POETA

2022-12-15

cargamos la librería que vamos a utilizar para dividir las palabras en sílabas

```
library(sylly)
```

```
## Warning: package 'sylly' was built under R version 4.2.2
```

```
#available.sylly.lang()
#install.sylly.lang("es")
library (sylly.es)
```

Creamos una función para pasarle los sonetos y que los divida en las estrofas correspondientes (estrofas formadas por 14 versos)

```
analizar_sonetos <- function(soneto){
  con <- file(description = soneto, open = 'rt', encoding= 'UTF-8')
  soneto <- readLines(con)
  soneto <- paste (soneto, sep = "\n")
  close(con)

  #reemplazo los signos de puntuación por ""
  soneto <- gsub("[[:punct:]]", "", soneto)

  #convierto a minúsculas el soneto
  soneto <- tolower(soneto)

  #obtenemos las estrofas del soneto mandado
  cont = 1
  estrofas <- list()
  for (i in 1:(trunc(length(soneto)/14))){
    estrofas[[i]] <- soneto[cont:(cont+13)]
    cont = cont+14
  }
  estrofas
}

estrofas <- analizar_sonetos('SONETO SOBRE LA RED DE AMOR.txt')
```

```
## Warning in readLines(con): incomplete final line found on 'SONETO SOBRE LA RED
## DE AMOR.txt'
```

```r
#Vemos la primera estrofa, por ejemplo
estrofas[9]
```

```
## [[1]]
## NULL
```

Creamos una función para contar las sílabas de cada verso, la cual devuelve una lista formada por objetos complejos (S4)

```r
#Debemos pasarle una estrofa solamente(14 versos)
contarSilabas <- function(estrofa){
  palabras <- list()
  silabas <- list()
  for (linea in 1:14){
    palabras[[linea]] <- unlist(strsplit(estrofa[linea], " "))
    silabas[[linea]] <-  hyphen(unlist(palabras[linea]), hyph.pattern="es", min.length = 1, quiet= TRUE)
  }
  silabas
}
```

```r
#estrofas es una lista de 1 o varios elmentos, donde cada uno corresponde a una estrofa
estrofas_silabas <- list()
for (i in 1:length(estrofas)){
  estrofas_silabas[[i]] <- contarSilabas(unlist(estrofas[i]))
}
#estrofas_silabas es una lista formada 14 listas
#(ya que hay 14 versos), donde cada una corresponde a un elemento complejo (S4).
```

```r
#Mostramos la separación por sílabas del primer verso de primera estrofa
estrofas_silabas[[1]][[1]]@hyphen[["word"]]
```

```
## [1] "dí-ga-me" "quien"    "lo"       "sa-be"    "có-mo"    "es"       "he-cha"
```

```r
#También podemos obtener el número de sílabas por verso
estrofas_silabas[[1]][[1]]@desc[["num.syll"]]
```

```
## [1] 12
```

Obtenemos una lista con el número de sílabas de cada verso.

```r
numero_silabas <- list()
for (i in 1:length(estrofas)){
  silabas <- list()
  for (j in 1:14){
    silabas[[j]] <- unlist(estrofas_silabas[[i]][[j]]@desc[["num.syll"]])
  }
  numero_silabas[[i]] <- unlist(silabas)
}
```

```
#mostramos el número de sílabas de cada verso en la primera estrofa,
#sin tener en cuenta aún las normas de la métrica (se aplicarán posteriormetne)
numero_silabas[[1]]
```

```
##  [1] 12 11 12 13 13 11 13 11 12 12 12 12 11 11
```

Obtenemos una lista la cual contenga como elemento la última palabra de cada verso

```
#Creamos una funcion para ello, hay que pasarle la separación en sílabas de una única estrofa
ultima_palabra <- function(silabas){
  lista_ultimas <- list()
  for (i in 1:14){
    verso <- silabas[[i]]@hyphen[["word"]]
    lista_ultimas[[i]] <- verso[length(verso)]
  }
  lista_ultimas
}
```

```
#obtenemos una lista con las últimas palabras de cada verso en cada estrofa
ultimas_total <- list()
for (i in 1:length(estrofas)){
  ultimas_total[[i]] <- unlist(ultima_palabra(estrofas_silabas[[i]]))
}
```

```
#Mostramos la lista formada por las últimas palabras de la primera estrofa por ejemplo
ultimas_total[[1]]
```

```
##  [1] "he-cha"     "pren-de"    "tien-de"    "des-he-cha" "fle-cha"
##  [6] "de-fien-de" "ven-de"     "fle-cha"    "vie-ne"     "cie-go"
## [11] "mi-ra"      "tie-ne"     "fue-go"     "ti-ra"
```

Creamos una función para detectar la existencia de sinalefas de los versos

```
library(stringr)
library(useful)
```

```
## Warning: package 'useful' was built under R version 4.2.2
```

```
## Loading required package: ggplot2
```

```
#le pasamos la lista de sílabas soneto a soneto (14 versos)
sinalefas <- function(silabas_soneto){
  #lista_sinalefas <- list()
  lista_inicio <- list()
  lista_fin <- list()
  for (i in 1:14){
    lista_inicio[[i]] <- str_detect(silabas_soneto[[i]]@hyphen[["word"]], pattern = "(^[aeiouáéíóúh])|(
    lista_fin[[i]] <- str_detect(silabas_soneto[[i]]@hyphen[["word"]], pattern = "([aeiouáéíóú]$)|(y$)"
  }
  lista_sinalefas <- list(lista_inicio, lista_fin)

}
```

```r
sinalefas_cont <- function(lista_inicio, lista_fin){
  cont <- unlist(list((1:14)*0))
  for (i in 1:14){
    inicio <- lista_inicio[[i]][2:length(lista_inicio[[i]])]
    fin <- lista_fin[[i]][1:(length(lista_inicio[[i]])-1)]
    for (j in 1:length(inicio)){
      if (inicio[j] == TRUE & fin[[j]] == TRUE){
        cont[[i]] <- cont[[i]] +1
    }
    }
  }
  cont
}
```

```r
#Obtenemos una lista de listas, donde cada elemento representa a un soneto y dentro de cada elemento (l
contador_sinalefas <- list()
for (i in 1:length(estrofas)){
  posibles_sinalefas <- sinalefas(estrofas_silabas[[i]])
  contador_sinalefas[[i]] <- sinalefas_cont(posibles_sinalefas[[1]], posibles_sinalefas[[2]])
}
contador_sinalefas
```

```
## [[1]]
##  [1] 1 1 1 3 3 0 3 2 1 1 1 1 3 0
##
## [[2]]
##  [1] 0 1 2 1 0 2 1 3 2 1 2 2 4 1
##
## [[3]]
##  [1] 0 1 0 0 1 1 1 1 1 0 1 0 0 1
```

```r
#devuelve el número de sinalefas que hay en cada verso para cada soneto
```

Creamos una función para conocer el número de sílabas que hay en cada verso y qué tipo de palabra es la última de cada verso (llana, aguda o esdrújula), ya que si esta es esdrújula, se resta uno a la métrica, si es aguda se suma, y si es llana se mantine igual.

```r
#Debemos pasarle la última palabra de cada verso
tipo_palabra_recuento <- function(ultima, recuento){
  #buscamos si es esdrújula, llana o aguda
  ultima <- unlist(strsplit(ultima, "-"))

  #si lleva tilde
  tilde <- grep(pattern = "[áéíóú]", rev(ultima))

  aguda_notilde <- grep(pattern = "[^nsaeiou]$", rev(ultima[1]))

  tipo_palabra <- c("llana")
  if (length(tilde) != 0){
    if (tilde >= 3){
      recuento <- recuento -1 #palabra esdrújula
      tipo_palabra = "esdrujula"
```

```
    }
    if (tilde==1){
      recuento <- recuento + 1 #palabra aguda
      tipo_palabra <- c("aguda")
    }
  }else
  {
    if (length(aguda_notilde) != 0){
      recuento <- recuento +1 #palabra aguda que no termina en vocal, n o s
      tipo_palabra <- c("aguda")
    }
  }
  sol <- c(recuento, tipo_palabra)
}

#devuelve una cadena, donde el primer elemento es la métrica, y el segundo el tipo de palabra
```

```
#Aplicamos el recuento a la lista de sílabas de cada verso (se suma uno si es esdrújula, se resta uno s
num_silabas_final <- list()
for (i in 1:length(estrofas)){
  silabas_final <- list()
  for (j in 1:14){
    ultima <- ultimas_total[[i]][j]
    recuento <- numero_silabas[[i]][j]
    silabas_final[[j]] <- tipo_palabra_recuento(ultima, recuento)[1]
  }
  num_silabas_final[[i]] <- silabas_final
}
```

```
#accedemos a cada elemento de la lista que contiene el número de sílabas de cada verso para cada soneto
silabas_menos_sinalefas <- list()
for (i in 1:length(estrofas)){
  total_silabas <- list()
  for (j in 1:14){
    total_silabas[[j]] <- unlist(as.numeric(num_silabas_final[[i]][j])) - contador_sinalefas[[i]][j]
    #es necesario pasar a tipo numérico
  }
  silabas_menos_sinalefas[[i]] <- total_silabas
}
```

```
#Mostramos el recuento actual de sílabas de la segunda estrofa y comparamos con la lista anterior
unlist(silabas_menos_sinalefas[[2]])
```

```
##  [1] 11 11 11 11 11 11 11  9 10 10  9  9 10 10
```

```
unlist(num_silabas_final[[2]])
```

```
##  [1] "11" "12" "13" "12" "11" "13" "12" "12" "12" "11" "11" "11" "14" "11"
```

Creamos una función para quedarnos con el final de la palabra desde la vocal tónica (necesario para ver la métrica)

```
library(stringr)

#Debemos pasarle la lista de las últimas palabras de cada verso una a una(ultimas_total)
vocal_tonica <- function(ultima, recuento){
  tipo_palabra <- tipo_palabra_recuento(ultima, recuento)[2]
  if (tipo_palabra == "esdrujula"){
    tonica <- str_match(pattern = "[ááíóú].*", ultima)
    tonica <- gsub("-", "", tonica)
  }else if (tipo_palabra == "aguda"){
    ultima <- unlist(strsplit(ultima, "-"))
    tonica <- str_match(pattern = "[aeiouáéíóú].*", rev(ultima)[1])
    tonica <- gsub("-", "", tonica)
  }else{
    ultima <- gsub("-", "", ultima)
    tonica <- str_match(pattern = "[aeiou][^(aeiou)]{1,4}[aeiou][^aeiou]{0,2}$|[aeiou][aeiou][^aeiou]{0
  }
  tonica
}
```

```
tonicas_total <- list()
for (i in 1:length(estrofas)){
  tonicas_estrofas <- list()
  for (j in 1:14){
      ultima <- ultimas_total[[i]][j]
      recuento <- numero_silabas[[i]][j]
      #recuento <- estrofas_silabas[[i]][[j]]@desc[["num.syll"]]
      tonicas_estrofas[[j]] <- unlist(vocal_tonica(ultima, recuento))
  }
  tonicas_total[[i]] <- unlist(tonicas_estrofas)
}
```

```
#Mostramos la lista formada por la terminación desde la vocal tónica de la última palabra de cada verso
#Esta lista corresponde a la primera estrofa.
tonicas_total[[1]]
```

```
##  [1] "echa" "ende" "ende" "echa" "echa" "ende" "ende" "echa" "ene"  "ego"
## [11] "ira"  "ene"  "ego"  "ira"
```

Creamos una función para comprobar la rima

```
#Debemos pasarle una lista con las terminaciones desde la vocal tónica de de las últimas palabras de lo
todasIguales <- function(vector_silabas){
  return (length(unique(vector_silabas))==1)
}
cumpleRima <- function(ultima){
  #Cuartetos
  A <- c(ultima[1], ultima[4], ultima[5], ultima[8])
  B <- c(ultima[2], ultima[3], ultima[6], ultima[7])

  if (todasIguales(A) & todasIguales(B)){
    #Tercetos
    C <- c(ultima[9], ultima[12])
```

```
    D <- c(ultima[10], ultima[13])
    E <- c(ultima[11], ultima[14])
    if (todasIguales(C) & todasIguales(D) & todasIguales(E)){
      return (TRUE)
    }else{
      C <- c(ultima[9], ultima[13])
      D <- c(ultima[10], ultima[12])
      E <- c(ultima[11], ultima[14])
      if (todasIguales(C) & todasIguales(D) & todasIguales(E)){
        return (TRUE)
      }else{
        C <- c(ultima[9], ultima[13])
        D <- c(ultima[10], ultima[12])
        E <- c(ultima[11], ultima[14])
        if (todasIguales(C) & todasIguales(D) & todasIguales(E)){
          return (TRUE)
        }else{
          return(FALSE)
        }
      }
    }
  }else{
    return(FALSE)
  }
}
```

```
#Comprobamos si cumple la rima
cumplir_rima <- list()
for (i in 1:length(estrofas)){
    cumplir_rima[[i]] <- unlist(cumpleRima(tonicas_total[[i]]))
}
```

```
#Es una lista de TRUE o FALSE, según el soneto cumpla la rima o no
cumplir_rima
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] FALSE
##
## [[3]]
## [1] FALSE
```

```
#Comprobamos si cumple la métrica o no
cumplir_metrica <- list()
for (i in 1:length(estrofas)){
  resultado <- TRUE
  for (j in 1:14){
    if (silabas_menos_sinalefas[[i]][j] != 11){
      resultado <- FALSE
    }
  }
```

```
  cumplir_metrica[[i]] <- resultado
}
```

```
#Es una lista de TRUE o FALSE, según el soneto cumpla la métrica o no
cumplir_metrica
```

```
## [[1]]
## [1] FALSE
##
## [[2]]
## [1] FALSE
##
## [[3]]
## [1] FALSE
```

```
#Indica TRUE si es un soneto y False si no lo es (en el orden de las estrofas)
final <- list()
for (i in 1:length(cumplir_rima)){
  if (cumplir_rima[i] == TRUE & cumplir_metrica[i]==TRUE){
    final[[i]] <- TRUE
  }else
    final[[i]] <- FALSE
}
final
```

```
## [[1]]
## [1] FALSE
##
## [[2]]
## [1] FALSE
##
## [[3]]
## [1] FALSE
```

```
for (i in 1:length(final)){
  if (final[[i]] == TRUE){
      cat('El soneto', i, 'tiene la siguiente métrica = ', unlist(silabas_menos_sinalefas[[i]]), '-->',
      print("")
      cat('Tiene rima ', cumplir_rima[[i]], ',por lo que sí es un soneto. ')
      print("")
      print("")
  }else{
    cat('El soneto', i, 'tiene la siguiente métrica = ', unlist(silabas_menos_sinalefas[[i]]), '-->', cu
    print("")
    cat('Tiene rima ', cumplir_rima[[i]], ',por lo que no es un soneto. ')
    print("")
    print("")
  }
}
```

```
## El soneto 1 tiene la siguiente métrica =  11 10 11 10 10 11 10 9 11 11 11 11 8 11 --> FALSE[1] ""
## Tiene rima  TRUE ,por lo que no es un soneto. [1] ""
```

```
## [1] ""
## El soneto 2 tiene la siguiente métrica =  11 11 11 11 11 11 11 9 10 10 9 9 10 10 --> FALSE[1] ""
## Tiene rima  FALSE ,por lo que no es un soneto. [1] ""
## [1] ""
## El soneto 3 tiene la siguiente métrica =  10 12 11 10 10 9 12 9 11 11 12 11 11 12 --> FALSE[1] ""
## Tiene rima  FALSE ,por lo que no es un soneto. [1] ""
## [1] ""
```