

Documento de cambios

Ángela Alarcón Ballester, Lucía Ponce Salmerón y Carlos Sánchez Polo

2023-11-13

Introducción

En este proyecto analizaremos y trataremos los datos por horas de calidad del aire de las estaciones de la red de vigilancia de la ciudad de Valencia. Vamos a realizar una exploración inicial de los datos y responderemos las preguntas que se deriven de ellos.

Importación y acondicionamiento de los datos

Cargamos la librerías que necesitaremos a lo largo del proyecto:

```
library(readr) # Entrada de datos
library(dplyr) # Manipulación de datos
library(tidyr) # Manipulación de datos
library(ggplot2) # Gráficas
library(gridExtra) # Gráficas
library(pheatmap) # Para realizar operaciones
library(purrr) # Mapas de calor
library(knitr) # Tablas
```

Cargamos y visualizamos los datos:

```
rvvcca_d_horarios_2016_2020 <-
  read_delim("ProyectoAED2023_files/data/rvvcca_d_horarios_2016-2020.csv",
    delim = ";", escape_double = FALSE,
    #Las columnas Fecha y Fecha creación las definimos como fechas en formato "YYYY-MM-DD".
    col_types = cols(Fecha = col_date(format = "%Y-%m-%d"),
    `Fecha creacion` = col_date(format = "%Y-%m-%d"), ), trim_ws = TRUE)
```

```
head(rvvcca_d_horarios_2016_2020)
```

Convertimos la columna Estacion y Dia de la semana en factor:

```
rvvcca_d_horarios_2016_2020$Estacion <- as.factor(rvvcca_d_horarios_2016_2020$Estacion)
rvvcca_d_horarios_2016_2020$`Dia de la semana` <- factor(rvvcca_d_horarios_2016_2020$`Dia de la semana`)
```

Nos damos cuenta de que la columna Hora está en un formato complejo de manejar:

```
unique(rvvcca_d_horarios_2016_2020$Hora)
```

Cambiamos el formato de la columna Hora a uno más manejable:

```
rvvcca_d_horarios_2016_2020$Hora <- format(rvvcca_d_horarios_2016_2020$Hora,  
                                           format="%H:%M:%S")  
unique(rvvcca_d_horarios_2016_2020$Hora)
```

Ordenamos por fecha:

```
indices_orden <- order(rvvcca_d_horarios_2016_2020$Fecha)  
rvvcca_d_horarios_2016_2020 <- rvvcca_d_horarios_2016_2020[indices_orden, ]
```

Comprobamos que la columna Fecha baja está vacía, por tanto, no nos aporta ninguna información y la eliminamos:

```
unique(rvvcca_d_horarios_2016_2020$`Fecha baja`)  
rvvcca_d_horarios_2016_2020$`Fecha baja` <- NULL
```

Utilizamos la función `glimpse` para obtener una visión general de los datos:

```
glimpse(rvvcca_d_horarios_2016_2020)
```

Observamos que contamos con las siguientes variables:

- Información general (Id, Fecha, Día de la semana, Día del mes, Hora, Fecha de creación). Son de tipo double, factor y fecha.
- Contaminantes atmosféricos (PM1, PM2.5, PM10, NO, NO2, NOx, O3, SO2, CO, NH3, C7H8, C6H6, C8H10) y Ruido. Todos son de tipo double.
- Información meteorológica (Velocidad del viento, Dirección del viento, Temperatura, Humedad relativa, Presión, Radiación, Precipitación, Velocidad máxima del viento). Todos son de tipo double.
- Estacion (Avda. Francia, Boulevard Sur, Molino del Sol, Pista Silla, Politécnico, Viveros, Centro, Consellería Meteo, Nazaret Meteo, Puerto València). Es de tipo factor.

Valores faltantes

El estudio de los valores faltantes es esencial para garantizar que los análisis de datos sean sólidos y confiables. Como en el resultado previo hemos identificado valores faltantes, en este apartado procederemos a analizarlos. Primero, realizamos un resumen de la cantidad de valores faltantes en cada columna:

```
valores_faltantes <- sapply(rvvcca_d_horarios_2016_2020, function(x) sum(is.na(x)))
```

Podemos ver como hay variables con un gran número de valores faltantes, en especial, en los contaminantes atmosféricos y la información meteorológica. Esto es debido a que cada estación dispone de unos sensores distintos de medición de estos fenómenos. Además, el ruido también dispone de muchos valores faltantes, ya que depende de lo tranquila que sea cada estación.

Por tanto, para poder realizar un mejor análisis vamos a agrupar nuestro dataframe por `Estacion`:

Id	0	Velocidad del viento	7874
Fecha	0	Direccion del viento	6020
Dia de la semana	0	NH3	313088
Dia del mes	0	C7H8	313156
Hora	0	C6H6	313635
Estacion	0	Ruido	309508
PM1	286441	C8H10	313053
PM2.5	183359	Temperatura	9033
PM10	183354	Humedad relativa	16357
NO	88740	Presion	6276
NO2	71383	Radiacion	6814
NOx	88738	Precipitacion	7414
O3	87741	Velocidad maxima del viento	8045
SO2	91245	Fecha creacion	0
CO	236277		

Table 1: Número de valores faltantes.

```
grupos <- rrvcca_d_horarios_2016_2020 %>% group_by(Estacion)

lista_dataframes <- split(grupos, f = grupos$Estacion)

nombres_dataframes <- unique(rrvcca_d_horarios_2016_2020$Estacion)
lista_dataframes <- setNames(lista_dataframes, nombres_dataframes)
```

Limpiamos las columnas vacías de cada dataframe:

```
for (estacion in nombres_dataframes) {
  lista_dataframes[[estacion]] <- lista_dataframes[[estacion]] %>%
    select_if(~!all(is.na(.)))
}
```

Convertimos la lista de dataframes en dataframes independientes:

```
list2env(lista_dataframes, envir = .GlobalEnv)
```

```
## <environment: R_GlobalEnv>
```

Ahora, veamos que parámetros se han medido en cada estación:

```
# Crear una lista con los nombres de los parámetros a observar
parametros <- names(rrvcca_d_horarios_2016_2020)

# Crear un dataframe vacío con las zonas como filas y los parámetros como columnas
tabla_zonas <- data.frame(Zona = character(0), stringsAsFactors = FALSE)

# Iterar sobre los dataframes y extraer los nombres de las columnas
for (zona in 1:length(lista_dataframes)) {
  # Crear una fila con el nombre de la zona
  fila_zona <- data.frame(Zona = names(lista_dataframes)[zona], stringsAsFactors = FALSE)
```

```

# Iterar sobre los parámetros y agregar TRUE o FALSE según si existen en la zona
for (parametro in parametros) {
  fila_zona[[parametro]] <- parametro %in% colnames(lista_dataframes[[zona]])
}

# Unir la fila de la zona al dataframe
tabla_zonas <- rbind(tabla_zonas, fila_zona)
}

rownames(tabla_zonas) <- tabla_zonas$Zona
colnames(tabla_zonas)[17] <- "V viento"
colnames(tabla_zonas)[29] <- "V_máx viento"
tabla_zonas <- subset(tabla_zonas, select = -c(Zona,Id,Fecha,`Dia de la semana`,`Dia del mes`,`Hora`,`Estado`))
tabla_numeric <- as.data.frame(sapply(tabla_zonas, as.numeric))

# Crea el heatmap con colores azul y amarillo
pheatmap(tabla_numeric, color=hcl.colors(2,palette = "BluYl"),
  labels_row = rownames(tabla_zonas), cluster_rows = F,
  cluster_cols = F, legend_breaks = c(0,0.25,0.75, 1),
  legend_labels = c("", "No se mide", "Sí se mide", ""),
  border_color = "black")

```

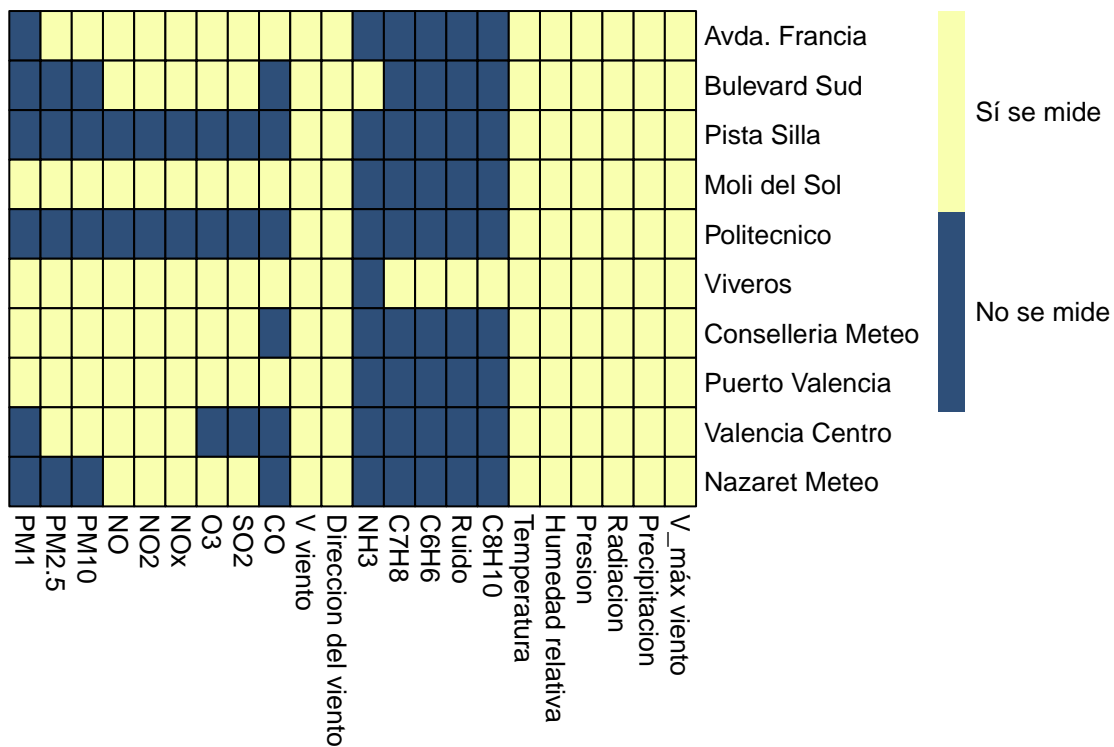


Figure 1: Variables medidas en cada estación.

Análisis de las variables

Análisis univariante

Hemos visto anteriormente que todos los contaminantes atmosféricos sí se miden en la estación Viveros. Por tanto, vamos a calcular los estadísticos básicos de los contaminantes atmosféricos para la estación Viveros:

```
resultados_totales <- Viveros %>%
  summarise(across(c("NO", "NO2", "NOx", "O3", "SO2", "CO"), list(mean = mean, sd = sd,
    median = median, IQR = IQR), na.rm = TRUE)) %>%
  pivot_longer(cols = -Estacion, names_to = "variable", values_to = "valor") %>%
  separate(variable, into = c("Contaminante", "stat"), sep = "_") %>%
  pivot_wider(names_from = "stat", values_from = "valor") %>%
  mutate(Estacion = "Viveros") %>%
  select(Contaminante, mean, sd, median, IQR) %>%
  arrange(Contaminante)

kable(resultados_totales, caption="Estadísticos básicos de la estación Viveros.")
```

Table 2: Estadísticos básicos de la estación Viveros.

Contaminante	mean	sd	median	IQR
CO	0.1735941	0.1138526	0.1	0.1
NO	19.7068219	28.4383802	10.0	20.0
NO2	33.0132081	23.6815240	28.0	33.0
NOx	63.0690382	62.9793882	44.0	63.0
O3	45.0142906	26.7762457	46.0	42.0
SO2	4.3111563	2.2229034	4.0	1.0

La tabla de estadísticas descriptivas para los contaminantes del aire destaca diferencias notables entre ellos. Mientras el monóxido de carbono (CO) muestra niveles bajos y poca variabilidad, el dióxido de nitrógeno (NO) exhibe una variabilidad significativa con una media más alta. El dióxido de nitrógeno (NO2) presenta una distribución asimétrica, y el óxido de nitrógeno (NOx) muestra una amplia dispersión de datos. El ozono (O3) tiene una distribución relativamente simétrica, y el dióxido de azufre (SO2) muestra concentración cercana a la media con moderada variabilidad.

—recolocar—

```
require("ggplot2")

dias_semana <- unique(Viveros$`Dia de la semana`)
par(mfrow = c(5, 2))

if (!dir.exists("./grafs")) {
  dir.create("./grafs")
}

library(dplyr)

orden_dias <- c("Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo")

df_media <- Viveros %>%
```

```
group_by(`Dia de la semana`, Hora) %>%
summarise(
  media_NO = mean(NO, na.rm = TRUE),
  media_NO2 = mean(NO2, na.rm = TRUE),
  media_NOx = mean(NOx, na.rm = TRUE),
  media_O3 = mean(O3, na.rm = TRUE),
  media_SO2 = mean(SO2, na.rm = TRUE),
  media_CO = mean(CO, na.rm = TRUE),
  media_C6H6 = mean(C6H6, na.rm = TRUE),
  media_C7H8 = mean(C7H8, na.rm = TRUE),
  media_C8H10 = mean(C8H10, na.rm = TRUE)
)
```

'summarise()' has grouped output by 'Dia de la semana'. You can override using
the '.groups' argument.

```
require("ggplot2")
dias_semana <- unique(Viveros$`Dia de la semana`)
par(mfrow = c(5, 2))

if (!dir.exists("./grafs")) {
  dir.create("./grafs")
}

orden_dias <- c("Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo")

for (dia in orden_dias) {
  df_media_dia <- df_media[df_media$`Dia de la semana` == dia, ]

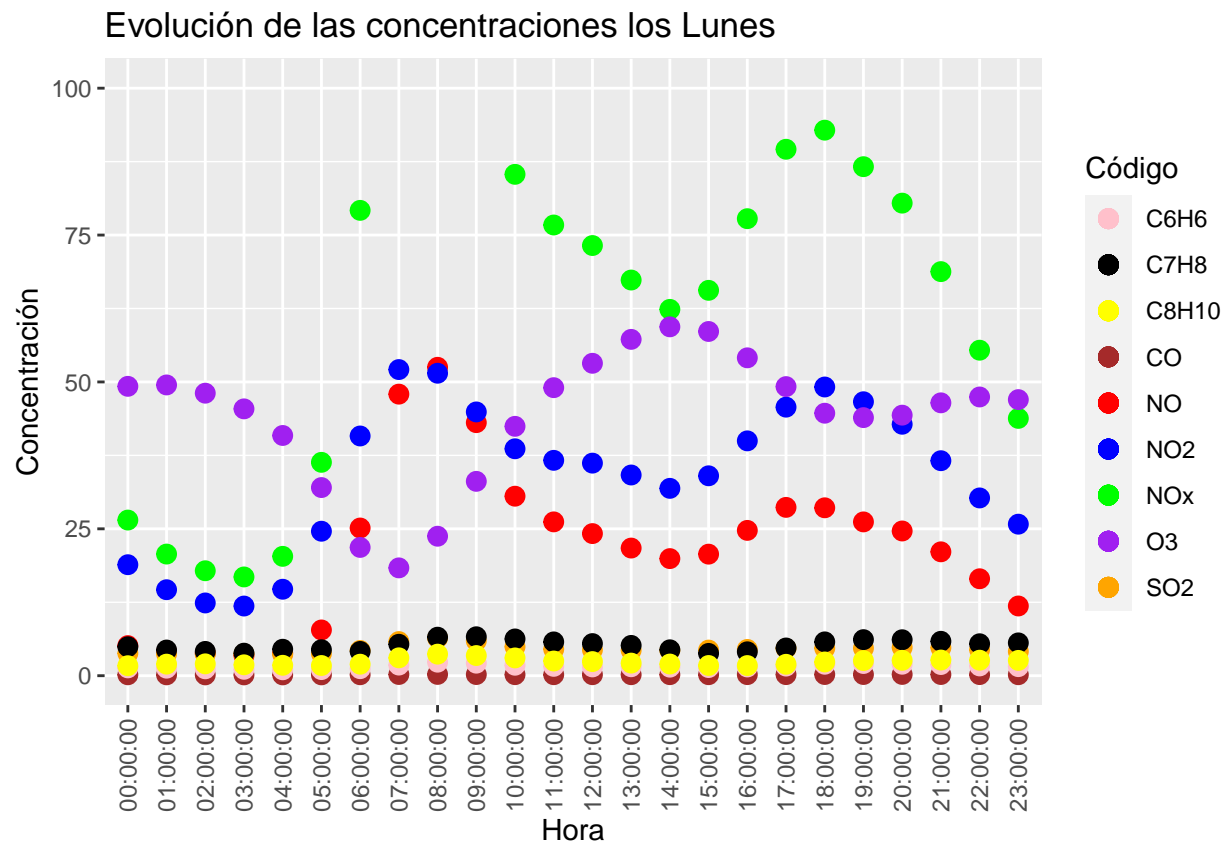
  if (nrow(df_media_dia) > 0) {
    # Ajusta el título para manejar correctamente las tildes y la s del plural
    titulo <- ifelse(dia == "Sabado", "Sábados",
                     ifelse(dia == "Domingo", "Domingos",
                             ifelse(dia == "Miercoles", "Miércoles", dia)))

    # Crea la gráfica con puntos
    p <- ggplot(df_media_dia, aes(x = Hora)) +
      geom_point(aes(y = media_NO, color = "NO"), size = 3) +
      geom_point(aes(y = media_NO2, color = "NO2"), size = 3) +
      geom_point(aes(y = media_NOx, color = "NOx"), size = 3) +
      geom_point(aes(y = media_O3, color = "O3"), size = 3) +
      geom_point(aes(y = media_SO2, color = "SO2"), size = 3) +
      geom_point(aes(y = media_CO, color = "CO"), size = 3) +
      geom_point(aes(y = media_C6H6, color = "C6H6"), size = 3) +
      geom_point(aes(y = media_C7H8, color = "C7H8"), size = 3) +
      geom_point(aes(y = media_C8H10, color = "C8H10"), size = 3) +
      ggtitle(paste("Evolución de las concentraciones los", titulo)) +
      xlab("Hora") +
      ylab("Concentración") +
      scale_color_manual(values = c("NO" = "red", "NO2" = "blue", "NOx" = "green", "O3" = "purple", "SO2" = "brown", "CO" = "black", "C6H6" = "pink", "C7H8" = "cyan", "C8H10" = "magenta")) +
      labs(color = "Código") +
      coord_cartesian(ylim = c(0, 100)) +
      theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) # Rotar las etiquetas del
```

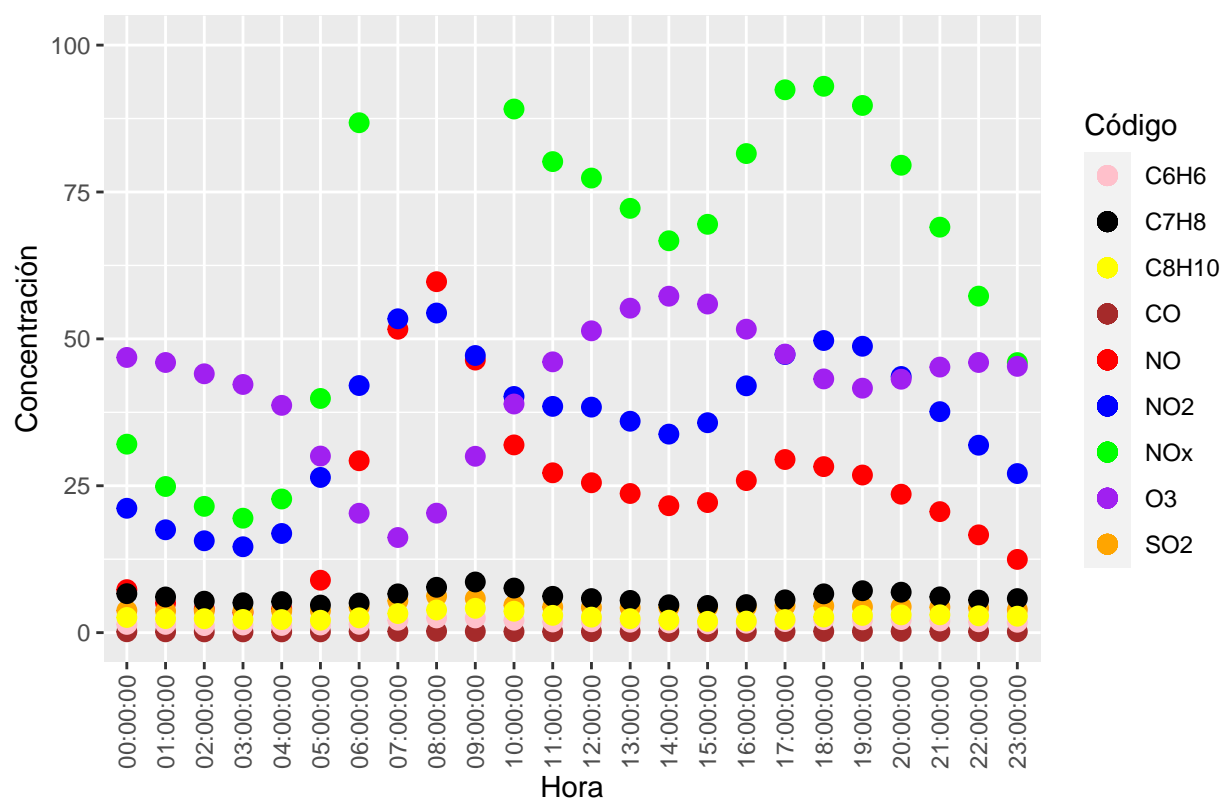
```

# Muestra la gráfica
print(p)
ggsave(paste("./grafs/grafico_", dia, ".png"), plot = p, device = "png", width = 8, height = 6, uni
}
}

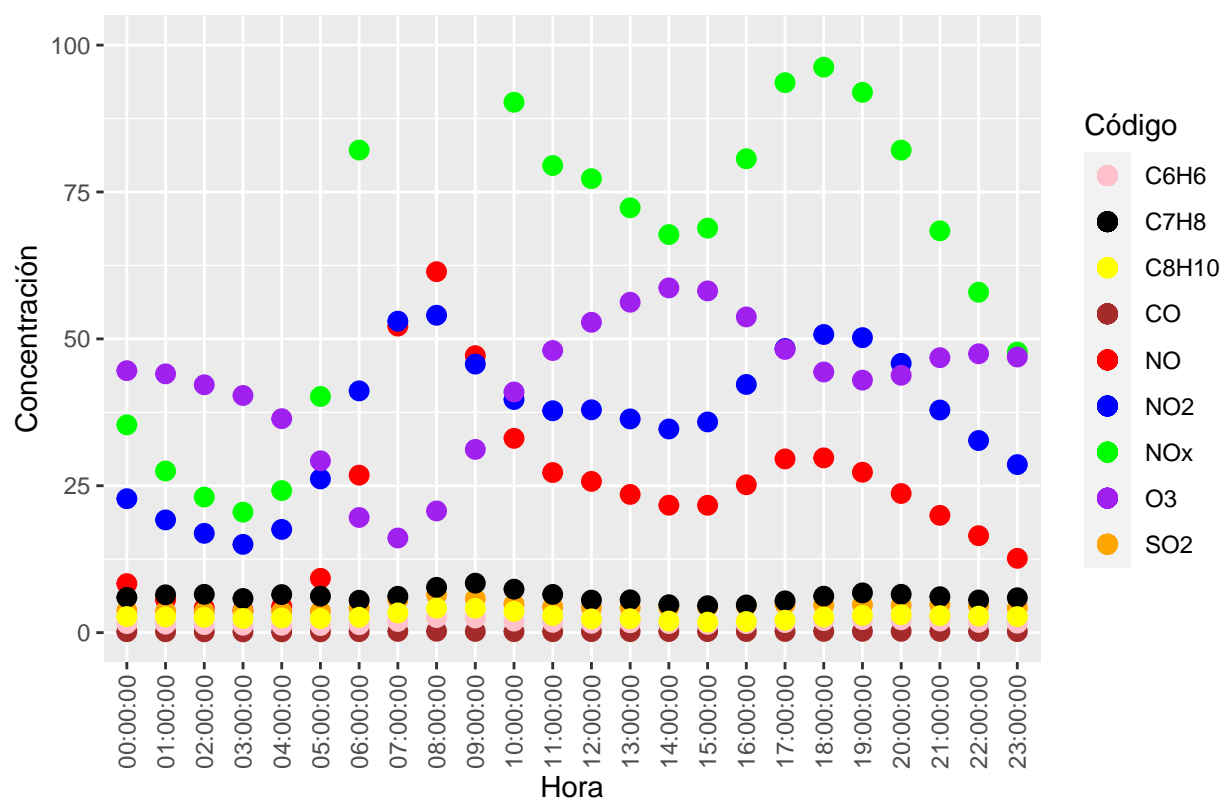
```



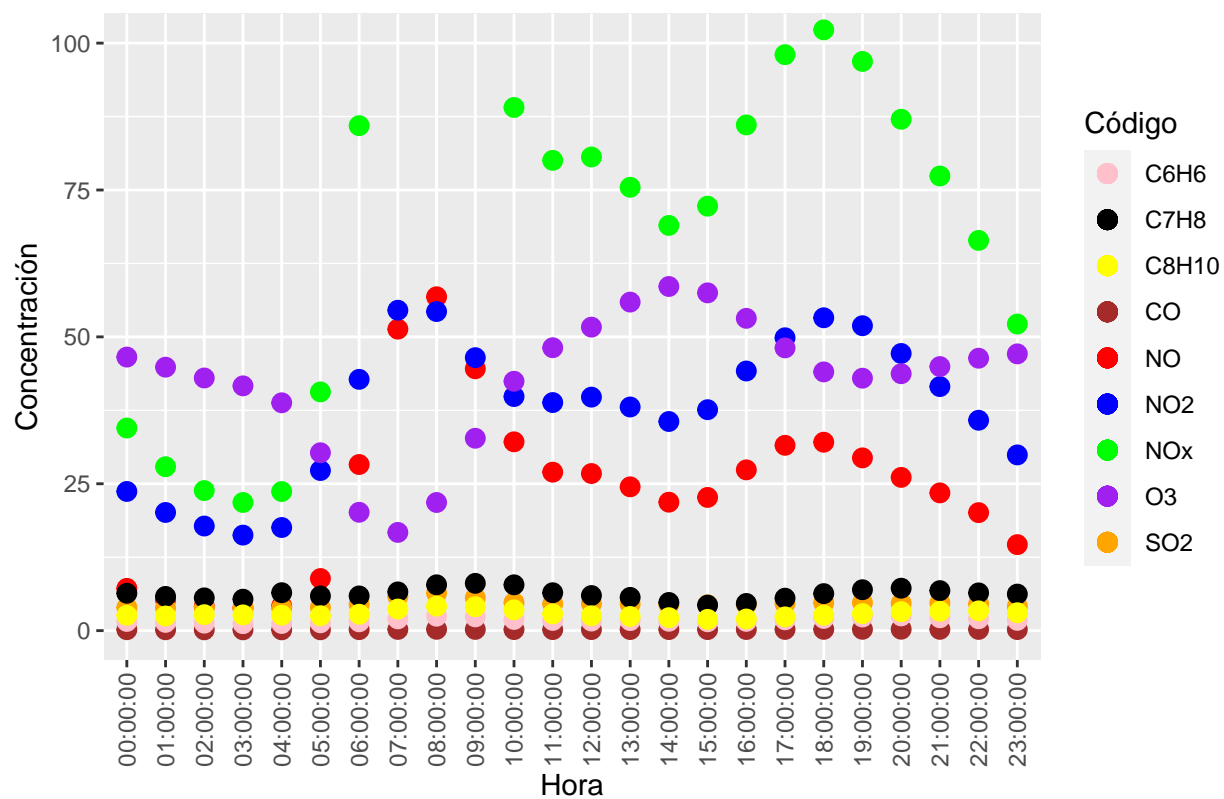
Evolución de las concentraciones los Martes



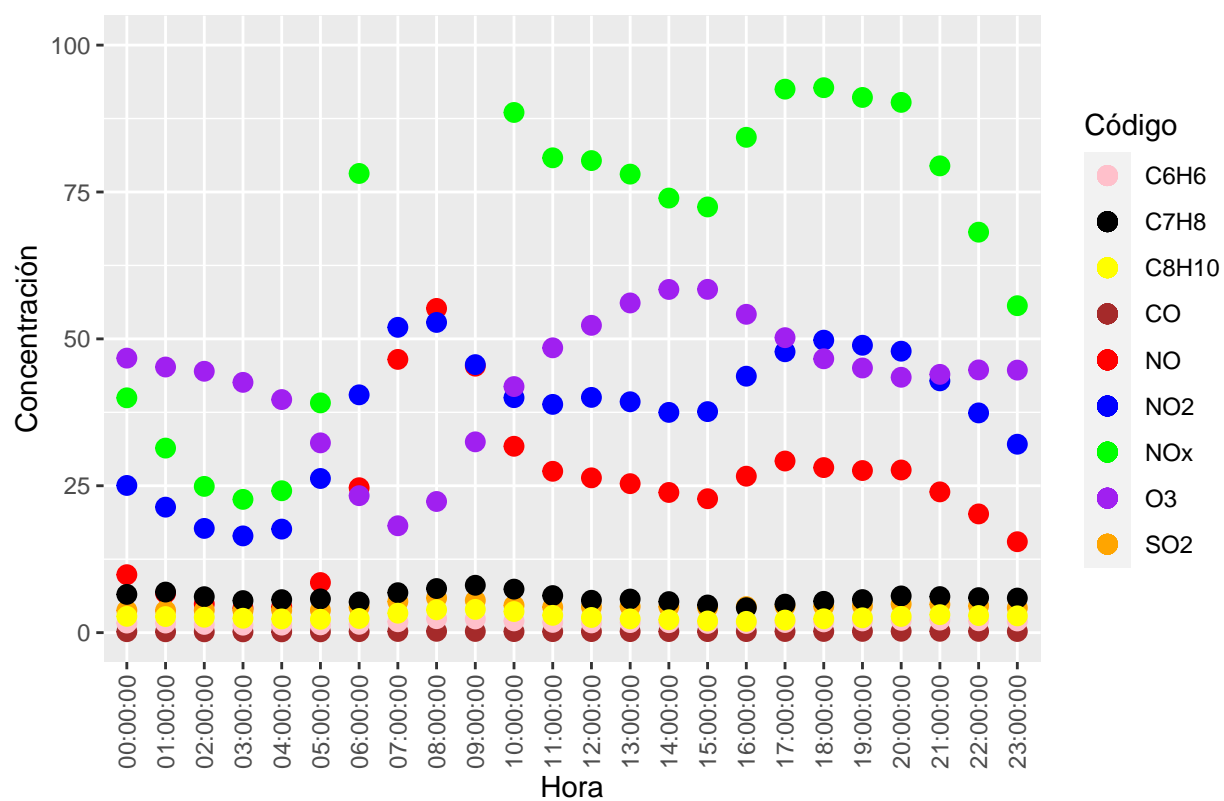
Evolución de las concentraciones los Miércoles



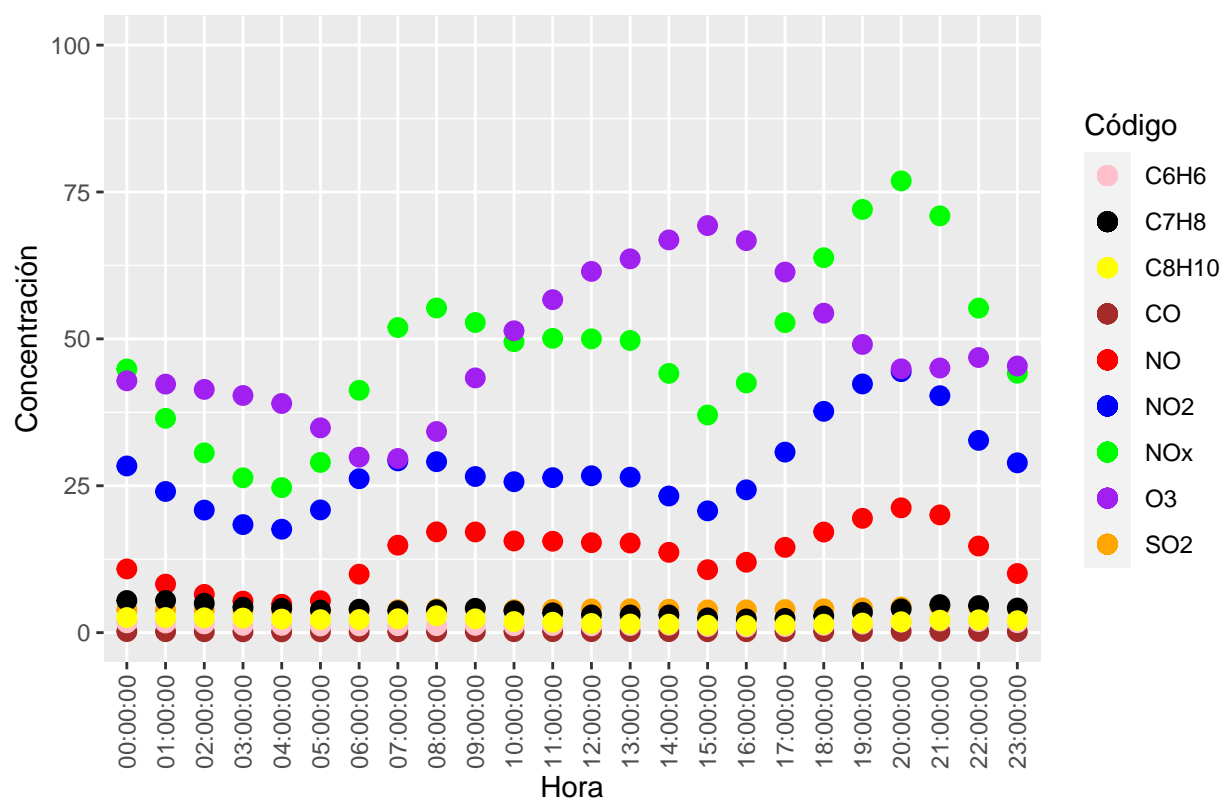
Evolución de las concentraciones los Jueves



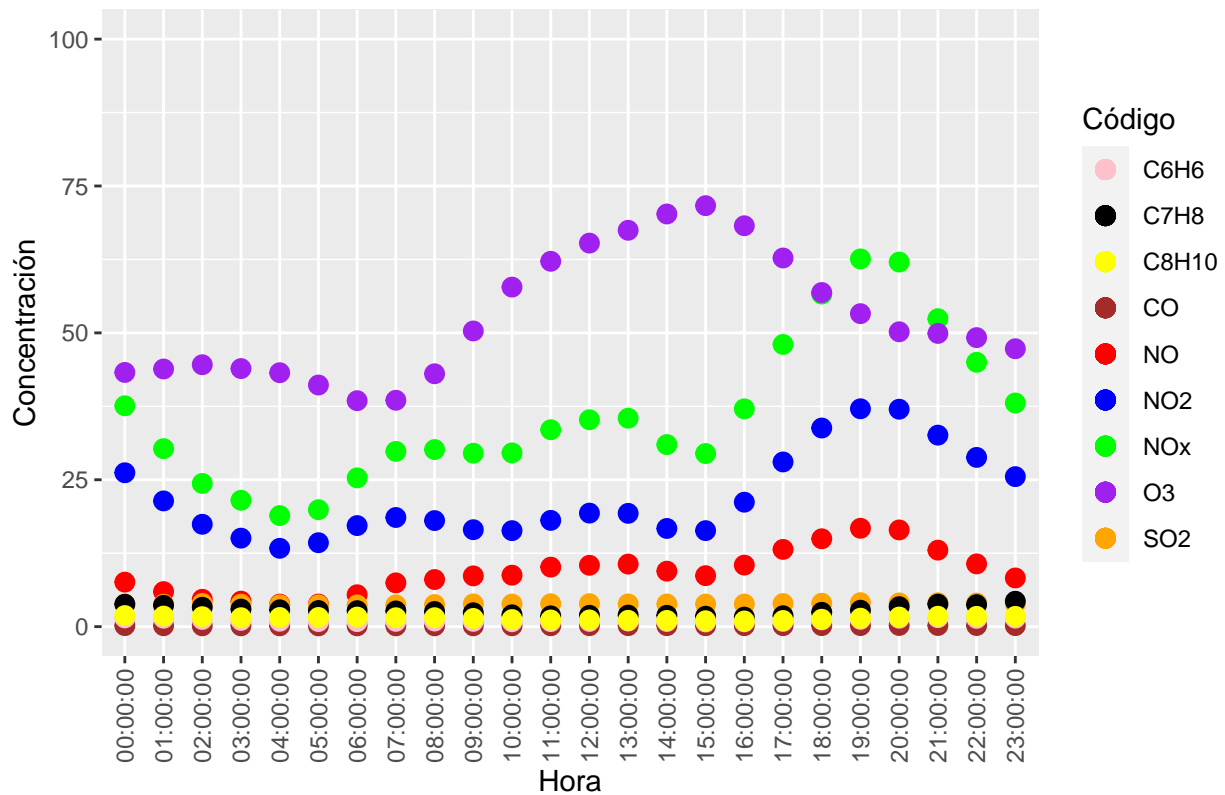
Evolución de las concentraciones los Viernes



Evolución de las concentraciones los Sábados



Evolución de las concentraciones los Domingos



```
require("dplyr")
```

```
Viveros$gases_total <- rowSums(Viveros[, c('NO', 'NO2', 'NOx', 'O3', 'SO2', 'CO', 'C6H6', 'C7H8', 'C8H10')])
```

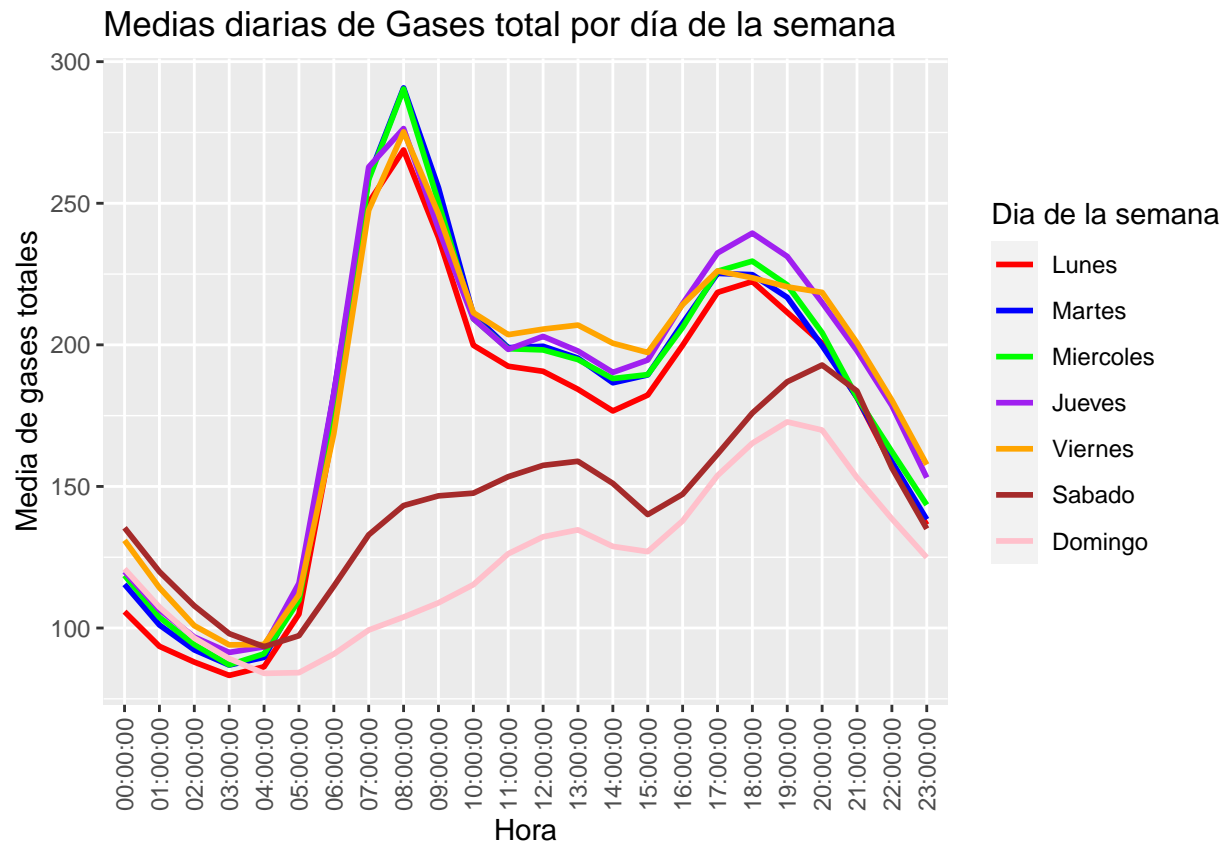
```
media_diaria <- Viveros %>%
  group_by(`Dia de la semana`, Hora) %>%
  summarise(media_gases_total = mean(gases_total, na.rm = TRUE))
```

```
## 'summarise()' has grouped output by 'Dia de la semana'. You can override using
## the '.groups' argument.
```

```
# Crea la gráfica de las medias diarias de 'gases_total' por día de la semana con puntos y dispersión h
p3<-ggplot(media_diaria, aes(x = Hora, y = media_gases_total, color = `Dia de la semana`, group = `Dia de la semana`)) +
  geom_line(size = 1) +
  ggtitle("Medias diarias de Gases total por día de la semana") +
  xlab("Hora") +
  ylab("Media de gases totales") +
  scale_color_manual(values = c("Lunes" = "red", "Martes" = "blue", "Miercoles" = "green", "Jueves" = "orange", "Viernes" = "purple", "Sabado" = "brown", "Domingo" = "pink")) +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1))
```

```
## Warning: Using 'size' aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use 'linewidth' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

```
print(p3)
```



```
ggsave("./grafs/medias_gases_total.png", p3, width = 10, height = 6, units = "in", dpi = 300)
```

—recolocar—

Análisis bivariante

```
#ggpairs(Viveros[, c('NO', 'NO2', 'NOx', 'O3', 'SO2', 'CO', 'C6H6', 'C7H8', 'C8H10')], na.rm=TRUE)
```

Análisis de outliers

A continuación, exploraremos la calidad del aire mediante boxplots que representan las concentraciones de dos sustancias fundamentales: dióxido de nitrógeno (NO₂) y ozono (O₃). Cada gráfico proporciona una visión detallada de la distribución de estas sustancias en las distintas estaciones de la red de vigilancia atmosférica de la ciudad de València.

```
generate_boxplot <- function(data, title) {
  boxplot(data, main = title, col = "lightpink", border = "darkblue")
}
```

```

par(mfrow = c(2, 4), mar = c(4, 4, 2, 1), cex.main = 0.8)

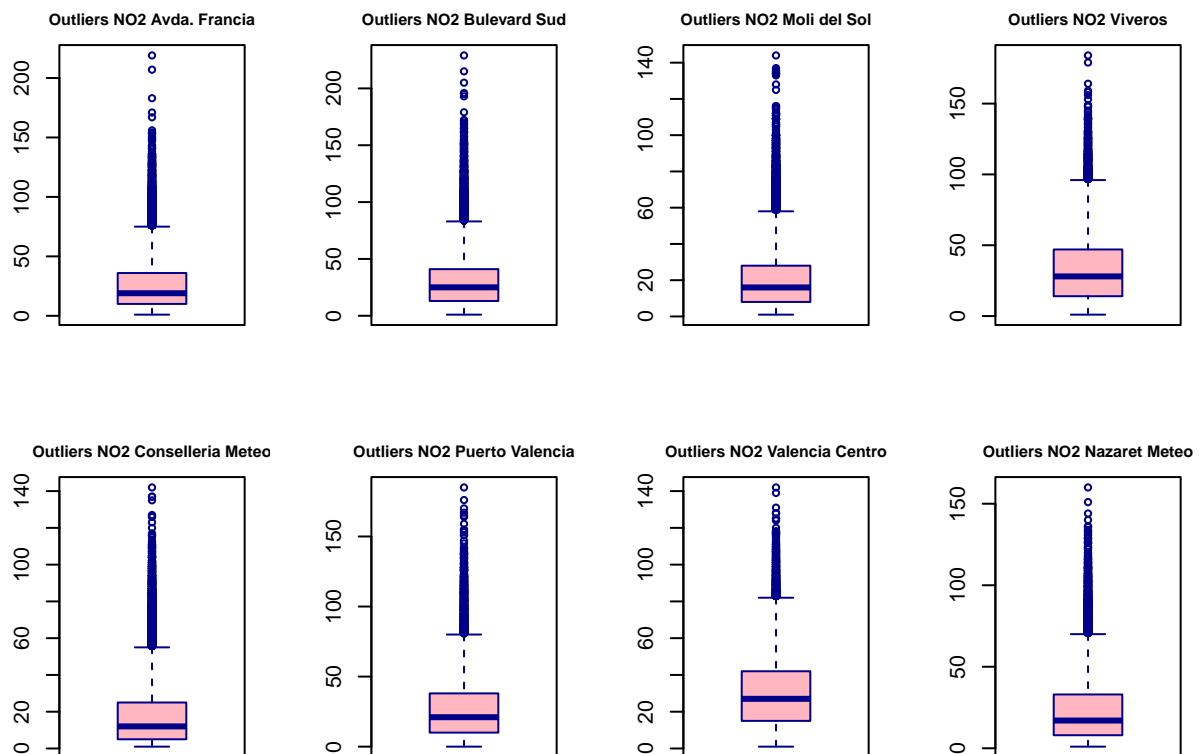
for (nombre_df in nombres_dataframes) {
  df <- get(nombre_df)

  if ("NO2" %in% names(df) && length(df$NO2) > 0) {
    generate_boxplot(df$NO2, paste("Outliers NO2", nombre_df, sep = " "))
  } else {
    cat("No se encontró la columna 'NO2' en", nombre_df, "\n")
  }
}

```

No se encontró la columna 'NO2' en Pista Silla

No se encontró la columna 'NO2' en Politecnico



Observamos una notable presencia de outliers en la variable NO2 en todas las estaciones, destacando en la parte superior de los boxplots. Estos boxplots son estrechos, lo que sugiere que la mayor parte de los datos se concentra en un rango reducido de concentraciones de dióxido de nitrógeno. Sin embargo, la presencia de numerosos outliers en la parte superior indica que a menudo se generan concentraciones inusualmente altas de esta sustancia en el aire.

```

par(mfrow = c(2, 4), mar = c(4, 4, 2, 1), cex.main = 0.8)

for (nombre_df in nombres_dataframes) {

```

```

df <- get(nombre_df)

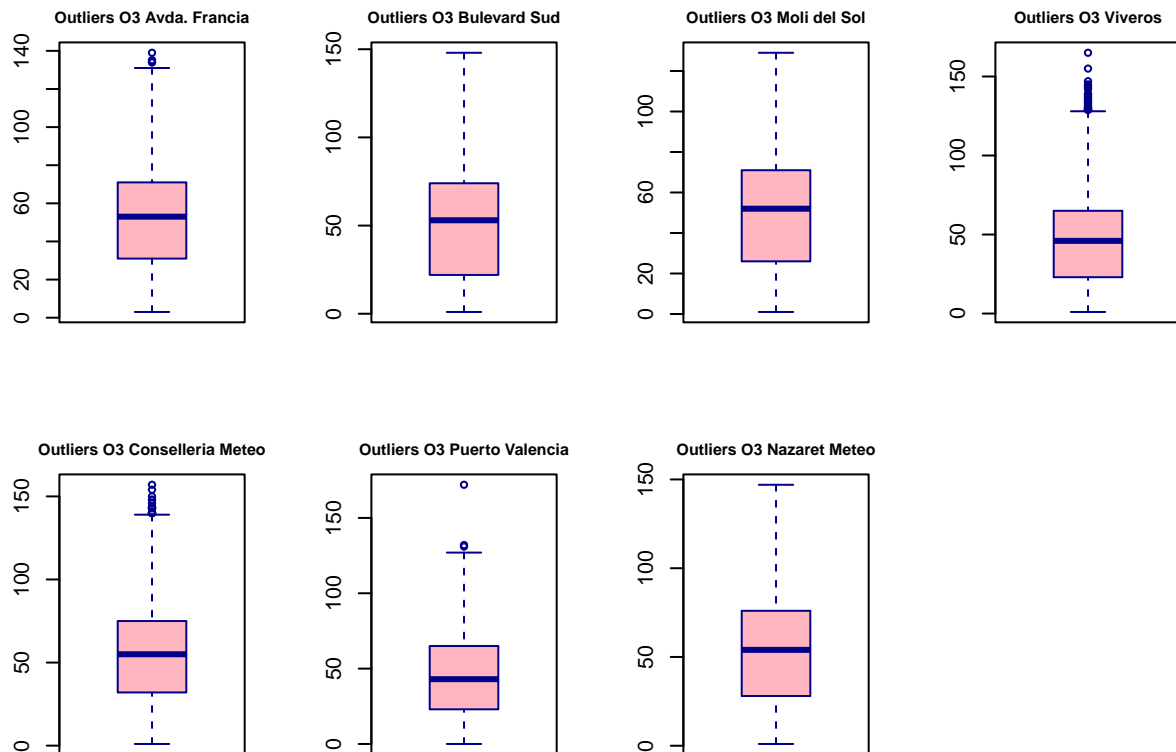
if ("O3" %in% names(df) && length(df$O3) > 0) {
  generate_boxplot(df$O3, paste("Outliers O3", nombre_df, sep = " "))
} else {
  cat("No se encontró la columna 'O3' en", nombre_df, "\n")
}
}

```

```
## No se encontró la columna 'O3' en Pista Silla
```

```
## No se encontró la columna 'O3' en Politecnico
```

```
## No se encontró la columna 'O3' en Valencia Centro
```



En contraste, la presencia de outliers en la variable de ozono es menos pronunciada. Aunque en las estaciones de Viveros y Consellería Meteo se observa una mayor proporción de outliers, en general, los valores presentan una distribución más uniforme. Los boxplots son más anchos, indicando una dispersión más amplia de las concentraciones de ozono.

También hemos considerado oportuno implementar funciones específicas para la identificación de outliers utilizando distintos métodos estadísticos, como la regla de 3 sigma, el identificador Hampel y la regla de percentiles. Este es otro modo de resaltar valores que podrían considerarse atípicos y podemos comparar con los resultados obtenidos mediante el método boxplot.

Función para la regla de 3 sigma:


```
reglasigma <- function(x) {
  media <- mean(x, na.rm = TRUE)
  desviacion <- sd(x, na.rm = TRUE)
  umbral_superior <- media + 3 * desviacion
  umbral_inferior <- media - 3 * desviacion
  outliers <- x[x > umbral_superior | x < umbral_inferior]
  return(outliers)
}
```

Función para el identificador Hampel:

```
reglahampel <- function(x) {
  mediana <- median(x, na.rm = TRUE)
  mad <- mad(x, na.rm = TRUE)
  umbral_superior <- mediana + 3 * mad
  umbral_inferior <- mediana - 3 * mad
  outliers <- x[x > umbral_superior | x < umbral_inferior]
  return(outliers)
}
```

Función para la regla del boxplot:

```
reglaboxplot <- function(x) {
  cuartil_75 <- quantile(x, 0.75, na.rm = TRUE)
  cuartil_25 <- quantile(x, 0.25, na.rm = TRUE)
  iqr <- cuartil_75 - cuartil_25
  umbral_superior <- cuartil_75 + 1.5 * iqr
  umbral_inferior <- cuartil_25 - 1.5 * iqr
  outliers <- x[x > umbral_superior | x < umbral_inferior]
  return(outliers)
}
```

Función para la regla de percentiles:

```
reglapercentil <- function(x) {
  percentil_5 <- quantile(x, 0.05, na.rm = TRUE)
  percentil_95 <- quantile(x, 0.95, na.rm = TRUE)
  outliers <- x[x < percentil_5 | x > percentil_95]
  return(outliers)
}
```

```
detectar_outliers <- function(x) {
  sol <- list(
    r_sigma = data.frame(Variable = colnames(x), Metodo = "Regla 3 sigma",
      Outliers = sapply(x, function(col) length(reglasigma(col)))),
    r_hampel = data.frame(Variable = colnames(x), Metodo = "Identificador Hampel",
      Outliers = sapply(x, function(col) length(reglahampel(col)))),
    r_boxplot = data.frame(Variable = colnames(x), Metodo = "Regla Boxplot",
      Outliers = sapply(x, function(col) length(reglaboxplot(col)))),
    r_percentiles = data.frame(Variable = colnames(x), Metodo = "Regla Percentiles",
      Outliers = sapply(x, function(col) length(reglapercentil(col))))
  )
}
```

```

    return(sol)
}

columnas_a_excluir <- c(1:6, ncol(df))
col <- setdiff(names(df), names(df)[columnas_a_excluir])
outliers_resultados <- detectar_outliers(Viveros[col])

plots <- lapply(outliers_resultados, function(result) {
  ggplot(result, aes(x = Variable, y = Outliers, fill = Variable)) +
    geom_bar(stat = "identity") +
    labs(title = unique(result$Metodo),
         x = "Variable",
         y = "Cantidad de Outliers") +
    theme_minimal() + theme(axis.text.x = element_blank(), legend.position = "none")
})

grid.arrange(grobs = plots, ncol = 2)

```

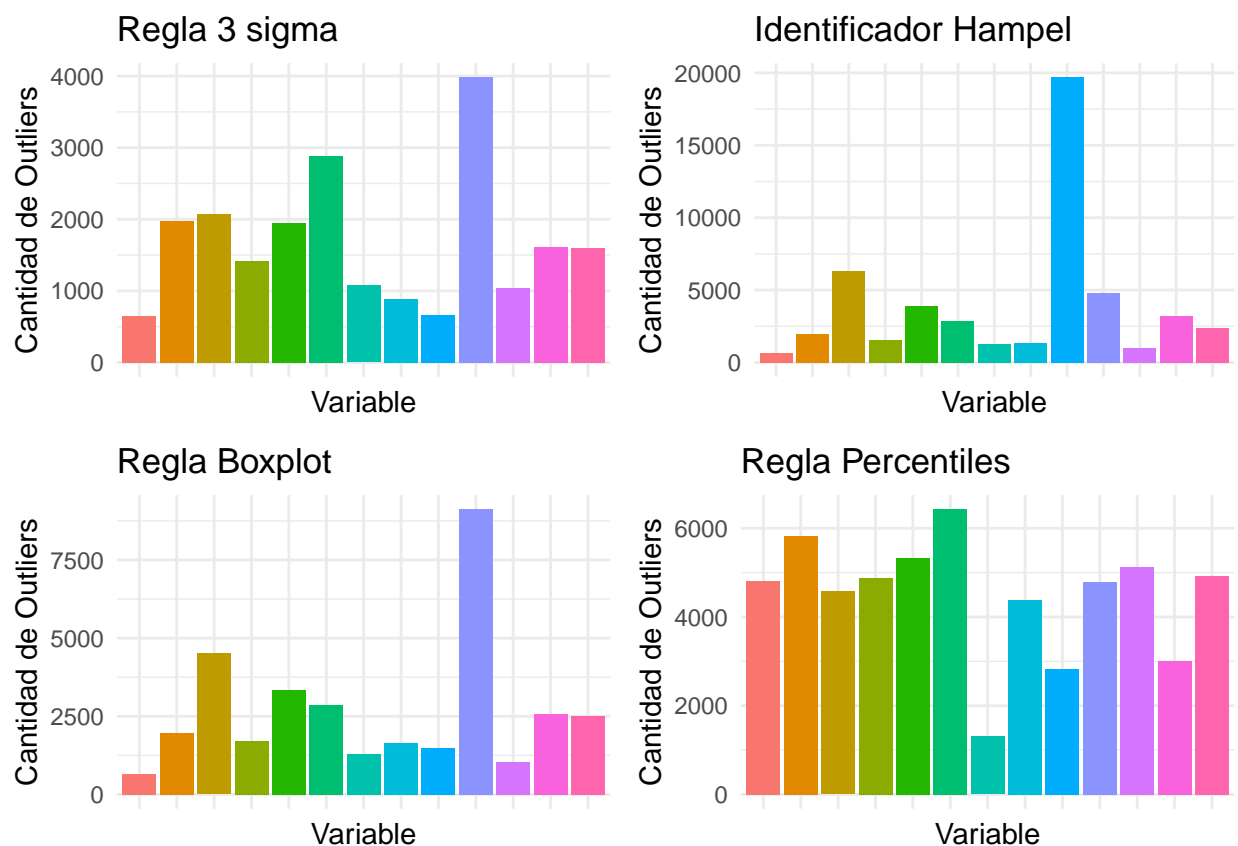


Figure 2: Gráficos de barras para detectar outliers.

La información resultante se presenta de manera clara y comparativa a través de gráficos de barras. Cada barra en estos gráficos representa la cantidad de outliers detectados para una variable específica, empleando diferentes métodos de detección. En este caso, lo hemos aplicado sobre los datos de la estación Viveros.

Conclusiones