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, Bulevard Sur, Molino del Sol, Pista Silla, Politécnico, Viveros, Centro, Conselleria 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)))
kable(valores_faltantes)
```

	x
Id	0
Fecha	0

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

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 otra parte, vemos en la gráfica **??** que los contaminantes atmosféricos y el ruido son los que proporcionalmente tienen más valores faltantes.

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

```
grupos <- rvvcca_d_horarios_2016_2020 %>% group_by(Estacion)

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

nombres_dataframes <- unique(rvvcca_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:

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

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

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

```r
# Crear una lista con los nombres de los parámetros a observar
parametros <- names(rvvcca_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,Esta
tabla_numeric <- as.data.frame(sapply(tabla_zonas, as.numeric))
```

```r
# 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")
```

Análisis de las variables

Análisis univariante

Hemos visto anteriormente que la información meteorológica sí se mide en todas las estaciones, lo cual es relevante para saber las condiciones meteorológicas de cada estación. Por tanto, vamos a calcular los estadísticos básicos de la información meteorológica más importante según la estación:

```r
calcular_estadisticas_estacion <- function(estacion_data, estacion_nombre) {
  estacion_data %>%
    summarise(across(c("Temperatura", "Presion", "Radiacion"), list(mean = mean, sd = sd,
```

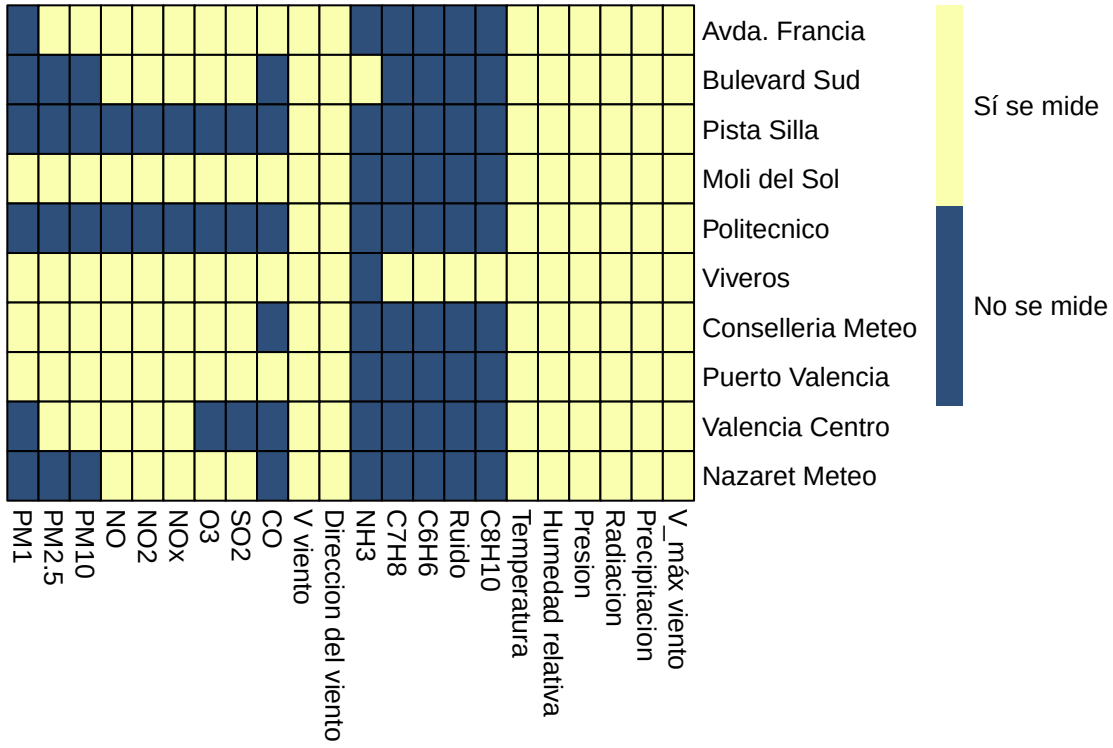


Figure 1: Variables medidas en cada estación.

```
      median = median, IQR = IQR), na.rm = TRUE)) %>%
    pivot_longer(cols = -Estacion, names_to = "variable", values_to = "valor") %>%
    separate(variable, into = c("Meteorologia", "stat"), sep = "_") %>%
    pivot_wider(names_from = "stat", values_from = "valor") %>%
    mutate(Estacion = estacion_nombre) %>%
    select(Meteorologia, Estacion, mean, sd, median, IQR)
}
resultados_totales <- lapply(nombres_dataframes, function(estacion) {
  calcular_estadisticas_estacion(lista_dataframes[[estacion]], estacion)
}) %>%
  bind_rows() %>%
  arrange(Meteorologia)
kable(resultados_totales)
```

Meteorologia	Estacion	mean	sd	median	IQR
Presion	Avda. Francia	1004.26881	6.669400	1004.0	8.0
Presion	Bulevard Sud	1004.26881	6.669400	1004.0	8.0
Presion	Pista Silla	1004.26881	6.669400	1004.0	8.0
Presion	Moli del Sol	1004.68799	6.550656	1004.0	8.0
Presion	Politecnico	1004.26881	6.669400	1004.0	8.0
Presion	Viveros	1004.26881	6.669400	1004.0	8.0
Presion	Conselleria Meteo	1004.26881	6.669400	1004.0	8.0
Presion	Puerto Valencia	1004.32024	6.412144	1004.0	7.0
Presion	Valencia Centro	1004.22302	6.762324	1004.0	9.0

Meteorologia	Estacion	mean	sd	median	IQR
Presion	Nazaret Meteo	1014.11387	6.526670	1014.0	8.0
Radiacion	Avda. Francia	201.97084	286.027377	10.0	367.0
Radiacion	Bulevard Sud	201.97084	286.027377	10.0	367.0
Radiacion	Pista Silla	201.97084	286.027377	10.0	367.0
Radiacion	Moli del Sol	203.06941	287.423969	10.0	369.0
Radiacion	Politecnico	201.97084	286.027377	10.0	367.0
Radiacion	Viveros	201.97084	286.027377	10.0	367.0
Radiacion	Conselleria Meteo	201.97084	286.027377	10.0	367.0
Radiacion	Puerto Valencia	201.28410	284.415913	9.0	369.0
Radiacion	Valencia Centro	199.89748	288.576920	8.0	359.0
Radiacion	Nazaret Meteo	177.59931	300.964471	0.0	288.0
Temperatura	Avda. Francia	18.28889	6.126433	18.1	9.8
Temperatura	Bulevard Sud	18.28889	6.126433	18.1	9.8
Temperatura	Pista Silla	18.28889	6.126433	18.1	9.8
Temperatura	Moli del Sol	18.35176	6.148472	18.2	9.9
Temperatura	Politecnico	18.28889	6.126433	18.1	9.8
Temperatura	Viveros	18.28889	6.126433	18.1	9.8
Temperatura	Conselleria Meteo	18.28889	6.126433	18.1	9.8
Temperatura	Puerto Valencia	18.16072	6.149404	18.0	9.9
Temperatura	Valencia Centro	18.49023	6.167002	17.9	9.7
Temperatura	Nazaret Meteo	19.85981	6.261606	19.8	10.2

En cuanto a la presión atmosférica, se observa que todas las estaciones tienen una media cercana a 1004, con variabilidad baja representada por la desviación estándar y la amplitud intercuartílica (IQR). En cuanto a la radiación, se observa una variabilidad significativa en los valores, con Avda. Francia mostrando una media considerablemente más baja que otras estaciones. Por último, en temperatura, la estación Nazaret Meteo presenta valores más altos en comparación con las demás, evidenciando una mayor variabilidad en este parámetro.

—-recolocar———

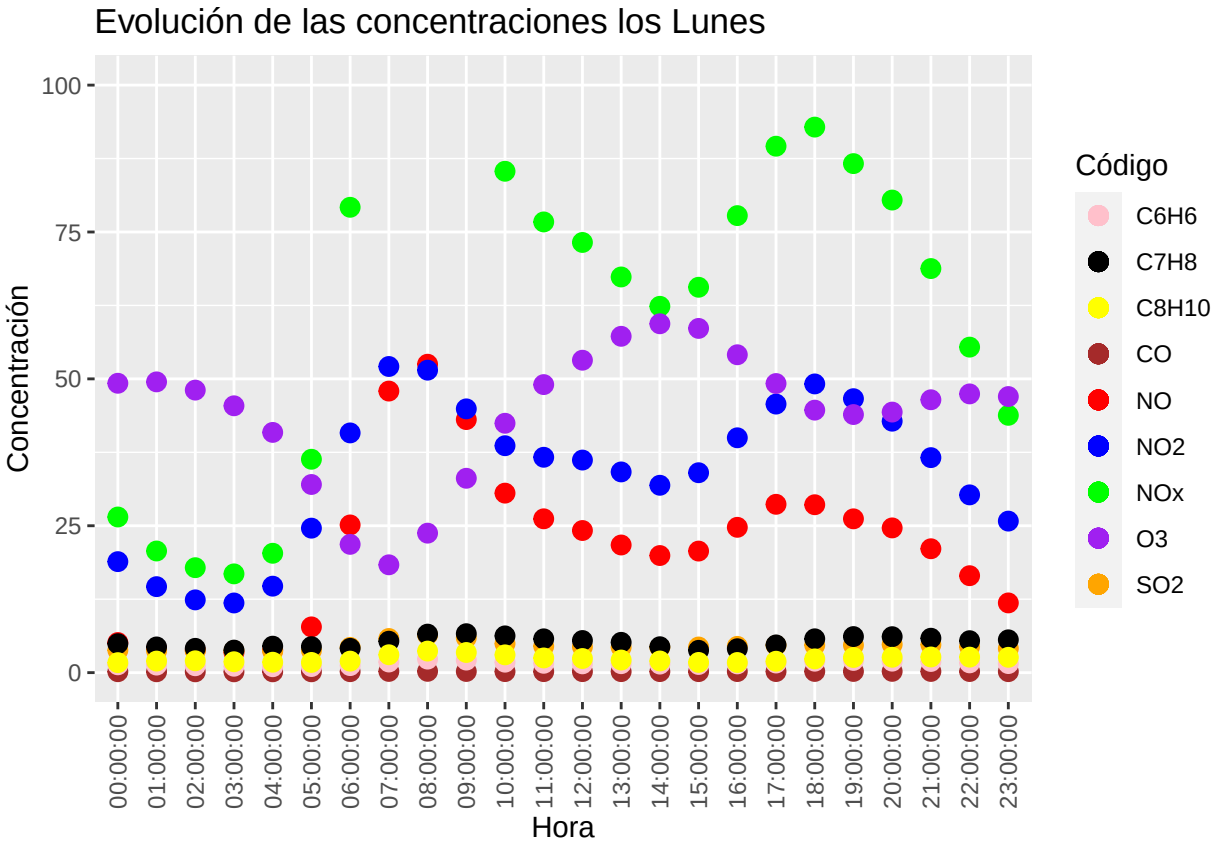
```
require("ggplot2")

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

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

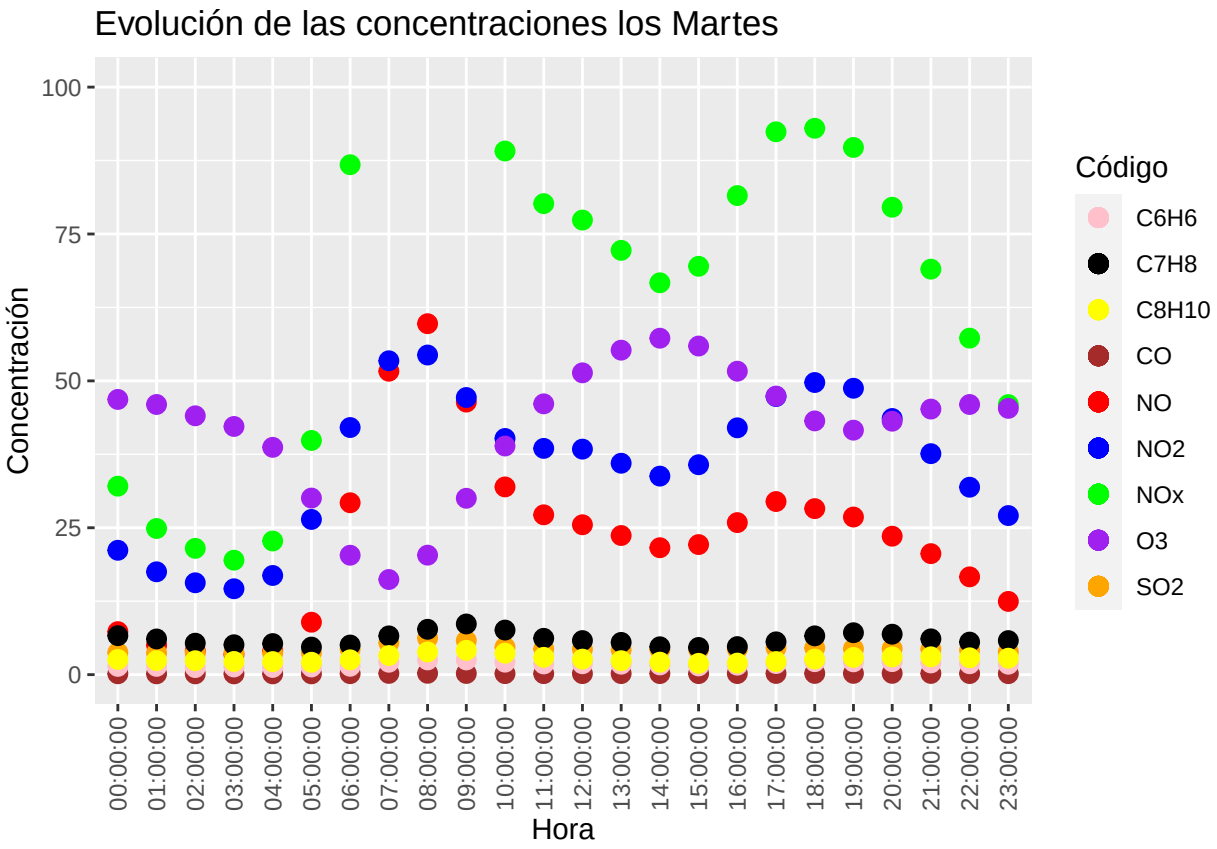
```
library(dplyr)

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),
```

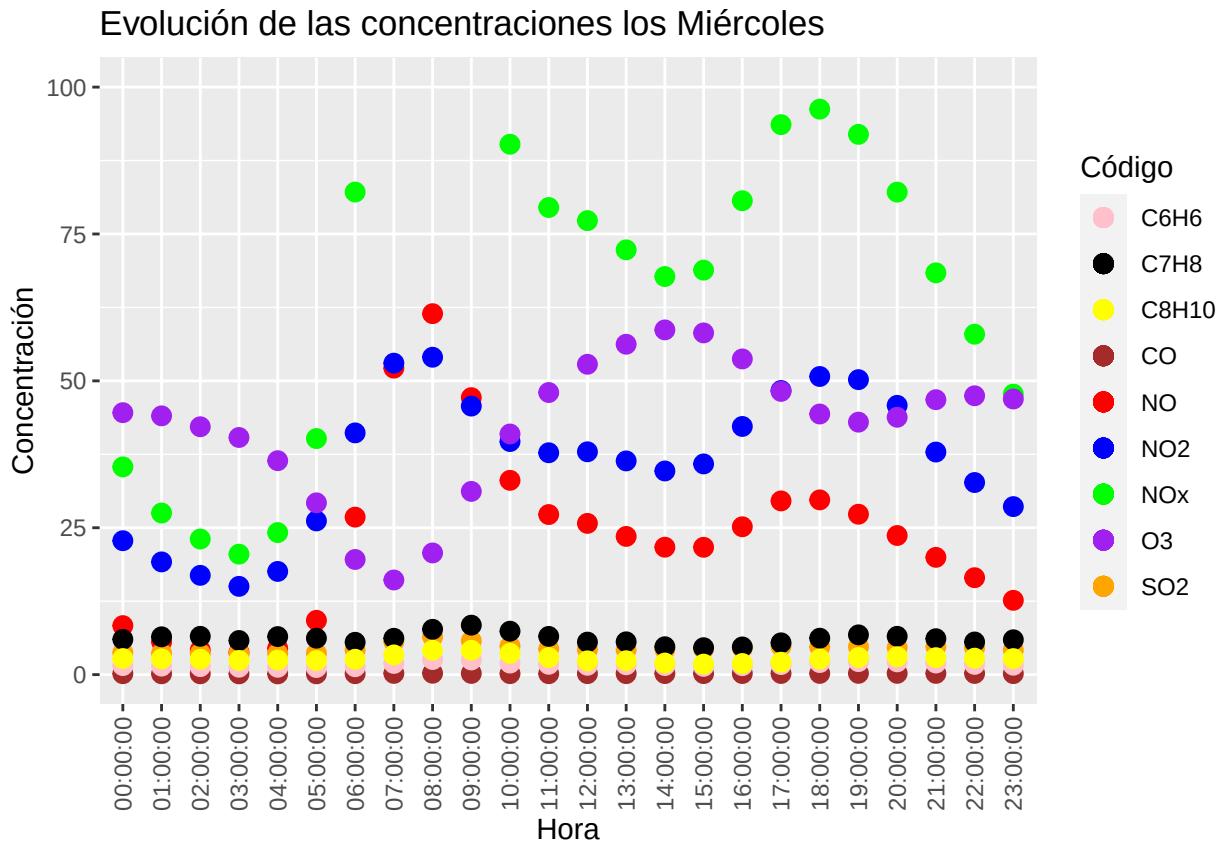
```
    media_CO = mean(CO, na.rm = TRUE),
    media_C6H6 = mean(C6H6, na.rm = TRUE),
    media_C7H8 = mean(C7H8, na.rm = TRUE),
    media_C8H10 = mean(C8H10, na.rm = TRUE)
  )
```

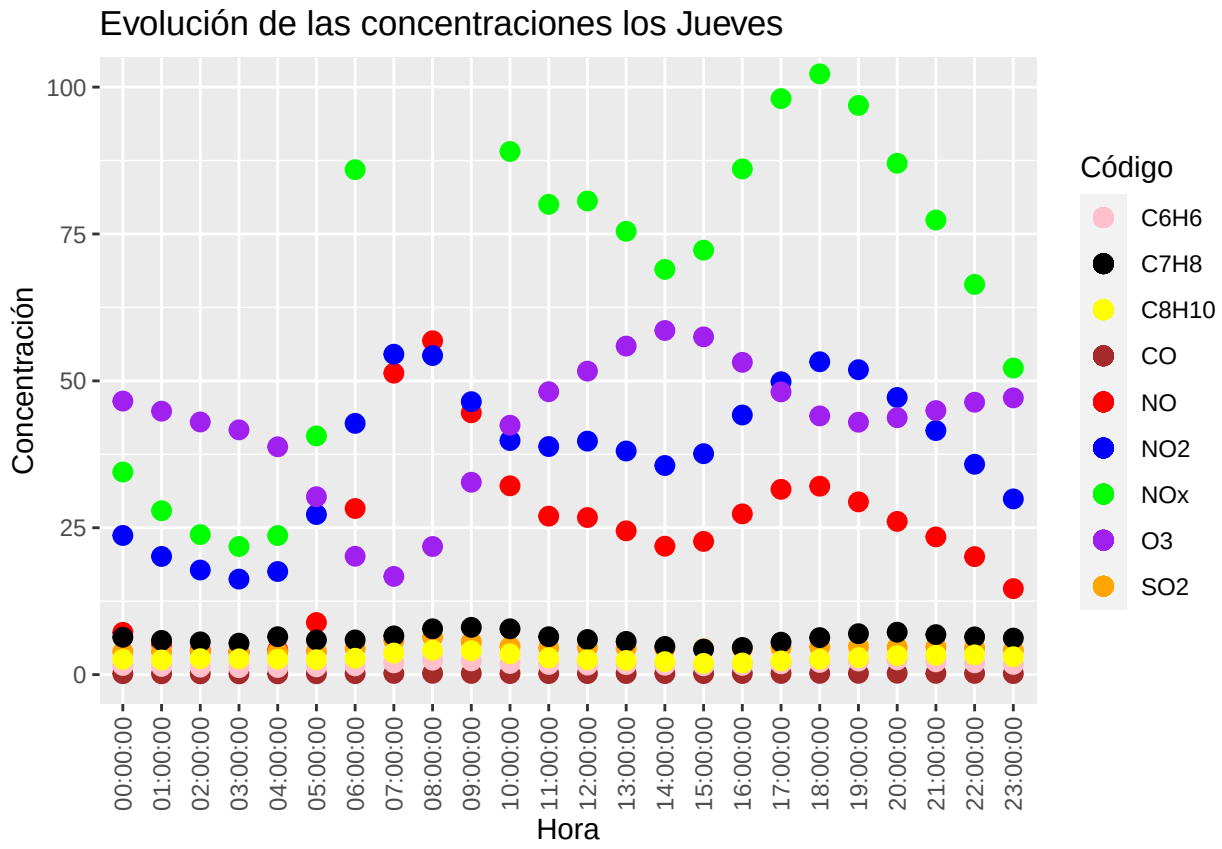
```
## `summarise()` has grouped output by 'Dia de la semana'. You can override using
## the `.groups` argument.
```

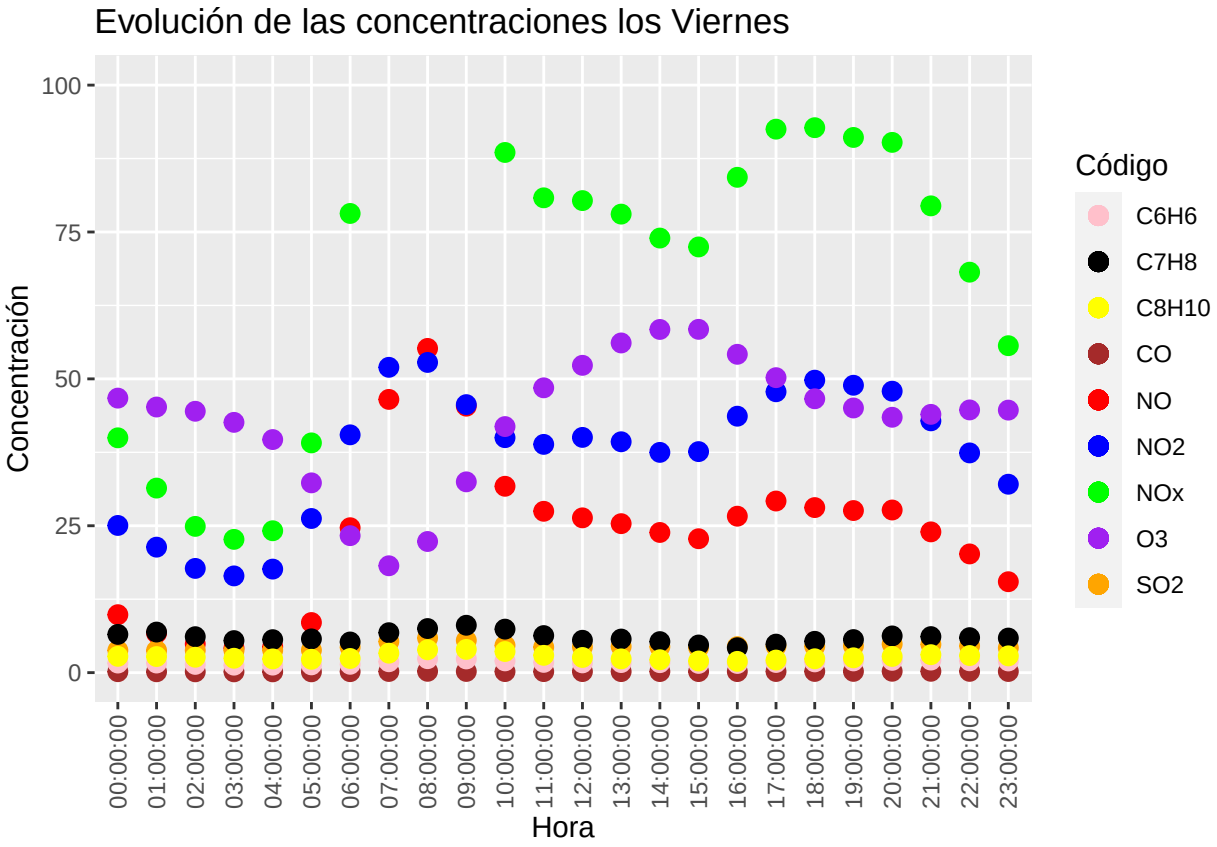
```
require("ggplot2")
dias_semana <- unique(Viveros$`Dia de la semana`)
par(mfrow = c(5, 2))

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", "SO
      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
  }
}
```

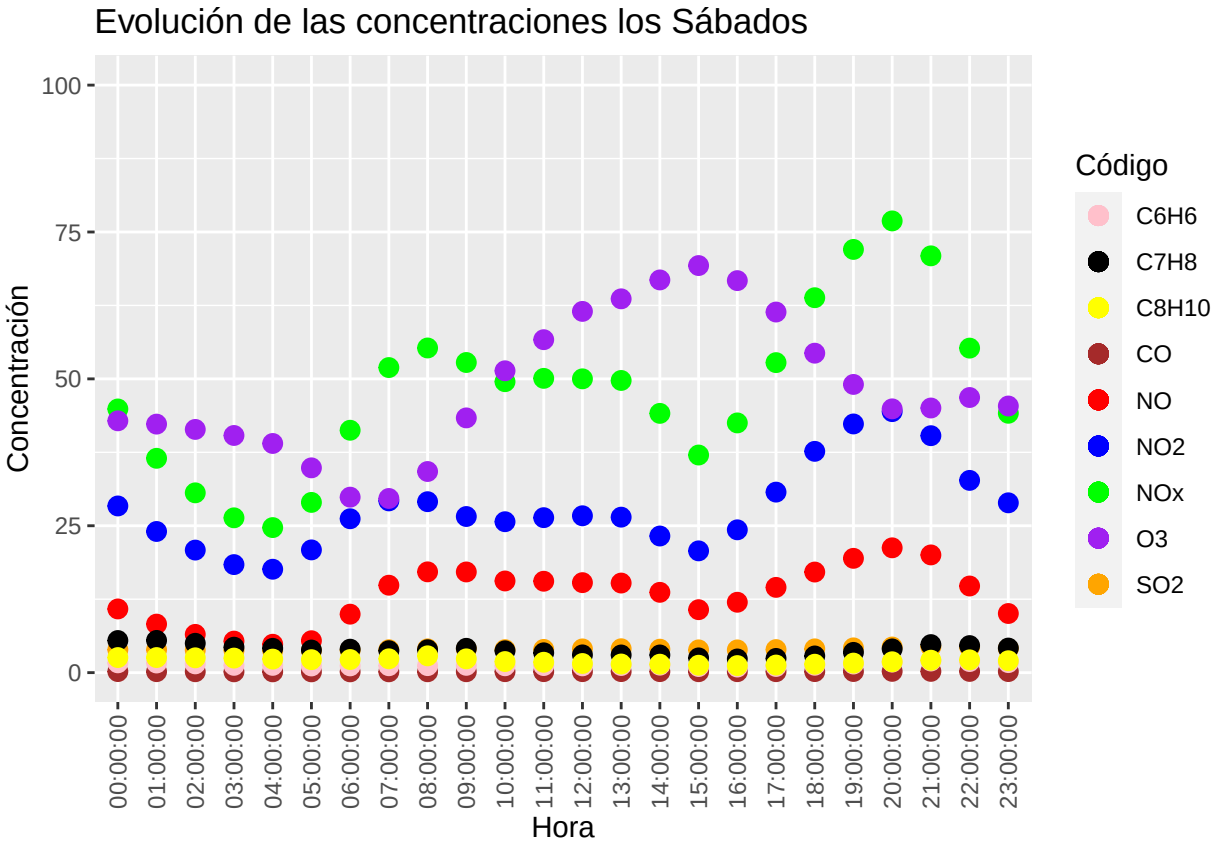

Evolución de las concentraciones los Lunes
Concentración
100
75
50
25
0
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:00
10:00:00
11:00:00
12:00:00
13:00:00
14:00:00
15:00:00
16:00:00
17:00:00
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
Hora
Código
C6H6
C7H8
C8H10
CO
NO
NO2
NOx
O3
SO2

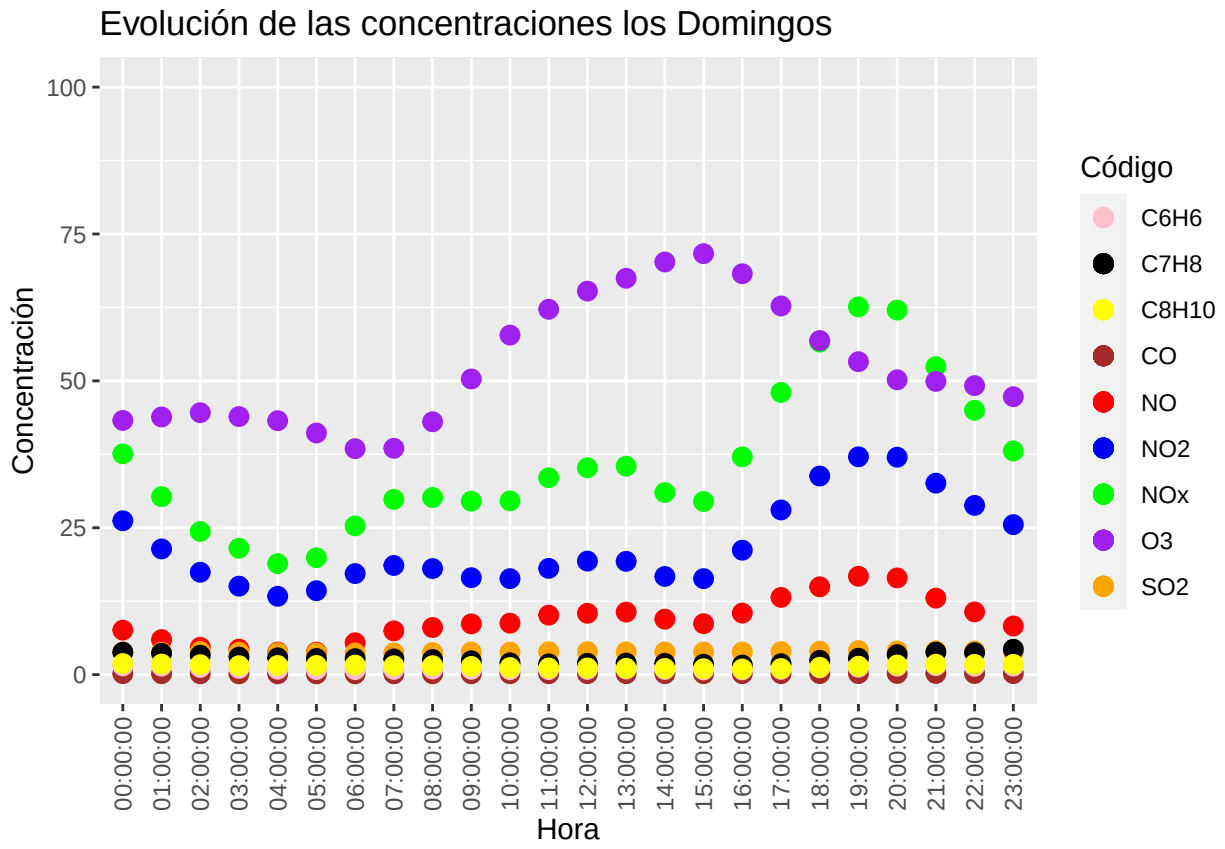

Evolución de las concentraciones los Martes
Concentración
100
75
50
25
0
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:00
10:00:00
11:00:00
12:00:00
13:00:00
14:00:00
15:00:00
16:00:00
17:00:00
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
Hora
Código
C6H6
C7H8
C8H10
CO
NO
NO2
NOx
O3
SO2


Evolución de las concentraciones los Miércoles
Concentración
100
75
50
25
0
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:00
10:00:00
11:00:00
12:00:00
13:00:00
14:00:00
15:00:00
16:00:00
17:00:00
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
Hora
Código
C6H6
C7H8
C8H10
CO
NO
NO2
NOx
O3
SO2


Evolución de las concentraciones los Jueves
Concentración
100
75
50
25
0
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:00
10:00:00
11:00:00
12:00:00
13:00:00
14:00:00
15:00:00
16:00:00
17:00:00
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
Hora
Código
C6H6
C7H8
C8H10
CO
NO
NO2
NOx
O3
SO2


Evolución de las concentraciones los Viernes
Concentración
100
75
50
25
0
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:00
10:00:00
11:00:00
12:00:00
13:00:00
14:00:00
15:00:00
16:00:00
17:00:00
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
Hora
Código
C6H6
C7H8
C8H10
CO
NO
NO2
NOx
O3
SO2


Evolución de las concentraciones los Sábados
Concentración
100
75
50
25
0
00:00:00
01:00:00
02:00:00
03:00:00
04:00:00
05:00:00
06:00:00
07:00:00
08:00:00
09:00:00
10:00:00
11:00:00
12:00:00
13:00:00
14:00:00
15:00:00
16:00:00
17:00:00
18:00:00
19:00:00
20:00:00
21:00:00
22:00:00
23:00:00
Hora
Código
C6H6
C7H8
C8H10
CO
NO
NO2
NOx
O3
SO2




```
require("dplyr")

Viveros$gases_total <- rowSums(Viveros[, c('NO', 'NO2', 'NOx', 'O3', 'SO2', 'CO', 'C6H6', 'C7H8', 'C8H1
```

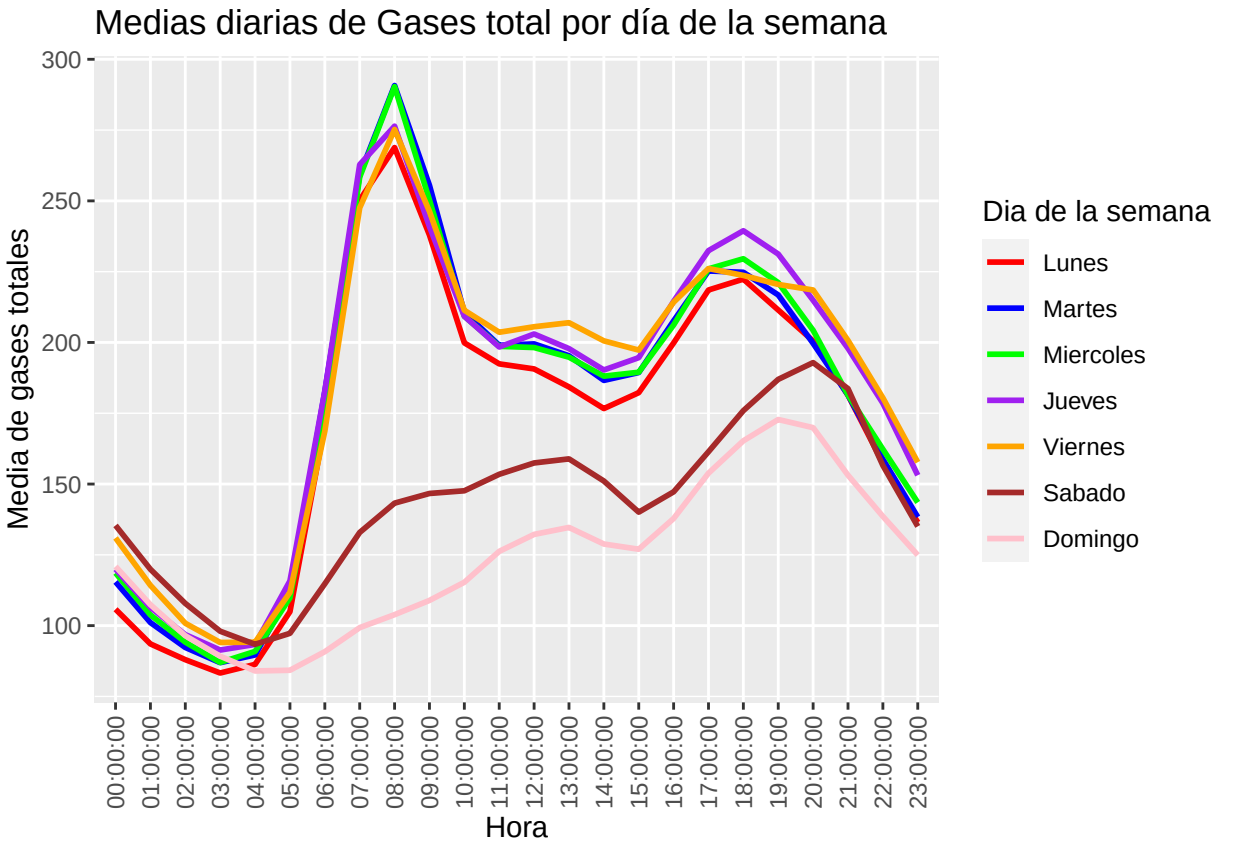
```
media_diaria <- Viveros %>%
  group_by(`Dia de la semana`, Hora) %>%
  summarise(media_gases_total = mean(gases_total, na.rm = TRUE))
```

```
## `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
  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" = "
  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

Análisis de outliers

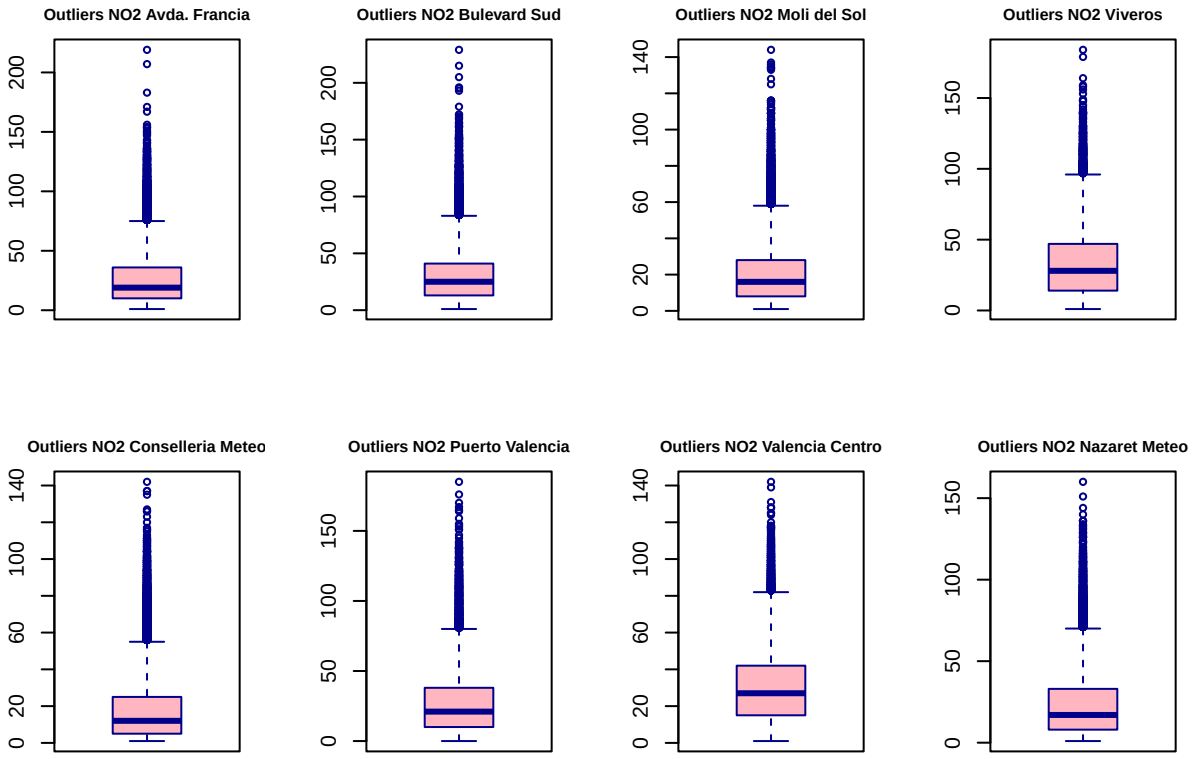
A continuación, exploraremos la calidad del aire mediante boxplots que representan las concentraciones de dos sustancias fundamentales: dióxido de nitrógeno (NO2) y ozono (O3). Cada gráfico proporciona una visión detallada de la distribución de estas sustancias en las distintas estaciones de la red de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

```r
  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.

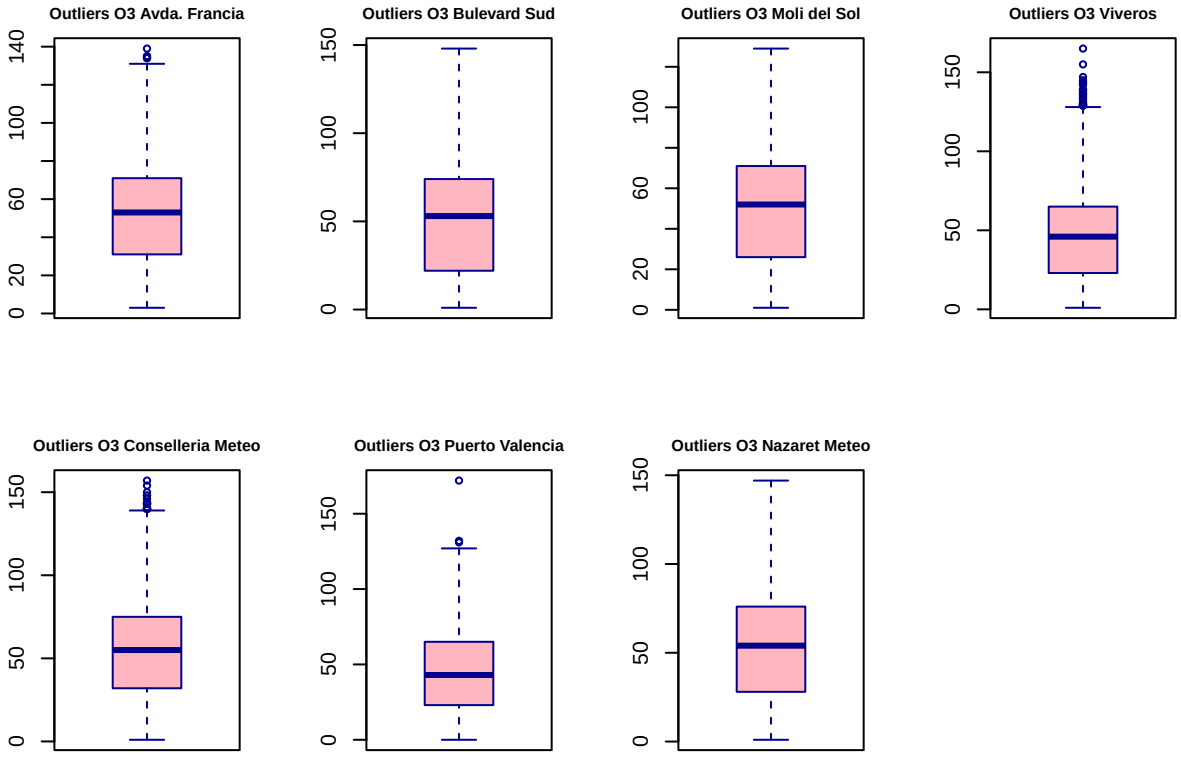
```r
par(mfrow = c(2, 4), mar = c(4, 4, 2, 1),cex.main = 0.8)

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 Conselleria 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)
}
```

```r
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)
```

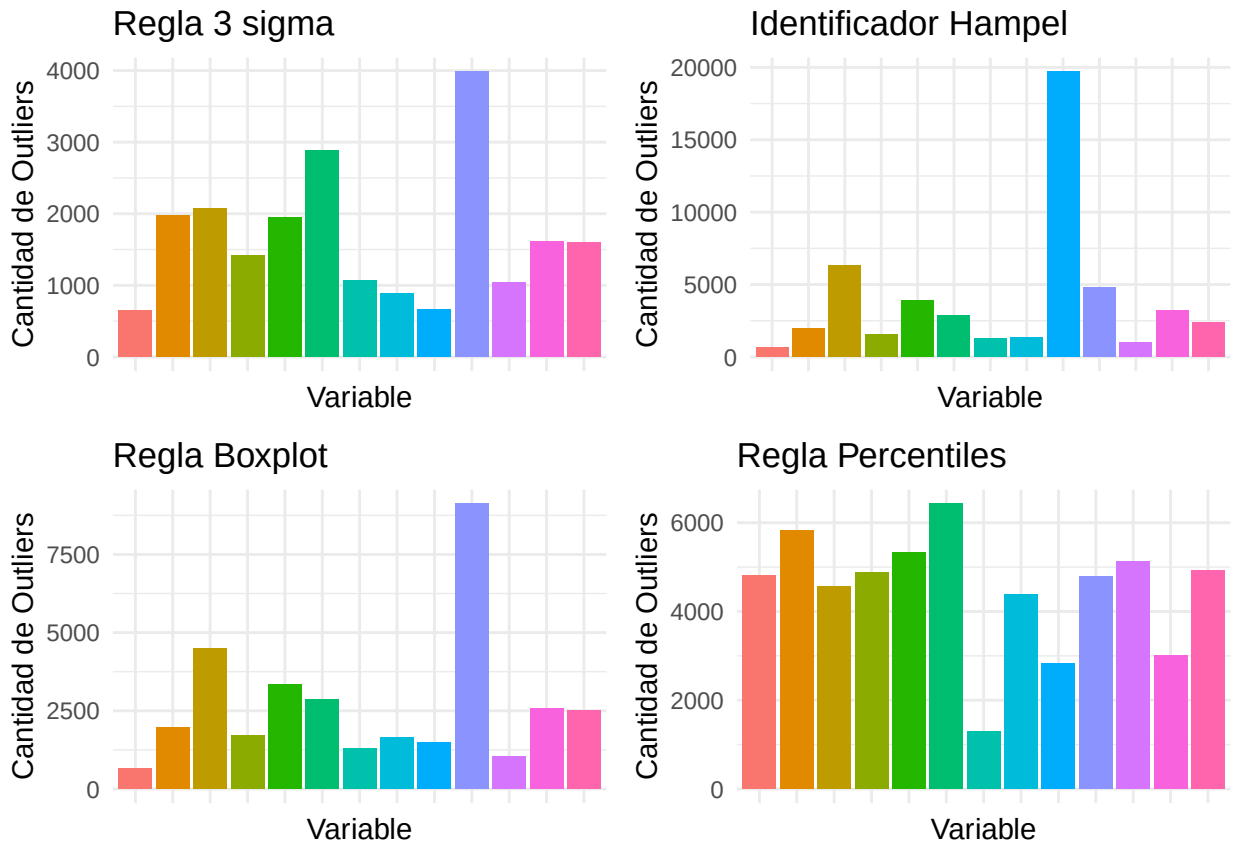


Figure 2: Gráficos de barras para detectar outliers.

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