ProyectoAED2023

Mateo López

2023-11-13

Contents

Carga de librerías

```
library(pacman)
packages = c("MASS","knitr","tidyverse","car",'dplyr','kableExtra',"tidyr","readr","magrittr","VIM","GG
pacman::p_load(char=packages)
```

Carga de ficheros

Para crear el dataset con el que vamos a tratar en este proyecto, hemos extraido varios archivos de la web del portal de datos abiertos del Ayuntamiento de Valencia. En ellos tenemos diferente información acerca de los 88 barrios que hay en Valencia, como pueden ser el número de zonas verdes, precio del alquiler, actividad comercial, renta, etc. Antes de atacar las preguntas que nuestro conjunto resolverá, vamos a cargar los datos y unirlos en un único dataset, con una variable común para todos, el barrio.

Vulnerabilidad

El primer dataset Vulnerabilidad nos da información general del barrio, como la densidad de población, la renta media, o el estado de vulnerabilidad. Esta última variable será de gran interés en nuestro análisis posterior.

```
vuln <- read_delim("./data/vulnerabilidad.csv",
    delim = ";", escape_double = FALSE, col_types = cols(`Geo Point` = col_skip(),
        `Geo Shape` = col_skip(), `Densitat_p` = col_skip()),
    trim_ws = TRUE)%>%arrange(nombre)

colnames(vuln)[colnames(vuln) == "nombre"] <-"Barrio"
colnames(vuln)[colnames(vuln) == "Index_Gl_1"] <-"Indice_Vuln"


vuln$Barrio<-factor(vuln$Barrio,levels = unique(vuln$Barrio))
vuln$Indice_Vuln<-factor(vuln$Indice_Vuln,levels = c("Vulnerable","Pot. Vulnerable","No Vulnerable"))

head(vuln)
```

```
# A tibble: 6 x 13
  Barrio      coddistbar coddistrit codbar `Zones verd` turismes_e atur_16_64 renda_mitj risc_pobre
  <fct>       <chr>           <dbl>  <dbl>        <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 AIORA       121                12    121         1786       12.2     180.       10228.       24.3
2 ALBORS      122                12    122          712       12.2      63.6      11500        21.0
3 ARRANCAPINS 034                 3     34         1826       11.7     135.       15599.       14.7
4 BENICALAP   161                16    161         6999       11.7     301.       10256.       24.8
5 BENIFARAIG  171                17    171          521       11.9       7.45     10361        17.4
6 BENIFERRI   182                18    182           NA       NA        NA           NA        NA
# i 4 more variables: Index_Equi <dbl>, Index_Soci <dbl>, Index_Glob <dbl>, Indice_Vuln <fct>
```

Podemos ver en el código que es importante transformar la variable `Barrio` en un factor, para poder graficar y tratar la información de forma adecuada. Repetiremos este proceso en cada conjunto de datos, además de poner a todos el mismo nombre para poder unirlos más adelante.

Precio de compra y alquiler

Estos dos datasets de compra y alquiler nos presentan informaciones similares, que es la media de precios de compra y alquiler en nuestros barrios en los años 2022 y 2010. Ya que son conjuntos muy similares, vamos a adelantarnos al próximo paso y fusionarlos en un único dataset, llamado `precios`.

```
p_compra <- read_delim("./data/precio-de-compra-en-idealista.csv",
    delim = ";", escape_double = FALSE, col_types = cols(`Geo Point` = col_skip(),
        `Geo Shape` = col_skip(), Fecha_creacion = col_skip(),`Max_historico (Euros/m2)` = col_skip(),
        Año_Max_Hist = col_skip()),
    trim_ws = TRUE)%>%arrange(BARRIO)

p_compra$BARRIO<-factor(p_compra$BARRIO,levels = unique(p_compra$BARRIO))

p_alquiler <- read_delim("./data/precio-alquiler-vivienda.csv",
    delim = ";", escape_double = FALSE,
    trim_ws = TRUE, col_types = cols(`Geo Point` = col_skip(), `Geo Shape` = col_skip(), Fecha_creacion
        Año_Max_Hist = col_skip()))%>%arrange(BARRIO)

p_alquiler$BARRIO<-factor(p_alquiler$BARRIO,levels = unique(p_alquiler$BARRIO))
```

```r
precios<-full_join(p_compra,p_alquiler,by="BARRIO",suffix = c(" de compra"," de alquiler"))
colnames(precios)[colnames(precios) == "BARRIO"] <-"Barrio"

head(precios)
```

```
# A tibble: 6 x 11
  coddistbar Barrio      codbarrio coddistrit `DISTRITO de compra` Precio_2022 (Euros/m2) de compr~1
       <dbl> <fct>           <dbl>      <dbl> <chr>                                              <dbl>
1        121 AIORA               1         12 CAMINS AL GRAU                                      1894
2        122 ALBORS              2         12 CAMINS AL GRAU                                      1864
3         34 ARRANCAPINS         4          3 EXTRAMURS                                           2263
4        161 BENICALAP           1         16 BENICALAP                                           1602
5        171 BENIFARAIG          1         17 POBLATS DEL NORD                                      NA
6        182 BENIFERRI           2         18 POBLATS LOEST                                         NA
# i abbreviated name: 1: `Precio_2022 (Euros/m2) de compra`
# i 5 more variables: `Precio_2010 (Euros/m2) de compra` <dbl>, `DISTRITO de alquiler` <chr>,
#   `Precio_2022 (Euros/m2) de alquiler` <dbl>, `Precio_2010 (Euros/m2) de alquiler` <dbl>,
#   `CodBar-CodDistrit` <dbl>
```

Recibos IBI

Vamos ahora con el dataset IBI, que nos da información de los diferentes recibos del IBI (Impuesto sobre Bienes Inmuebles) entre los años 2021 y 2023. Este conjunto nos va a dar una muy buena visión acerca de la actividad del barrio, tanto comercial como cultural, turística, religiosa, industrial, etc.

Debido a que en ningún momento vamos a tratar con tiempo en este dataset, vamos a eliminar los años haciendo la media de las observaciones de cada barrio durante estos tres años, para así obtener tantas observaciones como barrios, ya que si no habrá conflictos a la hora de unir los datos.

```r
ibi <- read_delim("./data/rebuts-ibi-2022.csv", delim = ";", escape_double = FALSE, col_types = cols(ID
  arrange(Barrio)%>%
  mutate_at(vars(-all_of(c("Distrito","Barrio"))), ~as.numeric(sub(",",".",.)))

ibi$Barrio<-factor(ibi$Barrio,levels = unique(ibi$Barrio))

# Hacemos la media de las observaciones de cada barrio en los tres año y nos quitamos 2/3 de las observ
ibi <- ibi %>% group_by(Barrio) %>%  mutate(across(where(is.numeric), mean, na.rm=TRUE))%>%distinct()

head(ibi)
```

```
# A tibble: 6 x 37
# Groups:   Barrio [6]
  Distrito Barrio `Cod. Barrio` Num. Recibos persona~1 Num. Recibos persona~2 Num.Recibos sin pers~3
  <chr>    <fct>          <dbl>                 <dbl>                 <dbl>                 <dbl>
1 CAMINS ~ AIORA            121                 18631                  1155                  31
2 CAMINS ~ ALBORS           122                  6373                  1127                  26.7
3 EXTRAMU~ ARRAN~            34                 19003                  2615                  73.3
4 BENICAL~ BENIC~           161                 30077                  3916                  75.3
5 POBLES ~ BENIF~           171                   806                    46.3                 1
6 POBLES ~ BENIF~           182                   790.                  248                   0
# i abbreviated names: 1: `Num. Recibos personalidad F`, 2: `Num. Recibos personalidad J`,
#   3: `Num.Recibos sin personalidad`
# i 31 more variables: `Num.Recibos Almacen-Estacionamiento` <dbl>,
#   `Num. Recibos Actv. Comercial` <dbl>, `Num. Recibos Actv. Cultural` <dbl>,
#   `Num. Recibos Actv. Deportiva` <dbl>, `Num.Recibos Actv.Edificio singular` <dbl>,
```

```
#   `Num. Recibos Actv. Espectaculos` <dbl>, `Num. Recibos Actv. Industrial` <dbl>,
#   `Num.Recibos Actv.Obras Urbanizacion` <dbl>, `Num.Recibos Actv.Ocio y Hostaleria` <dbl>, ...
```

Bancos por barrio

Por último, vamos con nuestro último conjunto de datos, barrios, que contiene mucha información acerca de la ubicación de las entidades bancarias en nuestra ciudad. Debido a que nosotros solo vamos a tratar con barrios y no con direcciones ni nada similar, hemos decidido que lo más interesante de este conjunto es el número de bancos que podemos encontrar en cada barrio (puede ser un buen indicador de riqueza o pobreza). Guardaremos esta información en un nuevo dataset llamado `num_bancos`.

```
bancos <- read_delim("./data/bancs-en-via-publica-bancos-en-via-publica.csv",
    delim = ";", escape_double = FALSE, col_types = cols(gid = col_skip(),
        `Num. Policia` = col_skip(), geo_point_2d = col_skip()),
    trim_ws = TRUE)%>%arrange(Barrio)

bancos$Barrio%<>%gsub("[0-9] - ","",.)
bancos$Barrio%<>%factor(levels = unique(bancos$Barrio))

num_bancos<-bancos%>%group_by(Barrio)%>%summarize(Num_bancos=n())
head(num_bancos)
```

```
# A tibble: 6 x 2
  Barrio     Num_bancos
  <fct>           <int>
1 -                   2
2 AIORA              31
3 BENICALAP         313
4 BENIFARAIG         15
5 BENIMACLET        234
6 BENIMAMET          35
```

Población

Este conjunto contiene el area y la población de los diferentes barrios. Hemos considerado este dataset debido a que los datos de areas y densidad de población que nos proporcionaba el conjunto vulnerabilidad y barrios no cuadraban con nuestras búsquedas y carecían de sentido. Por ello hemos considerado este otro que se ajusta mucho mejor. Para calcular la densidad usaremos la función mutate.

Este *dataset* se trata de un directorio donde se encuentran un total de 176 archivos Excel. Dos de estos aportan información relacionada con Valencia, mientras que el resto dan información específica de cada barrio (por cada barrio hay dos archivos, uno en castellano y otro en valenciano). Por tanto, hemos creado un programa que abra uno a uno los archivos en un idioma y obtenga la información que deseamos para después guardarla en un dataframe.

Para ello, primero hemos abierto uno de los archivos Excel (ver Figura 1) y se ha estudiado como abordar el problema. Después, se ha cargado uno de ellos, y se ha buscado la manera de obtener la información referida al nombre del barrio, el área y la población del mismo (recuadros amarillos en la Figura 1).

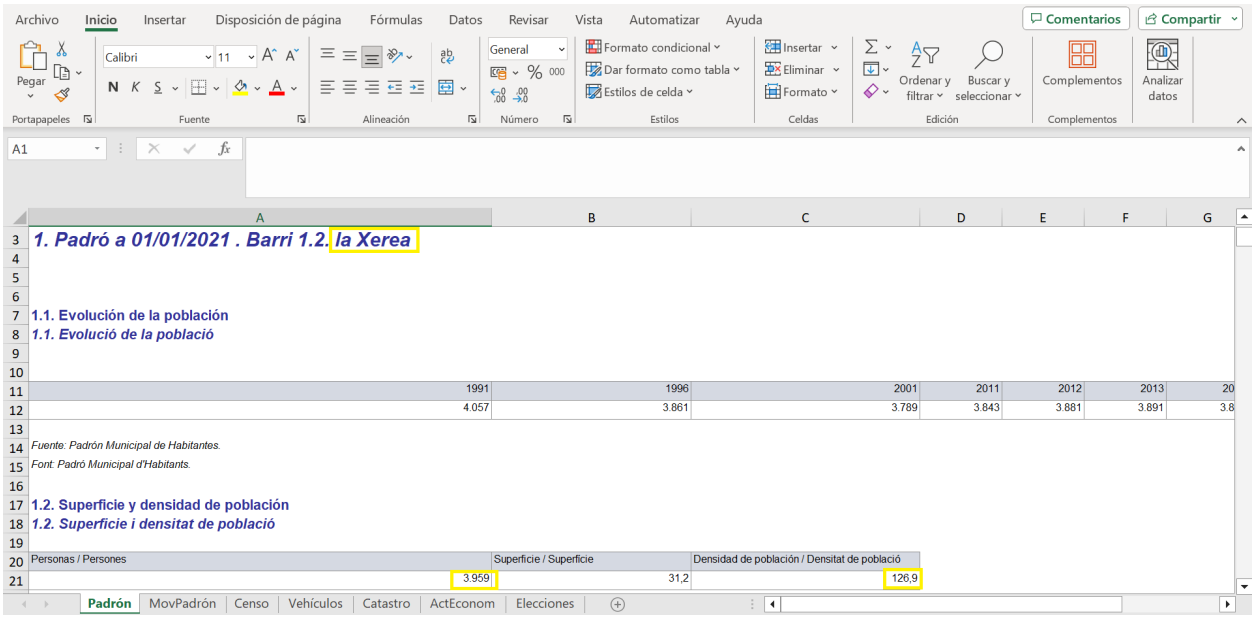


Figura 1: ejemplo de uno de los archivos Excel analizados.

```
#   SE HA PUESTO CON eval = FALSE PORQUE TARDA MUCHO EN CORRER
# Obtener las rutas de todos los archivos con los que se va a trabajar
archivos_barrios <- list.files(path = "./data/Barrios2022", pattern = 'Districte', full.names = TRUE)

# Creamos las listas donde vamos a guardar los datos de interés (la longitud es de 87 porque hay inform
area <- 1:87
poblacion <- 1:87
nombre <- 1:87

# Un bucle que recorre todos los archivos que utilizaremos y extrae los datos de interés
for (i in 1:length(archivos_barrios)) {
  Barrio <- read_excel(archivos_barrios[i], sheet = "Padrón")
  fila <- grep('Padró' ,Barrio[[1]])[1]
  nombre_barrio <- Barrio[[1]][fila]
  nombre[i] <-trimws(toupper(substring(nombre_barrio, 36)))
  fila <- grep('Superficie' ,Barrio$...2)
  area[i] <- Barrio$...2[fila + 1]
  fila <- grep('Personas', Barrio[[1]])[1]
  poblacion[i] <- Barrio[[1]][fila + 1]
}

demografico <- data.frame(nombre, area, poblacion)

# Hay algunos nombres que se han guardado mal, por lo que vamos a tener que tratarlos. Vemos que todos
for (i in 1:length(demografico$nombre)) {
  if (grepl('^[^A-Za-z]',demografico$nombre)[i]) {
    demografico$nombre[i] <- str_remove(demografico$nombre[i], "([0-9]|). ")
  }
}

# También falta el dato de un barrio, asi que hay una fila de más
demografico <- demografico[!(demografico$nombre == '88'), ]
```

```r
# Vemos quales de los nombres del dataframe nuevo no están en el principal
precios$Barrio[!(demografico$nombre %in% precios$Barrio)]

# Hay que poner de la misma maera que estan en el data-frame pricipal
demografico$nombre[demografico$nombre == "GRAN VIA"] <- "LA GRAN VIA"
demografico$nombre[demografico$nombre == "EXPOSICIÓ"] <- "EXPOSICIO"
demografico$nombre[demografico$nombre == "CIUTAT UNIVERSITÀRIA"] <- "CIUTAT UNIVERSITARIA"
demografico$nombre[demografico$nombre == "SANT MARCEL·LÍ"] <- "SANT MARCEL.LI"
demografico$nombre[demografico$nombre == "CAMÍ REAL"] <- "CAMI REAL"
demografico$nombre[demografico$nombre == "MONT-OLIVET"] <- "MONTOLIVET"
demografico$nombre[demografico$nombre == "FONTETA DE SANT LLUÍS"] <- "LA FONTETA S.LLUIS"
demografico$nombre[demografico$nombre == "CIUTAT DE LES ARTS I DE LES CIÈNCIES"] <- "CIUTAT DE LES ARTS
demografico$nombre[demografico$nombre == "EL CABANYAL- EL CANYAMELAR"] <- "CABANYAL-CANYAMELAR"
demografico$nombre[demografico$nombre == "EL BOTÀNIC"] <- "EL BOTANIC"
demografico$nombre[demografico$nombre == "BETERÓ"] <- "BETERO"
demografico$nombre[demografico$nombre == "CAMÍ FONDO"] <- "CAMI FONDO"
demografico$nombre[demografico$nombre == "CIUTAT JARDÍ"] <- "CIUTAT JARDI"
demografico$nombre[demografico$nombre == "LA BEGA BAIXA"] <- "LA VEGA BAIXA"
demografico$nombre[demografico$nombre == "CAMÍ DE VERA"] <- "CAMI DE VERA"
demografico$nombre[demografico$nombre == "ORRIOLS"] <- "ELS ORRIOLS"
demografico$nombre[demografico$nombre == "SANT LLORENÇ"] <- "SANT LLORENS"
demografico$nombre[demografico$nombre == "CASES DE BÀRCENA"] <- "LES CASES DE BARCENA"
demografico$nombre[demografico$nombre == "MAUELLA"] <- "MAHUELLA-TAULADELLA"
demografico$nombre[demografico$nombre == "\t\nBORBOTO"] <- "BORBOTO"
demografico$nombre[demografico$nombre == "\t\nBENIMAMET"] <- "BENIMAMET"
demografico$nombre[demografico$nombre == "EL CASTELLAR-L'OLIVERAR"] <- "CASTELLAR-L'OLIVERAL"

# Guardar el RData
save(demografico, file = "./data/demografico.RData")
```

Una vez creado, vamos a cargarlo:

```r
load("./data/Demografico.RData")

colnames(demografico)[colnames(demografico) == "nombre"] <-"Barrio"

demografico$Barrio<-factor(demografico$Barrio,levels = unique(demografico$Barrio))

demografico%<>%
  arrange(`Barrio`)%>%
  mutate(across(-c("Barrio"), as.numeric))%>%
  mutate(`Densidad`=`poblacion`/`area`)

head(demografico)
```

```
         Barrio area poblacion Densidad
1        LA SEU 22.1      3049 137.9638
2      LA XEREA 31.2      3959 126.8910
3      EL CARME 38.4      6590 171.6146
4      EL PILAR 16.2      4632 285.9259
5     EL MERCAT 17.3      3657 211.3873
6 SANT FRANCESC 43.9      5638 128.4282
```

Hemos tenido que aplicar una transformación a las columnas a traves de un mutate y un across ya que estas eran de tipo carácter, y no de tipo numérico.

Fusion de los dataset

```
df<-vuln%>%full_join(demografico,by="Barrio")%>%full_join(num_bancos,by="Barrio")%>%full_join(precios,by
```

```
dim(df)
```

```
[1] 92 63
```

```
tail(df)
```

```
# A tibble: 6 x 63
  Barrio   coddistbar.x coddistrit.x codbar `Zones verd` turismes_e atur_16_64 renda_mitj risc_pobre
  <fct>    <chr>               <dbl>  <dbl>        <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 TRINITAT 053                     5     53         4402       12.2       65.3     12704        20.6
2 VARA DE~ 083                     8     83         1770       11.5       85.0     11477.       18.6
3 MONTOLI~ <NA>                   NA     NA           NA         NA         NA        NA         NA
4 -        <NA>                   NA     NA           NA         NA         NA        NA         NA
5 <NA>     <NA>                   NA     NA           NA         NA         NA        NA         NA
6 FONTETA~ <NA>                   NA     NA           NA         NA         NA        NA         NA
# i 54 more variables: Index_Equi <dbl>, Index_Soci <dbl>, Index_Glob <dbl>, Indice_Vuln <fct>,
#   area <dbl>, poblacion <dbl>, Densidad <dbl>, Num_bancos <int>, coddistbar.y <dbl>,
#   codbarrio <dbl>, coddistrit.y <dbl>, `DISTRITO de compra` <chr>,
#   `Precio_2022 (Euros/m2) de compra` <dbl>, `Precio_2010 (Euros/m2) de compra` <dbl>,
#   `DISTRITO de alquiler` <chr>, `Precio_2022 (Euros/m2) de alquiler` <dbl>,
#   `Precio_2010 (Euros/m2) de alquiler` <dbl>, `CodBar-CodDistrit` <dbl>, Distrito <chr>,
#   `Cod. Barrio` <dbl>, `Num. Recibos personalidad F` <dbl>, ...
```

Vemos como a la hora de fusionar todos los datos en un solo dataset, tenemos un problema, y es que contamos con más observaciones de las esperadas. Deberíamos tener un total de 88 observaciones (una por cada barrio), pero en cambio, tenemos 92. Mirando el final del dataset, vemos como efectivamente hay cuatro observaciones que no se corresponden con lo deseado, así que vamos a arreglarlo.

```
# El barrio MONTOLIVET se llama únicamente en el dataset "vuln" y "num_bancos" como MONT-OLIVET
levels(vuln$Barrio)[vuln$Barrio=="MONT-OLIVET"]<-"MONTOLIVET"
levels(num_bancos$Barrio)[num_bancos$Barrio=="MONT-OLIVET"]<-"MONTOLIVET"

# Lo mismo ocurre con la Fonteta de sant lluis y el dataset "ibi
levels(ibi$Barrio)[ibi$Barrio=="FONTETA DE SANT LLUIS"]<-"LA FONTETA S.LLUIS"

# Además, la primera y última observación de barrios no corresponden a ningún barrio, por lo que tenemo
num_bancos%<>%slice(-c(1,length(num_bancos$Num_bancos)))
```

Vemos como dos de estas incongruencias se debían a la distinta forma de escribir el nombre de los barrios, mientras que las otras dos simplemente se debían a que algún dataset contenía información de barrios desconocidos, lo cual es mejor eliminar directamente.

Una vez solucionado, volvemos a crear el dataset:

```
df<-vuln%>%full_join(demografico,by="Barrio")%>%full_join(num_bancos,by="Barrio")%>%full_join(precios,by
```

```
dim(df)
```

```
[1] 88 63
```

```
head(df)
```

```
# A tibble: 6 x 63
  Barrio   coddistbar.x coddistrit.x codbar `Zones verd` turismes_e atur_16_64 renda_mitj risc_pobre
```

```
  <fct>    <chr>               <dbl>  <dbl>        <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 AIORA    121                    12    121         1786       12.2     180.       10228.       24.3
2 ALBORS   122                    12    122          712       12.2      63.6      11500        21.0
3 ARRANCA~ 034                     3     34         1826       11.7     135.       15599.       14.7
4 BENICAL~ 161                    16    161         6999       11.7     301.       10256.       24.8
5 BENIFAR~ 171                    17    171          521       11.9       7.45     10361        17.4
6 BENIFER~ 182                    18    182           NA       NA        NA           NA         NA
# i 54 more variables: Index_Equi <dbl>, Index_Soci <dbl>, Index_Glob <dbl>, Indice_Vuln <fct>,
#   area <dbl>, poblacion <dbl>, Densidad <dbl>, Num_bancos <int>, coddistbar.y <dbl>,
#   codbarrio <dbl>, coddistrit.y <dbl>, `DISTRITO de compra` <chr>,
#   `Precio_2022 (Euros/m2) de compra` <dbl>, `Precio_2010 (Euros/m2) de compra` <dbl>,
#   `DISTRITO de alquiler` <chr>, `Precio_2022 (Euros/m2) de alquiler` <dbl>,
#   `Precio_2010 (Euros/m2) de alquiler` <dbl>, `CodBar-CodDistrit` <dbl>, Distrito <chr>,
#   `Cod. Barrio` <dbl>, `Num. Recibos personalidad F` <dbl>, ...
```

Con esto, ya tenemos el número de observaciones deseadas, asi que podemos proceder con el siguiente paso.

Selección de variables inicial

Observando las 69 variables con las que cuenta nuestro conjunto, vemos como claramente hay muchas que no necesitamos. Primero, tenemos todos los códigos de los barrios, que prácticamente cada dataset de los anteriores contaba con una o más variables de este estilo, y con distintos nombres entre sí. Vamos a empezar eliminandolas aplicando expresiones regulares, ya que todas cuentan con una cosa en común, que empiezan por "cod":

```
codigos<-grepl("^[Cc]od",colnames(df))
df%<>%select(-colnames(df)[codigos])
```

Con esto nos hemos quitado un total de 11 variables, pero aún podemos hacer más. También tenemos otra variable redundante, que son los distritos. Como el distrito de compra es el único que no tiene ningún valor perdido, usaremos ese, y además, lo transformaremos en factor:

```
distritos<-colnames(df)[grepl("^DISTRITO|Distrito",colnames(df))]
distritos<-distritos[distritos!="DISTRITO de compra"]

df%<>%select(-distritos)

colnames(df)[colnames(df) == "DISTRITO de compra"] <-"Distrito"

df$Distrito<-factor(df$Distrito,levels = unique(df$Distrito))

df%<>%relocate(Barrio,Distrito,Indice_Vuln)
```

Por último, vemos como dentro del dataset IBI tenemos por un lado las variables que indica el número de recibos de un cierto tipo y en otra el importe. Consideramos que nos pueden ser más de utilidad las segundas, y para no ser reduntantes vamos a eliminar las de número de recibos. Además, algunas de estas cuentan con entradas decimales, lo cual es un poco extraño para lo que la variable representa.

```
num<-grepl("^Num\\.",colnames(df))
df%<>%select(-colnames(df)[num])
```

Finalmente tenemos nuestro conjunto de datos cargado y liberado de variables innecesarias, vamos a echar un vistazo:

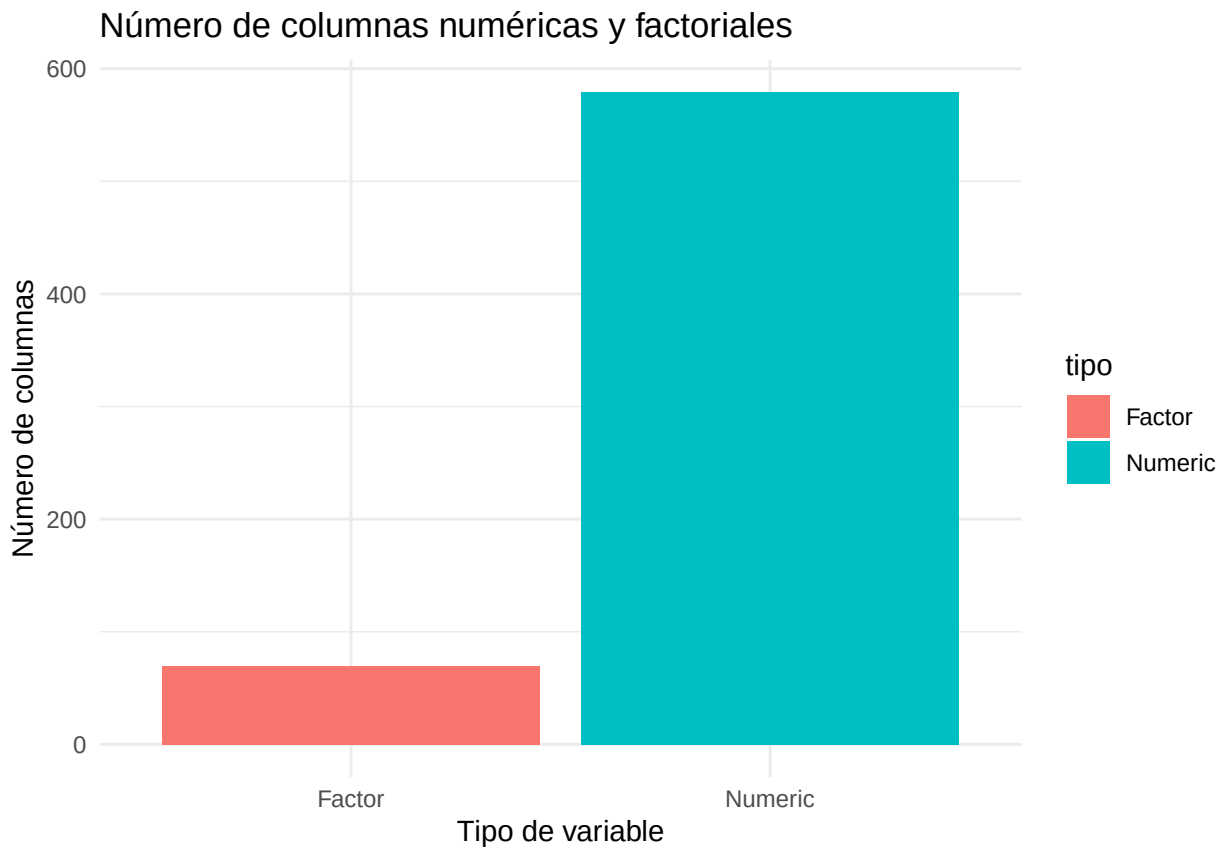
```
dim(df)
```

```
[1] 88 36
```

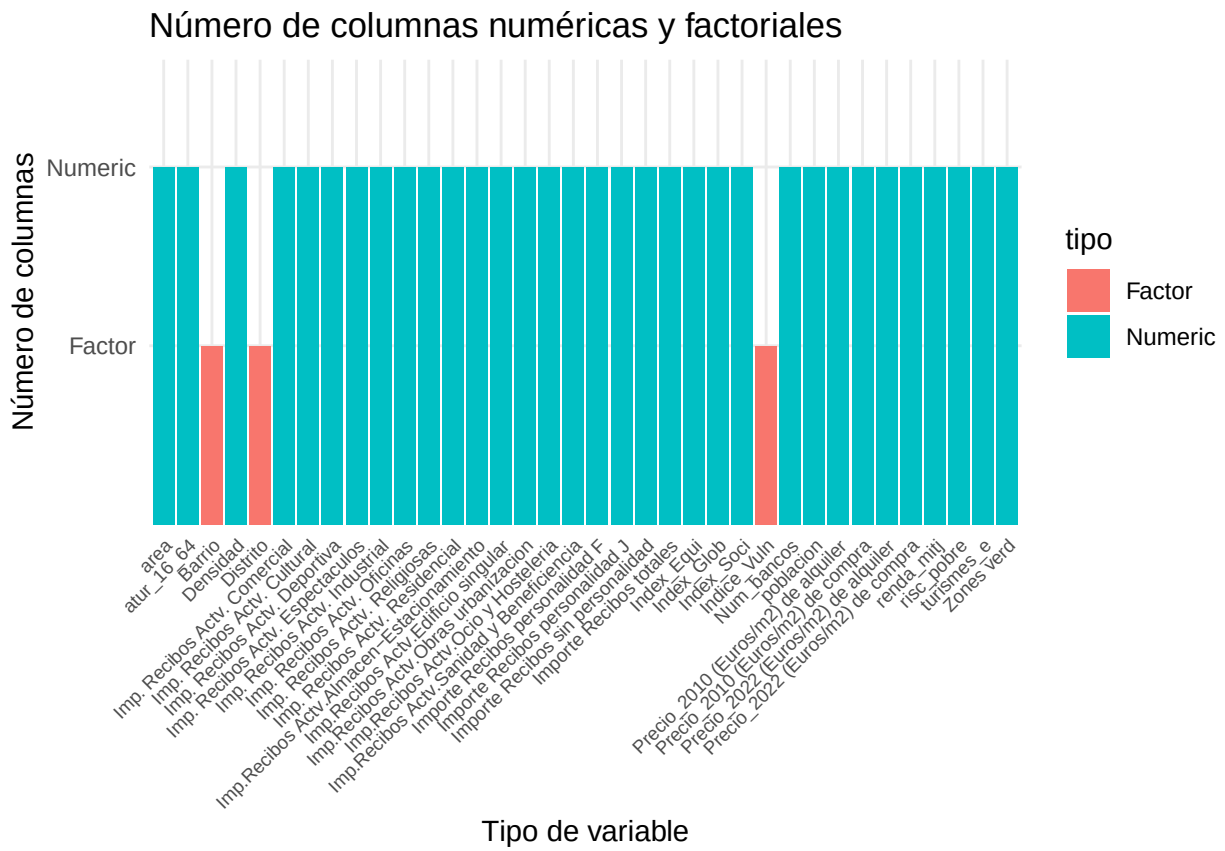
```r
head(df)
```

```
# A tibble: 6 x 36
  Barrio    Distrito Indice_Vuln `Zones verd` turismes_e atur_16_64 renda_mitj risc_pobre Index_Equi
  <fct>     <fct>    <fct>              <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 AIORA     CAMINS ~ Vulnerable          1786       12.2     180.       10228.       24.3       3.04
2 ALBORS    CAMINS ~ Vulnerable           712       12.2      63.6      11500        21.0       3.31
3 ARRANCAP~ EXTRAMU~ No Vulnera~         1826       11.7     135.       15599.       14.7       3.26
4 BENICALAP BENICAL~ Vulnerable          6999       11.7     301.       10256.       24.8       2.76
5 BENIFARA~ POBLATS~ Pot. Vulne~          521       11.9       7.45     10361        17.4       1.35
6 BENIFERRI POBLATS~ <NA>                  NA       NA        NA           NA        NA         NA
# i 27 more variables: Index_Soci <dbl>, Index_Glob <dbl>, area <dbl>, poblacion <dbl>,
#   Densidad <dbl>, Num_bancos <int>, `Precio_2022 (Euros/m2) de compra` <dbl>,
#   `Precio_2010 (Euros/m2) de compra` <dbl>, `Precio_2022 (Euros/m2) de alquiler` <dbl>,
#   `Precio_2010 (Euros/m2) de alquiler` <dbl>, `Importe Recibos personalidad F` <dbl>,
#   `Importe Recibos personalidad J` <dbl>, `Importe Recibos sin personalidad` <dbl>,
#   `Imp.Recibos Actv.Almacen-Estacionamiento` <dbl>, `Imp. Recibos Actv. Comercial` <dbl>,
#   `Imp. Recibos Actv. Cultural` <dbl>, `Imp. Recibos Actv. Deportiva` <dbl>, ...
```

```r
numeric_cols <- sapply(df, is.numeric)
factor_cols <- sapply(df, is.factor)

# Crear un data frame para el gráfico de barras
summary_df <- data.frame(
  variable = colnames(df),
  tipo = ifelse(numeric_cols,"Numeric", "Factor"),
  count = c(sum(numeric_cols), sum(factor_cols))
)

# Crear un gráfico de barras
ggplot(summary_df, aes(x = tipo, y = count, fill = tipo)) +
  geom_bar(stat = "identity") +
  labs(title = "Número de columnas numéricas y factoriales",
       x = "Tipo de variable",
       y = "Número de columnas") +
  theme_minimal()
```




```
ggplot(summary_df, aes(x = variable, y = tipo, fill = tipo)) +
  geom_bar(stat = "identity") +
  labs(title = "Número de columnas numéricas y factoriales",
       x = "Tipo de variable",
       y = "Número de columnas") +
  theme_minimal()+
  theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1,size=rel(0.8)))
```



Tenemos un total de 33 variables numéricas (cuantitativas) y 3 variables de tipo factor (cualitativas). Una vez creado y depurado nuestro conjunto de forma preliminar, vamos a plantear las preguntas que queremos resolver y acabar de poner nuestro dataset a punto.

Preguntas a resolver

Claramente este conjunto de datos gira entorno a un objeto, los barrios de Valencia. Además, la variable del Índice de Vulnerabilidad consideramos que tiene muchísimo interés. Por tanto, todo nuestro trabajo va a consistir en encontrar las variables que más influyen en esta vulnerabilidad, y encontrar la manera en la que estas contribuyen.

Dado que la variable vulnerabilidad es de tipo factor, hemos tratado de responder estas preguntas, primero, tratando de ver como están correlacionadas las variables numéricas entre sí, creando así unos grupos dentro de las variables. Posteriormente, hemos realizado diferentes gráficas para poder relacionar la compleja relación entre las mismas.

Estudio de la correlación

(Te encargas tu Pedro)

```
# Selecciona solo columnas numéricas
df_numeric <- select_if(df, is.numeric)

# Correlación de Pearson
cor_pearson <- cor(df_numeric, method = "pearson", use = "complete.obs")
```

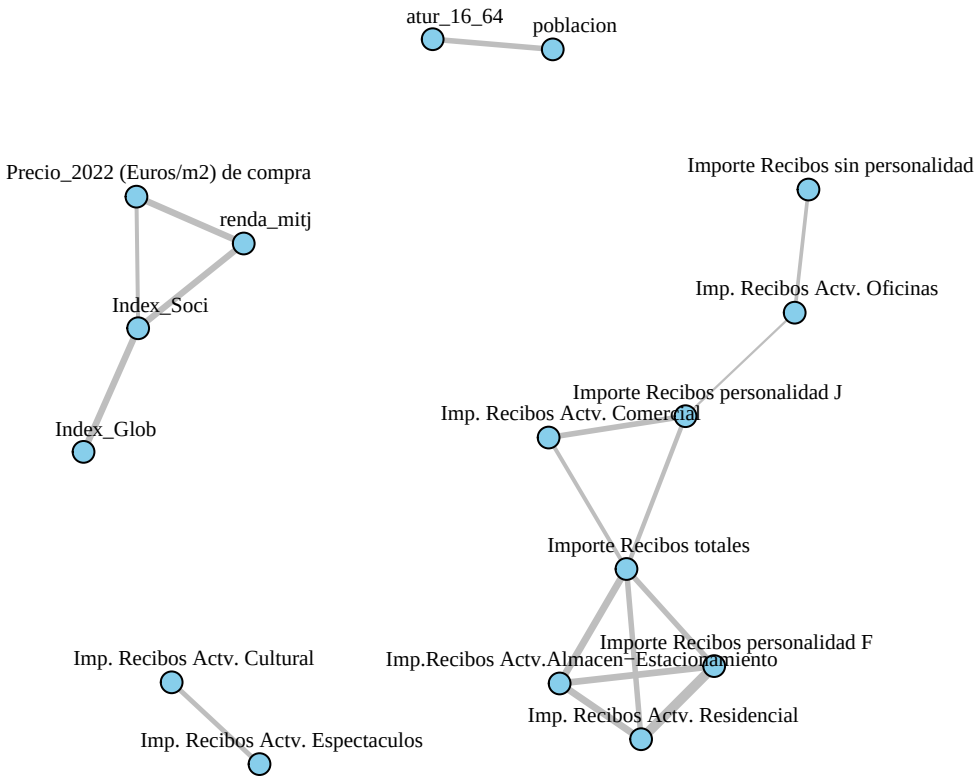
```r
# Correlación de Spearman
cor_spearman <- cor(df_numeric, method = "spearman", use = "complete.obs")
```

```r
# Convertir la matriz de correlación a un formato largo
cor_pearson_long <- as.data.frame(cor_pearson) %>%
  rownames_to_column(var = "Variable1") %>%
  gather(key = "Variable2", value = "Correlation", -Variable1)

# Filtrar las correlaciones mayores a 0.8 y diferentes de 1
strong_correlations <- cor_pearson_long %>%
  filter(abs(Correlation) > 0.8, abs(Correlation) < 1) %>%
  filter(!duplicated(t(apply(.[, c("Variable1", "Variable2")], 1, sort))))

# Mostrar los resultados
print(strong_correlations)
```

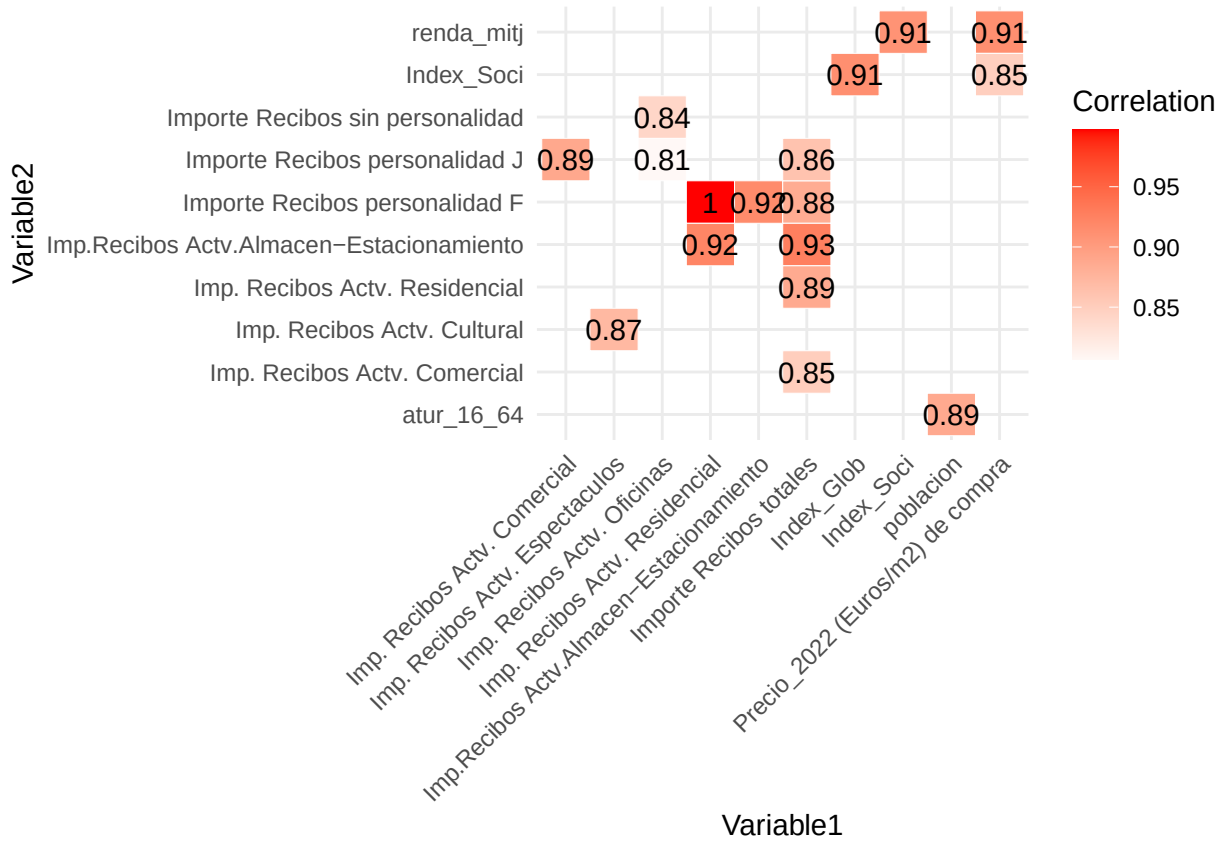
	Variable1	Variable2	Correlation
1	poblacion	atur_16_64	0.8877622
2	Index_Soci	renda_mitj	0.9089642
3	Precio_2022 (Euros/m2) de compra	renda_mitj	0.9132029
4	Index_Glob	Index_Soci	0.9131050
5	Precio_2022 (Euros/m2) de compra	Index_Soci	0.8495091
6	Imp.Recibos Actv.Almacen-Estacionamiento	Importe Recibos personalidad F	0.9171733
7	Imp. Recibos Actv. Residencial	Importe Recibos personalidad F	0.9974711
8	Importe Recibos totales	Importe Recibos personalidad F	0.8846799
9	Imp. Recibos Actv. Comercial	Importe Recibos personalidad J	0.8890860
10	Imp. Recibos Actv. Oficinas	Importe Recibos personalidad J	0.8068381
11	Importe Recibos totales	Importe Recibos personalidad J	0.8616861
12	Imp. Recibos Actv. Oficinas	Importe Recibos sin personalidad	0.8422134
13	Imp. Recibos Actv. Residencial	Imp.Recibos Actv.Almacen-Estacionamiento	0.9223414
14	Importe Recibos totales	Imp.Recibos Actv.Almacen-Estacionamiento	0.9276058
15	Importe Recibos totales	Imp. Recibos Actv. Comercial	0.8503477
16	Imp. Recibos Actv. Espectaculos	Imp. Recibos Actv. Cultural	0.8718010
17	Importe Recibos totales	Imp. Recibos Actv. Residencial	0.8869359

```r
# Establecer la semilla para reproducibilidad
set.seed(130)

# Crear un grafo
graph_data <- graph_from_data_frame(strong_correlations, directed = FALSE)

# Ajustar atributos del nodo
V(graph_data)$color <- "skyblue"
V(graph_data)$size <- 6
V(graph_data)$frame.color <- "black"

# Queremos que las líneas varíen entre 1 y 5 en grosor
cor_min <- 0.8
cor_max <- 1.0
width_min <- 1
width_max <- 5
E(graph_data)$width <- (abs(E(graph_data)$Correlation) - cor_min) / (cor_max - cor_min) *
                       (width_max - width_min) + width_min

# Elegir un layout que ofrezca más espacio y optimizar para evitar superposición
```

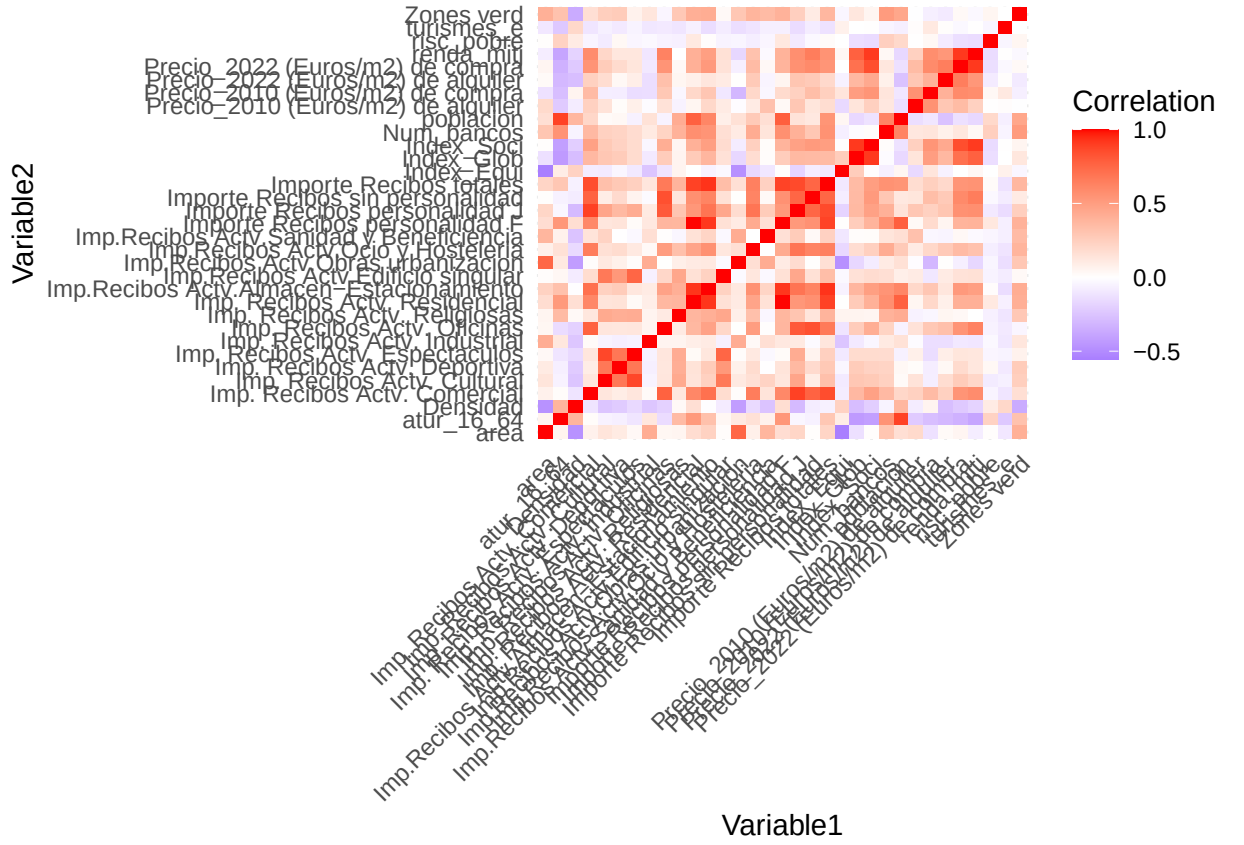
```r
layout <- layout_with_fr(graph_data)

# Dibujar el gráfico
par(mar = c(0, 0, 1.5, 0)) # Ajustar los márgenes si es necesario
plot(graph_data, layout = layout, vertex.label.color = "black", vertex.label.cex = 0.7,
     vertex.label.dist = 1.2, # Aumentar la distancia de las etiquetas de los nodos
     edge.label = NA, # Ocultar las etiquetas de las aristas para despejar el gráfico
     edge.color = "gray",
     main = "Red de Correlaciones Pearson > 0.8")
```




```r
# Crear un gráfico de etiquetas
ggplot(strong_correlations, aes(x = Variable1, y = Variable2, label = round(Correlation, 2))) +
  geom_tile(aes(fill = Correlation), color = "white") +
  geom_text() +
  scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0.8) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```




```r
# Convertir cor_pearson a formato largo
cor_pearson_long <- cor_pearson %>%
                    as.data.frame() %>%
                    rownames_to_column(var = "Variable1") %>%
                    gather(key = "Variable2", value = "Correlation", -Variable1)

# Visualizar con ggplot2
ggplot(cor_pearson_long, aes(x = Variable1, y = Variable2, fill = Correlation)) +
    geom_tile() +
    scale_fill_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



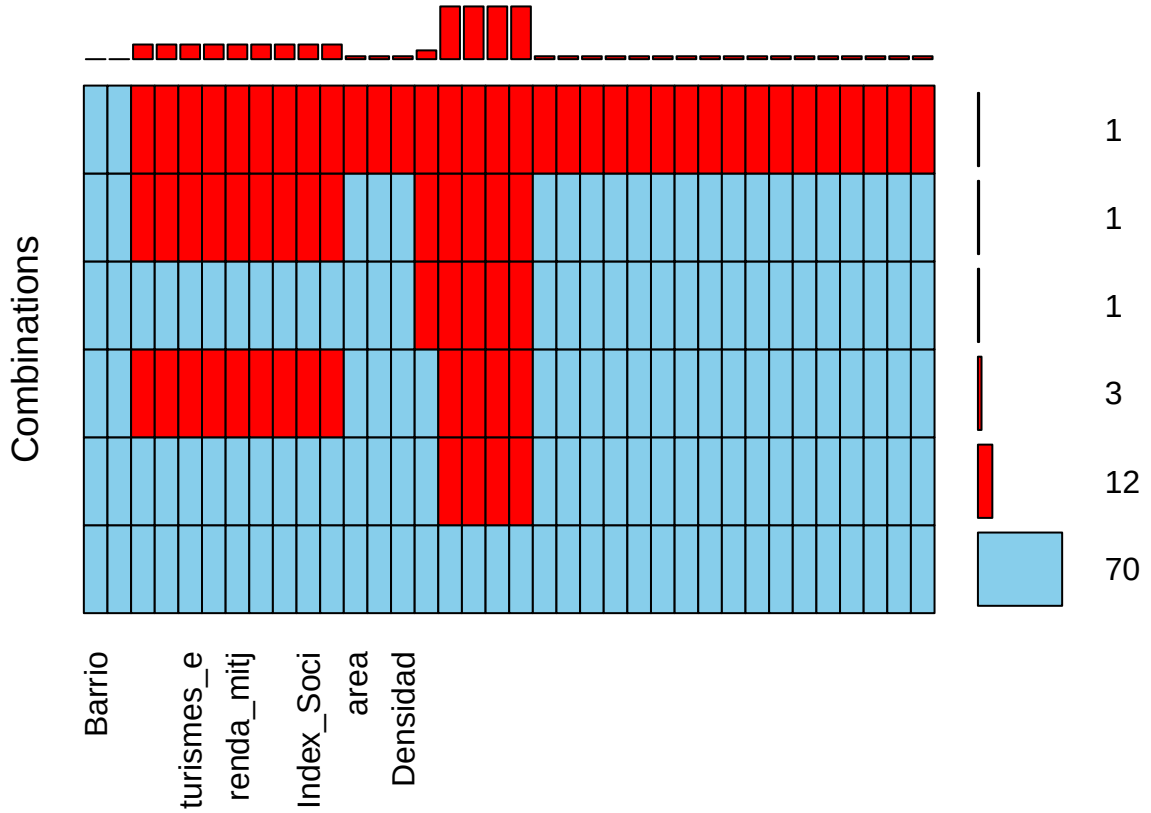

```
# Suponiendo que tu dataframe se llama df
write.table(df, file = "barrios.txt", sep = "\t")
```

Detección de anomalías

Detección de valores perdidos

Antes de tratar con nuestros datos, vamos a analizar la situación de nuestro dataset. Primero, vamos a analizar los valores perdidos.

```
aggr(df, prop = FALSE, combined = TRUE, numbers = TRUE, sortCombs = TRUE)
```



Este gráfico nos muestra las observaciones con valores perididdos y en qué columnas se hayan. Por ejemplo, para la primera observación, vemos como todas las columnas a excepción de dos cuentan con un NA, y así sucesivamente, hasta llegar a ver que hay 70 observaciones sin ningún valor perdido.

```
sum(is.na(df))
```

```
[1] 140
```

Podemos ver que la cantidad de valores perdidos en nuestro conjunto no es precisamente pequeña, y principalmente se debe al hecho de que no todos los conjuntos de datos que hemos fusionado contenían información de todos los barrios, por lo que a la hora de unirlos todos se han generado NAs en las observaciones donde no existían datos.

Una cosa que salta a la vista de las variables es esa observación que cuenta con casi todos los valores perdidos, que es la del barrio "RAFALELL-VISTABELLA", que no cuenta con ninguna información numérica en nuestro dataset. Por ello, lo mejor que podemos hacer es eliminarla.

```
df[df$Barrio=="RAFALELL-VISTABELLA",]
```

```
# A tibble: 1 x 36
  Barrio    Distrito Indice_Vuln `Zones verd` turismes_e atur_16_64 renda_mitj risc_pobre Index_Equi
  <fct>     <fct>    <fct>              <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
1 RAFALELL~ POBLATS~ <NA>                  NA         NA         NA         NA         NA         NA
# i 27 more variables: Index_Soci <dbl>, Index_Glob <dbl>, area <dbl>, poblacion <dbl>,
#   Densidad <dbl>, Num_bancos <int>, `Precio_2022 (Euros/m2) de compra` <dbl>,
#   `Precio_2010 (Euros/m2) de compra` <dbl>, `Precio_2022 (Euros/m2) de alquiler` <dbl>,
#   `Precio_2010 (Euros/m2) de alquiler` <dbl>, `Importe Recibos personalidad F` <dbl>,
#   `Importe Recibos personalidad J` <dbl>, `Importe Recibos sin personalidad` <dbl>,
#   `Imp.Recibos Actv.Almacen-Estacionamiento` <dbl>, `Imp. Recibos Actv. Comercial` <dbl>,
```

```
#   `Imp. Recibos Actv. Cultural` <dbl>, `Imp. Recibos Actv. Deportiva` <dbl>, ...
```

```
df%<>%drop_na("Importe Recibos personalidad F")

colSums(is.na(df))
```

```
                                    Barrio                                  Distrito
                                         0                                         0
                                Indice_Vuln                               Zones verd
                                         4                                         4
                                turismes_e                               atur_16_64
                                         4                                         4
                                renda_mitj                               risc_pobre
                                         4                                         4
                                Index_Equi                               Index_Soci
                                         4                                         4
                                Index_Glob                                     area
                                         4                                         0
                                 poblacion                                Densidad
                                         0                                         0
                                Num_bancos              Precio_2022 (Euros/m2) de compra
                                         2                                        17
       Precio_2010 (Euros/m2) de compra             Precio_2022 (Euros/m2) de alquiler
                                        17                                        17
     Precio_2010 (Euros/m2) de alquiler                Importe Recibos personalidad F
                                        17                                         0
         Importe Recibos personalidad J                Importe Recibos sin personalidad
                                         0                                         0
Imp.Recibos Actv.Almacen-Estacionamiento                   Imp. Recibos Actv. Comercial
                                         0                                         0
            Imp. Recibos Actv. Cultural                   Imp. Recibos Actv. Deportiva
                                         0                                         0
     Imp.Recibos Actv.Edificio singular                 Imp. Recibos Actv. Espectaculos
                                         0                                         0
          Imp. Recibos Actv. Industrial             Imp.Recibos Actv.Obras urbanizacion
                                         0                                         0
     Imp.Recibos Actv.Ocio y Hosteleria                    Imp. Recibos Actv. Oficinas
                                         0                                         0
          Imp. Recibos Actv. Religiosas                  Imp. Recibos Actv. Residencial
                                         0                                         0
Imp.Recibos Actv.Sanidad y Beneficiencia                       Importe Recibos totales
                                         0                                         0
```

En esta tabla podemos ver que variables son las que cuentan con datos perdidos, y por tanto las que debemos procesar. Vemos como las variables del dataset Vulnerabilidad cuentan todas con 4 NAs, que se deben a las observaciones de los barrios que no cuentan con Indice de Vulnerabilidad. Como esta va a ser nuestra variable objetivo durante todo el trabajo, hemos preferido no tratar con ellos, para no alterar así el resultado de nuestras observaciones.

Además, dado que algunas columnas, como las de precios, tienen un gran número de NAs, podemos usar el estudio de correlaciones que hemos visto anteriormente para sustituir el valor perdido por el equivalente en una de las columnas correlacionadas (usando una regresión, por ejemplo). En caso de tener un NA en la columna correlacionada, usaremos la mediana de los casos de su misma vulnerabilidad. En el caso de los precios de compra de 2022, usaremos la renta media del barrio, que cuentan con una correlación de 0.91:

```
reg<-lm(`Precio_2022 (Euros/m2) de compra`~renda_mitj,df)

df%<>%mutate(`Precio_2022 (Euros/m2) de compra`=ifelse(is.na(`Precio_2022 (Euros/m2) de compra`)&!is.na

df %<>%
  group_by(`Indice_Vuln`) %>%
  mutate(`Precio_2022 (Euros/m2) de compra`=ifelse(is.na(`Precio_2022 (Euros/m2) de compra`),median(`Pre
  ungroup()
```

Dado que el resto de variables con NAs no cuentan con ningun índice de correlación extremadamente altos, imputaremos los datos faltantes con la mediana de su mismo grupo de vulnerabilidad.

```
df %<>%
  group_by(`Indice_Vuln`) %>%
  mutate(`Precio_2010 (Euros/m2) de compra`=ifelse(is.na(`Precio_2010 (Euros/m2) de compra`),median(`Pre
  ungroup()
```

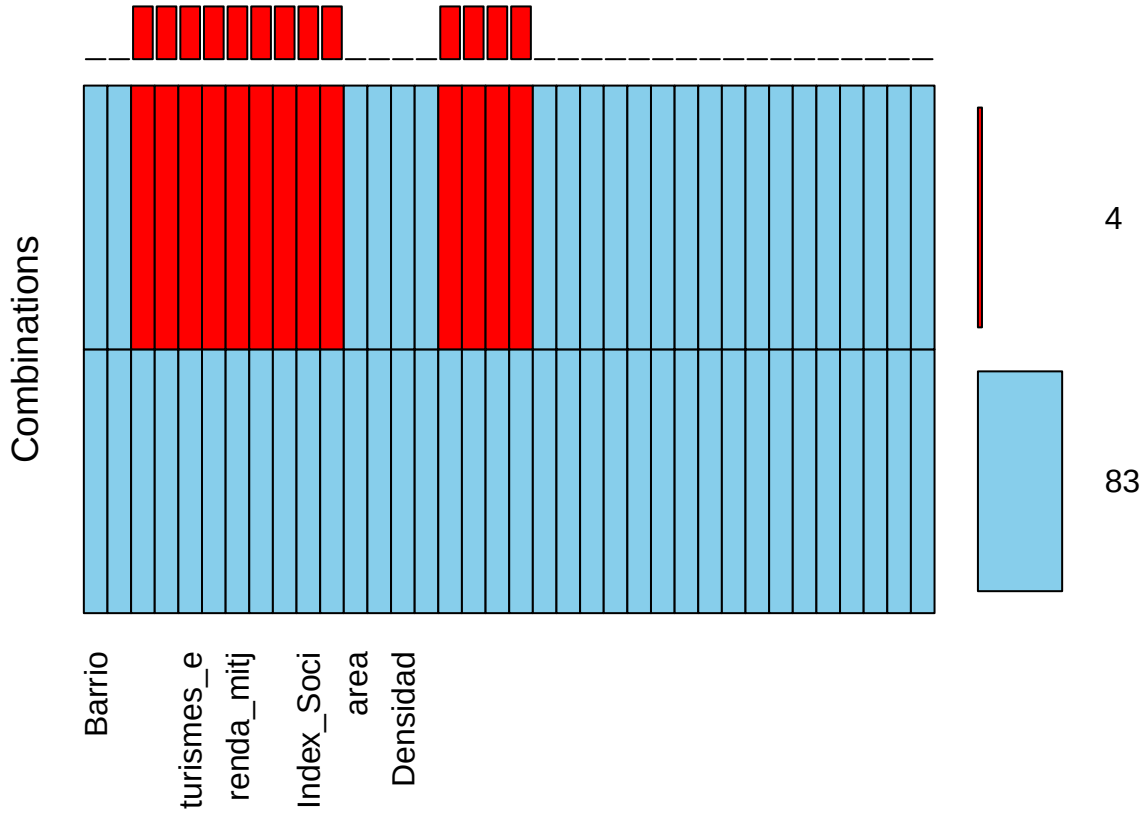
```
df %<>%
  group_by(`Indice_Vuln`) %>%
  mutate(`Precio_2022 (Euros/m2) de alquiler`=ifelse(is.na(`Precio_2022 (Euros/m2) de alquiler`),median
  ungroup()
```

```
df %<>%
  group_by(`Indice_Vuln`) %>%
  mutate(`Precio_2010 (Euros/m2) de alquiler`=ifelse(is.na(`Precio_2010 (Euros/m2) de alquiler`),median
  ungroup()
```

```
df %<>%
  group_by(`Indice_Vuln`) %>%
  mutate(`Num_bancos`=ifelse(is.na(`Num_bancos`),median(`Num_bancos`,na.rm = TRUE),`Num_bancos`))%>%
  ungroup()
```

Veamos las conclusiones:

```
aggr(df, prop = FALSE, combined = TRUE, numbers = TRUE, sortCombs = TRUE)
```



Tras esto hemos logrado pasar de tener un número muy elevado de NAs a tener solo 4 en ciertas varibles. Estos NAs están en los barrios que carecen de índice de vulverabilidad, por lo que tendríamos que esperar a ponerles una etiqueta a estos barrios para librarnos de los NAs de forma adecuada.

Detección de outliers

Vamos a tratar los outliers de nuestro conjunto antes de empezar a trabajar.

Para la detección de outliers vamos a usar los métodos 3-sigma y boxplot, con las funciones definidas en la práctica 5.

```
reglasigma<-function(x){
  x<-x[!is.na(x)& is.numeric(x)]
  out <- logical(length(x))

  for(i in 1:length(x)){
    if(abs(x[i]-mean(x))>3*sd(x)){
      out[i]<-TRUE
    }
  }
  if (all(!out)){
    return(NA)
  } else {
    return(out)
  }
}
```

```
reglaboxplot<-function(x){
  x<-x[!is.na(x)& is.numeric(x)]
  out <- logical(length(x))

  for(i in 1:length(x)){
    if(x[i]>quantile(x,0.75)+1.5*IQR(x)){
      out[i]<-TRUE
    } else if (x[i]<quantile(x,0.25)-1.5*IQR(x)){
      out[i]<-TRUE
    }
  }
  if (all(!out)){
    return(NA)
  } else {
    return(out)
  }
}
```

Vamos a aplicar las funciones vistas en busca de posibles outliers:

```
outliers <- df %>%
  summarise(across(where(is.numeric), list(Sigma = ~sum(reglasigma(.)), Boxplot = ~sum(reglaboxplot(.))

outliers%<>%
  pivot_longer(cols=everything(), names_to = "Var",values_to = "Valor")%>%
  separate(Var,into=c("Variable", "Regla"),sep=";")%>%
  spread(key=Regla,value=Valor)

outliers
```

```
# A tibble: 33 x 3
   Variable                          Boxplot Sigma
   <chr>                               <int> <int>
 1 area                                   11     3
 2 atur_16_64                              2     1
 3 Densidad                               NA    NA
 4 Imp. Recibos Actv. Comercial            9     3
 5 Imp. Recibos Actv. Cultural            10     1
 6 Imp. Recibos Actv. Deportiva           13     3
 7 Imp. Recibos Actv. Espectaculos        16     1
 8 Imp. Recibos Actv. Industrial           6     2
 9 Imp. Recibos Actv. Oficinas            11     2
10 Imp. Recibos Actv. Religiosas          10     5
# i 23 more rows
```

Viendo que la función boxplot detecta un número excesivo de outliers contando las pocas observaciones que tenemos, vamos a hacer caso a la regla sigma, y en caso de que haga falta modificar los outliers, solamente trataremos los que esta detecta, pasandolos a la mediana al igual que el ejemplo anterior, o usando alguna otra columna que esté muy correlacionada.

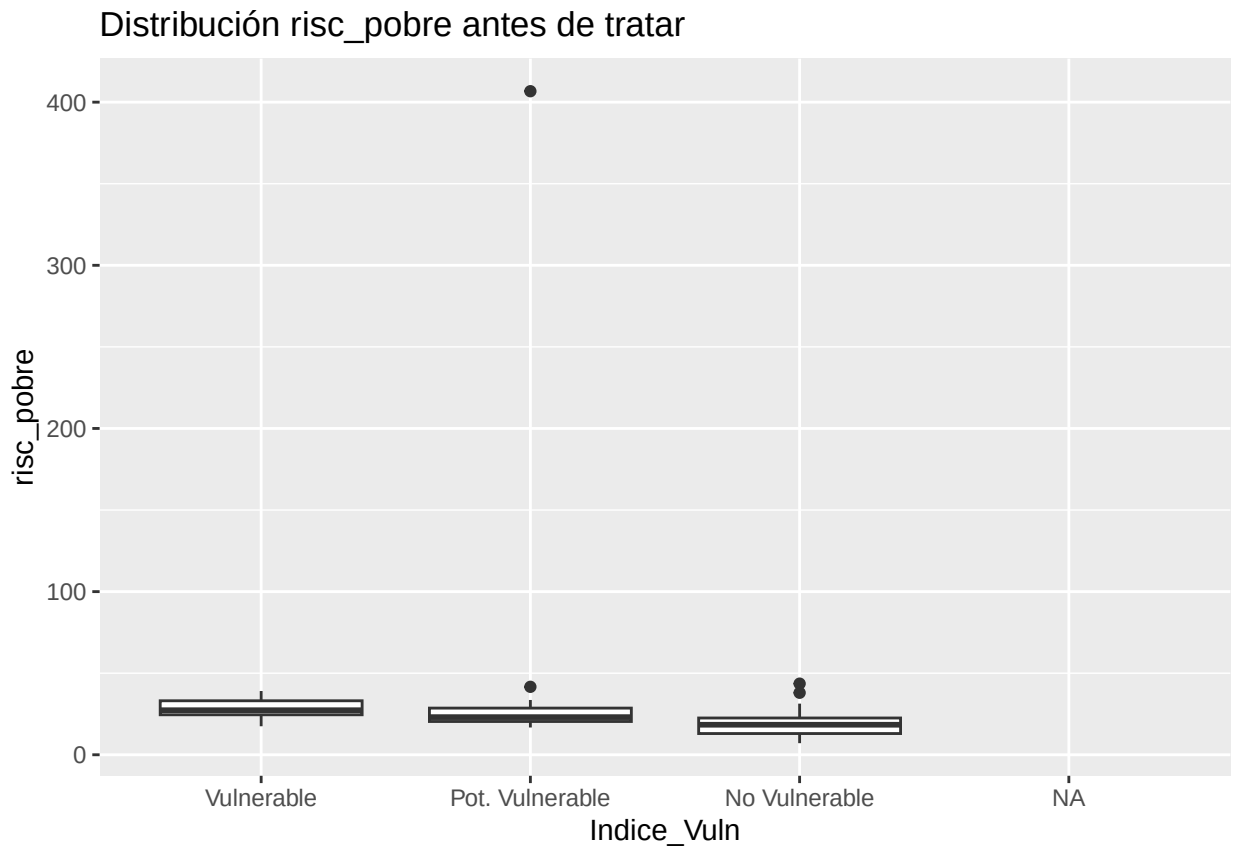
Observando y graficando detenidamente nuestro conjunto, hemos observado dos clases de outliers, un tipo del cual solo hemos detectado un caso, que concluimos que se debe a un fallo de inputación, y otro caso que se debe a un dato correcto, pero demasiado extremo, que rompe con la tendencia del resto de datos. Vamos a ver un ejemplo de cada uno.

Fallo de inputación

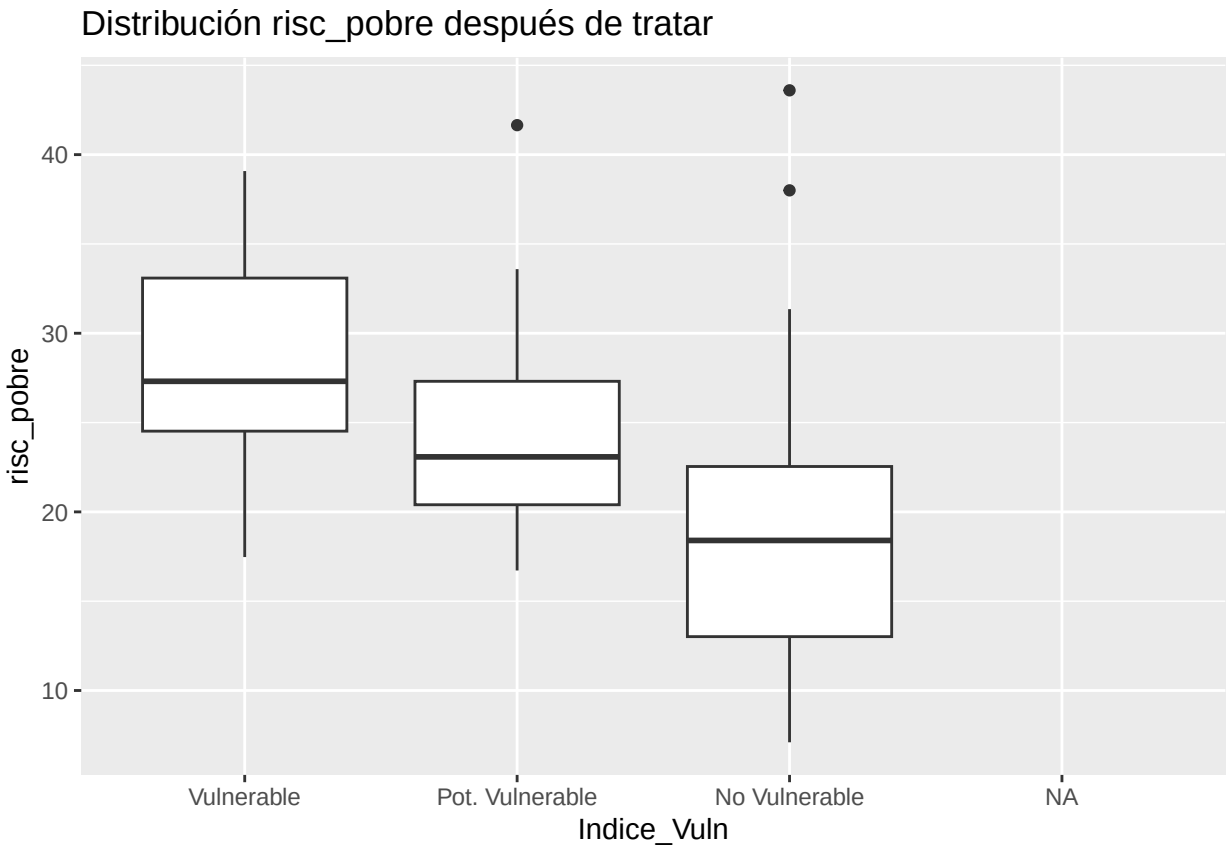
Este un ejemplo gráfico de otra forma de detectar ouliters. En el caso de la variable `risc_pobre`, el valor introducido para el barrio de Benimaclet, distaba 43.76 veces el rango intercuartílico de la mediana de la distribución.

Por tanto, se ha considerado un error de *input* y se le ha seleccionado un nuevo valor. Para ello, se ha tenido en cuenta que el análisis que se ha realizado ha sido mediante box-plots, donde se diferencian las distribuciones en función de la varaible categórica `Indice_Vuln`. Por tanto, para que no altere esta gráfica, el valor de la observación de Benimaclet se ha sustituido por la mediana correspondiente a la distribución con su misma vulnerabilidad.

```
# Corrección de outliers gráfica
ggplot(df, aes(x = Indice_Vuln, y = risc_pobre)) + geom_boxplot() + ggtitle('Distribución risc_pobre an
```




```
risc_pobre_filtrada <- df %>%
  filter(Indice_Vuln == 'Vulnerable') %>%
  filter(risc_pobre < 100) %>%
  select(risc_pobre)
df$risc_pobre[df['Barrio'] == 'BENIMACLET'] <- median(risc_pobre_filtrada[[1]])

ggplot(df, aes(x = Indice_Vuln, y = risc_pobre)) + geom_boxplot() + ggtitle('Distribución risc_pobre des
```

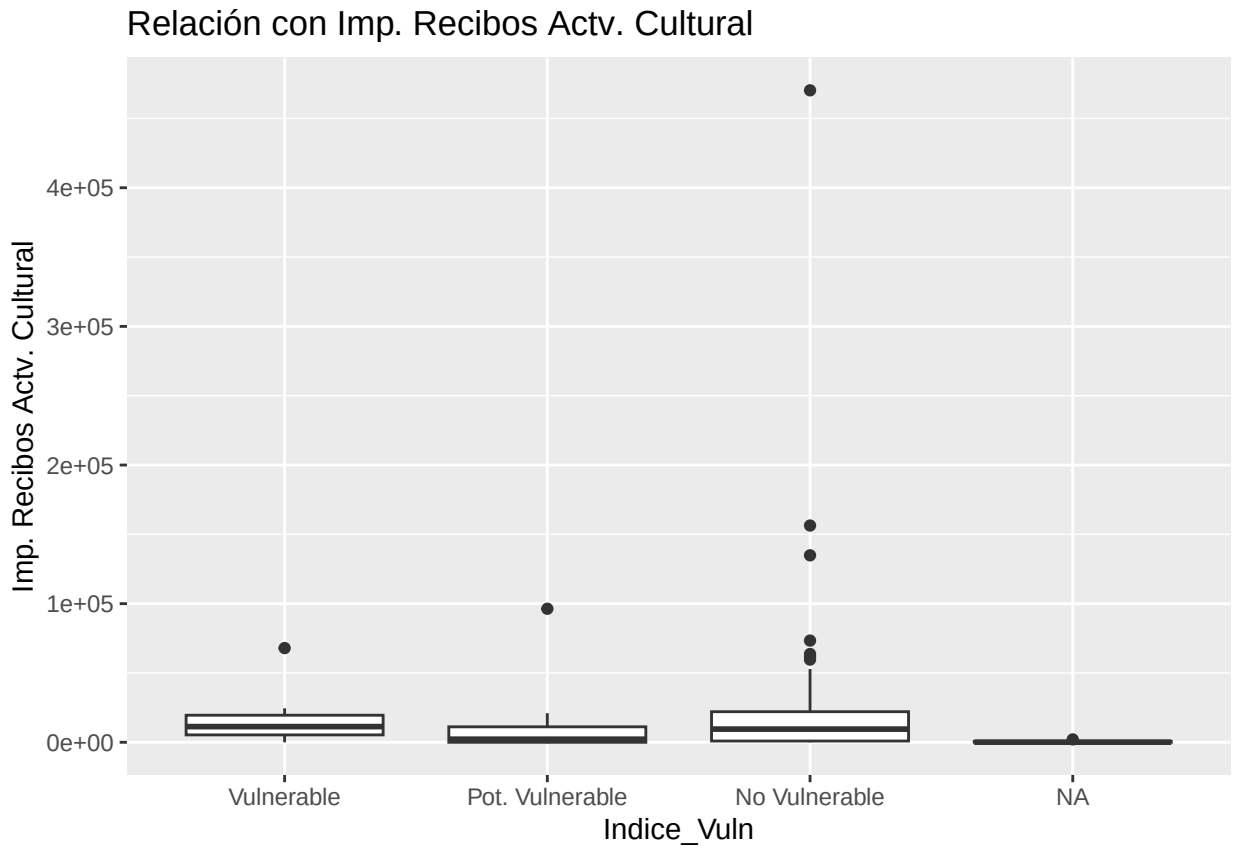


Vemos como era un outlier que no se correspondía con los datos, y al eliminarlo, tenemos una visión mucho más clara de nuestro conjunto.

Datos correctos

Un ejemplo de outlier puede verse en la variable que muestra la actividad cultural del barrio, viendo como la ciudad de las artes y las ciencias tiene un valor muchísimo más alto que el resto:

```
p<-ggplot(df[c("Imp. Recibos Actv. Cultural","Indice_Vuln")], aes(x = Indice_Vuln, y =.data[["Imp. Recil
    geom_boxplot() +
    ggtitle(paste("Relación con Imp. Recibos Actv. Cultural"))
  print(p)
```




```
print(df$`Imp. Recibos Actv. Cultural`[df$Barrio=="CIUTAT DE LES ARTS I DE LES CIENCIES"])
```

```
[1] 470265.7
```

Vemos como dentro de los barrios no vulnerables el de la ciudad de las artes y las ciencias tiene una actividad muchísimo mayor, con un valor de 470265.7. Aun así, debido a que este dato no se debe a un error a la hora de introducir el valor en el conjunto, pero se debe a que el barrio tiene una actividad cultural mayor debido a su situación. Por tanto, hemos decidido mantender estos outliers en nuestro dataset. En ciertos casos interesará eliminarlos siguiendo la regla sigma como hemos visto anteriormente, pero como nos limitamos a un análisis exploratorio, nos limitaremos a detectarlos.

Representación de los datos

Mapa de barrios vulnerables

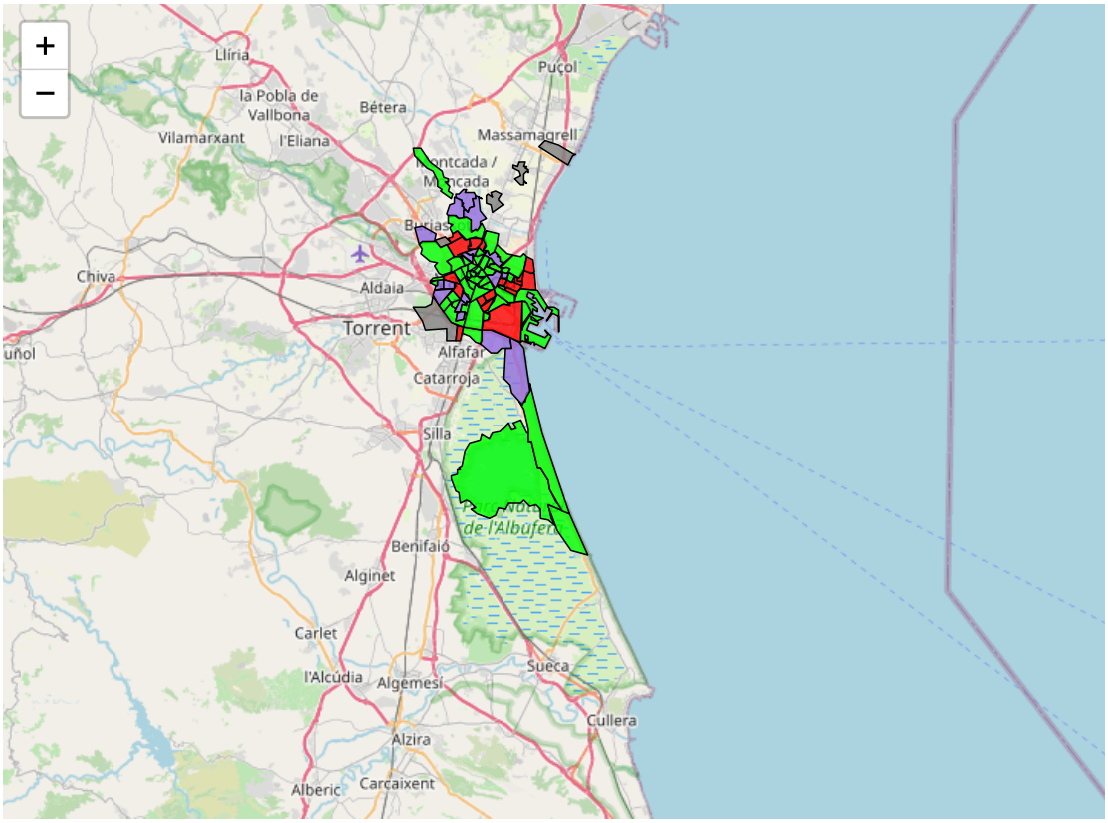
```
#primero creamos un mapa con los barrios y distrito

# Lee el archivo GeoJSON
datos_geojson <- st_read("./data/barris-barrios.geojson")
```

```
Reading layer `barris-barrios' from data source
  `C:\Users\mateo\OneDrive\Escritorio\Archivos uni\Master\Exploratorio\Proyecto\ProyectoAED2023\Proyecto
  using driver `GeoJSON'
Simple feature collection with 88 features and 6 fields
Geometry type: POLYGON
Dimension:     XY
```

```
Bounding box:  xmin: -0.432535 ymin: 39.27893 xmax: -0.2753685 ymax: 39.56659
Geodetic CRS:  WGS 84
```

```
#datos_geojson$nombre[datos_geojson$nombre %in% df$Barrio]
#df$Barrio[!df$Barrio %in% datos_geojson$nombre]
```

```
# Añado la columna Indice_Vuln al dataframe desde el cual hago el mapa
vuln2 <- vuln[c('Barrio','Indice_Vuln')]
colnames(vuln2) <- c('nombre','Indice_Vuln')
datos_geojson <- merge(x = datos_geojson, y = vuln2)

# Creo los popup del mapa
popups <- paste0("<b>", datos_geojson$nombre, "</b>", "<hr>", datos_geojson$Indice_Vuln)

# Escojo una paleta de colores
pal <- colorFactor(c('red','gray','blue','green'), levels = levels(datos_geojson$Indice_Vuln))

# Creo el mapa
leaflet(data = datos_geojson) %>%
  addTiles() %>%
  addPolygons(fillColor = pal(datos_geojson$Indice_Vuln),
              weight = 1,
              opacity = 1,
              highlightOptions = highlightOptions(color = "white",
                                                  weight = 2,
                                                  bringToFront = TRUE),
              color = 'black',
              fillOpacity = 0.8,
              popup = popups) %>%
  addLegend(data = datos_geojson,
            position = 'bottomright',
            pal = pal, values = ~Indice_Vuln,
            title = 'Leyenda',
            opacity = 1)
```



Leyenda

- Vulnerable
- Pot. Vulnerable
- No Vulnerable
- NA