Trabalho 2

Rodrigo Malta Esteves

02/03/2021

1. Etapa 1: Selecionando do texto
2. Etapa 2: Manipulando o texto
3. Etapa 3: Outras análises

Etapa 1: Selecionando do texto

"Não existe golaço de falta".

A afirmação, obviamente equivocada, é do meu amigo Fernando Martinho, jornalista, editor da Corner – onde lancei meu segundo livro. Uma alegação que, não só considero falsa, como penso exatamente o oposto: todo gol de falta é golaço.

O de Benítez, nesta terça-feira, contra o Palmeiras, é mais uma prova disso.

Veja bem, a cobrança de falta direta é um dos momentos mais previsíveis do futebol. Um dos que menos permite improvisações ou mudanças bruscas. Fica atrás apenas da cobrança de pênalti.

Até um lateral, por mais simples que pareça, tem suas variáveis – inclusive para o erro, como foi no que resultou o gol palmeirense. A batida direta, não.

É o árbitro quem define a posição da bola. É o goleiro adversário quem arma a barreira. E, pasmem, a cobrança ainda vem após um apito! Ou seja, é anunciado para todos em alto e bom som que está vindo o chute. Existe maior previsibilidade que essa? Não há.

A cobrança de falta nasceu para dar errado. Tudo é milimetricamente calculado para que o cobrador falhe. Até deitar atrás da barreira já fazem. É a barreira da barreira.

Em casos extremos, como aconteceu com Marcelinho Carioca diversas vezes nos anos 90 e início de 2000, um defensor ficava grudado na trave, abaixo do ângulo, para de alguma maneira proteger seu gol. Era quase constrangedor que não conseguisse alcançar a bola, tal qual uma criança tentando estourar, sem sucesso, o balão de doces no seu aniversário.

E é daí que nasce o golaço: da dificuldade.

Ora, se todos sabem quando, quem e onde será cobrada a falta, a desvantagem é imensa. Só o talento é capaz de superar todos estes fatores negativos. E se há qualidade incontestável num gol, é exatamente onde ele ganha ares de golaço.

Voltemos ao Allianz Parque.

Quando Cano sofreu falta de Felipe Mello, apenas dois jogadores do Vasco se posicionaram para a batida: Yago Pikachu e Benítez. No entanto, apesar da boa fase vivida pelo camisa 22, todos sabiam que a bola seria do argentino, que minutos antes já havia encontrado a cabeça de Léo Matos, que marcou, mas estava impedido.

O lateral/ponta/meia jamais marcou um gol de falta pelo Cruz-Maltino em cinco anos de casa. Não seria naquela noite.

Quando o Palmeiras posicionou seus dez jogadores na entrada da área, sendo cinco na barreira protegendo o canto direito do goleiro Jaílson, estava claro que haviam poucas opções Um time inteiro posicionado para evitar o gol, do goleiro ao ponta esquerda. Só havia uma alternativa: a perfeição.

Girando como uma dançarina de tango, a bola saiu dos pés de Benítez e passou entre o 3º e o 4º homem da barreira palmeirense. Num primeiro olhar, parecia viajar para o meio do gol. E é aí que está a beleza do lance.

A perfeição da batida não está simplesmente no ponto da rede que ela toca ou na posição em que supera o goleiro, mas em toda a sua trajetória. Mudar de direção é a única chance que ela tem de surpreender.

A curva é o drible da falta.

A passada lateral desesperada de Jaílson confirmava o inevitável: nascia ali mais um golaço de falta.

Fonte: http://blogs.lancenet.com.br/garone/o-golaco-de-falta/

Etapa 2: Manipulando o texto

Pacotes

```
library("stringr")   # pacote para processamento de texto
library("wordcloud")   # pacote para nuvem de palavras
library("tm")   # dependência do wordcloud
library("slam")   # dependência do wordcloud
library("wordcloud2")  # pacote para nuvem de palavras
```

Lendo o texto

```
#Lendo o texto
texto <- scan("C:\\Users\\malta\\Desktop\\Pós Graduação\\Mineração de Dados\\Trabalhos\\Atividade 2\\O g

# Limpando o texto/Tornando o texto mais fácil de ser lido
texto_split <- str_split(texto,"\n")   # dividindo por linhas
texto <- str_replace_all(texto,"\n"," ") # removendo os \n
texto <- str_to_lower(texto)
texto1 <- str_split(texto," ")
```

Retirando pontuação

```
#Retirando pontuação
texto2 <-  str_remove_all(texto1,"[[:punct:]]")
```

```
## Warning in stri_replace_all_regex(string, pattern,
## fix_replacement(replacement), : argument is not an atomic vector; coercing
```

Retirando símbolos

```
#Retirando símbolos
simbolos <- "[\\~\\!\\@\\#\\$\\%\\^\\&\\*\\(\\)\\{\\}\\_\\+\\:\\<\\>\\?\\,\\.\\/\\;\\'\\[\\]\\-\\=\\|\"]
texto3 <- str_remove_all(texto2,simbolos)
```

Retirando os números

```r
#Retirando os números (Apenas de forma ilustrativa já que são importantes)
numeros <- "[0-9]"
texto4 <- str_remove_all(texto3,numeros)

texto4_split <- str_split(texto4,"\n")   # dividindo por linhas
texto4 <- str_replace_all(texto4,"\n"," ") # removendo os \n
texto4 <- str_to_lower(texto4)
texto4 <- str_split(texto4," ")
```

Retirando stopwords

```r
#Lista de stopwords em portugues (Não consegui ler "crase" com o R)
StopW <- read.table("C:\\Users\\malta\\Desktop\\Pós Graduação\\Mineração de Dados\\Trabalhos\\Atividade

rmstopwords <- function(texto,stopwords){   # texto deve ser uma lista e stopwords deve ser um vetor
  n <- length(texto)
  aux <- texto
  for(i in 1:n){
    ind <- which(texto[[i]]%in%stopwords)   # índices das stopwords
    if(length(ind)>=1){   # evitar problemas nos índices quando não há stopwords
      aux[[i]] <- texto[[i]][-ind]
    }
  }
  return(aux)
}

texto5 <- rmstopwords(texto4,StopW)   # retirando as stopword
```

Gráfico de barras

```r
wordvector <- function(texto){   # texto deve ser uma lista
  n <- length(texto)
  aux <- c()
  for(i in 1:n){
    aux <- c(aux,texto[[i]])
  }
  aux <- aux[aux!=""]   # retirando o que for vazio
  return(aux)
}
palavras <- wordvector(texto5)
numpa <- length(palavras)   # número de palavras
numpa
```

```
## [1] 288
```

```r
numpa2 <- length(unique(palavras))   # número de palavras únicas
numpa2
```

```
## [1] 229
```

```
tab <- table(palavras)   # frequência das palavras
head(tab,n=30)
```

```
## palavras
##      abaixo   aconteceu  adversário   afirmação          aí       ainda
##           1           1           1           1           1           1
##    alcançar    alegação      alguma         ali     allianz alternativa
##           1           1           1           1           1           1
##        alto       amigo      ângulo aniversário        anos       antes
##           1           1           1           1           2           1
##   anunciado      apenas      apesar       apito        após     árbitro
##           1           2           1           1           1           1
##        área        ares   argentino        arma       atrás       balão
##           1           1           1           1           2           1
```

```
nometab <- names(tab)   # nomes na tabela
freqtab <- as.numeric(tab)   # frequências na tabela
freqmais <- sort(freqtab)[(numpa2-9):numpa2]   # dez maiores frequências
freqmais
```

```
##  [1]  3  3  4  4  4  4  6  6  7 10
```

```
ind <- which(freqtab>=freqmais[1])    # índices das dez maiores frequências
ind
```

```
##  [1]  31  32  35  37  56  99 110 111 112 158 215
```

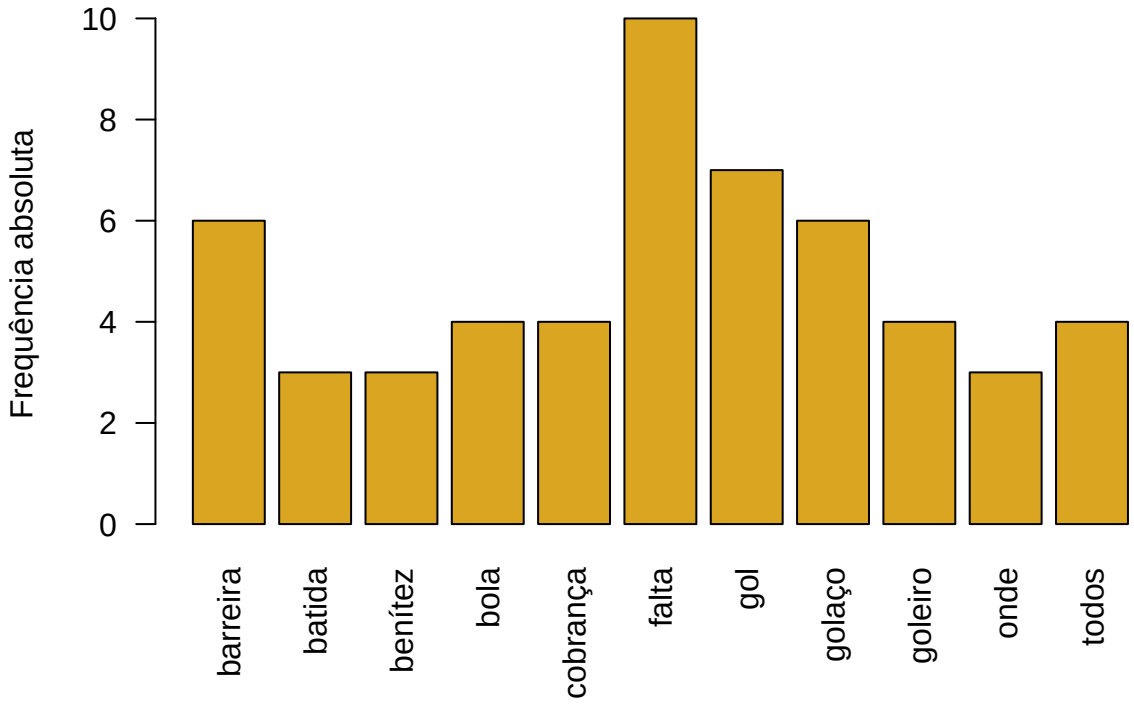
```
palafreq <- nometab[ind]   # palavras mais frequentes
freq <- freqtab[ind]   # frequências das palavras mais frequentes
palafreq
```

```
##  [1] "barreira" "batida"   "benítez"  "bola"     "cobrança" "falta"
##  [7] "gol"      "golaço"   "goleiro"  "onde"     "todos"
```

```
freq
```

```
##  [1]  6  3  3  4  4 10  7  6  4  3  4
```

```
barplot(freq,names=palafreq,las=2,ylab="Frequência absoluta",col="goldenrod")
```



Núvem de palavras

```
palavrasunicas <- names(tab)
freqpalavrasunicas <- as.numeric(tab)
palavras2 <- data.frame(palavrasunicas,freqpalavrasunicas)
head(palavras2)
```

```
##   palavrasunicas freqpalavrasunicas
## 1         abaixo                  1
## 2      aconteceu                  1
## 3     adversário                  1
## 4      afirmação                  1
## 5            aí                   1
## 6          ainda                  1
```

```
set.seed(12345)
wordcloud(words=palavras2[,1],freq=palavras2[,2],max.words=30,colors=2)
```


```
wordcloud(words=palavras2[,1],freq=palavras2[,2],max.words=30,colors=terrain.colors(20))
```


```
wordcloud2(data=palavras2,shape="pentagon")
```

Etapa 3: Outras análises

Análise de sentimentos

O objetivo dessa técnica é classificar sentenças, ou um conjunto de sentenças, como positivas, negativas ou neutras. Essa classificação é realizada automaticamente e extrai informações subjetivas de textos, criando conhecimento estruturado que pode ser utilizado por um sistema.

O pacote "lexiconPT" utiliza um dicionário em PT/BR para avaliação das palavras, atribuindo valores de -1 até 1 dependendo de quão positiva ela é em uma avaliação. Ele capta tanto palavras normais quanto emotes.

Essa avaliação das palavras não foi muito compatível como o texto. Idealmente teríamos um data frame com um texto em cada linha e uma variedade muito maior de palavras. Como se trata de uma crônica esportiva, certas palavras com alto índice negativo são utilizados de uma forma que não necessariamente indica esse sentimento. Por exemplo, "falta" não tem necessariamente uma conotação negativa dentro do contexto de um jogo de futebol, sendo comumente interpretada apenas como mais um evento possível dentro da partida.

```
#Pacotes
library(jsonlite)
library(tm)
library(tidytext)
library(tidyverse)
library(DT)
library(lexiconPT)
library(dplyr)
```

No primeiro passo, transformamos cada palavra em um "token" e criamos um id para cada linha do data frame resultante. Em seguida, agregamos agregamos por id e fazemos a soma dos valores de cada palavra usando como referência os valores do "lexiconPT" e damos uma nota para cada uma utilizando a média.

Em seguida, fazemos uma análise gráfica dos resultados. Plotamos a densidade dos valores de sentimento e a sua frequência.

Na última parte obsevamos as palavras mais positivas e mais negativas do texto e fazemos um gráfico de barras para sua frequência.

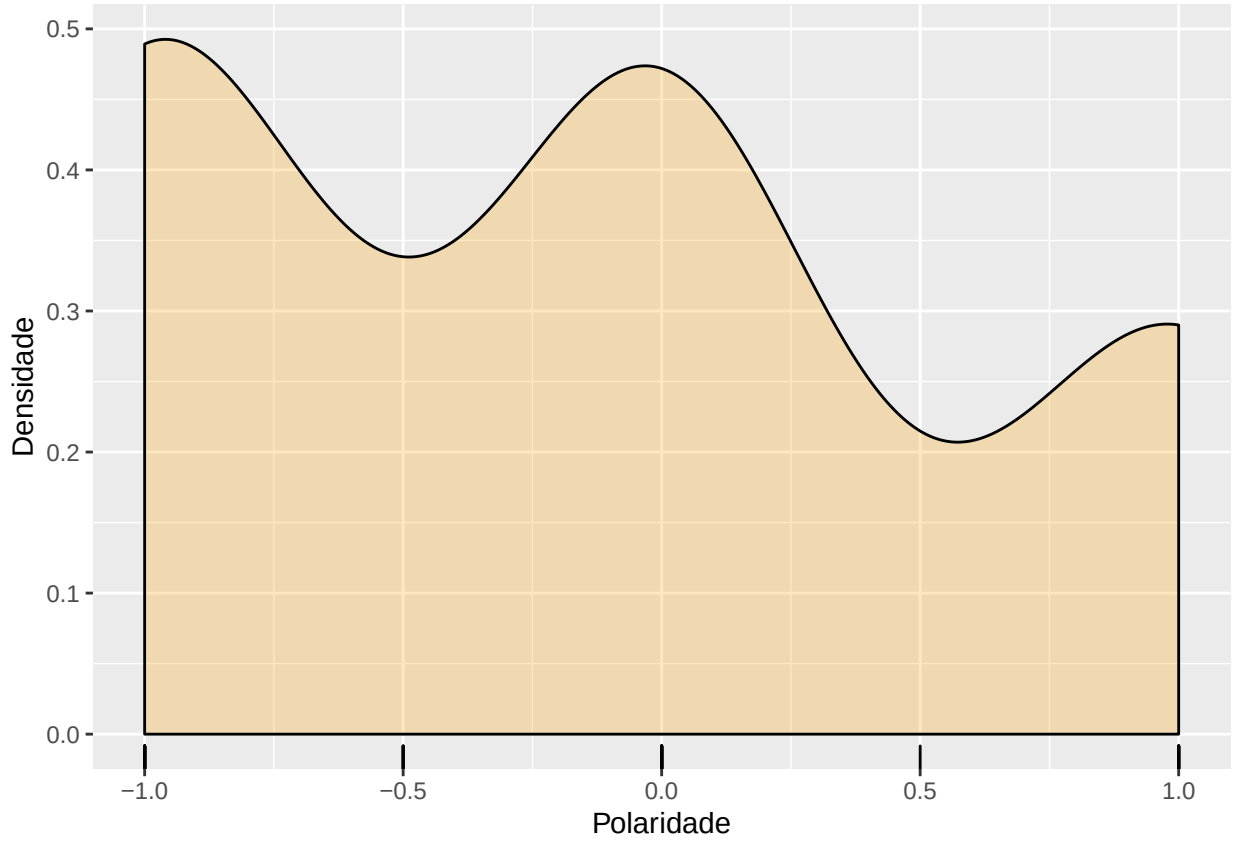
```
#http://www.leg.ufpr.br/~walmes/ensino/mintex/tutorials/03-sentimento.html

#Transformando a lista em tibble
new_tibble <- tibble(grp = unlist(texto5))

# Faz tokenização nas palavras individuais e empilha as palavras.
textoUN <- new_tibble %>%
    unnest_tokens(output = "words", input = grp)

tb_sen <- inner_join(textoUN,
                     oplexicon_v3.0[, c("term", "polarity")],
                     by = c("words" = "term"))

# Gerando um id para cada linha
tb_sen$id <- cumsum(!duplicated(tb_sen[1]))
#id <- rownames(tb_sen)
#tb_sen <- cbind(id=id, tb_sen)

sample_n(tb_sen, size = 20)
```

```
## # A tibble: 20 x 3
##    words       polarity    id
##    <chr>          <int> <int>
##  1 bruscas           -1     9
##  2 carioca            0    21
##  3 desesperada       -1    51
##  4 defensor           1    23
##  5 alto               0    13
##  6 olhar              0    44
##  7 errado            -1    18
##  8 batida            -1    46
##  9 falta             -1     7
## 10 extremos          -1    20
## 11 lateral            0    50
## 12 oposto            -1     7
## 13 direito            0    37
## 14 falta             -1    51
## 15 amigo              1     3
## 16 falta             -1    36
## 17 encontrado        -1    35
## 18 falta             -1    32
## 19 impedido           0    36
## 20 doces              1    28
```

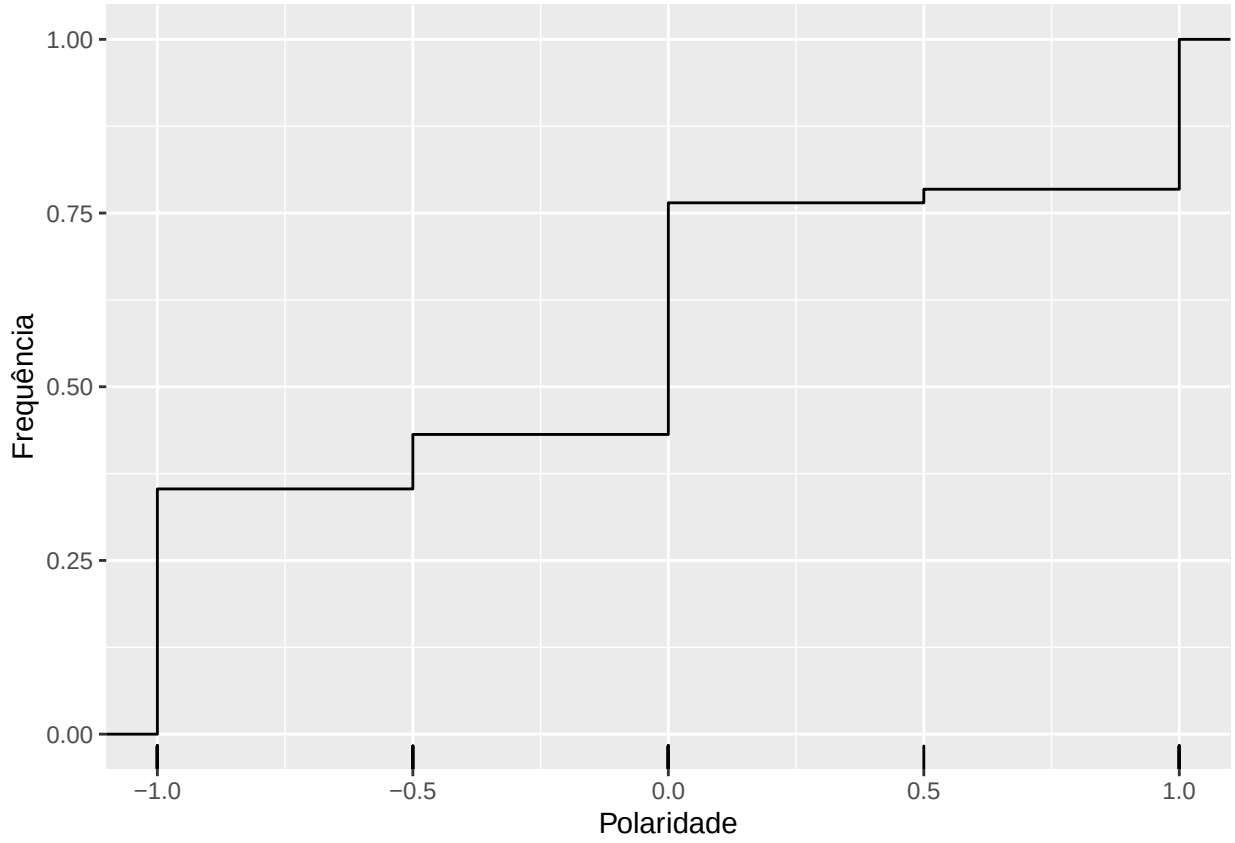
```r
# Faz a agregação da polaridade por documento.
tb <- tb_sen %>%
    group_by(id) %>%
    summarise(soma = sum(polarity),
              n = n(),
              sentiment = soma/n)
tb
```

```
## # A tibble: 51 x 4
##       id  soma     n sentiment
##    <int> <int> <int>     <dbl>
##  1     1    -2     2        -1
##  2     2     1     1         1
##  3     3     1     1         1
##  4     4     0     1         0
##  5     5    -1     1        -1
##  6     6     1     1         1
##  7     7    -3     3        -1
##  8     8    -1     1        -1
##  9     9    -1     1        -1
## 10    10     0     1         0
## # ... with 41 more rows
```

```r
# Desidade empírica 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(tb_sen[, c("id", "words")]) %>%
    select(sentiment, words)
```

```
## # A tibble: 11 x 2
##    sentiment words
##        <dbl> <chr>
##  1         1 equivocada
##  2         1 amigo
##  3         1 penso
##  4         1 bom
##  5         1 defensor
##  6         1 maneira
##  7         1 imensa
##  8         1 capaz
##  9         1 boa
## 10         1 claro
## 11         1 evitar
```

```
# As avaliações mais negativas.
tb %>%
    top_n(sentiment, n = -10) %>%
    inner_join(tb_sen[, c("id", "words")]) %>%
    select(sentiment, words)
```

```
## # A tibble: 26 x 2
##    sentiment words
##        <dbl> <chr>
##  1        -1 falta
##  2        -1 falta
##  3        -1 falsa
##  4        -1 oposto
##  5        -1 falta
##  6        -1 falta
##  7        -1 direta
##  8        -1 bruscas
##  9        -1 batida
## 10        -1 direta
## # ... with 16 more rows
```

```
# Para poder melhorar o dicionário.

# Determina as frequências dos termos de polaridade não nula.
tb_words <- tb_sen %>%
    count(words, polarity, sort = TRUE) %>%
    filter(polarity != 0)

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

```
## # A tibble: 32 x 3
##    words         negative positive
##    <chr>            <dbl>    <dbl>
##  1 amigo                0        1
##  2 batida               3        0
##  3 boa                  0        1
##  4 bom                  0        1
##  5 bruscas              1        0
##  6 capaz                0        1
##  7 claro                0        1
##  8 constrangedor        1        0
##  9 curva                1        0
## 10 dar                  1        0
## # ... with 22 more rows
```

```
# 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()

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")
```