Bioinformática: Análisis de texto de artículos de PUBMED usando R

Dr. Juan Emmanuel Martinez Ledesma

Búsqueda de artículos de PUBMED por medio R

Primero hay que instalar y cargar el paquete "RISmed". También definimos la opción para no se lean los strings como factores.

```
options(stringsAsFactors = F)
library(RISmed)
```

Creamos un query para buscar artículos de PUBMED de manera similar a como se hace en la página. Acuérdense que se pueden usar operaciones lógicas como *AND* y *OR*.

```
query_colon <- "\"colon\"[TIAB] AND \"cancer\"[TIAB] AND \"young\"[TIAB] AND
                (\"mutation\"[TIAB] OR \"alteration\"[TIAB] OR \"treatment\"[TIAB]
                OR \"hereditary\"[TIAB])"
```

Con el query buscamos cáncer de colon en jóvenes junto con alguna de las siguientes palabras: mutación, alteración, tratamiento o hereditario. El query lo hacemos así porque buscamos mutaciones y alteraciones, tratamientos y porque en algunos casos, el cáncer de colon es hereditario.

De acuerdo al paquete RISMED, debemos de crear el query de la siguiente manera:

```
search_query <- EUtilsSummary(query_colon)
```

Podemos mostrar un resumen de la búsqueda con la siguiente instrucción:

```
summary(search_query)
```

```
## Query:
## "colon"[TIAB] AND "cancer"[TIAB] AND "young"[TIAB] AND ("mutation"[TIAB] OR "alteration"[TIAB] OR "t
##
## Result count:  331
```

Si hacemos el query en PUBMED, la Figura 1 muestra que obtenemos la misma cantidad de artículos.

De acuerdo al paquete RISMED, ejecutamos el query para obtener los datos de PUBMED. Después, usamos algunas instrucciones de RISMED para obtener un data frame con el título, abstract y PUBMED ID de los artículos.

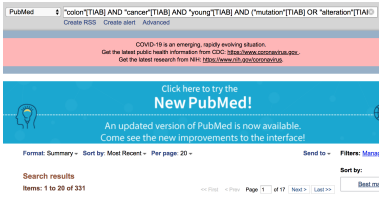


Figure 1: Resultados de PUBMED

```
records <- EUtilsGet(search_query)
pubmed_data <-data.frame('Title'=ArticleTitle(records), 'Abstract'=AbstractText(records),
                         'PID'=ArticleId(records))
pubmed_data[1:3,c("Title","PID")]
```

```
##                                                                        Title
## 1                                                Colorectal cancer statistics, 2020.
## 2                                      Colon Cancer: A Clinician's Perspective in 2019.
## 3 Effect of vitamin B17 on experimentally induced colon cancer in adult male albino rat.
##        PID
## 1 32133645
## 2 32095167
## 3 32073131
```

Hacemos algo de pre-procesamiento para quitar algunos caracteres (. : , ; [ ]) en el título como en el abstract.

```
pubmed_data$Title <- gsub(pattern="\\.|:|,|;|\\[|\\]", replacement="", pubmed_data$Title)
pubmed_data$Abstract <- gsub(pattern="\\.|:|,|;|\\[|\\]", replacement="", pubmed_data$Abstract)
```

Luego pasarlo todo a minúsculas.

```
pubmed_data$Title <- tolower(pubmed_data$Title)
pubmed_data$Abstract <- tolower(pubmed_data$Abstract)
pubmed_data[1,]
```

```
##                              Title
## 1 colorectal cancer statistics 2020
##
## 1 colorectal cancer (crc) is the second most common cause of cancer death in the united states every
##        PID
## 1 32133645
```

Podemos usar la función strsplit y unlist para obtener las palabras contenidas en el abstract.

```
unlist(strsplit(pubmed_data$Abstract[1], " "))[1:10]
```

```
##  [1] "colorectal" "cancer"     "(crc)"      "is"         "the"
##  [6] "second"     "most"       "common"     "cause"      "of"
```

Como podemos usar las palabras de los abstracts para conocer las palabras más frecuentes, debemos tener cuidado para descartar los artículos con abstracts vacíos.

```
which(pubmed_data$Abstract == "")
```

```
##  [1]  11  18  31  35  37  41  42  46  62  65  91  99 150
```

Podemos correr un ciclo for para obtener todas las palabras de los abstracts en un data frame. El data frame contiene el PUBMED ID del artículo y cada palabra del abstract. Dentro del ciclo, se revisa si el abstract está vacío o no.

```r
# data frame para guardar las palabras
word_list <- c()
#Ciclo para todos los abstracts
for(i in 1:length(pubmed_data$Abstract)){
    #Obtener las palabras como vector en lugar de lista
    aux_word <- unlist(strsplit(pubmed_data$Abstract[i], " "))
    #Si el abstract tiene palabras
    if(length(aux_word) > 0){
        #Se juntan las palabras y el PUBMED ID
        aux_list <- cbind(pubmed_data$PID[i], aux_word)
        #Se pega este data frame auxiliar al que guarda todo
        word_list <- rbind(word_list, aux_list)
    }
}
colnames(word_list) <- c("PID","Word")
dim(word_list)
```

```
## [1] 81936     2
```

```r
word_list[1:5,]
```

```
##      PID        Word
## [1,] "32133645" "colorectal"
## [2,] "32133645" "cancer"
## [3,] "32133645" "(crc)"
## [4,] "32133645" "is"
## [5,] "32133645" "the"
```

Usando la librería tm podemos obtener una lista de lo que se llama *stopwords* (artículos, adverbios, pronombres, etc.).

```r
library(tm)
```

```
## Loading required package: NLP
```

```r
stop_words <- stopwords(kind="en")
stop_words
```

```
##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
##  [21] "herself"    "it"         "its"        "itself"     "they"
##  [26] "them"       "their"      "theirs"     "themselves" "what"
##  [31] "which"      "who"        "whom"       "this"       "that"
##  [36] "these"      "those"      "am"         "is"         "are"
##  [41] "was"        "were"       "be"         "been"       "being"
##  [46] "have"       "has"        "had"        "having"     "do"
##  [51] "does"       "did"        "doing"      "would"      "should"
##  [56] "could"      "ought"      "i'm"        "you're"     "he's"
##  [61] "she's"      "it's"       "we're"      "they're"    "i've"
```

```
##  [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
##  [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
##  [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
##  [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
##  [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
##  [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
##  [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
## [101] "who's"      "what's"     "here's"     "there's"    "when's"
## [106] "where's"    "why's"      "how's"      "a"          "an"
## [111] "the"        "and"        "but"        "if"         "or"
## [116] "because"    "as"         "until"      "while"      "of"
## [121] "at"         "by"         "for"        "with"       "about"
## [126] "against"    "between"    "into"       "through"    "during"
## [131] "before"     "after"      "above"      "below"      "to"
## [136] "from"       "up"         "down"       "in"         "out"
## [141] "on"         "off"        "over"       "under"      "again"
## [146] "further"    "then"       "once"       "here"       "there"
## [151] "when"       "where"      "why"        "how"        "all"
## [156] "any"        "both"       "each"       "few"        "more"
## [161] "most"       "other"      "some"       "such"       "no"
## [166] "nor"        "not"        "only"       "own"        "same"
## [171] "so"         "than"       "too"        "very"
```

Podemos guardar los índices de las palabras de nuestra lista que forman parte de las *stopwords* y que deben ser removidas.

```
index_stop_word <- which(word_list[,2] %in% stop_words)
length(index_stop_word)
```

```
## [1] 28466
```

Después las quitamos de la lista.

```
dim(word_list)
```

```
## [1] 81936     2
```

```
word_list <- word_list[-index_stop_word,]
dim(word_list)
```

```
## [1] 53470     2
```

Usamos sort y table para mostrar las palabras más frecuentes, en este caso el top 10.

```
sort(table(word_list[,2]), decreasing=T)[1:10]
```

```
##
##     cancer   patients      colon      young colorectal        age
##       1285       1134        653        468        465        446
##      years  treatment       risk      study
##        306        283        241        223
```

Recordemos que tenemos 331 documentos, ¿por qué la palabra cáncer aparece más de 1000 veces? Esto se debe a que las palabras pueden estar repetidas en los mismos abstracts, por lo que debemos quitar las palabras repetidas en cada abstract. Para eso creamos un data frame igual que word_list pero agregando una columna con la concatenación de pubmed id y palabra.

```
word_df <- data.frame(PID=as.numeric(word_list[,1]), Word=word_list[,2],
                      PIDWord=as.character(apply(word_list, 1, paste, collapse="_")))
word_df[1:5,]
```

```
##        PID       Word             PIDWord
## 1 32133645 colorectal 32133645_colorectal
## 2 32133645     cancer     32133645_cancer
## 3 32133645      (crc)      32133645_(crc)
## 4 32133645     second     32133645_second
## 5 32133645     common     32133645_common
```

Con la columna PIDWord podemos ver cuál palabra está repetida en el mismo abstract de acuerdo a su PUBMED ID. La función duplicated nos puede servir para identificar duplicados en un vector; regresa FALSE para la primera ocurrencia de un elemento y TRUE para las siguientes ocurrencias.

```
dup_index <- duplicated(word_df$PIDWord)
word_df$PIDWord[1:10]
```

```
##  [1] "32133645_colorectal" "32133645_cancer"     "32133645_(crc)"
##  [4] "32133645_second"     "32133645_common"     "32133645_cause"
##  [7] "32133645_cancer"     "32133645_death"      "32133645_united"
## [10] "32133645_states"
```

```
dup_index[1:10]
```

```
##  [1] FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
```

```
length(which(dup_index))
```

```
## [1] 18434
```

Quitamos los duplicados.

```
dim(word_df)
```

```
## [1] 53470     3
```

```
word_df <- word_df[-which(dup_index),]
dim(word_df)
```

```
## [1] 35036     3
```

Volvemos a usar sort y table para mostrar las palabras más frecuentes, en este caso el top 50.

```
sort(table(word_df$Word), decreasing=T)[1:50]
```

```
##
##        cancer         colon         young      patients           age
##           303           295           281           224           180
##    colorectal     treatment         study         years          risk
##           178           168           122           122           115
##     diagnosis           may       disease    hereditary       patient
##           100            99            98            96            94
##      clinical        family         tumor    associated     incidence
##            90            90            90            88            88
##          also      compared         found       history        tumors
##            85            85            84            83            83
##       cancers     diagnosed      mutation         stage          data
##            82            81            81            80            79
##     increased      survival           one    identified         cases
##            79            79            75            73            71
##         among          high     mutations      analysis    background
##            70            69            69            68            68
##       results       surgery           two           can      syndrome
##            67            66            65            63            62
## significantly         early       however       younger        common
##            61            60            59            58            57
```

Quizá este grupo de palabras no sea muy informativo, por lo que una recomendación sería seguir filtrando palabras. Otra opción es buscar palabras específicas relacionadas a cáncer de colon como son los genes asociadas a este cáncer (ejemplo: APC, KRAS y TP53). Primero, podemos ordenar el data frame por PUBMED ID en orden decreciente para tener los artículos más recientes en la parte superior del data frame.

```
word_df <- word_df[order(word_df$PID, decreasing=T),]
index_genes <- which(word_df$Word %in% c("apc","kras","tp53"))
length(index_genes)
```

```
## [1] 28
```

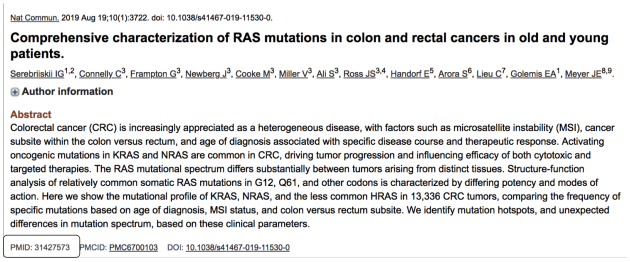
```
word_df[index_genes[1:5], c("PID","Word")]
```

```
##            PID Word
## 2309  31427573 kras
## 2412  31409086  apc
## 4130  30854646  apc
## 4761  30394984 kras
## 11383 27866339 tp53
```

Si buscamos en PUBMED el ID del primer elemento de la búsqueda (31427573), encontramos un artículo muy relacionado con nuestra situación problema (Figura 2). También si en nuestro data frame inicial (pubmed_data) buscamos este ID, podemos ver el título del artículo y el abstract.

```
pubmed_data$Title[which(pubmed_data$PID == "31427573")]
```

```
## [1] "comprehensive characterization of ras mutations in colon and rectal cancers in old and young pa
```



Nat Commun. 2019 Aug 19;10(1):3722. doi: 10.1038/s41467-019-11530-0.

Comprehensive characterization of RAS mutations in colon and rectal cancers in old and young patients.

Serebriiskii IG[1,2], Connelly C[3], Frampton G[3], Newberg J[3], Cooke M[3], Miller V[3], Ali S[3], Ross JS[3,4], Handorf E[5], Arora S[6], Lieu C[7], Golemis EA[1], Meyer JE[8,9].

Author information

Abstract

Colorectal cancer (CRC) is increasingly appreciated as a heterogeneous disease, with factors such as microsatellite instability (MSI), cancer subsite within the colon versus rectum, and age of diagnosis associated with specific disease course and therapeutic response. Activating oncogenic mutations in KRAS and NRAS are common in CRC, driving tumor progression and influencing efficacy of both cytotoxic and targeted therapies. The RAS mutational spectrum differs substantially between tumors arising from distinct tissues. Structure-function analysis of relatively common somatic RAS mutations in G12, Q61, and other codons is characterized by differing potency and modes of action. Here we show the mutational profile of KRAS, NRAS, and the less common HRAS in 13,336 CRC tumors, comparing the frequency of specific mutations based on age of diagnosis, MSI status, and colon versus rectum subsite. We identify mutation hotspots, and unexpected differences in mutation spectrum, based on these clinical parameters.

PMID: 31427573 PMCID: PMC6700103 DOI: 10.1038/s41467-019-11530-0

Figure 2: Resultados de búsqueda por PUBMED ID

```
pubmed_data$PID[which(pubmed_data$PID == "31427573")]
```

```
## [1] "31427573"
```

De esta forma el análisis de texto nos puede ayudar para encontrar información.