Mineração de texto aplicada à Lei de Acesso à informação - LAI

Packages for this routine

BASE DE DADOS E ANÁLISE EXPLORATÓRIA

Importação dos dados

Caminho do projeto

```
PATH = "..;/proj_eSIC_v10/textmining_pt/DATA/"
```

Importação ee estrutura dos dados

Tabela1: Pedidos e-SIC

- Pedidos e-SIC

Estrutura dos dados

```
glimpse(Pedidos_eSIC)
```

```
## Observations: 625
## Variables: 9
## $ Protocolo                                  <dbl> 1.685301e+16, 1.860...
## $ `Órgão Superior`                           <chr> "EPE - Empresa de P...
## $ `Data de Abertura`                         <dttm> 2017-08-19 20:26:4...
## $ `Prazo de Atendimento`                     <dttm> 2017-09-11 23:59:5...
## $ Situação                                   <chr> "Respondido", "Resp...
## $ `Descrição do Pedido`                      <chr> "A Empresa de Pesqu...
## $ `Descrição da Forma de Resposta do Pedido` <chr> "Pelo sistema (com ...
## $ `Resumo da Solicitação`                    <chr> "Empresa de Pesquis...
## $ `Data da Resposta`                         <dttm> 2017-08-30 21:19:1...
```

Tabela2: Respostas Diretorias da EPE

- Respostas e-SIC (DIRETORIAS EPE)

Estrutura dos dados

```
glimpse(Respostas_EPE)
```

```
## Observations: 705
## Variables: 3
## $ ProtocoloPedido                     <dbl> 9.993800e+16, 9.993800e+16...
## $ DataRegistro                        <dttm> 2015-07-24, 2015-07-28, 2...
## $ DiretoriaEPE_ResponsavelPelaDemanda <chr> "DGC", "DEA", "DEA", "DEE"...
```

Tabela3: Stopwords

- Stopwords

```
FILE2 = "DATA/stopwords_PT_FINAL.csv"
stopwords_pt = read.csv(paste0(PATH,FILE2), sep = ';', header = F, encoding = "UTF-8")
stopwords_pt = stopwords_pt[,-2];
cat(paste0("O nosso vetor de stopwords contém ",length(stopwords_pt), " palavras únicas"))
```

```
## O nosso vetor de stopwords contém 704 palavras únicas
```

```
## dim(stopwords_pt); class(stopwords_pt)
stopwords_pt = as.character(stopwords_pt)
stopwords_pt[1:14]
```

```
##  [1] "a"       "à"       "acerca"  "acesso"  "adeus"   "agora"   "aí"
##  [8] "ai"      "ainda"   "alem"    "além"    "algmas"  "algo"    "algumas"
```

Tabelas4,5,6: Dicionários de variáveis e-SIC

- Dicionário > BASE DE DADOS - REAL PRO TEXTO DO TCC

Dicionário de variáveis - PEDIDOS

```
dicionario = "DATA/Dicionario-Dados-Exportacao.txt"
dic_pedidos = read.delim(dicionario, sep = "-", skip = 3, header = FALSE, nrows = 21) %>%
  select(-V1)
colnames(dic_pedidos) = c("Nome das variáveis", "Tipo e descrição da variável")
#dimnames(dic_pedidos); View(dic_pedidos)
```

Dicionário de variáveis - RECURSOS

```
dic_recursos = read.delim(dicionario, sep = "-", skip = 30, header = FALSE, nrows = 17) %>%
  select(-V1)
colnames(dic_recursos) = c("Nome das variáveis", "Tipo e descrição da variável")
#dimnames(dic_recursos); View(dic_recursos)
```

Dicionário de variáveis - SOLICITANTES

```
dicionario = "DATA/Dicionario-Dados-Exportacao.txt"
dic_solicitantes = read.delim(file = dicionario, sep = "-", skip = 53, header = FALSE, nrows = 10) %>%
  select(-V1)
colnames(dic_solicitantes) = c("Nome das variáveis", "Tipo e descrição da variável")
#dimnames(dic_solicitantes); View(dic_solicitantes)
```

Transformação e pré-processamento dos dados

Filtra, Transforma e Unifica bases

Filtro1: tabela consulta de pedidos

Filtrando apenas as variáveis de interesse do estudo na tabela de consulta de pedidos

```
LAI = LAI %>% select(Protocolo, `Data de Abertura`, `Prazo de Atendimento`, `Descrição do Pedido`, `Resu
```

Transformação1: reescrevendo colunas

Reescrevendo o nome das variáveis de ambas tablelas

```r
colnames(LAI) = c("Protocolo", "DATA_REGISTRO", "DATA_PRAZOATEND", "DESCRI_PEDIDO",
                  "RESUMO_PEDIDO", "DATA_RESPOSTA")
colnames(LAI1) = c("Protocolo", "DATA_REGISTRO", "DIRETORIAS")
```

Análise1: Quantitativo de pedidos por diretoria

Transformação2: substitui NA por OUTROS (coluna DIRETORIAS)

```r
LAI1 =
  LAI1 %>%
  replace_na(list(DIRETORIAS = "OUTROS"))
diretorias = levels(as.factor(LAI1$DIRETORIAS))
```

Tabela1: Quantitativo de pedidos por diretoria - sem reclassificação

- Tabela 01 número de solcitações/pedidos de informação

```r
pedidos_diretoria = LAI1 %>%
 count(DIRETORIAS, sort = TRUE, name = "total_pedidos")
pedidos_diretoria %>%
  kable("latex", caption = "Quantitativo de solicitações por Diretoria/EPE via e-SIC - sem reclassifica
        booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 1: Quantitativo de solicitações por Diretoria/EPE via e-SIC - sem reclassificação

DIRETORIAS	total_pedidos
DEE	244
DEA	240
DGC	121
OUTROS	67
DPG	33

Verificamos a existência de 4 diretorias, sendo elas: *DEA*, *DEE*, *DGC*, *DPG* e *OUTROS*. Essa última é devido a existência de informações solicitadas que não são de competência de nenhuma das cinco diretorias, daí a necessidade de uma última categoria *OUTROS* para atender essas demandas.

Fica nítida o desbalanceamento do número de pedidos por categoria. Enquanto as diretorias *DEE* e *DEA* possuem, respectivamente, 244 e 240 pedidos verifica-se uma diferença grande do número de pedido das diretorias *DGC* e *DPG* e também da categoria *OUTRAS*, onde se forem somadas possuem um total de 221 pedidos conjuntamente.

A seguir, um passo importante de reclassificação será executado devido ao número pequeno de solicitações para as diretorias DGC e DPG Apenas uma solcitação existente no nosso banco de dados para essa diretoria. Iremos, portanto, unificar essa demanda à categoria *OUTROS*. A seguir, verificamos nas tabela 01 e 02 a distribuição de pedidos por diretoria antes e após reclassificação das mesmas.

Tabela1: Quantitativo de pedidos por diretoria - sem reclassificação

Respostas e-SIC - Reclassificação Diretorias

```r
LAI1 = LAI1 %>%
  mutate(DIRETORIA = ifelse(DIRETORIAS == "DGC", "OUTROS",
```

```
                              ifelse(DIRETORIAS == "DPG", "OUTROS",DIRETORIAS)))
diretorias1 = levels(as.factor(LAI1$DIRETORIA))
```

Tabela2: Quantitativo de pedidos por diretoria - após reclassificação

- Tabela 02 número de solcitações/pedidos de informação - após reclassificação

```
pedidos_diretoria1 = LAI1 %>%
 count(DIRETORIA, sort = TRUE, name = "total_pedidos")
pedidos_diretoria1 %>%
  kable("latex", caption = "Quantitativo de solicitações por Diretoria/EPE via e-SIC - após reclassifica
        booktabs = T) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 2: Quantitativo de solicitações por Diretoria/EPE via e-SIC - após reclassificação

DIRETORIA	total_pedidos
DEE	244
DEA	240
OUTROS	221

Temos, finalmente um maior balanceamento nas categorias da nossa variável resposta com 244, 240 e 221 pedidos que foram destinios à *DEE*, *DEA* e *OUTROS*, respectivamente. Onde *OUTROS* é a categoria formada com a união dos pedidos das diretorias *DGC*, *DPG* e *OUTROS*.

A reclassificação foi, também, uma decisão suportada por análises préveias do presente estudo. Foi avaliada a viabilidade de aplicar o estudo com as categorias originais, entretanto na fase de modelagem preditiva o desempenho do modelo do Random Forest foi muito inferior comparado ao modelo após reclassificação. Um motivo plausível para a melhoria de performance pode ser por conta do maior balanceamento entre as categorias da variável resposta **Diretoria**, em questão.

- Unificando as Bases

É necessário, agora, unificar as bases de dados pertinentes a solicitações e respostas.

Join1: União das bases em questão

```
LAI1 = LAI1 %>% select(-DATA_REGISTRO); #dim(LAI1)
DB = left_join(x = LAI, y = LAI1, by = "Protocolo") %>%
  drop_na()
#View(head(DB))
```

Ver Anexo 01 c/ amostra dos dados da tabela que serpá utilizada para manipulação daqui pra frente.

Mineração de texto

Palavras por pedido

Análise2: distribuição de frequência de palavras por diretoria e algumas estatísticas descritivas

Ferramentas

Iniciamos as manipulações utilizando recursos da função `unnest_tokens( )` do pacote `library(tidytext)` que nos permite trabalhar com textos em um formato `tidy`, ou seja que coloca uma palavra por linha em

uma única coluna, formando, assim, *termos/palavras* por linha. Utilizamos, também, ainda os recursos do pacote `library(diplyr)` para, posteriormente, agrupar esses termos por diretoria e calcular a frequência dos *termos*.

Verificamos que as 10 palavras mais frequentes em todos os pedidos realizados são palavras sem acréscimo contextual, pois essas não acrescentam nenhum sentido semântico como, por exemplo: preposições (de, da, do, para, em, no), conjunção (e) e artigos(o,a).

Citar o que é preoposição.

Tabela3: Palavras mais frequentes

- Tabela 03 Palavras mais frequentes no conjunto de solicitações por diretoria

```
library(tidytext)
palavras <- DB %>%
  unnest_tokens(palavra, DESCRI_PEDIDO) %>%
  count(palavra, sort = TRUE) %>%
  ungroup()

palavras[0:10,] %>%
  kable("latex", caption = "Principais palavras com stopwords",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 3: Principais palavras com stopwords

palavra	n
de	3.336
a	1.198
e	1.026
o	845
do	752
da	732
para	616
em	535
que	527
no	497

Tabelas4,5,6: Palavras mais frequentes por diretoria

Tabelas4: Palavras mais frequentes DEA

- Tabela 04 Palavras mais frequentes no conjunto de solicitações por diretoria

```
palavras_diretoria <- DB %>%
  unnest_tokens(palavra, DESCRI_PEDIDO) %>%
  count(DIRETORIA,palavra, sort = TRUE) %>%
  ungroup() %>%  droplevels() %>% drop_na()


palavras_diretoria %>%
  filter(DIRETORIA == "DEA") %>%
```

```r
    top_n(n = 10) %>%
  kable("latex", caption = "Principais palavras com stopwords (DEA)",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
## Selecting by n
```

Table 4: Principais palavras com stopwords (DEA)

DIRETORIA	palavra	n
DEA	de	1.193
DEA	a	367
DEA	e	331
DEA	o	276
DEA	do	256
DEA	da	226
DEA	dados	210
DEA	energia	210
DEA	para	210
DEA	no	206

Tabelas5: Palavras mais frequentes DEE

- Tabela 05 Palavras mais frequentes no conjunto de solicitações por diretoria

```r
palavras_diretoria %>%
  filter(DIRETORIA == "DEE") %>%
    top_n(n = 10) %>%
  kable("latex", caption = "Principais palavras com stopwords (DEA)",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
## Selecting by n
```

Table 5: Principais palavras com stopwords (DEA)

DIRETORIA	palavra	n
DEE	de	998
DEE	a	367
DEE	e	283
DEE	do	260
DEE	o	248
DEE	da	236
DEE	para	215
DEE	que	156
DEE	energia	155
DEE	no	153

Tabelas6: Palavras mais frequentes DEA

- Tabela 06 Palavras mais frequentes no conjunto de solicitações por diretoria

```r
palavras_diretoria %>%
  filter(DIRETORIA == "OUTROS") %>%
    top_n(n = 10) %>%
  kable("latex", caption = "Principais palavras com stopwords (OUTROS)",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

```
## Selecting by n
```

Table 6: Principais palavras com stopwords (OUTROS)

DIRETORIA	palavra	n
OUTROS	de	1.145
OUTROS	a	464
OUTROS	e	412
OUTROS	o	321
OUTROS	da	270
OUTROS	do	236
OUTROS	que	197
OUTROS	em	196
OUTROS	para	191
OUTROS	ou	171

Mesmo assim, abrindo para cada uma das 3 possíveis cateogorias da variável **Diretoria** temos que as principais palavras não agregam nenhum valor semântico, exceto pela palavra energia que apareceu na oitava e nona colocação de maior frequência dos documentos de pedidos enviados à *DEA* e *DEE*, respectivamente. Isso devido ao excesso de uso de **stop words** em textos humanos.

Nos passos seguintes iremos remover essas palavras, **stop words**, e trabalhar apenas com palavras de sentido semântico relevante aos subjetivos solicitados às diretorias, acrescentando assim maior assertividade na classificação do nosso modelo, objetivo principal desse estudo.

Verificamos, antes disso, o total, freq. e média de palavras por diretoria, bem como comparações 2 a 2 para cada uma das categorias.

Análise2: Comparação de freq. de palavras por diretoria

- Total de palavras por diretoria, total de pedidos por diretoria e número médio de palavras por pedido e diretoria

```r
total_palavras = palavras_diretoria %>%
  group_by(DIRETORIA) %>%
  summarize(total_palavras = sum(n))

total_palavras = left_join(x = total_palavras, y = pedidos_diretoria1,
                           by = "DIRETORIA") %>%
mutate(media_palavras_porpedidoEdiretoria = total_palavras/total_pedidos)
```

Tabelas7: Total de palavras por diretoria, total de pedidos por diretoria e número médio de palavras por pedido e diretoria

- Total de palavras por diretoria, total de pedidos por diretoria e número médio de palavras por pedido e diretoria

```r
total_palavras %>%
  kable("latex", caption = "Total de palavras, total de pedidos e número médio de palavras
        por pedido e diretoria",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 7: Total de palavras, total de pedidos e número médio de palavras por pedido e diretoria

DIRETORIA	total_palavras	total_pedidos	media_palavras_porpedidoEdiretoria
DEA	14.808	240	61,70000
DEE	13.429	244	55,03689
OUTROS	15.395	221	69,66063

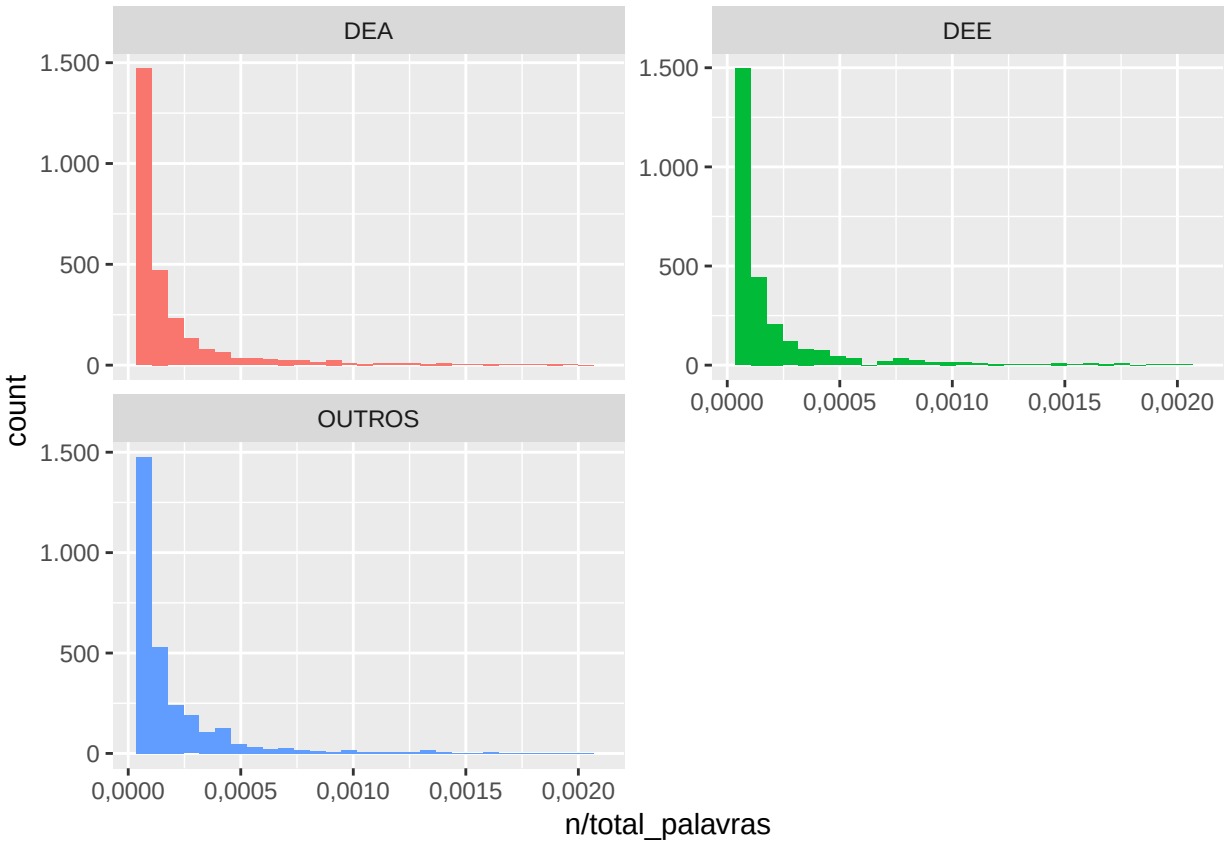
Temos que o número médio de palavras por pedido é parecido entre as diretorias. com médias de 55 palavras por pedido para DEE e 69,7 e 61,7, respectivamente para DEA e OUTROS.

Figura1: Distribuição de frequência de termos por diretoria

- Distribuição da freq. de palavras usadas em solicitações por diretoria (histograma)

```r
diretoria_palavras <- DB %>%
  unnest_tokens(palavra, DESCRI_PEDIDO) %>%
  count(DIRETORIA, palavra, sort = TRUE) %>%
  ungroup()

diretoria_palavras = left_join(diretoria_palavras, total_palavras, by = "DIRETORIA")

library(ggplot2)
gcomma <- function(x) format(x, big.mark = ".", decimal.mark = ",", scientific = FALSE)

ggplot(diretoria_palavras, aes(n/total_palavras, fill = DIRETORIA)) +
  geom_histogram(show.legend = FALSE) + xlim(NA, 0.0021) +
facet_wrap(~DIRETORIA, ncol = 2, scales = "free_y") +
  scale_y_continuous(labels=gcomma)  +
  scale_x_continuous(labels=gcomma, limits = c(NA, 0.0021))
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which
## will replace the existing scale.

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 175 rows containing non-finite values (stat_bin).

## Warning: Removed 3 rows containing missing values (geom_bar).
```



Pelos histogramas fica claro que as distribuições da frequência de termos por diretoria possuem caudas mais alongadas à direita, isso sem contar algumas frequências que não foram evidenciadas nessas figuras por questões de escala, do contrário seria possível, apenas, ver a frequência das palavras mais recorrentes no texto que, como vimos até o momento, são as de menor relevância em contexto e semântica.

Sabemos, portanto, que queremos encontrar valor exatamente nas partes mais longas à direita das distribuições de frequência de termos, uma vez que ali se encontram as palavras de maior valor contextual.

Logo, a seguir, usamos da definição da lei **Zipf** que afirma que a frequência que uma palavra (ou termo) aparece em um documento é inversamente proporcional ao seu ranque.

> lei de Zipf's

Citar, aqui, "There are very long tails to the right for these novels (those extremely common words!) that we have not shown in these plots. These plots exhibit similar distributions for all the novels, with many words that occur rarely and fewer words that occur frequently." pág. 31 (Silge, Robinson). Que averigua que documentos de texto tendem a ter distribuições de frequência de palavras similar, por conta das stopwords.

Ainda de acordo com os autores, "Distributions like those shown in Figure 3-1 are typical in language. In fact, those types of long-tailed distributions are so common in any given corpus of natural lan- guage (like a book, or a lot of text from a website, or spoken words) that the relation- ship between the frequency that a word is used and its rank has been the subject of study." e por essa razão e a relação verificada por George Zipf da relação inversa entre freq. de palavra e ranque tiramos valor dos documentos partindo dessas premissas.

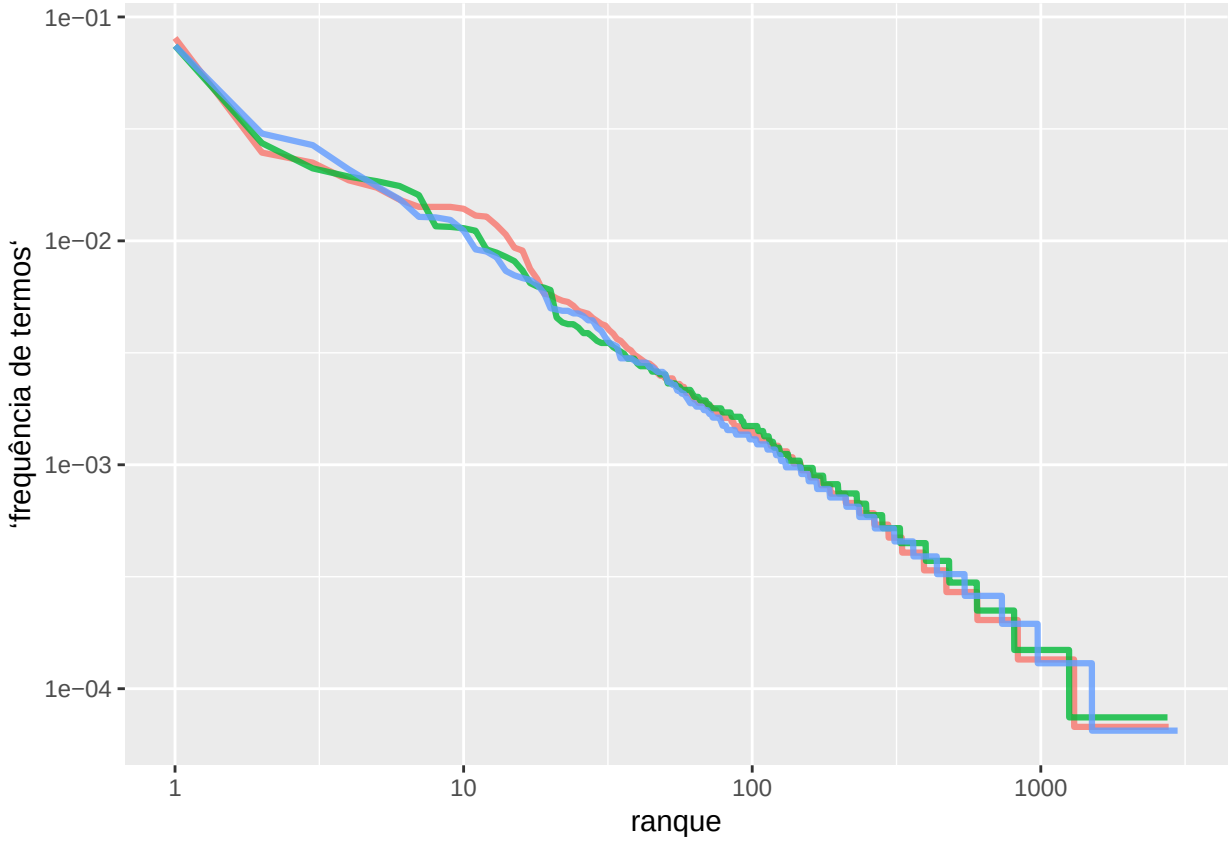
- Ranque de palavras pela pela lei de **Zipf**

```
freq_by_rank <- diretoria_palavras %>%
group_by(DIRETORIA) %>%
mutate(ranque = row_number(),
`frequência de termos` = n/total_palavras)
```

Figura1: Lei de Zipf

- Zipf's law

```
#plot1
freq_by_rank %>%
ggplot(aes(ranque, `frequência de termos`, color = DIRETORIA)) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) + scale_x_log10() +
  scale_y_log10()
```



Vemos que exatamente nas extremidades do gráfico tem-se uma não sobreposição de frequências por diretoria. Detalhe que o gráfico, em questão, está na escala logarítmica no eixo x (ranque) e eixo y (freq. de termos). Plotando desta forma, a relação inversamente proporcional terá uma inclinação constante e negativa.

Tendo em vista, portanto, que o gráfico referido está em cordenadas log-log e dado a semelhança de todos os documentos de texto das diferentes diretorias, afirmamos que para todas as diretorias pela Lei de **Zipf** a relação entre ranque e freq. de termos assumirá, sempre, uma inclinação negativa, ou seja,

Daí, aplicando a escala log-log temos que e podemos aplicar um ajuste a fim de encontrar um intercepto e coef. angular para traçar no gráfico anterior.

$$frequência \propto \frac{1}{ranque} \implies log(frequência) \propto log\left(\frac{1}{ranque}\right)$$

Reescrever e exlicar a seguimentação em 3 partes como uma "lei de potenciacao dividida em 3 partes" e então utilizar do seguimento do meio, onde as freq. de ternos sao mais semelhantes para diferentes ranques das diferentes diretorias. Fica claro pela eq.

"Notice that Figure 3-2 is in log-log coordinates. We see that all six of Jane Austen's novels are similar to each other, and that the relationship between rank and fre- quency does have negative slope. It is not quite

constant, though; perhaps we could view this as a broken power law with, say, three sections. Let's see what the exponent of the power law is for the middle section of the rank range."

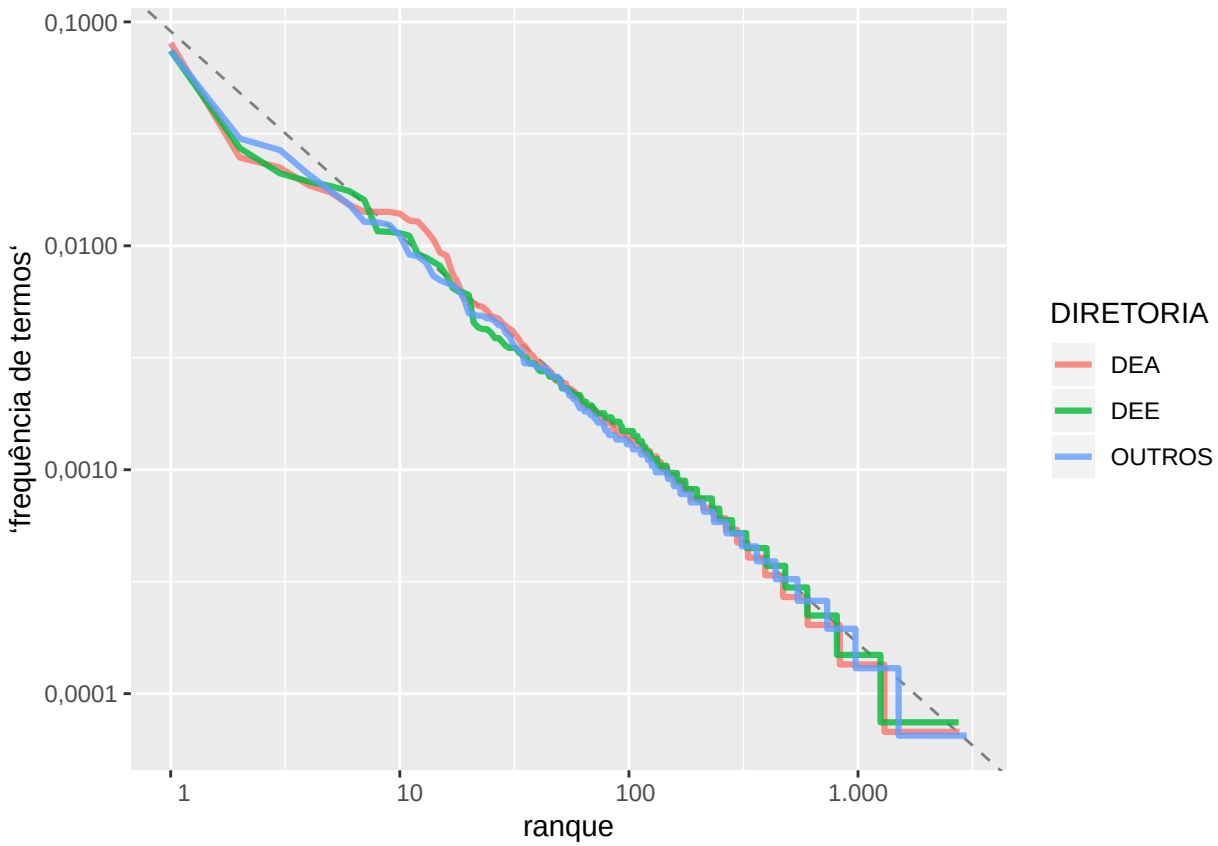
```
rank_subset <- freq_by_rank %>%
      filter(ranque < 500, ranque > 50)

(zipf_ajusteloglog <- lm(log10(`frequência de termos`) ~ log10(ranque),
                         data = rank_subset))
```

```
##
## Call:
## lm(formula = log10(`frequência de termos`) ~ log10(ranque),
##     data = rank_subset)
##
## Coefficients:
##   (Intercept)  log10(ranque)
##       -1.0419        -0.9112
```

Finalmente, traçando e sobrepondo o gráfico anterior com os valores de initercepto e coeficiente angular obtidos no ajuste do passo anterior temos a figura a seguir.

Figura1: Lei de Zipf + ajuste log-log

```
freq_by_rank %>%
ggplot(aes(ranque, `frequência de termos`, color = DIRETORIA)) +
geom_abline(intercept = coefficients(zipf_ajusteloglog)[1], slope = coefficients(zipf_ajusteloglog)[2],
geom_line(size = 1.1, alpha = 0.8, show.legend = TRUE) +
scale_x_log10(labels=gcomma) +
scale_y_log10(labels=gcomma)
```



The Bind tf_idf

Fundamentar o uso da estatística **tf_idf**, bem como descrever a definição.

> The idea of tf-idf is to find the important words for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that are not used very much in a collection or corpus of documents, in this case, the group of Jane Austen's novels as a whole. Calculating tf-idf attempts to find the words that are important (i.e., common) in a text, but not too common. Let's do that now. The bind_tf_idf function in the tidytext package takes a tidy text dataset as input with one row per token (term), per document. One column (word here) contains the terms/tokens, one column contains the documents (book in this case), and the last necessary column contains the counts, or how many times each document contains each term (n in this example). We calculated a total for each book for our explora- tions in previous sections, but it is not necessary for the bind_tf_idf function; the table only needs to contain all the words in each document.

```
round_df <- function(x, digits) {
    # round all numeric variables
    # x: data frame
    # digits: number of digits to round
    numeric_columns <- sapply(x, mode) == 'numeric'
    x[numeric_columns] <-  round(x[numeric_columns], digits)
    x
}
```

- Palavras mais relevantes de acordo com a estatística **tf_idf**

```r
diretoria_palavras_tfidf <- diretoria_palavras %>%
  bind_tf_idf(palavra, DIRETORIA, n) %>%
  select(-total_palavras, -total_pedidos, -media_palavras_porpedidoEdiretoria) %>%
  arrange(desc(tf_idf))

#options(digits=4)
set.seed(7456)
amostra1 = sample(seq(1:dim(diretoria_palavras_tfidf)[1]), 10, replace = FALSE)
round_df(diretoria_palavras_tfidf[amostra1,],5) # %>%
```

```
## # A tibble: 10 x 6
##    DIRETORIA palavra          n       tf   idf   tf_idf
##    <chr>     <chr>        <dbl>    <dbl> <dbl>    <dbl>
##  1 DEA       oil              2 0.000140 1.10  0.000150
##  2 DEE       obrigado        31 0.00231  0     0
##  3 DEE       cópias           1 0.000070 0     0
##  4 DEE       pelos            5 0.00037  0     0
##  5 OUTROS    suas             5 0.00032  0.405 0.000130
##  6 OUTROS    cabe             1 0.00006  0.405 0.00003
##  7 DEE       ifsul            1 0.000070 1.10  0.00008
##  8 OUTROS    estabelecida     1 0.00006  1.10  0.000070
##  9 DEE       valor           10 0.00074  0     0
## 10 DEE       rica             1 0.000070 0.405 0.00003
```

```r
  kable("latex", caption = "Total de palavras, total de pedidos e número médio de palavras
        por pedido e diretoria",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 8: Total de palavras, total de pedidos e número médio de palavras por pedido e diretoria

x
latex

A estatística faz um trabalho brilhante ao ressaltar as palavras mais relevantes dentro de cada conjunto de documentos (diretorias). As tabelas a seguir mostram as 10 palavras mais relevantes de acordo com a estatística tf_idf por diretoria

Tabela8: top 12 termos ordenados pela estatística tf_idf (DEE)

```r
round_df(diretoria_palavras_tfidf,5) %>%
  filter(DIRETORIA == "DEE") %>%
  top_n(12,tf_idf) %>%
  kable("latex", caption = "Top 10 termos (DEE)",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Tabela9: top 12 termos ordenados pela estatística tf_idf (DEA)

```r
round_df(diretoria_palavras_tfidf,5) %>%
  filter(DIRETORIA == "DEA") %>%
  top_n(12,tf_idf) %>%
  kable("latex", caption = "Top 10 termos (DEA)",
```

Table 9: Top 10 termos (DEE)

DIRETORIA	palavra	n	tf	idf	tf_idf
DEE	leilão	48	0,00357	1,09861	0,00393
DEE	eólica	29	0,00216	0,40547	0,00088
DEE	cadastrados	10	0,00074	1,09861	0,00082
DEE	cálculos	8	0,00060	1,09861	0,00065
DEE	fotovoltaicos	8	0,00060	1,09861	0,00065
DEE	parâmetros	8	0,00060	1,09861	0,00065
DEE	puc	8	0,00060	1,09861	0,00065
DEE	módulos	7	0,00052	1,09861	0,00057
DEE	kv	6	0,00045	1,09861	0,00049
DEE	porto	6	0,00045	1,09861	0,00049
DEE	suprimento	6	0,00045	1,09861	0,00049
DEE	termoelétricas	6	0,00045	1,09861	0,00049

```
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Table 10: Top 10 termos (DEA)

DIRETORIA	palavra	n	tf	idf	tf_idf
DEA	municípios	11	0,00074	1,09861	0,00082
DEA	nuclear	9	0,00061	1,09861	0,00067
DEA	faixa	8	0,00054	1,09861	0,00059
DEA	kwh	7	0,00047	1,09861	0,00052
DEA	porcentagem	7	0,00047	1,09861	0,00052
DEA	balanço	17	0,00115	0,40547	0,00047
DEA	solar	17	0,00115	0,40547	0,00047
DEA	condicionado	6	0,00041	1,09861	0,00045
DEA	eletrobras	6	0,00041	1,09861	0,00045
DEA	figura	6	0,00041	1,09861	0,00045
DEA	ambiental	15	0,00101	0,40547	0,00041
DEA	mwmed	14	0,00095	0,40547	0,00038

Tabela10: top 10 termos ordenados pela estatística tf_idf (OUTROS)

```
round_df(diretoria_palavras_tfidf,5) %>%
  filter(DIRETORIA == "OUTROS") %>%
  top_n(12,tf_idf) %>%
  kable("latex", caption = "Top 10 termos (DEA)",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Ou simplesmente verificamos através de um gráfico

Figura2: Termos mais relevantes por diretoria pela estatística tf_idf

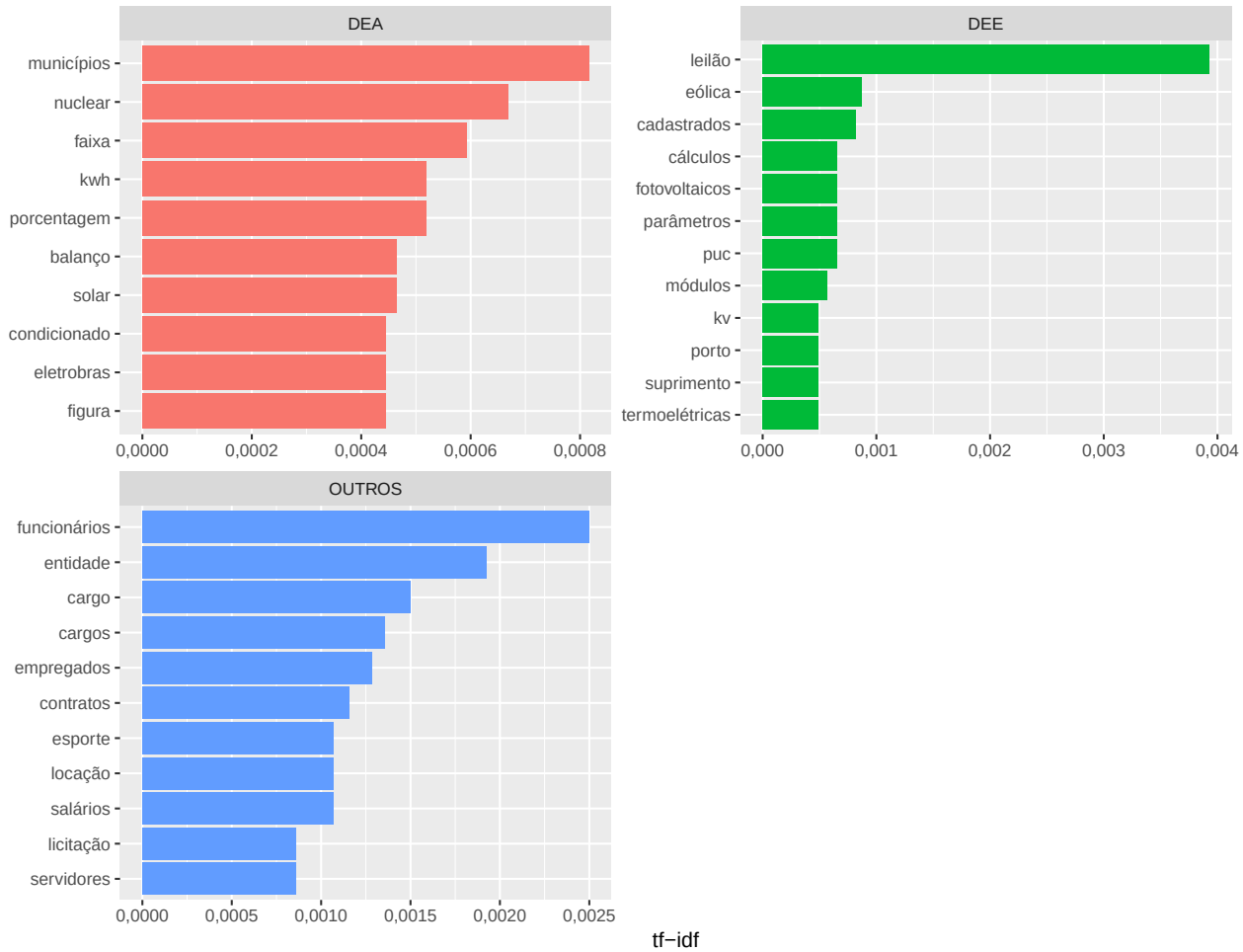
Table 11: Top 10 termos (DEA)

DIRETORIA	palavra	n	tf	idf	tf_idf
OUTROS	funcionários	35	0,00227	1,09861	0,00250
OUTROS	entidade	27	0,00175	1,09861	0,00193
OUTROS	cargo	21	0,00136	1,09861	0,00150
OUTROS	cargos	19	0,00123	1,09861	0,00136
OUTROS	empregados	18	0,00117	1,09861	0,00128
OUTROS	contratos	44	0,00286	0,40547	0,00116
OUTROS	esporte	15	0,00097	1,09861	0,00107
OUTROS	locação	15	0,00097	1,09861	0,00107
OUTROS	salários	15	0,00097	1,09861	0,00107
OUTROS	licitação	12	0,00078	1,09861	0,00086
OUTROS	servidores	12	0,00078	1,09861	0,00086
OUTROS	concurso	30	0,00195	0,40547	0,00079

```
diretoria_palavras <- DB %>%
  unnest_tokens(palavra, DESCRI_PEDIDO) %>%
  count(DIRETORIA, palavra, sort = TRUE) %>%
  ungroup()
diretoria_palavras = left_join(diretoria_palavras, total_palavras, by = "DIRETORIA")
```

Figura3: Top 10 termos por diretoria (ordenados pela estatística tf_idf e com stop words e sem stemming

```
plot_diretoria_palavras <- diretoria_palavras %>%
  bind_tf_idf(palavra, DIRETORIA, n) %>%
  arrange(desc(tf_idf)) %>%
  mutate(palavra = factor(palavra, levels = rev(unique(palavra)))) %>%
  mutate(DIRETORIA = factor(DIRETORIA, levels = c("DEA","DEE","OUTROS")))
#View(head(plot_diretoria_palavras))
#jpeg("02_freq_palavras_dir.jpeg")
plot_diretoria_palavras %>%
  group_by(DIRETORIA) %>%
  top_n(10, tf_idf) %>%
  ungroup() %>%
  mutate(palavra = reorder(palavra, tf_idf)) %>%
  ggplot(aes(palavra, tf_idf, fill = DIRETORIA)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~DIRETORIA, ncol = 2, scales = "free") +
  coord_flip() +
  scale_y_continuous(labels=gcomma)
```




```
#dev.off()
```

Filtrando um pedaço de texto

```
DB %>%
  filter(str_detect(DESCRI_PEDIDO, "r0")) %>%
  select(DESCRI_PEDIDO) %>%
  head()
```

```
## # A tibble: 6 x 1
##   DESCRI_PEDIDO
##   <chr>
## 1 "Prezados,\n\nSolicitamos que o deck do Newave 22.6 utilizado para Revis~
## 2 Gostaria de ter acesso à Nota Técnica EPE-DEE-RE-097/2016-r0, pois não a~
## 3 "Solicitamos para nossa análise cópias dos relatórios nºs EPE-DEE-RE-147~
## 4 Cópia do documento EPE-DEE-RE-083/2010-r0, "Estudo de Integração das Usi~
## 5 Solicito cópia da Nota Técnica EPE-DEE-RE-097/2016-r0
## 6 "Bom dia, por gentileza gostaria ter acesso ao parecer técnico EPE-DEE-P~
```

Uma limpeza removendo palavras sem significado semântico (**stop words**) pode auxiliar o algoritmo a retornar palavras ainda mais acertivas, bem como o tratamento de **stemming**, abordados a seguir.

Stemming

Podemos diminuir redundâncias por parte do algoritmo ensinando-o a compreender palavras que podem estar escritas de forma diferente mas que em significado semântico são semelhantes. Para isso, analisamos o radical de palavras com um mesmo prefixo mas com sufixos diferentes seja por quisistos como gênero ou plural.

Exemplos:

leilão $\propto$ leilões estado $\propto$ estados região $\propto$ regiões

Usando o pacote `ptstem`

```
library(ptstem)
stemming1 = ptstem(DB$DESCRI_PEDIDO)
```

- Frequência de palavras por diretoria do stemming 1

```
diretoria_palavras_stem1 <- DB %>%
  mutate(DESCRI_PEDIDO = stemming1) %>%
  unnest_tokens(palavra, DESCRI_PEDIDO) %>%
  count(palavra, sort = TRUE) %>%
  ungroup()
```

```
cat(paste0("Utilizando o algoritmo de stemming do pacote 'ptstem' o número de palavras chaves sem stemm
```

```
## Utilizando o algoritmo de stemming do pacote 'ptstem' o número de palavras chaves sem stemming reduz
```

Usando o pacote `rslp`

```
library(rslp)
stemming2 = rslp(DB$DESCRI_PEDIDO)
```

- Frequência de palavras por diretoria do stemming 2

```
diretoria_palavras_stem2 <- DB %>%
  mutate(DESCRI_PEDIDO = stemming2) %>%
  unnest_tokens(palavra, DESCRI_PEDIDO) %>%
  count(palavra, sort = TRUE) %>%
  ungroup()
```

```
cat(paste0("Utilizando o algoritmo de stemming do pacote 'rslp' o número de palavras chaves sem stemmin
```

```
## Utilizando o algoritmo de stemming do pacote 'rslp' o número de palavras chaves sem stemming reduziu
```

Uma redução considerável no número de termos ocorreu ao usar o algoritmo `ptstem`, cerca de 33% de redução de termos versus 7% utilziando o algoritmo `rslp`, ou seja, o algoritmo ptstem foi mais eficiente na tarefa de agrupar os semelhantes (termos únicos).

Vale ressaltar, tabmbém, o tempo de processamento que ambos os algoritmos requerem.

```
temp_stem1 = proc.time()
teste0 = ptstem(DB$DESCRI_PEDIDO)
tempo_stem1 = proc.time() - temp_stem1
remove(teste0)
```

```
temp_stem2 = proc.time()
teste00 = rslp(DB$DESCRI_PEDIDO)
tempo_stem2 = proc.time() - temp_stem2
remove(teste00)
```

O tempo decorrido para processamento do algoritmo do `ptstem` foi de aproximadamente 12,5 segundos

versus 0,9 segundo decorrido para o processamento do algoritmo do `rslp`. Logo, o `rslp` é quase 14 vezes mais eficiente em termos de tempo de processamento. Além disso, o `rslp` remove acentuações e caracteres como "ç". Isso irá nos ajudar mais a frente quando utilizarmos os principais termos como variáveis binárias e preditoras do modelo de classificação.

Entretanto, o algoritmo mais lento, `ptstem`, foi mais interessante em termos de redução do número de termos únicos, cerca de 26% menos termos únicos em relação ao outro algoritmo. Além disso, por se tratar de uma base de dados relativamente pequena, 625 pedidos, e pouco mais de 4 mil termos únicos em todo o conjunto de texto, além disso vamos utilizar de um alto poder de processamento da máquina no referido estudo. Optamos, portanto, por utilizar ambos algoritmos. Vamos, primeiro, aplicar o removedor de sufixos da lingua portuguesa `rslp` seguido do `ptstem`.

Cria, antes, uma variáveil DESCRI_PEDIDO1 que repete os passos feitos aos termos quanto ao stemming so que no texto fonte.

```
### CARACTERES
DB$DESCRI_PEDIDO1 = DB$DESCRI_PEDIDO
DB$DESCRI_PEDIDO1 = gsub("-","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:.:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:,:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:':]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:!:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:?:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:-:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:__:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:;:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:/:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:(:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:):]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:%:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:º:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:°:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:ª:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("\\d+","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[0-9]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:\n\t:]"," ",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:\t:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("[:\n:]","",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("\\s+"," ",DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 = gsub("\"","",DB$DESCRI_PEDIDO1)

### STEMMINGS
DB$DESCRI_PEDIDO1 = ptstem(rslp(DB$DESCRI_PEDIDO1), algorithm = "hunspell",
                           complete = TRUE)
DB$DESCRI_PEDIDO1 = rslp(ptstem(DB$DESCRI_PEDIDO1))
DB$DESCRI_PEDIDO1 = gsub("\\s+"," ",DB$DESCRI_PEDIDO1)


### PALAVRAS
DB$DESCRI_PEDIDO1 =gsub("\\b(Leiloes)", "leilao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Leiloar)", "leilao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(leiloes)", "leilao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(leiloar)", "leilao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(leiloes)", "leilao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Energetica)", "eletrica", DB$DESCRI_PEDIDO1)
```

```r
DB$DESCRI_PEDIDO1 =gsub("\\b(energetica)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Eletricas)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(eletricas)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Eletricos)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(eletricos)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Eletrico)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(eletrico)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Eletricidade)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(eletricidade)", "eletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Termoeletricas)", "termoeletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(termoeletricas)", "termoeletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Termeletrica)", "termoeletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(termeletrica)", "termoeletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Termeletricas)", "termoeletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(termeletricas)", "termoeletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Hidreletricas)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(hidreletricas)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Hidroeletricas)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(hidroeletricas)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Hidroeletricos)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(hidroeletricos)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Hidroeletrica)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(hidroeletrica)", "hidreletrica", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Administracao)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(administracao)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Administrativo)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(administrativo)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Administrativos)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(administrativos)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Administrativa)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(administrativa)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Administrativas)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(administrativas)", "administracao", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Consumo)", "consumo", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Consumidores)", "consumo", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(consumidores)", "consumo", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Consumidor)", "consumo", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(consumidor)", "consumo", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(Consumir)", "consumo", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =gsub("\\b(consumir)", "consumo", DB$DESCRI_PEDIDO1)
#DB$DESCRI_PEDIDO1 =gsub("\\b(http)>", "", DB$DESCRI_PEDIDO1)
DB$DESCRI_PEDIDO1 =tolower(DB$DESCRI_PEDIDO1)
#View(DB$DESCRI_PEDIDO1)
```

Frequência de palavras por diretoria

```r
diretoria_palavras_stem3 <- DB %>%
  unnest_tokens(palavra, DESCRI_PEDIDO1) %>%
  count(DIRETORIA, palavra, sort = TRUE) %>%
  ungroup()
```

Filtrando um pedaço de texto

```
DB %>%
  filter(str_detect(DESCRI_PEDIDO1, "leiloes")) %>%
  select(DESCRI_PEDIDO1) %>%
  head()
```

Comparação do texto original c/ os 2 algoritmos e o final implementados após diferentes **stemmings**

```
DB$DESCRI_PEDIDO[227]
```

```
## [1] "Prezados, boa tarde. Desejo obter informações acerca da destinação dada aos honorários de sucum
```

```
ptstem(DB$DESCRI_PEDIDO[227])
```

```
## [1] "Prezados, boa tarde. Desejo obter informações acerca da destinação dada aos honorários de sucum
```

```
rslp(DB$DESCRI_PEDIDO[227])
```

```
## [1] "Prezados, boa tarde. Desejo obter informacoes acerca da destinacao dada aos honorarios de sucum
```

```
DB$DESCRI_PEDIDO1[227]
```

```
## [1] "prezados boa tarde desejo obter eletrica cerca da eletrica dados aos eletrica de eletrica no el
```

```
DB$DESCRI_PEDIDO[350]
```

```
## [1] "Prezados(as) Senhores(as),\n\nsou o Eng° Eletricista Lidinei Sergio Mesquita Neri (ex-Colaborad
```

```
ptstem(DB$DESCRI_PEDIDO[350])
```

```
## [1] "Prezados(as) Senhores(as),\n\nsou o Eng° Eletricista Lidinei Sergio Mesquita Neri (ex-Colaborad
```

```
rslp(DB$DESCRI_PEDIDO[350])
```

```
## [1] "Prezados(as) Senhores(as),\n\nsou o Eng° Eletricista Lidinei Sergio Mesquita Neri (ex-Colaborad
```

```
DB$DESCRI_PEDIDO1[350]
```

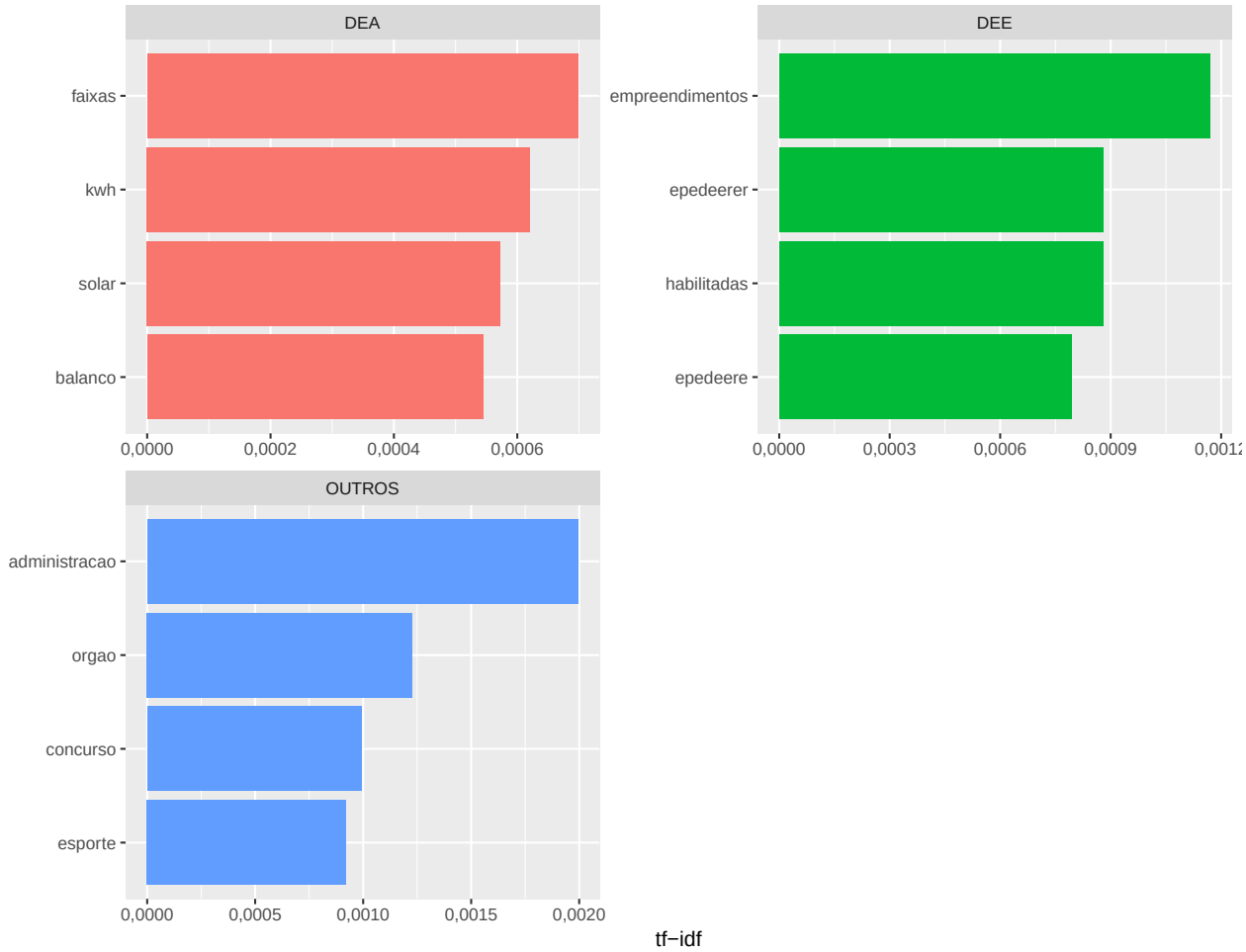
```
## [1] "prezadosas senhoresas series o eng eletricista lidinei sergio mesquita neri excolaborador da el
```

Figura4: Termos mais relevantes por diretoria pela estatística tf_idf, após stemming porém ainda com stop words

Vamos, agora, plotar as quinze palavras mais relevantes de acordo com a estatística **tf_idf**, por diretoria

```
plot_diretoria_palavras_stem <- diretoria_palavras_stem3 %>%
  bind_tf_idf(palavra, DIRETORIA, n) %>%
  arrange(desc(tf_idf)) %>%
  mutate(palavra = factor(palavra, levels = rev(unique(palavra)))) %>%
  mutate(DIRETORIA = factor(DIRETORIA, levels = c("DEA","DEE","OUTROS")))
#View(head(plot_diretoria_palavras))
#jpeg("02_freq_palavras_dir.jpeg")
plot_diretoria_palavras_stem %>%
  group_by(DIRETORIA) %>%
  top_n(4, tf_idf) %>%
  ungroup() %>%
  mutate(palavra = reorder(palavra, tf_idf)) %>%
  ggplot(aes(palavra, tf_idf, fill = DIRETORIA)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~DIRETORIA, ncol = 2, scales = "free") +
```

```
  coord_flip() +
  scale_y_continuous(labels=gcomma)
```




```
#dev.off()
```

Stopwords

Com o arquivo de **stop words** previamente inserido vamos, primeiramente, transforma-lo em um data_frame a fim de futuramente utilizá-lo para extrair do texto palavras em comum.

Freq. de palavras sem stopwords por diretoria

```
mystopwords <- data_frame(palavra = stopwords_pt)
```

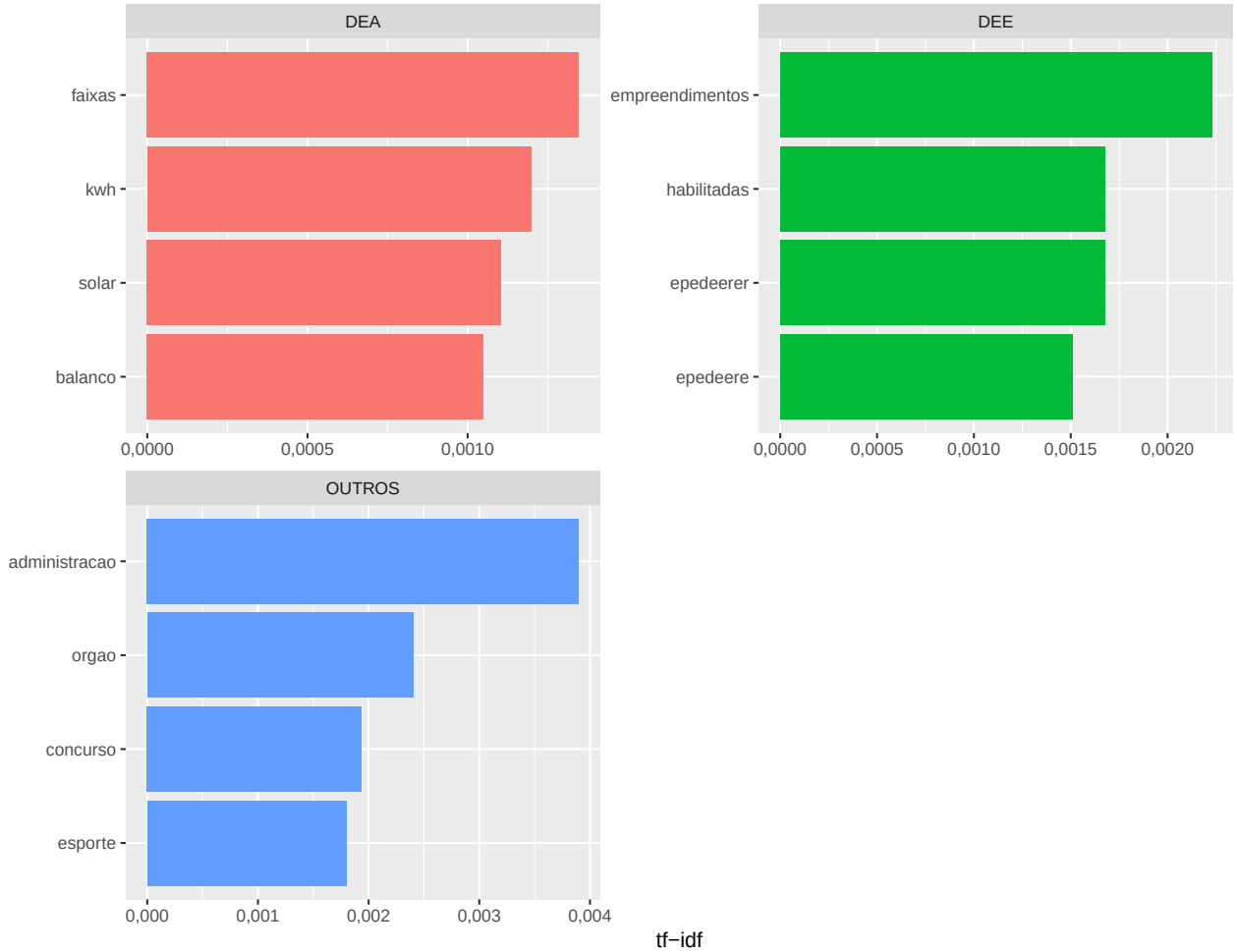
```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
diretoria_palavras_noSTOP <- anti_join(diretoria_palavras_stem3, mystopwords,
                                       by = "palavra")
```

Figura5: Termos mais relevantes por diretoria pela estatística tf_idf, após stemming e sem stop words

Sim, a remoção de **stop words** não alterou em nada a ordem das 4 palavras mais relevantes de acordo com a estatística.

```
#diretoria_palavras_noSTOP_noSTOP
plot_diretoria_palavras_noSTOP <- diretoria_palavras_noSTOP %>%
  bind_tf_idf(palavra, DIRETORIA, n) %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(palavra, levels = rev(unique(palavra)))) %>%
  mutate(DIRETORIA = factor(DIRETORIA, levels = c("DEA","DEE","OUTROS")))
#plot_diretoria_palavras_noSTOP
#windows.options(width=10, height=10)
#jpeg("03_freq_palavras_dir_nostop.jpeg")
plot_diretoria_palavras_noSTOP %>%
  group_by(DIRETORIA) %>%
  top_n(4, tf_idf) %>%
  ungroup() %>%
  mutate(palavra = reorder(palavra, tf_idf)) %>%
  ggplot(aes(palavra, tf_idf, fill = DIRETORIA)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~DIRETORIA, ncol = 2, scales = "free") +
  coord_flip() +
  scale_y_continuous(labels=gcomma)
```



```
#dev.off()
```

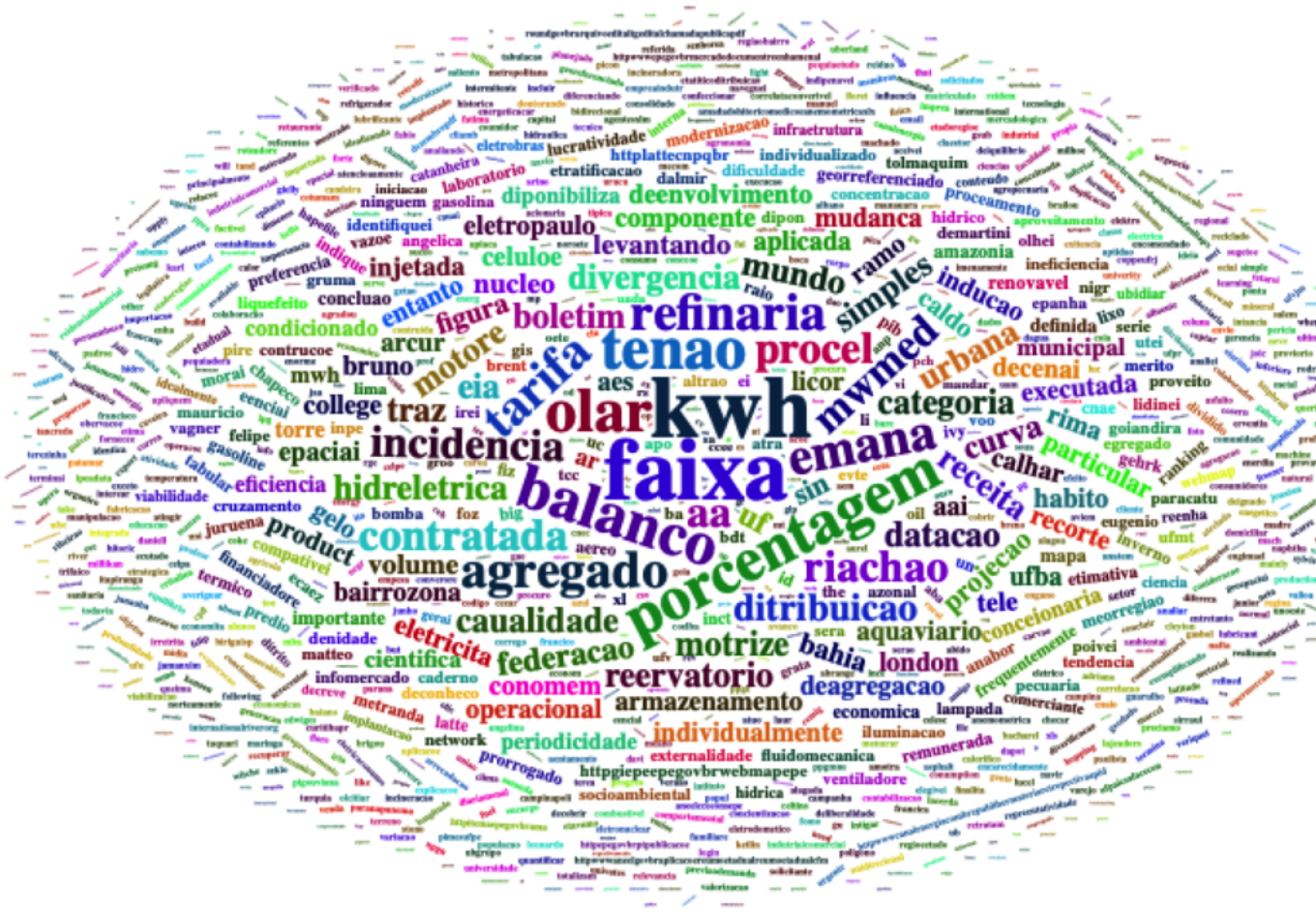
Wordcloud2 - DEE

```
set.seed(6423)
plot_diretoria_palavras_noSTOP %>%
  filter(DIRETORIA == "DEE") %>%
  select(word = palavra, freq = tf_idf) %>%
  mutate(word = as.factor(word)) %>%
  #top_n(150, freq) %>%
  as.data.frame() %>%
  wordcloud2(shuffle = TRUE, color = "random-dark", shape = "circle", size = 1.10)
```

Wordcloud2 - DEA

```
set.seed(6423)
plot_diretoria_palavras_noSTOP %>%
  filter(DIRETORIA == "DEA") %>%
  select(word = palavra, freq = tf_idf) %>%
  mutate(word = as.factor(word)) %>%
  #top_n(150, freq) %>%
```

```
  as.data.frame() %>%
  wordcloud2(shuffle = TRUE, color = "random-dark", shape = "circle", size = .25)
```



Wordcloud2 - OUTROS

```
set.seed(6423)
plot_diretoria_palavras_noSTOP %>%
  filter(DIRETORIA == "OUTROS") %>%
  select(word = palavra, freq = tf_idf) %>%
  mutate(word = as.factor(word)) %>%
  #top_n(150, freq) %>%
  as.data.frame() %>%
  wordcloud2(shuffle = TRUE, color = "random-dark", shape = "circle", size = 0.35)
```

Comparação de frequências dois a dois (sem stopwords e com stemming)

Vamos agora comparar a frequência de palavras entre diretorias. Antes disso, vamos criar documentos de texto no formato tidy separadamente para cada uma das 3 cateorias: *DEA*, *DEE* e *OUTROS*.

Gráficos de comparação de frequência de palavras por diretorias (2 a 2)

`COM STOPWORDS`

É importante ressaltar que os gráficos a seguir mostram, apenas, a comparação de frequência de palavras existentes em ambas diretorias. Ou seja, palavras existentes em apenas uma diretoria serão desconsideradas para a geração destes.

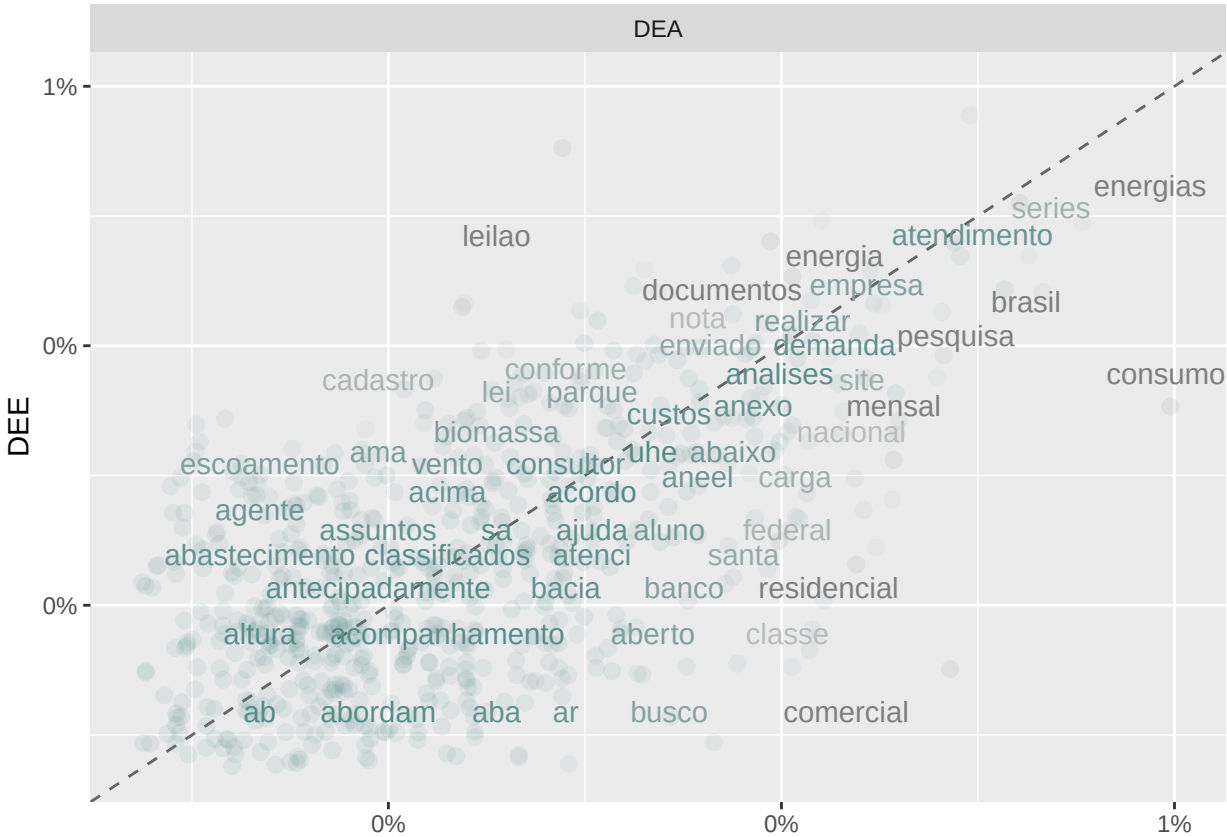
```
PROP_PALAVRA = diretoria_palavras_noSTOP %>%
    #mutate(palavra = str_extract(palavra, "[a-z']+")) %>%
    mutate(proportion = n / sum(n)) %>%
    select(-n) %>%
    spread(DIRETORIA, proportion)
```

- DEE X DEA

```
freq00 <- PROP_PALAVRA %>%
    gather(DIRETORIA, proportion, c(`DEA`))
```

```r
library(scales)
# expect a warning about rows with missing values being removed
ggplot(freq00, aes(x = proportion, y = `DEE`,
                   color = abs(`DEE` - proportion))) +
  geom_abline(color = "gray40", lty = 2) +
  geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
  geom_text(aes(label = palavra), check_overlap = TRUE, vjust = 1.5) +
  scale_x_log10(labels = percent_format(big.mark = ".", decimal.mark = ",",
                                        accuracy = 1), limits = c(NA, 0.01)) +
  scale_y_log10(labels = percent_format(big.mark = ".", decimal.mark = ",",
                                        accuracy = 1), limits = c(NA, 0.01)) +
  scale_color_gradient(limits = c(0, 0.001),
                       low = "darkslategray4", high = "gray75") +
  facet_wrap(~DIRETORIA, ncol = 1) +
  theme(legend.position="none") +
  labs(y = "DEE", x = NULL)
```

```
## Warning: Removed 2143 rows containing missing values (geom_point).

## Warning: Removed 2142 rows containing missing values (geom_text).
```




```r
cor.test(data = freq00[freq00$DIRETORIA == "DEA",],
         ~ proportion + `DEE`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and DEE
```

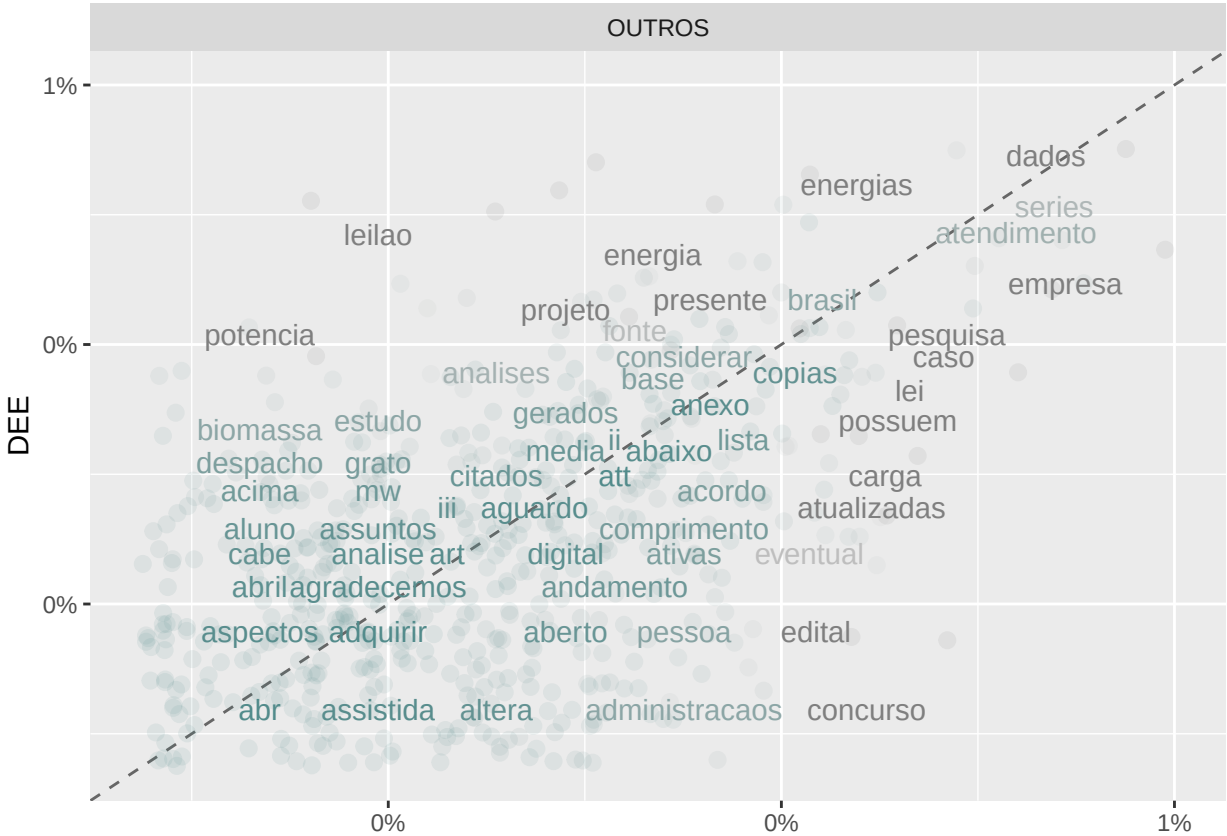
```
## t = 138.38, df = 647, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9808000 0.9858601
## sample estimates:
##       cor
## 0.9835216
```

- DEE X OUTROS

```r
freq03 <- PROP_PALAVRA %>%
    gather(DIRETORIA, proportion, c(`OUTROS`))

  library(scales)
  # expect a warning about rows with missing values being removed
  ggplot(freq03, aes(x = proportion, y = `DEE`,
                     color = abs(`DEE` - proportion))) +
    geom_abline(color = "gray40", lty = 2) +
    geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
    geom_text(aes(label = palavra), check_overlap = TRUE, vjust = 1.5) +
    scale_x_log10(labels = percent_format(big.mark = ".", decimal.mark = ",",
                                          accuracy = 1), limits = c(NA, 0.01)) +
    scale_y_log10(labels = percent_format(big.mark = ".", decimal.mark = ",",
                                          accuracy = 1), limits = c(NA, 0.01)) +
    scale_color_gradient(limits = c(0, 0.001),
                         low = "darkslategray4", high = "gray75") +
    facet_wrap(~DIRETORIA, ncol = 1) +
    theme(legend.position="none") +
    labs(y = "DEE", x = NULL)
```

```
## Warning: Removed 2217 rows containing missing values (geom_point).
```

```
## Warning: Removed 2217 rows containing missing values (geom_text).
```




```
cor.test(data = freq03[freq03$DIRETORIA == "OUTROS",],
             ~ proportion + `DEE`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and DEE
## t = 147.24, df = 571, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9847981 0.9890299
## sample estimates:
##      cor
## 0.987085
```

```
Warning messages:
1: Removed 4273 rows containing missing values (geom_point).
2: Removed 4274 rows containing missing values (geom_text).
```

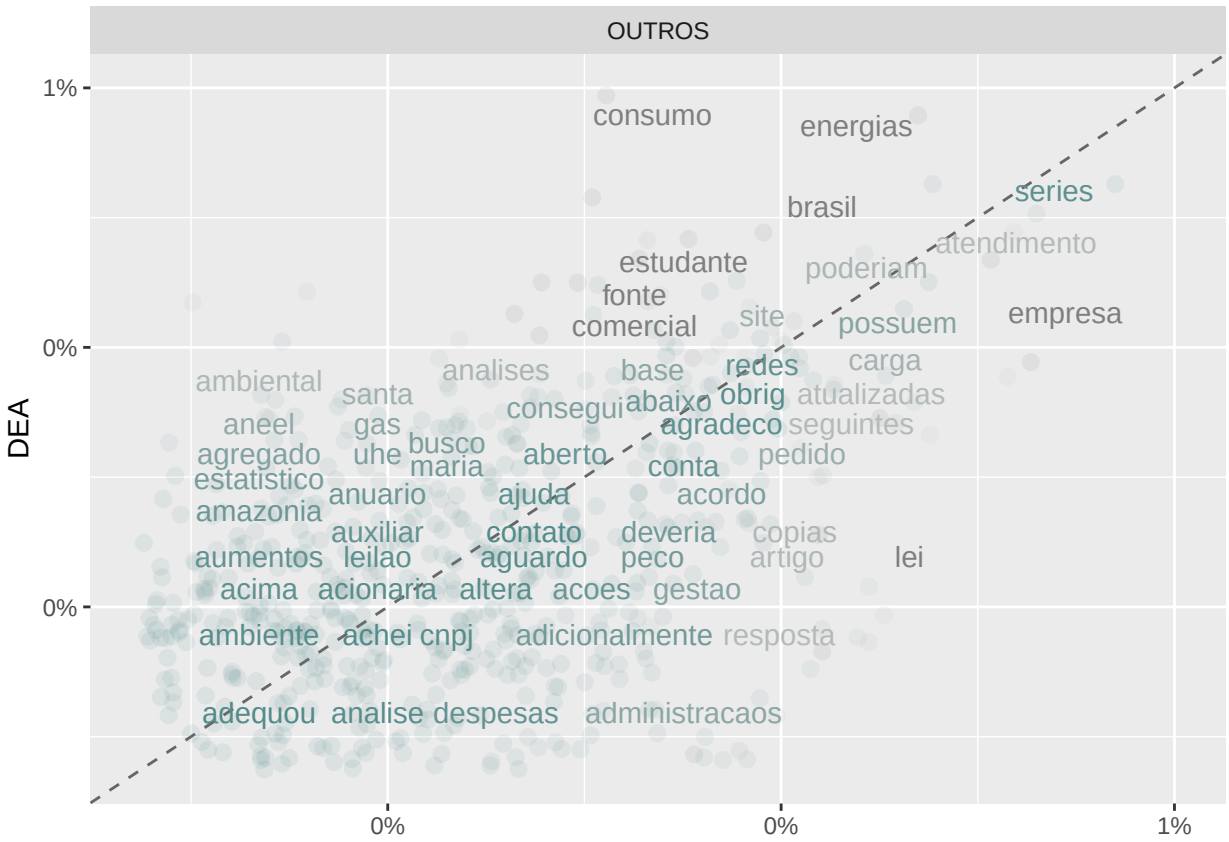
- DEA X OUTROS

```
freq06 <- PROP_PALAVRA %>%
    gather(DIRETORIA, proportion, c(`OUTROS`))

  library(scales)
  # expect a warning about rows with missing values being removed
  ggplot(freq06, aes(x = proportion, y = `DEA`,
                     color = abs(`DEA` - proportion))) +
    geom_abline(color = "gray40", lty = 2) +
```

```r
    geom_jitter(alpha = 0.1, size = 2.5, width = 0.3, height = 0.3) +
    geom_text(aes(label = palavra), check_overlap = TRUE, vjust = 1.5) +
    scale_x_log10(labels = percent_format(big.mark = ".", decimal.mark = ",",
                                          accuracy = 1), limits = c(NA, 0.01)) +
    scale_y_log10(labels = percent_format(big.mark = ".", decimal.mark = ",",
                                          accuracy = 1), limits = c(NA, 0.01)) +
    scale_color_gradient(limits = c(0, 0.001),
                         low = "darkslategray4", high = "gray75") +
    facet_wrap(~DIRETORIA, ncol = 1) +
    theme(legend.position="none") +
    labs(y = "DEA", x = NULL)
```

```
## Warning: Removed 2213 rows containing missing values (geom_point).
```

```
## Warning: Removed 2212 rows containing missing values (geom_text).
```




```
Warning messages:
1: Removed 4303 rows containing missing values (geom_point).
2: Removed 4304 rows containing missing values (geom_text).
```

```r
cor.test(data = freq06[freq06$DIRETORIA == "OUTROS",],
             ~ proportion + `DEA`)
```

```
##
##  Pearson's product-moment correlation
##
## data:  proportion and DEA
## t = 104.61, df = 577, p-value < 2.2e-16
```

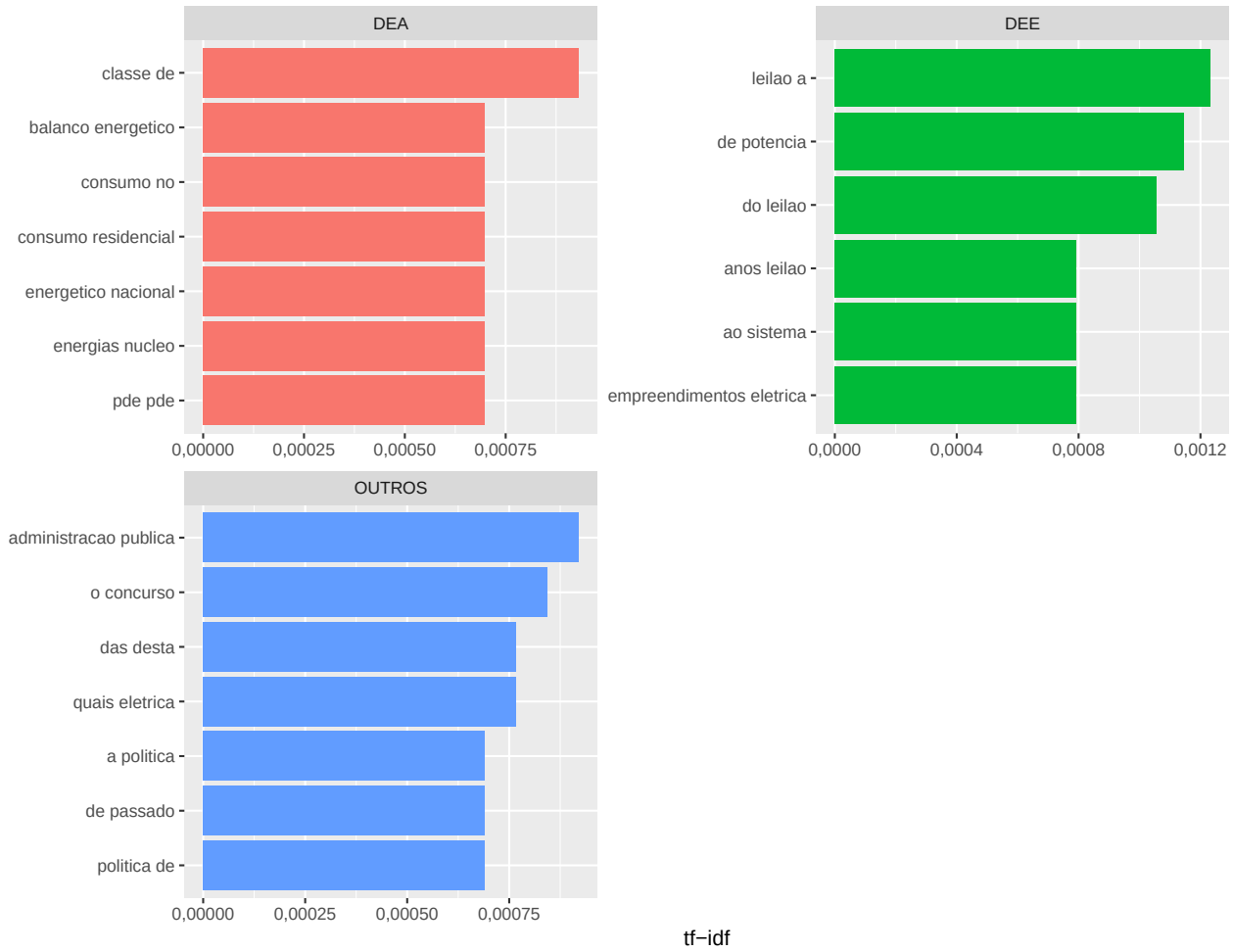
```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.9702007 0.9784153
## sample estimates:
##       cor
## 0.9746342
```

Usando bigram para n=2 palavras por token

top 6 de palavras por diretoria

Figura6: Termos (bigram) mais relevantes por diretoria pela estatística tf_idf, após stemming e sem stop words

```r
diretoria_palavras_bigram <- DB %>%
  select(DESCRI_PEDIDO1,DIRETORIA) %>%
  unnest_tokens(BIGRAM, DESCRI_PEDIDO1, token = "ngrams", n = 2) %>%
  count(DIRETORIA, BIGRAM, sort = TRUE) %>%
  ungroup()
#diretoria_palavras_bigram

plot_diretoria_palavras_bigram <- diretoria_palavras_bigram %>%
  bind_tf_idf(BIGRAM, DIRETORIA, n) %>%
  arrange(desc(tf_idf)) %>%
  mutate(BIGRAM = factor(BIGRAM, levels = rev(unique(BIGRAM)))) %>%
  mutate(DIRETORIA = factor(DIRETORIA, levels = c("DEA","DEE","OUTROS")))
#View(head(plot_diretoria_palavras_bigram))

#jpeg("02_freq_palavras_dir.jpeg")
plot_diretoria_palavras_bigram %>%
  group_by(DIRETORIA) %>%
  top_n(6, tf_idf) %>%
  ungroup() %>%
  mutate(BIGRAM = reorder(BIGRAM, tf_idf)) %>%
  ggplot(aes(BIGRAM, tf_idf, fill = DIRETORIA)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~DIRETORIA, ncol = 2, scales = "free") +
  coord_flip() +
  scale_y_continuous(labels=gcomma)
```




```
#dev.off()
```

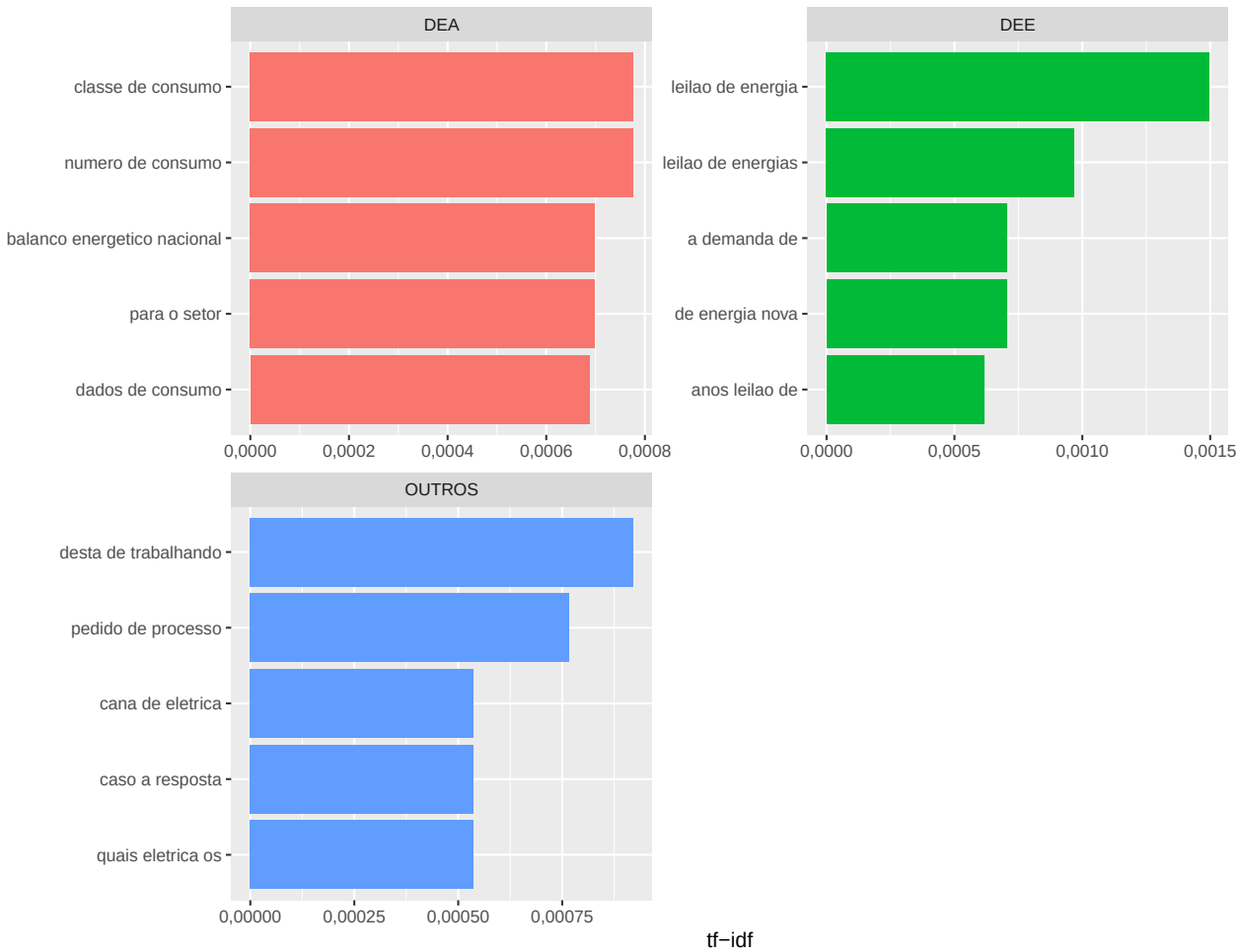
Usando bigram para n=3 palavras por token

Frequência de palavras por diretoria

Figura7: Termos (trigram) mais relevantes por diretoria pela estatística tf_idf, após stemming e sem stop words

```
diretoria_palavras_trigram <- DB %>%
  select(DESCRI_PEDIDO1,DIRETORIA) %>%
  unnest_tokens(TRIGRAM, DESCRI_PEDIDO1, token = "ngrams", n = 3) %>%
  count(DIRETORIA, TRIGRAM, sort = TRUE) %>%
  ungroup()
#diretoria_palavras_trigram

plot_diretoria_palavras_trigram <- diretoria_palavras_trigram %>%
  bind_tf_idf(TRIGRAM, DIRETORIA, n) %>%
  arrange(desc(tf_idf)) %>%
  mutate(TRIGRAM = factor(TRIGRAM, levels = rev(unique(TRIGRAM)))) %>%
  mutate(DIRETORIA = factor(DIRETORIA, levels = c("DEA","DEE","OUTROS")))
#View(head(plot_diretoria_palavras_trigram))
```

```r
#jpeg("02_freq_palavras_dir.jpeg")
plot_diretoria_palavras_trigram %>%
  group_by(DIRETORIA) %>%
  top_n(5, tf_idf) %>%
  ungroup() %>%
  mutate(TRIGRAM = reorder(TRIGRAM, tf_idf)) %>%
  ggplot(aes(TRIGRAM, tf_idf, fill = DIRETORIA)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~DIRETORIA, ncol = 2, scales = "free") +
  coord_flip() +
  scale_y_continuous(labels=gcomma)
```




```r
#dev.off()
```

tidy object into document-term matrix

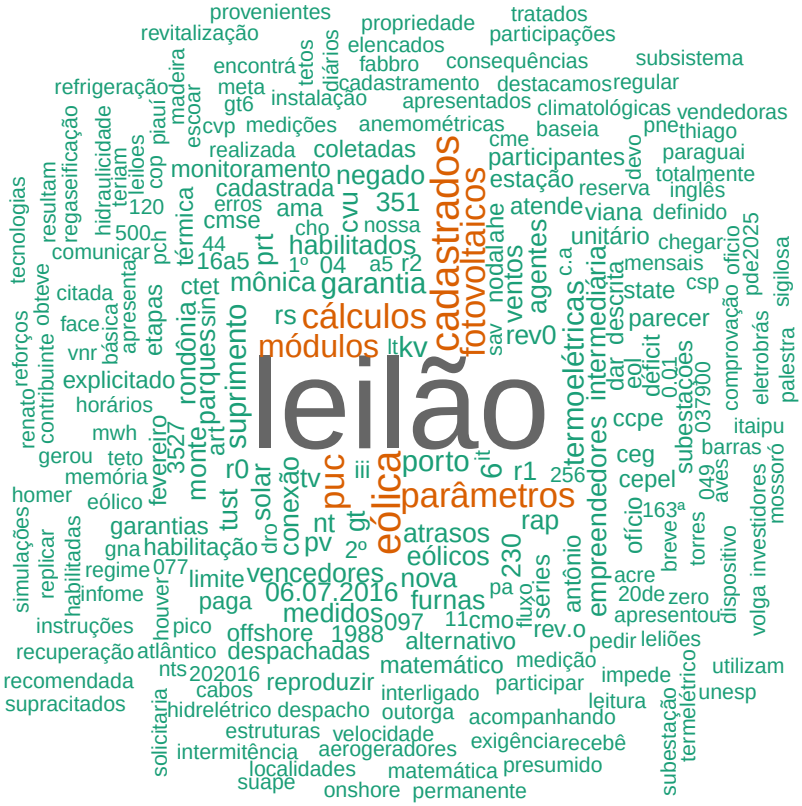
```r
plot_diretoria_palavras <- diretoria_palavras %>%
  bind_tf_idf(palavra, DIRETORIA, n) %>%
  arrange(desc(tf_idf)) %>%
  mutate(palavra = factor(palavra, levels = rev(unique(palavra)))) %>%
```

```
  mutate(DIRETORIA = factor(DIRETORIA, levels = c("DEA","DEE","OUTROS")))

dtm = plot_diretoria_palavras %>%
  cast_dtm(document = DIRETORIA, term = palavra, n)
```

Nuvem de palavras

Nuvem de palavras por diretoria - s/ steeming e/ c/ stopwords - onegram

```
#View(head(plot_diretoria_palavras))
library(wordcloud)
plot_diretorias_tf_dif = plot_diretoria_palavras %>%
  select(palavra, tf_idf, DIRETORIA) %>%
  mutate(palavra = reorder(palavra, tf_idf))

## DEE
#jpeg("XX_wordclou_tfidf_dir01_DEE.jpeg")
nuvem1 =
  plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "DEE") %>%
  select(-DIRETORIA, word = palavra,freq = tf_idf) %>%
  #top_n(150, freq) %>%
  as.data.frame()

set.seed(231321)
wordcloud(words = nuvem1$word, freq = nuvem1$freq, min.freq = 0.2,
          max.words=250, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(10, "Dark2"))
```



```
## DEA
#jpeg("XX_wordclou_tfidf_dir03_DEA.jpeg")
nuvem3 =
  plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "DEA") %>%
  select(-DIRETORIA, word = palavra,freq = tf_idf) %>%
  #top_n(150, freq) %>%
  as.data.frame()

set.seed(231321)
wordcloud(words = nuvem3$word, freq = nuvem3$freq, min.freq = 0.2,
          max.words=250, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(10, "Dark2"))
```


```
## OUTROS
#jpeg("XX_wordclou_tfidf_dir05_OUTROS.jpeg")
nuvem5 =
  plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "OUTROS") %>%
  select(-DIRETORIA, word = palavra,freq = tf_idf) %>%
  #top_n(150, freq) %>%
  as.data.frame()

set.seed(75437)
wordcloud(words = nuvem5$word, freq = nuvem5$freq, min.freq = 0.1,
          max.words=250, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(10, "Dark2"))
```


```
#View(head(plot_diretoria_palavras))
library(wordcloud2)

plot_diretorias_tf_dif = plot_diretoria_palavras %>%
  select(palavra, tf_idf, DIRETORIA) %>%
  mutate(palavra = reorder(palavra, tf_idf))
```

```
## DEE
#jpeg("XX_wordclou_tfidf_dir01_DEE.jpeg")
set.seed(233115)
plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "DEE") %>%
  top_n(150, tf_idf) %>%
  wordcloud2(shuffle = TRUE,
             color = "random-dark",
             shape = "circle")

## DGC
#jpeg("XX_wordclou_tfidf_dir01_DGC.jpeg")
set.seed(233115)
plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "DGC") %>%
  top_n(150, tf_idf) %>%
  wordcloud2()

## DEA
#jpeg("XX_wordclou_tfidf_dir01_DEA.jpeg")
set.seed(233115)
plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "DEA") %>%
  top_n(150, tf_idf) %>%
  wordcloud2()

## DPG
#jpeg("XX_wordclou_tfidf_dir04_DPG.jpeg")
set.seed(233115)
plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "DPG") %>%
  top_n(150, tf_idf) %>%
  wordcloud2()

## OUTROS
#jpeg("XX_wordclou_tfidf_dir01_OUTROS.jpeg")
set.seed(233115)
plot_diretorias_tf_dif %>%
  filter(DIRETORIA == "OUTROS") %>%
  top_n(150, tf_idf) %>%
  wordcloud2()
```

–>

Nuvem de palavras por diretoria - s/ steeming e/ou remoção de stopwords - bigram

```
r   plot_diretorias_tf_dif_bigram = DB %>%   select(DESCRI_PEDIDO,DIRETORIA) %>%   unnest_tokens(BIGRAM
DESCRI_PEDIDO, token = "ngrams", n = 2) %>%   count(DIRETORIA, BIGRAM, sort = TRUE)
%>%   bind_tf_idf(BIGRAM, DIRETORIA, n) %>%   arrange(desc(tf_idf)) %>%   mutate(BIGRAM =
factor(BIGRAM, levels = rev(unique(BIGRAM)))) %>%   mutate(DIRETORIA = factor(DIRETORIA,levels=c("DEA",
%>%   select(BIGRAM, tf_idf, DIRETORIA)
```

```
## DEE
#jpeg("XX_wordclou_tfidf_dir01_DEE.jpeg")
nuvem1.2 =
```

```
  plot_diretorias_tf_dif_bigram %>%
  filter(DIRETORIA == "DEE") %>%
  select(-DIRETORIA, word = BIGRAM,freq = tf_idf) %>%
  #top_n(150, freq) %>%
  as.data.frame()

set.seed(231321)
wordcloud(words = nuvem1.2$word, freq = nuvem1.2$freq, min.freq = 0.2,
          max.words=250, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(10, "Dark2"))
```


```
## DEA
#jpeg("XX_wordclou_tfidf_dir03_DEA.jpeg")
nuvem3.2 =
  plot_diretorias_tf_dif_bigram %>%
  filter(DIRETORIA == "DEA") %>%
  select(-DIRETORIA, word = BIGRAM,freq = tf_idf) %>%
  #top_n(150, freq) %>%
  as.data.frame()

set.seed(543453)
wordcloud(words = nuvem3.2$word, freq = nuvem3.2$freq, min.freq = 0.2,
          max.words=250, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(10, "Dark2"))
```


```
## OUTROS
#jpeg("XX_wordclou_tfidf_dir05_OUTROS.jpeg")
nuvem5.2 =
  plot_diretorias_tf_dif_bigram %>%
  filter(DIRETORIA == "OUTROS") %>%
  select(-DIRETORIA, word = BIGRAM,freq = tf_idf) %>%
  #top_n(150, freq) %>%
  as.data.frame()

set.seed(75437)
wordcloud(words = nuvem5.2$word, freq = nuvem5.2$freq, min.freq = 0.1,
          max.words=250, random.order=FALSE, rot.per=0.35,
          colors=brewer.pal(10, "Dark2"))
```

MODELAGEM - APLICAÇÃO E RESULTADOS

Preparação e partição de dados

Recapitulando, chegamos portanto, a uma base de dados donde foram aplicadas 2 diferentes técnicas de **stemming**, também a remoção de **stopwords** e fazendo uso da estatística **tf_idf** a fim de ressaltar os termos mais relevantes de cada documento de texto.

Vamos, portanto, contar o número de termos únicos dentro de cada diretoria.

Tabela11: Número de termos únicos por diretoria

```
key_DIR = plot_diretoria_palavras_noSTOP %>%
  group_by(DIRETORIA) %>%
  count(DIRETORIA)
 key_DIR %>%
   kable("latex", caption = "Número de termos por diretoria",
        booktabs = T, format.args = list(decimal.mark = ',', big.mark = ".")) %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

Vamos, agora, selecionar as $n = 250$ palavras mais importantes de cada uma das 4 diretorias e da categoria 'OUTROS'. Para isso vamos, primeiro, separar os documentos em 3 documentos distintos, um para cada diretoria.

```
n=450

termos_dir_DEE =
```

Table 12: Número de termos por diretoria

DIRETORIA	n
DEA	1.372
DEE	1.394
OUTROS	1.406

```
plot_diretoria_palavras_noSTOP %>%
filter(DIRETORIA == "DEE")
termos_DEE = termos_dir_DEE %>%
  top_n(n, tf_idf)

termos_dir_DEA =
plot_diretoria_palavras_noSTOP %>%
filter(DIRETORIA == "DEA")
termos_DEA = termos_dir_DEA %>%
  top_n(n, tf_idf)

termos_dir_OUTROS =
plot_diretoria_palavras_noSTOP %>%
filter(DIRETORIA == "OUTROS")
termos_OUTROS = termos_dir_OUTROS %>%
  top_n(n+200, tf_idf)

termos_dir = bind_rows(mutate(termos_DEE, DIRETORIA = "DEE"),
                         mutate(termos_DEA, DIRETORIA = "DEA"),
                         mutate(termos_OUTROS, DIRETORIA = "OUTROS")) %>%
  select(palavra) %>%
  unique()
```

```
gg <- gsub("[:.:]", "_", as.character(termos_dir$palavra))
gg <- gsub("[:,:]", "_", as.character(termos_dir$palavra))
fe <- matrix(data = 0, nrow = length(DB$DESCRI_PEDIDO1), ncol = length(gg))
fe <- data.frame(fe); colnames(fe) <- gg
i=j=0
for(i in 1:length(DB$Protocolo)){
        for(j in 1:length(gg)){
                g <- grepl(gg[j], DB$DESCRI_PEDIDO1[i])
                if(g == TRUE){
                        fe[i, j] <- 1
                }
        }
}
NumTermos = as_tibble(rbind(apply(fe,2,sum)))
NumTermos = gather(NumTermos, key = "termo", value = "Num_Pedidos")
NumTermos = NumTermos[order(NumTermos$Num_Pedidos, decreasing = TRUE), ]
```

```
highchart() %>%
  hc_add_series(data = NumTermos$Num_Pedidos,
                type = "bar",
                name = "# de pedidos",
                showInLegend = FALSE,
                tooltip = list(valueDecimals = 0, valuePrefix = "", valueSuffix = ""), color="blue") %>%
```

```
  hc_yAxis(title = list(text = "Quantitativo de pedidos"),
           allowDecimals = TRUE, max = (max(NumTermos$Num_Pedidos)+103),
           labels = list(format = "{value}")) %>%
  hc_xAxis(title = list(text = "Termo"),
           categories = NumTermos$termo,
           tickmarkPlacement = "on",
           opposite = FALSE) %>%
  hc_title(text = "Quantitativo de pedidos por termo (sem exclusividade)",
           style = list(fontWeight = "bold")) %>%
  hc_subtitle(text = paste("")) %>%
      hc_tooltip(valueDecimals = 2,
                 pointFormat = "{point.y} pedidos")%>%
                 #pointFormat = "Variável: {point.x} <br> Missing: {point.y}")
      hc_credits(enabled = TRUE,
                 text = "Fonte: CGU, e-SIC (2019). Elaboração: Ewerson Pimenta.",
                 style = list(fontSize = "10px")) %>%
  hc_exporting(enabled = TRUE, filename = "F3-filmes-genero-Pimenta")
```

```
db_modelo = as_tibble(cbind(select(DB,DIRETORIA),fe))
```

```
# __Porcentagem de ZEROS por variável__

zeros <- (colSums(fe==0)/nrow(fe)*100); var <- names(fe)
db_zero <- data.frame(var,zeros); rownames(db_zero) <- NULL
db_zero <- db_zero[order(db_zero$zeros, decreasing = TRUE), ]

hc4_1 <- highchart() %>%
  hc_add_series(data = db_zero$zeros,
                type = "bar",
                name = "Porcentagem de zeros",
                showInLegend = FALSE,
                tooltip = list(valueDecimals = 2, valuePrefix = "", valueSuffix = " %"), color="pink") %
  hc_yAxis(title = list(text = "Porcentagem de zero"),
           allowDecimals = TRUE, max = 100,
           labels = list(format = "{value}%")) %>%
  hc_xAxis(categories = db_zero$var,
           tickmarkPlacement = "on",
           opposite = FALSE) %>%
  hc_title(text = "Porcentagem de zeros por variável",
           style = list(fontWeight = "bold")) %>%
  hc_subtitle(text = paste("")) %>%
      hc_tooltip(valueDecimals = 2,
                 pointFormat = "Zeros: {point.y}")%>%
                 #pointFormat = "Variável: {point.x} <br> Missing: {point.y}")
      hc_credits(enabled = TRUE,
                 text = "Fonte: IMDB/KAGGLE. Elaboração: Ewerson Pimenta.",
                 style = list(fontSize = "10px")) %>%
  hc_exporting(enabled = TRUE, filename = "Fig00-Pimenta")
#hc <- hc %>%
#  hc_add_theme(hc_theme_darkunica())
hc4_1; remove(hc4_1, var, zeros)
```

Modelos de classificação

Partição dos dados

Particionando a base de dados em Treino e Teste, esses dois (Treino e Teste) também terão armazenados as diretorias que foram responsaveis por cada pedido via amostragem probabilística dos dados originais separadamente das bases de Treino e Teste.

Para amostragem aleatória simples

```
intrain <- createDataPartition(y = db_modelo$DIRETORIA, p = 0.65, list = FALSE)
training <- db_modelo[intrain,]
testing <- db_modelo[-intrain,]
```

Modelagem 1 - Random Forest (RF)

Random Forest (RF) - Metodologia

Descrição 1. Random Forest foi desenvolvido para agregar árvores de decisão (modelo de classificação);
2. Pode ser usado para modelo de classificação (p/ var. resposta categórica) ou regressão (no caso de haver variável resposta contínua);
3. Evita *overfitting*;
4. Permite trabalhar com um largo número de características de um conjunto de dados;
5. Auxilia na seleção de variáveis baseada em um algoritmo que calcula a importância por variável (assim, tendo conhecimento de quais variáveis são mais importantes, podemos usar essa informação para outros modelos de classificação);
6. User-friendly: apenas 2 parâmetros livres:

- Trees - ntrees, default 500 (Nº de árvores);
- Variáveis selecionadas via amostragem aleatória candidatas à cada "split" (quebra da árvore) - mtry, default $\sqrt{p}$ p/ classificação e $\frac{p}{3}$ p/ regressão (p: nº de features/variáveis);

Passo-a-Passo

É realizado em 3 passos:

1. Desenha as amostras via bootstrap do número de árvores *ntrees*;

2. Para cada amostra via bootstrap, cresce o número de árvores "un-puned" para a escolha da melhor quebra da árvore baseado na amostra aleatória do valor predito de mtry a cada nó da árvore;

- 3. Faz classificação de novos valores usando a maioria de votos p/ classificação e usa a média p/ regressão baseada nas amostras de ntrees.

Random Forest - Aplicação e Resultados

Inicialmente utilizaremos o pacote `randomForest` que implmenta o algoritmo de Random Forest de Breiman (baseado na clusterização de Breiman, originalmente codificada em Fortran) que tem por finalidade classificar e/ou criar regressão. Além disso, pode ser usado em um modelo não supervisionado para avaliar proximidades entre pontos.

Estamos usando, a partir daqui, a base de treino.

```
#library(randomForest)
#library(rpart)
#library(rpart.plot)
#rf <- randomForest(proximity = T,ntree = 38,do.trace = T,WR~.,data=training)
```

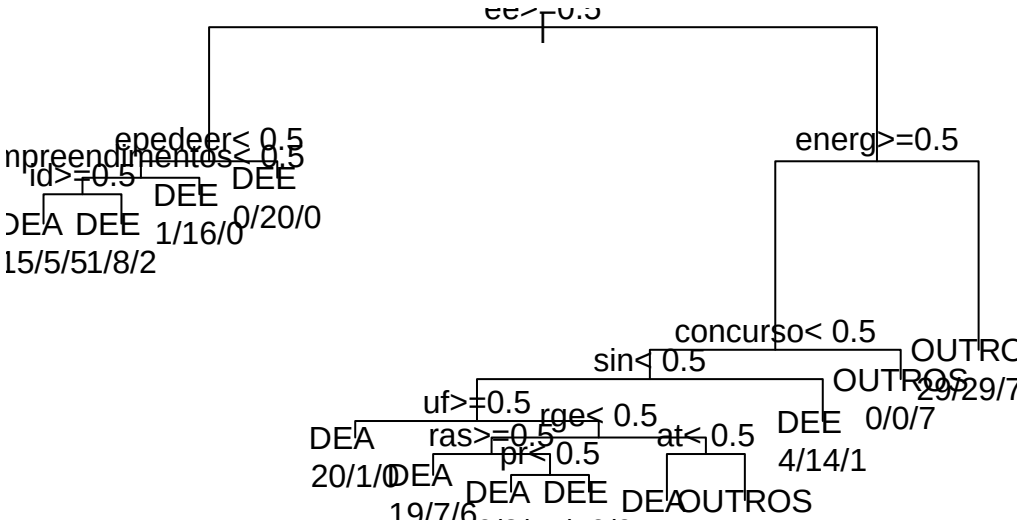
```r
set.seed(9984512)
# Training with classification tree
rf <- rpart(DIRETORIA ~ ., data=training, method="class", xval = 4, )
print(rf, digits = 3)
```

```
## n= 407
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 407 267 DEA (0.3440 0.3366 0.3194)
##     2) ee>=0.5 73  24 DEE (0.2329 0.6712 0.0959)
##       4) epedeer< 0.5 53  24 DEE (0.3208 0.5472 0.1321)
##         8) empreendimentos< 0.5 36  20 DEA (0.4444 0.3611 0.1944)
##          16) id>=0.5 25  10 DEA (0.6000 0.2000 0.2000) *
##          17) id< 0.5 11   3 DEE (0.0909 0.7273 0.1818) *
##         9) empreendimentos>=0.5 17   1 DEE (0.0588 0.9412 0.0000) *
##       5) epedeer>=0.5 20   0 DEE (0.0000 1.0000 0.0000) *
##     3) ee< 0.5 334 211 DEA (0.3683 0.2635 0.3683)
##       6) energ>=0.5 202 108 DEA (0.4653 0.2921 0.2426)
##        12) concurso< 0.5 195 101 DEA (0.4821 0.3026 0.2154)
##          24) sin< 0.5 176  86 DEA (0.5114 0.2557 0.2330)
##            48) uf>=0.5 21   1 DEA (0.9524 0.0476 0.0000) *
##            49) uf< 0.5 155  85 DEA (0.4516 0.2839 0.2645)
##              98) rge< 0.5 64  32 DEA (0.5000 0.3906 0.1094)
##               196) ras>=0.5 32  13 DEA (0.5938 0.2188 0.1875) *
##               197) ras< 0.5 32  14 DEE (0.4062 0.5625 0.0312)
##                 394) pr< 0.5 9   3 DEA (0.6667 0.2222 0.1111) *
##                 395) pr>=0.5 23   7 DEE (0.3043 0.6957 0.0000) *
##              99) rge>=0.5 91  53 DEA (0.4176 0.2088 0.3736)
##               198) at< 0.5 47  21 DEA (0.5532 0.2128 0.2340) *
##               199) at>=0.5 44  21 OUTROS (0.2727 0.2045 0.5227) *
##          25) sin>=0.5 19   5 DEE (0.2105 0.7368 0.0526) *
##        13) concurso>=0.5 7   0 OUTROS (0.0000 0.0000 1.0000) *
##       7) energ< 0.5 132  58 OUTROS (0.2197 0.2197 0.5606) *
```

```r
attributes(rf)
```

```
## $names
##  [1] "frame"               "where"               "call"
##  [4] "terms"               "cptable"             "method"
##  [7] "parms"               "control"             "functions"
## [10] "numresp"             "splits"              "variable.importance"
## [13] "y"                   "ordered"
##
## $xlevels
## named list()
##
## $ylevels
## [1] "DEA"    "DEE"    "OUTROS"
##
## $class
## [1] "rpart"
```

```r
plot(rf)
text(rf, use.n = TRUE)
```




```r
# Predict the testing set with the trained model
predictions <- predict(rf, testing, type = "class")

# Accuracy and other metrics
confusionMatrix(predictions, as.factor(testing$DIRETORIA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction DEA DEE OUTROS
##     DEA     27  18     14
##     DEE     15  31     10
##     OUTROS  32  24     46
##
## Overall Statistics
##
##                Accuracy : 0.4793
##                  95% CI : (0.4112, 0.5479)
##     No Information Rate : 0.341
##     P-Value [Acc > NIR] : 1.819e-05
##
##                   Kappa : 0.2214
##  Mcnemar's Test P-Value : 0.004465
##
## Statistics by Class:
##
##                      Class: DEA Class: DEE Class: OUTROS
## Sensitivity              0.3649     0.4247        0.6571
## Specificity              0.7762     0.8264        0.6190
## Pos Pred Value           0.4576     0.5536        0.4510
## Neg Pred Value           0.7025     0.7391        0.7913
## Prevalence               0.3410     0.3364        0.3226
## Detection Rate           0.1244     0.1429        0.2120
## Detection Prevalence     0.2719     0.2581        0.4700
```

```
## Balanced Accuracy        0.5705     0.6255        0.6381
```

Olhando as 6 primeiras observações real X predito

```
p1 <- predict(rf,training)
head(p1)
```

```
##         DEA        DEE     OUTROS
## 1 0.2196970 0.21969697 0.56060606
## 2 0.9523810 0.04761905 0.00000000
## 3 0.5937500 0.21875000 0.18750000
## 4 0.2727273 0.20454545 0.52272727
## 5 0.2105263 0.73684211 0.05263158
## 6 0.6000000 0.20000000 0.20000000
```

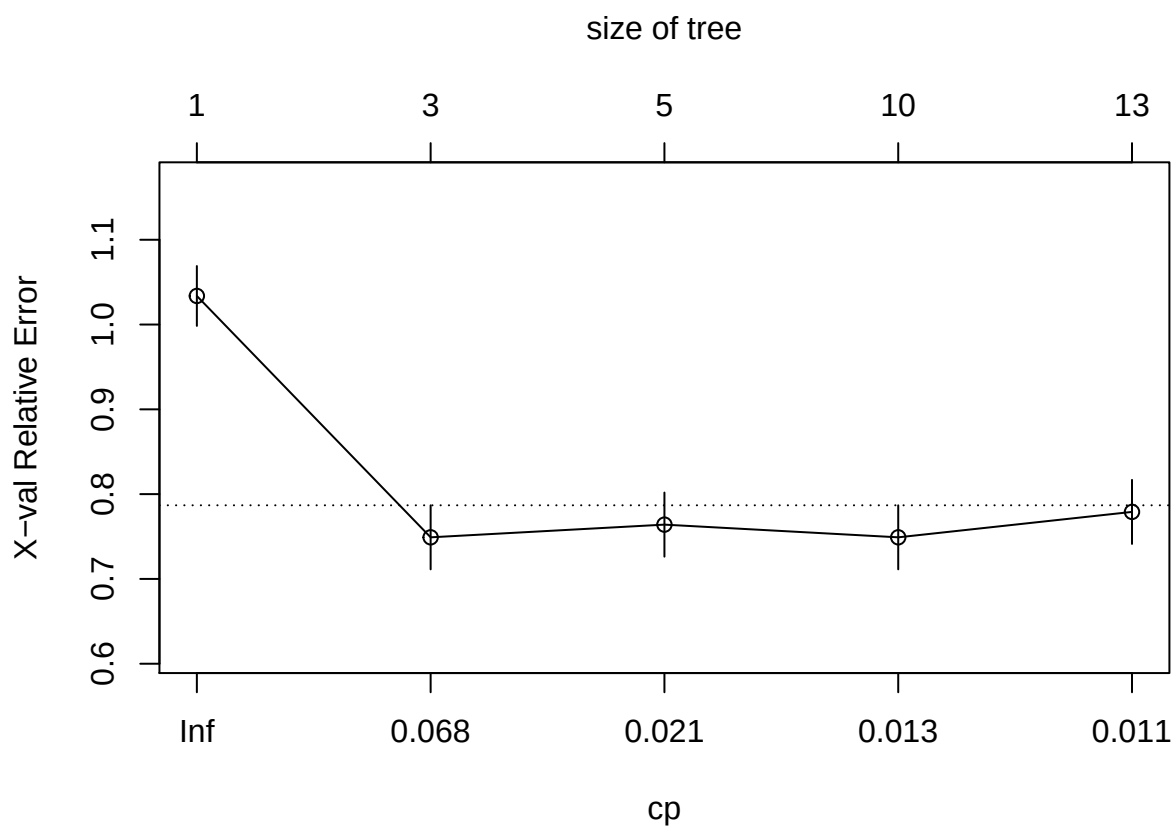
```
head(training$DIRETORIA)
```

```
## [1] "OUTROS" "DEA"    "OUTROS" "DEA"    "DEA"    "DEA"
```

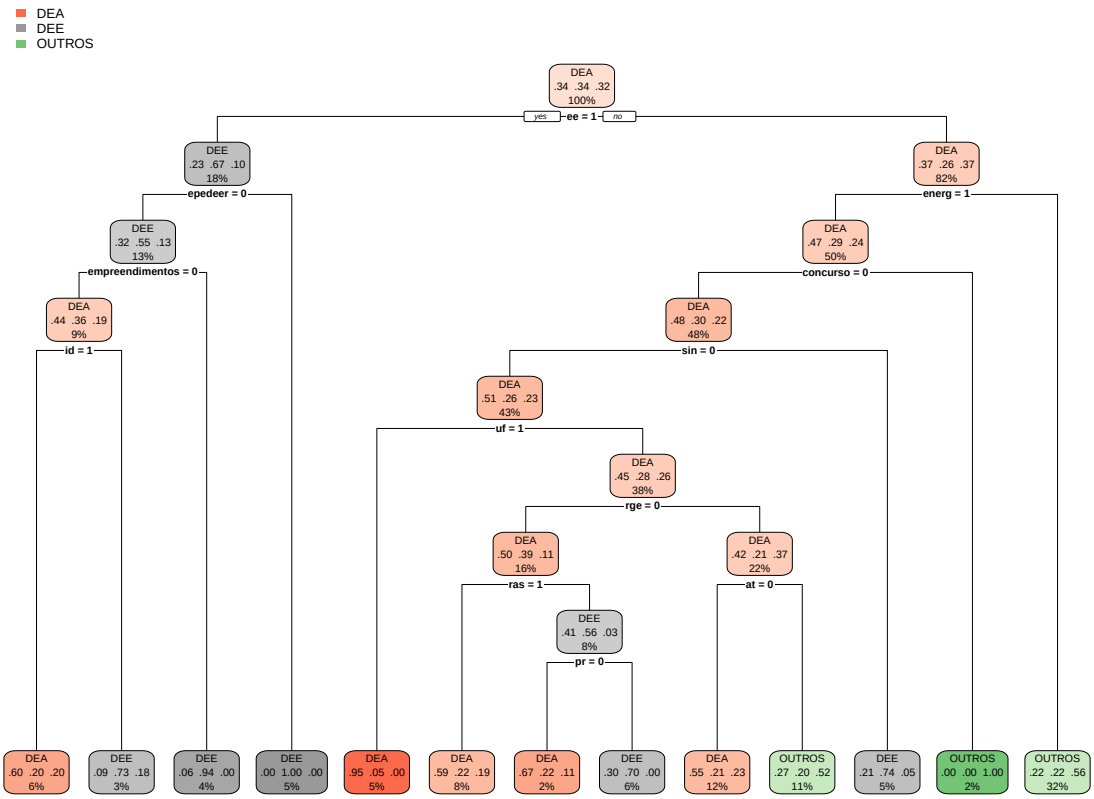
Selecionando uma árvore

```
rp <- rpart::rpart(formula = DIRETORIA~.,data=training)
```

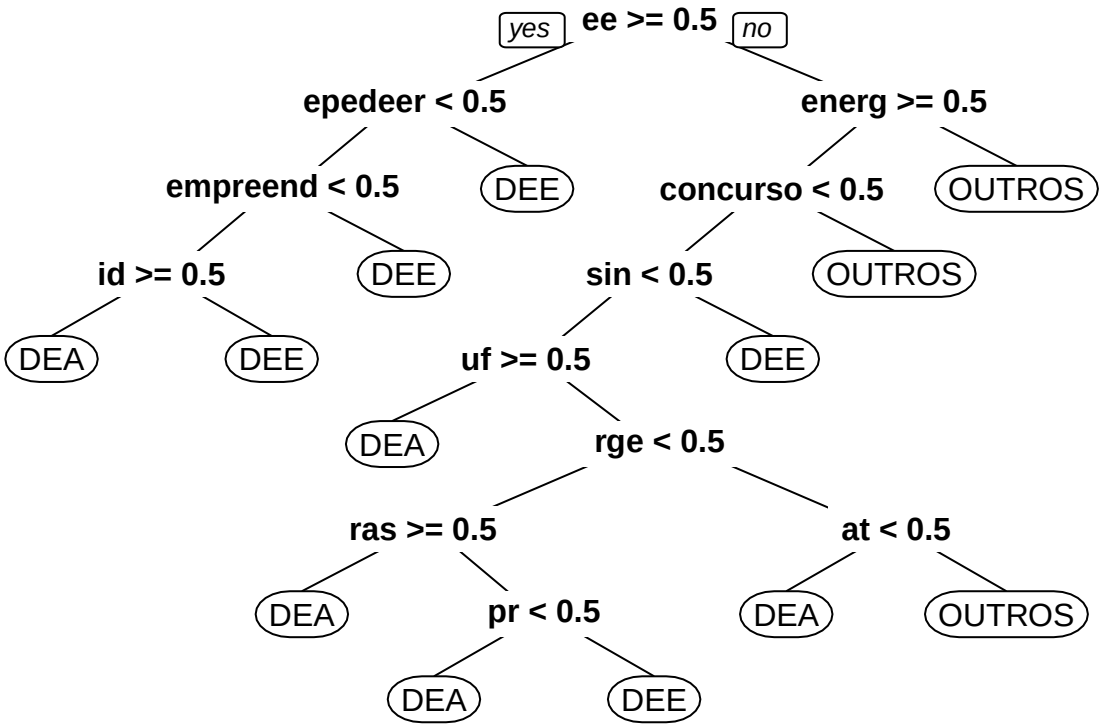
```
rpart::plotcp(rf)
```




```
rpart.plot(rf)
```




```
rpart.plot.version1(rf)
```



Outra forma de escrever o modelo é usando a função `randomForest( )`

```
db_modelo = db_modelo %>% select(-`pad's`)
intrain <- createDataPartition(y = db_modelo$DIRETORIA, p = 0.65, list = FALSE)
```

```r
training <- db_modelo[intrain,]
testing <- db_modelo[-intrain,]

set.seed(09986755)
rf1 <- randomForest(as.factor(DIRETORIA) ~ ., data=training,
                    importance = TRUE,
                    proximity = TRUE)
rf1
```

```
##
## Call:
##  randomForest(formula = as.factor(DIRETORIA) ~ ., data = training,      importance = TRUE, proximity
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 45
##
##         OOB estimate of  error rate: 36.36%
## Confusion matrix:
##        DEA DEE OUTROS class.error
## DEA    101  21     18   0.2785714
## DEE     50  80      7   0.4160584
## OUTROS  39  13     78   0.4000000
```

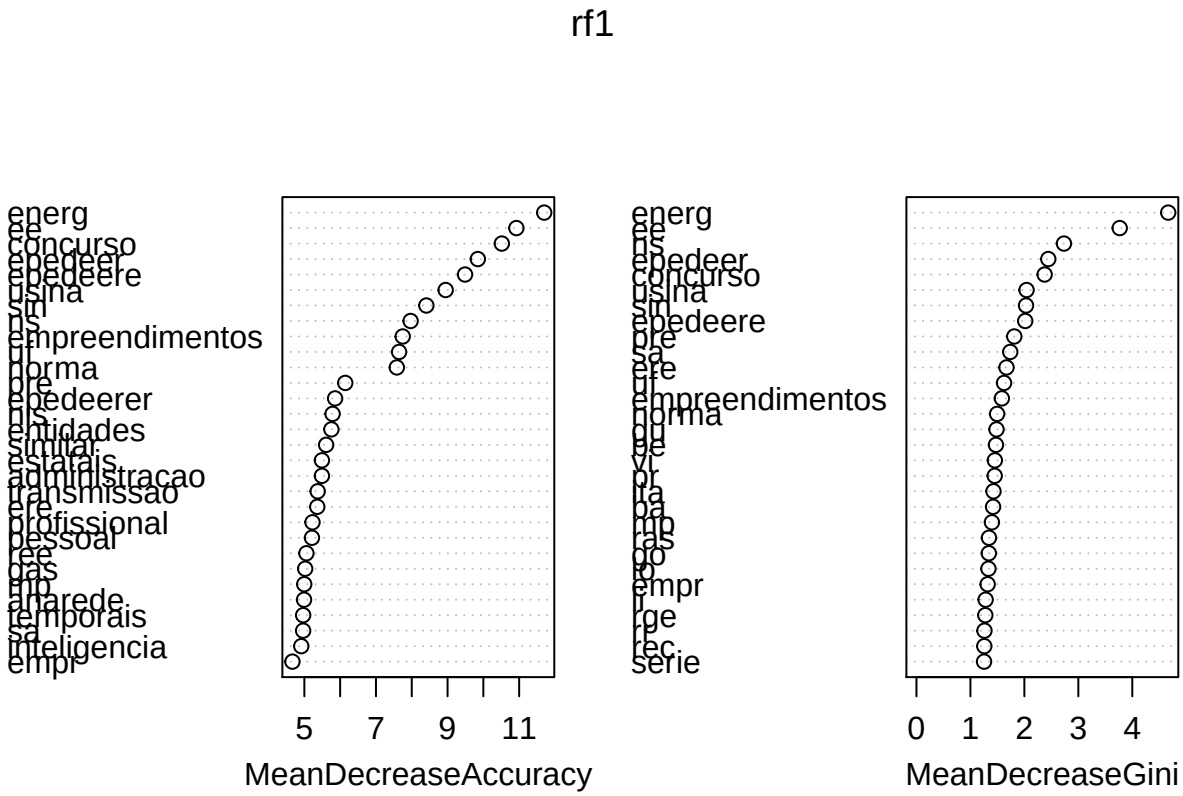
```r
# Predict the testing set with the trained model
predictions1 <- predict(rf1, testing, type = "class")

# Accuracy and other metrics
confusionMatrix(predictions1, as.factor(testing$DIRETORIA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction DEA DEE OUTROS
##     DEA     50  23     19
##     DEE     11  42     12
##     OUTROS  13   8     39
##
## Overall Statistics
##
##                Accuracy : 0.6037
##                  95% CI : (0.5352, 0.6692)
##     No Information Rate : 0.341
##     P-Value [Acc > NIR] : 2.619e-15
##
##                   Kappa : 0.4045
##  Mcnemar's Test P-Value : 0.1041
##
## Statistics by Class:
##
##                      Class: DEA Class: DEE Class: OUTROS
## Sensitivity              0.6757     0.5753        0.5571
## Specificity              0.7063     0.8403        0.8571
## Pos Pred Value           0.5435     0.6462        0.6500
## Neg Pred Value           0.8080     0.7961        0.8025
## Prevalence               0.3410     0.3364        0.3226
```

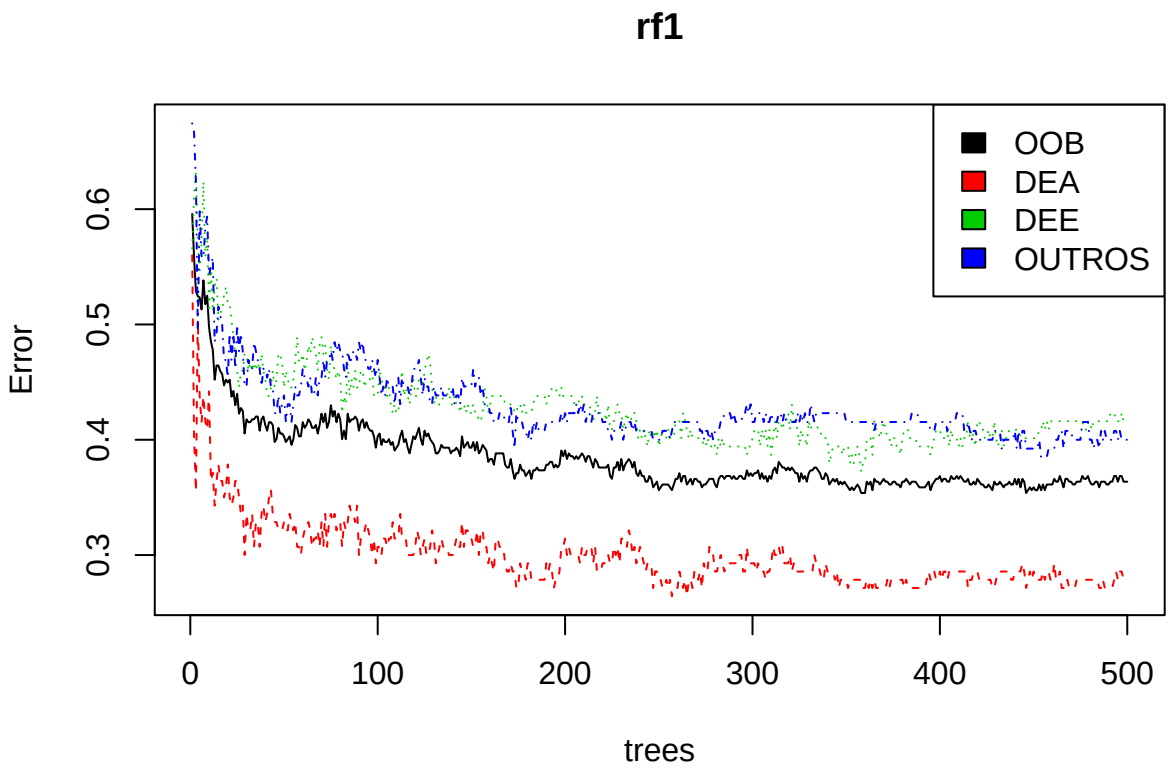
```
## Detection Rate           0.2304     0.1935        0.1797
## Detection Prevalence     0.4240     0.2995        0.2765
## Balanced Accuracy        0.6910     0.7078        0.7071
```

Importância de variáveis

```
RF_importance = randomForest::importance(rf1)[order(randomForest::importance(rf1)[,1], decreasing = TRU
randomForest::varImpPlot(rf1)
```




```
plot(rf1)
legend('topright', colnames(rf1$err.rate), col=1:5, fill=1:5)
```



A partir de $n = 300$ árvores a taxa do erro **OOB (Out of Bag)** tende a estabilizar.

Tuning do modelo

Fixando, então, $n = 300$ árvores

Aparentemente $mtry = 45$ parece ser um bom palpite para o segundo parâmetro do random forest, uma vez que esse retornou menor taxa de erro **OOB**, $37,35\%$. Entretanto esse erro ainda é muito alto. Vamos reescrever o modelo com os parâmetros tunados.

```
set.seed(09986755)
rf2 <- randomForest(as.factor(DIRETORIA) ~ ., data=training,
                    ntree = 300,
                    mtry = 45,
                    importance = TRUE,
                    proximity = TRUE)
rf2
```

```
##
## Call:
##  randomForest(formula = as.factor(DIRETORIA) ~ ., data = training,      ntree = 300, mtry = 45, impo
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 45
##
##         OOB estimate of  error rate: 36.86%
## Confusion matrix:
##        DEA DEE OUTROS class.error
## DEA     99  22     19   0.2928571
## DEE     48  82      7   0.4014599
## OUTROS  38  16     76   0.4153846
```

```r
# Predict the testing set with the trained model
predictions2 <- predict(rf2, testing, type = "class")

# Accuracy and other metrics
confusionMatrix(predictions2, as.factor(testing$DIRETORIA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction DEA DEE OUTROS
##     DEA     51  23     16
##     DEE     11  42     14
##     OUTROS  12   8     40
##
## Overall Statistics
##
##                Accuracy : 0.6129
##                  95% CI : (0.5446, 0.6781)
##     No Information Rate : 0.341
##     P-Value [Acc > NIR] : 2.866e-16
##
##                   Kappa : 0.4183
##  Mcnemar's Test P-Value : 0.09193
##
## Statistics by Class:
##
##                      Class: DEA Class: DEE Class: OUTROS
## Sensitivity              0.6892     0.5753        0.5714
## Specificity              0.7273     0.8264        0.8639
## Pos Pred Value           0.5667     0.6269        0.6667
## Neg Pred Value           0.8189     0.7933        0.8089
## Prevalence               0.3410     0.3364        0.3226
## Detection Rate           0.2350     0.1935        0.1843
## Detection Prevalence     0.4147     0.3088        0.2765
## Balanced Accuracy        0.7082     0.7009        0.7177
```

```r
p2 <- predict(rf2,training)
as.character(head(p2))
```

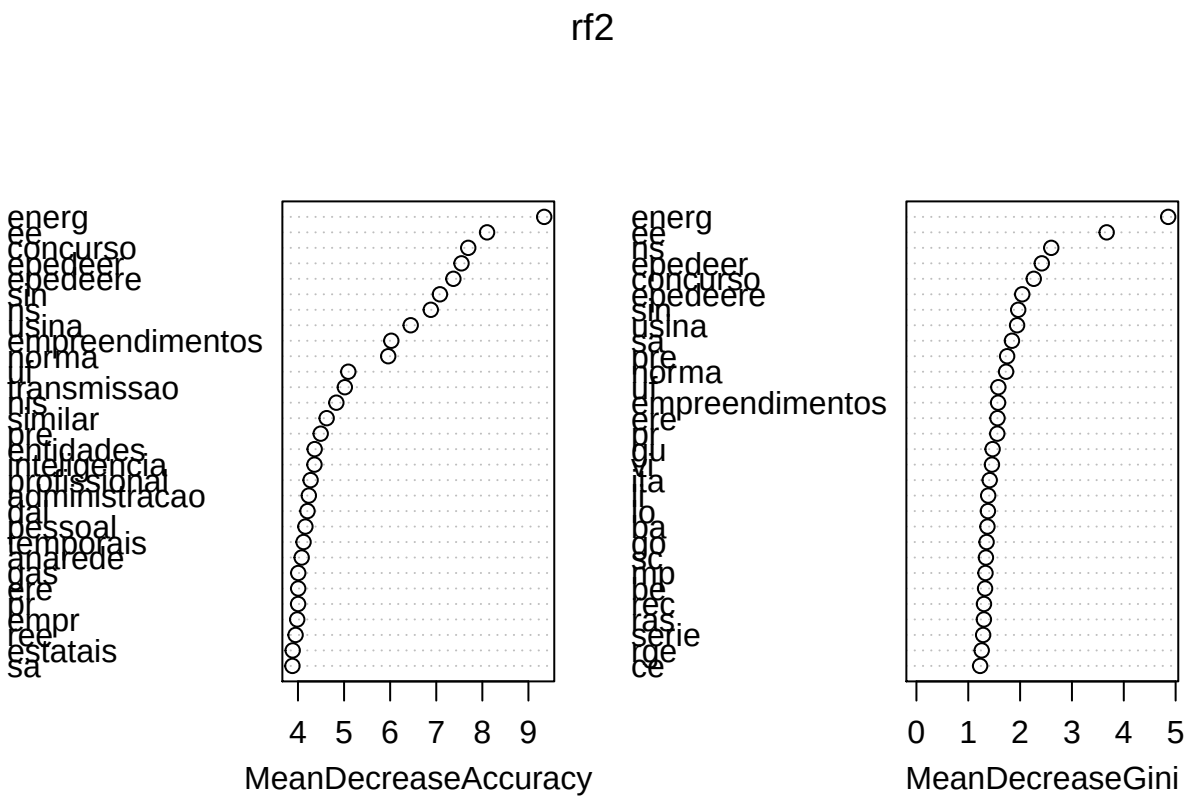
```
## [1] "OUTROS" "DEA"    "DEE"    "DEA"    "DEA"    "DEA"
```

```r
head(training$DIRETORIA)
```

```
## [1] "OUTROS" "DEA"    "DEE"    "DEA"    "OUTROS" "DEA"
```

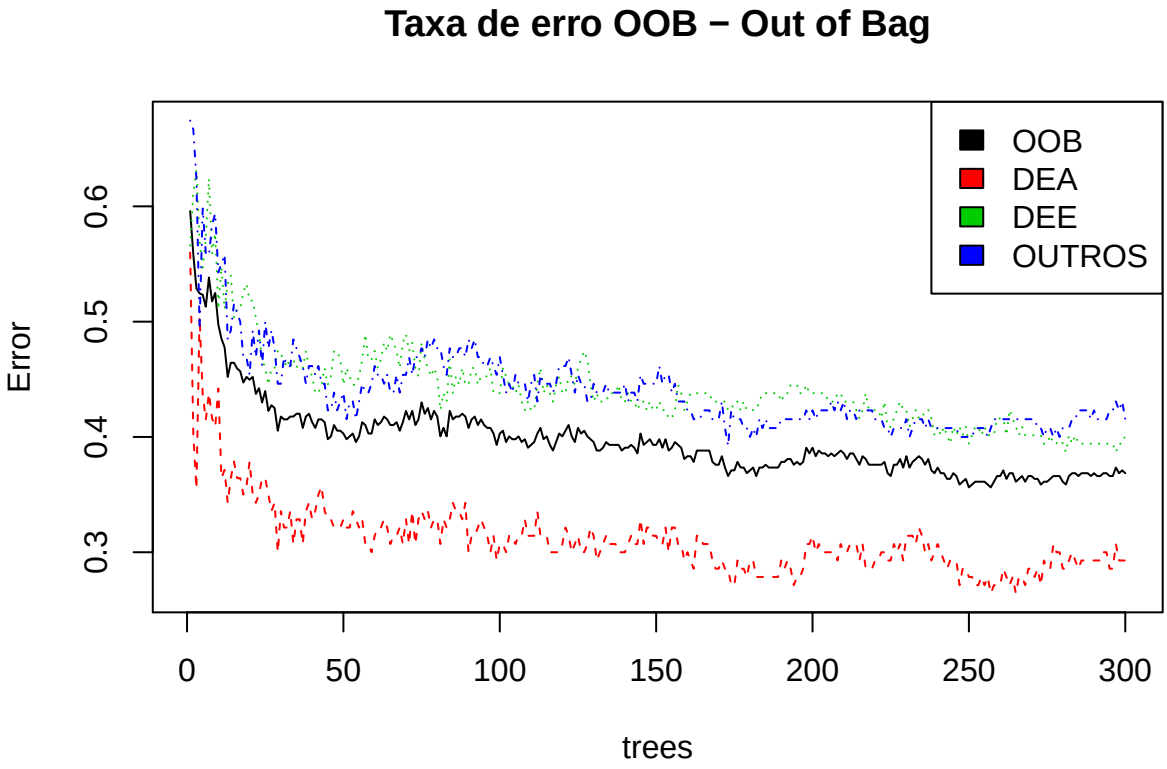
Acurácia de aproximadamente 66% na base de teste. E as taxas de erro de classificação foram 25%, 44% e 45% para *DEA*, *DEE* e *OUTROS*, respectivamente. Houve um melhor desempenho na classificação do modelo para a categoria *DEA*

```r
RF_importance = randomForest::importance(rf2)[order(randomForest::importance(rf2)[,1], decreasing = TRU
randomForest::varImpPlot(rf2)
```



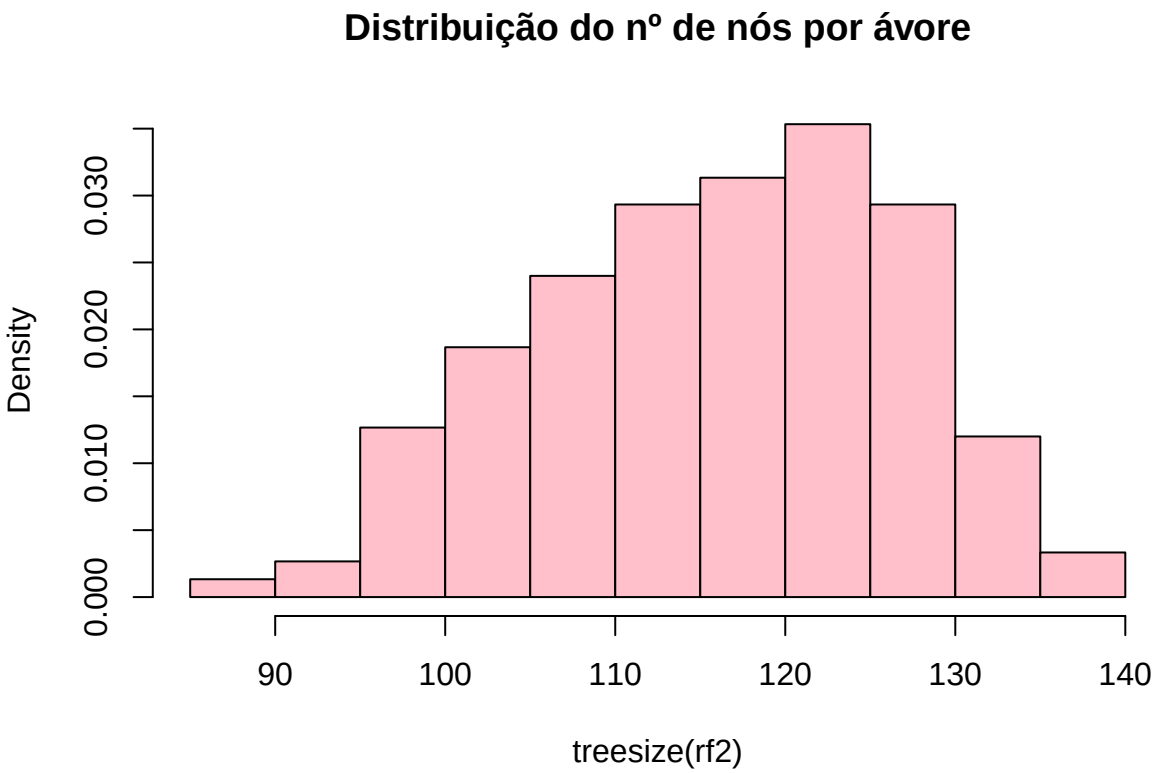
Taxa de Erro Random Forest

```
plot(rf2, main = "Taxa de erro OOB - Out of Bag")
legend('topright', colnames(rf2$err.rate), col=1:5, fill=1:5)
```



Histograma do Número de nós por árvore

```
hist(treesize(rf2), probability = T,
     main = "Distribuição do nº de nós por ávore",
     col = "pink")
```



Vamos excluir as variáveis que não retornaram valor de importância para o algoritmo do random forest.

```
RF_importance = randomForest::importance(rf2)[order(randomForest::importance(rf2)[,1], decreasing = TRU
```

```
RF1 = data.frame(variables = rownames(RF_importance), importance = RF_importance[,4])
RF1 = RF1[order(RF1$importance, decreasing = TRUE),]
rownames(RF1) <- NULL
#summary(RF1)
```

```
RF2 = RF1[1:20,]
#library("highcharter")
hc6_1 <- highchart() %>%
  hc_add_series(data = RF2$importance,
                type = "bar",
                name = "Importância",
                showInLegend = FALSE,
                tooltip = list(valueDecimals = 2, valuePrefix = "", valueSuffix = "")) %>%
  hc_yAxis(title = list(text = "Importância"),
           allowDecimals = TRUE, max = 12,
           labels = list(format = "{value}")) %>%
  hc_xAxis(title = list(text = "Fatores"),
           categories = RF2$variables,
           tickmarkPlacement = "on",
           opposite = FALSE) %>%
  hc_title(text = "Importância por fator - Random Forest",
```

```
           style = list(fontWeight = "bold")) %>%
  hc_subtitle(text = paste("")) %>%
      hc_tooltip(valueDecimals = 2,
                 pointFormat = "Importância: {point.y}")%>%
                 #pointFormat = "Variável: {point.x} <br> Importância: {point.y}")
      hc_credits(enabled = TRUE,
                 text = "Fonte: CGU, e-SIC. Elaboração: Leal, Alize; Pimenta, Ewerson.",
                 style = list(fontSize = "10px")) %>%
  hc_exporting(enabled = TRUE, filename = "F6_1-importance-Pimenta")
#hc <- hc %>%
#  hc_add_theme(hc_theme_darkunica())
hc6_1
```

Vamos excluir todas as variáveis que retornaram importância menor ou igual a zero.

```
variaveis_sem_importancia = RF1 %>% filter(as.character(importance) <= 0)
#summary(variaveis_sem_importancia)
variaveis_sem_importancia = as.character(variaveis_sem_importancia$variables)
```

```
training1 = training %>% select(-(variaveis_sem_importancia))
testing1 = testing %>% select(-(variaveis_sem_importancia))
db_modelo1 = db_modelo %>% select(-(variaveis_sem_importancia))
```

```
set.seed(756446)
rf3 <- randomForest(as.factor(DIRETORIA) ~ ., data=db_modelo1,
                    ntree = 300,
                    mtry = 45,
                    importance = TRUE,
                    proximity = TRUE)
rf3
```

```
##
## Call:
##  randomForest(formula = as.factor(DIRETORIA) ~ ., data = db_modelo1,      ntree = 300, mtry = 45, imp
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 45
##
##         OOB estimate of  error rate: 36.06%
## Confusion matrix:
##        DEA DEE OUTROS class.error
## DEA    130  44     40   0.3925234
## DEE     47 138     25   0.3428571
## OUTROS  39  30    131   0.3450000
```

```
# Predict the testing set with the trained model
predictions3 <- predict(rf3, testing1, type = "class")

# Accuracy and other metrics
confusionMatrix(predictions3, as.factor(testing1$DIRETORIA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction DEA DEE OUTROS
##     DEA     73   0      0
```

```
##     DEE      0  73      0
##     OUTROS   1   0     70
##
## Overall Statistics
##
##                Accuracy : 0.9954
##                  95% CI : (0.9746, 0.9999)
##     No Information Rate : 0.341
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9931
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: DEA Class: DEE Class: OUTROS
## Sensitivity              0.9865     1.0000        1.0000
## Specificity              1.0000     1.0000        0.9932
## Pos Pred Value           1.0000     1.0000        0.9859
## Neg Pred Value           0.9931     1.0000        1.0000
## Prevalence               0.3410     0.3364        0.3226
## Detection Rate           0.3364     0.3364        0.3226
## Detection Prevalence     0.3364     0.3364        0.3272
## Balanced Accuracy        0.9932     1.0000        0.9966
```

```r
p3 <- predict(rf3,training1)
as.character(head(p3))
```

```
## [1] "OUTROS" "DEA"    "DEE"    "DEA"    "OUTROS" "DEA"
```

```r
head(training1$DIRETORIA)
```

```
## [1] "OUTROS" "DEA"    "DEE"    "DEA"    "OUTROS" "DEA"
```

```r
set.seed(756446)
rf4 <- randomForest(as.factor(DIRETORIA) ~ ., data=db_modelo,
                    ntree = 300,
                    mtry = 45,
                    importance = TRUE,
                    proximity = TRUE)
rf4
```

```
##
## Call:
##  randomForest(formula = as.factor(DIRETORIA) ~ ., data = db_modelo,      ntree = 300, mtry = 45, impo
##                Type of random forest: classification
##                      Number of trees: 300
## No. of variables tried at each split: 45
##
##         OOB estimate of  error rate: 33.81%
## Confusion matrix:
##        DEA DEE OUTROS class.error
## DEA    143  42     29   0.3317757
## DEE     52 141     17   0.3285714
## OUTROS  50  21    129   0.3550000
```

```r
# Predict the testing set with the trained model
predictions4 <- predict(rf4, testing, type = "class")

# Accuracy and other metrics
confusionMatrix(predictions4, as.factor(testing$DIRETORIA))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction DEA DEE OUTROS
##     DEA     71   0      2
##     DEE      2  73      0
##     OUTROS   1   0     68
##
## Overall Statistics
##
##                Accuracy : 0.977
##                  95% CI : (0.9471, 0.9925)
##     No Information Rate : 0.341
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9654
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: DEA Class: DEE Class: OUTROS
## Sensitivity              0.9595     1.0000        0.9714
## Specificity              0.9860     0.9861        0.9932
## Pos Pred Value           0.9726     0.9733        0.9855
## Neg Pred Value           0.9792     1.0000        0.9865
## Prevalence               0.3410     0.3364        0.3226
## Detection Rate           0.3272     0.3364        0.3134
## Detection Prevalence     0.3364     0.3456        0.3180
## Balanced Accuracy        0.9727     0.9931        0.9823
```

```r
p4 <- predict(rf4,training)
as.character(head(p4))
```

```
## [1] "OUTROS" "DEA"    "DEE"    "DEA"    "OUTROS" "DEA"
```

```r
head(training$DIRETORIA)
```

```
## [1] "OUTROS" "DEA"    "DEE"    "DEA"    "OUTROS" "DEA"
```

Comparacao do poder de predicao dos modelos treinados e propostos

```r
as.character(p1[1:15])
```

```
##  [1] "0.21969696969697"   "0.952380952380952"  "0.59375"
##  [4] "0.272727272727273"  "0.210526315789474"  "0.6"
##  [7] "0.553191489361702"  "0.210526315789474"  "0.210526315789474"
## [10] "0.21969696969697"   "0.304347826086957"  "0.0588235294117647"
## [13] "0"                  "0.553191489361702"  "0"
```

```r
as.character(rf$y[1:15])
```

```
##  [1] "3" "1" "3" "1" "1" "1" "2" "1" "2" "1" "1" "1" "2" "1" "2"
```

```r
as.character(p2[1:15])
```

```
##  [1] "OUTROS" "DEA"    "DEE"    "DEA"    "DEA"    "DEA"    "DEE"
##  [8] "OUTROS" "DEA"    "DEA"    "DEE"    "DEA"    "DEE"    "DEA"
## [15] "OUTROS"
```

```r
as.character(p4[1:15])
```

```
##  [1] "OUTROS" "DEA"    "DEE"    "DEA"    "OUTROS" "DEA"    "DEE"
##  [8] "OUTROS" "DEA"    "DEA"    "DEE"    "DEA"    "DEE"    "DEA"
## [15] "OUTROS"
```

```r
training$DIRETORIA[1:15]
```

```
##  [1] "OUTROS" "DEA"    "DEE"    "DEA"    "OUTROS" "DEA"    "DEE"
##  [8] "OUTROS" "DEA"    "DEE"    "DEE"    "DEA"    "DEE"    "DEA"
## [15] "OUTROS"
```

```r
edit(MDSplot)
fig.align="center"
(MDIM_treino = MDSplot(rf4, training$DIRETORIA, pch=20))
(MDIM_teste = MDSplot(rf4, testing$DIRETORIA, pch=20))
sum(MDIM_treino$eig[1:2])
sum(MDIM_teste$eig[1:2])
```