Practica3

2023-04-28

Integrantes

- Melissa Yauri
- Antonio Atanacio

Pregunta 1

Descomprimir el fichero comprimido que contiene los registros del servidor, y a partir de los datos extraídos, cargar en data frame los registros con las peticiones servidas.

Resolución

La resolución consta en convertir el archivo en un **data.frame()** y convertir cada uno de los datos en el tipo de dato apropiado con las funciones **as.factor()** y **as.numeric()**. Todo ello se muestra en el siguiente código.

```
# Importando el archivo
server_data <-
  read_table("D:\\MAESTRIA-CIBERSEGURIDAD\\SEMINARIO_DATA_SCIENCE\\LABS\\Lab03_lengR\\epa-http.csv",
col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   X3 = col_character(),
##   X4 = col_character(),
##   X5 = col_character(),
##   X6 = col_double(),
##   X7 = col_character()
## )
```

```
# Renonbrando los encabezados de las columnas
colnames(server_data) <- c("Ip_address", "Date", "Method", "Resource",
                           "Protocol", "Response_code", "Bytes")

# PREGUNTA 1
# Conversión de datos

# Formateando la columna Method de "\"GET" a "GET"
server_data$Method <- str_sub(server_data$Method, 2)
# Convirtiendo a datos categóricos de la columna Method
server_data$Method <- as.factor(server_data$Method)
```

```r
# Formateando la columna Protocol de "HTTP/1.0\"" a "HTTP/1.0"
server_data$Protocol <- str_sub(server_data$Protocol, end = -2)
# Convirtiendo a datos categóricos
server_data$Protocol <- as.factor(server_data$Protocol)

# Convirtiendo a datos catégoricos la columna Response_code
server_data$Response_code <- as.factor(server_data$Response_code)

# Formateando la columna Bytes,reemplazando "-" por 0
server_data$Bytes <- str_replace(server_data$Bytes, "-", "0")
# Convirtiendo a datos números
server_data$Bytes <- as.numeric(server_data$Bytes)
```

Pregunta 2

Identificar el número único de usuarios que han interactuado directamente con el servidor de forma segregada según si los usuarios han tenido algún tipo de error en las distintas peticiones ofrecidas por el servidor.

Resolución

La resolución consta de buscar los usuarios únicos con su respectivo código de respuesta, para ello se realiza un filtro solo de valores únicos en de la columna **Ip_address** con el valor de la columna **Response_code** con la función **table** y **filter()**. Luego según el tipo de código se guarda la tabla en una variable para seguidamente realizar el conteo de los usuarios con la función **nrow()**. El resultado es el siguiente:

Code	200	302	304	400	403	404	500	501
Usuarios	2296	970	505	1	5	152	29	11

Todo lo mencionado se realizó con el siguiente código.

```r
# Pregunta 2

# Creando la tabla con las columnas Ip_address y Response_code
address_table<- data.frame(Ip_address = server_data$Ip_address,
                           Response_code = server_data$Response_code)
# Tabla de Frecuencias de la columna Ip_address con Response_code
frequency <- as.data.frame(table(address_table))
# Obteniendo los datos existentes
address_data <- filter(frequency, Freq > 0)
# Ordenando de forma ascendente
address_data <- arrange(address_data, Response_code)
# Obteniendo los valores únicos de la columna Response_code
# Los valores son 200, 302, 304, 400, 403, 404, 500, 501
value_codes <- unique(address_data$Response_code)
# Creando nuevas tablas según el código de respuesta
code_data<- map(value_codes, ~address_data[address_data$Response_code == .x, ])
# Cada tabla según el código de respuesta es guardada en cada variable como data_code_200
names(code_data) <- paste0("data_code_", value_codes)
# Hallando el n° de usuarios según el código de respuesta
code200_users <- nrow(code_data$data_code_200)
code200_users
```

```
## [1] 2296
```

```
# Hay 2296 usuarios con el código de respuesta 200
code302_users <- nrow(code_data$data_code_302)
code302_users
```

```
## [1] 970
```

```
# Hay 970 usuarios con el código de respuesta 302
code304_users <- nrow(code_data$data_code_304)
code304_users
```

```
## [1] 505
```

```
# Hay 505 usuarios con el código de respuesta 304
code400_users <- nrow(code_data$data_code_400)
code400_users
```

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

```
# Hay 1 usuario con el código de respuesta 400
code403_users <- nrow(code_data$data_code_403)
code403_users
```

```
## [1] 5
```

```
# Hay 5 usuarios con el código de respuesta 403
code404_users <- nrow(code_data$data_code_404)
code404_users
```

```
## [1] 152
```

```
# Hay 152 usuarios con el código de respuesta 404
code500_users <- nrow(code_data$data_code_500)
code500_users
```

```
## [1] 29
```

```
# Hay 29 usuarios con el código de respuesta 500
code501_users <- nrow(code_data$data_code_501)
code501_users
```

```
## [1] 11
```

```
# Hay 11 usuarios con el código de respuesta 501
```

Pregunta 3

Analizar los distintos tipos de peticiones HTTP (GET, POST, PUT, DELETE) gestionadas por el servidor, identificando la frecuencia de cada una de estas. Repetir el análisis, esta vez filtrando previamente aquellas peticiones correspondientes a recursos ofrecidos de tipo imagen.

Resolución

La primera parte consta en hallar el número de repeticiones según el tipo de método, todo ello se realizó con la función **table()** y se obtuvo el siguiente resultado:

Método	GET	HEAD	POST
Frequencia	46020	106	1622

El resulatdo presente se halló con el siguiente código.

```
# Pregunta 3
# Hallando el n° de repeticiones según el método GET, POST, HEAD (columma Method)
method_frequency <- table(server_data$Method)
# Mostrando la tabla de frecuencia según el método
method_data <- data.frame(method = names(method_frequency), Frequency = as.vector(method_frequency))
# Muestra de los resultados
print(method_data)
```

```
##   method Frequency
## 1    GET     46020
## 2   HEAD       106
## 3   POST      1622
```

```
# Método GET (46020 repeticiones)
# Método HEAD (106 repeticiones)
# Método POST (1622 repeticiones)
```

La segunda parte de esta pregunta es hallar la frecuencia de los métodos previamente discriminando los recursos tipo imagen, para ello se filtra todos los recursos que no son imagenes con la función **filter()** y **grepl()**. Luego para hallar el n° de repeticiones se utiliza la función **table()**. El resultado es el siguiente:

Método	GET	HEAD	POST
Frequencia	23841	50	1416

Lo mencionado se realizó con el siguiente código.

```
# Hallando el n° de repeticiones descartando los recursos tipo imagen
# Filtrando los recursos que no contengan extensiones tipo imagen,
# discriminando mayúsculas y minusculas de las extensiones
noimages_data <- filter(server_data, !grepl("(?i)\\.(gif|jpg|jpeg|png|bmp)$", Resource))
# Hallando el n° de repeticiones del filtro anterior
method_frequency2 <- table(noimages_data$Method)
# Mostrando la tabla
method2_data <- data.frame(Method = names(method_frequency2),
                           Frequency = as.vector(method_frequency2))
# Muestra de los resultados
print(method2_data)
```

```
##   Method Frequency
## 1    GET     23841
## 2   HEAD        50
## 3   POST      1416
```

```
#    GET     23841
#   HEAD        50
#   POST      1416
```

Pregunta 4

Generar un gráfico que permita visualizar las respuestas del servidor, es decir, la distribución de peticiones según el código de respuesta de esta. Probad distintos tipos de gráficos (por lo menos 2 distintos e incluid estos en el documento RMarkdown).

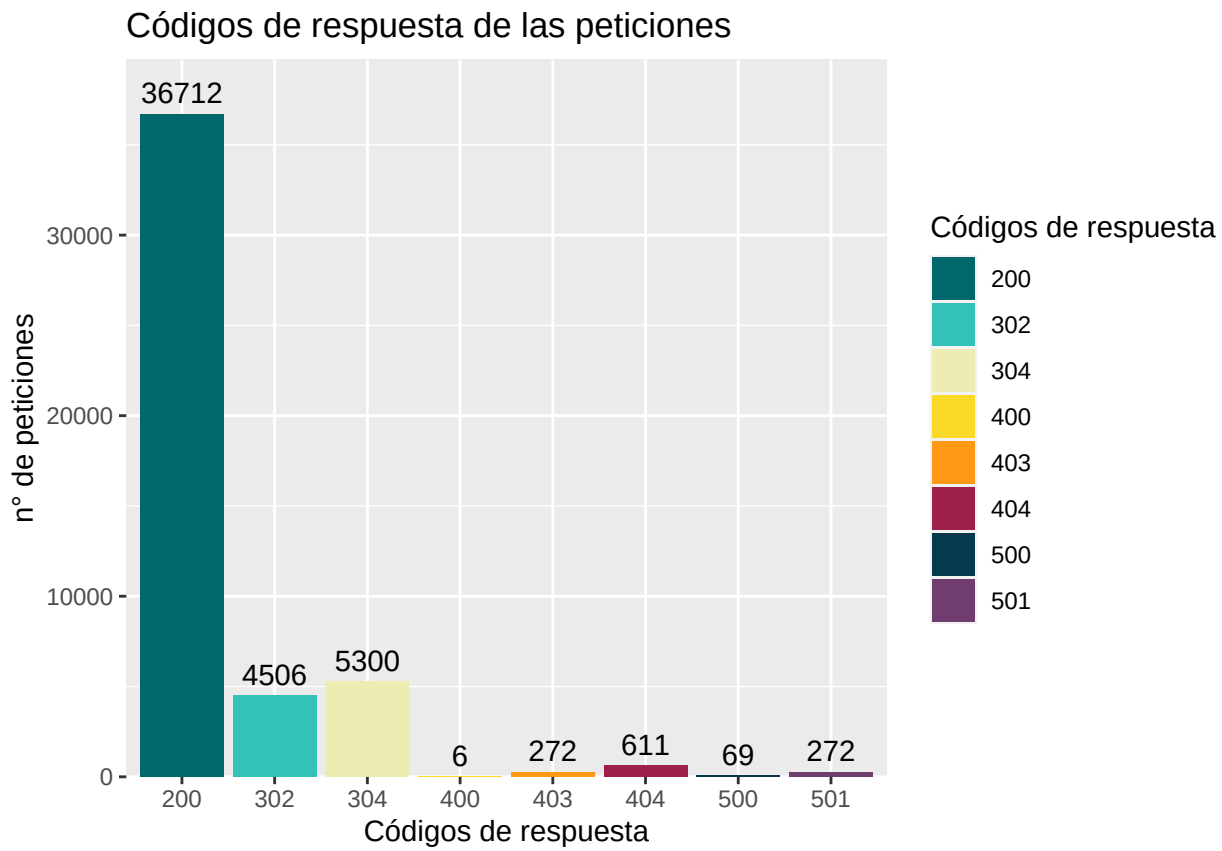
Resolución:

En este caso se eligio dos tipos de gráficos que son los siguientes:

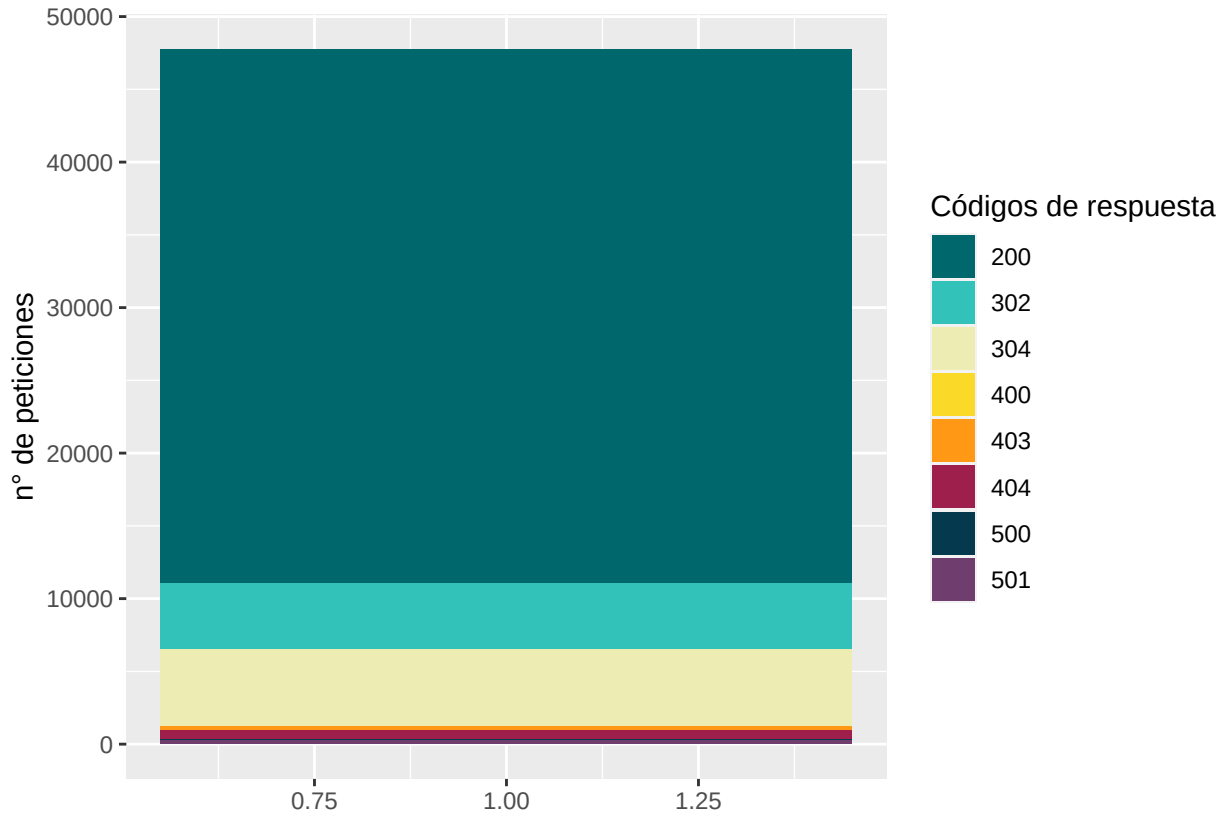
- Graáficos de barras agrupadas y apiladas
- Gráfico de torta

Estas dos gráficas son las más adecuadas para una variable categorica, que en este caso es la columna "Response_code" porque con estas gráficas los datos se representan de manera clara y se logra visualizar la comparación entre las frecuencias a través de las barras y segmentos circulares

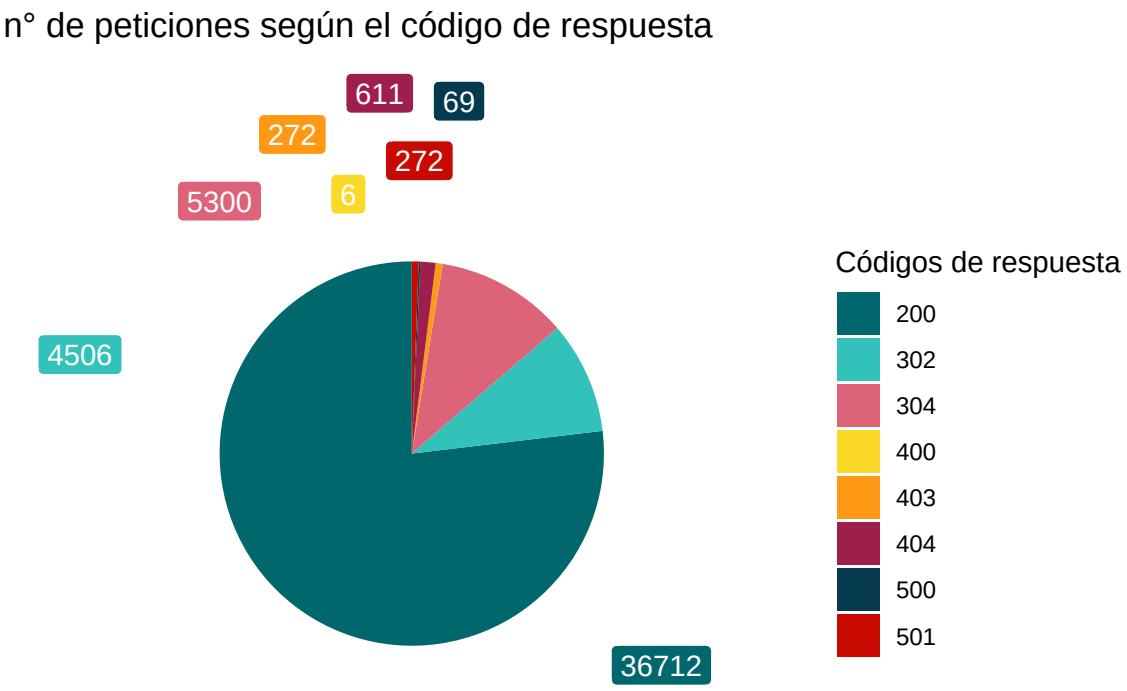
```
# Pregunta 4

# Agrupamos los datos de la columna Response_code y hallamos la frequencia.
response_code_data <- summarize(group_by(server_data, Response_code), Freq = n())

# Gráfico de barras
ggplot(response_code_data, aes(x = Response_code, y = Freq, fill = Response_code)) +
  # Creación del gráfico de barras
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#00686c", "#32c2b9", "#edecb3", "#fad928",
                               "#ff9915", "#9e1e4c","#04394e","#703d6f"),
                    name = "Códigos de respuesta") +
  # Etiquetas del gráfico
  labs(x = "Códigos de respuesta", y = "n° de peticiones",
       title = "Códigos de respuesta de las peticiones") +
  geom_text(aes(label=Freq), vjust= -0.5) +
  # Escala del eje y
  scale_y_continuous(expand = c(0,0), limits = c(-5, max(response_code_data$Freq)+ 3000)) +
  theme_gray()
```




```r
# Gráfico de barras apiladas
ggplot(response_code_data, aes(x = 1, y = Freq, fill = Response_code)) +
  # Creación del gráfico de barras
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#00686c", "#32c2b9", "#edecb3", "#fad928",
                               "#ff9915", "#9e1e4c","#04394e","#703d6f"),
                    name = "Códigos de respuesta") +
  # Etiquetas del gráfico
  xlab("") +
  ylab("n° de peticiones") +
  theme_gray()
```




```
# Gráfico circular
ggplot(response_code_data, aes(x = "", y = Freq, fill = Response_code)) +
  # Creación del gráfico chart
  geom_bar(stat = "identity", width = 1) +
  coord_polar("y", start = 0) +
  theme_void() +
  scale_fill_manual(values =  c("#00686c", "#32c2b9", "#dc6378", "#fad928",
                                "#ff9915", "#9e1e4c","#04394e","#c90a02"),
                    name = "Códigos de respuesta") +
  labs(title = "n° de peticiones según el código de respuesta") +
  # Incorporando los valores con la libreria ggrepel
  geom_label_repel(aes(label = Freq, x = 2.2, y = cumsum(Freq) - 0.5 * Freq),
                   show.legend = FALSE, color = "white", size = 4)
```



Pregunta 5

Utilizando un algoritmo de aprendizaje no supervisado, realizad un análisis de clústering con k-means para los datos del servidor.

Resolución:

Se agrega una nueva columna llamada **length_Resource** para el n° de carácteres de la columna **Resource**. Además se realiza la conversión a binarios a través de la función **one_hot** de las columnas que tiene datos tipo factor como Method, Response_code y Protocol. En base a ello se aplica la función **kmeans()** para realizar el agrupamiento. Lo mencionado se realizó con el siguiente código.

```
# Pregunta 5
# Creando nueva columna con el número de carácteres de la columna Resource
server_data$length_Resource <-  str_length(server_data$Resource)
# Creando una nueva tabla con las columnas tipo factor
endpoints_data <- server_data[, c("Method", "Response_code", "Protocol")]
# Convirtiendo datos tipo factor a datos binarios
one_hot_data <- one_hot(as.data.table(endpoints_data), sparsifyNAs = TRUE)

# Obteniendo el kmeans
set.seed(50) # Fijando los centroides iniciales
# Agrupamiento de 7
value7_kmeans <- kmeans(one_hot_data, centers = 7)
#set.seed(150) # Fijando los centroides iniciales
# Agrupamiento de 4
```

```r
value4_kmeans <- kmeans(one_hot_data, centers = 4)
#set.seed(100)# Fijando los centroides iniciales
# Agrupamiento de 3
value9_kmeans <- kmeans(one_hot_data, centers = 9)
```

Pregunta 6

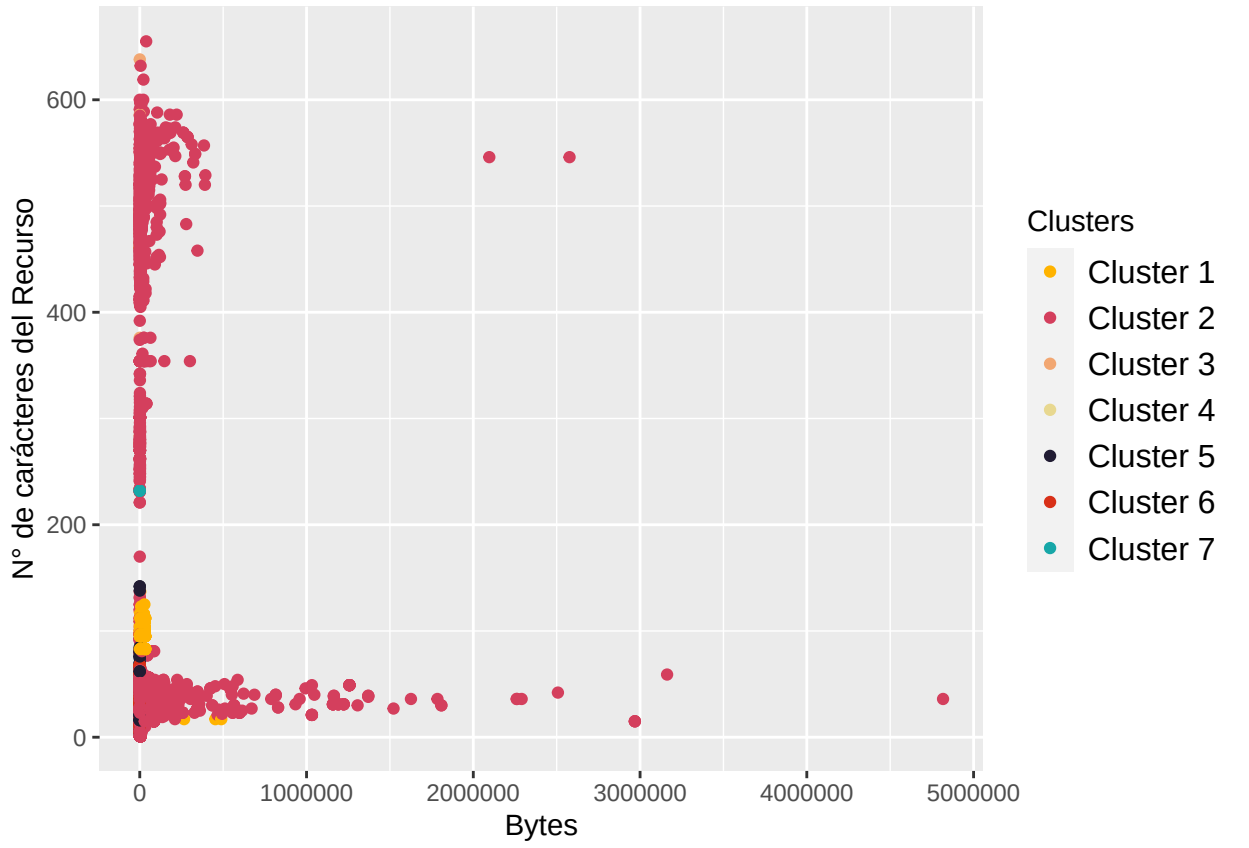
Representad visualmente en gráficos de tipo scatter plot el resultado de vuestros clústering.

Resolución:

En esta pregunta se realiza las siguientes gráficas:

- La primera gráfica es en base a los 7 centros con las columnas length_Resource y Bytes. En esta gráfica se aprecia lo siguiente:
  - En el rango de 0 bytes a lo largo de todo el eje Y del n° de carácteres del recurso se aprecia que existe mayor cantidad del cluster 2.
  - En ese mismo rango pero en el eje Y por debajo del valor 200, existe un cantidad regular del cluster 1 y una mínima cantidad de cluster 5.
  - En cambio en el rango entre 200 y 300 en el eje Y sólo existe una sola cantidad del cluster 7.
  - Finalmente, en el eje Y entre el rango 0 y 100 y a lo largo de todo el eje x se visualiza que el cluster 2 va disminuyendo.

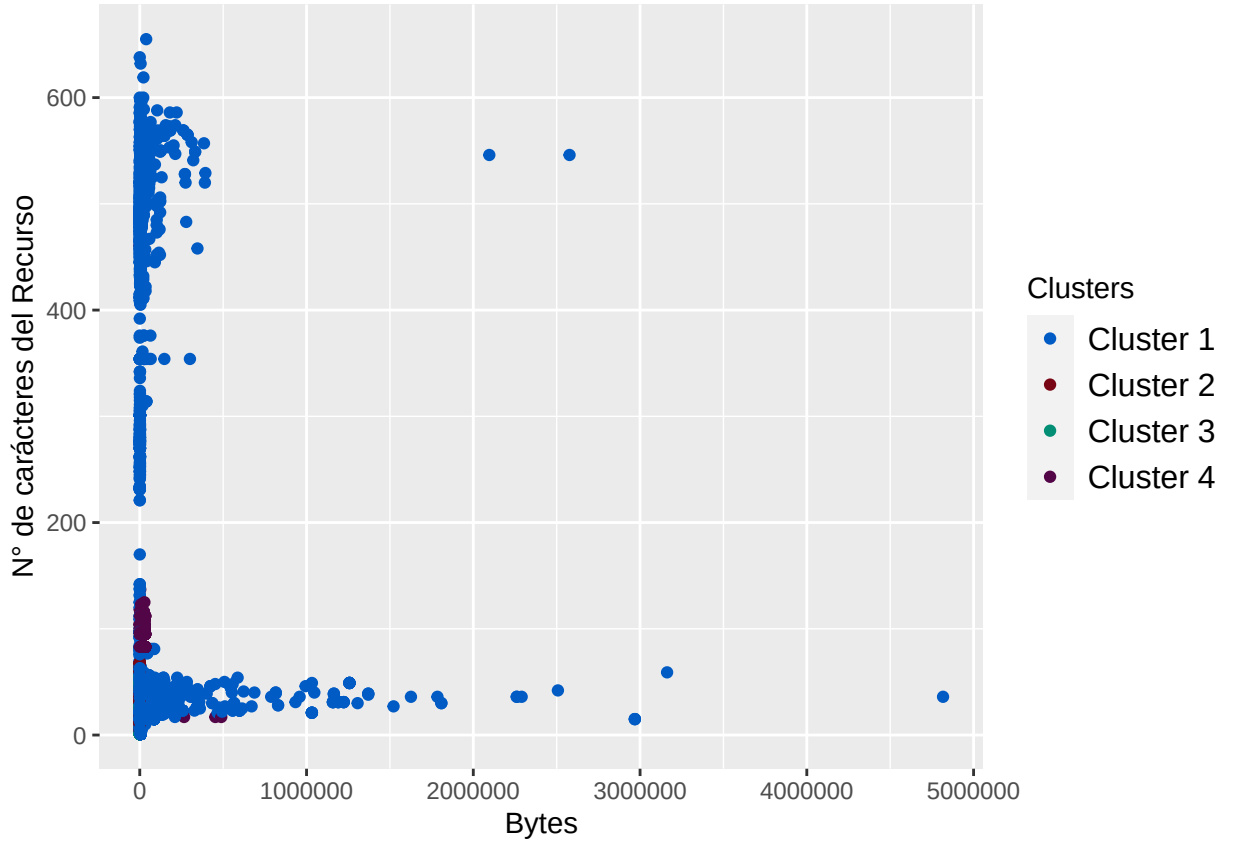
```r
#pregunta 6

# Agrupamiento de 7, se convierte en factor para la separación de cada cluster
server_data$clusters_7 <- as.factor(value7_kmeans$cluster)

# Gráfica en base a la columna bytes y length_Resource
ggplot(server_data, aes(x = Bytes, y = length_Resource, color = clusters_7)) +
  # Creación de la gráfica scatter
  geom_point() +
  # Convirtiendo la escala x en valores números
  scale_x_continuous(labels = function(x) as.integer(floor(x))) +
  labs(color = "Clusters") +
  xlab("Bytes") +
  ylab("N° de carácteres del Recurso") +
  scale_color_manual(
    values = c("#ffb300", "#d43f5d", "#f2a772", "#e8d890",
               "#211c33","#d83018","#17a7a8"),
                  # especificando el orden de los colores
                  breaks = c(1, 2, 3, 4, 5, 6, 7),
                  labels = c("Cluster 1", "Cluster 2", "Cluster 3",
                             "Cluster 4", "Cluster 5", "Cluster 6", "Cluster 7")
  ) +
  theme(legend.text=element_text(size=12))
```



- La segunda gráfica es en base a los 4 centros con las columnas length_Resource y Bytes. En esta gráfica se aprecia lo siguiente:
  - En el eje X, en el valor 0 bytes se aprecia una gran cantidad de cluster 1 a lo largo del eje Y del n° de carácteres del recurso.
  - En cambio a lo largo del eje X (Bytes) se aprecia que el cluster 1 va disminuyendo.
  - Por otro lado, en el intermedio del rango de 0 a 200 se aprecia que hay una pequeña cantidad de cluster 4 y por debajo de ese intermedio también hay una pequeña cantidad de cluster 2.

```
# Agrupamiento de 4
server_data$clusters_4 <- as.factor(value4_kmeans$cluster)

# Gráfica en base a la columna bytes y length_Resource
  ggplot(server_data, aes(x = Bytes, y = length_Resource, color = clusters_4)) +
    # Creación de la gráfica scatter
    geom_point() +
    # Convirtiendo la escala x en valores númericos
    scale_x_continuous(labels = function(x) as.integer(floor(x))) +
    labs(color = "Clusters") +
    xlab("Bytes") +
    ylab("N° de carácteres del Recurso") +
    # Definiendo los colores
    scale_color_manual(
      values = c("#005bc5", "#790614", "#028f76", "#520647"),
      breaks = c(1, 2, 3, 4),
```

```
    labels = c("Cluster 1", "Cluster 2", "Cluster 3", "Cluster 4")) +
  theme(legend.text=element_text(size=12))
```



- Esta 3ra gráfica es en base al n° de slashes del recurso (eje x) y el n° de carácteres del recurso con 9 centros. En ello se aprecia lo siguiente:
  - En todo el gráfico se aprecia que el cluster 3 es el que predomina, sin embargo en los intermedios de los rangos 2-5 y 5 -7 hay mayor cantidad de cluster 3 a lo largo del eje Y.
  - Por otro lado, existen pequeñas cantidades del Cluster 5 entre el rango de 0 a 5 y sobrepasando el valor de 7 en el eje X, cada uno de estos estan por debajo del valor 200 del eje Y.
  - Y en ese mismo valor del eje Y, existen pequeñas cantidades del cluster 7 pero en el rango de 5 a más en el eje X.
  - Finalmente existe una mínima cantidad del cluster 8 entre los rangos de 2 a 5 del eje X pero que no sobrepasan el valor de 100 en el eje Y.

```
# Agrupamiento de 9
server_data$clusters_9 <- as.factor(value9_kmeans$cluster)
# Generando una nueva grafica en base al n° de slashes
server_data$slashes_number <- str_count(server_data$Resource, "/")

# Gráfica del n° de slashes del Recurso con el n° de carácteres del Recurso
ggplot(server_data, aes(x = slashes_number, y = length_Resource, color = clusters_9)) +
  # Creación de la gráfica scatter
  geom_point() +
  # Convirtiendo la escala x en valores números
```

```r
scale_x_continuous(labels = function(x) as.integer(floor(x))) +
labs(color = "Clusters") +
xlab("N° de slashes del Recurso") +
ylab("N° de carácteres del Recurso") +
# Definiendo los colores
scale_color_manual(
  values = c("#2b818c", "#ffc6a5", "#fa6900", "#028f76",
             "#6d0839","#110303","#e4d829", "#fe59c2","#8a8780"),
  breaks = c(1, 2, 3, 4, 5, 6, 7, 8, 9),
  labels = c("Cluster 1", "Cluster 2", "Cluster 3","Cluster 4",
             "Cluster 5", "Cluster 6","Cluster 7", "Cluster 8", "Cluster 9"))
```