# Practica3_1

2023-05-07

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, first, last

##
## Attaching package: 'lubridate'

## The following objects are masked from 'package:data.table':
##
##     hour, isoweek, mday, minute, month, quarter, second, wday, week,
##     yday, year

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union

##
## Attaching package: 'mltools'

## The following object is masked from 'package:tidyr':
##
##     replace_na
```

### Pregunta 1.1

```r
#cargar datos (editar para la carga local de datos)
epa_http <- read_table("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()
## )
```

```r
#nombre columnas
names(epa_http)<- c("URL", "Time", "Tipo", "Recurso", "Protocolo", "Post", " Bytes ")
```

### Pregunta 1.2

```r
epa_merge <- data.frame(Directions = epa_http$URL, Response_code =epa_http$Post)
epa_merge_agrup <- as.data.frame(table(epa_merge))

### Filtrando los valores existentes y ordenando por codigo de respuesta
#### 200, 302, 304, 400, 403, 404, 500, 501

epa_dir <- filter(epa_merge_agrup, Freq > 0)


epa_dir <- epa_dir %>%
  arrange(Response_code)
View(epa_dir)
epa_dir_unique <- unique(epa_dir$Response_code)
##### 200, 302, 304, 400, 403, 404, 500, 501
for (code in epa_dir_unique) {
  filtered_table <- filter(epa_dir, Response_code == code)
  assign(paste0("code_", code), filtered_table)
}

### Hallando numero de usuarios según el código de respuesta
nrow(code_200)
```

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

```r
nrow(code_302)
```

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

```
nrow(code_304)
```

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

```
nrow(code_400)
```

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

```
nrow(code_403)
```

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

```
nrow(code_404)
```

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

```
nrow(code_500)
```

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

```
nrow(code_501)
```

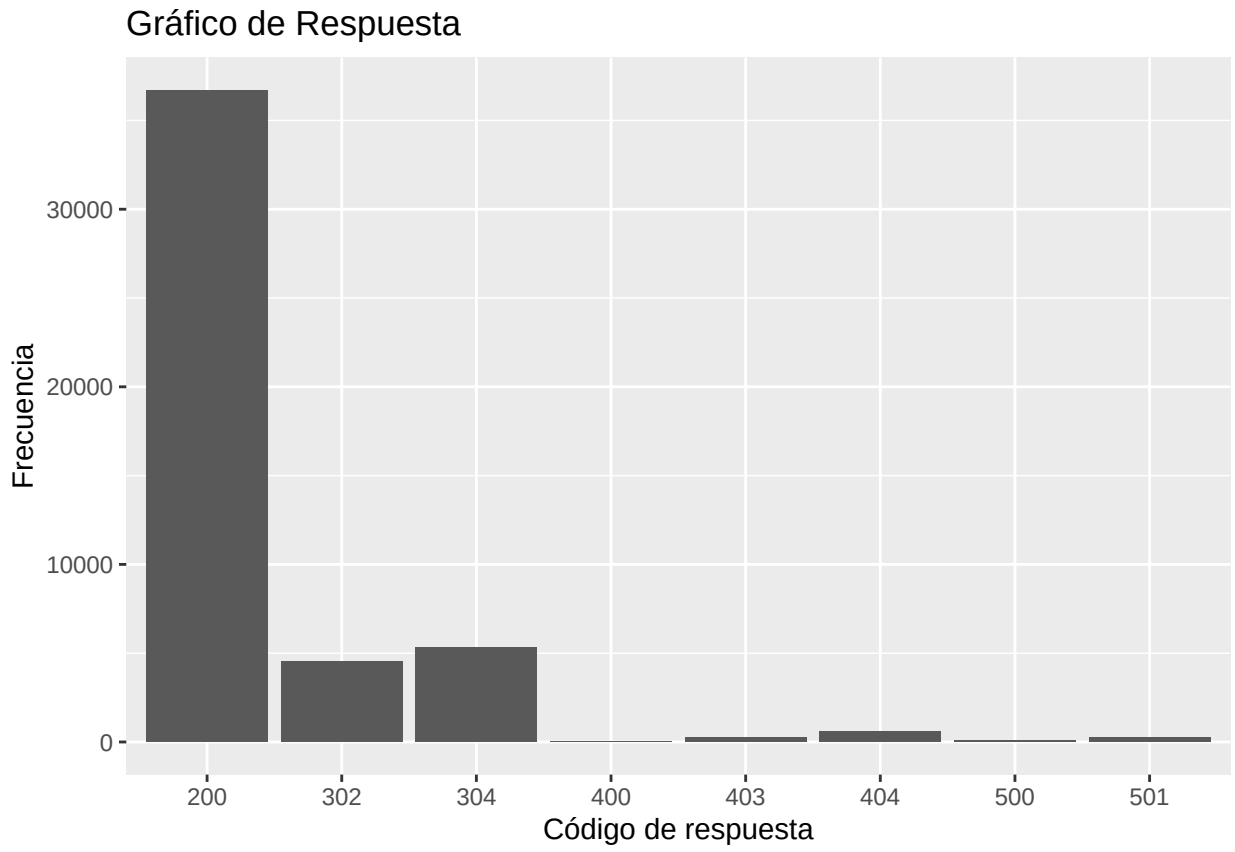
```
## [1] 11
```

### Pregunta 1.3

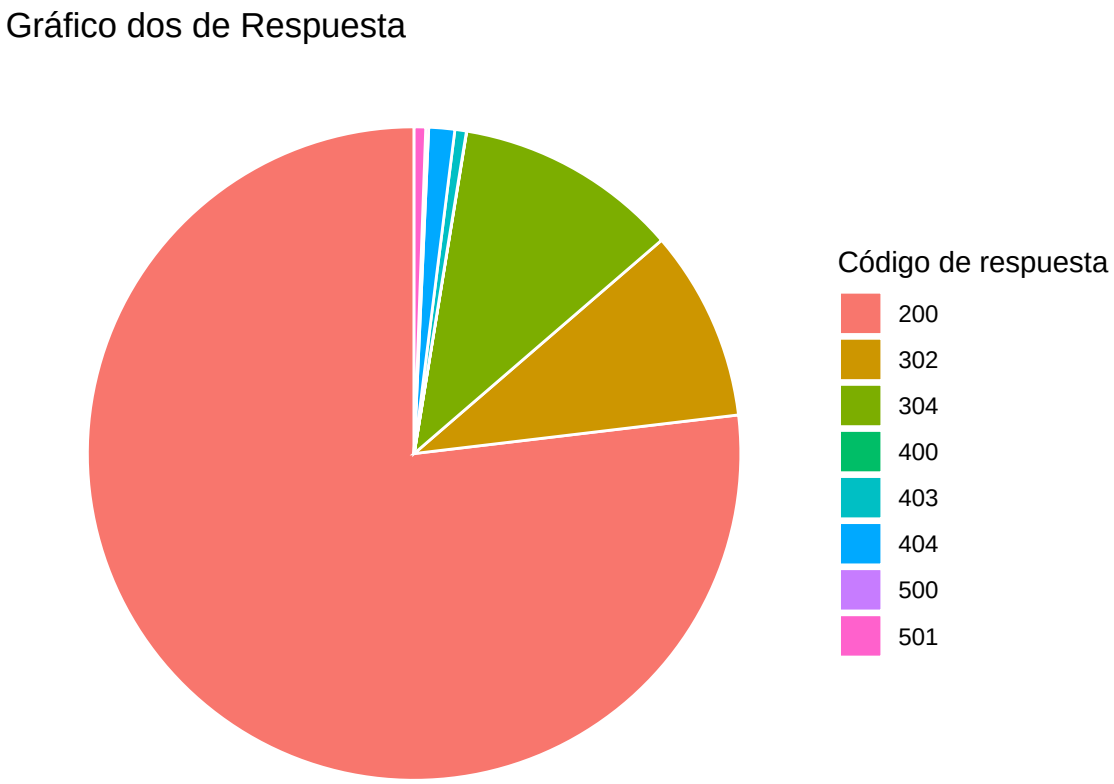
```
### contar la frecuencia de la columna http

http_freq <- table(epa_http$Tipo)
data_tipo <- data.frame(http = names(http_freq), http_freq = as.vector(http_freq))


### Hallando la frecuencia de la columna http, filtrando previamente los recursos tipo imagen
epa_http$Tipo <- as.factor(epa_http$Tipo)
epa_http$Protocolo <- as.factor(epa_http$Protocolo)
epa_http$Post <- as.factor(epa_http$Post)
epa_http$` Bytes ` <- as.numeric(epa_http$` Bytes `)


epa_data_img <- epa_http %>%
  filter(!grepl("(?i)\\.(gif|jpg|jpeg|png|bmp)$", Recurso))


epa_data_img2 <- filter(epa_http, grepl(pattern = "\\.jpg|\\.png|\\.gif|\\.ico", Recurso))


http_freqq <- table(epa_data_img2$Tipo)
data_anotherm <- data.frame(http = names(http_freqq), http_freqq = as.vector(http_freqq))
```

### Pregunta 1.4

```
epa_response <-table(epa_http$Post)
epa_response_tb <- data.frame(Response_code = names(epa_response),
                              Freq = as.vector(epa_response))



ggplot(epa_response_tb, aes(x = Response_code, y = Freq)) +
  geom_bar(stat = "identity") +
  labs(title = "Gráfico de Respuesta",
       x = "Código de respuesta",
       y = "Frecuencia")
```




```
ggplot(epa_response_tb, aes(x = "", y = Freq, fill = Response_code)) +
  geom_bar(stat = "identity", color = "white") +
  coord_polar("y", start = 0) +
  labs(title = "Gráfico dos de Respuesta",
       fill = "Código de respuesta") +
  theme_void()
```



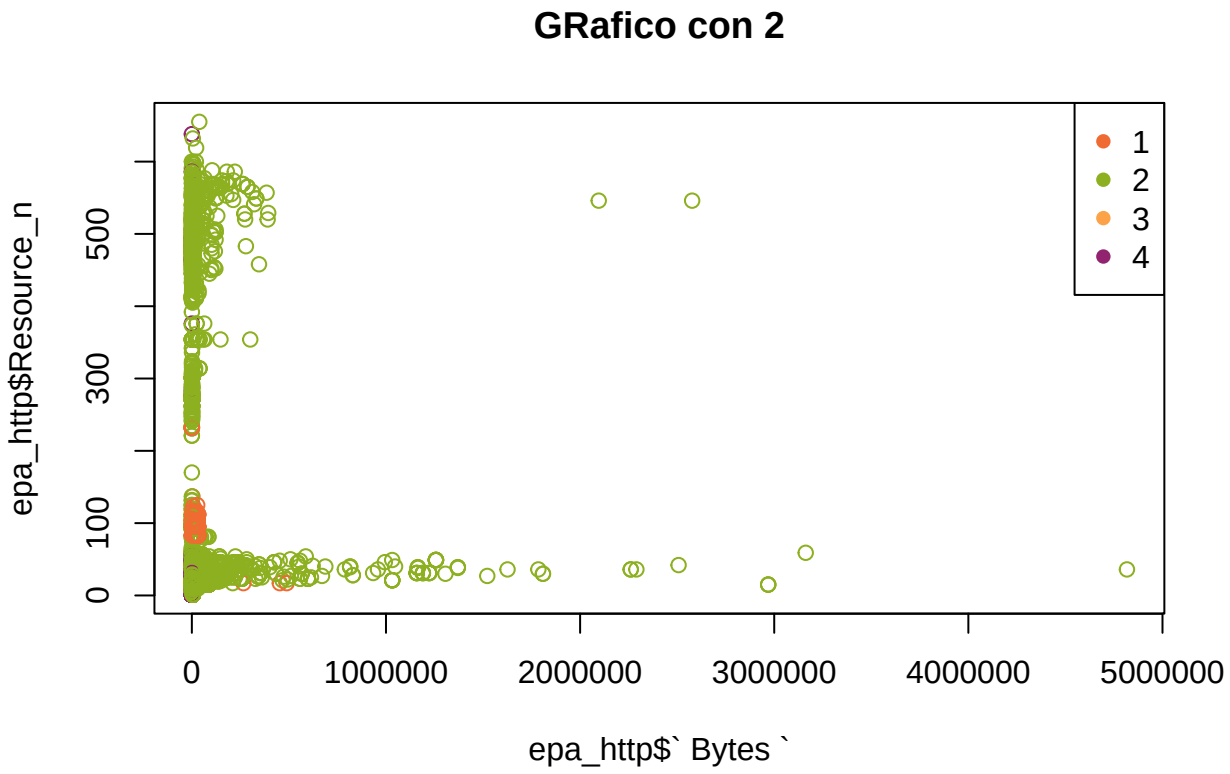

### Pregunta 1.5

```
epa_http_filter <- epa_http[, c("Tipo", "Post", "Protocolo")]

epa_http_one_hot <- one_hot(as.data.table(epa_http_filter), sparsifyNAs = TRUE)

epa_http$Resource_n <- nchar(epa_http$Recurso)

### Agrupamiento de 4 y 3, esto se puede cambiar por otros valores como 5, 6, 7, etc

results2 <- kmeans(epa_http_one_hot, centers = 4)

results3 <- kmeans(epa_http_one_hot, centers = 3)
```

### Pregunta 1.6

```
### Gráficas en base a la columna Bytes y Resource_n segun el tipo de agrupamiento
### Solo para que los resultados sean reproducibles y no aleatorios
set.seed(123)

#### Gráfica con cluster 4
#### Solo usar si quieres colores aleatorios
```

```r
colores1 <- c("#f06b32", "#8cb01f", "#fca349", "#91236f")
grap1 <- plot(x = epa_http$` Bytes `, y = epa_http$Resource_n, col = colores1[results2$cluster], main="
# solo usar esta opcion si quieren cambiar la escala de notación cientifica a número
options(scipen = 999)
# Creando leyenda
legend("topright", legend = levels(factor(results2$cluster)), col = colores1, pch = 16)
```




```r
### Gráfica con Cluster 3


colores2 <- c("#ad21fe","#00912d", "#dbfe1c")
grap2 <- plot(x = epa_http$` Bytes `, y = epa_http$Resource_n, col = colores2[results3$cluster], main="
# solo usar esta opcion si quieren cambiar la escala de notación cientifica a número
options(scipen = 999)
# Creando leyenda
legend("topright", legend = levels(factor(results3$cluster)), col = colores2, pch = 16)
```

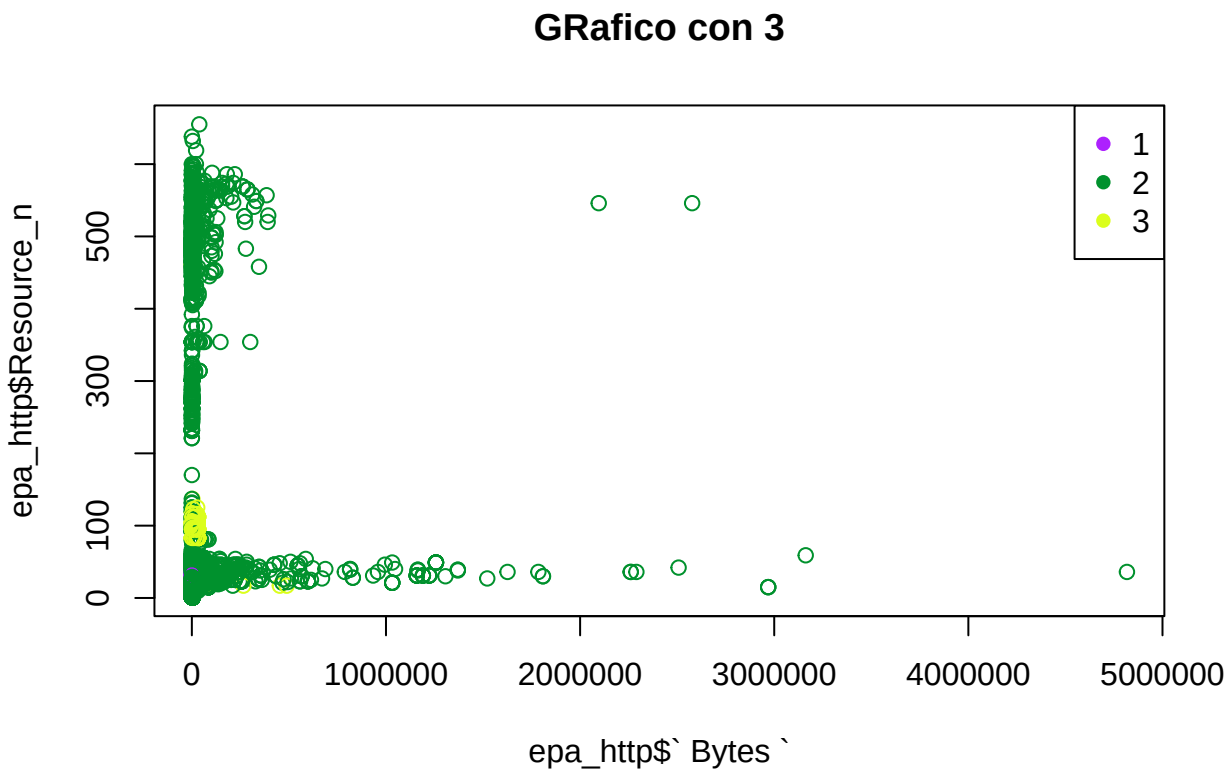


GRafico con 3
epa_http$Resource_n
0
100
300
500
0
1000000
2000000
3000000
4000000
5000000
epa_http$` Bytes `
1
2
3