Estatística é com R!

Como comprar um carro usado com o R

Publicado em novembro 19, 2018 por uffadmin

By Prof. Dr. Steven Dutt Ross

Introdução

Estou começando o processo de compra de um carro usado. Como costumo tomar essas decisões fazendo uma pesquisa sobre o assunto, escrevi um código R para raspar dados de *websites* e construir um banco de dados. Acredito que esse processo pode ser replicado para comprar passagens aéreas, apartamentos, e outros elementos

que envolvam a coleta de dados para reduzir a assimetria de informações. Acredito que saber o preço de outros carros com características semelhantes reduz essa assimetria.

A ideia deste texto surge a partir de diversos cursos e palestras sobre o R que fiz em 2018. Um ótimo exemplo disso é o minicurso realizado pela Karla Esquerre e Adelmo Filho sobre *Web scraping* no III Seminário Internacional de Estatística com R. Também recomendo o minicurso do CURSO-R sobre Web scraping.

Seria a Ciência de Dados um nome bonito para a Estatística?

Existe muita discussão sobre ciência de dados ser apenas um nome bonito para a estatística. Seria a mesma coisa com um outro nome? Recentemente assisti uma palestra da Jenny Bryan sobre o assunto e fiquei convencido que **Não**. Entre os argumentos colocados, Jenny Bryan coloca uma lista (incompleta) de ações que um cientista de dados consegue fazer e um estatístico não. Tive que concordar com ela. Sendo um estatístico, frequentemente vi a minha inabilidade para:

1. Raspar dados da *web*,
2. Solicitar dados por meio de uma API,
3. Analisar dados em formato de lista,
4. Entender formatos como o JSON ou GeoJSON,
5. Utilizar dados no formato XML,
6. Ler um simples banco de dados do formato SQL, entre outras coisas são comuns para um cientista de dados.

Como graduado em Estatística, a minha maior preocupação sempre foi o modelo, os seus pressupostos e o procedimento para o teste de hipóteses.

Mas isso não quer dizer que não podemos aprender um pouco de ciência de dados. Esse artigo é justamente para compartilhar o que aprendi sobre raspagem de dados, porque acredito que este é um ótimo exemplo de como alguém pode usar o R de uma forma funcional para a tomada de decisões simples (como comprar um carro usado). Ainda estou no processo de aprendizado. Assim, quero deixar claro que...

fonte: Jkust

Parte 1: Identificando a estrutura do site

Para fazer a escolha do carro que vou comprar estou buscando vou raspar dados do site do vivalocal com o pacote

Rvest. Ao fazer um levantamento da estrutura do site, percebi que o site tem uma estrutura do tipo:

SITE BÁSICO / NÚMERO DA PAGINA / COMPLEMENTO DO SITE

Onde: SITE BÁSICO = http://search.vivalocal.com/auto-veiculo-usado/rio-de-janeiro/t+

COMPLEMENTO DO SITE = ?lb=new&search=1&start_field=1&keywords=.....

Exemplos da estrutura:

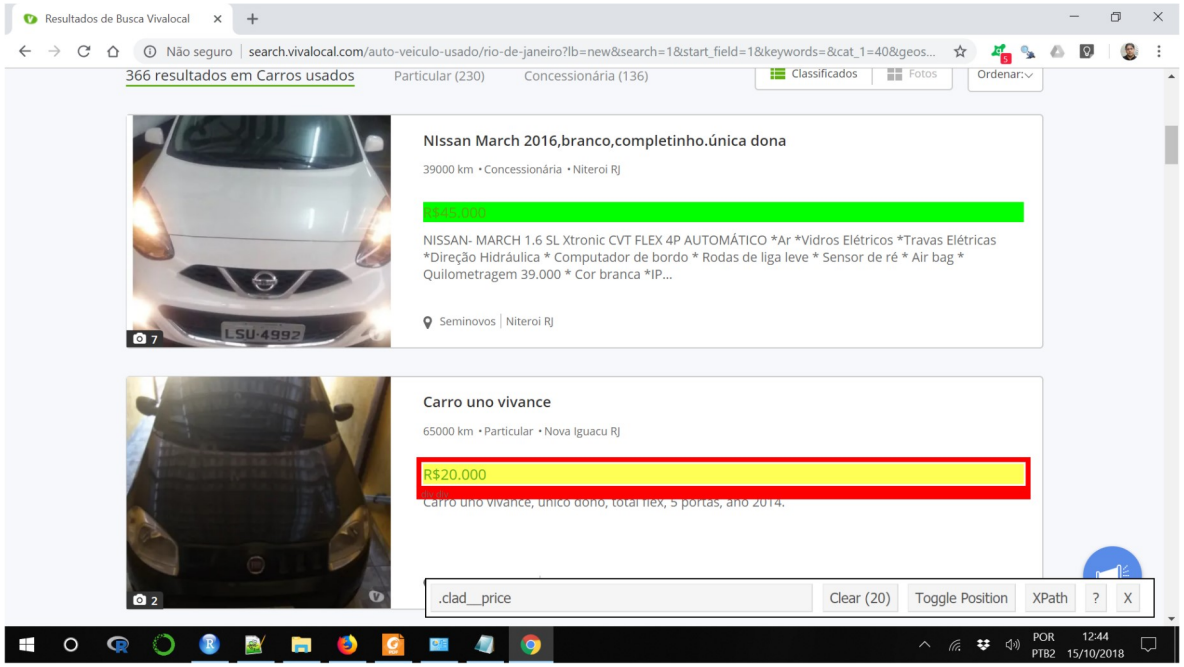
http://search.vivalocal.com/auto-veiculo-usado/rio-de-janeiro/t+2?lb=new&search=1&start_field=1&keywords=&cat_1=40&geosearch_text=Rio+de+Janeiro+-+RJ&searchGeoId=74&sp_common_price%5Bstart%5D=&sp_common_price%5Bend%5D=&sp_vehicules_mileage%5Bstart%5D=&sp_vehicules_mileage%5Bend%5D=&sp_common_year%5Bstart%5D=&sp_common_year%5Bend%5D=&sp_vehicules_energy=

http://search.vivalocal.com/auto-veiculo-usado/rio-de-janeiro/t+19?lb=new&search=1&start_field=1&keywords=&cat_1=40&geosearch_text=Rio+de+Janeiro+-+RJ&searchGeoId=74&sp_common_price%5Bstart%5D=&sp_common_price%5Bend%5D=&sp_vehicules_mileage%5Bstart%5D=&sp_vehicules_mileage%5Bend%5D=&sp_common_year%5Bstart%5D=&sp_common_year%5Bend%5D=&sp_vehicules_energy=

Identificar essa estrutura será uma das etapas mais importantes no processo de raspagem de dados.

Também descobri que o site tem 19 páginas.

Com a ferramenta [selectorgadget](), pude identificar os elementos do site que estou interessado. Um exemplo do uso do selectorgadget pode ser vista na figura abaixo:



Os elementos que estou interessado para construir o banco de dados são:

h4
.clad__price
.vip-enabled
.clad__summary , .not-tablet-mobile

Com a identificação dos elementos de que preciso, posso iniciar o processo de raspagem de dados com o R.

Parte 2: Raspagem de dados

A raspagem de dados pode ser realizada com o pacote *rvest*. O objeto **nome_carro** receberá um vetor com o nome de todos os carros da primeira página do site. O mesmo acontecerá com os objetos **preco_carro**,**km_local_carro** e **texto_carro**. O objetivo aqui é criar um vetor para cada elemento.

```
library(rvest)
```

```
## Loading required package: xml2
```

```
site_carro <- read_html('http://search.vivalocal.com/auto-veiculo-usado/rio-de-janeiro?lb=new&search=1&start_field=1&keywords=&c
```

```
nome_carro <- site_carro %>%
  html_nodes("h4") %>%
  html_text()
########################################
preco_carro <- site_carro %>%
  html_nodes(".clad__price") %>%
  html_text()
########################################
km_local_carro <- site_carro %>%
```

```
  html_nodes(".vip-enabled") %>%
  html_text()
########################################
texto_carro <- site_carro %>%
  html_nodes(".clad__summary , .not-tablet-mobile") %>%
  html_text()
```

Vamos dar uma olhada no que conseguimos. Com o comando `head` podemos ver as primeiras linhas do vetor.

```
head(nome_carro)
```

```
## [1] "Jac 2- Jac motors- seminovo"
## [2] "Fiat Uno Vivace 1.0 8V (Flex) 4p 2013"
## [3] "Chevrolet Onix 1.0 LT SPE4"
## [4] "Proteção Veicular"
## [5] "Venda"
## [6] "ONIX - 2014 1.4 MPFI LT 8V FLEX 4P MANUAL"
```

```
head(preco_carro)
```

```
## [1] "R$19.000"  "R$14.000"  "R$18.000"  "R$9.000"   "R$500.000" "R$35.900"
```

```
head(km_local_carro)
```

```
## [1] "50 km Particular Volta Redonda RJ "
## [2] "62000 km Particular Niteroi RJ "
## [3] "80000 km Particular Niteroi RJ "
## [4] "1010101 km Particular Rio de Janeiro RJ "
## [5] "100 km Particular Angra dos Reis RJ "
## [6] "61000 km Particular Petropolis RJ "
```

```
head(texto_carro,10)
```

```
##  [1] "Inserir anúncio"
##  [2] "\n            Procurar\n        "
##  [3] "\n                                                        Particular  (226)
##  [4] "\n                                                        Concessionária  (143)
##  [5] "\n            \n                Ordenar: \n                     Últimos\n                         \n
##  [6] "\n                \n                    \n                    Classificados\n                    \n                    I
##  [7] " 192945467  Jac 2- Jac motors- seminovo  50 km Particular Volta Redonda RJ   R$19.000 Jacmotors- J2 1.4 semi novo compl
##  [8] "Jacmotors- J2 1.4 semi novo completo, sem dividas documento em dia. contato zap( 21)969475739 / 24 992762557"
##  [9] " 192922832  Fiat Uno Vivace 1.0 8V (Flex) 4p 2013  62000 km Particular Niteroi RJ   R$14.000 Fiat Uno Vivace 1.0 8V (Fl
## [10] "Fiat Uno Vivace 1.0 8V (Flex) 4p 2013 Ano 2013 62.000 Km Cor Preto Câmbio manual 4 Portas Flex, Final da placa 5 distril
```

Podemos observar que conseguimos todos os dados que queriamos, mas as informações do objeto `texto_carro` tem

dois problemas: Primeiro, as seis primeiras linhas do banco de dados não têm informação sobre os carros. Segundo, as linhas parecem replicadas. Isto é, cada registro está sendo contado duas vezes. Podemos ver isso utilizando o comando `length`.

```
length(nome_carro)
```

```
## [1] 20
```

```
length(preco_carro)
```

```
## [1] 20
```

```
length(km_local_carro)
```

```
## [1] 20
```

```
length(texto_carro)
```

```
## [1] 46
```

Corrigindo o objeto “texto_carro”. Encontrei a solução para isso em uma pergunta do stackoverflow.com. Admito que não é uma solução elegante, mas resolveu o problema.

```
# Resolvendo o problema 1
  texto_carro <- texto_carro[-c(1,2,3,4,5,6)]
# Resolvendo o problema 2
  to_Delete <- seq(1, length(texto_carro), 2)
  texto_carro <-  texto_carro[-to_Delete]
  length(texto_carro)
```

```
## [1] 20
```

Como os dois problemas corrigidos, podemos montar uma base de dados com o comando `data.frame()`. Este comando será utilizado para “juntar” tudo em um único banco de dados.

```
carro_vivalocal<- data.frame(nome_carro,preco_carro,km_local_carro,texto_carro,stringsAsFactors = FALSE)
```

Como, até agora, a raspagem está funcionando para a primeira página, podemos tentar desenvolver uma função para ir da página 01 até a página 19. O objetivo é criar uma função que colete todos os dados da página 01, passe para a página 02, colete todos os dados dessa página, passe para a próxima página e assim por diante até a página 19. A função `Sys.sleep()` é utilizada para desacelerar a rotina da **função for**. Pelo que pude entender, esta função é importante porque ao fazer muitos pedidos ao site do viva local podemos sobrecarregar o servidor.

```
n_paginas <- 19
banco_carro_i <- c()
banco_carro   <- c()


for (i in 0:n_paginas) {
  url_number <- 19 - i
  # Buscando a página
  url <- paste0('http://search.vivalocal.com/auto-veiculo-usado/rio-de-janeiro/t+',url_number,'?lb=new&search=1&start_field=1&ke

    # lendo a página
  page <- read_html(url)
  names_i <- page %>%   html_nodes("h4") %>% html_text()
  names_j <- page %>%   html_nodes(".clad__price") %>% html_text()
  names_k <- page %>%   html_nodes(".vip-enabled") %>% html_text()
  names_l <- page %>%   html_nodes(".clad__summary , .not-tablet-mobile") %>% html_text()
  # Resolvendo o Problema 1
  names_l <- names_l[-c(1,2,3,4,5,6)]
```

```
  # Resolvendo o Problema 2
  to_Delete <- seq(1, length(names_l), 2)
  names_l <-  names_l[-to_Delete]

  # Construindo o banco de dados de cada etapa
  banco_carro_i<- data.frame(names_i,names_j,names_k,names_l,stringsAsFactors = FALSE)
  # Alimentando o banco de dados total
  banco_carro<-rbind(banco_carro, banco_carro_i)
  # Mostra a página em que o R está
  print(i)
  # Para suspender a execução do R
  Sys.sleep(3)
}
```

Vamos ver a estrutura do banco de dados. O comando `str()` exibe a estrutura interna de um objeto R (é uma ótima alternativa ao `summary()`).

```
str(banco_carro)
```

```
## 'data.frame':    165 obs. of  8 variables:
##  $ names_i: chr  "Logus AP1.8 DocumentoOk ReciboAberto Zap(21)98403-1482" "Mercedes Benz Classe C 180 19971998 Classic" "Miniv
##  $ names_j: chr  "R$3.800" "R$24.000" "R$18.000" "R$70.000" ...
##  $ names_k: chr  "127000 km Particular Rio de Janeiro RJ " "187200 km Particular Campos dos Goytacazes RJ " "158000 km Concess
##  $ names_l: chr  "vendo logus 94 motor AP 1.8 nunca levo gas ! E meu a 5 anos no meu nome e assim consta ! manutenção em dia t
##  $ preco  : num  3800 24000 18000 70000 16500 7500 15000 30000 9900 39900 ...
##  $ Km     : chr  "127000" "187200" "158000" "95000" ...
```

```
##  $ Tipo   : chr  "Particular" "Particular" "Concessionária" "Concessionária" ...
##  $ cidade : chr  "Rio de Janeiro RJ " "Campos dos Goytacazes RJ " "Rio de Janeiro RJ " "Rio de Janeiro RJ " ...
```

Parte 3: Transformação de dados

Já estamos com o banco de dados, mas ainda precisamos transformar essa estrutura para uma forma mais analítica. Já vi diversos termos para essa fase. Os mais comuns são limpeza de dados, higienização de dados, e transformação de variáveis.

A primeira etapa é transformar o preço do carro em número. Nesse momento, a variável está no formato texto (*caracter*).

```
# Tirando o ponto
banco_carro$preco <- gsub('\\.', '', banco_carro$names_j)
# Tirando o R$
banco_carro$preco <- chartr("R$","  ", banco_carro$preco)
# Tirando o espaço
banco_carro$preco <- gsub('\\s+', '', banco_carro$preco)
# Transformando em número
banco_carro$preco<-as.numeric(banco_carro$preco)
```

Temos uma variável com três informações: Kilometragem, Tipo, e cidade/local. A segunda etapa é dividir as colunas da variável. Isto é, quero dividir a variável em 03 outras variáveis. A primeira contendo a kilometragem. A segunda com as classes “Particular/Concessionária. A última com o local do carro. (Esse segmento ficou um pouco

repetitivo. Preciso de uma programação um pouco mais funcional).

Repare que para dividir a primeira coluna, podemos usar Km como um marcador. Por exemplo,

```
######################################################################
# Km
######################################################################
mini_banco_carro<-banco_carro$names_k
mini_banco_carro<-data.frame(mini_banco_carro)
colnames(mini_banco_carro)<-'nome'

library(tidyr)
mini_banco_carro<-separate(data = mini_banco_carro, col = nome, into = c("Km", "resto"), sep = "\\ km ")

# Juntando as novas variaveis ao banco
library(dplyr)
banco_carro$chave<-rownames(banco_carro)
mini_banco_carro$chave<-rownames(mini_banco_carro)
banco_carro<-full_join(banco_carro, mini_banco_carro)
remove(mini_banco_carro)

######################################################################
# Tipo
######################################################################
mini_banco_carro<-banco_carro$resto
mini_banco_carro<-data.frame(mini_banco_carro)
colnames(mini_banco_carro)<-'nome'
mini_banco_carro<-separate(data = mini_banco_carro, col = nome, into = c("Tipo", "resto2"), sep = "\\s+")
# Juntando as novas variaveis ao banco
```

```
banco_carro$chave<-rownames(banco_carro)
mini_banco_carro$chave<-rownames(mini_banco_carro)
banco_carro<-full_join(banco_carro, mini_banco_carro,by = "chave")
remove(mini_banco_carro)

######################################################################
# Cidade
######################################################################

banco_carro$cidade <- gsub('\\Particular ', '', banco_carro$resto)
banco_carro$cidade <- gsub('\\Concessionária ', '', banco_carro$cidade)

######################################################################
# Retirando os excessos
######################################################################
nomes<-c("names_i", "names_j", "names_k", "names_l","preco","Km","Tipo","cidade")
banco_carro<-banco_carro[,nomes]
```

Parte 4: Análise de texto

Agora precisamos de uma ferramenta para a análise de texto para avaliar o conteúdo de cada anúncio. Vamos utilizar o pacote chamado (tidytext)[https://cran.r-project.org/web/packages/tidytext/index.html]

A primeira etapa para a análise de texto é colocar no formato *tidy*. O formato *tidy* é um formato especifico onde:

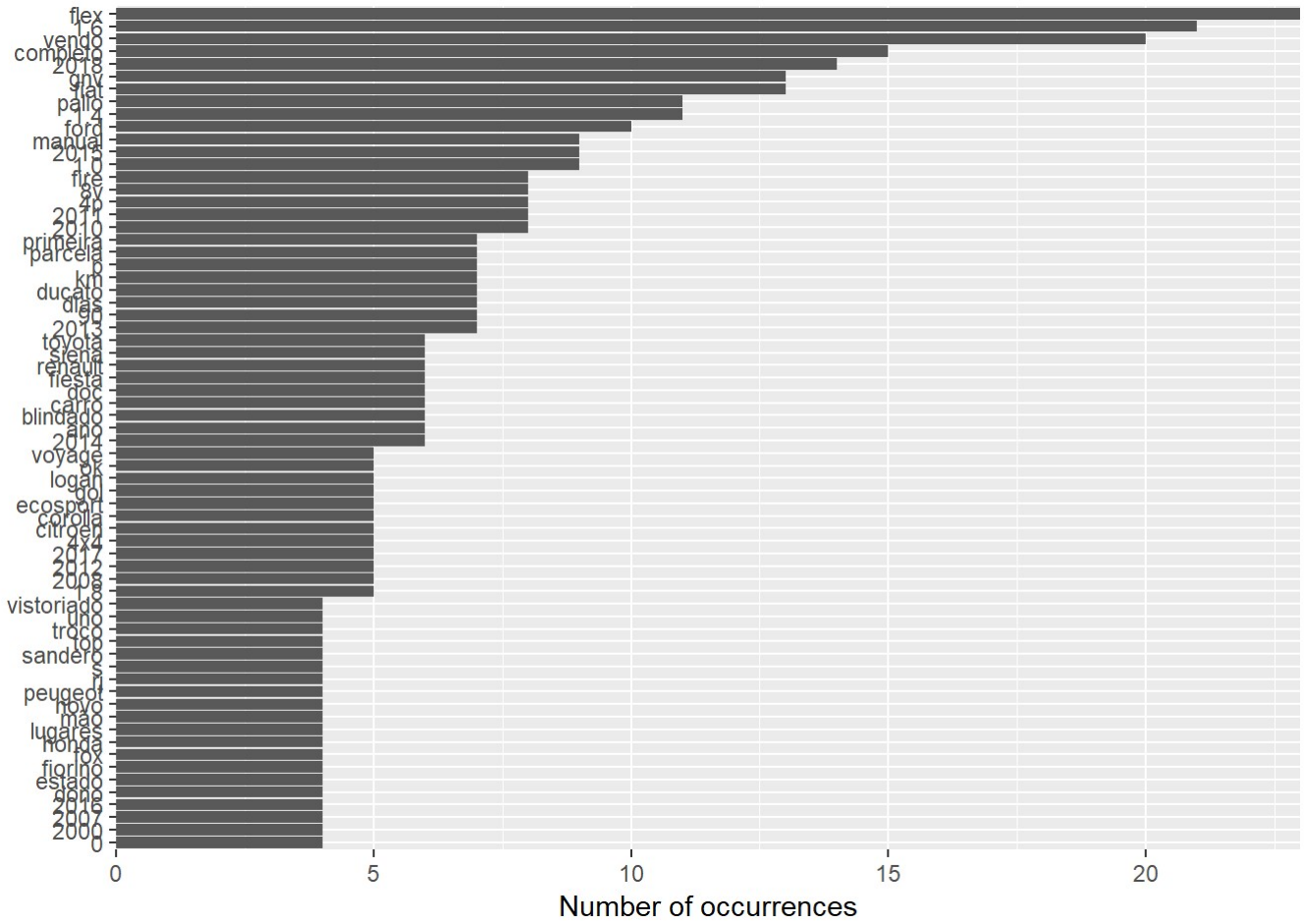
1. Cada coluna é uma variável,
2. Cada linha é uma observação, e

3. Cada célula é um valor.

Depois de colocar no formato *tidy*, vamos criar um gráfico com as palavras mais repetidas dos anúncios. Nesse gráfico, podemos ver que temos várias palavras que não são interessantes para a nossa análise. Por exemplo: “vendo”,“completo”, “primeira”,“parcela”,“dias”, etc. Precisamos remover essas palavras para gerar uma visualização de dados.

```
library(dplyr)
texto1<-banco_carro$names_i
texto1<-tbl_df(texto1)
colnames(texto1)<-'text'

library(tidytext)
tidy_texto1 <- texto1 %>%
    mutate(line = row_number()) %>%
    unnest_tokens(word, text)
library(forcats)
library(ggplot2)
tidy_texto1 %>%
    anti_join(get_stopwords(language = "pt")) %>%
    count(word, sort = TRUE) %>%
    top_n(50) %>%
    ggplot(aes(fct_reorder(word, n), n)) +
    geom_col() +
    coord_flip() +
    scale_y_continuous(expand = c(0,0)) +
    labs(x = NULL, y = "Number of occurrences")
```



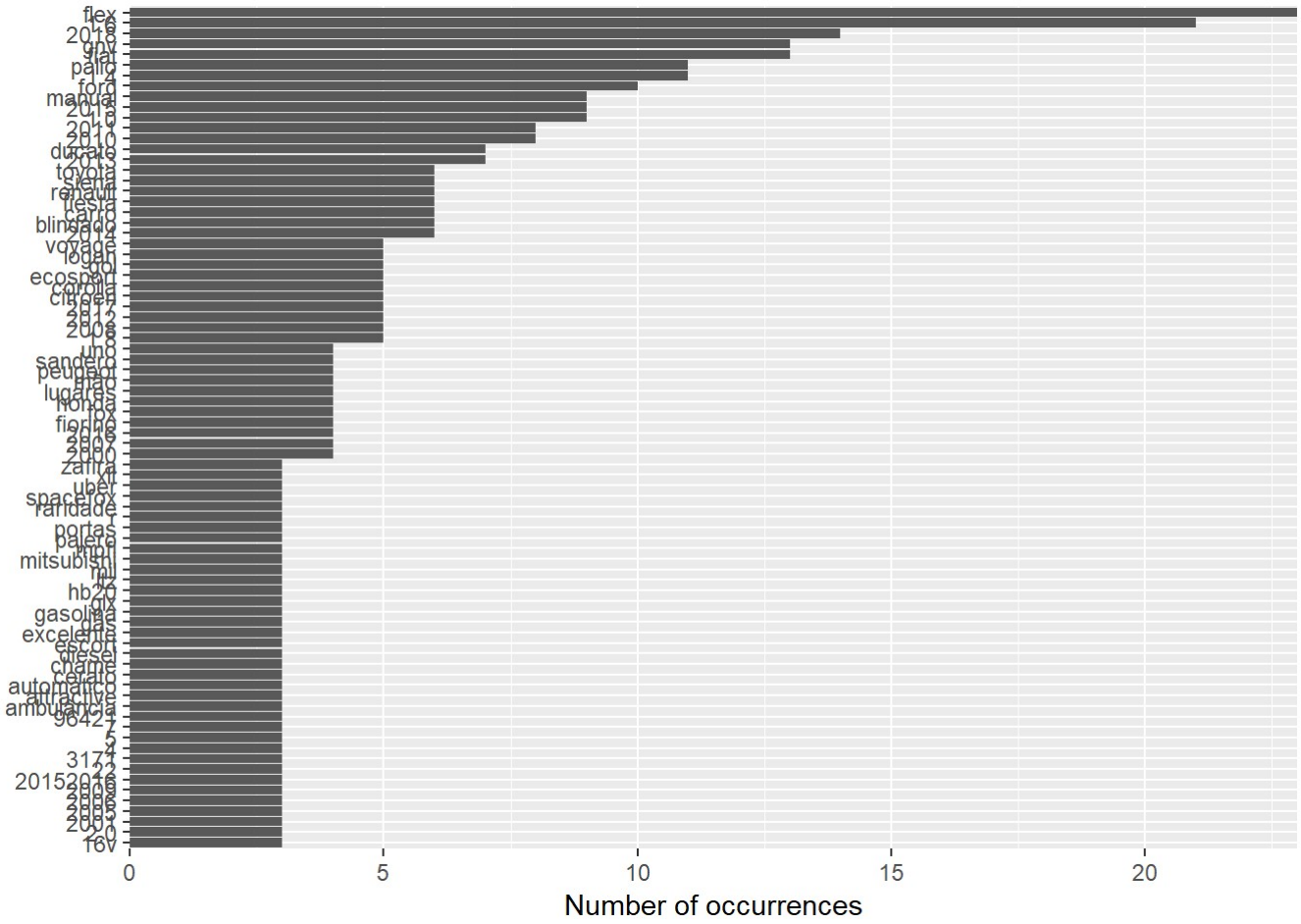

```
# tirando as stopwords
pt_stop<-get_stopwords(language = "pt")

palavras_extras<- data.frame(word = c("vendo","completo","fire","8v","4p","primeira","parcela","p","km","dias","90","doc","ano",'

# juntando as minhas palavras com a lista de stopwords
```

```
palavra_onibus<-pt_stop %>% bind_rows(palavras_extras)

tidy_texto1 %>%
    anti_join(palavra_onibus) %>%
    count(word, sort = TRUE) %>%
    top_n(50) %>%
    ggplot(aes(fct_reorder(word, n), n)) +
    geom_col() +
    coord_flip() +
    scale_y_continuous(expand = c(0,0)) +
    labs(x = NULL, y = "Number of occurrences")
```



Flex é uma palavra interessante para a decisão de comprar um carro. Podemos ter como um dos objetivos comparar os anúncios de carros Flex. Para fazer uma análise do preço do carro Flex, precisamos incluir uma variável binária (zero = carro não-Flex e um = carro Flex). Desse modo, temos:

```
# diversas maneiras de escrever a palavra
```

```
Flex<-c("flex","Flex","FLEX")
banco_carro$Flex <- as.integer(grepl(paste(Flex,collapse="|"), banco_carro$names_i))
```

Vamos olhar como ficou resultado com o pacote *DT*. O *DT* pode ser utilizado para apresentar o *data.frame* que conseguimos. Apresentar o banco de dados é cada vez mais importante na pesquisa reprodutível. Como consequencia da Reprodutibilidade científica, os critérios do SciELO buscam visam ampliar a transparência na pesquisa. Acredito que, em breve, todas as revistas devem pedir o banco de dados (e o código) junto com o artigo para publicação. Para quem quiser saber mais, recomendo o livro Reproducible Research with R and R Studio.

```
library(DT)
datatable(banco_carro, options = list(pageLength = 5))
```

A partir deste momento será possível muitas análises. Por exemplo, utilizar o pacote *lexiconPT* do Sillas Gonzaga para fazer uma análise no texto. Podemos também fazer uma comparação do Chevrolet Corsa com o Fiat Palio. Esse é um bom momento para construir alguns objetivos de pesquisa. Depois disso, podemos passar para uma análise estatística tradicional. Vou ficar por aqui.

Referências:

1. Allaire,JJ; Xie, Yihui; McPherson,Jonathan; Luraschi, Javier;Ushey, Kevin;Atkins, Aron;Wickham, Hadley;Cheng, Joe;Chang, Winston *rmarkdown: Dynamic Documents for R*, 2018 Disponível aqui
2. CURSO R: **Minicurso de Webscraping**. Disponível aqui

3. Esquerre,Karla e Filho,Adelmo **Minicurso de Webscraping com pacote Rvest no III SER**. Disponível aqui
4. Gandrud, C. Reproducible Research with R and RStudio. CRC Press, 2014.
5. Gonzaga, S. *lexiconPT: Lexicons for Portuguese Text Analysis* R package version 0.1.0 https://CRAN.R-project.org/package=lexiconPT
6. Knoepfle, D. **How to buy a used car with R** Disponível aqui
7. Seminário de Estatística com o R – SER. Disponível aqui. Acesso em 07 de novembro de 2018.
8. Wickham, H. rvest: Easily Harvest (Scrape) Web Pages. R package version 0.3.2. https://CRAN.R-project.org/package=rvest

Share this:

Compartilhar

Relacionado

WordCloud no R - SER
julho 2, 2016
Em "Pacotes"

Conheça o R-Fiddle
setembro 18, 2015
Em "Ferramentas"

Instalação do R e do RStudio
julho 8, 2015
Post similar

Comments

2 comments

2 comentários Classificar por **Mais antigos**

Adicione um comentário...

José Maria S. Freitas

Estou começando agora com o R. E este post foi extremamente esclarecedor. muito bom mesmo! obrigado!

Curtir · Responder · 19 sem

Marcos Monteiro

Parabéns ótima matéria, estou iniciando em R , e vai me ajudar bastante.

Curtir · Responder · 4 sem

Plugin de comentários do Facebook

Esta entrada foi publicada em Uncategorized e marcada com a tag comprar carro, rstats, rvest, web scraping. Adicione o link permanente aos seus favoritos.

Estatística é com R!

Orgulhosamente criado com WordPress.