Practica_PLN_Grupo08

2026-01-12

Práctica 8

GRUPO 8.

Emails :

Pablo Grano de oro Sevillano -> pablo.granodeoro@alumnos.upm.es

Miguel Sanchez Lerin -> miguel.sanchez.lerin@alumnos.upm.es

Carolina Simal Liñan -> carolina.simal@alumnos.upm.es

Alberto Gil Castro -> alberto.gilcastro@alumnos.upm.es

Gerardo Harguindey Rodriguez -> gerardo.harguindey@alumnos.upm.es

Gonzalo García Macias -> gonzalo.garcia.macias@alumnos.upm.es

CUIDADO AL EJECUTAR (Limpia el Global Enviroment)

```
rm(list = ls())
```

1º Parte

Lectura del archivo 10000 palabras

```
# Leer el archivo
lineas <- readLines("datos/10000_formas_ortograficas.txt",
                    encoding = "UTF-8")
lineas[0:4]
```

```
## [1] "Forma\tFrecuencia\tFrec. norm." ",\t27823866\t56483.65"
## [3] "de\t26286396\t53362.52"         ".\t19226627\t39030.88"
```

```
# Buscar donde empieza la priemra linea
linea_inicio <- grep("^[^0-9\t\n]+\t[0-9]+\t[0-9]+", lineas)[1]

# Extraer 10000 líneas a partir del inicio
datos <- lineas[linea_inicio:(linea_inicio + 9999)]

# Separar cada línea por tabulador (\t)
partes <- strsplit(datos, "\t")
partes[0:4]
```

```
## [[1]]
## [1] ","         "27823866" "56483.65"
##
## [[2]]
## [1] "de"        "26286396" "53362.52"
##
## [[3]]
## [1] "."         "19226627" "39030.88"
##
## [[4]]
## [1] "la"        "15799962" "32074.6"
```

```
# Extraer las columnas
formas <- sapply(partes, function(x) x[1])
frecuencias <- as.numeric(sapply(partes, function(x) x[2]))
frec_norm <- as.numeric(sapply(partes, function(x) x[3]))

# Crear el dataframe
tabla <- data.frame(Forma = formas,
                    Frecuencia = frecuencias,
                    Frec.norm = frec_norm,
                    stringsAsFactors = FALSE)
head(tabla, 5)
```

```
##   Forma Frecuencia Frec.norm
## 1     ,   27823866  56483.65
## 2    de   26286396  53362.52
## 3     .   19226627  39030.88
## 4    la   15799962  32074.60
## 5   que   13350795  27102.69
```

```
tail(tabla, 5)
```

```
##                 Forma Frecuencia Frec.norm
## 9996          perciben       3421      6.94
## 9997          católico       3421      6.94
## 9998         convertía       3421      6.94
## 9999  especialización       3421      6.94
## 10000           golpeó       3421      6.94
```

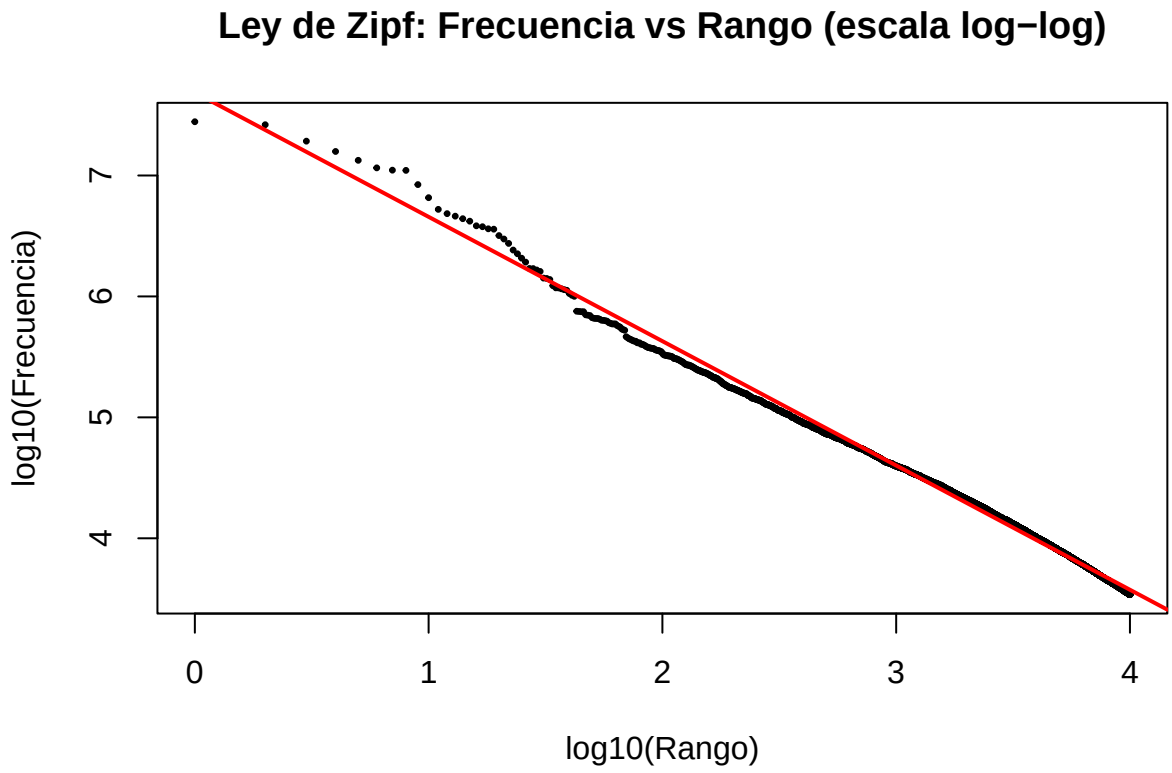
Ley de Zipf

```
## APARTADO 1
# Creamos la columna de rangos
tabla$rango <- 1:nrow(tabla)

# Gráfico log-log
plot(log10(tabla$rango),
     log10(tabla$Frecuencia),
     pch = 20, cex = 0.5,
     xlab = "log10(Rango)",
```

```
     ylab = "log10(Frecuencia)",
     main = "Ley de Zipf: Frecuencia vs Rango (escala log-log)")

# Ajuste lineal para visualizar la tendencia
modelo <- lm(log10(Frecuencia) ~ log10(rango), data = tabla)
abline(modelo, col = "red", lwd = 2)
```




```
cat("\nAjuste Zipf (lista total, log-log):\n")
```

```
##
## Ajuste Zipf (lista total, log-log):
```

```
print(summary(modelo)$coefficients)
```

```
##                Estimate   Std. Error   t value Pr(>|t|)
## (Intercept)    7.685801 0.0023535766  3265.584        0
## log10(rango) -1.027861 0.0006551978 -1568.781        0
```

Apartado 1.1

```
duplicadas1 <- tabla$Forma[duplicated(tabla$Forma)]
duplicadas1 <- unique(duplicadas1)
```

```r
cat("Número de formas repetidas exactas:", length(duplicadas1), "\n")
```

```
## Número de formas repetidas exactas: 0
```

```r
# 2) Construir lista detallada (forma + línea + frecuencia) para cada copia
if (length(duplicadas1) > 0) {

  lista_repetidas1 <- lapply(duplicadas1, function(f) {
    indices <- which(tabla$Forma == f)
    data.frame(
      Forma = f,
      Linea = indices,
      Frecuencia = tabla$Frecuencia[indices],
      stringsAsFactors = FALSE
    )
  })

  resultado1 <- do.call(rbind, lista_repetidas1)

  cat("Número total de apariciones (filas) dentro de duplicados:", nrow(resultado1), "\n\n")


  if (nrow(resultado1) > 20) {
    cat("Mostrando 5 primeras y 5 últimas filas de la lista de duplicados:\n\n")
    print(head(resultado1, 5))
    cat("\n...\n\n")
    print(tail(resultado1, 5))
  } else {
    print(resultado1)
  }

} else {
  cat("No hay formas repetidas exactas en este listado.\n")
}
```

```
## No hay formas repetidas exactas en este listado.
```

Apartado 1.2

```r
library(stringi)
## Apartado 1.2

# Pasar todo a minúsculas
tabla$forma_base <- stri_trans_general(tolower(tabla$Forma), "Latin-ASCII")


# Contar cuántas formas básicas tienen variantes
conteo <- table(tabla$forma_base)
duplicados <- conteo[conteo > 1]

cat("Formas básicas con duplicados no exactos:", length(duplicados), "\n")
```

```
## Formas básicas con duplicados no exactos: 832
```

```r
# Mostrar primeros y últimos 5 ejemplos
n_dups <- length(duplicados)

if (n_dups > 10) {
  # Bloque de los 5 primeros
  for (i in 1:5) {
    base_actual <- names(duplicados)[i]
    # Buscamos en la tabla original todas las formas que coinciden con esa base
    variantes <- unique(tabla$Forma[tabla$forma_base == base_actual])
    cat(i, " '", base_actual, "' tiene las variantes: ", paste(variantes, collapse = ", "), "\n", sep =
  }

  # Separador visual conforme a la práctica
  cat("...\n")

  # Bloque de los 5 últimos
  for (i in (n_dups - 4):n_dups) {
    base_actual <- names(duplicados)[i]
    variantes <- unique(tabla$Forma[tabla$forma_base == base_actual])
    cat(i, " '", base_actual, "' tiene las variantes: ", paste(variantes, collapse = ", "), "\n", sep =
  }

} else {
  # Si hay pocos resultados, se muestran todos
  for (i in 1:n_dups) {
    base_actual <- names(duplicados)[i]
    variantes <- unique(tabla$Forma[tabla$forma_base == base_actual])
    cat(i, " '", base_actual, "' tiene las variantes: ", paste(variantes, collapse = ", "), "\n", sep =
  }
}
```

```
## 1 ''' tiene las variantes: ', ’, ‘
## 2 '-' tiene las variantes: -, –, -
## 3 '!' tiene las variantes: !, ¡
## 4 '"' tiene las variantes: ", ", "
## 5 '?' tiene las variantes: ?, ¿
## ...
## 828 'x' tiene las variantes: X, x
## 829 'y' tiene las variantes: y, Y
## 830 'ya' tiene las variantes: ya, Ya
## 831 'yo' tiene las variantes: yo, Yo
## 832 'zona' tiene las variantes: zona, Zona
```

Apartado 1.3

```r
library(stringi)
#Normalización
formas_norm <- stri_trans_nfc(tabla$Forma) # normaliza acentos
formas_norm <- gsub("\u00A0", " ", formas_norm)
formas_norm <- gsub("\t", " ", formas_norm)
```

```r
patron <- "^[A-Za-zñÑáéíóúÁÉÍÓÚüÜ0-9[:space:][:punct:]]+$"

# Marcamos como no español todo lo que NO encaje exactamente con ese patrón de letras
no_espanol <- !stri_detect_regex(formas_norm, patron)

# Conteo de resultados
num_no_espanol <- sum(no_espanol)
cat("Con la criba estricta, se han detectado", num_no_espanol, "formas no españolas.")
```

```
## Con la criba estricta, se han detectado 9 formas no españolas.
```

```r
# Verificación de lo que HEMOS QUITADO
cat("\nEjemplos de formas EXCLUIDAS (No españolas):\n")
```

```
##
## Ejemplos de formas EXCLUIDAS (No españolas):
```

```r
print(head(tabla$Forma[no_espanol], 5))
```

```
## [1] " "   "+"   "$"   "="   "US$"
```

```r
print(tail(tabla$Forma[no_espanol], 5))
```

```
## [1] "US$"   "Barça" "Nº"    "|"     ">"
```

```r
#  Verificación de lo que SE QUEDA
cat("\nEjemplos de formas PERMITIDAS (Españolas):\n")
```

```
##
## Ejemplos de formas PERMITIDAS (Españolas):
```

```r
print(head(tabla$Forma[!no_espanol], 5))
```

```
## [1] ","   "de"  "."   "la"  "que"
```

```r
print(tail(tabla$Forma[!no_espanol], 5))
```

```
## [1] "perciben"        "católico"        "convertía"       "especialización"
## [5] "golpeó"
```

2º Parte

Lectura del archivo 1M palabras

```r
# Leer el archivo
lineas2 <- readLines("datos/frecuencia_formas_ortograficas_1_3.txt",
                     encoding = "UTF-8")
# Buscar donde empieza la priemra linea
linea_inicio2 <- grep("^[^0-9\t\n]+\t[0-9]+\t[0-9]+", lineas2)[1]

# Extraer todas las líneas a partir del inicio
datos2 <- lineas2[linea_inicio2:(linea_inicio2 + length(lineas2)-2)] #El -2 pq sinó aparecen 2 lineas v

# Separar cada línea por tabulador (\t)
partes2 <- strsplit(datos2, "\t")
# Extraer las columnas
formas2 <- sapply(partes2, function(x) x[1])
frecuencias2 <- as.numeric(sapply(partes2, function(x) x[2]))
frec_norm2 <- as.numeric(sapply(partes2, function(x) x[3]))

# Crear el dataframe
tabla2 <- data.frame(Forma = formas2,
                     Frecuencia = frecuencias2,
                     Frec.norm = frec_norm2,
                     stringsAsFactors = FALSE)
head(tabla2, 5)
```

```
##   Forma Frecuencia Frec.norm
## 1     ,   27823866  56483.65
## 2    de   26286396  53362.52
## 3     .   19226627  39030.88
## 4    la   15799962  32074.60
## 5   que   13350795  27102.69
```

```r
tail(tabla2, 5)
```

```
##                   Forma Frecuencia Frec.norm
## 1484612        Paulitis          1         0
## 1484613       Paulistáo          1         0
## 1484614      Paulistana          1         0
## 1484615         Paulist          1         0
## 1484616 pauljohnsonesco          1         0
```

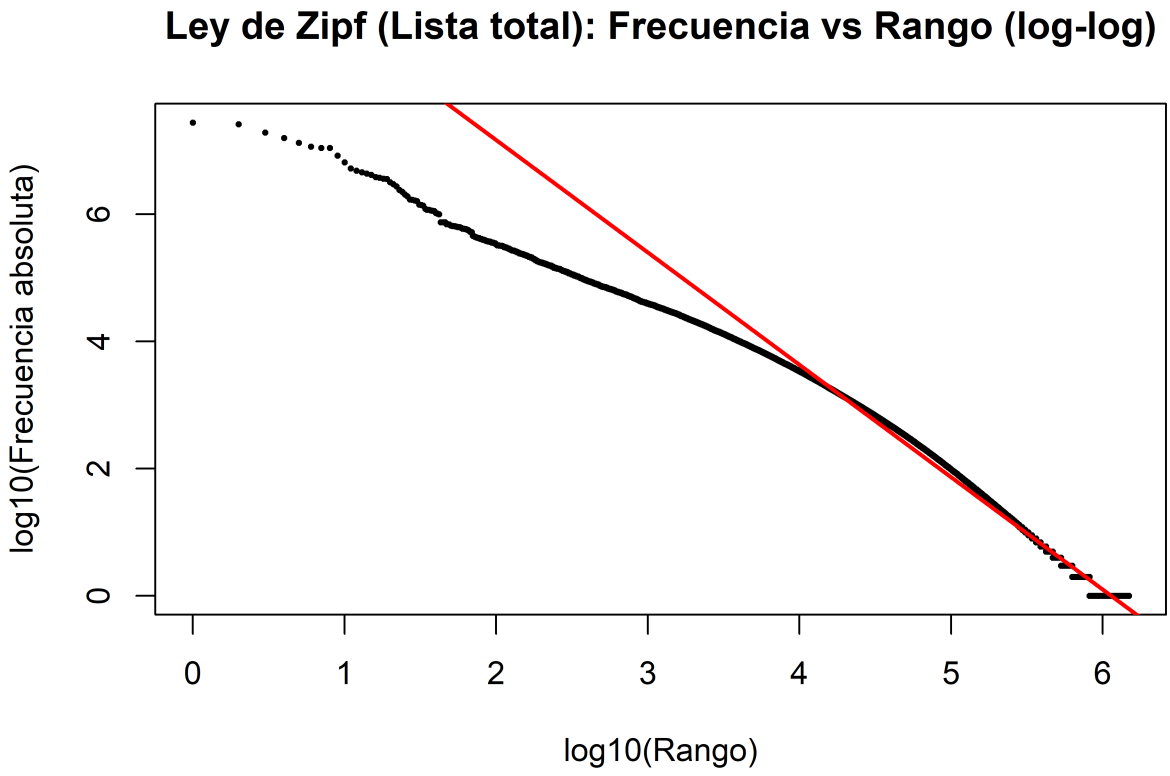
Ley de Zipf 2

```r
tabla2_ord <- tabla2[order(-tabla2$Frecuencia), ]
tabla2_ord$rango <- 1:nrow(tabla2_ord)

# Gráfico
plot(
  log10(tabla2_ord$rango),
  log10(tabla2_ord$Frecuencia),
  pch = 20, cex = 0.5,
  xlab = "log10(Rango)",
```

```
  ylab = "log10(Frecuencia absoluta)",
  main = "Ley de Zipf (Lista total): Frecuencia vs Rango (log-log)"
)

# Ajuste lineal para visualizar la recta
modelo2 <- lm(log10(Frecuencia) ~ log10(rango), data = tabla2_ord)
abline(modelo2, col = "red", lwd = 2)
```




```
# mostrar coeficientes del ajuste en consola
cat("\nAjuste Zipf (lista total, log-log):\n")
```

```
##
## Ajuste Zipf (lista total, log-log):
```

```
print(summary(modelo2)$coefficients)
```

```
##                Estimate   Std. Error   t value Pr(>|t|)
## (Intercept)  10.709561 0.0011882653  9012.770        0
## log10(rango) -1.768268 0.0002065207 -8562.184        0
```

Apartado 2.1

```r
## Apartado 2.1

# 1) Formas duplicadas exactas (mismas letras)
duplicadas2 <- tabla2$Forma[duplicated(tabla2$Forma)]
duplicadas2 <- unique(duplicadas2)

cat("Número de formas repetidas exactas:", length(duplicadas2), "\n")
```

```
## Número de formas repetidas exactas: 7
```

```r
# 2) Lista detallada: forma + línea(s) + frecuencia(s)
if (length(duplicadas2) > 0) {

  lista_repetidas2 <- lapply(duplicadas2, function(f) {
    indices <- which(tabla2$Forma == f)
    data.frame(
      Forma = f,
      Linea = indices,
      Frecuencia = tabla2$Frecuencia[indices],
      stringsAsFactors = FALSE
    )
  })

  resultado2 <- do.call(rbind, lista_repetidas2)

  # Mostrar cuántas filas (apariciones totales de duplicados)
  cat("Número total de apariciones (filas) dentro de duplicados:", nrow(resultado2), "\n\n")

  # Si es muy largo, mostramos 5 primeras y 5 últimas (como pide el enunciado)
  if (nrow(resultado2) > 20) {
    cat("Mostrando 5 primeras y 5 últimas filas de la lista de duplicados:\n\n")
    print(head(resultado2, 5))
    cat("\n...\n\n")
    print(tail(resultado2, 5))
  } else {
    print(resultado2)
  }

} else {
  cat("No hay formas repetidas exactas en este listado.\n")
}
```

```
## Número total de apariciones (filas) dentro de duplicados: 14
##
##      Forma   Linea Frecuencia
## 1     AT&T   45435        387
## 2     AT&T  682579          2
## 3      C&T  137667         53
## 4      C&T  705286          2
## 5  Partido    1313      31446
## 6  Partido  734611         NA
## 7      O&M  129793         60
```

```
## 8      O&M  812304          2
## 9        &    3200      13032
## 10       &  824851          1
## 11     S&P   38775        497
## 12     S&P  929219          1
## 13 crédito    2241      18923
## 14 crédito 1193546          1
```

Apartado 2.2

```
library(stringi)
## Apartado 2.2

# Pasar todo a minúsculas
tabla2$forma_base <- stri_trans_general(tolower(tabla2$Forma), "Latin-ASCII")
# Contar cuántas formas básicas tienen variantes
conteo <- table(tabla2$forma_base)
duplicados <- conteo[conteo > 1]

cat("Formas básicas con duplicados no exactos:", length(duplicados), "\n")
```

```
## Formas básicas con duplicados no exactos: 240736
```

```
# Mostrar primeros y últimos 5 ejemplos
n_dups <- length(duplicados)
cat("\nMostrando formas con variantes (5 primeras y 5 últimas):\n")
```

```
##
## Mostrando formas con variantes (5 primeras y 5 últimas):
```

```
if (n_dups > 10) {
  # Bloque de los 5 primeros
  for (i in 1:5) {
    base_actual <- names(duplicados)[i]
    # Buscamos en la tabla original todas las formas que coinciden con esa base
    variantes <- unique(tabla2$Forma[tabla2$forma_base == base_actual])
    cat(i, " '", base_actual, "' tiene las variantes: ", paste(variantes, collapse = ", "), "\n", sep =
  }

  # Separador visual
  cat("...\n")

  # Bloque de los 5 últimos
  for (i in (n_dups - 4):n_dups) {
    base_actual <- names(duplicados)[i]
    variantes <- unique(tabla2$Forma[tabla2$forma_base == base_actual])
    cat(i, " '", base_actual, "' tiene las variantes: ", paste(variantes, collapse = ", "), "\n", sep =
  }

} else {
  # Si hay pocos resultados, se muestran todos
```

```
  for (i in 1:n_dups) {
    base_actual <- names(duplicados)[i]
    variantes <- unique(tabla2$Forma[tabla2$forma_base == base_actual])
    cat(i, " '", base_actual, "' tiene las variantes: ", paste(variantes, collapse = ", "), "\n", sep =
  }
}
```

```
## 1 ''' tiene las variantes: ', ’, ‘
## 2 '-' tiene las variantes: -, -, -,  ,  , -,  ,
## 3 '-a' tiene las variantes: a,  A,  a,  a, -a
## 4 '-ademas' tiene las variantes:  además, -además
## 5 '-agrego-' tiene las variantes:  agregó ,  agregó
## ...
## 240732 ' ' tiene las variantes:  , T
## 240733 ' ' tiene las variantes: Φ,
## 240734 '   ' tiene las variantes: Φ  ,
## 240735 ' ' tiene las variantes: X,
## 240736 ' ' tiene las variantes:  , Ψ
```

Apartado 2.3

```
library(stringi)

#Normalización
formas_norm2 <- stri_trans_nfc(tabla2$Forma) # normaliza acentos
formas_norm2 <- gsub("\u00A0", " ", formas_norm2)
formas_norm2 <- gsub("\t", " ", formas_norm2)


patron <- "^[A-Za-zñÑáéíóúÁÉÍÓÚüÜ0-9[:space:][:punct:]]+$"

# Marcamos como no español todo lo que NO encaje exactamente con ese patrón de letras
no_espanol2 <- !stri_detect_regex(formas_norm2, patron)

# Conteo de resultados
num_no_espanol2 <- sum(no_espanol2)
cat("Con la criba estricta, se han detectado", num_no_espanol2, "formas no españolas.")
```

```
## Con la criba estricta, se han detectado 22630 formas no españolas.
```

```
# Verificación de lo que HEMOS QUITADO
cat("\nEjemplos de formas EXCLUIDAS (No españolas):\n")
```

```
##
## Ejemplos de formas EXCLUIDAS (No españolas):
```

```
head(tabla2$Forma[no_espanol2], 5)
```

```
## [1] " "   "+"   "$"   "="   "US$"
```

```r
tail(tabla2$Forma[no_espanol2], 5)
```

```
## [1] "Pâtre"              "PÀTRICIA"           "patrilla$$$thrilla"
## [4] "Patrimônio"         "Paulão"
```

```r
#  Verificación de lo que SE QUEDA
cat("\nEjemplos de formas PERMITIDAS (Españolas):\n")
```

```
##
## Ejemplos de formas PERMITIDAS (Españolas):
```

```r
head(tabla2$Forma[!no_espanol2], 5)
```

```
## [1] ","   "de"  "."   "la"  "que"
```

```r
tail(tabla2$Forma[!no_espanol2], 5)
```

```
## [1] "Paulitis"        "Paulistáo"       "Paulistana"      "Paulist"
## [5] "pauljohnsonesco"
```

Explicación partes más relevantes

Lo primero que conviene destacar es el salto de escala: no es lo mismo procesar un corpus de 10.000 palabras que uno de 1.000.000. Ese crecimiento implica que cualquier operación innecesaria (recorridos repetidos, transformaciones redundantes, etc.) se vuelve muy costosa. Por eso, el enfoque general del trabajo ha sido mantener scripts lo más eficientes posible, con complejidades simples y evitando pasos prescindibles, especialmente teniendo en cuenta que no disponemos de equipos especialmente potentes.

El primer reto importante fue la lectura de los archivos. Inicialmente descargamos el contenido en formato HTML, pero extraer el texto útil de forma limpia resultaba bastante tedioso. Finalmente encontramos una forma de descargar el contenido directamente en .txt, lo que simplificó mucho el proceso.

Decidimos leer los ficheros en UTF-8 para conservar correctamente caracteres como la "ñ" y las tildes. A partir de ahí, detectamos el inicio del texto relevante mediante una expresión regular (aprovechando que esa cabecera era distinta del resto) y nos quedamos con el contenido a partir de ese punto. Para el análisis posterior, construimos una tabla de frecuencias con la forma y su frecuencia de aparición.

La implementación de la Ley de Zipf no fue especialmente compleja a nivel lógico. Sin embargo, el principal problema apareció en la segunda pregunta, al intentar exportar los resultados a PDF: al manejar tantos datos, la figura no terminaba de renderizarse y el documento no se generaba.

La solución fue configurar el chunk de RMarkdown con:

{r, dev='png', dpi=600, cache=TRUE}

Esta configuración es mucho más eficiente que dejar los valores por defecto, porque fuerza el renderizado del gráfico como imagen raster (mapa de bits) en alta resolución, en lugar de gráfico vectorial. Así evitamos que el PDF intente representar cada punto como un objeto independiente (lo que lo vuelve pesado o incluso bloquea el proceso). Además, el caché evita recalcular lo mismo varias veces, reduciendo tiempos de compilación.

Apartados 1.1 y 2.1 (duplicados exactos)

En estos apartados no hubo especial dificultad: se detectaron las formas exactamente iguales y, mediante una condición, en caso de existir repeticiones se generó una lista con:

la forma repetida,

la(s) línea(s) en las que aparece,

y su frecuencia.

Apartados 1.2 y 2.2 (duplicados no exactos)

Como aquí se pedían duplicados no exactos, el paso clave fue realizar una normalización previa: convertir a minúsculas y eliminar marcas diacríticas (tildes) y otros signos, incluyendo la sustitución de ñ → n, usando "Latin-ASCII" para asegurarnos de no dejar casos sin cubrir.

Somos conscientes de que esta normalización es relativamente agresiva, pero en la práctica ofrece resultados consistentes y permite detectar un gran número de duplicados que, de otro modo, quedarían separados por pequeñas diferencias ortográficas.

Apartados 1.3 y 2.3 (caracteres no españoles)

En este último bloque el objetivo era identificar formas con caracteres no propios del alfabeto español. La principal complicación fue que no todos los acentos (y ciertos espacios/tabuladores) estaban codificados de forma uniforme. Por ello, primero realizamos una normalización adicional para unificar:

variantes de acentuación,

tabuladores,

y ciertos tipos de espacios "raros" que daban problemas.

Después definimos un patrón mediante expresiones regulares que selecciona cualquier carácter fuera del conjunto que consideramos válido en español. Finalmente, se listaron todas las formas que contenían al menos un carácter no contemplado, identificándolas como candidatas a "no españolas".

Lista de tareas y autores:

Trabajo gestado y organizado por: Pablo Grano de oro Sevillano

Pregunta 1:

Lectura archivos 10K palabras -> Carolina Simal Liñan

Ley Zipf 1 -> Gerardo Harguindey Rodriguez

1.1 -> Gerardo Harguindey Rodriguez

1.2 -> Carolina Simal Liñan

1.3 -> Pablo Grano de oro Sevillano

Pregunta 2:

Lectura archivos 1M palabras -> Pablo Grano de oro Sevillano

Ley Zipf 2 -> Alberto Gil Castro

2.1 -> Alberto Gil Castro

2.2 -> Gonzalo García Macias

2.3 -> Miguel Sanchez Lerin