Portfolio de Data Science

Manuel Alejandro Ruiz Miranda

Manuel Alejandro Ruiz Miranda

Índice

En este portfolio, voy a mostrar distintos tipos de situaciones y casos que se resuelven
mediante el uso de técnicas de Data Science a través del lenguaje R.

Los dataset que uso los obtengo de la base de datos del gobierno de la Ciudad de Buenos
Aires y de la pagina Kaggle ,los cuales están disponibles para uso publico, quiero mencionar que
ejemplificar ,a través, de casos reales tomados de entornos de trabajo es imposible por acuerdos de confidencialidad.

En el presente trabajo, se muestran distintas inferencias obtenidas en el estudio de los recorridos de bicicletas en la ciudad de buenos aires, un ejemplo de regresión lineal múltiple ,análisis geo espacial ,machine Learning ,con el algoritmo de Random Forest .

Manuel Alejandro Ruiz Miranda

Importar Datos, Librerías a Utilizar

```
library(tidyverse)
library(skimr)
library(lubridate)
library(stringr)
library(ggplot2)
library(dplyr)
library(sf)
library(pyramid)
library(leaps)
library(utils)

temp = tempfile()

download.file('https://cdn.buenosaires.gob.ar/datosabiertos/datasets/
transporte/bicicletas-publicas/recorridos-realizados-2018.zip',temp)

recorridos_2018 <- read_csv(unz(temp,'recorridos-realizados-2018.csv')) %>%
clean_names()

lista_usuarios<-map(2015:2018,~read_csv(paste0("usuarios-ecobici-",.x,".csv")))

df_usuarios <- do.call('rbind', lista_usuarios)
```

Analisis Exploratorio de Datos , EDA

Skim(recorridos_2018)

```
> skim(recorridos_2018)
── Data Summary ────────────────────────
                           Values
Name                       recorridos_2018
Number of rows             2619968
Number of columns          16
_______________________
Column type frequency:
  character                8
  numeric                  8
_______________________
Group variables            None

── Variable type: character ────────────────────────────────────────────────────
  skim_variable             n_missing complete_rate min max empty n_unique
1 genero_usuario                    0         1       1   1     0        3
2 fecha_origen_recorrido            0         1      19  19     0  2435266
3 nombre_estacion_origen            0         1       3  35     0      198
4 domicilio_estacion_origen     30224         0.988   8  78     0      195
5 duracion_recorrido            43723         0.983  25  25     0    10495
6 fecha_destino_recorrido       43723         0.983  19  19     0  2394923
7 nombre_estacion_destino           0         1       3  35     0      198
8 domicilio_estacion_destino    30486         0.988   8  78     0      195
  whitespace
── Variable type: numeric ──────────────────────────────────────────────────────
skim_variable         n_missing complete_rate     mean         sd      p0
periodo                       0         1       2018          0       2018
id_usuario                    0         1     328122.     186442.       8
id_estacion_origen        30224         0.988    94.9       58.6        1
long_estacion_origen      30224         0.988   -58.4        0.0219   -58.5
lat_estacion_origen       30224         0.988   -34.6        0.0151   -34.6
id_estacion_destino       30486         0.988    95.3       58.4        1
long_estacion_destino     30486         0.988   -58.4        0.0219   -58.5
lat_estacion_destino      30486         0.988   -34.6        0.0151   -34.6
     p25       p50       p75      p100 hist
  2018      2018      2018      2018
157791    353194    491737    672449
    40        91       146       200
   -58.4     -58.4     -58.4     -58.4
   -34.6     -34.6     -34.6     -34.6
    41        92       146       200
   -58.4     -58.4     -58.4     -58.4
   -34.6     -34.6     -34.6     -34.6
```

Analisis Exploratorio de Datos , EDA

De lo anterior se desprende que:

•El dataset se compone de 16 variables, con 2619968 observaciones.

•Hay aproximadamente un 3.79% de valores faltantes NA(n_missing).

•Los mismos no son consecutivos, representan una cantidad minoritaria, por estas razones los voy a eliminar del Dataframe.

•Tenemos una representatividad superior al 90 % ,96.21%

•Hay datos que tienen tipos erróneos ,no corresponden al tipo de variable ,por ejemplo, las fechas ,genero usuario , son tipo carácter.

•Se deben aplicar cambios de tipo a las variables necesarias, para asegurar una coherencia en el análisis

•La gran mayoría de los datos se distribuyen entre el percentil p50 y el p100

recorridos_2018<-recorridos_2018%>%
filter(complete.cases(.))

,con esta instrucción excluyo los valores Na del Dataframe

De manera análoga podemos concluir que tomando en cuenta la salida del comando anterior ,skim, ya no tenemos valores perdidos o incompletos, los datos conservan la distribución .

```
> summary(recorridos_2018)
   periodo       id_usuario     genero_usuario     fecha_origen_recorrido
Min.   :2018   Min.   :     8   Length:2619968     Length:2619968
1st Qu.:2018   1st Qu.:157791   Class :character   Class :character
Median :2018   Median :353194   Mode  :character   Mode  :character
Mean   :2018   Mean   :328122
3rd Qu.:2018   3rd Qu.:491737
Max.   :2018   Max.   :672449

id_estacion_origen nombre_estacion_origen long_estacion_origen
Min.   :  1.00     Length:2619968         Min.   :-58.45
1st Qu.: 40.00     Class :character       1st Qu.:-58.42
Median : 91.00     Mode  :character       Median :-58.40
Mean   : 94.94                            Mean   :-58.40
3rd Qu.:146.00                            3rd Qu.:-58.38
Max.   :200.00                            Max.   :-58.36
NA's   :30224                             NA's   :30224
lat_estacion_origen domicilio_estacion_origen duracion_recorrido
Min.   :-34.64      Length:2619968            Length:2619968
1st Qu.:-34.61      Class :character          Class :character
Median :-34.60      Mode  :character          Mode  :character
Mean   :-34.60
3rd Qu.:-34.59
Max.   :-34.57
NA's   :30224
> skim(recorridos_2018)
── Data Summary ────────────────────────
                           Values
Name                       recorridos_2018
Number of rows             2520545
Number of columns          16
_____________________
Column type frequency:
  character                8
  numeric                  8
_____________________
Group variables            None

── Variable type: character ────────────────────────────────────────────────────
  skim_variable             n_missing complete_rate min max empty n_unique whitespace
1 genero_usuario                    0             1   1   1     0        3          0
2 fecha_origen_recorrido            0             1  19  19     0  2348307          0
3 nombre_estacion_origen            0             1   3  35     0      196          0
4 domicilio_estacion_origen         0             1   8  78     0      195          0
5 duracion_recorrido                0             1  25  25     0    10495          0
6 fecha_destino_recorrido           0             1  19  19     0  2346615          0
7 nombre_estacion_destino           0             1   3  35     0      196          0
8 domicilio_estacion_destino        0             1   8  78     0      195          0
```

Analisis Exploratorio de Datos , EDA

Skim(df_usuarios)

```
> skim(df_usuarios)
── Data Summary ────────────────────────
                           Values
Name                       df_usuarios
Number of rows             194282
Number of columns          5
___________________________
Column type frequency:
  character                2
  difftime                 1
  numeric                  2
___________________________
Group variables            None

── Variable type: character ────────────────────────────────────────────────────
  skim_variable  n_missing complete_rate min max empty n_unique whitespace
1 genero_usuario       130         0.999   1   1     0        3          0
2 fecha_alta             0         1       8   8     0     1367          0

── Variable type: difftime ─────────────────────────────────────────────────────
  skim_variable n_missing complete_rate min    max        median       n_unique
1 hora_alta             0             1 0 secs 86398 secs 43151.5 secs    52138

── Variable type: numeric ──────────────────────────────────────────────────────
  skim_variable n_missing complete_rate     mean       sd   p0     p25     p50     p75
1 id_usuario            0             1 373375.  148988.  7682 242736. 369190. 494190.
2 edad_usuario          0             1     34.4     12.0    0     26      31      41
    p100 hist
1 672973 ▃▇▃▁
2    160 ▇▆▁▁
```

De lo anterior se desprende que:

•El dataset se compone de 5 variables, con 194282 observaciones cada una.

•Hay aproximadamente un 0.07% de valores faltantes NA(n_missing) ,que corresponden a la columna de genero_usuario.

•Los mismos no son consecutivos, representan una cantidad minoritaria, por estas razones los voy a eliminar del Dataframe.

•Tenemos una representatividad superior al 90 % ,99.93%.

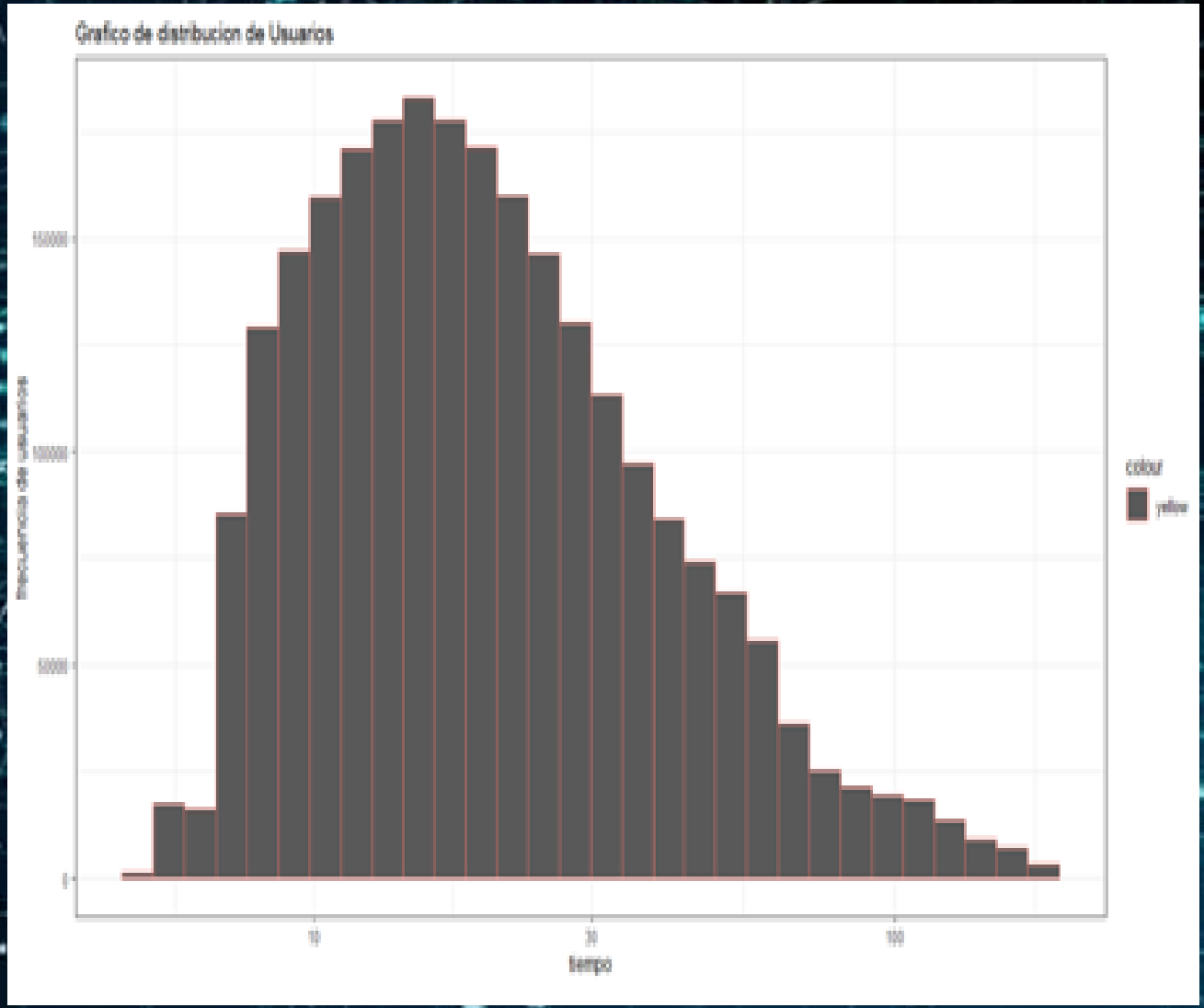
•Hay datos que tienen tipos erróneos ,no corresponden al tipo de variable ,por ejemplo, genero_usuario y fecha_alta son tipo carácter.

•Se deben aplicar cambios de tipo a las variables necesarias, para asegurar una coherencia en el análisis.
•La gran mayoría de los datos se distribuyen entre el percentil p50 y el p75

Visuales univariadas de variables de interés ,nos permiten ver su comportamiento en el tiempo :

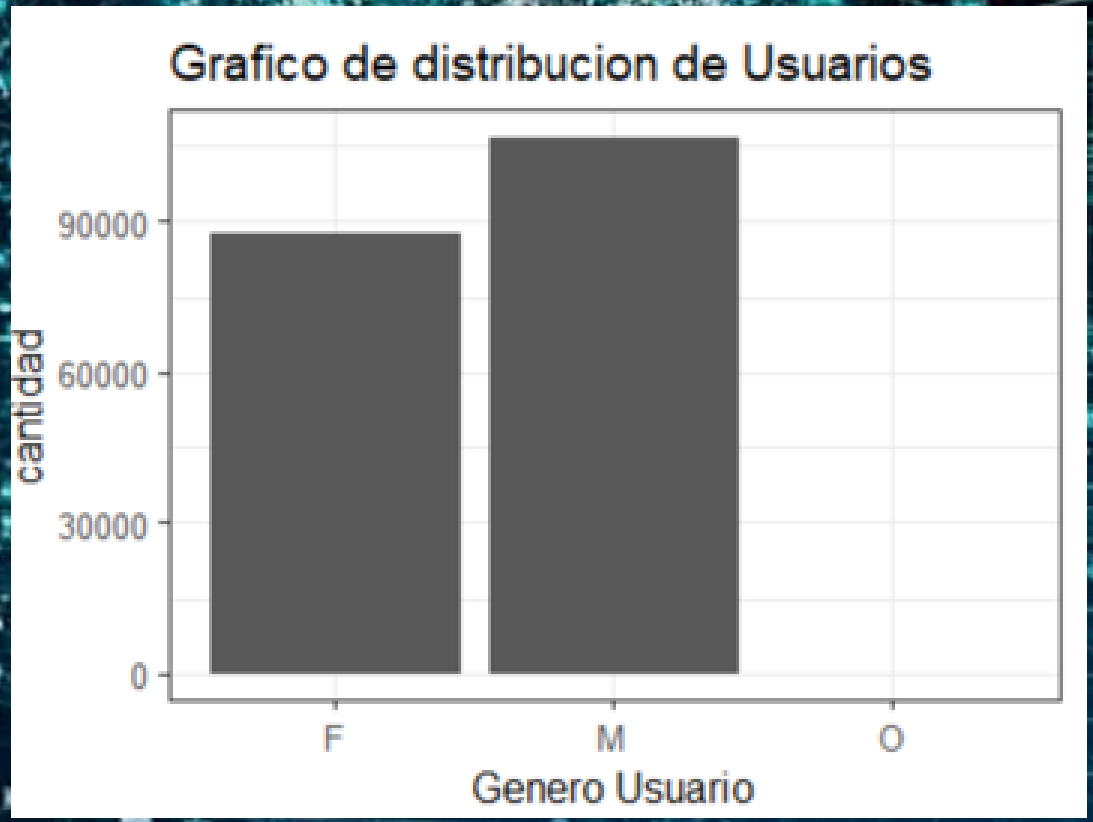
Aplico conversiones de tipo a las variables de fecha que estaban en formato carácter, esto me permite calcular el tiempo que demoró en llegar un usuario desde la estación de partida a la de llegada.

```
df_usuarios <- df_usuarios %>% filter(complete.cases(.))
recorridos_2018 <- recorridos_2018 %>%
mutate(fecha_origen_recorrido = ymd_hms(fecha_origen_recorrido),
fecha_destino_recorrido = ymd_hms(fecha_destino_recorrido))

recorridos_2018 <- recorridos_2018 %>%
mutate(tiempo_recorrido = round(difftime(fecha_destino_recorrido,
fecha_origen_recorrido),digits = 3))

ggplot(data = recorridos_2018,
aes(x = as.numeric(tiempo_recorrido),binwidth = 60,col = 'yellow'))+
geom_histogram()+
scale_x_log10()+
theme_bw()+
labs(title = "Grafico de distribucion de Usuarios")+
ylab("tiempo recorrido")+
xlab("frecuencia de usuarios")
```



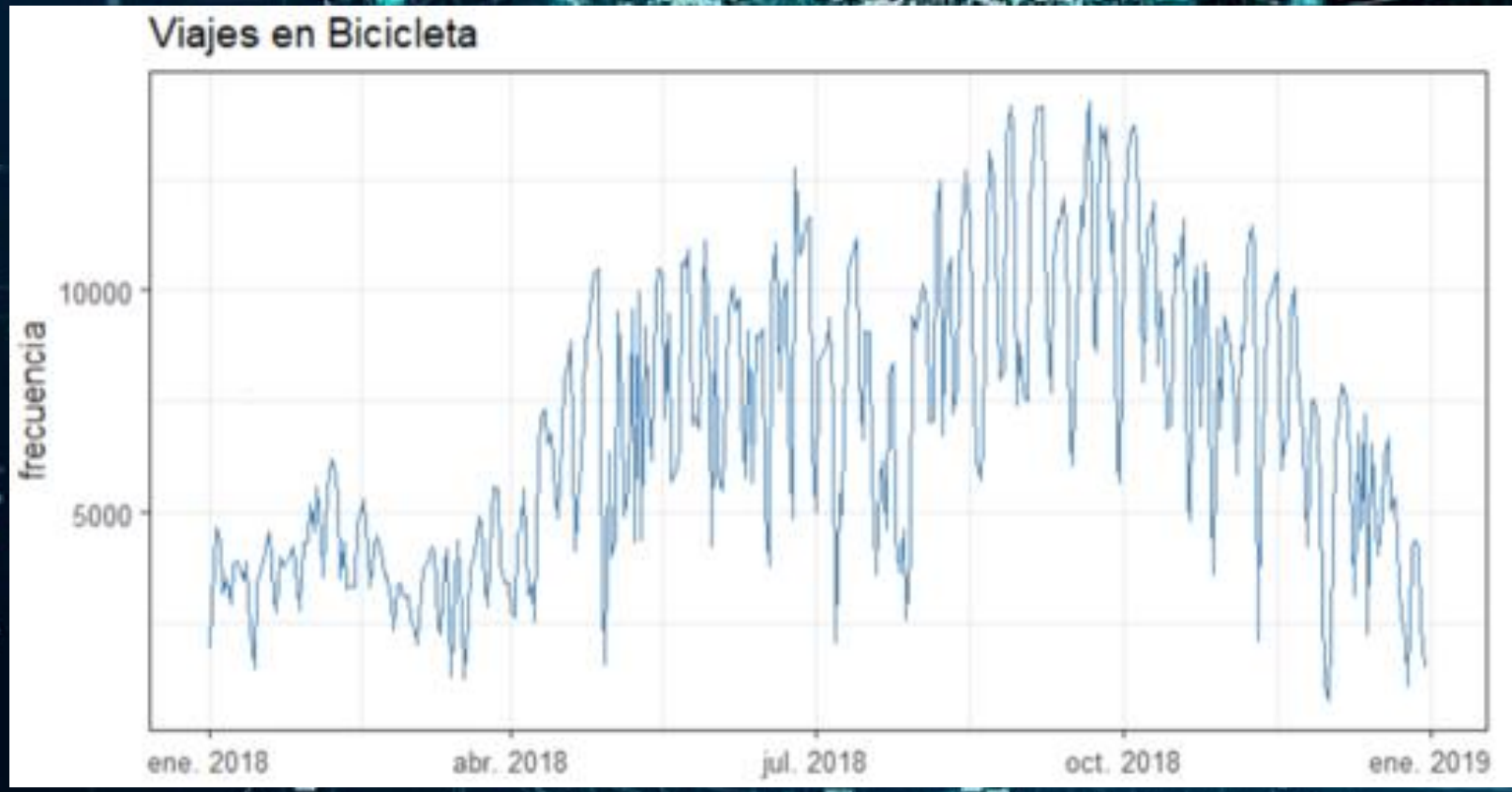
También podemos representar la distribución de Usuarios por genero :

```
ggplot(data = df_usuarios) +
geom_bar(aes(x =
as.factor(genero_usuario))) +
theme_bw()+
labs(title = "Grafico de distribucion
de Usuarios")+
ylab("cantidad")+
xlab("Genero Usuario")
```



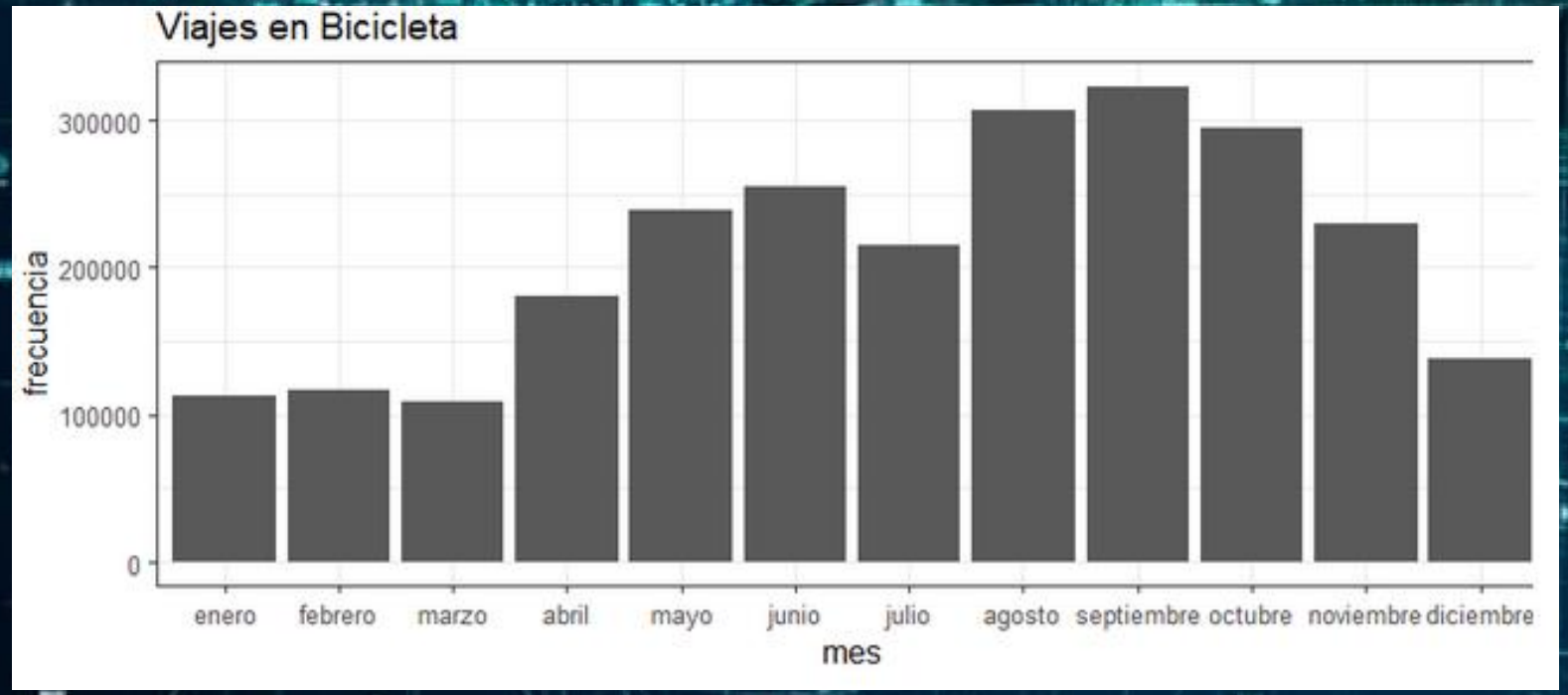
Vamos a estimar, el total de recorridos generados cada día y mostrar la serie de tiempo asociada a el :

```
recorr_por_dia <- recorridos_2018 %>%
count(dia = (as.Date(fecha_origen_recorrido)))
recorr_por_dia
ggplot(data = recorr_por_dia,aes(x = dia , y = n))+
geom_line(color = 'steelblue')+
theme_bw()+
labs(title = "Viajes en Bicicleta")+
ylab("frecuencia")+
xlab("dia de semana")
```



Si observamos la frecuencia de viajes agrupados por mes los podemos ver en un gráfico de barras o Barplot ,ya que hay una única observación mensual

```
recorr_por_mes <- recorridos_2018 %>%
count(mes = month(fecha_origen_recorrido,label = TRUE,abbr = F))
recorr_por_mes
ggplot(data = recorr_por_mes,aes(x = mes,y = n))+
geom_bar(stat = 'identity')+
scale_y_continuous(labels=function(n){format(n, scientific = FALSE)}) +
labs(title = "Viajes en Bicicleta")+
ylab("frecuencia")+
xlab("mes")+
theme_bw()
```



Podemos ver las tablas de datos correspondientes a los gráficos anteriores y sus respectivos estadísticos

```
> recorr_por_dia
          dia    n
1  2018-01-01 1913
2  2018-01-02 3861
3  2018-01-03 4653
4  2018-01-04 4351
5  2018-01-05 3178
6  2018-01-06 3496
7  2018-01-07 2918
8  2018-01-08 3840
9  2018-01-09 3894
10 2018-01-10 3736
11 2018-01-11 3481
12 2018-01-12 3925
13 2018-01-13 2364
14 2018-01-14 1493
15 2018-01-15 3444
16 2018-01-16 3725
17 2018-01-17 3887
18 2018-01-18 4438
19 2018-01-19 4601
20 2018-01-20 3109
21 2018-01-21 2771
```

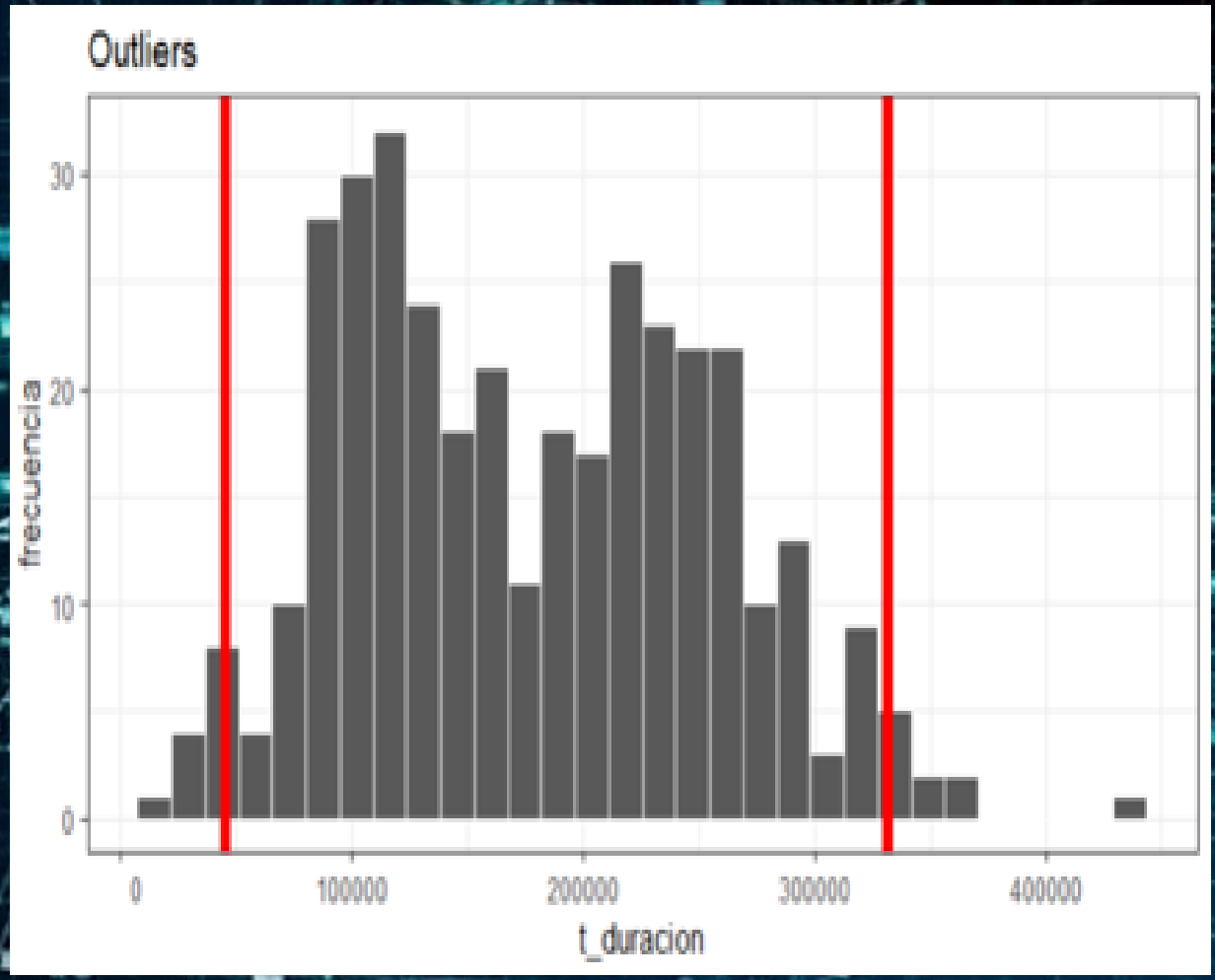
```
> recorr_por_mes
          mes      n
1       enero 113091
2     febrero 117137
3       marzo 108377
4       abril 181306
5        mayo 238620
6       junio 254981
7       julio 214901
8      agosto 306964
9  septiembre 323157
10     octubre 295029
11  noviembre 229316
12  diciembre 137666
```

```
> summary(recorr_por_mes)
      mes          n
 enero  :1   Min.   :108377
 febrero:1   1st Qu.:132534
 marzo  :1   Median :222109
 abril  :1   Mean   :210045
 mayo   :1   3rd Qu.:264993
 junio  :1   Max.   :323157
 (Other):6
```

```
> summary(recorr_por_dia)
      dia                 n
 Min.   :2018-01-01   Min.   :  757
 1st Qu.:2018-04-01   1st Qu.: 4196
 Median :2018-07-01   Median : 6564
 Mean   :2018-07-01   Mean   : 6925
 3rd Qu.:2018-09-30   3rd Qu.: 9535
 Max.   :2018-12-30   Max.   :14239
```

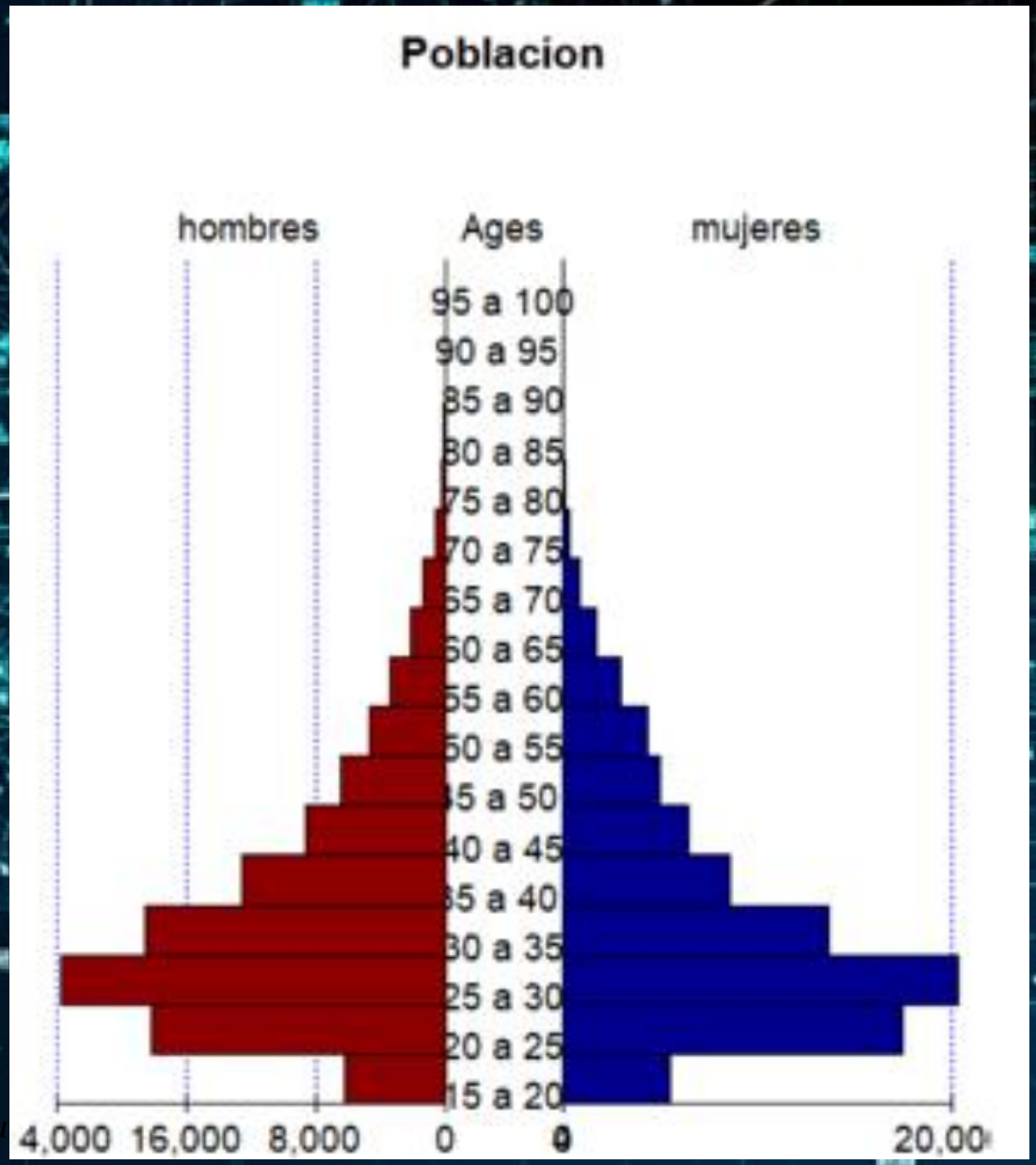
A partir del criterio de rango Inter cuartil, podemos identificar los outliers u observaciones extremas, para una confianza del 95%,del vector de duración de tiempo y representarlos en un histograma.

```
ggplot(recor_X_dia,aes(x = t_duracion))+
geom_histogram(col = 'white')+
geom_vline(xintercept =
quantile(recor_X_dia$t_duracion, .025), col = 'red', size = 2) +
scale_x_continuous(labels=function(n){format(n, scientific =
FALSE)})+
geom_vline(xintercept =
quantile(recor_X_dia$t_duracion, .975), col = 'red', size = 2)
```



Con los datos de usuarios, vamos a discretizar la variable edad, armando rangos etarios que vayan de 15 a 100 años en intervalos de 5 años y a representarlos en una pirámide poblacional para conocer como se distribuyen :

```
df_usuarios_segmentado <- df_usuarios %>%
select(-c(id_usuario,fecha_alta,hora_alta)) %>%
filter(between(edad_usuario,15,100)) %>%
mutate(edad_cada_5 = cut(df_usuarios_segmentado$edad_usuario,
breaks = 5*(3:20),include.lowest = TRUE))

df_usuarios_segmentado %>%
count(edad_cada_5,genero_usuario,name = "hombres") %>%
group_by(edad_cada_5) %>%
filter(genero_usuario == "M") -> Hombres %>% as.data.frame()

df_usuarios_segmentado %>%
count(edad_cada_5,genero_usuario,name = "mujeres") %>%
group_by(edad_cada_5) %>%
filter(genero_usuario == "F") -> Mujeres %>% as.data.frame()

py.edades <- c("15 a 20","20 a 25","25 a 30","30 a 35","35 a 40 ","40 a 45",
"45 a 50 ","50 a 55","55 a 60","60 a 65","65 a 70","70 a 75",
"75 a 80","80 a 85","85 a 90","90 a 95 ","95 a 100")

py.hombres <- Hombres$hombres
py.mujeres <- Mujeres$mujeres
names(py.edades) <- py.edades
pyramids(Left = py.hombres, Llab = "hombres",Right = py.mujeres,
Lcol = "Dark Red",
Rcol = "Dark Blue",
Rlab = "mujeres",
Center = py.edades,
Laxis = seq(0,24000 ,len = 4),
Raxis = c(0,20000,len = 4),AxisFM="d",AxisBM=",",
main = "Poblacion")
```

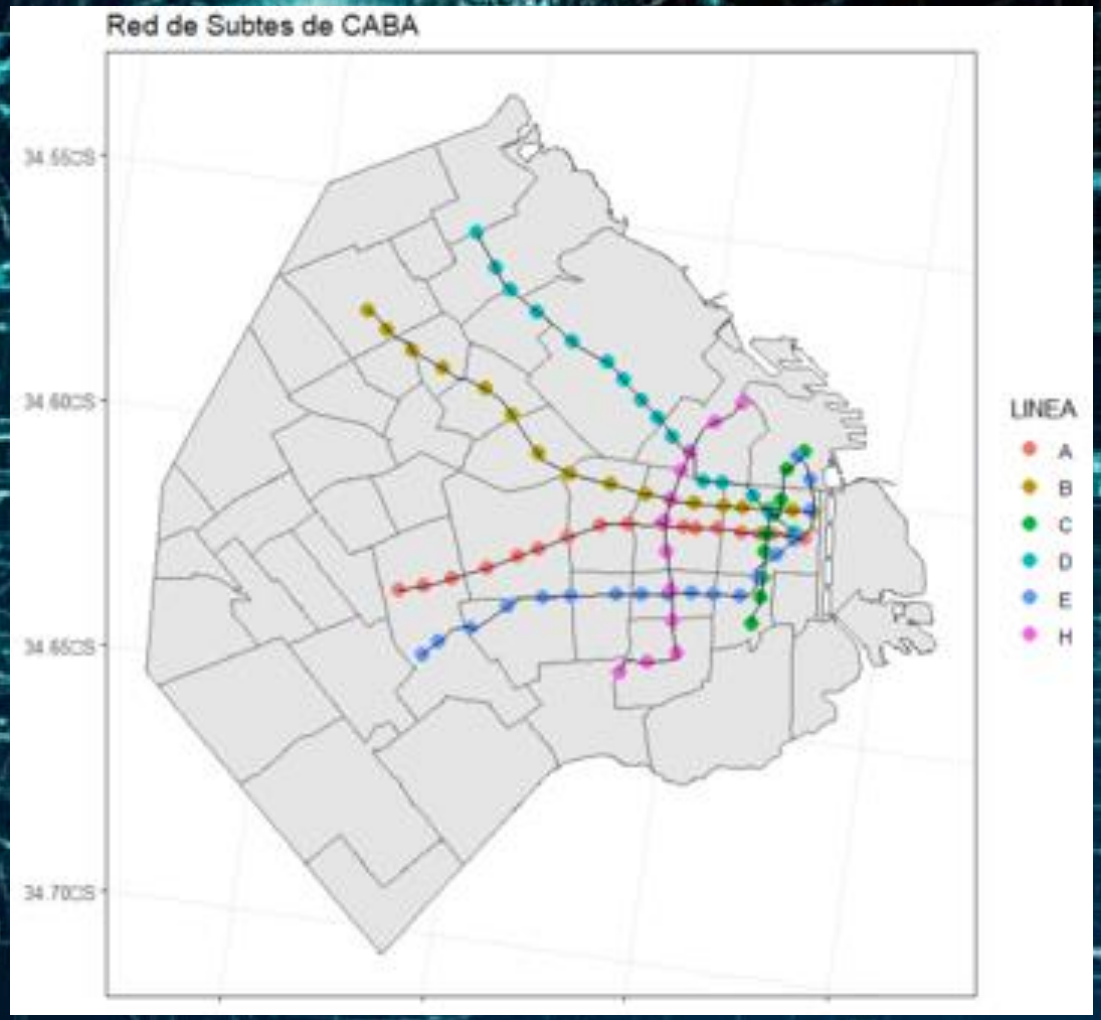


Análisis Geoespacial

Vamos a mostrar, mediante un cruce de datos, los barrios de capital federal ,las estaciones de subte de cada uno de ellos.

Se aplican transformaciones a los datasets originales para que todos tengan el mismo CRS ,de esta forma, se puede operar con ellos.

```
subte_est <- st_read("subte-estaciones/estaciones-de-subte.shp")%>%
st_transform(5343)

subte_lineas <- st_read("subte-lineas/lineas-subte.shp")%>%
st_transform(5343)

ferias <- st_read('ferias/ferias.shp')%>% st_transform(5343)
comunas <- st_read('comunas/comunas_wgs84.shp')%>%
st_transform(5343)

barrios <- st_read("barrios/barrios_badata_wgs84.shp")%>%
st_transform(5343)


ggplot() +
geom_sf(data = barrios,lwd = 3) +
geom_sf(data = subte_est, aes(col = LINEA),cex = 3) +
geom_sf(data = subte_lineas,lwd = 4) +
ggtitle('Red de Subtes de CABA')+
theme_bw()
```

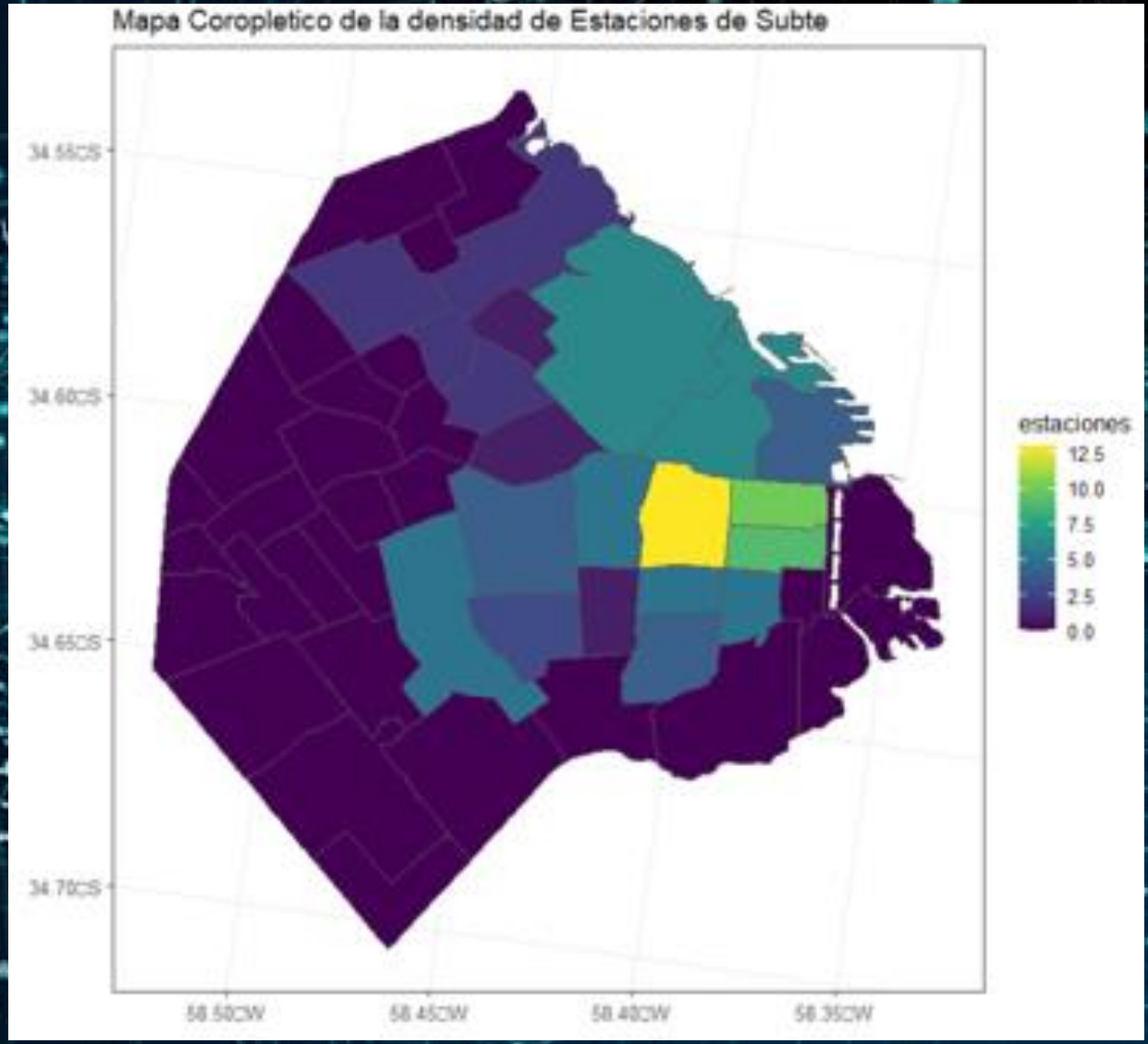


Manuel Alejandro Ruiz Miranda

Análisis Geoespacial

A través de un join espacial, vamos a determinar la cantidad de estaciones en cada barrio, y a representar esto, a través, de un mapa coroplético donde el color representa la densidad de estaciones.

```
estac_por_barrio <- barrios %>%
st_join(subte_est) %>%
mutate(tiene_estacion = !is.na(ESTACION)) %>%
group_by(BARRIO) %>%
summarise(estaciones = sum(tiene_estacion,na.rm = T))

ggplot()+
geom_sf(data = estac_por_barrio, aes(fill = estaciones)) +
theme_bw()+
scale_fill_viridis_c()
```



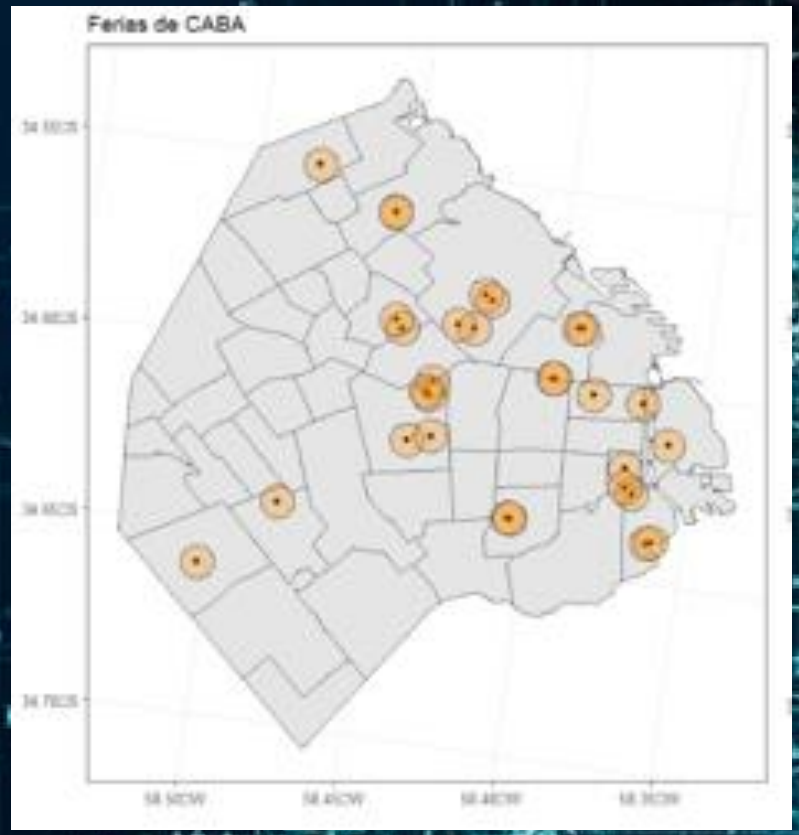
Análisis Geoespacial

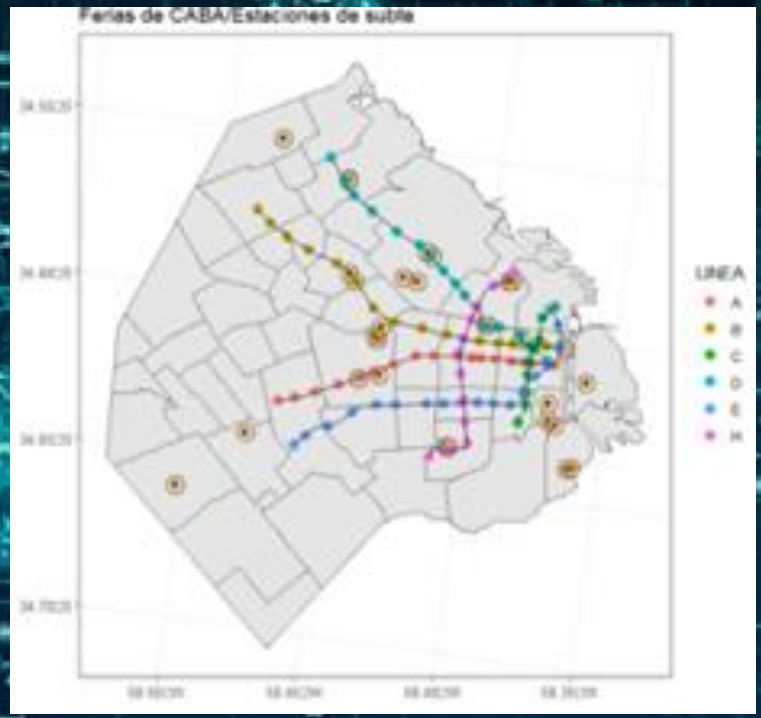
Vamos a representar las distintas ferias que se establecen sobre la ciudad de Buenos Aires, estableciendo un centro para cada una de ellas y también un radio de 500 metros alrededor ,con el propósito de ver su cercanía.

```
centroides_ferias <- ferias %>% st_centroid()
buffer_ferias <- centroides_ferias %>%
st_buffer(dist = 500)

ggplot()+
geom_sf(data = barrios)+
geom_sf(data = centroides_ferias)+
geom_sf(data = buffer_ferias,fill = 'yellow',alpha = .3)+
theme_bw()+
ggtitle('Ferias de CABA')
```

En el gráfico de la derecha podemos apreciar la distribución de las ferias con respecto a las estaciones de subte.

```
ggplot()+
geom_sf(data = barrios)+
geom_sf(data = barrios,lwd = 3) +
geom_sf(data = centroides_ferias)+
geom_sf(data = buffer_ferias,fill = 'dark orange',alpha =
.3)+
geom_sf(data = subte_est, aes(col = LINEA),cex = 3) +
geom_sf(data = subte_lineas,lwd = 4)+
theme_bw()+
ggtitle('Ferias de CABA/Estaciones de Subte')
```





Manuel Alejandro Ruiz Miranda

Regresión Lineal Múltiple

Este conjunto de datos tiene datos recopilados de Nueva York, California y Florida sobre 50 empresas emergentes "17 en cada estado". Las variables utilizadas en el conjunto de datos son Beneficio, Gasto en I+D, Gasto en administración y Gasto en marketing

Librerias

```
library(tidyverse)
library(janitor)
library(Metrics)
library(leaps)
library(skimr)
library(corrplot)
library(car)
library(lmtest)
```

Filter

	r_d_spend	administration	marketing_spend	state	profit
1	165349.20	136897.80	471784.10	New York	192261.83
2	162597.70	151377.59	443898.53	California	191792.06
3	153441.51	101145.55	407934.54	Florida	191050.39
4	144372.41	118671.85	383199.62	New York	182901.99
5	142107.34	91391.77	366168.42	Florida	166187.94
6	131876.90	99814.71	362861.36	New York	156991.12
7	134615.46	147198.87	127716.82	California	156122.51
8	130298.13	145530.06	323876.68	Florida	155752.60

Manuel Alejandro Ruiz Miranda

Regresión Lineal Múltiple

Cargamos el dataset ,estableciendo el directorio de trabajo y estandarizamos los nombres de las columnas con la funcion clean_names del paquete janitor.

temp50=tempfile()
download.file("https://www.kaggle.com/datasets/abhishek14398/50startups
/download?datasetVersionNumber=1",temp)

X50_Startups < read_csv(temp,"50Startups.csv") %>% clean_names()

Realizamos un breve EDA

Tenemos un dataframe compuesto por 5 columnas ,4 de tipo numérico y una es tipo carácter que serian las variables, cada columna tiene 50 elementos u observaciones ,vamos a intentar predecir la variable profit, que es numérica .
La ecuación representativa para la regresión lineal múltiple con predictores a estimar esta dada por:
y = β1r_d_spend +β2administration + β3marketing_spend + β4state

***dataset tomado de** Kaggle*

```
> str(X50_Startups)
spc_tbl_ [50 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
 $ r_d_spend      : num [1:50] 165349 162598 153442 144372 142107 ...
 $ administration : num [1:50] 136898 151378 101146 118672 91392 ...
 $ marketing_spend: num [1:50] 471784 443899 407935 383200 366168 ...
 $ state          : chr [1:50] "New York" "California" "Florida" "New York"
 $ profit         : num [1:50] 192262 191792 191050 182902 166188 ...
```

```
> skim(X50_Startups)
── Data Summary ────────────────────────
                           Values
Name                       X50_Startups
Number of rows             50
Number of columns          5
──────────────────────────
Column type frequency:
  character                1
  numeric                  4
──────────────────────────
Group variables            None

── Variable type: character ────────────────────────────────────────────────────
  skim_variable n_missing complete_rate min max empty n_unique whitespace
1 state                 0             1   7  10     0        3          0
── Variable type: numeric ──────────────────────────────────────────────────────
  skim_variable   n_missing complete_rate    mean      sd     p0     p25     p50
1 r_d_spend               0             1  73722.  45902.     0   39936.  73051.
2 administration          0             1 121345.  28018. 51283. 103731. 122700.
3 marketing_spend         0             1 211025. 122290.     0 129300. 212716.
4 profit                  0             1 112013.  40306. 14681.  90139. 107978.
      p75    p100 hist
1 101603. 165349.
2 144842. 182646.
3 299469. 471784.
4 139766. 192262.
```

Regresión Lineal Múltiple

Vamos a generar nuestro modelo de Regresión Lineal Múltiple, con el propósito de poder hacer predicciones ,el primer paso es plantear el Modelo Train-Test, para lo cual seleccionamos aleatoriamente el 75% de las observaciones del dataset para entrenamiento y el 25% restante para el conjunto de testeo.

```
selected_rows <- sample(1:nrow(X50_Startups), floor(.75*nrow(X50_Startups)))

train_model_1 <- X50_Startups[selected_rows,] %>% filter(complete.cases(.))

test_model_1 <- X50_Startups[-selected_rows,] %>% filter(complete.cases(.))
```

A continuación aplicamos la estrategia de Best subset selection BSS,que nos permite seleccionar los predictores mas adecuados para la eficiencia de nuestro modelo ,que intenta predecir la ganancia de las empresas.

```
bss_selection <- regsubsets(profit~. ,data = train_model_1 , method = 'exhaustive',nvmax = 8)

summary(bss_selection)
summary(bss_selection)$adjr2
summary(bss_selection)$bic
summary(bss_selection)$adjr2 %>% which.max()
summary(bss_selection)$adjr2 %>% max()
summary(bss_selection)$bic %>% which.min()
summary(bss_selection)$bic%>% min()
```

***Análisis* de Relación entre variables**

correl <- cor(X50_Startups %>% select(-state))
corrplot(correl,method = "number",type = "full")

Las correlaciones mas altas que se presentan en relación a profit ,que es la variable que intentaremos predecir son:
r_d_spend ---------- 97 %
marketing_spend -------- 75 %
por lo tanto lo mas probable es que debamos descartar alguna de las dos.

Manuel Alejan

Regresión Lineal Múltiple

Best subset selection,nos permite seleccionar en base a distintos criterios cuales serian nuestros mejores predictores para optimizar el modelo en cuestión.

En este caso aplicando el criterio de r2 ajustado seria el modelo 2 y aplicando el criterio de bayes tendríamos el modelo 1,vamos a evaluar cada uno de ellos para ver cual se ajusta mejor a la realidad.

```
> bss_selection <- regsubsets(profit~. ,data = train_model_1 ,
+                             method = 'exhaustive',nvmax = 8)
> summary(bss_selection)
Subset selection object
Call: regsubsets.formula(profit ~ ., data = train_model_1, method = "exhaustive",
    nvmax = 8)
5 Variables  (and intercept)
                 Forced in Forced out
r_d_spend            FALSE      FALSE
administration       FALSE      FALSE
marketing_spend      FALSE      FALSE
stateFlorida         FALSE      FALSE
stateNew York        FALSE      FALSE
1 subsets of each size up to 5
Selection Algorithm: exhaustive
         r_d_spend administration marketing_spend stateFlorida stateNew York
1  ( 1 ) "*"       " "            " "             " "          " "
2  ( 1 ) "*"       " "            "*"             " "          " "
3  ( 1 ) "*"       "*"            "*"             " "          " "
4  ( 1 ) "*"       "*"            "*"             " "          "*"
5  ( 1 ) "*"       "*"            "*"             "*"          "*"
> summary(bss_selection)$adjr2
[1] 0.9350644 0.9362173 0.9344323 0.9325363 0.9303633
> summary(bss_selection)$bic
[1] -94.99176 -93.11621 -89.58864 -86.06150 -82.45231
> summary(bss_selection)$adjr2 %>% which.max()
[1] 2
> summary(bss_selection)$adjr2 %>% max()
[1] 0.9362173
> summary(bss_selection)$bic %>% which.min()
[1] 1
> summary(bss_selection)$bic%>% min()
[1] -94.99176
```

Regresión Lineal Múltiple

Generamos el primer modelo de regresión lineal múltiple, sin hacer ningún tipo de ingeniería de atributos ,esto es con todas las variables del dataset original, para poder contrastarlo con el modelo seleccionado con BSS.

model_0 <- lm(profit~.,data = X50_Startups)
summary(model_0)

Implementamos el modelo con los predictores según el Best Subset Selection

selected_rows <- sample(1:nrow(X50_Startups), floor(.75*nrow(X50_Startups)))
train_model<- X50_Startups[selected_rows,] %>% filter(complete.cases(.))

test_mode <- X50_Startups[-selected_rows,] %>% filter(complete.cases(.))

#Adjr2

model_1 <- lm(profit~r_d_spend+marketing_spend,data = train_model)
summary(model_1)

#BIC
model_2 <- lm(profit~r_d_spend,data = train_model)
summary(model_2)

Vemos las distintas salidas del comando summary para cada modelo y analizamos cual es el mas eficiente

```
> summary(model_0)

Call:
lm(formula = profit ~ ., data = X50_Startups)

Residuals:
   Min     1Q Median     3Q    Max
-33504  -4736     90   6672  17338

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     5.013e+04  6.885e+03   7.281 4.44e-09 ***
r_d_spend       8.060e-01  4.641e-02  17.369  < 2e-16 ***
administration -2.700e-02  5.223e-02  -0.517    0.608
marketing_spend 2.698e-02  1.714e-02   1.574    0.123
stateFlorida    1.988e+02  3.371e+03   0.059    0.953
stateNew York  -4.189e+01  3.256e+03  -0.013    0.990
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9439 on 44 degrees of freedom
Multiple R-squared:  0.9508,    Adjusted R-squared:  0.9452
F-statistic: 169.9 on 5 and 44 DF,  p-value: < 2.2e-16
```

Regresión Lineal Múltiple

Tomando en cuenta la información que nos entrega cada modelo ,vemos que, en todos la variable realmente significativa es r_d_spend ,lo cual concuerda con lo expresado por el criterio Bayesiano en el feature selection (BSS),por lo tanto el modelo a estimar va a ser rmodel_2,el cual explica en esta primer instancia al 93.5% de las observaciones

```
> summary(model_1)

Call:
lm(formula = profit ~ r_d_spend + marketing_spend, data = train_model)

Residuals:
   Min     1Q Median     3Q    Max
-33981  -4612   -672   7068  16663

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.742e+04  3.491e+03  13.585 2.69e-15 ***
r_d_spend      7.902e-01  5.344e-02  14.785 2.27e-16 ***
marketing_spend 2.749e-02  1.957e-02   1.405    0.169
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10080 on 34 degrees of freedom
Multiple R-squared:  0.9397,    Adjusted R-squared:  0.9362
F-statistic:   265 on 2 and 34 DF,  p-value: < 2.2e-16
```

```
> summary(model_2)

Call:
lm(formula = profit ~ r_d_spend, data = train_model)

Residuals:
   Min     1Q Median     3Q    Max
-34738  -4005  -1570   7242  17077

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 4.942e+04  3.232e+03   15.29   <2e-16 ***
r_d_spend   8.447e-01  3.727e-02   22.67   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 10220 on 35 degrees of freedom
Multiple R-squared:  0.9362,    Adjusted R-squared:  0.9344
F-statistic: 513.8 on 1 and 35 DF,  p-value: < 2.2e-16
```

Regresión Lineal Múltiple

Validación de condiciones, vamos a aplicar diferentes criterios para evaluar si nuestro modelo,model_2,cumple con las condiciones para poder realizar una regresión lineal múltiple sobre los datos y a partir de este modelo obtener predicciones.

Verificamos, Normalidad, Independencia y homocedasticidad.

shapiro.test(resid(model_2))

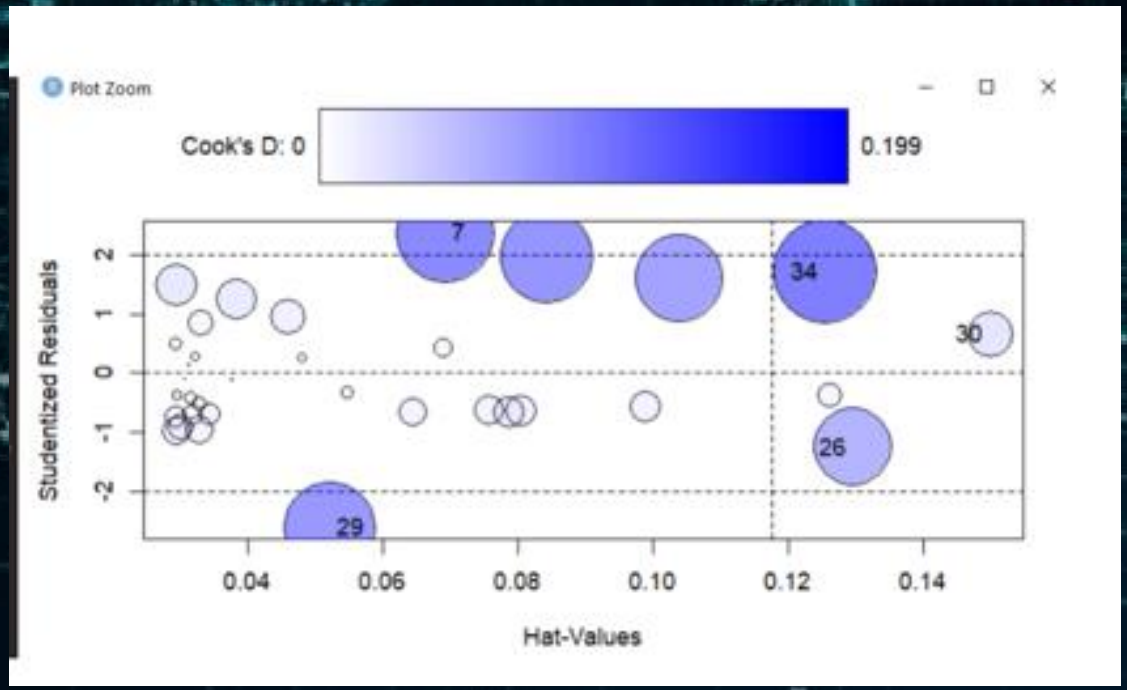
dwtest(model_2)

bptest(model_2)

```
> shapiro.test(resid(model_2))

        Shapiro-Wilk normality test

data:  resid(model_2)
W = 0.91499, p-value = 0.007903

> dwtest(model_2)

        Durbin-Watson test

data:  model_2
DW = 1.6769, p-value = 0.161
alternative hypothesis: true autocorrelation is greater than 0

> bptest(model_2)

        studentized Breusch-Pagan test

data:  model_2
BP = 2.3138, df = 1, p-value = 0.1282
```

Como podemos ver en esta primera aproximación el p-value de shapiro es menor a 0.05 ,lo cual estaría incumpliendo con la condición de que los residuos están distribuidos de forma homogénea, el siguiente grafico, muestra la situación.

i***nfluencePlot(model_2)***



Regresión Lineal Múltiple

Determinamos los outliers que sean mayores a 3 veces la media de la distancia de cook y filtrarlos para acotar nuestras observaciones a las mas representativas de la muestra.

X50_Startups <- X50_Startups %>% mutate(d_cook = cooks.distance(model_0)) %>%
filter(d_cook<3*mean(d_cook,na.rm = TRUE))

train_model_1<- X50_Startups[selected_rows,] %>%
filter(complete.cases(.))

test_mode_1 <- X50_Startups[-selected_rows,] %>%
filter(complete.cases(.))

aplicando nuevamente la regresión lineal y evaluando su desempeño ,vemos que se cumple la condición de Normalidad impuesta en el test de shapiro.

model_2 <- lm(profit~r_d_spend,data = train_model_1)

summary(model_2)

shapiro.test(resid(model_2))
dwtest(model_2)
bptest(model_2)

Como conclusión podemos decir que después de haber acotado la dispersión de residuos ,nuestro modelo, cumple con las condiciones para una regresión lineal múltiple ya que los p-value superan el valor de 0.05,nuestro modelo obtuvo un incremento en su performance de aproximadamente un 2.5 %,performando ahora en un 97 %.

```
> shapiro.test(resid(model_2))

        Shapiro-Wilk normality test

data:  resid(model_2)
W = 0.94824, p-value = 0.1088

> dwtest(model_2)

        Durbin-Watson test

data:  model_2
DW = 2.5224, p-value = 0.9391
alternative hypothesis: true autocorrelation is greater than 0

> bptest(model_2)

        studentized Breusch-Pagan test

data:  model_2
BP = 0.077124, df = 1, p-value = 0.7812
```

```
> summary(model_2)

Call:
lm(formula = profit ~ r_d_spend, data = train_model_1)

Residuals:
   Min     1Q Median     3Q    Max
-17117  -4658  -2529   4234  15619

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.107e+04  2.639e+03   19.36   <2e-16 ***
r_d_spend   8.379e-01  2.992e-02   28.00   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7329 on 32 degrees of freedom
Multiple R-squared:  0.9608,    Adjusted R-squared:  0.9596
F-statistic: 784.3 on 1 and 32 DF,  p-value: < 2.2e-16
```

Manuel Alejandro Ruiz Miranda

Regresión Lineal Múltiple

Vamos a generar un vector de predicciones para nuestro modelo y a evaluar su rmse.

pred_l <- predict(model_1,test_mode_1)

rmse(test_mode_1$profit,pred_l)

summary(X50_Startups$profit)

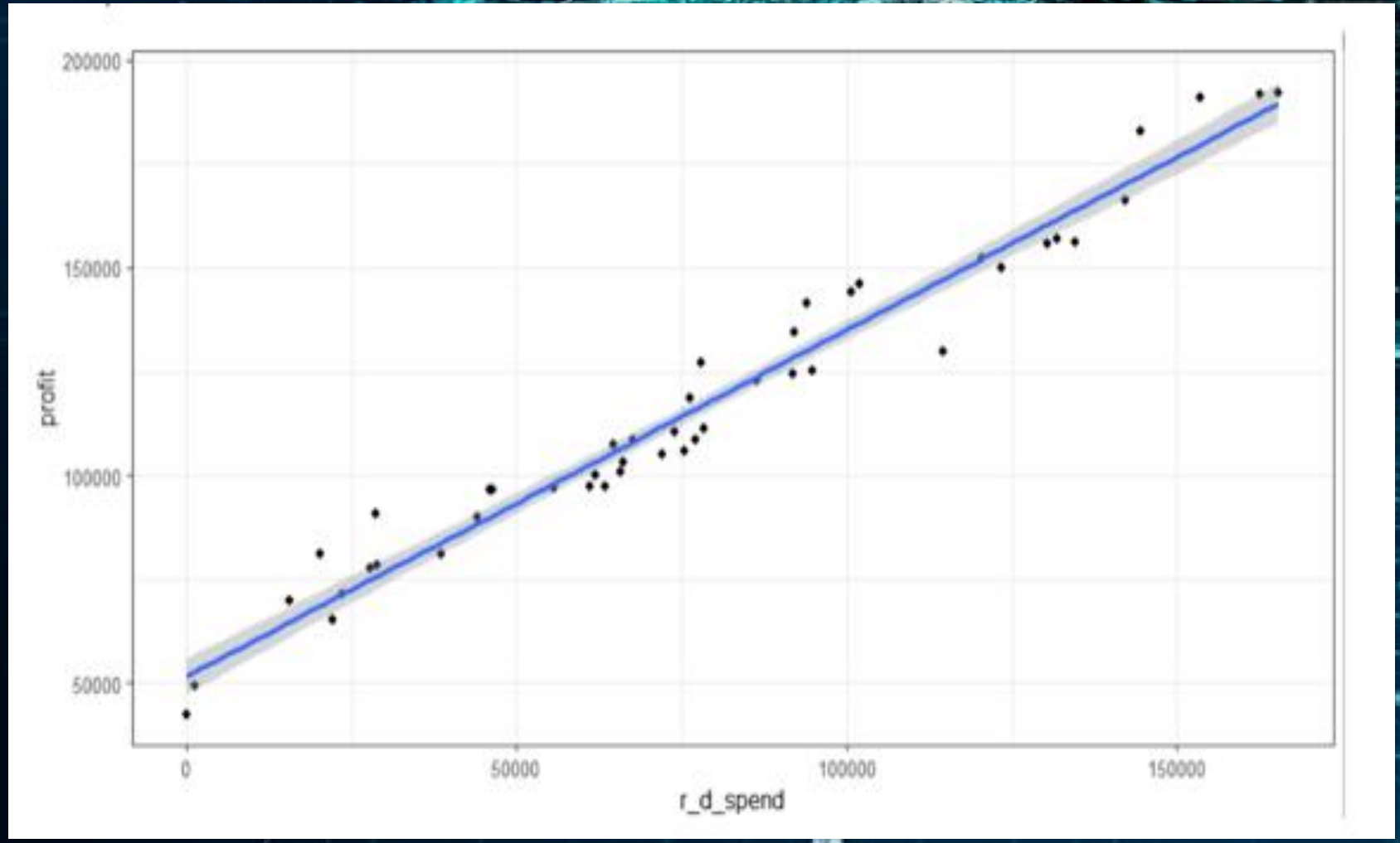
sd(X50_Startups$profit)

Como podemos observar nuestro modelo de predicciones nos entrega 12 valores en relación al train-test ,con un rmse acorde a la dispersión de los datos ,el valor del mismo es bastante aceptable.

```
> pred_l <- predict(model_2,test_mode_1)
> pred_l
       1        2        3        4        5        6        7        8
189621.43 163868.93 152076.89 136466.86 129722.14 114191.34 103018.15 102299.13
       9       10       11       12
 89973.03  74443.68  70880.84  64064.17
> rmse(test_mode_1$profit,pred_l)
[1] 6420.445
> summary(X50_Startups$profit)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  42560   96538  108643  116364  143591  192262
> sd(X50_Startups$profit)
[1] 36754.94
```

Regresión Lineal Múltiple

```
ggplot(X50_Startups,aes(r_d_spend,profit))+
geom_point()+
geom_smooth(method = "lm",level = 0.97)+
theme_bw()
```



Machine learning

Random Forest

Con dispositivos como Jawbone Up , Nike FuelBand y Fitbit , ahora es posible recopilar una gran cantidad de datos sobre la actividad personal de forma relativamente económica. Este tipo de dispositivos son parte del auto movimiento cuantificado: un grupo de entusiastas que se toman medidas sobre sí mismos regularmente, para mejorar su salud, con el propósito de encontrar patrones en su comportamiento o porque son fanáticos de la tecnología.

Una cosa que las personas hacen con regularidad es cuantificar qué tanto de una actividad en particular hacen, pero rara vez cuantifican qué tan bien lo hacen.

El objetivo de este proyecto es predecir la forma en que realizaron el ejercicio. Esta es la variable "clase" en el conjunto de entrenamiento.

Manuel Alejandro Ruiz Miranda

Extracción y carga del Dataframe

Los datos que vamos a utilizar fueron obtenidos de http://groupware.les.inf.puc-rio.br/har.

```
library(tidyverse)
library(caret)
library(ggplot2)
library(utils)
library(skimr)
(scipen=999)


temp = tempfile()
download. file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv',temp)
temp1 = tempfile()
download. file('https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv',temp1)
df_fit <- read.csv(temp)
test_fit <- read.csv(temp1)
```

Al efectuar un EDA a los datasets, encuentro valores NA en varias de sus columnas, valores n_missing, los cuales no nos sirven para nuestro modelo de machine learning, también observo que son demasiados para imputarlos mediante la técnica de Knn, por lo cual los voy a excluir.

Conjunto de Prueba:

> str(df_fit)

```
> skim(df_fit)
── Data Summary ────────────────────────
                           Values
Name                       df_fit
Number of rows             19622
Number of columns          160
_____________________
Column type frequency:
  character                37
  numeric                  123
_____________________
Group variables            None
```

── Variable type: character ──

	skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
1	user_name	0	1	5	8	0	6	0
2	cvtd_timestamp	0	1	16	16	0	20	0
3	new_window	0	1	2	3	0	2	0
4	kurtosis_roll_belt	0	1	0	9	19216	397	0
5	kurtosis_picth_belt	0	1	0	9	19216	317	0
6	kurtosis_yaw_belt	0	1	0	7	19216	2	0
7	skewness_roll_belt	0	1	0	9	19216	395	0
8	skewness_roll_belt.1	0	1	0	9	19216	338	0
9	skewness_yaw_belt	0	1	0	7	19216	2	0
10	max_yaw_belt	0	1	0	7	19216	68	0
11	min_yaw_belt	0	1	0	7	19216	68	0
12	amplitude_yaw_belt	0	1	0	7	19216	4	0
13	kurtosis_roll_arm	0	1	0	8	19216	330	0
14	kurtosis_picth_arm	0	1	0	8	19216	328	0
15	kurtosis_yaw_arm	0	1	0	8	19216	395	0
16	skewness_roll_arm	0	1	0	8	19216	331	0
17	skewness_pitch_arm	0	1	0	8	19216	328	0
18	skewness_yaw_arm	0	1	0	8	19216	395	0
19	kurtosis_roll_dumbbell	0	1	0	7	19216	398	0
20	kurtosis_picth_dumbbell	0	1	0	7	19216	401	0
21	kurtosis_yaw_dumbbell	0	1	0	7	19216	2	0
22	skewness_roll_dumbbell	0	1	0	7	19216	401	0
23	skewness_pitch_dumbbell	0	1	0	7	19216	402	0
24	skewness_yaw_dumbbell	0	1	0	7	19216	2	0
25	max_yaw_dumbbell	0	1	0	7	19216	73	0
26	min_yaw_dumbbell	0	1	0	7	19216	73	0
27	amplitude_yaw_dumbbell	0	1	0	7	19216	3	0
28	kurtosis_roll_forearm	0	1	0	7	19216	322	0
29	kurtosis_picth_forearm	0	1	0	7	19216	323	0
30	kurtosis_yaw_forearm	0	1	0	7	19216	2	0
31	skewness_roll_forearm	0	1	0	7	19216	323	0
32	skewness_pitch_forearm	0	1	0	7	19216	319	0
33	skewness_yaw_forearm	0	1	0	7	19216	2	0
34	max_yaw_forearm	0	1	0	7	19216	45	0
35	min_yaw_forearm	0	1	0	7	19216	45	0
36	amplitude_yaw_forearm	0	1	0	7	19216	3	0
37	classe	0	1	1	1	0	5	0

Machine learning

Random Forest

── Variable type: numeric ──

	skim_variable	n_missing	complete_rate	mean	sd
1	x	0	1	9.81e+3	5665.
2	raw_timestamp_part_1	0	1	1.32e+9	204928.
3	raw_timestamp_part_2	0	1	5.01e+5	288223.
4	num_window	0	1	4.31e+2	248.
5	roll_belt	0	1	6.44e+1	62.8
6	pitch_belt	0	1	3.05e-1	22.4
7	yaw_belt	0	1	-1.12e+1	95.2
8	total_accel_belt	0	1	1.13e+1	7.74
9	max_roll_belt	19216	0.0207	-6.67e+0	94.6
10	max_picth_belt	19216	0.0207	1.29e+1	8.01
11	min_roll_belt	19216	0.0207	-1.04e+1	93.6
12	min_pitch_belt	19216	0.0207	1.08e+1	7.47
13	amplitude_roll_belt	19216	0.0207	3.77e+0	25.3
14	amplitude_pitch_belt	19216	0.0207	2.17e+0	2.36
15	var_total_accel_belt	19216	0.0207	9.26e-1	2.22
16	avg_roll_belt	19216	0.0207	6.81e+1	63.1
17	stddev_roll_belt	19216	0.0207	1.34e+0	2.44
18	var_roll_belt	19216	0.0207	7.70e+0	23.2
19	avg_pitch_belt	19216	0.0207	5.20e-1	22.4
20	stddev_pitch_belt	19216	0.0207	6.03e-1	0.639
21	var_pitch_belt	19216	0.0207	7.66e-1	1.76
22	avg_yaw_belt	19216	0.0207	-8.83e+0	93.5
23	stddev_yaw_belt	19216	0.0207	1.34e+0	10.3
24	var_yaw_belt	19216	0.0207	1.07e+2	1656.
25	gyros_belt_x	0	1	-5.59e-3	0.207
26	gyros_belt_y	0	1	3.96e-2	0.0782
27	gyros_belt_z	0	1	-1.31e-1	0.241
28	accel_belt_x	0	1	-5.59e+0	29.6
29	accel_belt_y	0	1	3.02e+1	28.6
30	accel_belt_z	0	1	-7.26e+1	100.
31	magnet_belt_x	0	1	5.56e+1	64.2
32	magnet_belt_y	0	1	5.94e+2	35.7
33	magnet_belt_z	0	1	-3.45e+2	65.2
34	roll_arm	0	1	1.78e+1	72.7
35	pitch_arm	0	1	-4.61e+0	30.7
36	yaw_arm	0	1	-6.19e-1	71.4
37	total_accel_arm	0	1	2.55e+1	10.5
38	var_accel_arm	19216	0.0207	5.32e+1	54.0
39	avg_roll_arm	19216	0.0207	1.27e+1	68.6
40	stddev_roll_arm	19216	0.0207	1.12e+1	17.1
41	var_roll_arm	19216	0.0207	4.17e+2	2007.
42	avg_pitch_arm	19216	0.0207	-4.90e+0	26.8
43	stddev_pitch_arm	19216	0.0207	1.04e+1	9.40
44	var_pitch_arm	19216	0.0207	1.96e+2	293.
45	avg_yaw_arm	19216	0.0207	2.36e+0	61.3
46	stddev_yaw_arm	19216	0.0207	2.23e+1	23.7
47	var_yaw_arm	19216	0.0207	1.06e+3	2722.
48	gyros_arm_x	0	1	4.28e-2	1.99
49	gyros_arm_y	0	1	-2.57e-1	0.851
50	gyros_arm_z	0	1	2.69e-1	0.553
51	accel_arm_x	0	1	-6.02e+1	182.
52	accel_arm_y	0	1	3.26e+1	110.
53	accel_arm_z	0	1	-7.12e+1	135.
54	magnet_arm_x	0	1	1.92e+2	444.
55	magnet_arm_y	0	1	1.57e+2	202.
56	magnet_arm_z	0	1	3.06e+2	327.
57	max_roll_arm	19216	0.0207	1.12e+1	26.9
58	max_picth_arm	19216	0.0207	3.58e+1	69.6
59	max_yaw_arm	19216	0.0207	3.55e+1	10.4
60	min_roll_arm	19216	0.0207	-2.12e+1	28.7
61	min_pitch_arm	19216	0.0207	-3.39e+1	60.8
62	min_yaw_arm	19216	0.0207	1.47e+1	9.11
63	amplitude_roll_arm	19216	0.0207	3.25e+1	27.4
64	amplitude_pitch_arm	19216	0.0207	6.97e+1	67.0
65	amplitude_yaw_arm	19216	0.0207	2.08e+1	12.3
66	roll_dumbbell	0	1	2.38e+1	69.9
67	pitch_dumbbell	0	1	-1.08e+1	37.0
68	yaw_dumbbell	0	1	1.67e+0	82.5
69	max_roll_dumbbell	19216	0.0207	1.38e+1	48.3
70	max_picth_dumbbell	19216	0.0207	3.27e+1	93.4
71	min_roll_dumbbell	19216	0.0207	-4.12e+1	34.7
72	min_pitch_dumbbell	19216	0.0207	-3.32e+1	74.3
73	amplitude_roll_dumbbell	19216	0.0207	5.50e+1	54.9
74	amplitude_pitch_dumbbell	19216	0.0207	6.59e+1	65.2

Manuel Alejandro Ruiz Miranda

Selecciono y excluyo todas aquellas columnas que contengan un porcentaje de NA mayor al 20%.

- df_fit <- df_fit[, -which(colMeans(is.na(df_fit))>=0.2)]

Selecciono y excluyo todas aquellas columnas que empiezen con 'kurt','ske','max','min','ampli', por ser columnas con valores vacíos.

- df_fit <- df_fit %>% select(-starts_with(c('kurt','ske','max','min','ampli')))

Convierto la variable classe a factor, ya que es numérica.

- df_fit <- df_fit %>% mutate (classe = as.factor(classe))

Aplico una conversión de tipo a la variable cvtd_timestamp, ya que es tipo carácter y expresa una fecha.

df_fit <- df_fit %>% mutate (cvtd_timestamp = as.Date(cvtd_timestamp,"%d/%m/%y"))

Preparar Datos:

Antes de lanzar el modelo de machine learning, debemos realizar varios pasos, que nos permiten crear un modelo de forma óptima.

Feature Selection,encontrar variables con varianza cero:

```
num_cols <- sapply(df_fit, is.numeric)
varianza <- nearZeroVar(df_fit[num_cols],saveMetrics = T)
varianza
table(varianza$nzv)
FALSE
  56
```

Como vemos no tenemos variables con varianza cero que debamos excluir.

Buscar variables correlacionadas:

```
train_fit_cor <- cor(df_fit[num_cols])
eliminate <- findCorrelation(train_fit_cor,verbose = T, names = T)
```

```
> findCorrelation(train_fit_cor,verbose = T,names = T)
Compare row 14  and column  5 with corr  0.992
  Means:  0.261 vs 0.157 so flagging column 14
Compare row 5  and column  13 with corr  0.925
  Means:  0.241 vs 0.154 so flagging column 5
Compare row 13  and column  8 with corr  0.928
  Means:  0.225 vs 0.151 so flagging column 13
Compare row 12  and column  6 with corr  0.966
  Means:  0.233 vs 0.147 so flagging column 12
Compare row 23  and column  22 with corr  0.918
  Means:  0.087 vs 0.147 so flagging column 22
Compare row 50  and column  35 with corr  0.914
  Means:  0.094 vs 0.15 so flagging column 35
Compare row 50  and column  37 with corr  0.933
  Means:  0.077 vs 0.153 so flagging column 37
```

Como Podemos apreciar estas columnas las debemos eliminar por estar altamente correlacionadas.

También buscamos variables que sean combinaciones lineales.

findLinearCombos(train_fit_cor)

> findLinearCombos(train_fit_cor)

$linearCombos

list()

$remove

NULL

En consecuencia, eliminamos las columnas que son correlaciones.

df_fit <- df_fit %>% select (-eliminate [1:7])

Aplicando un análisis de componentes principales PCA con el objetivo de reducir dimensiones y lograr asi un mejor proceso de nuestro algoritmo.

pre_pca <- preProcess(df_fit,method = "pca",thresh = 0.8)

df_preProc <- predict(pre_pca,df_fit)

> pre_pca

Created from 19622 samples and 60 variables

Pre-processing:

- centered (56)
- ignored (4)
- principal component signal extraction (56)
- scaled (56)

PCA needed 14 components to capture 80 percent of the variance

> dim(df_preProc)

[1] 19622 19

Nuestro dataframe ahora consta de 19 predictores incluyendo la variable classe que es lo que vamos a predecir, con esto concluimos la transformación de datos.

#Crear partición de datos en train y test:

```
intrain <- createDataPartition(y = df_preProc$classe,p = 0.85,list = F)
training <- df_preProc[intrain,]
testing <- df_preProc[-intrain,]
```

#Paralelizacion

```
library(doParallel)
cl=makePSOCKcluster(5)
registerDoParallel(cl)
```

#Modelización

```
set.seed(1235)
cross_valid <- trainControl(method = "repeatedcv",
                            number = 10,
                            repeats = 10)

model_rf <- train (classe~., data = training,method = "rf",trControl = cross_valid)
```

> model_rf

Random Forest

16680 samples

18 predictor

5 classes: 'A', 'B', 'C', 'D', 'E'

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 10 times)

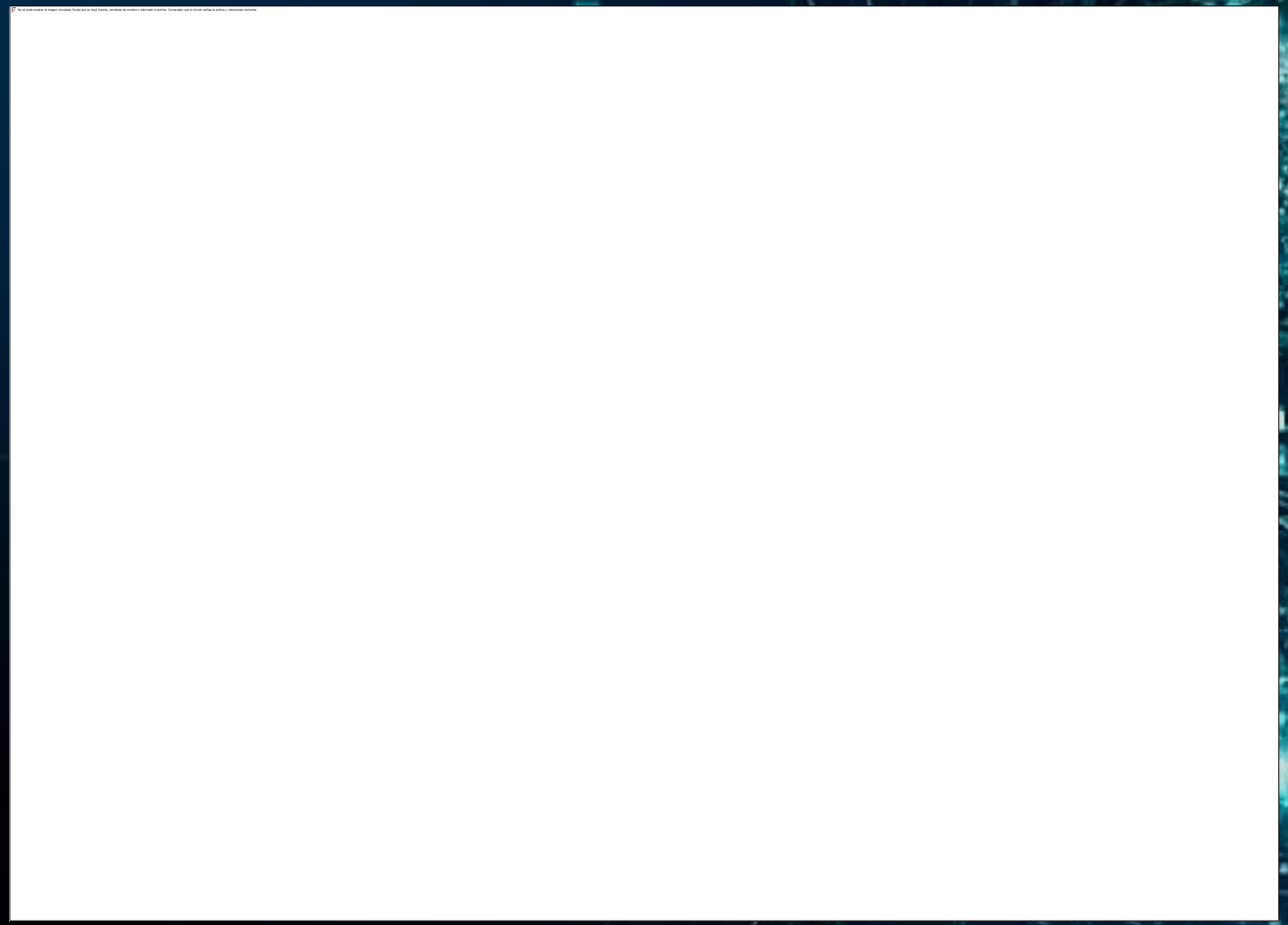
Summary of sample sizes: 15010, 15011, 15011, 15011, 15013, 15012, ...

Resampling results across tuning parameters:

mtry	Accuracy	Kappa
2	0.9672303	0.9585482
12	0.9717568	0.9642798
22	0.9574043	0.9461222

Machine learning

Random Forest



Manuel Alejandro Ruiz Miranda

Machine learning

Random Forest

```
pred <- predict(model_rf,testing)
 conf_matr <- confusionMatrix(pred,testing$classe)
 > conf_matr
Confusion Matrix and Statistics
          Reference
Prediction   A   B   C   D   E
         A 826   9   1   0   1
         B   7 549   3   1   1
         C   2   7 506  23   0
         D   1   3   3 456   3
         E   1   1   0   2 536
 Overall Statistics
               Accuracy : 0.9765
                 95% CI : (0.9704, 0.9817)
    No Information Rate : 0.2845
    P-Value [Acc > NIR] : < 2.2e-16
                  Kappa : 0.9703
 Mcnemar's Test P-Value : NA
```

Manuel Alejandro Ruiz Miranda

Statistics by Class:

	Class: A	Class: B	Class: C	Class: D	Class: E
Sensitivity	0.9869	0.9649	0.9864	0.9461	0.9908
Specificity	0.9948	0.9949	0.9868	0.9959	0.9983
Pos Pred Value	0.9869	0.9786	0.9405	0.9785	0.9926
Neg Pred Value	0.9948	0.9916	0.9971	0.9895	0.9979
Prevalence	0.2845	0.1934	0.1744	0.1638	0.1839
Detection Rate	0.2808	0.1866	0.1720	0.1550	0.1822
Detection Prevalence	0.2845	0.1907	0.1829	0.1584	0.1835
Balanced Accuracy	0.9908	0.9799	0.9866	0.9710	0.9945

Conclusion:

Como resultado de aplicar el algoritmo de Random Forest obtenemos una muy buena performance con un Accuracy : 0.9765 y un valor Kappa : 0.9703 ,los cuales indican que el modelo etiqueta correctamente y clasifica con un alto desempeño, también podemos ver que el parámetro optimizable mtry, indica el número máximo de variables en el modelo creado, alcanza un valor optimo en 12, caret estima este valor automáticamente.

Finalmente tomamos una muestra aleatoria de 20 valores del vector de predicciones

predicciones <- head(pred,20)