AULA 2 - ANÁLISE DE SENTIMENTOS, MODELAGEM DE TÓPICOS E CO-OCORRÊNCIA DE PALAVRAS

Análise de Sentimentos: A partir de mineração de textos, a análise de sentimento tem por objetivo a partir de extrair opiniões e emoções contidas em uma grande quantidade de textos. Em grande parte nós vamos dividir um texto em sentimentos Negativos, Neutros e Positivos.

Exemplo de Análise de Sentimentos

Um conjunto de dados de avaliações de clientes sobre um produto ou um serviço específico Você quer entender se os comentários são, em geral, positivos, negativos ou neutros.

- "Adorei o produto! Superou minhas expectativas."
- "O produto é bom, mas o atendimento ao cliente foi ruim."
- "Não gostei do produto. A qualidade deixou a desejar."
- "O produto é aceitável, mas poderia ser melhor."

Antes de realizar a análise de sentimentos, os dados são preparados:

- **Limpeza:** Remover pontuações e caracteres especiais.
- **Tokenização:** Dividir o texto em palavras ou tokens.
- **Normalização:** Converter todas as palavras para minúsculas

Análise e Resultados

- **Positivo:** Comentários que expressam satisfação e aprovação.
- **Negativo:** Comentários que expressam insatisfação e crítica.
- **Neutro:** Comentários que são mais descritivos ou mistos, sem uma forte inclinação positiva ou negativa

Modelagem de Tópicos: Busca extrair tópicos ocultos dentro de uma grande quantidade de texto a partir de modelos matemáticos-estatísticos. O modelo de LDA (Latent Dirichlet Allocation) é um dos tipos de modelagem de tópicos mais utilizados. Cada tópico é representado por um conjunto de palavras que frequentemente aparecem juntas. Por exemplo, um tópico pode estar relacionado a "saúde", "exercício" e "dieta", enquanto outro pode estar relacionado a "tecnologia", "computadores" e "software".

Exemplo de Modelagem de Tópicos

Suponha que você tenha um conjunto de documentos sobre esportes e tecnologia. Com LDA, você define que deseja identificar 2 tópicos. Após a aplicação da LDA você pode obter os seguintes resultados:

- **Tópico 1:** Palavras mais comuns incluem "futebol", "time", "jogo", "campeonato".
- **Tópico 2:** Palavras mais comuns incluem "computador", "software", "programação", "tecnologia".
- **Análise:** Mais associado ao Tópico 1 (esporte).
- **Análise:** Mais associado ao Tópico 2 (tecnologia)

Co-ocorrência de Palavras: A análise de co-ocorrência de palavras examina como e com que frequência diferentes palavras aparecem juntas em um texto ou conjunto de textos. Ela ajuda a identificar padrões de palavras que frequentemente ocorrem próximas umas das outras, sugerindo possíveis relações ou temas comuns.

Exemplo de Co-ocorrência de Palavras

Suponha que você tenha o texto: "O gato está no tapete. O gato brinca com uma bola."

- **Co-Ocorrência de Palavras:** Se você analisar as co-ocorrências, pode descobrir que "gato" e "tapete" frequentemente aparecem juntas, sugerindo uma relação comum em textos sobre gatos e tapetes.

- **Bigramas:** Os bigramas extraídos seriam:
    - "O gato"
    - "gato está"
    - "está no"
    - "no tapete"
    - "O gato" (repetido)
    - "gato brinca"
    - "brinca com"
    - "com uma"
    - "uma bola"

Visualização e Análise

- **Nuvem de Palavras:** Pode-se usar nuvens de palavras para visualizar a frequência de bigramas ou palavras que co-ocorrem.

- **Gráficos de Co-Ocorrência:** Gráficos de rede podem mostrar como palavras estão conectadas com base em suas co-ocorrências.

QUAL O NOSSO OBJETIVO NESSA AULA?

Analisar as avaliações feitas por clientes em estabelecimentos comerciais de Bauru/SP.

RASPAGEM DOS DADOS

Utilização da extensão do Chrome Instant Data Scrapper.

Extensão:
https://chromewebstore.google.com/detail/instant-data-scraper/ofaokhiedipichpaobibbnahnkdoiiah

BASE DE DADOS

Variável	Definição
Comentário	Avaliação feita por clientes nas páginas do google maps de estabelecimentos comerciais de Bauru. Comentários não são anônimos e estão disponíveis.
ID	No R geramos um ID para cada comentário.

MODELAGEM DE TÓPICOS

Bibliotecas Necessárias

```
library(tidyverse)   # Recursos de manipulação e visualização.
library(tidytext)    # Manipulação de texto a la tidyverse.
library(tm)          # Mineração de texto.
library(topicmodels) # Modelagem de tópicos.
library(wordcloud)   # Núvem de palavras.
library(ggtern)      # Gráfico ternário.
library(LDAvis)      # Visualização de modelos LDA.
library(slam)
```

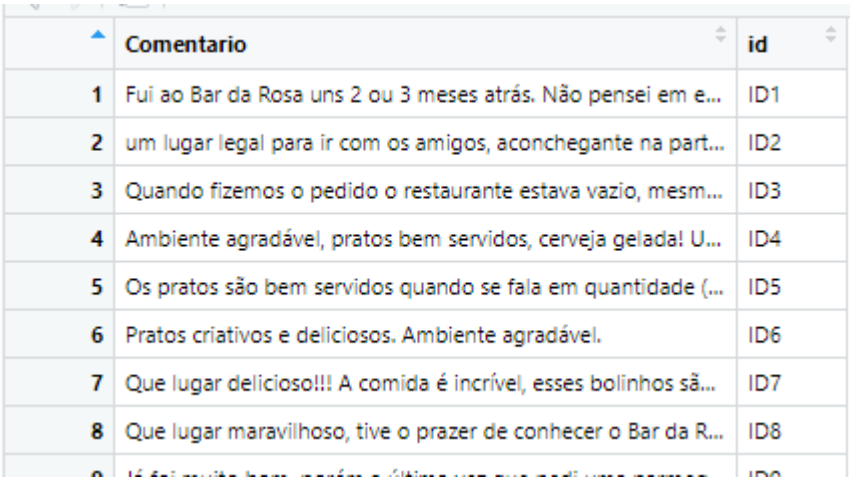
Puxando Dados

```
dados <- read.csv("confianca.csv", sep = ";", header = TRUE) |> as_tibble()
```

Adicionando uma coluna com ID aleatória para cada linha

```
dados <- dados %>%
  mutate(id = paste0("ID", 1:n()))
```

View(dados)



	Comentario	id
1	Fui ao Bar da Rosa uns 2 ou 3 meses atrás. Não pensei em e...	ID1
2	um lugar legal para ir com os amigos, aconchegante na part...	ID2
3	Quando fizemos o pedido o restaurante estava vazio, mesm...	ID3
4	Ambiente agradável, pratos bem servidos, cerveja gelada! U...	ID4
5	Os pratos são bem servidos quando se fala em quantidade (...	ID5
6	Pratos criativos e deliciosos. Ambiente agradável.	ID6
7	Que lugar delicioso!!! A comida é incrível, esses bolinhos sã...	ID7
8	Que lugar maravilhoso, tive o prazer de conhecer o Bar da R...	ID8

Criação do corpus

```
cps <- VCorpus(VectorSource(dados$Comentario),
               readerControl = list(language = "portuguese"))
```

o que é Corpus?: Conjunto estruturado de textos (documentos, frases, palavras) que é usado para análises linguísticas, mineração de texto, e processamento de linguagem natural (NLP). Ele serve como base de dados para examinar padrões, frequências de palavras, e outros aspectos do texto.

Visualização dos Documentos Após Pré-processamento

```
sapply(cps[1:3], content) %>%
  map(str_wrap, width = 60) %>%
  walk(cat, "\n\n")
```

```
Fui ao Bar da Rosa uns 2 ou 3 meses atrás. Não pensei em
escrever uma avaliação na época, mas a experiência foi tão
boa que tive que me redimir! …

um lugar legal para ir com os amigos, aconchegante na parte
interna, o atendimento da maioria é excelente

Quando fizemos o pedido o restaurante estava vazio, mesmo
assim demorou 50 minutos pro nosso pedido chegar, após os
pedidos de absolutamente TODAS as outras mesas já terem
chegado. Além disso, o mignom esta super passado do ponto,
ruim mesmo.
```

Sobre o código

sapply(cps[1:3]*, content*): Extrai o conteúdo textual dos três primeiros documentos do corpus.

map(str_wrap, width = 60): Formata o texto para que tenha um limite de 60 caracteres por linha.
walk(cat, "\n\n"): Exibe cada documento formatado, separando-os com duas quebras de linha para facilitar a leitura.

Funções de Pré-Processamento de Dados

```
#Trocando toda pontuação do texto por espaços
replacePunctuation <-
  content_transformer(FUN = function(x) {
    return(gsub(pattern = "[[:punct:]]+",
                replacement = " ",
                x = x))
  })
```

Stop Words Personalizadas

```
my_sw <- c(
  "mercado", "supermercado", "bom", "produto", "cliente", "loja", "ano", "local",
  "rede", "dia", "fazer", "ser", "muito", "pouco", "melhor", "outro", "tudo",
  "pra", "bem", "vez", "super", "mais"
)
```

O que são stopwords? Define uma lista de palavras que devem ser removidas do texto porque são consideradas irrelevantes para a análise (palavras muito comuns ou específicas do contexto, mas sem valor analítico).

Limpeza dos Dados

```
cps2 <- cps %>%
tm_map(FUN = content_transformer(tolower)) %>% # Converte todo o texto para letras minúsculas
tm_map(FUN = replacePunctuation)  %>%         # Substitui pontuação por espaços
tm_map(FUN = removeWords, words = stopwords("portuguese"))    %>% #Remove stopwords padrão em português
tm_map(FUN = removeWords, words = my_sw)    %>%    # Remove stopwords personalizadas
tm_map(FUN = stemDocument, language = "portuguese")  %>% # Aplica stemming para reduzir palavras às suas raízes
tm_map(FUN = removeNumbers)          %>%     # Remove números do texto
tm_map(FUN = stripWhitespace)       %>%       # Remove espaços em branco extras
tm_map(FUN = content_transformer(trimws))  # Remove espaços em branco no início e fim
```

Criação da Matriz Documento-Termos

```
dtm <- DocumentTermMatrix(cps2)
dtm
```

```
<<DocumentTermMatrix (documents: 420, terms: 1383)>>
Non-/sparse entries: 4397/576463
Sparsity           : 99%
Maximal term length: 18
Weighting          : term frequency (tf)
```

Matriz Documento-Termos (DTM) a partir do corpus pré-processado `cps2`. A DTM é uma matriz que quantifica a frequência de termos (palavras) em cada documento do corpus.

Linhas: Cada linha da matriz representa um documento.
Colunas: Cada coluna representa um termo (palavra).
Valores: Cada célula da matriz contém o número de vezes que o termo aparece no documento correspondente.

Reducao da esparsidade

```
rst <- removeSparseTerms(dtm, sparse = 0.975)
rst
```

Reduz a esparsidade da matriz (termos com pouca presença) que fizemos anteriormente. Mantemos os termos que aparecem pelo menos em 2,5% dos documentos (0.975)

Verifique a soma das entradas de cada linha

```
row_sums_rst <- row_sums(rst)
```

Remova as linhas que não contêm termos

```
rst <- rst[row_sums_rst > 0, ]

#Modelo LDA
fit <- LDA(rst, k = 3, control = list(seed = 1234))
```

Aqui estamos ajustando o modelo de LDA

LDA(rst, k = 3, control = list(seed = 1234)): Aplica o modelo LDA à matriz Documento-Termos rst. Você pode alterar o número de K.
k = 3: Define o número de tópicos que o modelo deve identificar. No seu caso, o LDA tentará descobrir 3 tópicos distintos nos documentos.
control = list(seed = 1234): Define uma semente para o gerador de números aleatórios, garantindo que os resultados sejam reproduzíveis.

Análise de Tópicos

```
table(topics(fit))
```

```
> table(topics(fit))

  1   2   3
148 120 130
```

terms(fit)

```
> terms(fit)
      Topic 1       Topic 2       Topic 3
     "comida" "atendimento" "atendimento"
```

Conferindo apenas as propriedades dos coeficientes

```
fit_coefs <- posterior(fit)
str(fit_coefs)
```

```
> str(fit_coefs)
List of 2
 $ terms : num [1:3, 1:58] 0.0103 0.0156 0.0135 0.0192 0.0233 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "1" "2" "3"
  .. ..$ : chr [1:58] "aconchegant" "agradável" "além" "almoço" ...
 $ topics: num [1:398, 1:3] 0.329 0.331 0.327 0.34 0.33 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:398] "1" "2" "3" "4" ...
  .. ..$ : chr [1:3] "1" "2" "3"
>
```

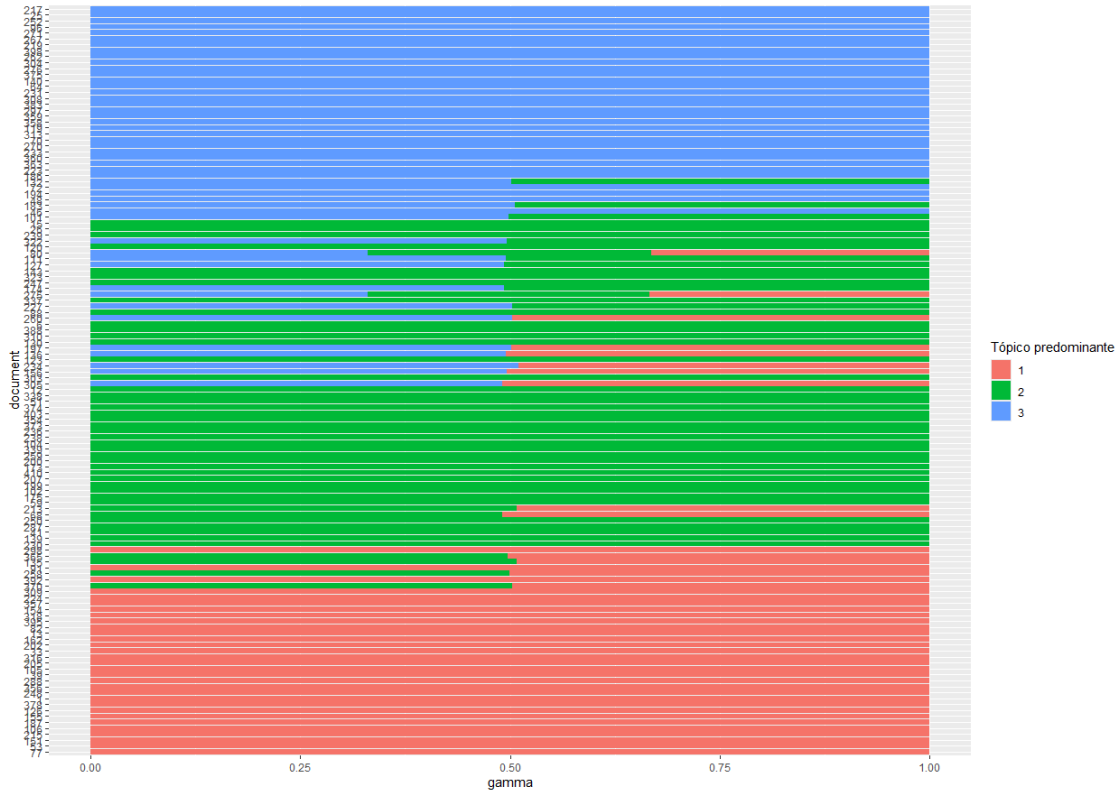
Distribuição dos tópicos

```
topic_coef <- tidy(fit, matrix = "gamma")
aux <- sample_n(topic_coef, size = 150) %>%
  arrange(topic, gamma) %>%
  mutate(document = fct_reorder(document, row_number()))
```

topic_coef <- tidy(fit, matrix = "gamma"): Extrai os coeficientes dos tópicos do modelo LDA. A matriz "gamma" fornece a proporção de cada tópico em cada documento.
aux <- sample_n(topic_coef, size = 150) %>%: Seleciona uma amostra de 150 linhas dos coeficientes dos tópicos. Isso é útil para reduzir a quantidade de dados e focar em uma parte representativa.
arrange(topic, gamma) %>%: Ordena os dados pela variável topic e, dentro de cada tópico, pela proporção gamma (a importância do tópico em cada documento).
mutate(document = fct_reorder(document, row_number())): Reordena os documentos para que eles apareçam de forma mais clara no gráfico.

Gráfico

```
ggplot(data = aux) +
  aes(x = document, y = gamma, fill = factor(topic)) +
  geom_col(position = "fill") +
  labs(fill = "Tópico predominante") +
  coord_flip()
```



Distribuição dos termos dos tópicos

```
terms_coef <- tidy(fit, matrix = "beta")
topn_terms <- terms_coef %>%
  group_by(topic) %>%
  top_n(n = 50, wt = beta) %>%
  ungroup()
```

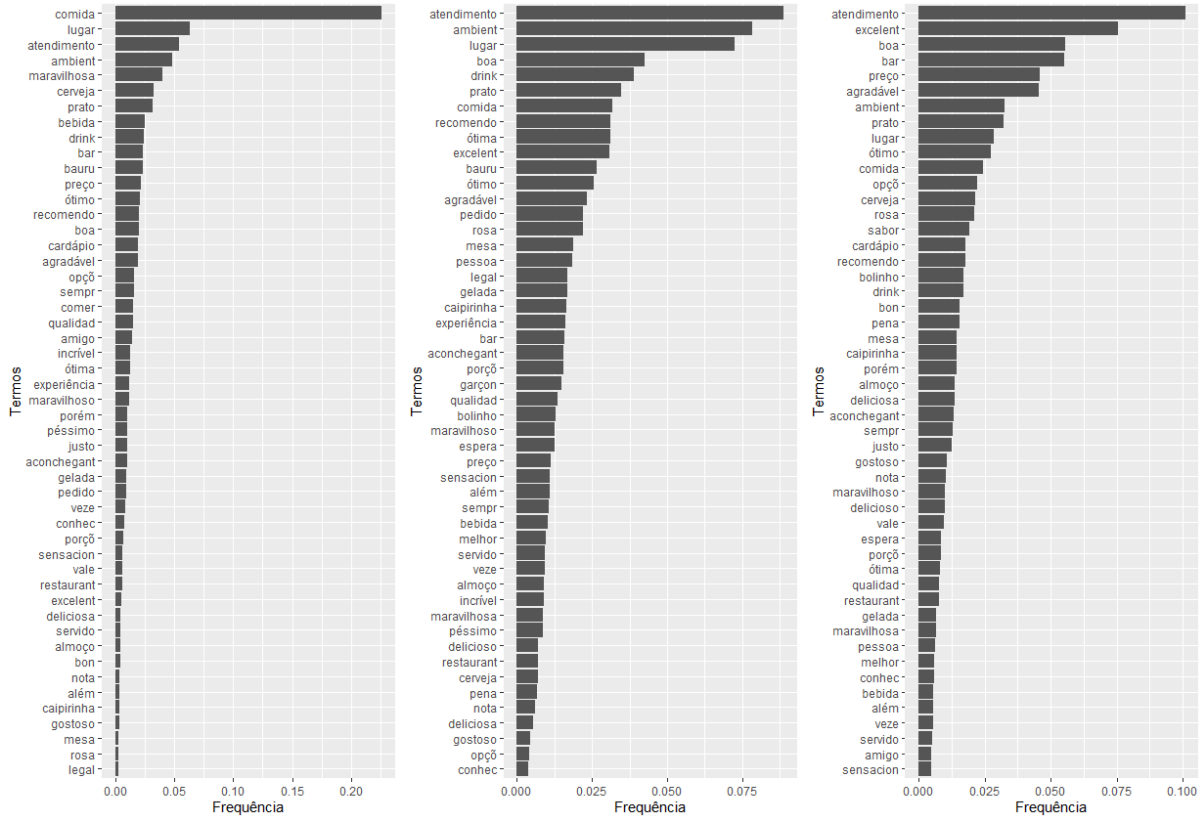
terms_coef <- tidy(fit, matrix = "beta"): Extrai os coeficientes dos termos do modelo LDA. A matriz "beta" fornece a importância de cada termo para cada tópico, ou seja, a probabilidade de cada termo dado um tópico.
topn_terms <- terms_coef %>%: Inicia a manipulação dos dados extraídos.
group_by(topic) %>%: Agrupa os coeficientes por tópico, para que possamos analisar os termos para cada tópico individualmente.
top_n(n = 50, wt = beta) %>%: Seleciona os 50 termos mais importantes (com maior valor de beta) para cada tópico. O argumento wt = beta especifica que estamos ordenando os termos com base na importância do termo para o tópico.
ungroup(): Remove o agrupamento, resultando em um dataframe com os termos mais importantes para cada tópico

Gráfico por termo de tópico

```
pp <- topn_terms %>%
  group_by(topic) %>%
  do(plot = {
    ggplot(.) +
      aes(x = reorder(term, beta), y = beta) +
```

```
    geom_col() +
    labs(x = "Termos", y = "Frequência") +
    coord_flip()
  })
length(pp$plot)

do.call(what = gridExtra::grid.arrange, args = c(pp$plot, nrow = 1))
```



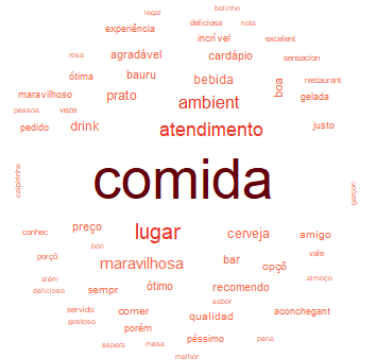
Nuvem de palavras por tópicos

```
topn_terms <- terms_coef %>%
  group_by(topic) %>%
  top_n(300, beta) %>%
  ungroup()

i <- 0
pal <- c("Reds", "Blues", "Greens", "Purples")[1:fit@k]

oldpar <- par()
par(mfrow = c(2, 2), mar = c(0, 0, 0, 0))
topn_terms %>%
  group_by(topic) %>%
  do(plot = {
    i <<- i + 1
    wordcloud(words = .$term,
              freq = .$beta,
```

```
        min.freq = 1,
        max.words = 300,
        random.order = FALSE,
        colors = tail(brewer.pal(9, pal[i]), n = 5))
})
```



MODELAGEM DE TÓPICOS (mais simples)

```
library(stm)
library(dplyr)
library(knitr)
library(tm)
library(SnowballC)

#Limpar Texto

limpar_texto <-\(x){
  x <- gsub("\\b[IVXLCDM]+\\b", "", x)
  return(x)}
```

limpar_texto <- \(x){ ... }: Define uma função anônima em R. Essa função é usada para limpar o texto fornecido.
x <- gsub("\\b[IVXLCDM]+\\b", "", x): Dentro da função, gsub é usado para remover sequências de caracteres que correspondem a números romanos. O padrão \\b[IVXLCDM]+\\b corresponde a qualquer sequência de caracteres que são números romanos (I, V, X, L, C, D, M) e gsub substitui essas sequências por uma string vazia.
return(x): Retorna o texto limpo.

```
analise <- limpar_texto(dados$Comentario)
```

Aplica o limpar texto aos comentários e atribui ele a análise.

Pré-processamento

```
processed <- textProcessor(analise,
                           metadata = dados,
                           ucp = TRUE,
                           onlycharacter = TRUE,
                           wordLengths = c(3, Inf),
                           language = "portuguese")
```

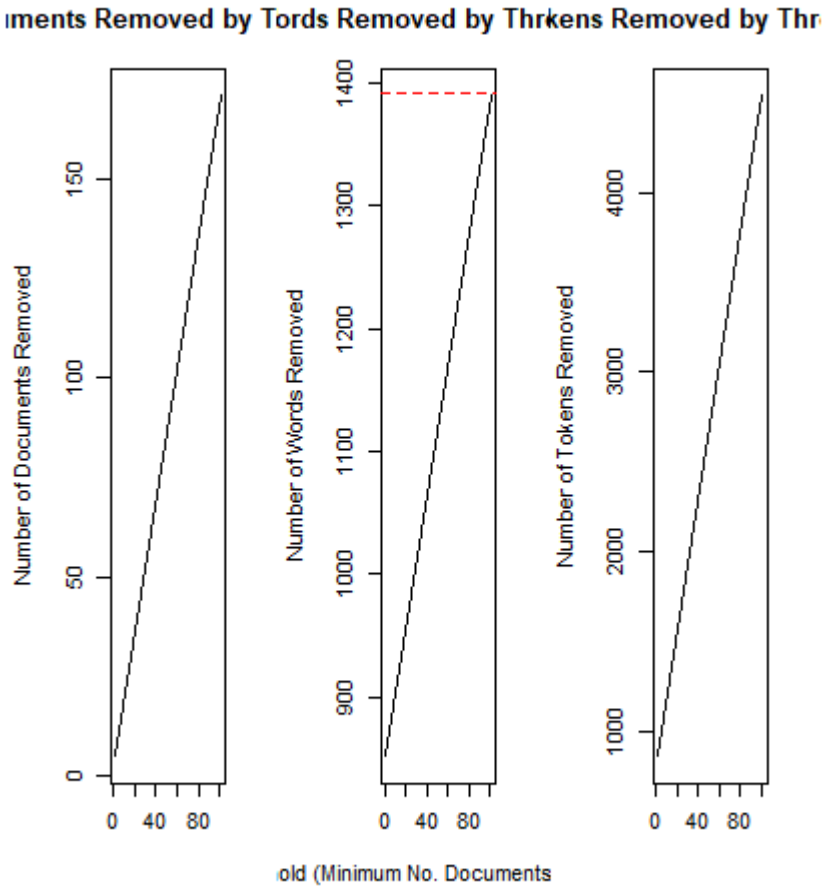
Função textProcessor:
textProcessor(analise, ...): Aplica uma série de pré-processamentos ao texto analise (que você obteve após limpar o texto). O textProcessor é usado para transformar o texto de acordo com as especificações fornecidas.
analise: O texto limpo que será processado.
metadata = dados: Adiciona metadados ao processo, aqui referenciando o dataframe dados. Os metadados podem incluir informações adicionais sobre o texto, como IDs de documentos ou outras características.
ucp = TRUE: Pode indicar que a função deve usar um processamento de caracteres Unicode para garantir que caracteres especiais sejam tratados corretamente.
onlycharacter = TRUE: Especifica que apenas caracteres (e não números ou pontuações) devem ser considerados.
wordLengths = c(3, Inf): Define o comprimento mínimo e máximo das palavras a serem mantidas. Nesse caso, palavras com comprimento de 3 ou mais caracteres serão mantidas, e palavras com comprimento menor serão removidas.
language = "portuguese": Define o idioma para as operações de processamento, garantindo que funções como remoção de stopwords considerem o idioma português.

Associando texto com metadados

```
str(processed)
```

Visualização de Termos Removidos

```
plotRemoved(processed$documents, lower.thresh = seq(1, 200, by = 100))
```



Preparação dos textos para modelagem

```
out <- prepDocuments(processed$documents,
                     processed$vocab,
                     processed$meta,
                     lower.thresh = 1)

docs <- out$documents
vocab <- out$vocab
meta <- out$meta
```

```
> str(processed)
List of 4
 $ documents   :List of 419
  ..$ 1  : int [1:2, 1:12] 122 1 131 1 144 1 173 1 494 1 ...
  ..$ 2  : int [1:2, 1:9] 21 1 75 1 118 1 541 1 740 1 ...
  ..$ 3  : int [1:2, 1:23] 4 1 54 1 92 1 105 1 248 1 ...
  ..$ 4  : int [1:2, 1:13] 39 1 71 1 75 1 156 1 164 2 ...
  ..$ 5  : int [1:2, 1:21] 59 1 164 1 180 1 219 1 542 1 ...
  ..$ 6  : int [1:2, 1:5] 39 1 71 1 343 1 377 1 1038 1
```

```
 $ vocab       : chr [1:1392] "abaixo" "aberto" "abr" "absolutament" ...
 $ meta        :'data.frame':   419 obs. of  2 variables:
  ..$ Comentario: chr [1:419] "Fui ao Bar da Rosa uns 2 ou 3 meses atrás. Não pensei em escrever
uma avaliação na época, mas a experiência foi"| __truncated__ "um lugar legal para ir com os ami
gos, aconchegante na parte interna, o atendimento da maioria é excelente" "Quando fizemos o pedi
do o restaurante estava vazio, mesmo assim demorou 50 minutos pro nosso pedido chegar, após"| __t
runcated__ "Ambiente agradável, pratos bem servidos, cerveja gelada! Um lugar pra ir comer bem e
bater papo com os amigos! …" ...
  ..$ id        : chr [1:419] "ID1" "ID2" "ID3" "ID4" ...
 $ docs.removed: int 419
 - attr(*, "class")= chr "textProcessor"
```

prepDocuments: Prepara os documentos, vocabulário e metadados para a modelagem de tópicos.
lower.thresh = 1: Define um limiar para a frequência mínima dos termos que serão mantidos. Nesse caso, qualquer termo que apareça pelo menos uma vez será incluído.
docs: Documentos preparados para a modelagem.
vocab: Vocabulário filtrado com base no limiar de frequência.
meta: Metadados associados aos documentos

```
fit <- stm(
  documents = docs, vocab = vocab, data = meta,  K = 10,
  max.em.its = 75, init.type = "Spectral",seed = 05959 ,verbose = FALSE)
```

stm: Ajusta um modelo de tópicos com **Structural Topic Modeling (STM)**.
documents = docs: Documentos preparados para a modelagem.
vocab = vocab: Vocabulário preparado.
data = meta: Metadados associados aos documentos.
K = 10: Define o número de tópicos que o modelo deve identificar (10 tópicos).
max.em.its = 75: Define o número máximo de iterações para o algoritmo de Expectation-Maximization.
init.type = "Spectral": Tipo de inicialização do modelo (Spectral é um método para inicializar os parâmetros).
seed = 05959: Define uma semente para garantir a reprodutibilidade dos resultados.
verbose = FALSE: Desativa a saída detalhada durante o ajuste do modelo.

Rotulação dos Tópicos

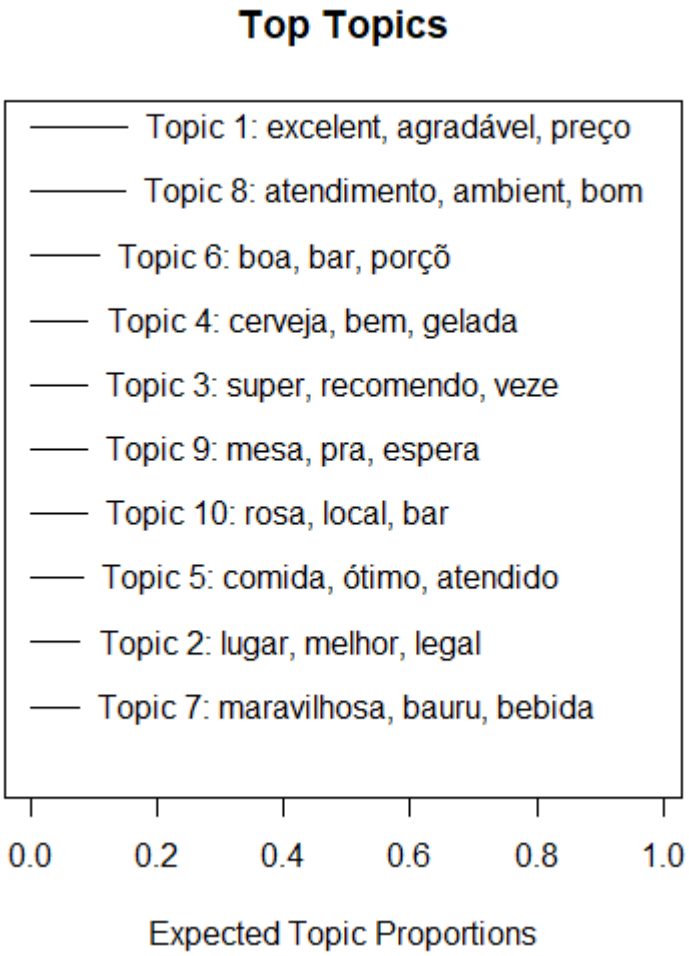
```
labelTopics(fit)
```

labelTopics: Identifica e rotula os tópicos no modelo ajustado. Essa função ajuda a entender e interpretar o conteúdo de cada tópico com base nas palavras mais relevantes.

```
Topic 1 Top Words:
         Highest Prob: excelent, agradável, preço, drink, bem, prato, recomendo
         FREX: excelent, sensacion, qualidad, diferenciado, justo, produto, show
         Lift: descontraido, diferenciado, diverso, fantástica, produto, recepção, sensacior
         Score: excelent, fantástica, agradável, qualidad, drink, justo, cardápio
Topic 2 Top Words:
         Highest Prob: lugar, melhor, legal, restaurant, incrível, amigo, poderia
         FREX: lugar, legal, poderia, melhor, restaurant, elevado, excelência
         Lift: elevado, excelência, oferecido, origin, poderia, legal, estar
         Score: lugar, origin, melhor, legal, mto, poderia, tradicion
Topic 3 Top Words:
         Highest Prob: super, recomendo, veze, atencioso, funcionário, bolinho, bar
         FREX: atencioso, funcionário, feijoada, conta, porco, saboroso, principalment
         Lift: apertado, compartilhar, conta, corpo, feijoada, mudança, nao
         Score: ouro, super, atencioso, funcionário, feijoada, baião, principalment
Topic 4 Top Words:
         Highest Prob: cerveja, bem, gelada, pedido, péssimo, servido, veio
         FREX: servido, muita, cerveja, camisinha, gelada, dono, outro
         Lift: servido, aguardar, algun, camisinha, cerva, cobraram, dão
         Score: vai, cerveja, gelada, pedido, camisinha, servido, veio
Topic 5 Top Words:
         Highest Prob: comida, ótimo, atendido, chegar, fiquei, poi, mil
         FREX: ótimo, comida, interessant, necessário, atendido, desejar, próxima
         Lift: necessário, desejar, drinqu, podia, interessant, vou, meia
         Score: comida, necessário, ótimo, min, poi, atendido, fiquei
Topic 6 Top Words:
         Highest Prob: boa, bar, porçõ, prato, gostoso, nota, almoço
         FREX: cidad, culinária, nordestina, toqu, adorei, vegetariana, boa
         Lift: cidad, acessível, atendem, betão, culinária, deseja, dúvida
         Score: boa, betão, bar, toqu, conseguir, cidad, nordestina
Topic 7 Top Words:
         Highest Prob: maravilhosa, bauru, bebida, caipirinha, opçõ, deliciosa, sempr
         FREX: bebida, bauru, maravilhosa, sobremesa, hambúrguer, deliciosa, variedad
         Lift: duro, hambúrguer, incrívei, seo, entregar, sobremesa, bebida
         Score: duro, maravilhosa, bauru, bebida, caipirinha, hambúrguer, deliciosa
Topic 8 Top Words:
         Highest Prob: atendimento, ambient, bom, ótima, aconchegant, pena, conhec
         FREX: conhec, feita, atendimento, pena, ótima, top, vale
         Lift: combinaçõ, criativo, feita, frequentado, gelado, linda, rapidament
         Score: atendimento, gelado, bom, ambient, ótima, aconchegant, pena
Topic 9 Top Words:
         Highest Prob: mesa, pra, espera, fila, tempo, bolinho, todo
         FREX: fila, perfeito, casa, nada, seca, sentar, hora
         Lift: bolso, buteco, casa, coalho, costela, enquanto, entrega
         Score: vegana, mesa, pra, fila, nada, seca, carn
Topic 10 Top Words:
         Highest Prob: rosa, local, bar, dia, ponto, além, cozinha
         FREX: nova, rosa, cozinha, ponto, local, achei, dia
```

Visualização dos tópicos por principais palavras

plot(fit, type = "summary", xlim = c(0, 0.99))

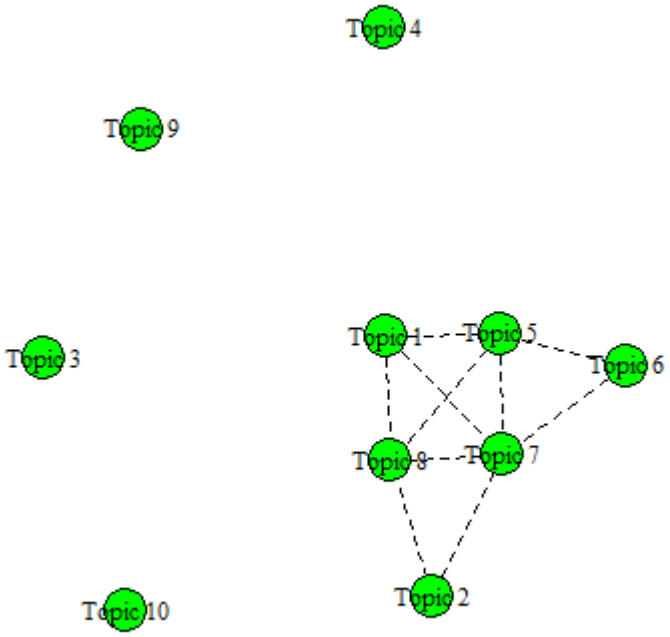


plot(fit, type = "summary"): Cria um gráfico resumido do modelo STM, mostrando a distribuição dos tópicos e suas principais características.
xlim = c(0, 0.99): Define o intervalo do eixo x para a visualização, ajustando o intervalo de valores exibidos.

Comparação entre tópicos
plot(fit, type = "perspectives", topics = c(3, 10))

plot(fit, type = "perspectives", topics = c(3, 10)): Cria gráficos de perspectivas para os tópicos especificados (3 e 10). Esse tipo de gráfico mostra como os tópicos são distribuídos ao longo dos documentos e suas relações.

Correlação entre os tópicos
mod.out.corr <- topicCorr(fit)
plot(mod.out.corr)



topicCorr(fit): Calcula a correlação entre os tópicos identificados pelo modelo STM. Ajuda a entender como os tópicos estão relacionados entre si.
plot(mod.out.corr): Plota a matriz de correlação entre os tópicos, permitindo visualizar a força e a direção das relações entre eles.

CO-OCORRÊNCIA DE PALAVRAS

Carregando pacotes
library(tm) **# Texto Mining - Para processamento de texto e análise de texto.**
library(tidytext) **# Análise de texto em formato tidy - Para manipulação e visualização de dados textuais.**
library(dplyr) **# Manipulação de dados - Para transformar e resumir dados.**
library(igraph) **# Análise de redes - Para criar, manipular e visualizar gráficos e redes**.
library(tidyr) **# Manipulação de dados - Para arrumar e transformar dados em formato tidy.**
library(ggraph) **# Visualização de redes - Para criar gráficos de redes e visualizações baseadas em grafos.**

Criando um novo corpus para o texto
document <- Corpus(VectorSource(dados$Comentario))

 # Aplicar as transformações no texto
document <- tm_map(document, content_transformer(tolower)) **# Converte para minúsculas**
document <- tm_map(document, removeNumbers) **# Remove números**

document <- tm_map(document, removeWords, stopwords("portuguese")) **#Remove palavras de parada em português**
document <- tm_map(document, removePunctuation) **# Remove pontuações**

Criação de Dataframe
dadosN <- data.frame(text = sapply(document, as.character), stringAsFactors = FALSE)

Criação do data.frame: data.frame é uma estrutura de dados tabular que armazena os textos processados em uma coluna chamada text.
Conversão dos Textos: sapply(document, as.character) converte cada item da lista document em um texto (caracteres) e os coloca na coluna text do data.frame.
Prevenção de Fatores: stringAsFactors = FALSE evita que as strings sejam convertidas em fatores, mantendo os dados textuais como strings.

Criando Bigramas
New_bigramas <- dadosN%>%
  unnest_tokens(New_bigramas, text, token = "ngrams", n = 2)
New_bigramas

	stringAsFactors	New_bigramas
1	FALSE	bar rosa
2	FALSE	rosa uns
3	FALSE	uns meses
4	FALSE	meses atrás
5	FALSE	atrás pensei
6	FALSE	pensei escrever

Cria Bigrams: A função unnest_tokens divide os textos em pares de palavras (bigrams).
Resultado: O data.frame New_bigramas conterá esses bigramas, onde cada linha representa um bigrama extraído de um texto do data.frame original.

Separar Bigramas

bigramassep <- New_bigramas %>%
  separate(New_bigramas, c("word1", "word2"), sep = " ")

Limpar Stopwords dos bigramas
bigramasfiltr <- bigramassep %>%
  filter(!word1 %in% stop_words$word) %>%
  filter(!word2 %in% stop_words$word)

Remove Bigrams com Stopwords: Remove bigramas que contêm palavras de parada, ajudando a focar em bigramas que são mais relevantes e significativos para a análise.
Resultado: O data.frame bigramasfiltr conterá bigramas onde nem a primeira nem a segunda palavra são stopwords, o que pode melhorar a qualidade e a relevância da análise.

Contagem de Bigramas

```
bigramascont <- bigramasfiltr %>%
  count(word1, word2, sort = TRUE)
bigramascont
```

```
> bigramascont
            word1        word2  n
1          comida          boa 27
2          comida  maravilhosa 23
3          comida            é 23
4        ambiente    agradável 22
5             bom  atendimento 19
6             bar         rosa 18
7          ótimo  atendimento 16
```

Só por aqui já é possível verificar alguns pontos de associação

Remover linhas com NA's

```
bigramascont <- bigramascont %>%
  drop_na()
```

Visualização Final da Contagem de Bigramas

```
View(bigramascont)
```

	word1	word2	n
1	comida	boa	27
2	comida	maravilhosa	23
3	comida	é	23
4	ambiente	agradável	22
5	bom	atendimento	19
6	bar	rosa	18
7	ótimo	atendimento	16

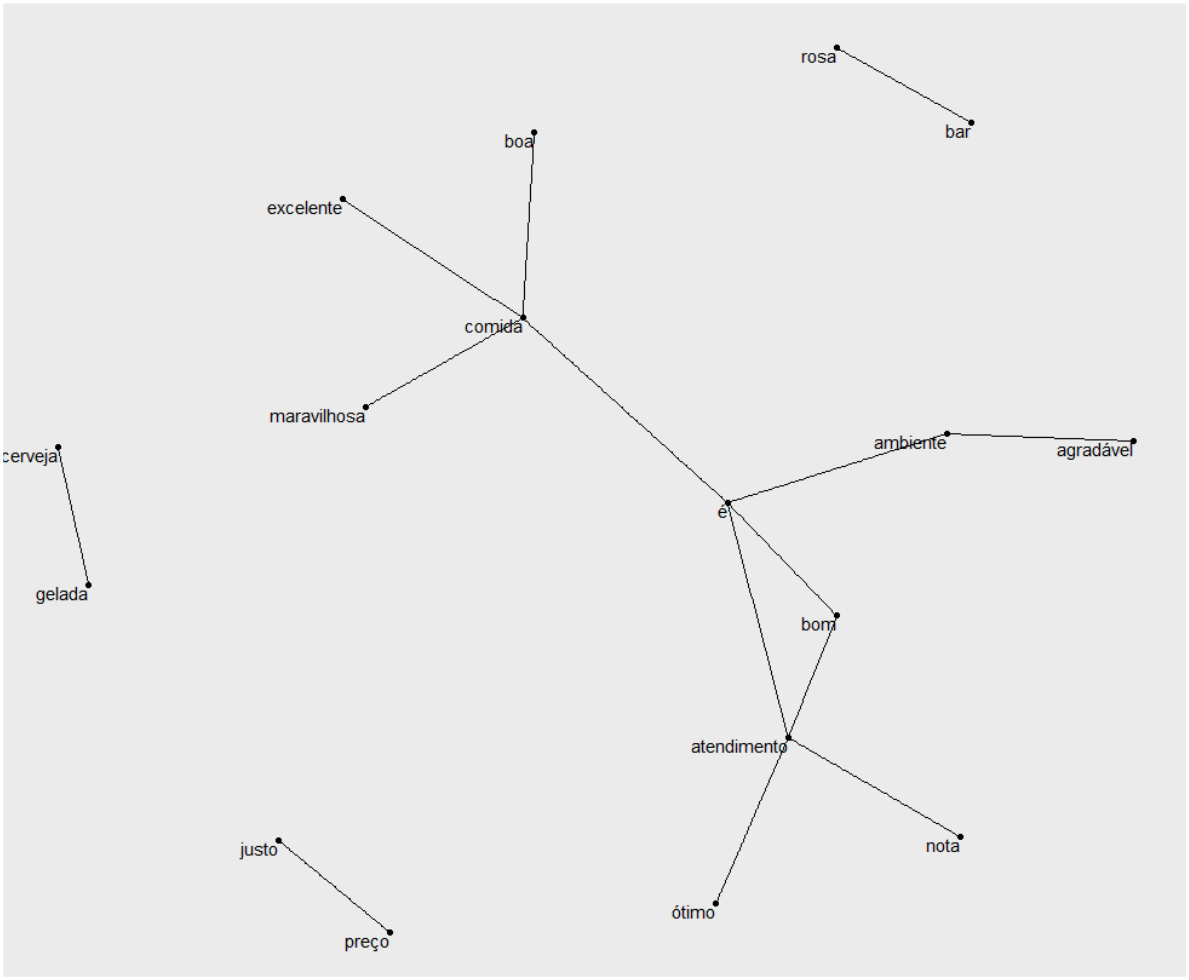
Criar o grafo a partir do dataframe limpo

```
grafico <- bigramascont %>%
  filter(n > 10) %>%
  graph_from_data_frame()
```

Aqui indicamos que queremos no nosso dataframe apenas pares que tenham mais de 10 ligações

```
#Criar grafico baseado nas co-ocorrencias
set.seed(2017)
ggraph(grafico, layout = "nicely")+
  geom_edge_link() +
  geom_node_point()+
```

geom_node_text(aes(label = name), vjust = 1, hjust = 1)



ANÁLISE DE SENTIMENTOS

Carrega biblioteca

```
library(tm)
library(tidytext)
library(tidyverse)
library(DT)
library(wordcloud)
library(tidytext)
library(dplyr)
library(stringr)
library(tibble)
library(tidyr)
```

Carregando Dados

```
dados <- read.csv("bar.csv", sep = ";", header = TRUE)
```

```
View(dados)
```

Adicionando uma coluna com ID aleatória para cada linha

```
dados <- dados %>%
  mutate(id = paste0("ID", 1:n()))
```

Ver Dados

```
View(dados)
```

Carregamento dicionário lexico portugues

```
install.packages("lexiconPT")
library(lexiconPT)
ls("package:lexiconPT")
```

Faz o pré-processamento do texto

```
dados$Comentario <- dados$Comentario %>%
  str_to_lower() %>%                    # Caixa baixa.
  str_replace_all(" *-+ *", "") %>%       # Remove hífen.
  str_replace_all("[[:punct:]]", " ") %>% # Pontuação por espaço.
  removeNumbers() %>%                     # Remove números.
  trimws()                              # Remove espaços nas bordas.
```

Stop words padrão do idioma português

```
stop_words_pt <- stopwords(kind = "pt")
```

Remoção das stop words

```
dados$Comentario <- dados$Comentario %>%
  removeWords(words = c("bom", "muito", "pouco", stop_words_pt))
```

Filtra documentos vazios ou com menos de 1 palavra

```
dados <- dados %>%
  filter(nchar(Comentario) > 0)
```

Faz tokenização nas palavras individuais e empilha as palavras

```
texto_un <- dados %>%
  unnest_tokens(output = "words", input = Comentario)
```

Tokenização:

Tokens: São as unidades de análise que resultam da tokenização. No contexto de palavras, os tokens são palavras individuais.

Objetivo: Facilitar a análise de texto ao dividir o texto em partes menores e mais manejáveis, o que permite realizar operações como contagem de palavras, análise de frequência, e outros tipos de processamento textual.

Exemplo:

Considere o texto: "O cachorro correu no parque."

Texto Original: "O cachorro correu no parque."
Tokens (Palavras): "O", "cachorro", "correu", "no", "parque"

Verifica se a tokenização foi bem-sucedida

```
if (nrow(texto_un) == 0) {
  warning("A tokenização não produziu resultados. Verifique os dados de entrada.")
} else {
  print(texto_un)
}
```

	id	words
1	ID1	fui
2	ID1	bar
3	ID1	rosa
4	ID1	uns
5	ID1	2
6	ID1	3
7	ID1	meses
8	ID1	atrás
9	ID1	não
10	ID1	pensei

Operações para Determinar Polaridade

Uma amostra do dicionário de termos rotulados.

```
sample_n(oplexicon_v3.0, size = 20) %>%
  arrange(polarity)
```

Contagem por polaridade.

```
oplexicon_v3.0 %>%
  count(polarity, sort = TRUE)
```

Faz junção dos de termos de duas tabelas diferentes

```
tb_sen <- inner_join(texto_un,
                     oplexicon_v3.0[, c("term", "polarity")],
                     by = c("words" = "term"),
                     relationship = "many-to-many")
```

Explicação:

inner_join():

Função: Realiza a junção de dois data.frames com base em uma chave comum. No caso de uma junção interna (inner_join), o resultado contém apenas as linhas que têm correspondências em ambos os data.frames.

Argumentos da Função:

texto_un: O data.frame que contém as palavras tokenizadas.

oplexicon_v3.0[, c("term", "polarity")]: O data.frame de lexicon de polaridade, contendo as colunas term e polarity. Aqui, você está selecionando apenas essas duas colunas.

term: Coluna que contém as palavras (ou termos) para correspondência.

polarity: Coluna que contém a polaridade associada a cada termo.

by = c("words" = "term"): Especifica que a junção deve ser feita com base na correspondência entre a coluna words de texto_un e a coluna term de oplexicon_v3.0.

relationship = "many-to-many": Especifica o tipo de relação entre os data.frames, indicando que cada palavra em texto_un pode corresponder a múltiplas entradas em oplexicon_v3.0 e vice-versa. Note que o argumento relationship pode não ser necessário e não é suportado por todas as versões ou pacotes, então é importante verificar a documentação da função e da versão do pacote que você está usando.

Agora o termos tem sua polaridade presente na tabela.
sample_n(tb_sen, size = 20)

```
> sample_n(tb_sen, size = 20)
      id          words polarity
1   ID61       melhorar        0
2  ID407     requintado        1
3  ID235         comida       -1
4  ID413          novas        1
5   ID96         justos        1
6  ID119    maravilhosa        1
7  ID286 vanguardistas        1
8   ID44      excelente        1
```

Faz a agregação da polaridade por documento.

```
tb <- tb_sen %>%
  group_by(id) %>%
  summarise(soma = sum(polarity),
            n = n(),
            sentiment = soma/n)
tb
```

	id	soma	n	sentiment
	<chr>	<int>	<int>	<dbl>
1	ID1	1	3	0.333
2	ID10	2	6	0.333
3	ID100	-2	5	-0.4
4	ID101	0	2	0
5	ID102	0	2	0
6	ID103	-1	4	-0.25
7	ID104	2	4	0.5
8	ID105	4	9	0.444
9	ID106	2	4	0.5
10	ID107	-1	2	-0.5

Explicação:

group_by(id): **Função**: Agrupa o data.frame tb_sen por uma variável de identificação id. Isso significa que as operações subsequentes serão realizadas dentro de cada grupo definido por id.

summarise(): **Função**: Resumir os dados de cada grupo criado pelo group_by(). Você pode calcular estatísticas agregadas para cada grupo.

Cálculos:

soma = sum(polarity): Calcula a soma das polaridades para cada grupo. polaridade é a coluna que contém os valores de polaridade dos tokens.

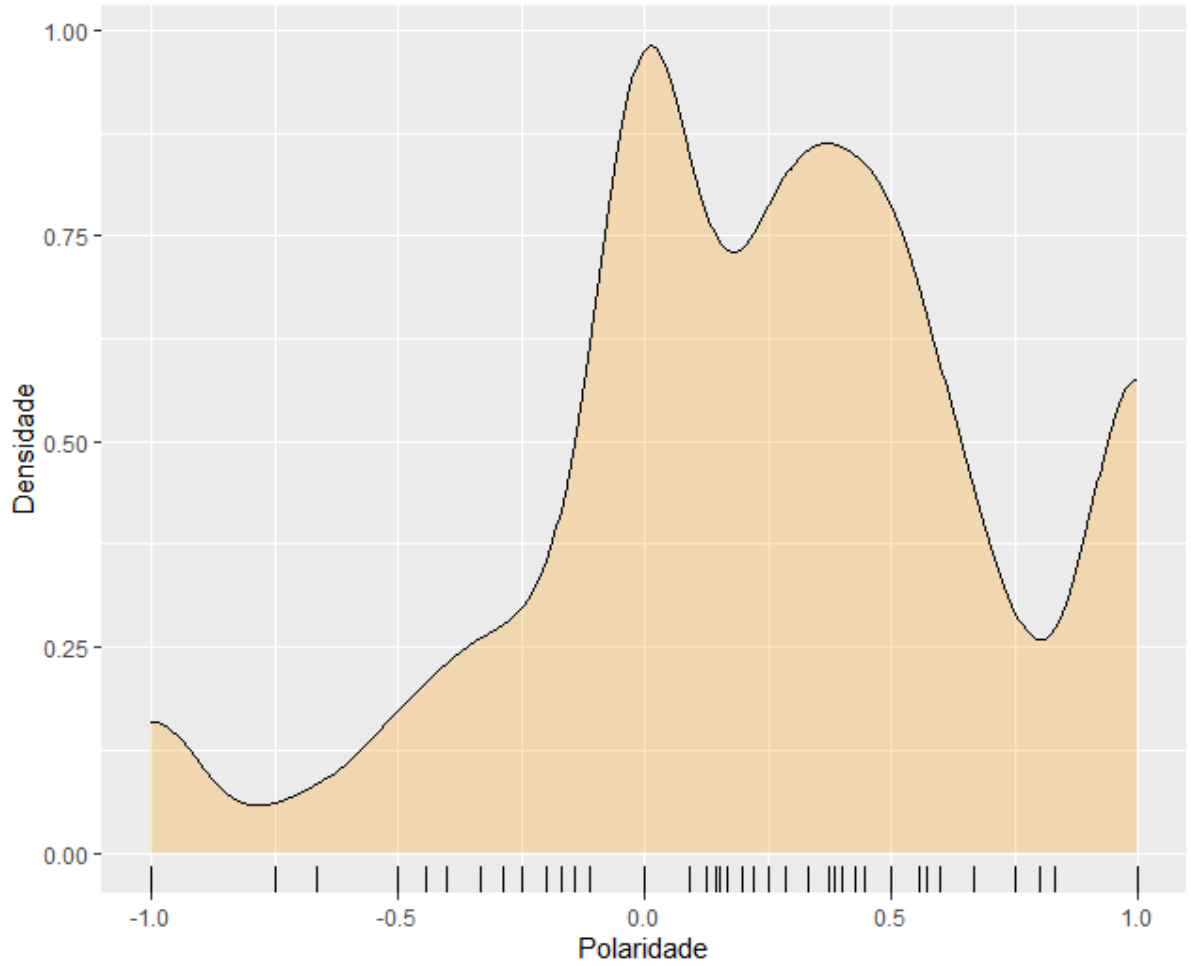
n = n(): Conta o número de registros (ou tokens) em cada grupo.

sentiment = soma / n: Calcula a média da polaridade para cada grupo, que representa o sentimento médio. Isso é feito dividindo a soma das polaridades pelo número de registros.

tb: **Resultado**: Um novo data.frame onde cada linha corresponde a um grupo id, com a soma das polaridades, a contagem de tokens e o sentimento médio calculado.

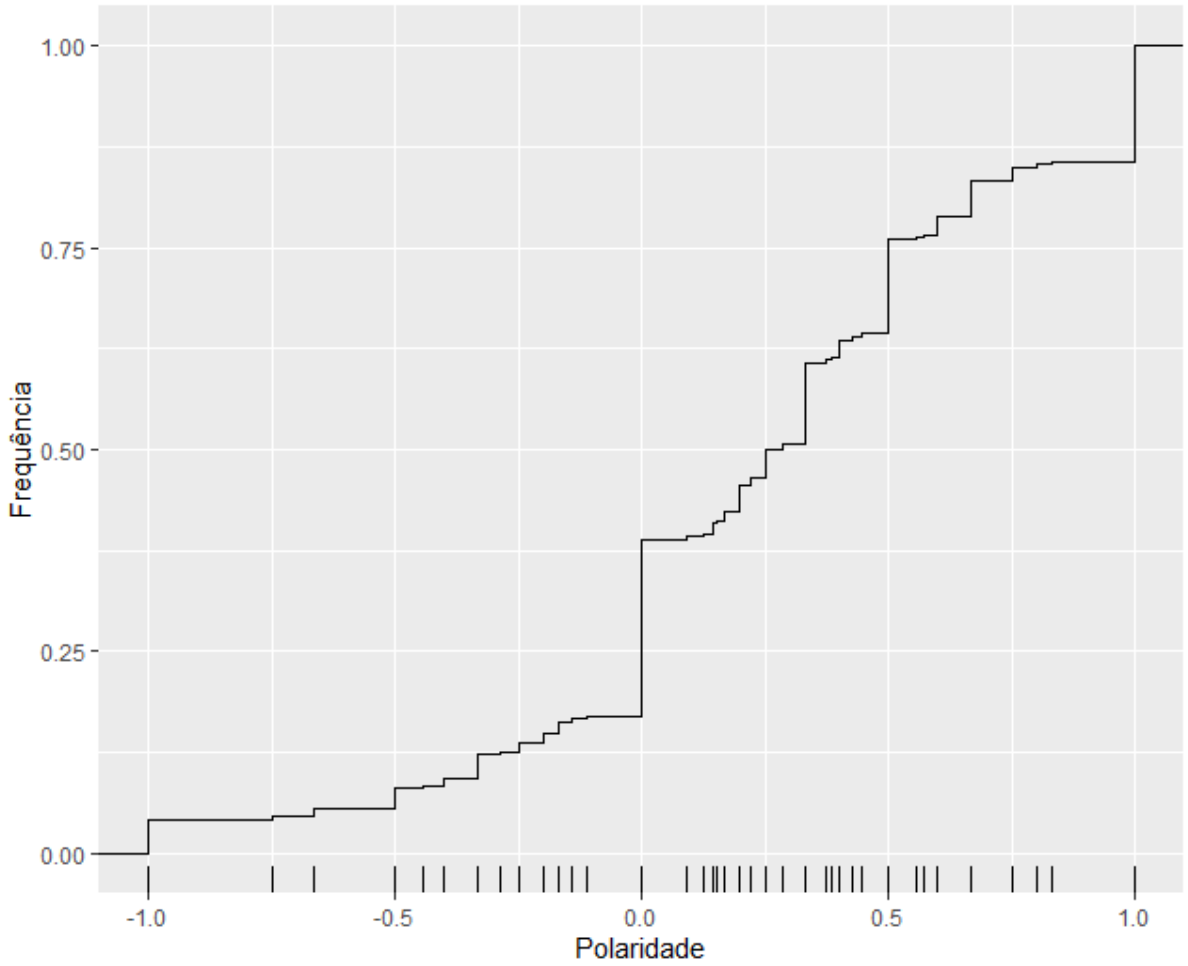
Desidade expírica kernel do escore de sentimento.

```
ggplot(tb, aes(x = sentiment)) +
  geom_density(fill = "orange", alpha = 0.25) +
  geom_rug() +
  labs(x = "Polaridade", y = "Densidade")
```



Frequência relativa acumulada.

```
ggplot(tb, aes(x = sentiment)) +
  stat_ecdf() +
  geom_rug() +
  labs(x = "Polaridade", y = "Frequência")
```



As avaliações mais positivas.

```
tb %>%
  top_n(sentiment, n = 10) %>%
  inner_join(dados[, c("id", "Comentario")]) %>%
  select(sentiment, Comentario)
```

```
   sentiment Comentario
       <dbl> <chr>
 1         1 "   tudo  sempre  gostoso"
 2         1 "chegamos cedo  lá       pegar fila  havia mesas porém  …
 3         1 "porções bem feitas  preço justo   atendimento   voltari…
 4         1 " melhor bar  bauru   esquina super gostosa  drinks exce…
 5         1 "drinques deliciosos  moqueca baiana   perfeita  atendim…
 6         1 "atendimento rapido  porçao  bolinho  mandioquinha  shim…
 7         1 "ambiente temático  pratos  qualidade  preço justo  aten…
 8         1 "excelente atendimento  bons petiscos"
 9         1 " lugar mt gostoso  confortável      namorado   rinite  …
10         1 "porções generosas  cardápio interessante"
```

As avaliações mais negativas.

```
tb %>%
  top_n(sentiment, n = -100) %>%
  inner_join(dados[, c("id", "Comentario")]) %>%
```

```
  select(sentiment, Comentario)
```

```
   sentiment Comentario
       <dbl> <chr>
 1    -0.4    " ruim  porção  bolinho  carne seca extremamente oleoso …
 2     0      "culinária  tanto quanto  tempero  ambiente totalmente d…
 3     0      "comida excelente  atendimento rápido  cortez"
 4    -0.25   " piores músicas  playlist  comida ainda é boa  pena  …
 5    -0.5    "achei  porções pequenas  cerveja  cara  relação  outro…
 6    -0.333  "  porém  preços  elevados  atendimento é superrrr dem…
 7    -0.333 "amo  lugar  comida  bebidas maravilhosos"
 8     0      " ambiente é agradável  comida é sensacional  drinks"
 9     0      "comida ótima  preço  porção justa  almoços"
10    -0.5    "dê uns tempos pra cá  atendimento  ruim  comida  gost…
```

Exibição dos resultados.

```
# Tabela com as avaliações originais sem o preprocessamento.
tb_u <- tb %>%
  inner_join(dados[, c("id", "Comentario")]) %>%
  select(id, sentiment, Comentario)
```

Explicação
`inner_join(dados[, c("id", "Comentario")], by = "id")`:

`inner_join()`: Realiza uma junção interna entre `tb` e uma seleção de colunas do `data.frame` `dados`.

`dados[, c("id", "Comentario")]`: Seleciona as colunas `id` e `Comentario` do `data.frame` `dados`.

`by = "id"`: Especifica a coluna comum para a junção, que é `id`. Isso significa que a junção será feita com base na coluna `id` presente em ambos os `data.frames`.

`select(id, sentiment, Comentario)`:

`select()`: Escolhe as colunas a serem incluídas no `data.frame` resultante.

`id`: Mantém a coluna de identificação.

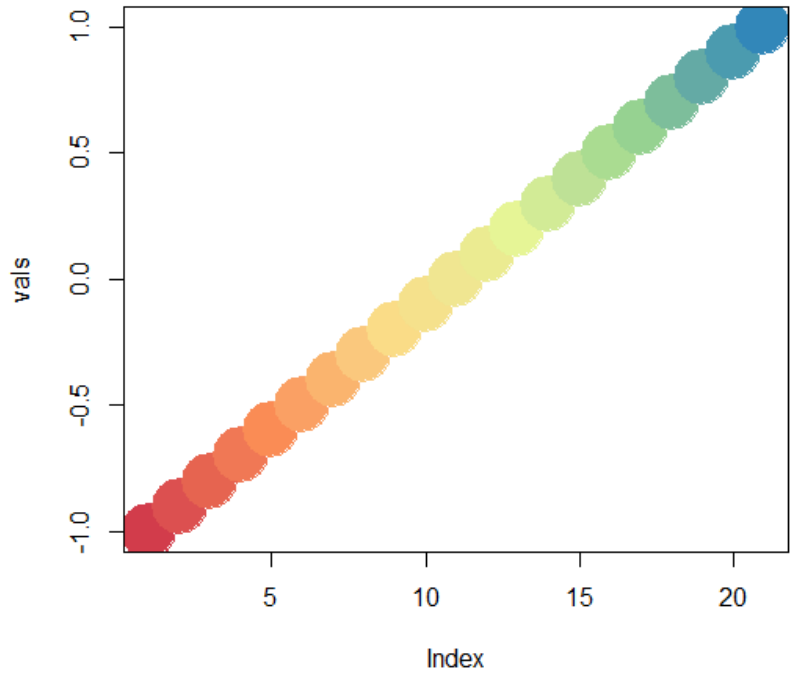
`sentiment`: Inclui a coluna com a média de polaridade calculada anteriormente.

`Comentario`: Inclui a coluna original de comentários.

Valores e cores para formatação condicional das cédulas da tabela.

```
vals <- seq(-1, 1, by = 0.1)  #Gera uma sequência de valores de -1 a 1 com intervalo de 0.1
cols <- colorRampPalette(
  RColorBrewer::brewer.pal(n = 6, name = "Spectral")  # Cria uma paleta de cores com 6 cores da paleta "Spectral"
)(length(vals))  # Aplica a paleta de cores ao comprimento dos valores

# Plota os valores com as cores especificadas
plot(vals, col = cols, pch = 19, cex = 5)
```



Esse gráfico só mostra as cores que aparecerão na pagina que vamos criar com o relatório por frase.

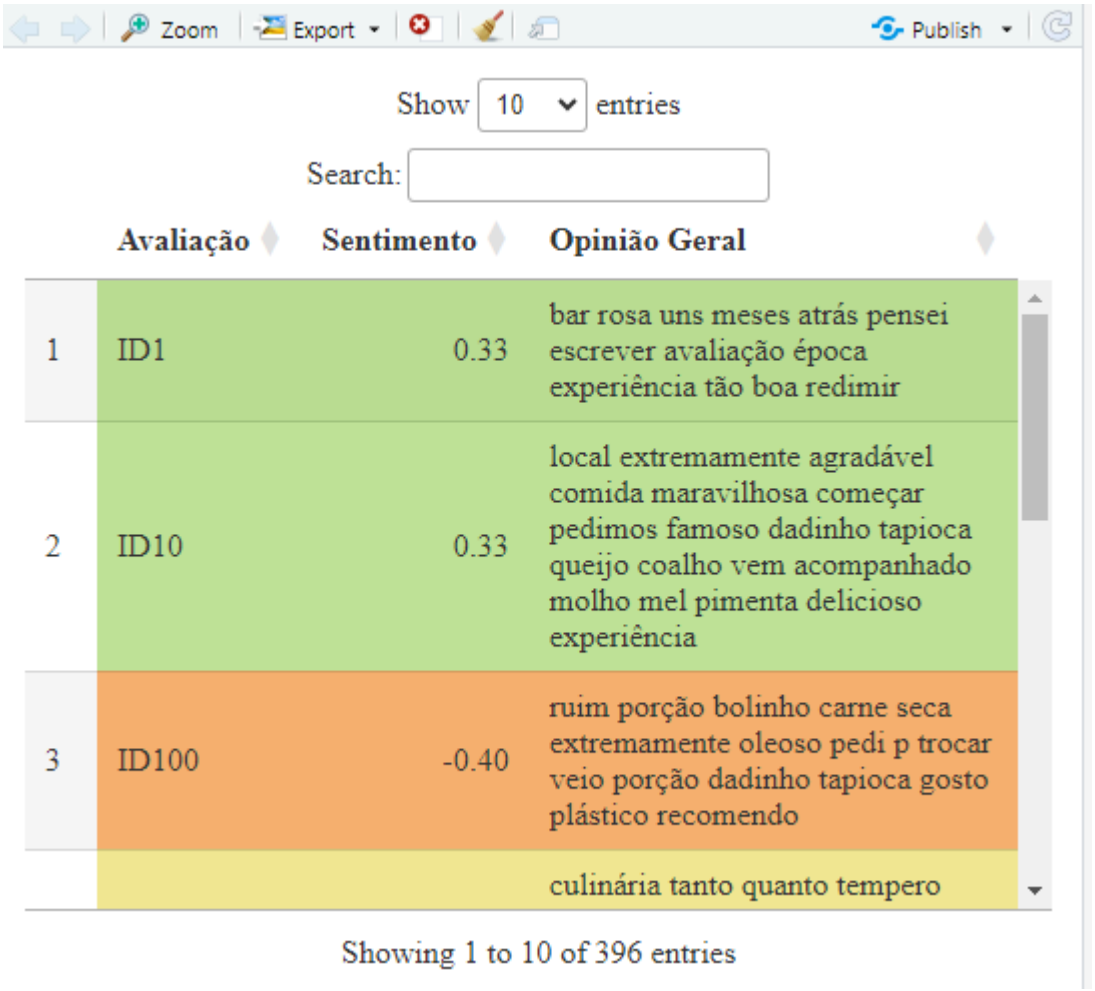
Define o estilo de formatação condicional.

```
style <- styleInterval(cuts = head(vals[-1], n = -1),
                       values = cols[-1])
```

Gera visualização

```
html_table <-
  datatable(tb_u,
            colnames = c("Avaliação",
                         "Sentimento",
                         "Opinião Geral")) %>%
  formatRound(columns = "sentiment", digits = 2) %>%
  formatStyle(columns = names(tb_u),
```

```
        valueColumns = "sentiment",
        target = "cell",
        backgroundColor = style)
html_table
```



Página

Show 10 entries Search:

	Avaliação	Sentimento	Opinião Geral
1	ID1	0.33	bar rosa uns meses atrás pensei escrever avaliação época experiência tão boa redimir
2	ID10	0.33	local extremamente agradável comida maravilhosa começar pedimos famoso dadinho tapioca queijo coalho vem acompanhado molho mel pimenta delicioso experiência
3	ID100	-0.40	ruim porção bolinho carne seca extremamente oleoso pedi p trocar veio porção dadinho tapioca gosto plástico recomendo
4	ID101	0.00	culinária tanto quanto tempero ambiente totalmente despreparado verão passei calor imenso
5	ID102	0.00	comida excelente atendimento rápido cortez
6	ID103	-0.25	piores músicas playlist comida ainda é boa pena bolinhos ultimamente chegam encharcados
7	ID104	0.50	comida sensacional suco feito frutas frescas produtos qualidade
8	ID105	0.44	ambiente é gostoso comida é boa porém achei sistema separar bar goró rosa meio chato pois é preciso ficar saindo atravessando rua pegar drinks entendo ideia achei prático voltaria
9	ID106	0.50	sei escrever sobre bar rosa pq pra mim vai além restaurante é experiência tudo moacir toca vira ouro boca podem pedir qualquer coisa desse lugar resultado indico bolinho feijoada tapioca baião torta búlgara favoritos
10	ID107	-0.50	achei porções pequenas cerveja cara relação outros bares região

Showing 1 to 10 of 396 entries Previous 1 2 3 4 5 … 40 Next

Frequencia de palavra positiva/negativa

Determina as frequências dos termos de polaridade não nula.

```
tb_words <- tb_sen %>%
  count(words, polarity, sort = TRUE) %>%
  filter(polarity != 0)
```

Transforma dados em nuvem de palavras

```
tb_cloud <- tb_words %>%
  spread(key = "polarity", value = "n", fill = 0) %>%
  rename("negative" = "-1", "positive" = "1")
tb_cloud
```

```
> tb_cloud
             words negative positive
1          absurda        1        0
2        abundante        0        1
3         abusivos        1        0
4           aceita        0        1
5        acolhedor        0        4
6     aconchegante        0       25
```

Cria um dataframe com as palavras

```
tb <- as.data.frame(tb_cloud[, c("negative", "positive")])
rownames(tb) <- tb_cloud$words
head(tb)
```

```
  id      soma      n sentiment
  <chr> <int> <int>      <dbl>
1 ID1       1     3      0.333
2 ID10      2     6      0.333
3 ID100    -2     5     -0.4
4 ID101     0     2      0
5 ID102     0     2      0
6 ID103    -1     4     -0.25
```

Faz nuvem de palavras.

```
comparison.cloud(tb,
                 colors = c("red", "blue"),
                 max.words = min(nrow(tb), 200))
```

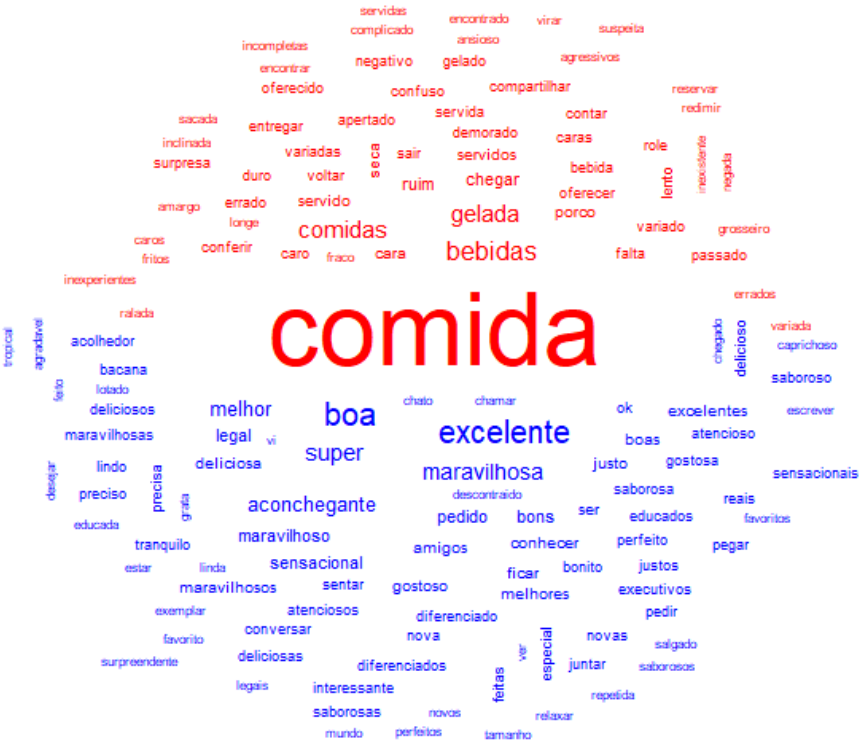
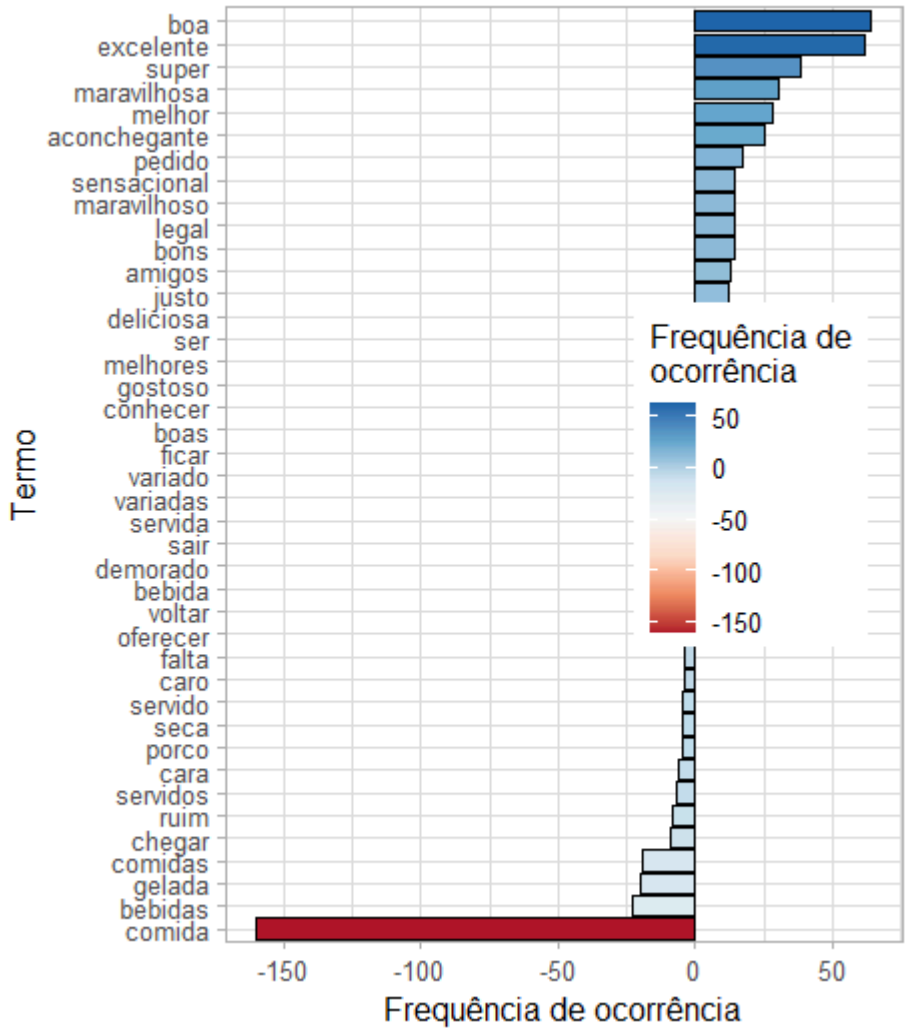


Gráfico de barras para as palavras de maior ocorrência.

```
n_words <- 20
tb_bars <- tb_words %>%
  mutate(score = polarity * n) %>%
  group_by(polarity) %>%
  top_n(n, n = n_words) %>%
  ungroup()
```

Gráfico

```
ggplot(data = tb_bars,
       mapping = aes(x = reorder(words, score),
                     y = score,
                     fill = score)) +
  geom_col(color = "black") +
  scale_fill_distiller(palette = "RdBu", direction = 1) +
  coord_flip() +
  theme_light() +
  theme(legend.position = c(0.95, 0.5),
        legend.justification = c(1, 0.5)) +
  labs(y = "Frequência de ocorrência",
       x = "Termo",
       fill = "Frequência de\nocorrência")
```


boa
excelente
super
maravilhosa
melhor
aconchegante
pedido
sensacional
maravilhoso
legal
bons
amigos
justo
deliciosa
ser
melhores
gostoso
conhecer
boas
ficar
variado
variadas
servida
sair
demorado
bebida
voltar
oferecer
falta
caro
servido
seca
porco
cara
servidos
ruim
chegar
comidas
gelada
bebidas
comida
Termo
Frequência de ocorrência
50
0
-50
-100
-150
-150
-100
-50
0
50
Frequência de ocorrência