Análise de Texto - C.E.2

2022.1 - UnB

Letícia Lino, Lucas Augusto, Lucas Coelho e Pedro Aguiar

31/08/2022

Table of contents I

Section 1

Introdução

Mineração de dados

A mineração de dados (*data mining*) é um processo técnico, automático ou semiautomático, que analisa grandes quantidades de informações dispersas para que tenham sentido e sejam convertidas em conhecimento. Busca anomalias, padrões ou correlações entre milhões de registros para predizer resultados.

Mineração e análise de texto

O termo mineração de texto (*text mining)* é um pouco menos conhecido, mas a ideia é bastante semelhante. A diferença reside principalmente no tipo de dado analisado. Enquanto a de dados lida mais com bancos de dados, o *text mining* faz essa mineração em dados não estruturados.

A análise de texto combina um conjunto de técnicas de aprendizado de máquina, estatísticas e linguísticas para processar grandes volumes de texto não estruturado ou texto que não tem um formato predefinido (Word e PDF's, por exemplo), para derivar percepções e padrões. Textos extraídos de redes sociais também podem ser processados.

A mineração de texto e a análise de texto costumam ser usadas de forma intercambiável. O termo mineração de texto é geralmente usado para derivar percepções qualitativas de texto não estruturado, enquanto a análise de texto fornece resultados quantitativos.Por exemplo, a mineração de texto pode ser usada para identificar se os clientes estão satisfeitos com um produto, analisando suas avaliações e pesquisas. A análise de texto é usada para informações mais profundas, como identificar um padrão ou tendência de um texto não estruturado (para entender um aumento negativo na experiência do cliente ou na popularidade de um produto, por exemplo).

Técnicas de análise de texto e casos de uso

- Análise de sentimentos
- Modelagem de tópicos
- Reconhecimento de entidade nomeada (NER)
- Frequência do termo - frequência inversa do documento (TF-IDF)
- Extração de evento
  - Análise de links
  - Análise geoespacial
  - Monitoramento de riscos de negócios

Etapas da análise de texto

1. Coleta de dados
2. Preparação de dados
    1. Tokenização
    2. Marcação de parte do discurso
    3. Lematização e origem
    4. Remoção de palavras irrelevantes
3. Análise de texto
4. Visualização

Section 2

Mineração e análise de texto de dados não estruturados com *R*

O Formato *Tidy Text*

Serão analisados os seguintes artigos científicos:

- "The Production of Comedy: The Joke in the Age of Social Media" Sturges, Paul
- "The Strives, Struggles, and Successes of Women Diagnosed With ADHD as Adults" Glaser, Mira; Langvik, Eva
- "Social Revolutions: Their Causes, Patterns, and Phases" Tiruneh, Gizachew

Lendo os artigos em PDF no *R*.

```
require(pdftools)
require(stringr)
require(dplyr)

humorepiadas <- pdf_text("A piada na era da mídia social - alt
  str_c(collapse = "") %>%
  str_replace_all(pattern = "\n", replacement = " PAGINA ") %>
  str_squish() %>%
  str_split(pattern = "PAGINA") %>%
  unlist
```

```
revsociais <- pdf_text("Revoluções Sociais - alt.pdf") %>%
  str_c(collapse = "") %>%
  str_replace_all(pattern = "\n", replacement = " PAGINA ") %>
  str_squish() %>%
  str_split(pattern = "PAGINA") %>%
  unlist

tdahmulheres <- pdf_text("Os esforços, lutas e sucessos de mul
  str_c(collapse = "") %>%
  str_replace_all(pattern = "\n", replacement = " PAGINA ") %>
  str_squish() %>%
  str_split(pattern = "PAGINA") %>%
  unlist
```

Transformando em um *dataframe* com "***tidytext***"

```
require(tidytext)

humorepiada_df <- humorepiadas %>%
  as.data.frame() %>%
  mutate(numerolinha = row_number()) %>%
  rename("texto" = ".") %>%
  unnest_tokens(word, texto) %>%
  as_tibble() %>%
  mutate(Autor = "Sturges")
```

```
# A tibble: 5,411 x 3
   numerolinha word       Autor
         <int> <chr>      <chr>
 1           1 the        Sturges
 2           1 production Sturges
 3           1 of         Sturges
 4           1 comedy     Sturges
 5           1 the        Sturges
 6           1 joke       Sturges
 7           1 in         Sturges
 8           2 the        Sturges
 9           2 age        Sturges
10           2 of         Sturges
# ... with 5,401 more rows
# i Use `print(n = ...)` to see more rows
```

```
revsociais_df <- revsociais %>%
  as.data.frame() %>%
  mutate(numerolinha = row_number()) %>%
  rename("texto" = ".") %>%
  unnest_tokens(word, texto) %>%
  as_tibble() %>%
  mutate(Autor = "Tiruneh")
```

```
# A tibble: 8,856 x 3
   numerolinha word        Autor
         <int> <chr>       <chr>
 1           1 social      Tiruneh
 2           1 revolutions Tiruneh
 3           1 their       Tiruneh
 4           1 causes      Tiruneh
 5           1 patterns    Tiruneh
 6           2 and         Tiruneh
 7           2 phases      Tiruneh
 8           4 gizachew    Tiruneh
 9           4 tiruneh1    Tiruneh
10           8 abstract    Tiruneh
# ... with 8,846 more rows
# i Use `print(n = ...)` to see more rows
```

```
tdahmulheres_df <- tdahmulheres %>%
  as.data.frame() %>%
  mutate(numerolinha = row_number()) %>%
  rename("texto" = ".") %>%
  unnest_tokens(word, texto) %>%
  as_tibble() %>%
  mutate(Autor = "Glaser and Langvik")
```

```
# A tibble: 8,714 x 3
   numerolinha word      Autor
         <int> <chr>     <chr>
 1           1 the       Glaser and Langvik
 2           1 strives   Glaser and Langvik
 3           1 struggles Glaser and Langvik
 4           1 and       Glaser and Langvik
 5           1 successes Glaser and Langvik
 6           1 of        Glaser and Langvik
 7           1 mira      Glaser and Langvik
 8           1 elise     Glaser and Langvik
 9           1 glaser    Glaser and Langvik
10           2 holthe1   Glaser and Langvik
# ... with 8,704 more rows
# i Use `print(n = ...)` to see more rows
```

Removendo palavras "vazias"

```
data(stop_words)
humorepiada_df <- humorepiada_df %>%
  anti_join(stop_words)

data(stop_words)
revsociais_df <- revsociais_df %>%
  anti_join(stop_words)

data(stop_words)
tdahmulheres_df <- tdahmulheres_df %>%
  anti_join(stop_words)
```

Contagem - Humor & Piada

```
# A tibble: 1,273 x 2
   word           n
   <chr>      <int>
 1 comedy        70
 2 jokes         42
 3 material      41
 4 media         35
 5 comedians     34
 6 comic         25
 7 social        22
 8 comedian      20
 9 joke          19
10 content       17
# ... with 1,263 more rows
# i Use `print(n = ...)` to see more rows
```

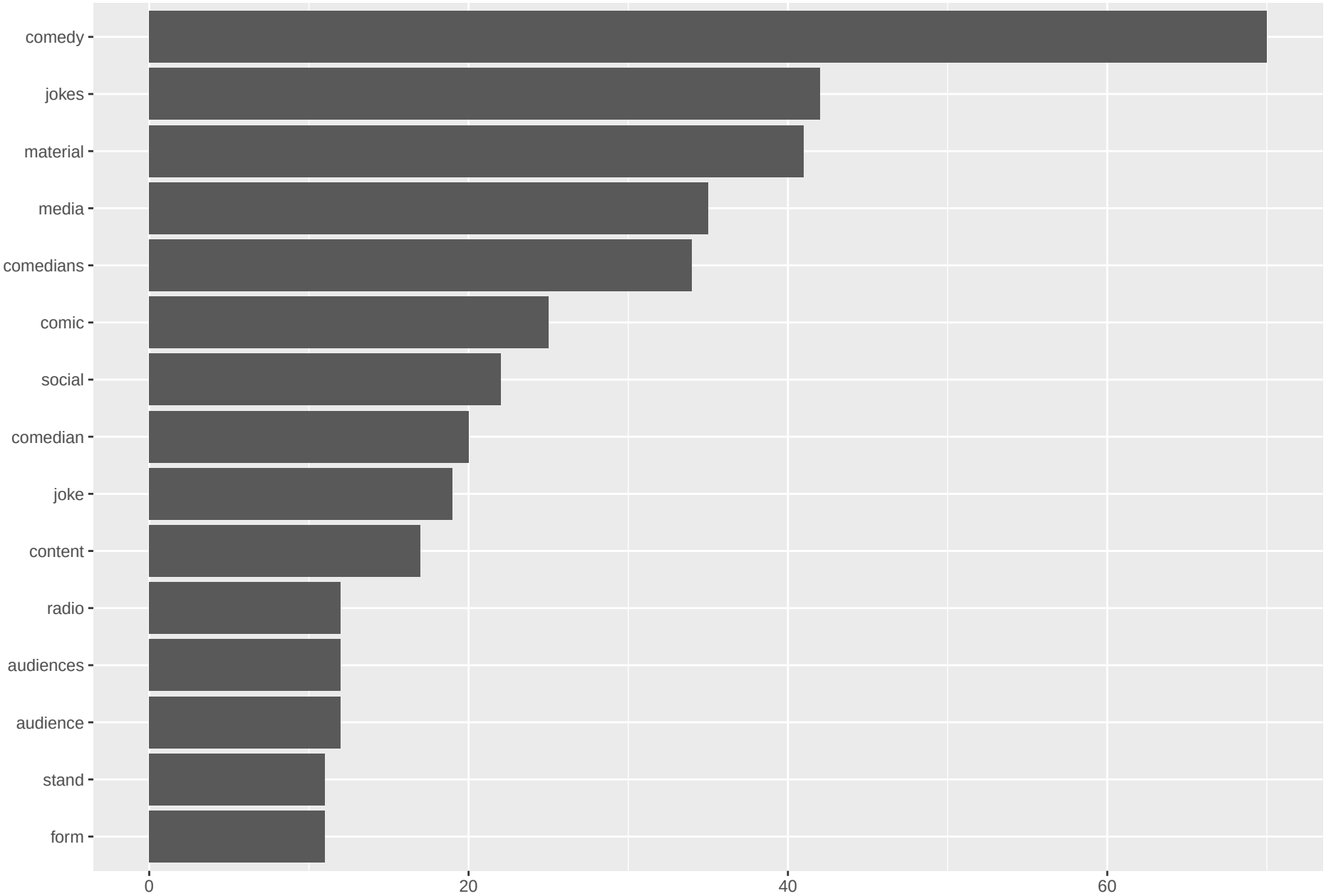
Contagem - Revoluções Sociais

```
# A tibble: 1,318 x 2
   word             n
   <chr>        <int>
 1 revolution     169
 2 revolutions    116
 3 political       79
 4 social          66
 5 economic        63
 6 people          38
 7 onset           36
 8 revolutionary   36
 9 planned         35
10 development     32
# ... with 1,308 more rows
# i Use `print(n = ...)` to see more rows
```

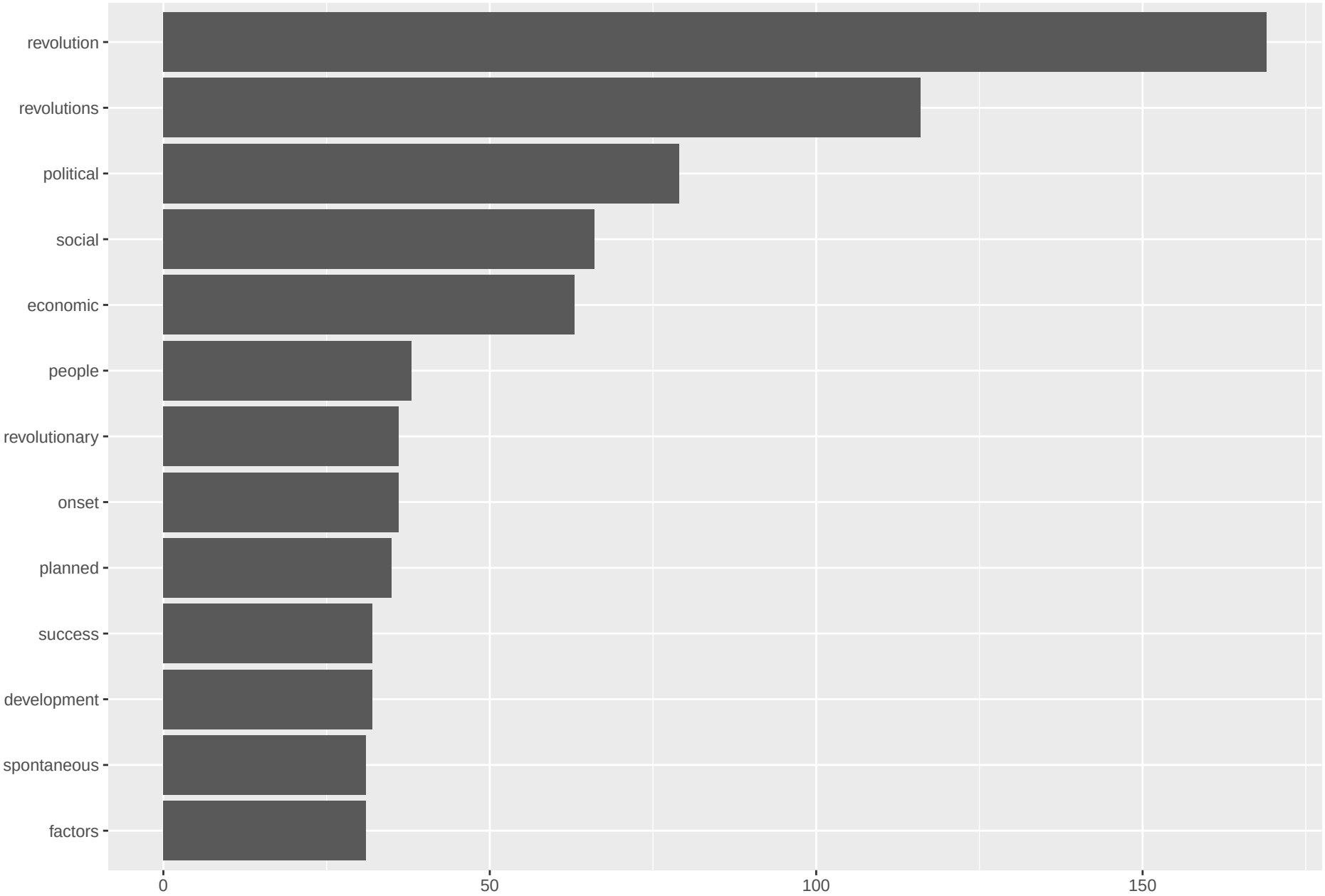
Contagem - TDAH em Mulheres

```
# A tibble: 1,421 x 2
   word             n
   <chr>        <int>
 1 adhd           190
 2 women           74
 3 al              46
 4 diagnosis       42
 5 symptoms        41
 6 participants    37
 7 stigma          30
 8 difficulties    28
 9 diagnosed       25
10 time            25
# ... with 1,411 more rows
# i Use `print(n = ...)` to see more rows
```

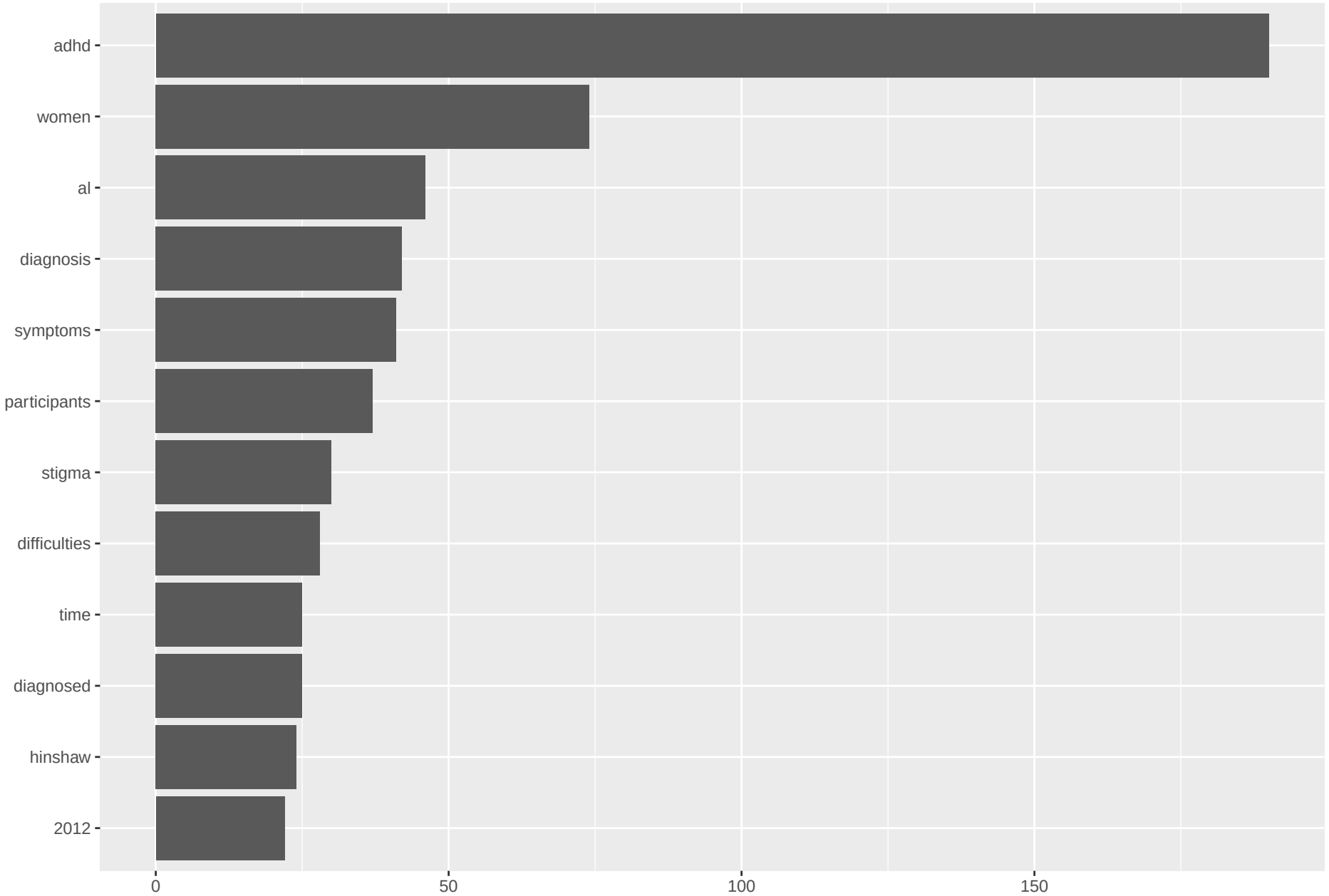
Visualização - Humor & Piada



Visualização - Revoluções Sociais



Visualização - TDAH em Mulheres



Análise de sentimentos

Bibliotecas para análise de sentimentos com "*textdata*"

"*Afinn*"

```
require(textdata)
get_sentiments("afinn")
```

```
# A tibble: 2,477 x 2
   word       value
   <chr>      <dbl>
 1 abandon       -2
 2 abandoned     -2
 3 abandons      -2
 4 abducted      -2
 5 abduction     -2
 6 abductions    -2
 7 abhor         -3
 8 abhorred      -3
 9 abhorrent     -3
10 abhors        -3
# ... with 2,467 more rows
```

"*Bing*"

```
# A tibble: 6,786 x 2
   word        sentiment
   <chr>       <chr>
 1 2-faces     negative
 2 abnormal    negative
 3 abolish     negative
 4 abominable  negative
 5 abominably  negative
 6 abominate   negative
 7 abomination negative
 8 abort       negative
 9 aborted     negative
10 aborts      negative
# ... with 6,776 more rows
# i Use `print(n = ...)` to see more rows
```

"NRC"

```
# A tibble: 13,872 x 2
   word        sentiment
   <chr>       <chr>
 1 abacus      trust
 2 abandon     fear
 3 abandon     negative
 4 abandon     sadness
 5 abandoned   anger
 6 abandoned   fear
 7 abandoned   negative
 8 abandoned   sadness
 9 abandonment anger
10 abandonment fear
# ... with 13,862 more rows
# i Use `print(n = ...)` to see more rows
```

"*NRC* - Revoluções Sociais - Medo"

```
nrc_medo <- get_sentiments("nrc") %>%
  filter(sentiment == "fear")
```

```
# A tibble: 56 x 2
   word           n
   <chr>      <int>
 1 revolution   169
 2 military      29
 3 violence      14
 4 collapse      11
 5 government    11
 6 uprising      11
 7 violent        9
 8 discontent     8
 9 war            7
10 change         6
# ... with 46 more rows
# i Use `print(n = ...)` to see more rows
```

"*BING* - Humor & Piadas - Positivo"

```
# A tibble: 77 x 2
   word             n
   <chr>        <int>
 1 effectively      7
 2 variety          7
 3 authentic        5
 4 popular          5
 5 fresh            4
 6 success          4
 7 contribution     3
 8 humorous         3
 9 novelty          3
10 precisely        3
# ... with 67 more rows
# i Use `print(n = ...)` to see more rows
```

"*AFINN* - TDAH em Mulheres - 3 negativo"

```
# A tibble: 16 x 2
   word          n
   <chr>     <int>
 1 guilt         6
 2 bad           5
 3 fake          5
 4 abuse         3
 5 lost          3
 6 worrying      3
 7 anger         2
 8 boring        2
 9 dumb          2
10 losing        2
11 worried       2
12 worry         2
13 angry         1
14 panic         1
```

Palavras positivas e negativas - Quantidade

Humor & Piadas

```
humorepiada_sent_bing <- humorepiada_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("bing")) #jokes é negativo; não te

humorepiada_sent_afinn <- humorepiada_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("afinn")) # jokes é 2; comedy é 1;

humorepiada_sent_nrc <- humorepiada_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("nrc")) # jokes é negativo; não te
```

Revoluções Sociais

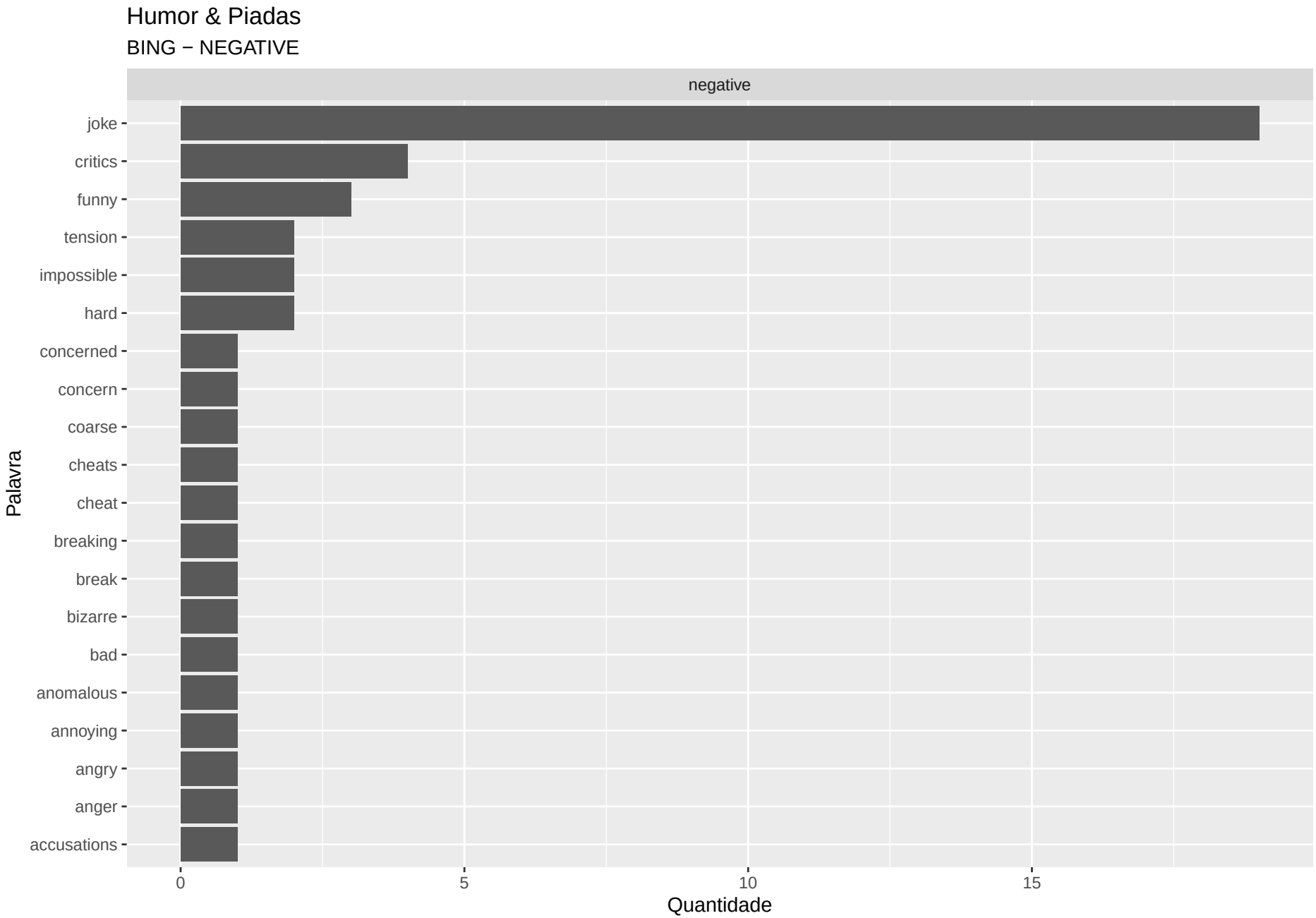
```
revsociais_sent_bing <- revsociais_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("bing")) #tem revolutionary mas nã
# defeat é positivo;

revsociais_sent_afinn <- revsociais_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("afinn")) # não tem revolution - p
#talvez algumas palavras não tem classificação?

revsociais_sent_nrc <- revsociais_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("nrc")) #revolution tem vários ser
#defeat é negativo
```

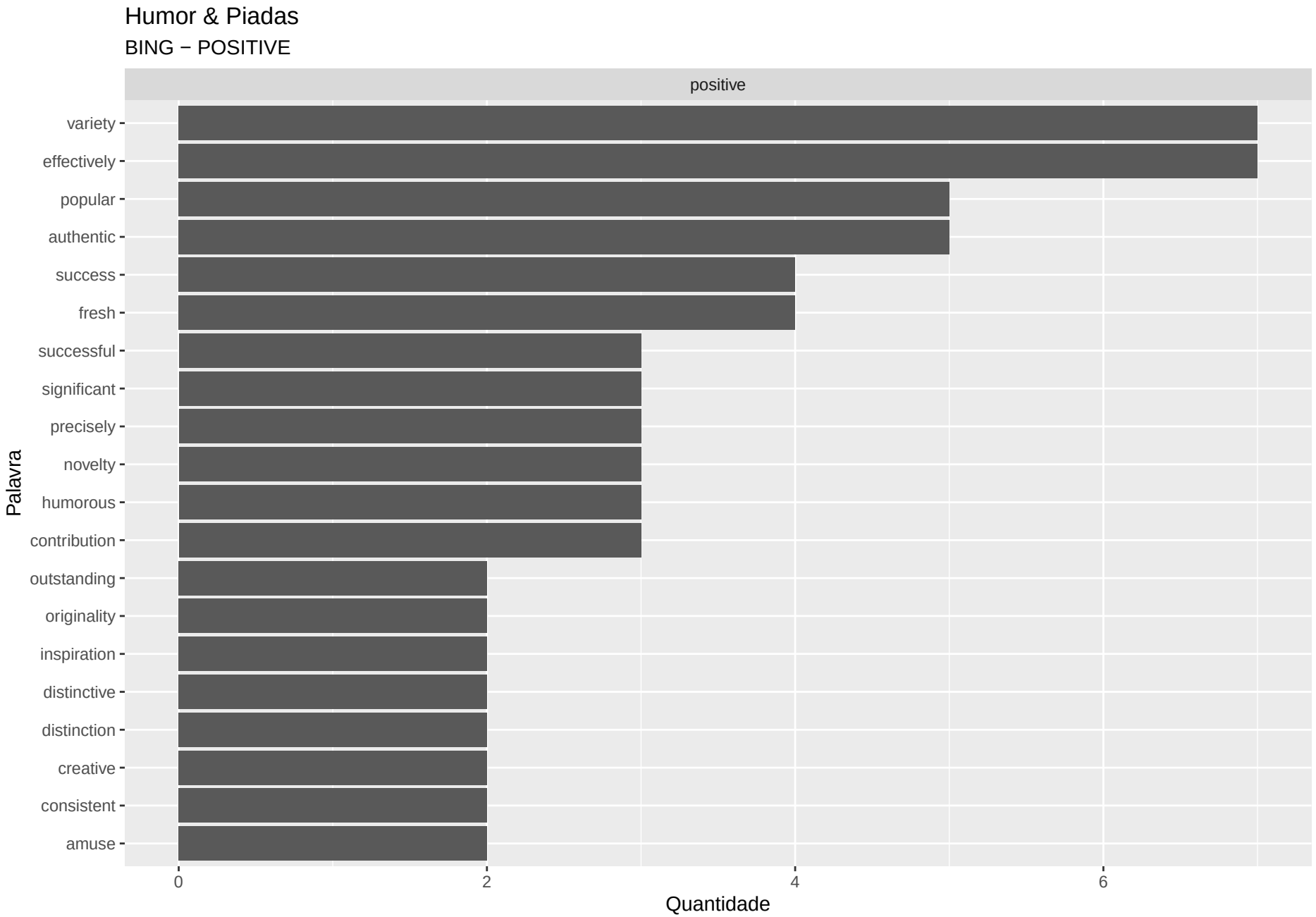
TDAH em Mulheres

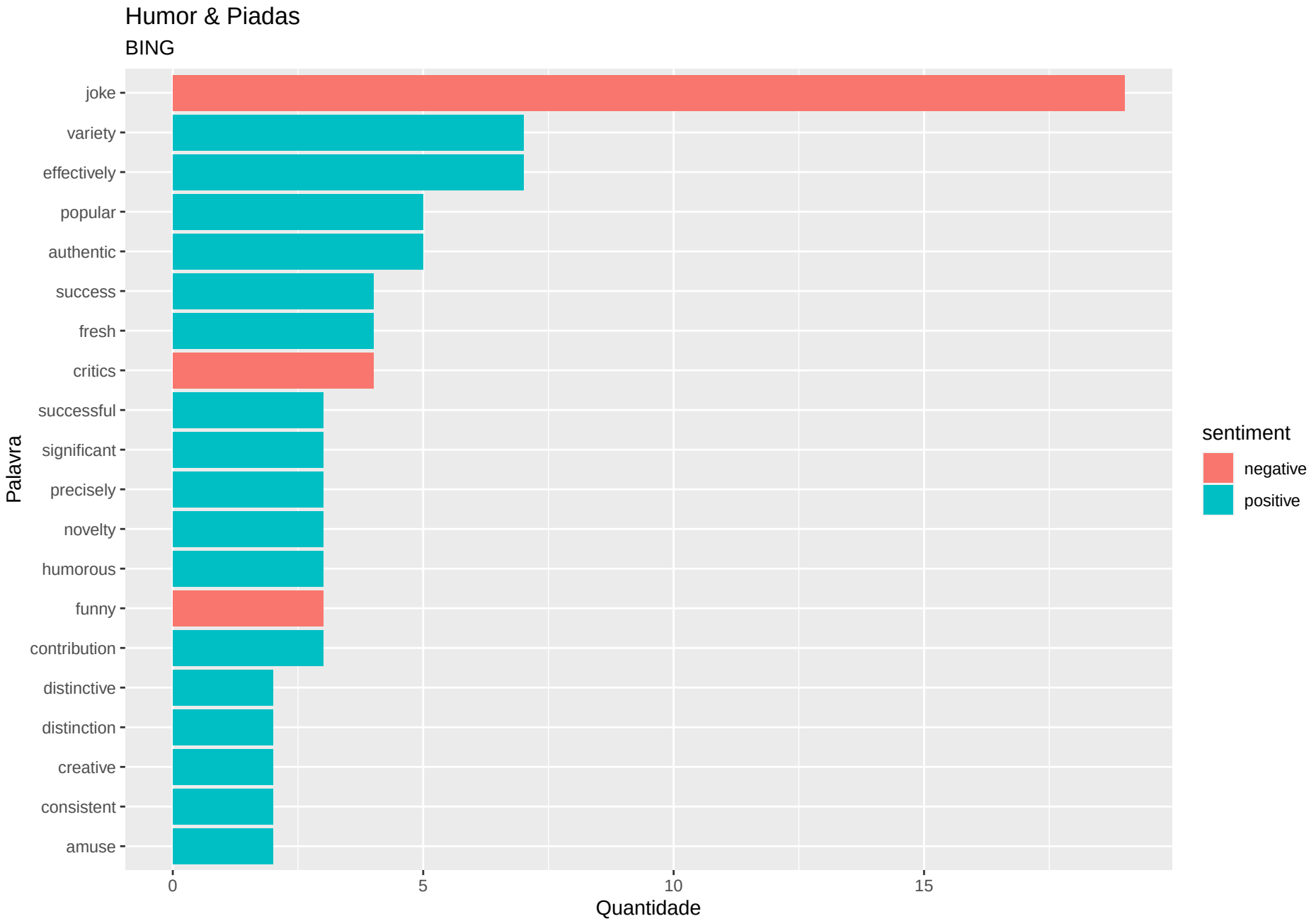
```
tdahmulheres_sent_bing <- tdahmulheres_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("bing")) #symptoms é negativo; lac
#a maioria das palavras mais frequentes não aparecem também

tdahmulheres_sent_afinn <- tdahmulheres_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("afinn")) #as 10 palavras mais fre
#positive é 2 e negative é -2;

tdahmulheres_sent_nrc <- tdahmulheres_df %>%
  count(word, sort = TRUE) %>%
  inner_join(get_sentiments("nrc")) #multisentimentalidade; es
```

Visualização

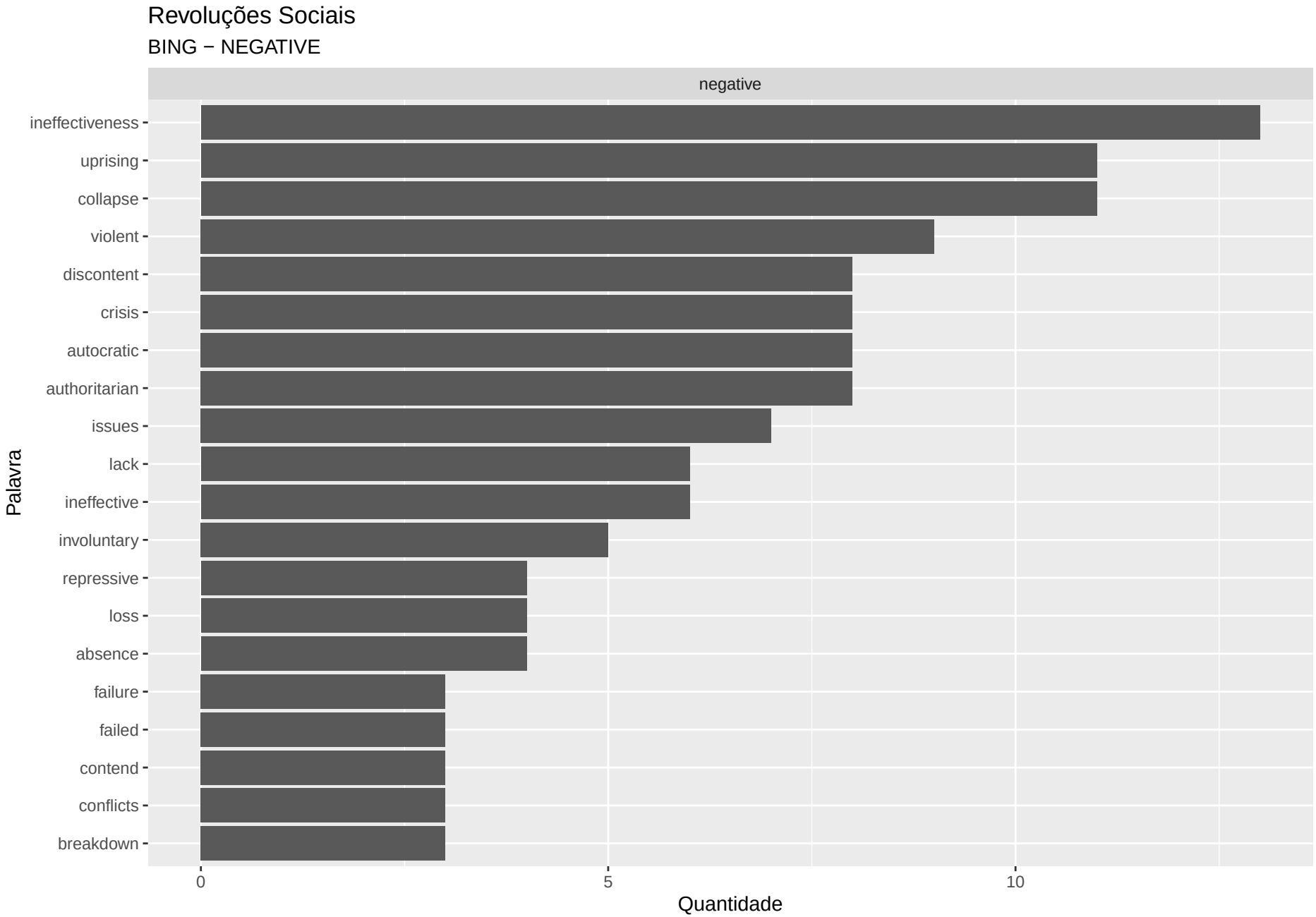
Bing - Humor & Piadas

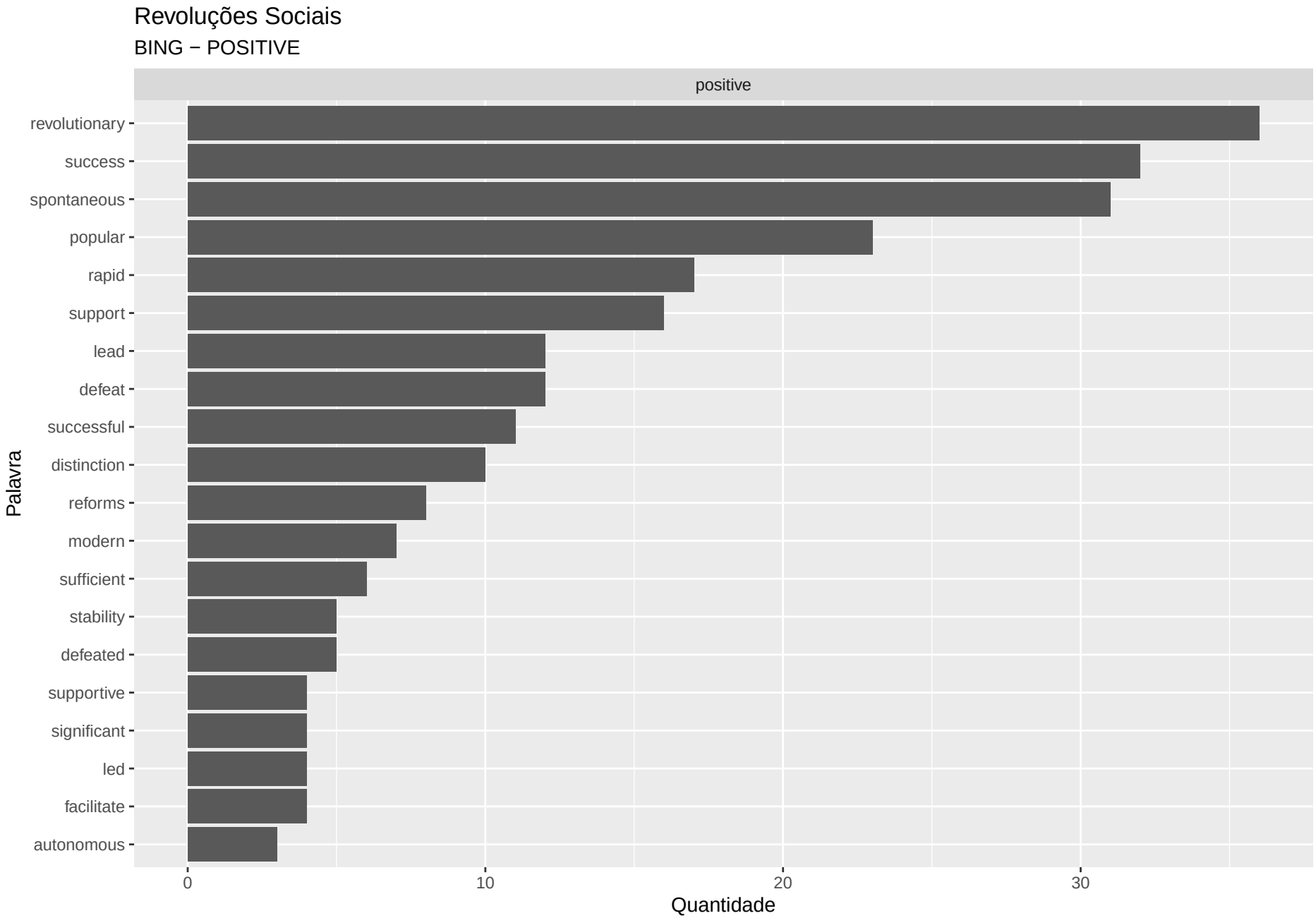

Humor & Piadas
BING − NEGATIVE
negative
joke
critics
funny
tension
impossible
hard
concerned
concern
coarse
cheats
cheat
breaking
break
bizarre
bad
anomalous
annoying
angry
anger
accusations
Palavra
0
5
10
15
Quantidade

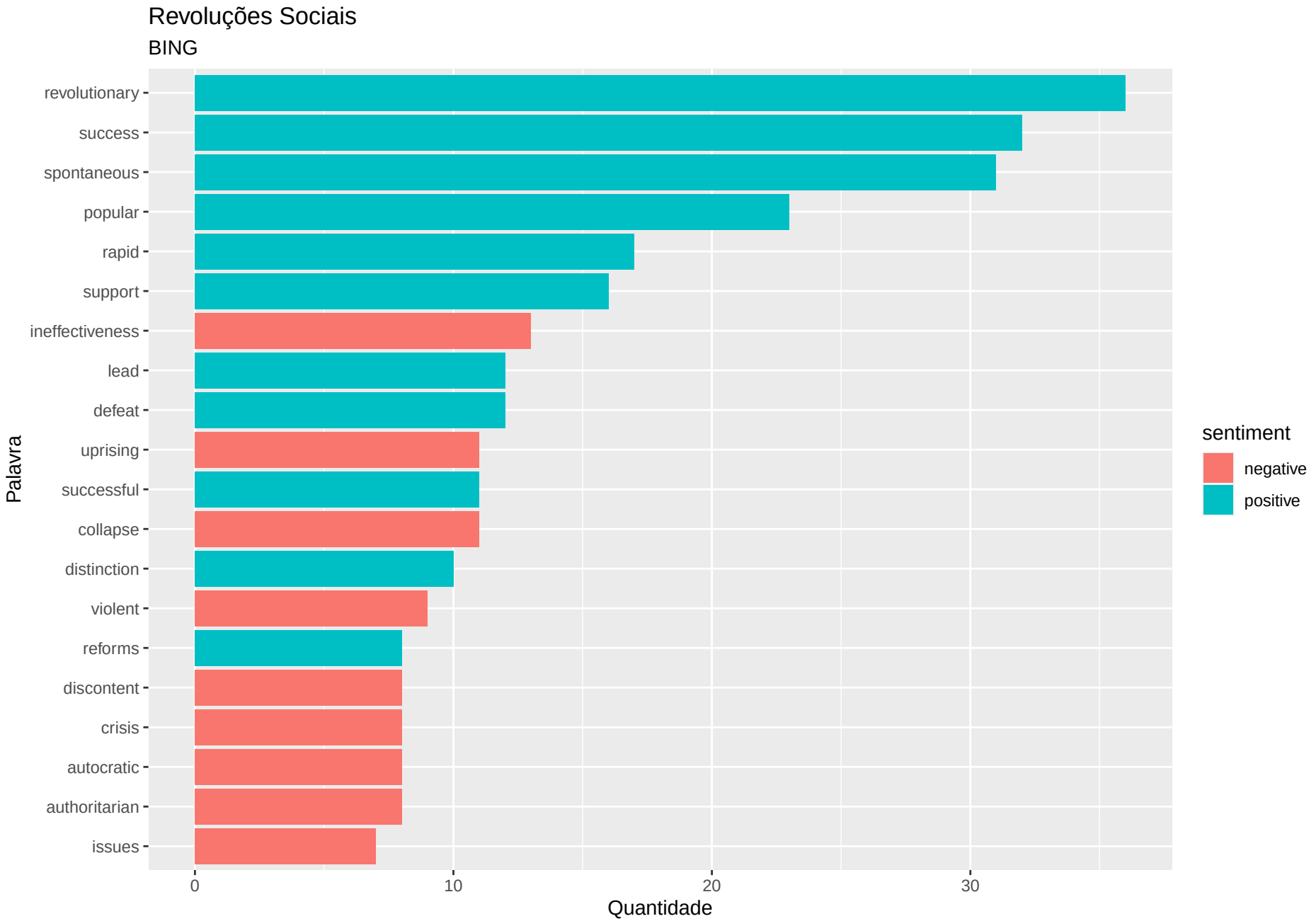

Humor & Piadas
BING − POSITIVE
positive
variety
effectively
popular
authentic
success
fresh
successful
significant
precisely
novelty
humorous
contribution
outstanding
originality
inspiration
distinctive
distinction
creative
consistent
amuse
Palavra
0
2
4
6
Quantidade


Humor & Piadas
BING
joke
variety
effectively
popular
authentic
success
fresh
critics
successful
significant
precisely
novelty
humorous
funny
contribution
distinctive
distinction
creative
consistent
amuse
Palavra
0
5
10
15
Quantidade
sentiment
negative
positive

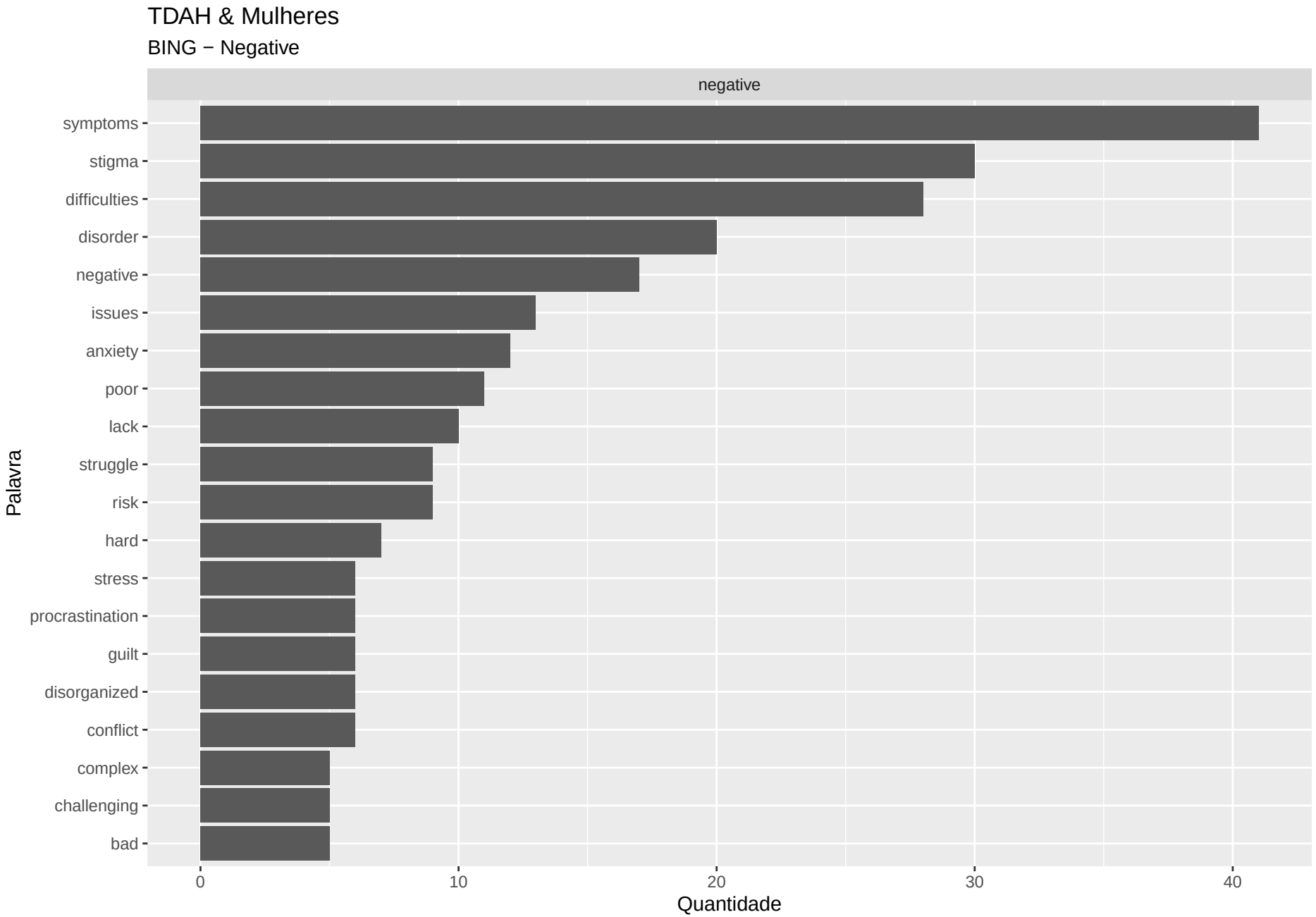
BING - Revoluções Sociais

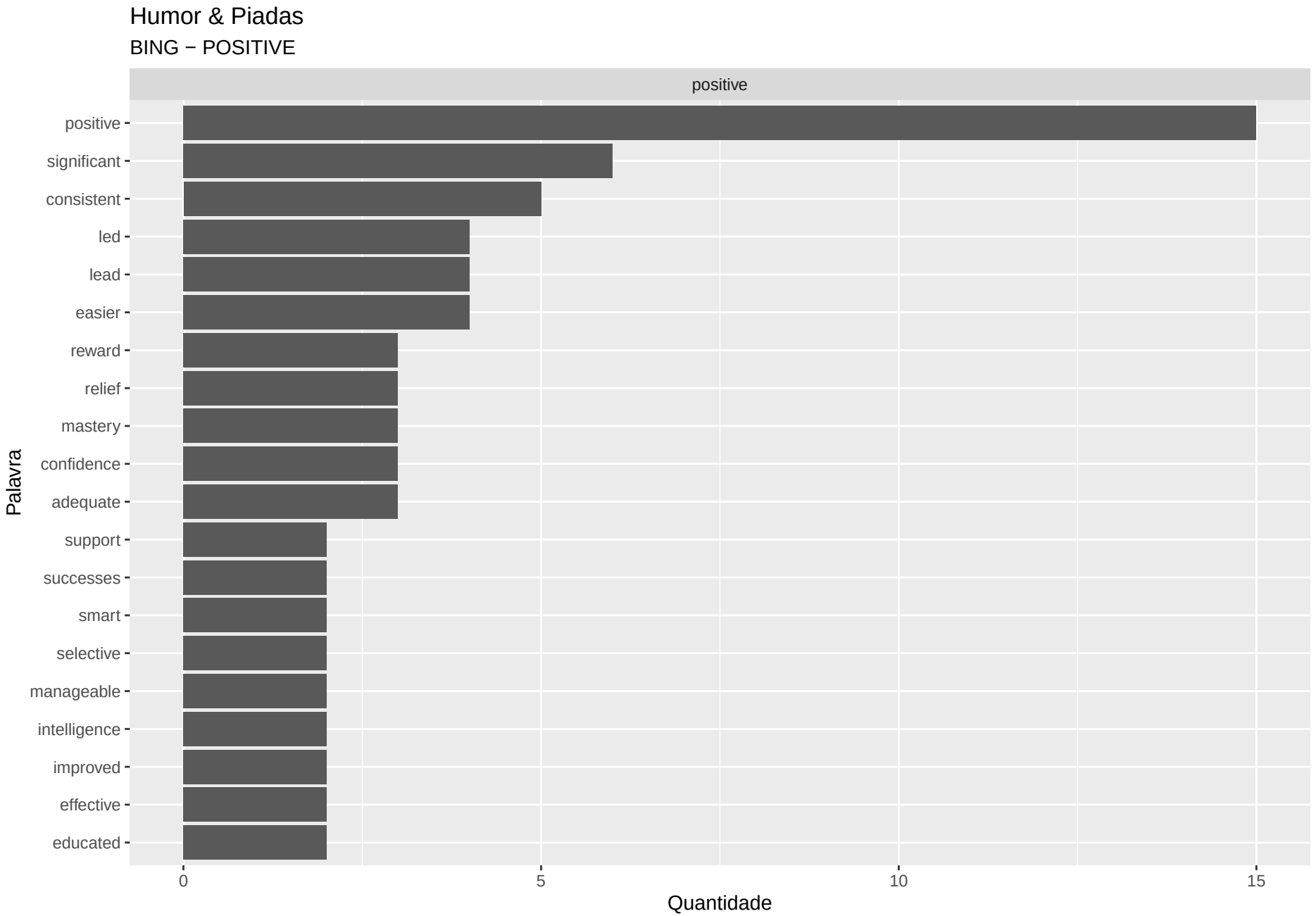

Revoluções Sociais
BING − NEGATIVE
negative
ineffectiveness
uprising
collapse
violent
discontent
crisis
autocratic
authoritarian
issues
lack
ineffective
involuntary
repressive
loss
absence
failure
failed
contend
conflicts
breakdown
Palavra
0
5
10
Quantidade

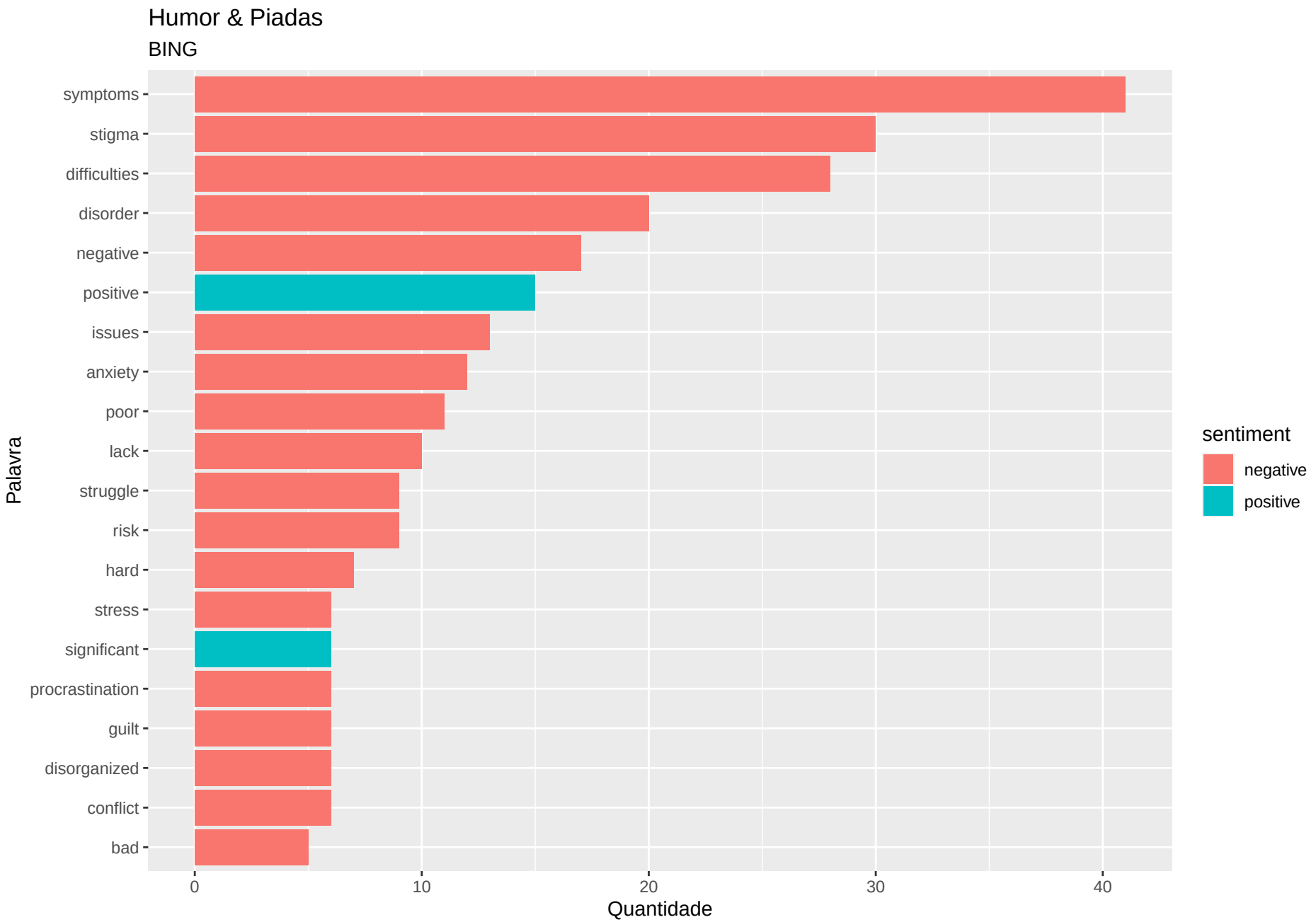

Revoluções Sociais
BING − POSITIVE
positive
revolutionary
success
spontaneous
popular
rapid
support
lead
defeat
successful
distinction
reforms
modern
sufficient
stability
defeated
supportive
significant
led
facilitate
autonomous
Palavra
0
10
20
30
Quantidade


Revoluções Sociais
BING
revolutionary
success
spontaneous
popular
rapid
support
ineffectiveness
lead
defeat
uprising
successful
collapse
distinction
violent
reforms
discontent
crisis
autocratic
authoritarian
issues
Palavra
0
10
20
30
Quantidade
sentiment
negative
positive

BING - TDAH em Mulheres


TDAH & Mulheres
BING − Negative
negative
symptoms
stigma
difficulties
disorder
negative
issues
anxiety
poor
lack
struggle
risk
hard
stress
procrastination
guilt
disorganized
conflict
complex
challenging
bad
Palavra
0
10
20
30
40
Quantidade


Humor & Piadas
BING − POSITIVE
positive
positive
significant
consistent
led
lead
easier
reward
relief
mastery
confidence
adequate
support
successes
smart
selective
manageable
intelligence
improved
effective
educated
Palavra
0
5
10
15
Quantidade

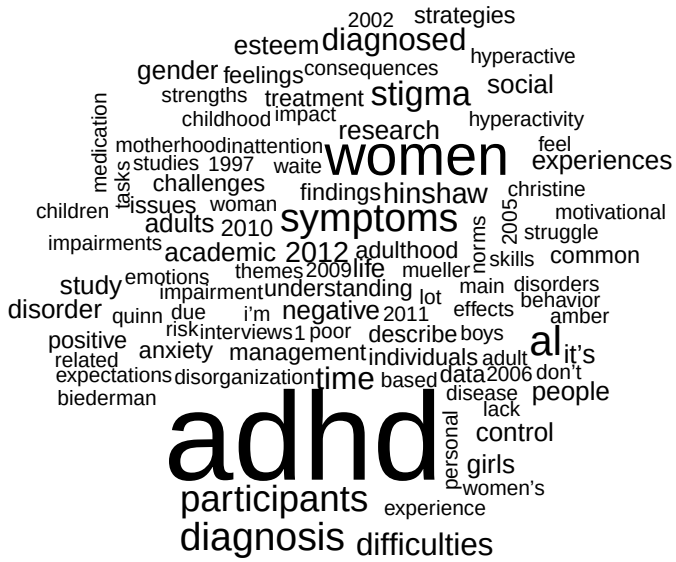

Humor & Piadas
BING
symptoms
stigma
difficulties
disorder
negative
positive
issues
anxiety
poor
lack
struggle
risk
hard
stress
significant
procrastination
guilt
disorganized
conflict
bad
Palavra
0
10
20
30
40
Quantidade
sentiment
negative
positive

Nuvens de palavras

Wordcloud - Humor & Piadas

Wordcloud - Revoluções Sociais

Wordcloud - TDAH em Mulheres



negative

positive

Ponderação de termos

- Parte da ideia de que, dentro de um contexto, alguns termos podem ser mais importantes do que outros para descrever o conteúdo dos documentos;
- Por exemplo, um termo que apareça em todos os documentos da base não tem muita utilidade para indexação. Por outro lado, um termo raro pode ter grande importância;
- Por sua vez, um termo que apareça muitas vezes em um documento específico pode, em muitos casos, dar uma ideia melhor sobre o conteúdo desse documento do que um termo que apareça poucas vezes.

Document-term matrix (DTM)

- Todos os termos de um conjunto de documentos são colocados em uma DTM,
- Uma DTM é uma matriz matemática que descreve a frequência dos termos que ocorrem em uma coleção de documentos.
- Linhas: Termos
- Colunas: Documentos
- Esquema de ponderação de termos: Term Frequency, Binary Weight, TF-IDF, etc.

Um exemplo de DTM

- Doc1: I like R
- Doc2: I like Python

Termos	Doc1	Doc2
I	1	1
Like	1	1
Python	0	1
R	1	0

No R: Criando uma DTM

- Podemos usar o pacote `tm` que contém as seguintes funções:
- A função `VectorSource()` interpreta os elementos de um vetor x como documentos;
- A função `Corpus()` cria um objeto do tipo corpus;
- A função `TermDocumentMatrix()` recebe um objeto do tipo corpus como argumento, e cria uma DTM;
- A função `Inspect()` apresenta informações detalhadas sobre uma DTM;

- Exemplo:

```
library(magrittr)
library(tm) ## package for text mining
```

```
<<TermDocumentMatrix (terms: 4, documents: 2)>>
Non-/sparse entries: 6/2
Sparsity           : 25%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
        Docs
Terms    1 2
  i      1 1
  like   1 1
  python 0 1
  r      1 0
```

No R: Diferentes esquemas de Ponderação (binary weighting)

- Simples esquema de ponderação por binários
- Se o termo está no documento é atribuído o valor de 1, se ele não está no documento é atribuído o valor de 0.

- Exemplo:

```
## various term weighting schemes
m %>% weightBin() %>% inspect() ## binary weighting
```

```
<<TermDocumentMatrix (terms: 4, documents: 2)>>
Non-/sparse entries: 6/2
Sparsity           : 25%
Maximal term length: 6
Weighting          : binary (bin)
Sample             :
        Docs
Terms    1 2
  i      1 1
  like   1 1
  python 0 1
  r      1 0
```

- Se baseia na premissa de que quanto mais vezes um termo aparece em um documento, maior sua capacidade de descrever seu conteúdo.
- Assim, o peso do termo no documento é proporcional a sua frequência.

- Exemplo:

```
m %>% weightTf() %>% inspect() ## term frequency
```

```
<<TermDocumentMatrix (terms: 4, documents: 2)>>
Non-/sparse entries: 6/2
Sparsity           : 25%
Maximal term length: 6
Weighting          : term frequency (tf)
Sample             :
        Docs
Terms    1 2
  i      1 1
  like   1 1
  python 0 1
  r      1 0
```

No R: Diferentes esquemas de Ponderação (TF-IDF)

- TF-IDF é uma medida estatística que tem o intuito de indicar a importância de uma palavra de um documento em relação a uma coleção de documentos.
- Term Frequency (TF): é o número de ocorrências do termo t_i no documento d_j
- Inverse Document Frequency (IDF) é um peso atribuído para cada termo para medir seu grau de importância em relação à uma coleção de documentos de texto.

- O IDF de um termo t é definido como:

$$idf(term) = ln(\frac{n_{documents}}{n_{documents\,containing\,term}})$$

$$tfidf(term) = tf_{ij} \,.\, idf_{term}$$

- O IDF reduz o peso de termos que ocorrem com frequência em documentos e aumenta o peso de termos que ocorrem raramente.
- Busca expressar a importância de um termo dentro da base de documentos segundo sua raridade;
- TD-IDF é o esquema de ponderação mais popular na prática;

- No R, podemos usar a função `weightTfIdf()`:

```
m %>% weightTfIdf(normalize=F) %>% inspect()
```

```
<<TermDocumentMatrix (terms: 4, documents: 2)>>
Non-/sparse entries: 2/6
Sparsity           : 75%
Maximal term length: 6
Weighting          : term frequency - inverse document frequer
Sample             :
        Docs
Terms    1 2
  i      0 0
  like   0 0
  python 0 1
  r      1 0
```

No R: Diferentes esquemas de Ponderação (normalized TF-IDF)

- A normalização é usada para evitar viés na frequência de termos em documentos mais curtos ou mais longos.

```
m %>% weightTfIdf(normalize=T) %>% inspect()
```

```
<<TermDocumentMatrix (terms: 4, documents: 2)>>
Non-/sparse entries: 2/6
Sparsity           : 75%
Maximal term length: 6
Weighting          : term frequency - inverse document frequen
Sample             :
        Docs
Terms            1         2
  i      0.0000000 0.0000000
  like   0.0000000 0.0000000
  python 0.0000000 0.3333333
  r      0.3333333 0.0000000
```

Relações entre palavras

Até agora...

- Palavras como unidades individuais;
- Relações com sentimentos ou documentos.

No entanto...

- Relações entre palavras;
- Quais palavras tendem a seguir outras imediatamente;
- Co-ocorrer dentro dos mesmos documentos.

Ou seja... Exploraremos alguns dos métodos para calcular e visualizar relacionamentos entre palavras em seu conjunto de dados de texto.

Tokenização por n-grama

O que é um n-grama? - Um n-grama é uma sequência contínua de *n* itens de uma determinada amostra de texto.

A função ***unnest_tokens()*** também pode ser utilizada para tokenizar sequências consecutivas de palavras, ou seja, n-gramas.

Podemos ver com que frequência a palavra X é seguida pela palavra Y, através um modelo de relação entre ambas.

Para a demosntração, uilizaremos os seguintes pacotes:

```
library(dplyr)
library(tidytext)
library(pdftools)
library(stringr)
library(ggplot2)
library(tm)
library(wordcloud)
library(tidyverse)
library(ggraph)
library(igraph)
```

E o livro "*Dom Casmurro*", de Machado de Assis, como texto para análise.

Trasformando cada Token = n-gramas

O ***bigram***, utilizado no código abaixo, representa a definição de n = 2, ou seja, estamos examinando pares de duas palavras consecutivas.

```
bigram_dc <- dom_casmurro %>%
  unnest_tokens(bigram, linha, token = "ngrams", n = 2)
```

Para examinarmos 3 palavras consecutivas, utilizamos o ***trigram*** no lugar do ***bigram***, e mudamos o n = 3.

```
trigram_dc <- dom_casmurro %>%
  unnest_tokens(trigram, linha, token = "ngrams", n = 3)
```

O resultado fica assim para n = 2, onde cada token representa um bigrama:

```
# A tibble: 61,022 x 2
   capitulo bigram
      <int> <chr>
 1        1 capítulo primeiro
 2        1 <NA>
 3        1 <NA>
 4        1 do título
 5        1 <NA>
 6        1 <NA>
 7        1 <NA>
 8        1 uma noite
 9        1 noite destas
10        1 destas vindo
# ... with 61,012 more rows
# i Use `print(n = ...)` to see more rows
```

O resultado fica assim para n = 3, onde cada token representa um trigrama:

```
# A tibble: 54,642 x 2
   capitulo trigram
      <int> <chr>
 1        1 <NA>
 2        1 <NA>
 3        1 <NA>
 4        1 <NA>
 5        1 <NA>
 6        1 <NA>
 7        1 <NA>
 8        1 uma noite destas
 9        1 noite destas vindo
10        1 destas vindo da
# ... with 54,632 more rows
# i Use `print(n = ...)` to see more rows
```

Podemos contar e filtrar n-gramas

```
# A tibble: 37,637 x 2
   bigram          n
   <chr>       <int>
 1 <NA>         1056
 2 que não       180
 3 o que         170
 4 que me        170
 5 é que         164
 6 minha mãe     148
 7 josé dias     141
 8 que eu        132
 9 que a         127
10 que o         112
# ... with 37,627 more rows
# i Use `print(n = ...)` to see more rows
```

Muitos dos bigramas mais comuns são pares de palavras comuns (stopwords), como "*que a*", "*que o*", e entre outros. Para resolvermos esse problema, utilizaremos a função ***separate()*** e ***filter()***. Vale ressaltar que o *stopwords* é do pacote ***tm***.

Separando

```
library(tm)
# Separando o Bigramas
bigrams_separados <- bigram_dc %>%
  separate(bigram, c("palavra1", "palavra2"), sep = " ")
```

```
# A tibble: 61,022 x 3
   capitulo palavra1 palavra2
      <int> <chr>    <chr>
 1        1 capítulo primeiro
 2        1 <NA>     <NA>
 3        1 <NA>     <NA>
 4        1 do       título
 5        1 <NA>     <NA>
 6        1 <NA>     <NA>
 7        1 <NA>     <NA>
 8        1 uma      noite
 9        1 noite    destas
10        1 destas   vindo
# ... with 61,012 more rows
# i Use `print(n = ...)` to see more rows
```

Filtrando

```
# Filtrando o Bigramas
filtrando_bigramas <- bigrams_separados%>%
  filter(!palavra1 %in% stopwords('pt')) %>%
  filter(!palavra2 %in% stopwords('pt'))
```

```
# A tibble: 13,697 x 3
   capitulo palavra1 palavra2
      <int> <chr>    <chr>
 1        1 capítulo primeiro
 2        1 <NA>     <NA>
 3        1 <NA>     <NA>
 4        1 <NA>     <NA>
 5        1 <NA>     <NA>
 6        1 <NA>     <NA>
 7        1 noite    destas
 8        1 destas   vindo
 9        1 engenho  novo
10        1 rapaz    aqui
# ... with 13,687 more rows
# i Use `print(n = ...)` to see more rows
```

Nova contagem

```
# Criando uma nova contagem dos Bigramas
novo_bigramas_dc <- filtrando_bigramas %>%
  count(palavra1, palavra2, sort = TRUE)
```

Por fim, após removermos as stopwords, temos o seguinte resultado:

```
# A tibble: 10,937 x 3
   palavra1 palavra2     n
   <chr>    <chr>    <int>
 1 <NA>     <NA>      1056
 2 josé     dias       141
 3 prima    justina     45
 4 tio      cosme       43
 5 outra    vez         31
 6 pode     ser         29
 7 outra    coisa       22
 8 alguma   coisa       21
 9 mata     cavalos     20
10 padre    cabral      19
# ... with 10,927 more rows
# i Use `print(n = ...)` to see more rows
```

Para recombinar, basta utilizar a função ***unite()***:

```
bigramas_dc_unido <- filtrando_bigramas %>%
  unite(bigram, palavra1, palavra2, sep = " ")
```

```
# A tibble: 13,697 x 2
   capitulo bigram
      <int> <chr>
 1        1 capítulo primeiro
 2        1 NA NA
 3        1 NA NA
 4        1 NA NA
 5        1 NA NA
 6        1 NA NA
 7        1 noite destas
 8        1 destas vindo
 9        1 engenho novo
10        1 rapaz aqui
# ... with 13,687 more rows
# i Use `print(n = ...)` to see more rows
```

Visualizando uma rede de bigramas com ggraph e igraph

Podemos visualizar todas as relações entre as palavras simultaneamente, em vez de apenas as primeiras de cada vez.
Como? Podemos organizar as palavras em uma rede, ou "gráfico". O pacote **igraph** tem funções que auxiliam a manipulação e a análise de redes.
Podemos utilizar a função ***graph_from_data_frame()*** para criar um objeto igraph a partir de dados arrumados. Essa função recebe um quadro de dados de arestas com colunas atributos **"from"**, **"to"** e ***"weight"*** (neste caso n).

- **from** : o nó de onde uma aresta está vindo
- **to** : o nó para o qual uma aresta está indo
- **weight** : Um valor numérico associado a cada aresta

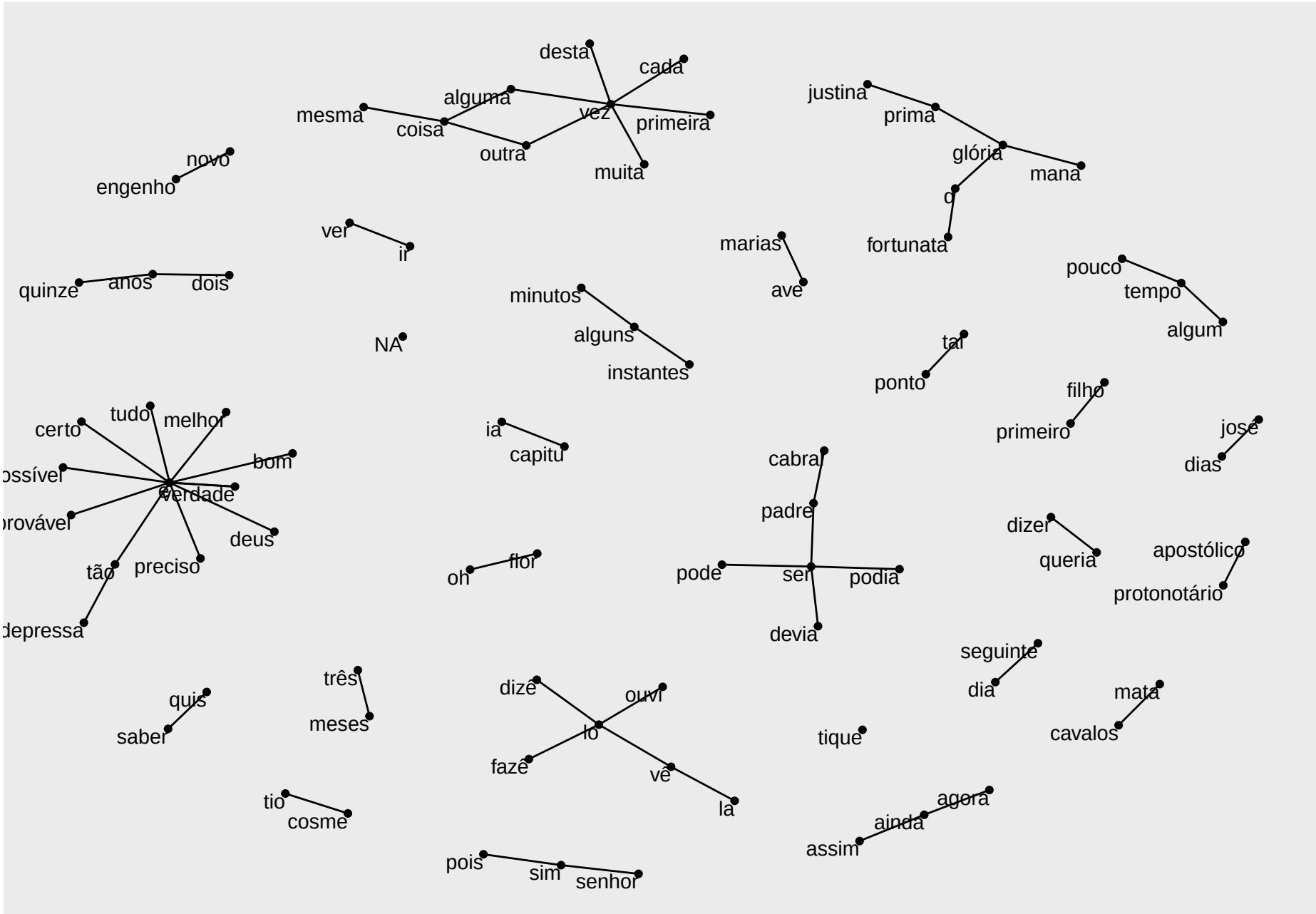
Gerando uma rede utilizando o *igraph*

```
library(igraph)
rede_bigrama_dc <- novo_bigramas_dc %>%
  filter(n > 5) %>%
  graph_from_data_frame()
```

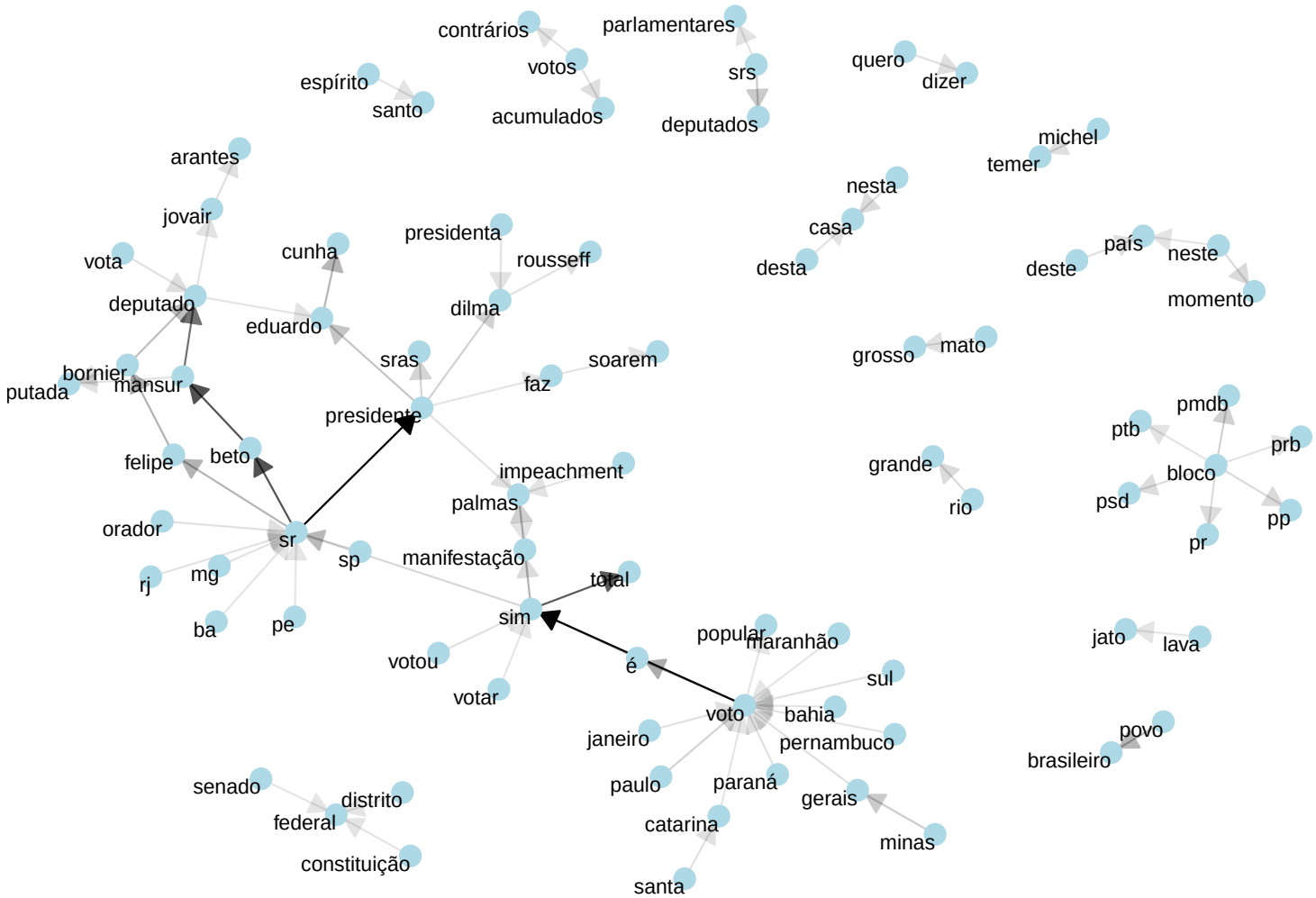
```
IGRAPH b49cfd7 DN-- 86 63 --
+ attr: name (v/c), n (e/n)
+ edges from b49cfd7 (vertex names):
 [1] NA     ->NA         josé   ->dias      prima  ->justina
 [5] outra  ->vez        pode   ->ser       outra  ->coisa
 [9] mata   ->cavalos    padre  ->cabral     ser    ->padre
[13] dia    ->seguinte   muita  ->vez       podia  ->ser
[17] ainda  ->agora      alguma ->vez       devia  ->ser
[21] fazê   ->lo         quis   ->saber     tudo   ->é
[25] ainda  ->assim      algum  ->tempo     d      ->fortunata
[29] desta  ->vez        é      ->possível  é      ->tão
+ ... omitted several edges
```

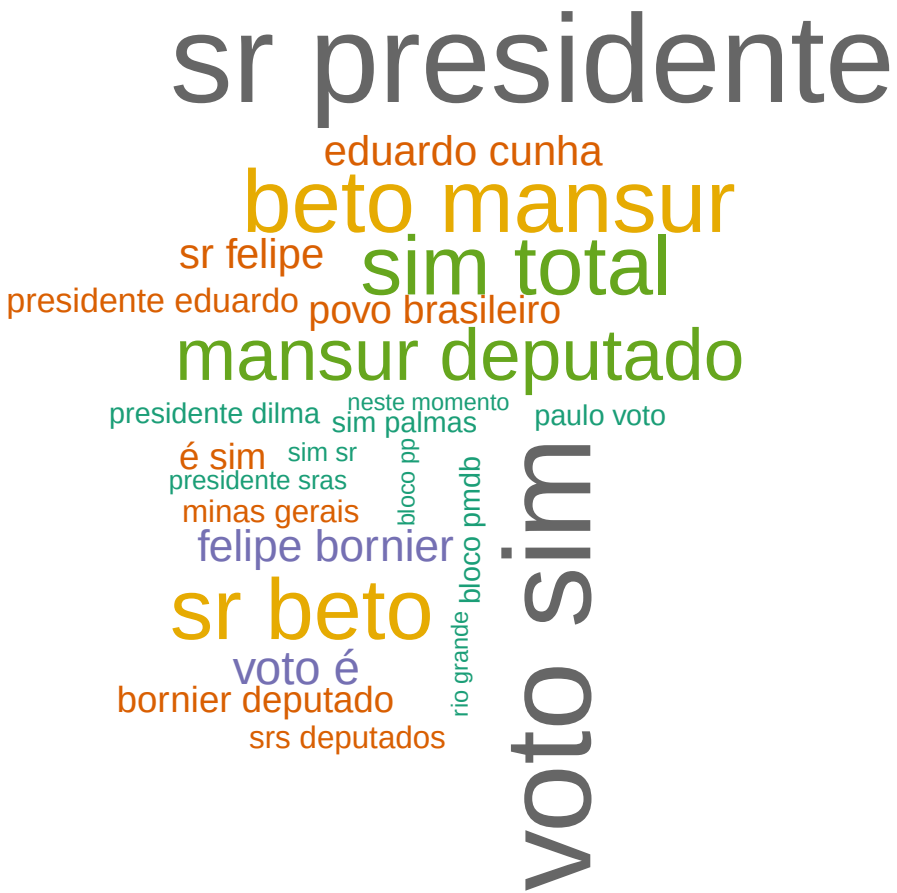
O igraph tem funções de plotagem embutidas, mas não é ele que reproduz a visualização. Para isso, podemos converter um objeto igraph em um **ggraph** com a função **ggraph()**, após adicionarmos camadas a ele, o transmitiremos ao **ggplot2**.

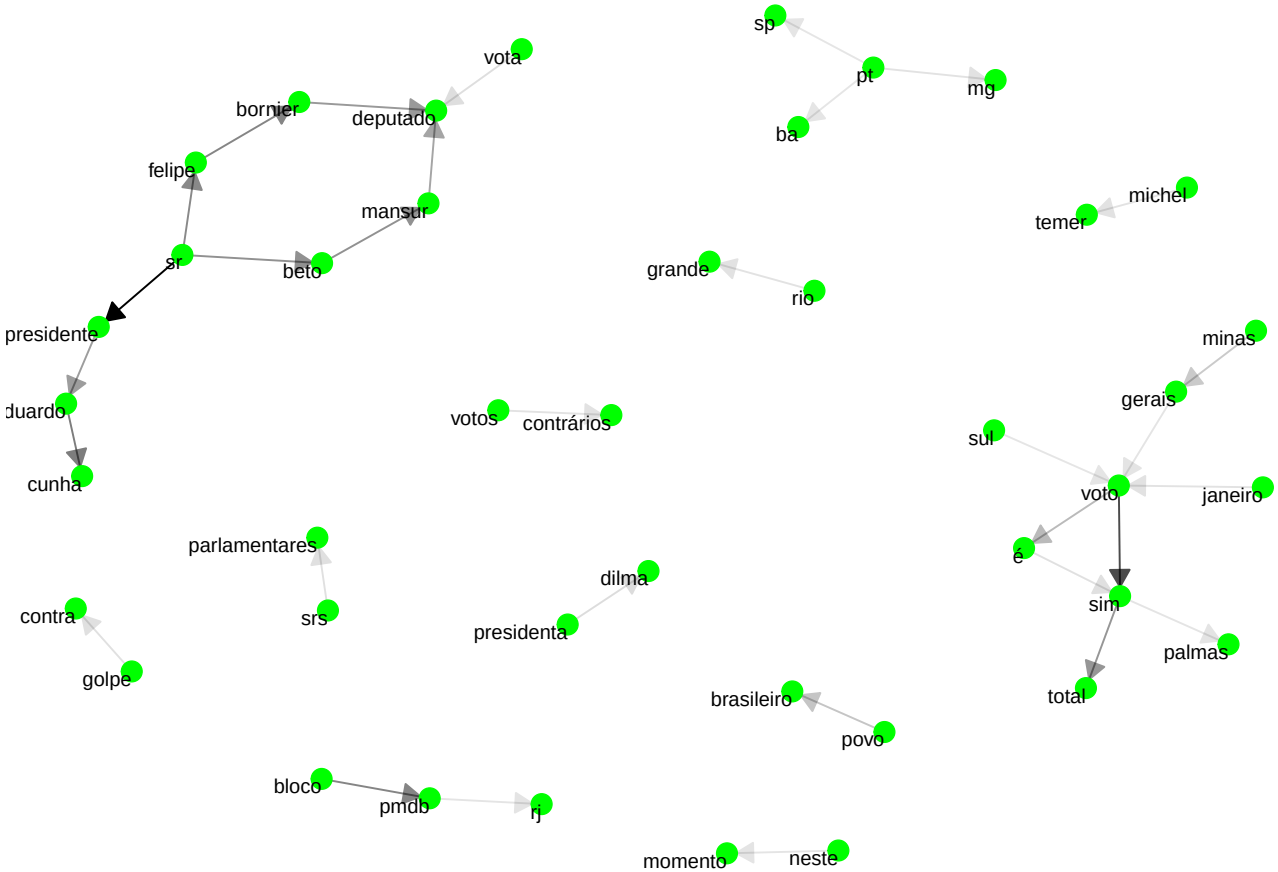
```
library(ggraph)
set.seed(2017)
grafico <- ggraph(rede_bigrama_dc, layout = "fr") +
  geom_edge_link() +
  geom_node_point() +
  geom_node_text(aes(label = name), vjust = 1, hjust = 1)
```

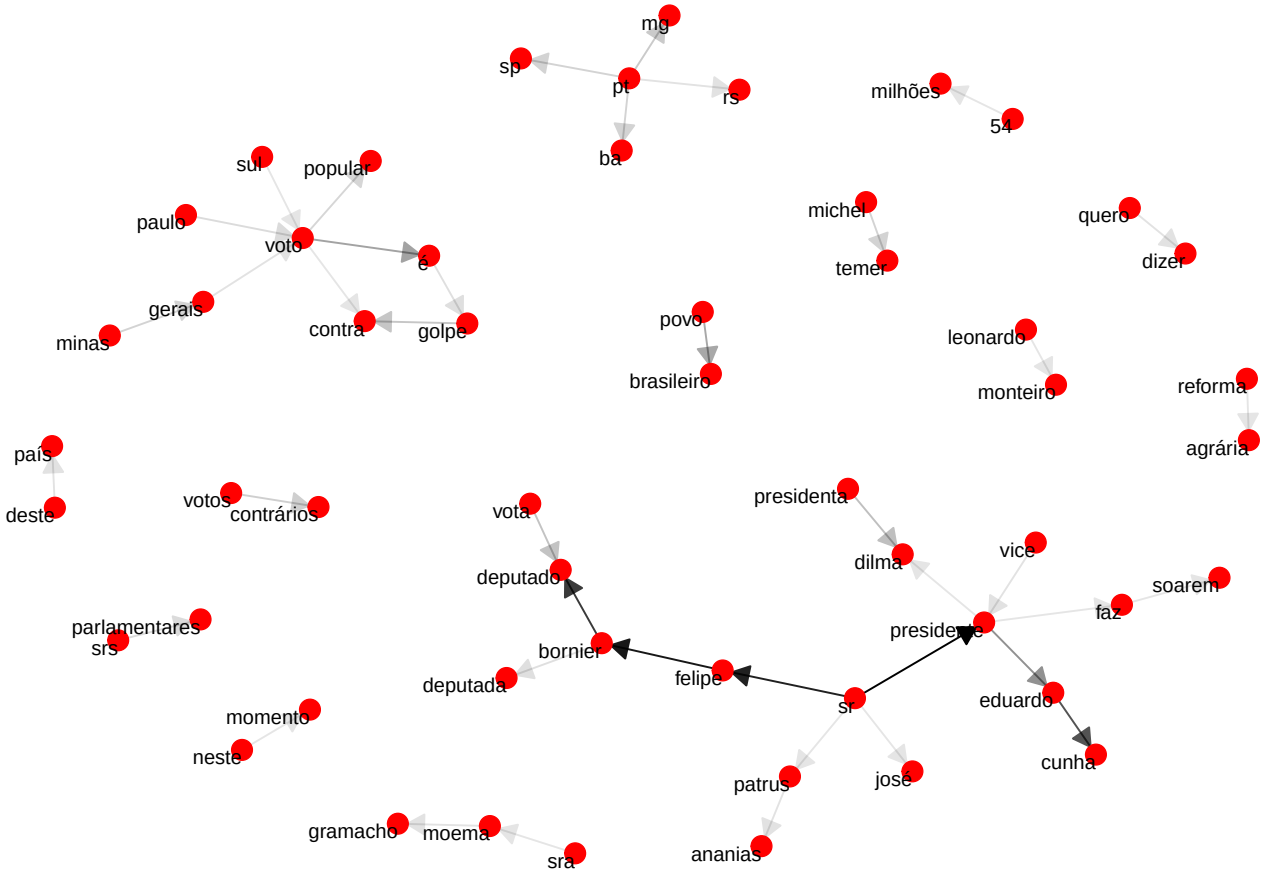

desta
cada
alguma
mesma
vez
primeira
coisa
outra
muita
novo
engenho
justina
prima
glória
mana
d
fortunata
ver
ir
marias
ave
pouco
tempo
algum
quinze
anos
dois
minutos
alguns
instantes
NA
tal
ponto
filho
primeiro
jose
dias
certo
tudo
melhor
bom
ossível
verdade
rovável
deus
tão
preciso
depressa
ia
capitu
cabra
padre
pode
ser
podia
devia
dizer
queria
apostólico
protonotário
oh
flor
três
meses
quis
saber
dize
ouvi
lo
faze
ve
la
seguinte
dia
mata
cavalos
tique
tio
cosme
agora
ainda
assim
pois
sim
senhor

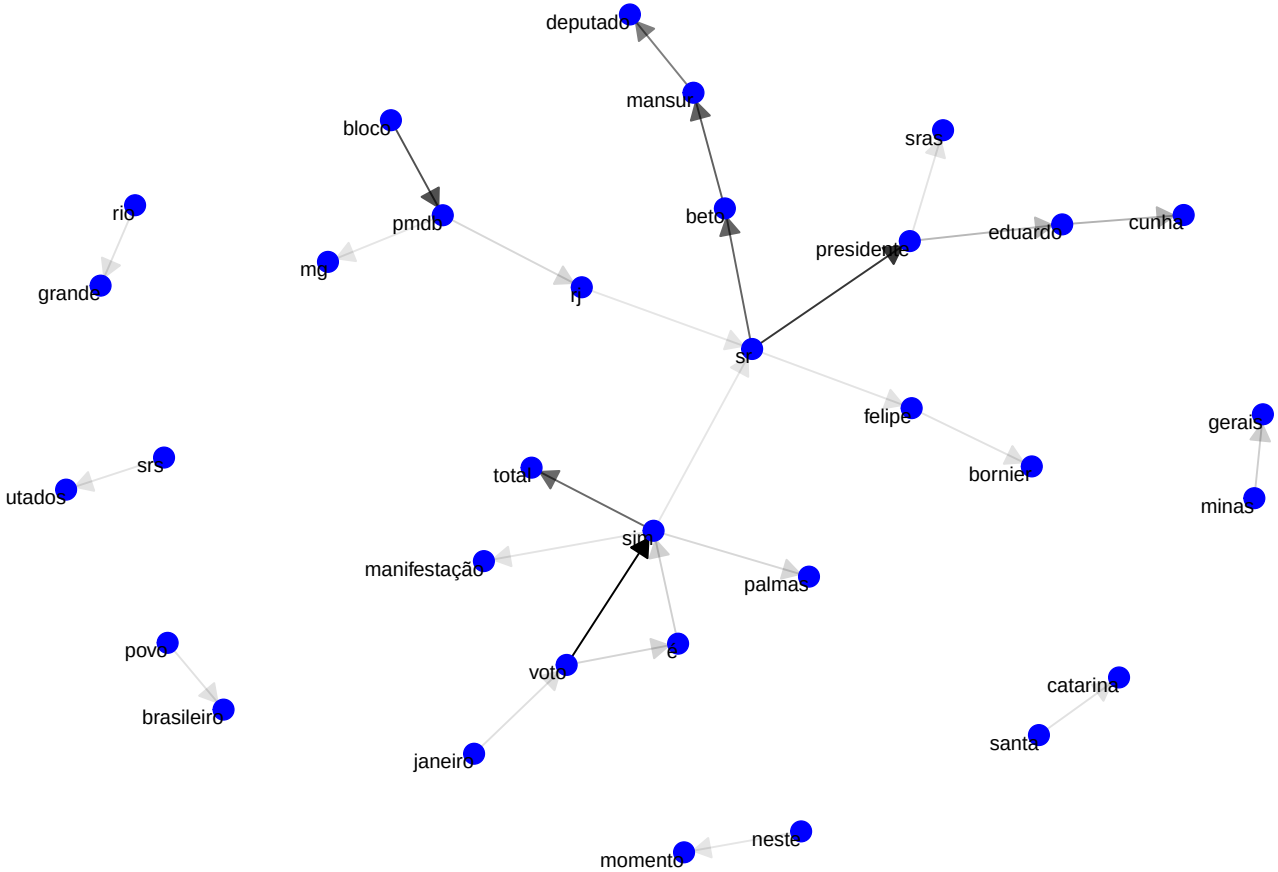
Aplicação interessante...




sr presidente
eduardo cunha
beto mansur
sr felipe
sim total
presidente eduardo
povo brasileiro
mansur deputado
presidente dilma
neste momento
sim palmas
paulo voto
é sim
sim sr
presidente sras
bloco pp
minas gerais
bloco pmdb
felipe bornier
sr beto
voto é
rio grande
bornier deputado
srs deputados
voto sim


vota
bornier
deputado
felipe
mansur
sr
beto
presidente
duardo
cunha
sp
pt
mg
ba
michel
temer
grande
rio
votos
contrários
minas
gerais
sul
voto
janeiro
é
sim
palmas
total
parlamentares
srs
dilma
presidenta
contra
golpe
brasileiro
povo
bloco
pmdb
rj
momento
neste


mg
sp
pt
rs
ba
milhões
54
sul
popular
paulo
voto
é
gerais
minas
contra
golpe
michel
temer
quero
dizer
povo
brasileiro
leonardo
monteiro
reforma
agrária
país
deste
votos
contrários
vota
deputado
presidenta
dilma
vice
soarem
faz
parlamentares
srs
bornier
deputada
felipe
presidente
sr
eduardo
cunha
momento
neste
patrus
jose
gramacho
moema
sra
ananias


deputado
mansur
bloco
sras
rio
pmdb
beto
presidente
eduardo
cunha
mg
grande
pj
sr
felipe
gerais
srs
utados
total
bornier
minas
sim
manifestação
palmas
povo
voto
s
catarina
brasileiro
santa
janeiro
neste
momento

Fontes

- "Text Mining with R" - Julia Silge & David Robinson
  - tidytextmining.com
- https://www.tibco.com/pt-br/reference-center/what-is-text-analytics