento. Além disso, outros procedimentos mais simples de análise também podem ser aplicados, como estatísticas simples, visualizações gráficas de frequência, visualizações de palavras mais ou menos frequentes (nuvem de palavras) etc.

A partir dos resultados da análise pode-se inferir conclusões a respeito dos termos e conteúdo dos respectivos documentos, como classes, grupos, temas, assuntos mais abordados etc., de acordo com o contexto e os objetivos do pesquisador.

Tarefa 1: Preparando o R

Para começar este laboratório, instale no seu equipamento os pacotes que você precisará para analisar textos utilizando o R. Você só precisa fazer este passo uma vez.

Atenção: neste laboratório utilizamos a versão R 3.4.1 de 64 bits, pode ser que nem todos os pacotes utilizados estejam disponíveis para outras versões anteriores.

```
Needed <- c("tm", "SnowballC", "RColorBrewer", "ggplot2", "wordcloud", "biclust",
     "cluster", "igraph", "fpc")
install.packages(Needed, dependencies = TRUE)

install.packages("Rcampdf", repos = "http://datacube.wu.ac.at/", type = "source")
```

Se você receber a seguinte mensagem: 'Atualizar todos / alguns / nenhum? [a / s / n]: 'digite "a" e pressione [Enter].

Tarefa 2: Carregando Textos

Comece por salvar seus arquivos de texto em uma pasta intitulada, por exemplo, "textos" Ela conterá o "**corpus**", ou seja, o corpo dos textos que você extrairá para análise.

Nota: Os textos usados neste laboratório exemplo são alguns de artigos do SEMEAD 2017 que foram copiados e colados em um documento de texto. Você pode usar uma variedade de mídias para isso, como textos em formato TXT, PDF ou HTML. O exemplo de textos para este laboratório foi escolhido apenas por curiosidade e como introdução à análise de textos.

Por favor, leia atentamente essa próxima parte, pois você precisa fazer três coisas aqui:

1. Crie uma pasta chamada "textos" onde você manterá seus dados textuais, ou documentos de textos.
2. Salve os arquivos que contém os textos a serem analisados nesta pasta.
3. Para isso, copie e cole os scripts apropriados conforme abaixo.

Use o seguinte trecho de código para carregar os textos a serem analisados no R.

```
textos <- DirSource(directory = "C:/TEXTOS/", encoding = "UTF-8")
textos
```

Carregue o pacote R para análise de textos e depois carregue seus textos no seu projeto R.

```
library(tm)
## Carregando o pacote requerido: NLP
docs <- VCorpus(x=textos)
summary(docs)
```

Para obter detalhes sobre documentos no corpus, use o comando `inspect(docs)`.

```
inspect(docs[1])
## <<VCorpus>>
## Metadata:  corpus specific: 0, document level (indexed): 0
## Content:  documents: 1
##
## [[1]]
## <<PlainTextDocument>>
## Metadata:  7
## Content:  chars: 6215
```

Nesse caso, você está obtendo os detalhes apenas do primeiro documento no *corpus*. Mas isso não é muita informação. Essencialmente, tudo que você obtém é o número de caracteres no corpo de cada documento. Os documentos são identificados pelo número em que são carregados.

Se desejar, você pode ler seus documentos no RStudio usando a instrução `writeLines(as.character(docs))`. Ou, se você preferir ver apenas um dos documentos que você carregou, então pode especificar qual deles deseja usando algo como:

```
writeLines(as.character(docs[2]))
list(list(content = c("A existência de produtos que satisfazem as mais singulares
necessidades torna os atos de comprar, usar, armazenar e descartar tão naturais que
passam despercebidos no dia a dia. Cada item consumido pela sociedade demanda o emprego
de algum recurso para ser produzido e, fatalmente, deixará algum resíduo ao ser
rejeitado. Entretanto, vivemos em um planeta de recursos finitos, e, desta forma, no
ritmo acelerado de consumo em que estamos inseridos, o planeta pode vir a não suportar a
produção de bens e as sobras geradas. Diversas iniciativas têm questionado o sistema
atual de comercialização, trazendo à tona a preocupação com as relações sociais e o meio
ambiente. Um exemplo é o consumo de brinquedo infantil.  Segundo Linn (2006, apud
SANTOS; GROSSI, 2014) o consumo infantil fatura cerca de quinze bilhões de dólares por
ano e o poder de persuasão das crianças nas compras dos adultos aproxima-se de 600
bilhões de dólares. Uma pesquisa realizada pelo Serviço de Proteção ao Crédito (SPC
Brasil) em 2015 constatou que uma em cada cinco crianças toma a decisão final na compra
de brinquedos e jogos; seu grau de influência chega a 7 em uma escala de zero a dez. A
```

```
pesquisa aponta ainda que 30% dos pais percebem a insaciedade dos filhos em relação às
compras e, mesmo comprando boa parte dos itens que as crianças almejam, elas permanecem
insatisfeitas, desejando novos produtos. A evolução humana, o progresso tecnológico e o
interesse publicitário transformaram o ato de consumir; antes o consumo era para
subsistência, depois passou a abarcar necessidades secundárias (MAURER et al., 2015).
Com a produção de bens cada vez mais descartáveis, atrelado ao poder do marketing, a
sociedade tornou o ato de consumir um ritual que satisfaz distintas necessidades. A
cultura do consumo é estimulada pela propaganda, na qual o foco das vendas deixa de ser
o produto e passa a destacar os sentimentos que a pessoa vai experimentar com a
aquisição daquele bem. Desse modo, uma pessoa compra um carro pensando no sentimento de
liberdade que ele vai lhe proporcionar, ou compra uma roupa nova para suprir sua falta
de autoestima. Botsman e Rogers (2011) afirmam que só nos últimos 50 anos já consumimos
mais bens e serviços do que em todas as gerações anteriores reunidas, e que o mecanismo
de consumo e descarte está cada vez mais acelerado. Como os recursos dos quais dispomos
para produzir bens e serviços são finitos, se faz urgente e necessário achar um modelo
mais sustentável. O ato de consumir em excesso não é inerente ao ser humano, é um hábito
adquirido através da convivência em sociedade. Exemplo disto é o consumo de brinquedos,
que evidencia o comportamento de consumo excessivo no sistema em que estamos vivendo.
Desde pequenas as crianças são estimuladas a desejar produtos que não precisam como
presente de Natal ou aniversário. Mesmo que nada necessitem, as crianças são encorajadas
a pedir alguma coisa, condicionando sua realização e bem-estar ao recebimento do
presente pedido. Não raro acabam frustradas por não terem recebido o modelo sonhado, a
quantidade solicitada ou a marca desejada, tornando estressante a experiência. O
consumismo infantil é uma realidade mundial que vem sendo cada vez mais discutida pela
sociedade. Segundo o relatório da Associação Brasileira dos Fabricantes de Brinquedos
(ABRINQ) de 2017, no Brasil, o faturamento com a venda de brinquedos aumentou quase 7%
em 2016, chegando a movimentar mais de seis bilhões de reais. Uma pesquisa divulgada
pelo SPC Brasil apurou que 64% das mães entrevistadas adquirem produtos desnecessários
quando solicitado pelos filhos e metade das compras de brinquedos, jogos, roupas e
calçados são realizadas por impulso. Romper com este ciclo requer reflexão e força de
vontade para modificar a forma como encaramos a posse de bens. Para Botsman e Rogers
(2011, p. 83), deter a propriedade de um produto ou serviço não é essencial, o que ",
" ", "realmente importa é o benefício que ele produz. Ter a possibilidade de usufruir
das vantagens proporcionadas pelo uso de um equipamento, sem a necessidade de comprá-lo,
se torna interessante do ponto de vista econômico, ambiental e social. Nesse contexto
verifica-se uma conjuntura favorável para o mercado de locação de produtos,
principalmente aqueles relacionados à primeira infância. Para cada fase do
desenvolvimento infantil existem diferentes tipos de brinquedo para estimular e entreter
os pequenos apropriadamente. É comum que, neste processo, a criança rapidamente perca o
interesse em um item, pois passou para um novo estágio de crescimento e, para manter as
crianças curiosas e interessadas, os pais acabam comprando novos artigos. A consequência
de alimentar esse ciclo é o acúmulo de produtos não usados. Esse processo não é
sustentável, ocupa espaço e custa muito dinheiro. Botsman e Rogers (2011) apresentaram
um modelo de negócio baseado no aluguel de brinquedos para crianças. No exemplo, os
pais, mediante uma assinatura, alugam brinquedos por um determinado período e depois
trocam por outro diferente quando aquele item se tornar dispensável para a criança.
Existem diversas empresas que oferecem esse serviço ao redor do mundo e no Brasil, desde
2010, esta atividade já é oferecida por alguns estabelecimentos.  Assim, ao invés dos
produtos ficarem esquecidos em uma residência, eles podem rodar pela casa de outras
crianças que poderão aproveitá-lo, reduzindo a necessidade de compra dessas famílias. O
efeito benéfico pode ser ainda mais profundo, pois "à medida que estes canais de
distribuição se tornarem integrados com os próprios fabricantes de produtos, será do
interesse deles tornar seus produtos mais duráveis a fim de lidar com múltiplos usuários
e uso intensivo" (BOTSMAN; ROGERS, 2011, p. 88). Adicionalmente, esse modelo de negócio
possibilita que as crianças tenham contato com diferentes brinquedos adequados para cada
fase de desenvolvimento. Com a não compra, os pais economizam dinheiro, espaço e ainda
auxiliam na redução de geração de resíduos. Além disto, a criança passa a ter contato
com conceitos relacionados à sustentabilidade e consumo compartilhado, contribuindo para
```

```
que desde pequenos compreendam que a diversão não está na posse do brinquedo, mas no seu
uso.  Neste sentido, o estabelecimento de um negócio neste ramo torna-se atraente.  ",
" ", "A temática deste artigo surgiu da percepção da problemática do alto custo de
aquisição de produtos para a primeira infância quando comparados ao seu tempo de uso. Na
pesquisa por alternativas de consumo, foi localizada uma empresa paulistana que loca
desde bombas para aleitamento, cadeiras de transporte infantil até brinquedos,
entretanto, com atendimento restrito a cidade de São Paulo. Ao continuar a busca por
empresas do ramo no Rio Grande do Sul, localizou-se duas empresas que trabalham com a
locação de brinquedo infantil em Porto Alegre. Foi neste contexto, considerando a
natureza volátil do interesse das crianças, que surgiu a ideia de modelar uma empresa
para locação de brinquedos para crianças até seis anos de idade. Apesar de existirem
estabelecimentos que já atendem a esta demanda, percebe-se que a proposta ainda é pouco
explorada em Porto Alegre, havendo potencial para o seu desenvolvimento, visto que o Rio
Grande do Sul é o sexto estado brasileiro que mais consome brinquedos: 5,5% das vendas
nacionais são realizadas para o estado gaúcho (ABRINQ, 2017). Neste contexto, a questão
geral que a presente pesquisa busca responder é: Como podemos amenizar o problema com o
acúmulo de brinquedos nos lares? O objetivo é elaborar um modelo de negócios para um
novo empreendimento de aluguel de brinquedos para crianças na cidade de Porto Alegre.
Para atingi-lo será elaborado um modelo de negócios baseado em hipóteses iniciais, as
quais serão testadas junto aos clientes potenciais; em seguida será elaborado um mínimo
produto viável (MVP) e, com a análise dos dados, será elaborada a versão final do modelo
de negócio. Para modelar o negócio foi adotado o modelo de desenvolvimento de clientes
sistematizado por Steve Blank (2012), utilizando o Modelo de ",
"", " ", "Negócios CANVAS de Osterwalder e Pigneur (2011). O modelo de negócios
elaborado foi testado e validado como parte do processo de desenvolvimento do cliente. A
versão final do modelo de negócios traz informações importantes para estabelecer um novo
empreendimento de aluguel de brinquedos em Porto Alegre. ", " "), meta = list(author =
character(0), datetimestamp = list(sec = 56.5238358974457, min = 45, hour = 3, mday = 2,
mon = 10, year = 118, wday = 5, yday = 305, isdst = 0),
    description = character(0), heading = character(0), id = "1038.txt", language =
"en", origin = character(0))))
## list()
## list()
```

Neste caso, você chamou apenas o segundo documento que carregou - como indicado pelo índice '[2]'. Seja cuidadoso. Qualquer um desses comandos encherá sua tela de textos rapidamente.

Tarefa 3: Pré-Processamento dos Textos

Uma vez que você tenha certeza de que todos os documentos a serem analisados foram carregados corretamente, poderá préprocessar seus textos para análise. Esta etapa permite que você execute uma série de tratamentos nos seus textos, essenciais para prepará-los para uma análise de qualidade, como por exemplo:

1. Remover números;
2. Transformar letras maiúsculas em minúsculas;
3. Remover palavras comuns (*stopwords*), sem interesse para análise;
4. Remover pontuação etc.

A preparação dos textos pode ser um pouco demorada e exigente, mas no final vale a pena para se obter análises de alta qualidade.

Processamento 1: Removendo Pontuação

Seu computador obviamente ainda não pode realmente ler. A pontuação e outros caracteres especiais parecem ser apenas mais palavras ou dados para o seu computador e para o R. Use o seguinte método para removê-los do seu texto.

```
docs <- tm_map(docs,removePunctuation)
writeLines(as.character(docs[1])) # Analise o document para verificar se funcionou.
```

Se for necessário, como quando estiver trabalhando com documentos ou e-mails padronizados, você também pode remover caracteres especiais como "@", "%", "$" etc.

Esta lista foi personalizada para remover a pontuação que você costuma encontrar em textos de e-mails. Você pode personalizar o que será removido alterando-os conforme for melhor para atender às suas necessidades específicas.

```
for (j in seq(docs)) {
     docs[[j]] <- gsub("/", " ", docs[[j]])
     docs[[j]] <- gsub("@", " ", docs[[j]])
     docs[[j]] <- gsub("\\|", " ", docs[[j]])
     docs[[j]] <- gsub("\u2028", " ", docs[[j]])
     # Este é apenas um caracter ascii, portanto deve ser removido.
}
writeLines(as.character(docs[1]))
# Você pode verificar um documento (neste caso o primeiro) para ver se a limpeza
# funcionou.
```

Processamento 2: Removendo Números

Na grande maioria dos casos de análise de textos números não interessam. Para removê-los utilize as instruções abaixo:

```
docs <- tm_map(docs, removeNumbers)
writeLines(as.character(docs[1]))
# Verifique o documento para ver se a limpeza funcionou.
```

Processamento 3: Convertendo Letras para Minúsculas

Para uma análise de qualidade, é importante que uma palavra pareça exatamente a mesma toda vez que aparecer no texto. Assim, portanto, vamos mudar todas as palavras para minúsculas, de forma a não haver diferenciação neste aspecto entre elas.

```
docs <- tm_map(docs, tolower)
docs <- tm_map(docs, PlainTextDocument)
writeLines(as.character(docs[1]))
# Verifique o documento para ver se a limpeza funcionou.
```

Processamento 4: Removendo Palavras Comuns (*stopwords*)

Em todos os textos, há muitas palavras comuns e desinteressantes (a, e, também, a, etc.), que normalmente não possuem valor analítico. Tais palavras são frequentes por sua natureza e confundirão sua análise se permanecerem no texto. Porém, antes de removermos as stopwords vamos fazer uma cópia dos "docs" para o "docscopia", de forma que se alguma coisa sair errado, poderemos reiniciar o processamento a partir desta cópia e não desde o início. Sugerimos que você utilize esta técnica sempre que for realizar uma operação mais complexa, como remover

palavras do seu conjunto de textos. Caso alguma coisa não saia como desejado, você poderá retornar para esta cópia.

```
docsCopia <- docs
# Para uma lista de stopwords, execute (use "portuguese" para vê-las em português):
length(stopwords("portuguese"))
stopwords("portuguese")
docs <- tm_map(docs, removeWords, stopwords("portuguese"))
docs <- tm_map(docs, PlainTextDocument)
writeLines(as.character(docs[1]))
Verifique o documento para ver se a limpeza funcionou.
```

Processamento 5: Removendo Palavras Específicas

Além das *stopwords*, determinadas palavras podem aparecem nos textos, mas não possuir valor para sua análise em particular. Neste caso você pode removê-las, especificamente, uma a uma, dos textos.

```
docs <- tm_map(docs, removeWords, c("sobre", "ser", "pode", "esta", "desta"))
# Remova as palavras "sobre", "ser", "pode", "esta", "desta".
# Estas palavras existem realmente nestes textos, mas não estão incluídas nas stopwords
# e não tem significado relevante para a nossa análise.
writeLines(as.character(docs[1]))
```

Processamento 6: Combinando Palavras

Se você deseja preservar um conceito que é formado por uma coleção de duas ou mais palavras (trecho), então deve combiná-las ou reduzi-las para um acrônimo significativo antes de começar a sua análise. Neste exemplo, estamos usando casos que são particulares para análise de dados qualitativos. Note que as palavras que formam cada conceito em questão foram unidas pelo sublinhado (*underscore*), de forma a serem reconhecidas pelo algoritmo de análise como uma única palavra.

```
for (j in seq(docs))
{
  docs[[j]] <- gsub("governo federal", "governo_federal", docs[[j]])
  docs[[j]] <- gsub("cadeia suprimentos", "cadeia_suprimentos", docs[[j]])
  docs[[j]] <- gsub("ensino distância", "ensino_distância", docs[[j]])
}
docs <- tm_map(docs, PlainTextDocument)
```

Processamento 7: Removendo Palavras Derivadas (*stemming*)

Palavras que por exemplo em inglês terminam "ing", "es", "s", ou em português terminam com “ando”, “endo”, “mente”, “s” são decorrentes ou derivadas de outras palavras básicas ou radicais. O processo de eliminação de palavras derivadas ou decorrentes é referido como "*stemming*" do documento. Muitas vezes é conveniente simplificar os textos dos documentos para que uma palavra seja reconhecível para o computador pelo seu radical, apesar de ter ou não uma variedade de possíveis terminações ou variações no texto original.

Nota: A função "stemming" é atualmente problemática no R, e as palavras resultantes são muitas vezes inadequadas para se ler. Por enquanto, manteremos esta seção comentada. Mas você pode experimentar essas funções (removendo o # do início da linha) se elas o interessarem. Apenas não espere que eles funcionem com perfeição. Especialmente para textos em português, a função “stemming” precisará ser utilizada com bastante cuidado.

Como procedimento de *stemming* tem sido um pouco problemático, mudaremos o nome do objeto de dados quando fizermos isso, de forma a evitar substituir permanentemente o tratamento que fizemos até este ponto, podendo retornar para ele caso o *stemming* não funcione bem.

```
# Nota: Você pode não executar esta seção de código neste particular exemplo.
for (j in seq(docs))
{
    docs[[j]] <- stemDocument(docs[[j]], language="portuguese")
}
docs_st <- tm_map(docs_st, PlainTextDocument)
writeLines(as.character(docs_st[1]))
# Verifique se a limpeza das palavras decorrentes funcionou bem. Em caso positivo
atualize o docs. Em caso negativo, deixe o docs como estiver.
# docs <- docs_st
```

Em seguida, adicione terminações comuns para melhorar a interpretabilidade.

```
# O "steamming" não parece estar funcionando bem. Você pode experimentar, mas há
inúmeros relatos da função stemCompletion, não ser atualmente operacional.
# Nota: Esse código não foi executado para esse exemplo.
# Leia-o como um potencial "Como fazer".
docs_stc <- tm_map(docs_st, stemCompletion, dictionary = DocsCopy, lazy=TRUE)
docs_stc <- tm_map(docs_stc, PlainTextDocument)
writeLines(as.character(docs_stc[1]))
# Verifique para ver se funcionou.... Somente atualize o seu docs se considerar que sim.
# docs <- docs_stc
```

Se você achar que a transformação de palavras decorrentes para o seu radical funcionou bem, converta a cópia do corpus (backup) para "docs" usando 'docs <- docs_st' ou 'docs <- docs_stc'.

Processamento 8: Removendo Espaços em Branco

Os pré-processamentos efetuados até agora deixarão os documentos com muitos "espaços em branco". O espaço em branco é o resultado de todos os espaços restantes que não foram removidos juntamente com as palavras que foram excluídas. O espaço em branco pode, e deve ser removido.

```
docs <- tm_map(docs, stripWhitespace)
writeLines(as.character(docs[1])) # Verifique se funcionou.
```

Conclusão do Pré-Processamento Básico

Para concluir, certifique-se de executar o seguinte script depois de finalizar o pré-processamento. Isso instrui ao R para tratar seus documentos pré-processados como documentos de texto.

```
docs <- tm_map(docs, PlainTextDocument)
```

Fim do estágio de pré-processamento dos textos.

Tarefa 4: Criação da Matriz de Termos dos Documentos (DTM)

Para prosseguir, crie uma matriz de termos de documentos (DTM - *Document Term Matrix*). Esta matriz será a principal fonte de dados que você vai usar a partir deste ponto para analisar os textos dos seus documentos.

```
dtm <- DocumentTermMatrix(docs)
dtm
## <<DocumentTermMatrix (documents: 127, terms: 8780)>>
## Non-/sparse entries: 33306/1081754
## Sparsity           : 97%
## Maximal term length: 75
## Weighting          : term frequency (tf)
## Você possui 127 documentos e 8.780 termos, muito esparsos, isto é, muitos termos
##(97%)ocorrem nos documentos com uma frequência muito baixa.
```

Para inspecionar a matriz DTM, você pode usar: `inspect(dtm)`.

Isso, no entanto, encherá seu terminal de textos rapidamente. Então, você pode preferir ver apenas um subconjunto:

`inspect(dtm[1:5, 1:20])` veja os primeiros 5 documentos e os primeiros 20 termos - modifique como quiser `dim(dtm)` Isso exibirá a quantidade de documentos e termos (nessa ordem).

Para continuar a sua análise você também precisará uma transposição desta matriz. Crie-a usando:

```
tdm <- TermDocumentMatrix(docs)
tdm
<<TermDocumentMatrix (terms: 8780, documents: 127)>>
Non-/sparse entries: 33306/1081754
Sparsity           : 97%
Maximal term length: 75
Weighting          : term frequency (tf)
```

Tarefa 5: Explorar Dados

Primeiro organize termos obtidos na matriz DTM pela sua frequência:

```
freq <- colSums(as.matrix(dtm))
length(freq)
## [1] 8780
ord <- order(freq)
```

Você pode preferir exportar a matriz para o Excel para visualizar melhor os termos obtidos:

```
m <- as.matrix(dtm)
dim(m)
## [1]   11 8780
write.csv(m, file="DocumentTermMatrix.csv")
```

Tarefa 6: Focando nos Dados Interessantes

Você pode se concentrar apenas nas palavras interessantes, ou seja, aquelas que ocorrem com maior frequência nos textos analisados. A função 'removeSparseTerms ()' removerá as palavras raramente usadas, deixando apenas as palavras mais utilizadas no corpo dos textos.

Porém, cuidado, palavras raras, ou seja, que aparecem poucas vezes nos textos, podem ser exatamente o que você está procurando, pois poderão distinguir um texto de outro. Portanto, esta decisão deve ser muito bem pensada.

```
#  Removendo termos esparsos:
dtms <- removeSparseTerms(dtm, 0.4)
dtms
## <<DocumentTermMatrix (documents: 127, terms: 57)>>
## Non-/sparse entries: 3734/3505
## Sparsity           : 48%
## Maximal term length: 15
## Weighting          : term frequency (tf)
## Agora a sua matriz possui 127 documentos mas apenas 57 termos, pois os muito esparsos
## foram eliminados.
```

Tarefa 7: Análise da Frequência de Termos

Como ainda há 57 termos (na matriz dtms), por enquanto, apenas confira algumas das palavras mais e menos frequentes. Para um olhar mais detalhado na frequência de termos, podemos ver uma tabela dos termos que selecionamos quando removemos os termos esparsos acima.

```
freq <- colSums(as.matrix(dtms))
freq
ainda            além           análi        analisar             ano
          146             119             142              93             123
        área          artigo         aspecto           assim        atividad
          107             136              89             157              99
      brasil      brasileiro           busca            cada    conhecimento
          201             108              76              93             116
    contexto           dado desenvolvimento            dess           dessa
          157             116             184             103              95
       dest           difer         empresa          estudo          forma
          107             107             420             444             228
     gestão          import          maior            meio        mercado
          217              85             173             156             198
  necessidad            ness            nest        objetivo     organizaçõ
          123              95              81             251             142
     outro            paí           part          partir       pesquisa
          103             145             103              99             430
     podem         prática         present        problema       processo
          100             118             125             106             236
      quai          quanto         recurso         relação      resultado
          113              83             137             152             166
    segundo           sendo          setor            tema       trabalho
           94             115             178             109             219
     valor            vez
          174             101
```

Note que há palavras com alta frequência que não foram eliminadas pelas stopwords, porém como não agregam significado para análise poderiam ser eliminadas: *ainda, além, assim, cada, dessa, dess, maior, ness, nest, outro, podem, quai, quanto, sendo, vez*. Para isso, teríamos que reprocessar a partir de “docs”.

A função 'colSums ()' gera uma tabela relatando com que frequência *cada frequência de palavras* ocorre. Usando a função 'head()', abaixo, podemos ver a distribuição das palavras menos utilizadas.

```
head(table(freq), 20)
# 20 indica que queremos apenas as primeiras 20 frequências.
freq
 76  81  83  85  89  93  94  95  99 100 101 103 106 107 108 109 113 115 116 118
  1   1   1   1   1   2   1   2   2   1   1   3   1   3   1   1   1   1   2   1
```

A saída resultante serão duas linhas de números. O número superior é a frequência com que as palavras aparecem e o número inferior reflete quantas palavras aparecem com esta frequência. Aqui, considerando apenas as 20 frequências de palavras mais baixas, podemos ver que 1602 termos aparecem apenas uma vez. Há também muitos outros que aparecem muito raramente.

Para ver os termos mais utilizados, podemos usar a função 'tail()'.

```
tail(table(freq), 20)
# 20 indica que queremos apenas as primeiras 20 frequências.
# Sinta-se à vontade para alterar esse número.
freq
145 146 152 156 157 166 173 174 178 184 198 201 217 219 228 236 251 420 430 444
  1   1   1   1   2   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
```

Considerando apenas as 20 maiores frequências, podemos ver que há uma grande disparidade na frequência com que alguns termos aparecem.

Para visualizar melhor o resultado obtido, podemos também ordenar, por ordem decrescente, a frequência das palavras obtidas na matriz dtms, conforme a seguir.

```
freq <- sort(colSums(as.matrix(dtms)), decreasing=TRUE)
estudo        pesquisa         empresa        objetivo         processo
           444             430             420             251             236
         forma        trabalho          gestão          brasil         mercado
           228             219             217             201             198
desenvolvimento           setor           valor           maior       resultado
           184             178             174             173             166
         assim        contexto            meio         relação           ainda
           157             157             156             152             146
           paí           análi      organizaçõ         recurso          artigo
           145             142             142             137             136
       present             ano      necessidad           além         prática
           125             123             123             119             118
  conhecimento           dado           sendo           quai            tema
           116             116             115             113             109
    brasileiro           área            dest           difer        problema
           108             107             107             107             106
          dess           outro            part            vez           podem
           103             103             103             101             100
      atividad          partir           dessa            ness         segundo
            99              99              95              95              94
      analisar           cada         aspecto          import          quanto
            93              93              89              85              83
          nest          busca
            81              76
```

Uma outra visão alternativa para a frequência de termos é identificar todos os termos que aparecem com determinada frequência (neste exemplo, 50 ou mais vezes).

```
findFreqTerms(dtm, lowfreq=50)

# Altere "50" para o que for mais apropriado para os seus dados de texto.
[1] "açõ"             "acordo"          "administração"   "ainda"
  [5] "além"            "alimento"        "ambient"         "análi"
  [9] "analisar"        "ano"             "apena"           "apresenta"
 [13] "área"            "artigo"          "aspecto"         "assim"
 [17] "atividad"        "ativo"           "através"         "autor"
 [21] "avaliação"       "base"            "bem"             "brasil"
 [25] "brasileira"      "brasileiro"      "busca"           "cada"
 [29] "campo"           "capacidad"       "capit"           "característica"
 [33] "caso"            "competência"     "comportamento"   "compra"
 [37] "compreend"       "conceito"        "conhecimento"    "construção"
 [41] "consumidor"      "contexto"        "corrupção"       "crescimento"
 [45] "criação"         "curso"           "custo"           "dado"
 [49] "decisão"         "desempenho"      "desenvolvimento" "dess"
 [53] "dessa"           "dest"            "diant"           "difer"
 [57] "diversa"         "diverso"         "economia"        "econômica"
 [61] "econômico"       "educação"        "efeito"          "empresa"
 [65] "ensino"          "estado"          "estratégia"      "estratégica"
 [69] "estrutura"       "estud"           "estudo"          "exist"
 [73] "fator"           "fim"             "financeira"      "financeiro"
 [77] "forma"           "formação"        "fundo"           "geral"
 [81] "gestão"          "gestor"          "governança"      "grand"
 [85] "identificar"     "impacto"         "import"          "importância"
 [89] "indivíduo"       "indústria"       "influência"      "informaçõ"
 [93] "inovação"        "instituiçõ"      "interess"        "investimento"
 [97] "literatura"      "maior"           "meio"            "melhor"
[101] "mercado"         "modelo"          "mudança"         "município"
[105] "nacion"          "necessidad"      "negócio"         "ness"
[109] "nest"            "nível"           "nova"            "novo"
[113] "objetivo"        "organização"     "organizacion"    "organizacionai"
[117] "organizaçõ"      "outro"           "paí"             "part"
[121] "partir"          "período"         "perspectiva"     "pesquisa"
[125] "pessoa"          "podem"           "poi"             "política"
[129] "ponto"           "prática"         "present"         "principai"
[133] "problema"        "processo"        "produção"        "produto"
[137] "programa"        "projeto"         "pública"         "público"
[141] "quai"            "qualidad"        "quanto"          "questão"
[145] "recurso"         "rede"            "refer"           "relação"
[149] "relev"           "relevância"      "resultado"       "risco"
[153] "seção"           "seguint"         "segundo"         "sendo"
[157] "sentido"         "serviço"         "setor"           "silva"
[161] "sistema"         "sociai"          "social"          "sociedad"
[165] "superior"        "sustentabilidad" "tai"             "tanto"
[169] "tecnologia"      "têm"             "tema"            "teoria"
[173] "termo"           "tipo"            "todo"            "trabalho"
[177] "universidad"     "uso"             "valor"           "variável"
[181] "vez"             "vista"
```

Mais uma maneira de fazer isso:

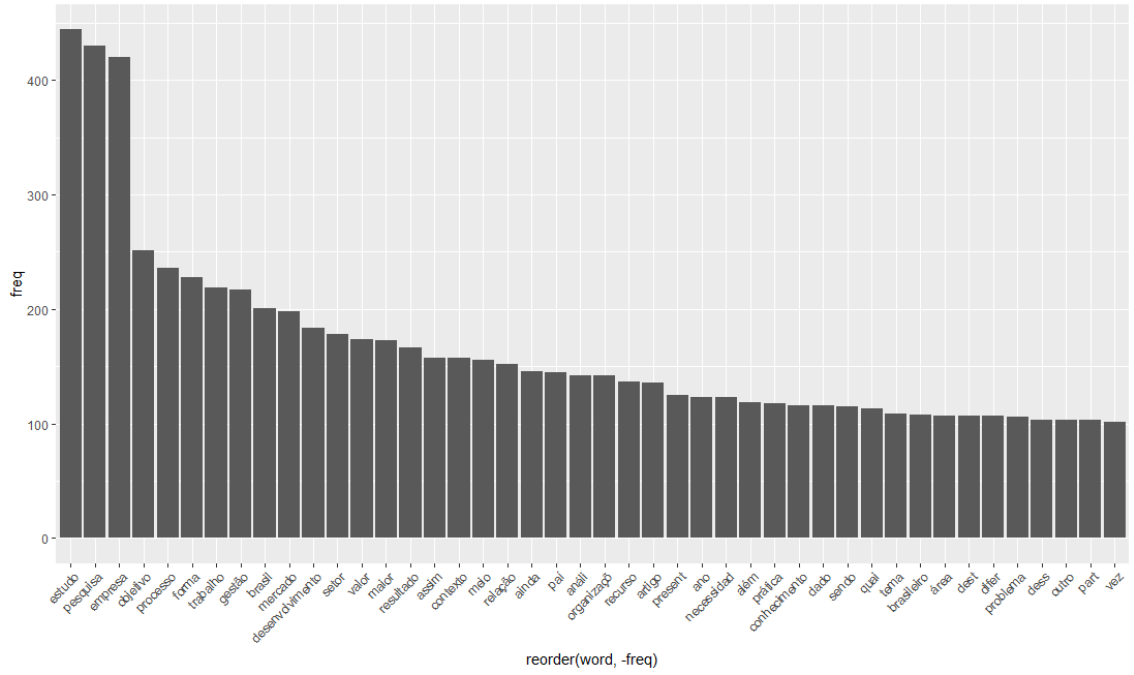
```
wf <- data.frame(word=names(freq), freq=freq)
head(wf)
             word freq
estudo     estudo  444
pesquisa pesquisa  430
empresa   empresa  420
objetivo objetivo  251
processo processo  236
forma       forma  228
```

Utilizando estes recursos podemos analisar as palavras mais frequentes encontradas e verificar que estão de acordo com os nossos objetivos. Será possível também notar diversas palavras que possuem alta frequência, mas que não possuem significado importante para a análise e objetivos desejados. Estas palavras podem ser eliminadas, repetindo o processamento a partir do passo 5, de forma a refinar a análise.

Tarefa 8: Plotar Frequências de Termos em Gráfico

Apesar dos números das frequências já informarem bastante, mostrar graficamente as palavras que aparecem com determinada frequência (pelo menos 100 vezes) nos textos analisados pode ajudar bastante a sua análise.

```
library(ggplot2)
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:NLP':
##
##     annotate
p <- ggplot(subset(wf, freq>100), aes(x = reorder(word, -freq), y = freq)) +
          geom_bar(stat = "identity") +
          theme(axis.text.x=element_text(angle=45, hjust=1))
p
```



Tarefa 9: Análise de Correlações entre Termos

Se você tem um termo em mente que achou ser particularmente significativo para sua análise, então pode achar útil identificar as palavras que mais se correlacionam (aparecem juntas) com esse termo.

Se as palavras analisadas sempre aparecerem juntas nos seus textos, então a correlação será igual a 1,0. Vamos realizar esta análise para as três palavras mais frequentes: estudo, pesquisa e empresa. Note que aqui voltamos a utilizar a matriz de termos completa (dtm), sem a remoção dos termos esparsos.

```
findAssocs(dtm, c("estudo" , "pesquisa", "empresa"), corlimit=0.60)
# especificando um limite de correlação de 0,60
$estudo
numeric(0)

$pesquisa
                razõ         contabilidad                 "be               "porqu
                0.64                 0.61                 0.60                 0.60
            publish           abundância            afiliação                apres
                0.60                 0.60                 0.60                 0.60
              àquel            batizado              binomi              citação
                0.60                 0.60                 0.60                 0.60
             citaçõ                cite            começava             comunica
                0.60                 0.60                 0.60                 0.60
           comunicar           confrontar                dará              estrato
                0.60                 0.60                 0.60                 0.60
           evangelho            flexível          ilustração               inclui
                0.60                 0.60                 0.60                 0.60
           interagem                judg           lentament                leung
                0.60                 0.60                 0.60                 0.60
              mateus              merton              natural              parcial
                0.60                 0.60                 0.60                 0.60
             perish"           posiciona           prestígio          qualidade"
                0.60                 0.60                 0.60                 0.60
            recebida           rejeitada              renom             signific
                0.60                 0.60                 0.60                 0.60
socialconstrutivista          stremersch              tirado           titulação
                0.60                 0.60                 0.60                 0.60
           transitam       universalista             vanish"              vernier
                0.60                 0.60                 0.60                 0.60
           versículo              visavi
                0.60                 0.60

$empresa
corporativa
       0.67
```

Neste exemplo, apenas "pesquisa" revelou uma correlação com vários outros termos. A configuração `corlimit=`para 0.60 impediu que a lista fosse excessivamente longa. Fique à vontade para ajustar o `corlimit=`valor que você achar necessário.

Tarefa 10: Visualizando em Nuvem de Palavras

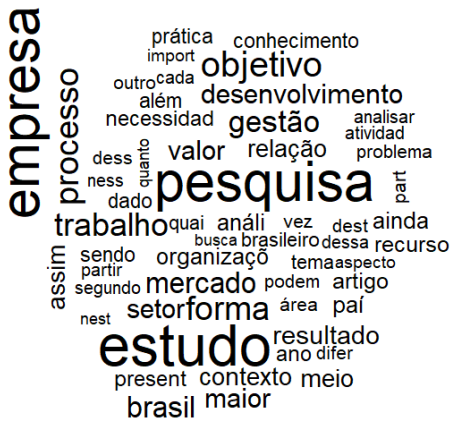
Nós seres humanos somos geralmente poderosos em análises visuais. Isso é parte do motivo pelo qual estas se tornaram tão populares. O que mostramos a seguir é uma variedade de alternativas para a construção de nuvens de palavras com os resultados dos textos analisados.

Mas primeiro você precisará carregar o pacote que faz nuvens de palavras em R. Vamos lá!!

```
library(wordcloud)
## Loading required package: RColorBrewer
```

Mostre na nuvem de palavras as que ocorrem pelo menos 50 vezes

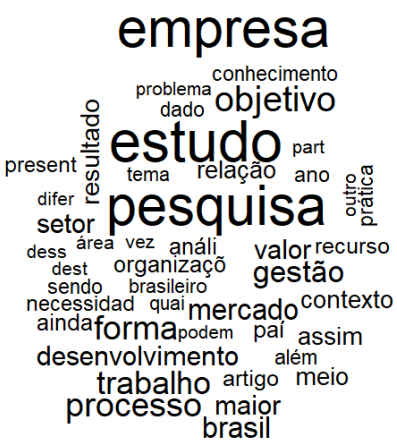
```
set.seed(142)
wordcloud(names(freq), freq, min.freq=50)
```



Nota: A função "set.seed () apenas torna consistente a configuração do layout das nuvens de palavras sempre que você as gera. Você pode omitir essa parte se não estiver preocupado em preservar um layout específico.

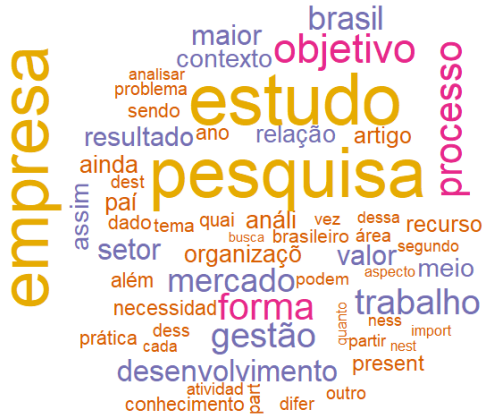
Mostre para as 100 palavras mais utilizadas.

```
set.seed(142)
wordcloud(names(freq), freq, max.words=100)
```



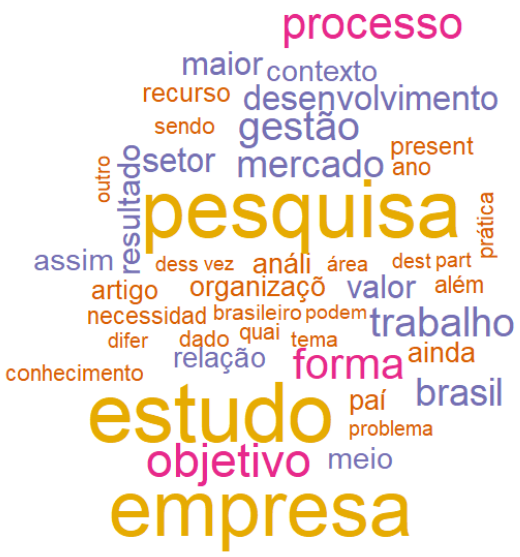
Adicione algumas palavras em cores que ocorram pelo menos 50 vezes.

```
set.seed(142)
wordcloud(names(freq), freq, min.freq=50, scale=c(5, .1), colors=brewer.pal(6, "Dark2"))
```



Mostre as 100 palavras mais frequentes.

```
set.seed(142)
dark2 <- brewer.pal(6, "Dark2")
wordcloud(names(freq), freq, max.words=100, rot.per=0.2, colors=dark2)
```



Tarefa 11: Análise de Agrupamento (Cluster)

Atividade 1: Similaridade de Termos

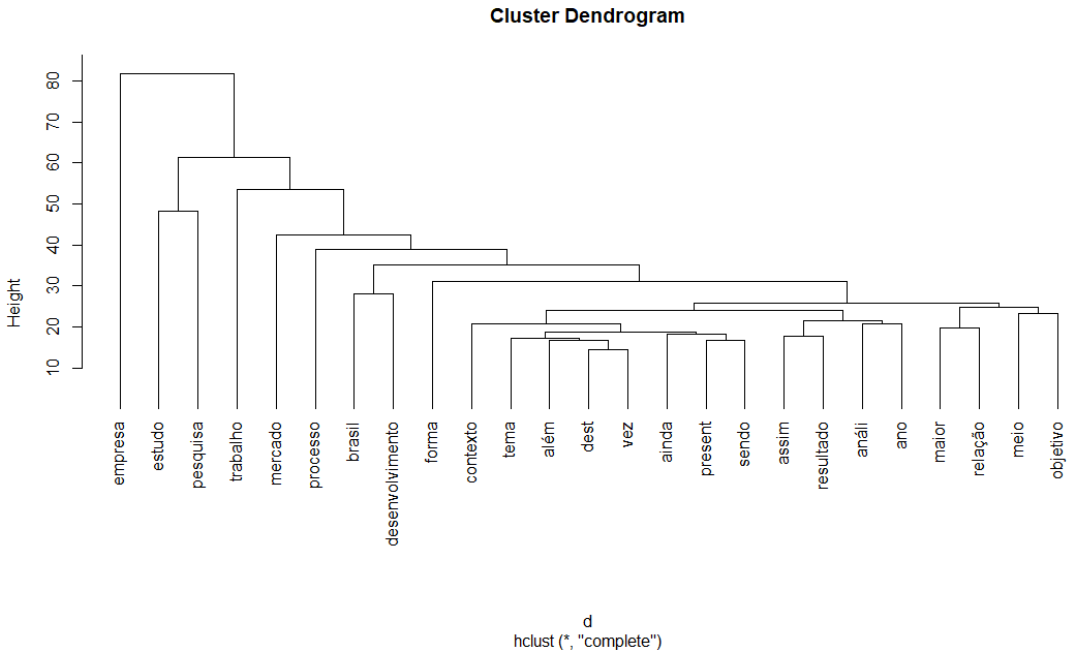
Para fazer bem uma análise de agrupamento, vamos utilizar a matriz dtms, onde já removemos uma grande parte das palavras desinteressantes ou pouco frequentes (esparsas) da matriz DTM. Se você ainda não o fez, você pode removê-los com a seguinte instrução.

```
dtms <- removeSparseTerms(dtm, 0.50)
# Isso faz com que uma matriz possua no máximo 50% de espaço vazio.
dtms
<<DocumentTermMatrix (documents: 127, terms: 25)>>
Non-/sparse entries: 1931/1244
Sparsity           : 39%
Maximal term length: 15
Weighting          : term frequency (tf)
```

Atividade 2: Agrupamento Hierárquico

Primeiro, calcule a distância entre (a frequência das) palavras e, em seguida, agrupe-as de acordo com a semelhança.

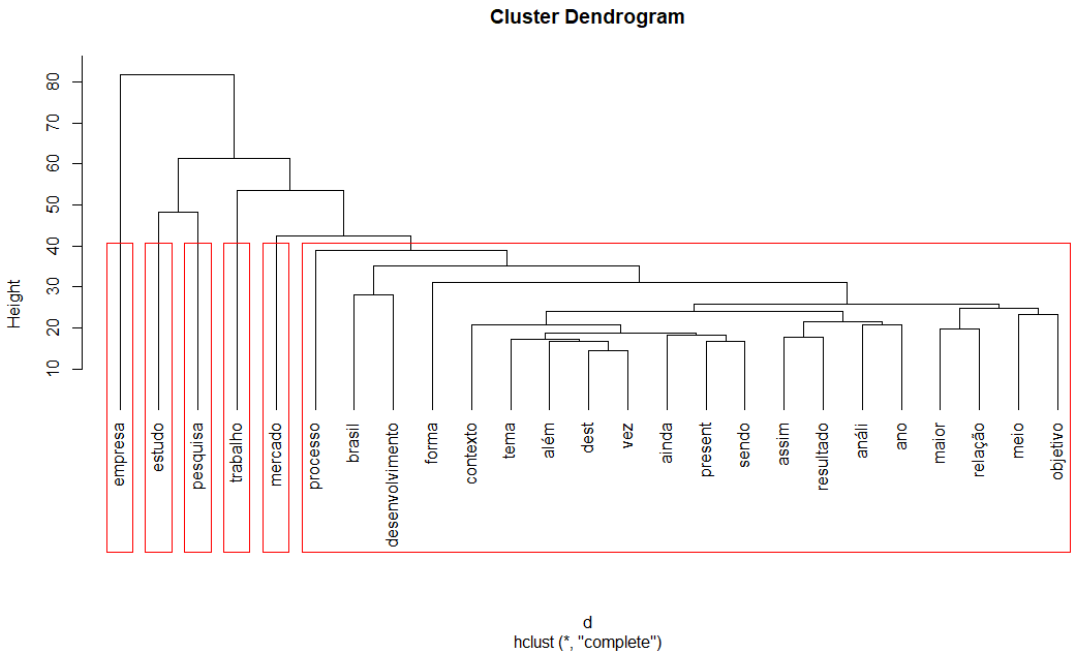
```
library(cluster)
d <- dist(t(dtms), method="euclidian")
fit <- hclust(d=d, method="complete")
# para uma aparência diferente, tente substituir: method="ward.D"
fit
##
## Call:
## hclust(d = d, method = "complete")
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 25
plot(fit, hang=-1)
```



Algumas pessoas acham que os dendrogramas são bastante claros para ler. Outros simplesmente os acham desconcertantes. Aqui, podemos ver dois, três, quatro, cinco, seis, sete ou mais grupos identificáveis no dendrograma.

Se você achar dendrogramas difíceis de ler, então ainda há esperança. Para ter uma melhor ideia de onde os grupos estão no dendrograma, você também pode pedir ao R para ajudar a identificar os clusters visualmente. Aqui, escolhemos arbitrariamente olhar cinco clusters, conforme indicado pelas caixas vermelhas. Se você deseja destacar um número diferente de grupos, sinta-se à vontade para alterar o código de acordo.

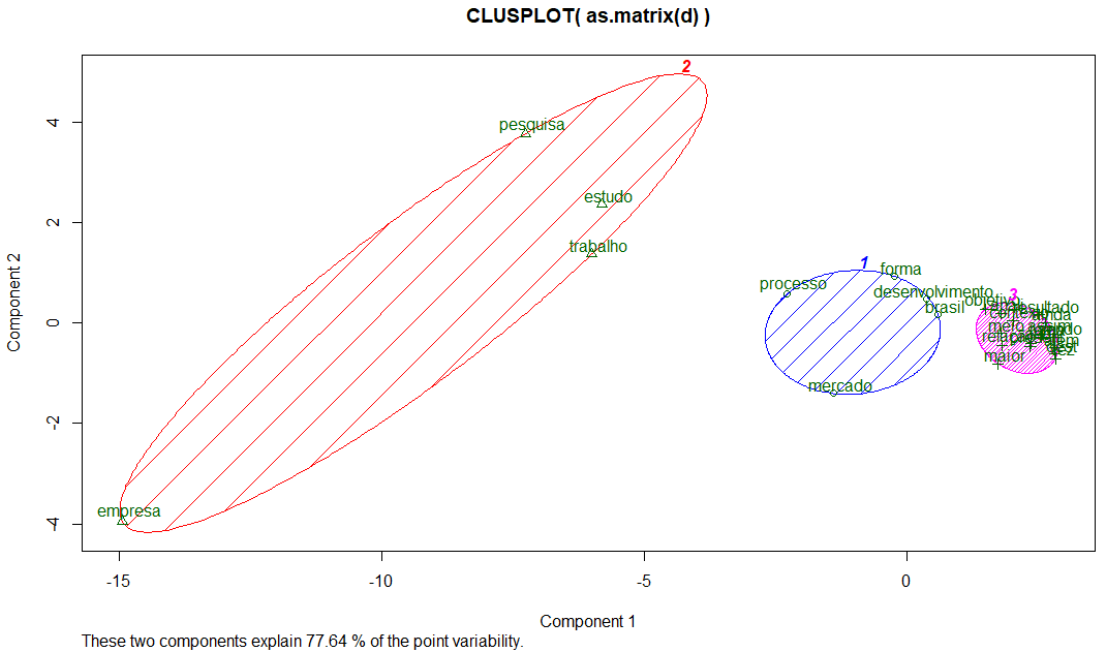
```
plot.new()
plot(fit, hang=-1)
groups <- cutree(fit, k=6)
# "k=" define o número de clusters, que você está usando
rect.hclust(fit, k=6, border="red")
# gerar dendograma com bordas vermelhas ao redor 6 clusters
```



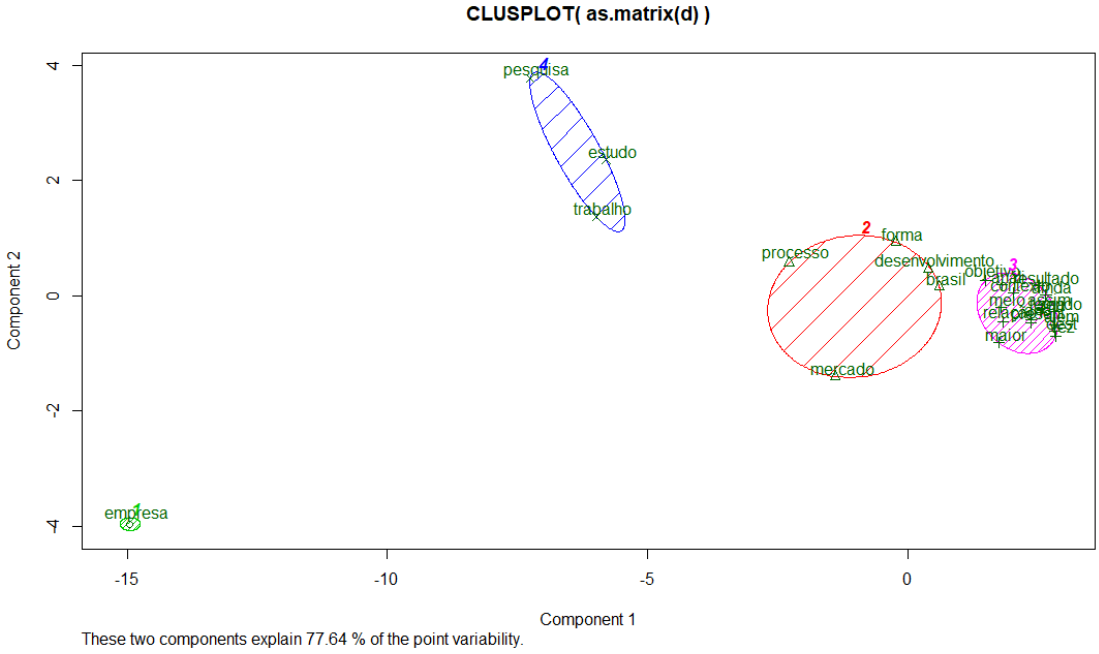
Atividade 3: Agrupamento por Partição (K-Means)

O método de agrupamento K-Means tentará agrupar palavras em um número especificado de grupos (neste primeiro caso apenas 3), de modo que a soma de distâncias quadradas entre palavras individuais e um dos centros do grupo seja mínima. Você pode alterar o número de grupos que procura, alterando o número especificado no comando kmeans().

```
library(fpc)
d <- dist(t(dtms), method="euclidian")
kfit <- kmeans(d, 3)
clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)
```



Agora utilizando 4 grupos:



Conclusão

Como você pode perceber este é apenas um roteiro introdutório para você se iniciar nas técnicas básicas de análise de textos. Para conhecer mais sobre mineração e análise de textos sugerimos que você consulte *Data Mining com Rattle e R: The Art of Excavating Data for Knowledge Discovery de Graham Williams*. Muito do material deste laboratório foi derivado de seu capítulo "*Text Mining*".

Roteiro Geral para Análise de Textos

Para executar novamente a análise de textos como a exemplificada no exercício anterior aplicada a um outro conjunto de documentos, utilize os scripts resumidos abaixo. Você sempre poderá voltar para reler o material completo acima para entender melhor o que você está fazendo em cada passo.

Instale os Pacotes R Necessários

Você só precisa fazer isso na primeira vez:

```
# * * Para começar, * * instale os pacotes que você precisa para o seu texto
#     Você só precisa fazer esta etapa uma vez.

Needed <- c("tm", "SnowballC", "RColorBrewer", "ggplot2", "wordcloud", "biclust",
"cluster", "igraph", "fpc")
install.packages(Needed, dependencies=TRUE)

install.packages("Rcampdf", repos = "http://datacube.wu.ac.at/", type = "source")

# Se você receber a seguinte mensagem:
#       Update all/some/none? [a/s/n]:
#   Digite "a" pressione [Enter]
```

Informe o seu Computador onde Encontrar os Textos.

Leia atentamente esta parte. Você precisa de três coisas únicas aqui:

a. Crie um arquivo chamado "textos" onde você manterá seus dados.
b. Salve o arquivo em um local específico
c. Drive C:\Textos
d. Copie e cole os scripts apropriados abaixo.

```
###############################################################################################
#                                    Carregando Textos
###############################################################################################
#
#     Começar, salvando seus arquivos de texto em uma pasta intitulada: "textos"
#     Este será o "corpus" (corpo) de textos de mineração.
#
# * Em um PC *, salve a pasta de seu texto na unidade c:
# use o fragmento de código a seguir:

textos <- DirSource(directory = "C:/TEXTOS/", encoding = "UTF-8")
textos
```

Usando a Função tm_map() do Pacote R tm

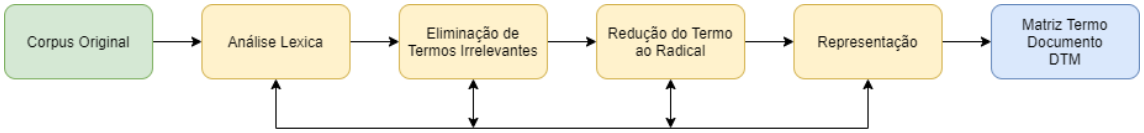
A implementação da análise é feita com a utilização de funções disponibilizadas no pacote R **tm** e, assim é necessário que ele esteja disponível no ambiente R. As instruções para instalação do pacote e para carregamento dos dados dos documentos são:

```
install.packages(tm)
library(tm)
corpus <- Corpus(df_textos)
```

A função Corpus() coloca os textos contidos em um dataframe em uma estrutura de dados adequada para que outras funções do pacote **tm** sejam aplicadas. A sintaxe e os parâmetros para uso da função **tm_map()** são:

```
corpus_tf <- tm_map(corpus, operação)
```

- **corpus** é corresponde ao conjunto de textos a ser analisado. Eles podem estar no formato vetorial ou podem ser um corpus de alguma execução precedente.
- **operação** é a indicação de qual processamento deve ser executado sobre os textos – pode ser qualquer um dos processamentos envolvidos nas fases de pré-processamento dos textos conforme ilustrado na figura a seguir.
- **corpus_tf** representa o corpus no formato vetorial ou matricial DTM (documento x termo x frequência).



Para a análise léxica, devem ser consideradas as operações de capitalização, remoção de pontuação, remoção de espaços em branco e remoção de números. Essas ações podem ser executadas conforme exemplificado a seguir. Utilize-se a função **inspect()** para visualizar o conteúdo produzido após a execução de cada operação.

```
corpus tf <- tm map(corpus, contente transformer(tolower))
corpus_tf <- tm_map(corpus_tf, removePunctuation)
corpus_tf <- tm_map(corpus_tf, removeNumbers)
```

Note que a matriz corpus_tf é utilizada como entrada para cada operação. Em muitos casos é prudente manter uma cópia inalterada da matriz corpus_tf antes de cada operação visando facilitar um retorno à operação anterior, caso necessário.

A remoção automática de termos irrelevantes é implementada a partir da escolha de uma lista de palavras (*stopwords*) disponíveis no pacote **tm**, entre elas a lista de ***stopwords*** em português é exemplificada a seguir. Note que a lista de *stopwords* em português pode não estar completa ou ser adequada para os objetivos da análise e, por esse motivo pode ser revisada e substituída ou complementada:

```
corpus_TM <- tm_map(corpus_TM, removeWords, stopwords("portuguese"))
```

É comum ser necessário adicionar novas palavras específicas à lista de *stopwords*. Neste caso a chamada da função **tm_map()** pode incluir um vetor com as novas palavras. Um exemplo de como usar esse recurso é apresentado a seguir:

```
corpus_TM <- tm_map(corpus_TM, removeWords, stopwords("portuguese"), c(palavra01,
palavra02,...,palavraNN)
```

A próxima etapa é a redução dos termos aos seus radicais, conforme exemplificado abaixo. A operação de redução (*steamming*) pode não ser perfeita para a textos em língua portuguesa pois foi originalmente desenvolvida para a língua inglesa. Recomendamos verificar a existência de configurações específicas para a língua portuguesa na versão do pacote **tm** utilizada. Note que por esse motivo geramos o corpus_tfs, de forma que você possa verificar o resultado sem perder o corpus_tf anterior.

```
corpus_tfs <- (corpus_tf, stemDocument)
```

Neste ponto o corpus já está preparado para ser transformado em uma representação vetorial para análise. Para construir uma representação baseada na frequência dos termos nos documentos, a função **TermDocumentMatrix()** deve ser utilizada conforme a sintaxe e parâmetros apresentados a seguir.

```
corpus_tf <- TermDocumentMatrix(corpus_tf, filtro)
```

- **corpus** corresponde ao conjunto de dados textuais a serem analisados.
- **filtro** permite selecionar termos com número mínimo de caracteres e/ou com frequência mínima.
- **TermDocumentMatrix** retorna uma matriz de frequência de termos.

```
corpus_tf <- TermDocumentMatrix(corpus_tf, control = list(minWordLengh = 1, minDocFreq =
1))
```

No pacote **tm** da linguagem R, a métrica ***tf – idf*** (frequência do termo – frequência invertida do termo) é obtida através a com a função **weightTfIdf()** que deve ser aplicada sobre a matriz termo documento gerada pela função **TermDocumentMatrix()**.

```
corpus_tf_idf <- weightTfIdf(corpus_tf, tipo)
```

- **corpus_tf** é a matriz de frequências de termos do corpus.
- **tipo** permite escolher entre gerar uma medida normalizada (TRUE) ou não (FALSE).

```
corpus_tf_idf <- weightTfIdf(corpus_tf, normalize=TRUE)
```

A partir dos conjuntos de dados representados pelas matrizes corpus_tf e corpus_tf_idf, diversos algoritmos de análise de dados podem ser aplicados para a obtenção de diferentes tipos de análise, como classificação ou agrupamento. Além disso, outros procedimentos mais simples de análise também podem ser aplicados, como estatísticas simples, visualizações gráficas de frequência, visualizações de palavras mais ou menos frequentes (nuvem de palavras) etc.

A partir dos resultados da análise pode-se inferir conclusões a respeito dos termos e conteúdo dos respectivos documentos, como classes, grupos, temas, assuntos mais abordados etc., de acordo com o contexto e os objetivos do pesquisador.

Execute as Preparações Desejadas

```
##############################################################################################
#                                Preparando os Dados                                         #
##############################################################################################
# ** Carregue o pacote R para mineração de texto
# Em seguida, carregue seus textos para R.* *
library(tm)
docs <- VCorpus(x=textos)
## Preprocessing
docs <- tm_map(docs,removePunctuation)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, tolower)
docs <- tm_map(docs, removeWords, stopwords("portuguese"))
docs <- tm_map(docs, removeWords, c("syllogism", "tautology"))
docs <- tm_map(docs, stripWhitespace)
docs <- tm_map(docs, PlainTextDocument)
# * Este é o fim da fase de pré-processamento.*
```

```
### Stage the Data
dtm <- DocumentTermMatrix(docs)
tdm <- TermDocumentMatrix(docs)
```

Execute as Análises Desejadas

```
###############################################################################################
#                                Analisando os Dados                                          #
###############################################################################################
freq <- colSums(as.matrix(dtm))
length(freq)
ord <- order(freq)
m <- as.matrix(dtm)
dim(m)
# gravar a matriz no Excel (csv)
write.csv(m, file="DocumentTermMatrix.csv")
###-FOCAR apenas nas coisas interessantes...
#  Comece removendo termos esparsos:
dtms <- removeSparseTerms(dtm, 0.1)
# Isso faz com que uma matriz que é 10% de espaço vazio, máximo.
### Word Frequency
head(table(freq), 20)
# A saída acima é de duas linhas de números. O primeiro número é a frequência com que
# palavras aparecem e o número de baixo reflete quantas palavras que frequentemente
# aparecem.
#
tail(table(freq), 20)
# Considerando apenas as 20 maiores frequências
#
# ** Ver uma tabela dos termos depois de remover termos esparsos, como acima.
freq <- colSums(as.matrix(dtms))
freq
# A matriz acima foi criada usando uma transformação de dados que fizemos anteriormente.
# ** Uma visão alternativa da frequência do termo:**
# Isto irá identificar todos os termos que aparecem com frequência
# (no caso, 50 ou mais vezes).
findFreqTerms(dtm, lowfreq=50)
# Mudança "50" para o que for mais apropriado para seus dados.
##
### Plotar Frequencias de Palavras
# ** Palavras que aparecem pelo menos 50 vezes.**
library(ggplot2)
wf <- data.frame(word=names(freq), freq=freq)
p <- ggplot(subset(wf, freq>50), aes(word, freq))
p <- p + geom_bar(stat="identity")
p <- p + theme(axis.text.x=element_text(angle=45, hjust=1))
p
#
## Relacionamentos entre termos
### Correlações de termo
# Veja a descrição acima para obter mais orientações com correlações.
# Se palavras aparecem sempre juntas, então correlação = 1.0.
findAssocs(dtm, c("palavra1" , "palavra2"), corlimit=0.70)
# especificando um limite de correlação de 0,70
findAssocs(dtms, "palavra3", corlimit=0.60)
# especificando um limite de correlação de 0,85
#
# Troque as palavras utilizadas anteriormente
```

```
# para termos que realmente aparecem em seus textos.
# Também ajustar o "corlimit= " para qualquer valor que você sente que é necessário.
#
### Nuvens de palavras!
# Primeiro, carregue o pacote que faz nuvens de palavras em R.
library(wordcloud)
dtms <- removeSparseTerms(dtm, 0.50) # Prepare para no máximo 50% de termos esparsos
freq <- colSums(as.matrix(dtm)) # Encontre as frequências das palavras
dark2 <- brewer.pal(6, "Dark2")
wordcloud(names(freq), freq, max.words=100, rot.per=0.2, colors=dark2)
```

Descubra Grupos de Textos pela Frequência de Termos

```
### Agrupamento por similaridade de termo

### Hierarchal Clustering
dtms <- removeSparseTerms(dtm, 0.50) # Apenas 50% de espaços vazios (termos esparsos)
library(cluster)
d <- dist(t(dtms), method="euclidian")   # First calculate distance between words
fit <- hclust(d=d, method="complete")    # Also try: method="ward.D"
plot.new()
plot(fit, hang=-1)
groups <- cutree(fit, k=6)   # "k=" defines the number of clusters you are using
rect.hclust(fit, k=6, border="red") # draw dendogram with red borders around the 5
clusters

### K-means clustering
library(fpc)
library(cluster)
dtms <- removeSparseTerms(dtm, 0.50) # Prepara para 50% de termos esparsos
d <- dist(t(dtms), method="euclidian")
kfit <- kmeans(d, 4)
clusplot(as.matrix(d), kfit$cluster, color=T, shade=T, labels=2, lines=0)
```

Etapa Final: Relatório de Elaboração do Laboratório

Você deve entregar um relatório com os resultados das etapas elaboradas neste laboratório no e-Disciplinas, para formatá-lo siga estas orientações:

1. Crie um documento Word e identifique-o com o nome do laboratório, data de elaboração e o seu nome ou da dupla que o elaborou;
2. Crie um tópico para cada resultado que você considerar relevante (manipulação de dados ou resultado de algum processamento) identificando-o com um título e uma breve explicação. Os resultados podem ser imagens de gráficos gerados ou de listas de valores ou dados de resultados obtidos. Não devem ser incluídos os scripts ou instruções de processamento utilizados, inclua apenas os resultados que você considerar relevantes.
3. No final do relatório crie um último tópico denominado "Conclusões" e elabore comentários, sugestões e conclusões sobre o que você pode aprender com a elaboração deste laboratório.

Referências

- Introdução à Mineração de Dados com Aplicações em R, Leandro Silva, Sarajane Peres e Clodis Boscarioli, Elsevier, 2016