

Untitled

2025-05-21

```
library(dplyr)

##
## Attaching package: 'dplyr'

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

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

library(tidyr)
library(ggplot2)
library(naniar)
library(lubridate)

##
## Attaching package: 'lubridate'

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

library(zoo)

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##     as.Date, as.Date.numeric

library(stringr)
library(emmeans)

## Welcome to emmeans.
## Caution: You lose important information if you filter this package's results.
## See '? untidy'
```

```

library(forecast)

## Registered S3 method overwritten by 'quantmod':
##   method           from
##   as.zoo.data.frame zoo

library(car)

## Loading required package: carData

##
## Attaching package: 'car'

## The following object is masked from 'package:dplyr':
## 
##     recode

library(lmtest)
library(emmeans)
library(mice)

##
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
## 
##     filter

## The following objects are masked from 'package:base':
## 
##     cbind, rbind

library(FactoMineR)
library(factoextra)

## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa

library(pcaMethods)

## Loading required package: Biobase

## Loading required package: BiocGenerics

## Loading required package: generics

##
## Attaching package: 'generics'

```

```

## The following object is masked from 'package:lubridate':
##
##     as.difftime

## The following object is masked from 'package:dplyr':
##
##     explain

## The following objects are masked from 'package:base':
##
##     as.difftime, as.factor, as.ordered, intersect, is.element, setdiff,
##     setequal, union

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:mice':
##
##     cbind, rbind

## The following object is masked from 'package:dplyr':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, saveRDS, table, tapply, unique,
##     unsplit, which.max, which.min

## Welcome to Bioconductor
##
## Vignettes contain introductory material; view with
##   'browseVignettes()'. To cite Bioconductor, see
##   'citation("Biobase")', and for packages 'citation("pkgname")'.

##
## Attaching package: 'pcaMethods'

## The following object is masked from 'package:stats':
##
##     loadings

```

```
library(tibble)
library(cluster)
library(pheatmap)
```

```
df <- read.csv("~/Documents/Drive/UPV/Segundo/2ºCuatri/Proyecto/base_para_obj1.csv", stringsAsFactors = TRUE)
df <- df %>%
  mutate(FechaHora = ymd_hms(paste(Fecha, Hora, "00", "00", sep = ":"), tz = "UTC"))
df$Origen <- as.factor(df$Origen)
```

Selección de variables

Para este análisis vamos a utilizar las variables de interés que son las siguientes:

- FechaHora
- Origen
- NO
- NO_2
- NO_x
- O_3
- PM_{10}
- $PM_{2.5}$
- SO_2

```
df <- df %>%
  select(FechaHora, Origen, NO, NO2, NOx, O3, PM10, PM2.5, SO2)
```

Selección de estaciones

Para seleccionar las estaciones vamos a ver aquellas que tienen todos los contaminantes en común.

Antes de ello vamos a ver el número de faltantes por contaminante y por estación.

```
missing_pct <- df %>%
  pivot_longer(
    cols      = c(NO2, NO, NOx, PM10, `PM2.5`, SO2, O3),
    names_to  = "Contaminante",
    values_to = "Valor"
  ) %>%
  group_by(Origen, Contaminante) %>%
  summarise(
    Porcentaje_Faltantes = sum(is.na(Valor)) / n() * 100,
    .groups = "drop"
  ) %>%
  arrange(Porcentaje_Faltantes)
```

```
print(missing_pct)
```

```
## # A tibble: 140 x 3
```

```

##    Origen          Contaminante Porcentaje_Faltantes
##    <fct>           <chr>                 <dbl>
## 1 Benetusser UM      PM10                  0
## 2 Benetusser UM      PM2.5                 0
## 3 Massanassa_UM     PM10                  0
## 4 Massanassa_UM     PM2.5                 0
## 5 València Olivereta NO                   0
## 6 València Olivereta NO2                  0
## 7 València Olivereta NOx                 0
## 8 València Port llit antic Túria PM10   0.0335
## 9 València Port llit antic Túria PM2.5  0.0335
## 10 València Olivereta PM10                0.0614
## # i 130 more rows

```

```
miss_var_summary(df)
```

```

## # A tibble: 9 x 3
##   variable n_miss pct_miss
##   <chr>     <int>   <num>
## 1 PM10      266941   39.9
## 2 PM2.5     266645   39.9
## 3 S02       186240   27.8
## 4 O3        184673   27.6
## 5 N02       110311   16.5
## 6 NO         110310   16.5
## 7 NOx       110310   16.5
## 8 FechaHora 0         0
## 9 Origen    0         0

```

Vamos a eliminar aquellas estaciones que tengan un 100% de datos faltantes en algún contaminante.

```

df <- df %>%
  group_by(Origen) %>%
  filter(if_all(
    c(NO2, NO, NOx, PM10, `PM2.5`, S02, O3),
    ~ mean(is.na(.)) < 1
  )) %>%
  ungroup()

```

```
miss_var_summary(df)
```

```

## # A tibble: 9 x 3
##   variable n_miss pct_miss
##   <chr>     <int>   <num>
## 1 PM10      18046   5.28
## 2 PM2.5     17743   5.19
## 3 NO         13751   4.02
## 4 N02       13751   4.02
## 5 NOx       13750   4.02
## 6 S02       12755   3.73
## 7 O3        7210    2.11
## 8 FechaHora 0        0
## 9 Origen    0        0

```

Al eliminar esas estaciones la cantidad de faltantes disminuye considerablemente ya que las estaciones restantes miden todas las variables.

Ahora vamos a comprobar que no exista algún año con un 100% de faltantes para alguno de los contaminantes.

```

missing_pct_year <- df %>%
  mutate(year = year(FechaHora)) %>%
  pivot_longer(
    cols      = c(NO2, NO, NOx, PM10, `PM2.5`, SO2, O3),
    names_to  = "Contaminante",
    values_to = "Valor"
  ) %>%
  group_by(year, Contaminante) %>%
  summarise(
    PctMissing = mean(is.na(Valor)) * 100,
    .groups     = "drop"
  )

print(missing_pct_year)

## # A tibble: 42 x 3
##       year Contaminante PctMissing
##   <dbl> <chr>           <dbl>
## 1 2019  NO             1.52
## 2 2019  NO2            1.52
## 3 2019  NOx            1.52
## 4 2019  O3             0.716
## 5 2019  PM10           1.53
## 6 2019  PM2.5          1.53
## 7 2019  SO2            1.42
## 8 2020  NO             1.11
## 9 2020  NO2            1.11
## 10 2020 NOx            1.11
## # i 32 more rows

```

Hay muy poco porcentaje de faltantes en los contaminantes, por lo que no es necesario eliminar ning n a o.

Imputación de los datos

Vamos a crear una función para imputar los datos.

La función imputará los valores faltantes de las variables por la media de los valores de esa variable de la misma estación en los instantes más cercanos.

```
imputar_numerico_por_fecha <- function(df, columnna) {  
  df <- df %>%  
    arrange(Origen, FechaHora) %>%  
    group_by(Origen) %>%  
    mutate(  
      !!sym(columnna) := ifelse(  
        is.na (!!sym(columnna)),  
        rowMeans(cbind(  
          zoo::na.locf (!!sym(columnna)), na.rm = FALSE, fromLast = FALSE),
```

```

        zoo::na.locf(!!sym(columna), na.rm = FALSE, fromLast = TRUE)
    ), na.rm = TRUE),
    !!sym(columna)
)
) %>%
ungroup()
return(df)
}

datos_limpios <- df %>%
imputar_numerico_por_fecha("PM10") %>%
imputar_numerico_por_fecha("PM2.5") %>%
imputar_numerico_por_fecha("NO") %>%
imputar_numerico_por_fecha("NO2") %>%
imputar_numerico_por_fecha("NOx") %>%
imputar_numerico_por_fecha("SO2") %>%
imputar_numerico_por_fecha("O3")

```

Una vez imputados, comprobamos que no hay faltantes y que la imputación ha funcionado correctamente.

```
miss_var_summary(datos_limpios)
```

```

## # A tibble: 9 x 3
##   variable n_miss pct_miss
##   <chr>     <int>    <num>
## 1 FechaHora     0      0
## 2 Origen        0      0
## 3 NO            0      0
## 4 NO2           0      0
## 5 NOx          0      0
## 6 O3            0      0
## 7 PM10          0      0
## 8 PM2.5         0      0
## 9 SO2           0      0

```

Se han imputado correctamente. Guardamos la base de datos.

```
#write.csv(datos_limpios, "Calidad_Aire_limpio.csv", row.names = FALSE)
```

Ahora vamos a comprobar si las distribuciones se ven afectadas al hacer la imputación de los datos.

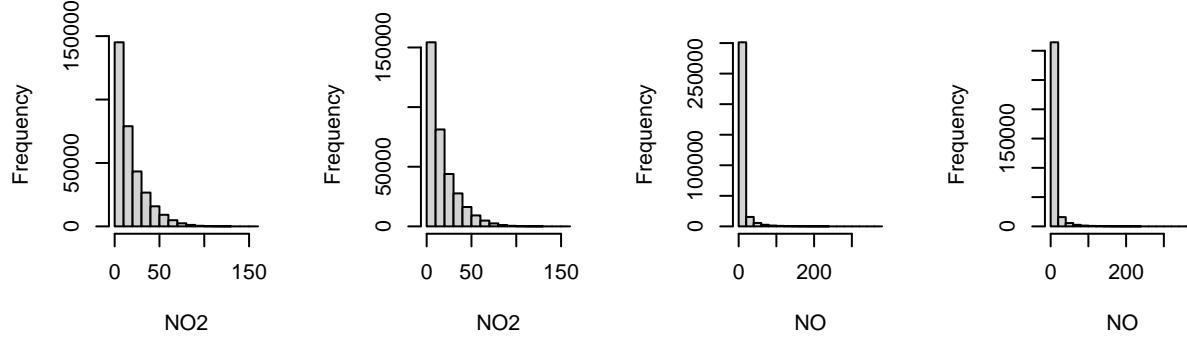
```

contaminantes <- c("NO2", "NO", "NOx", "PM10", "PM2.5", "SO2", "O3")

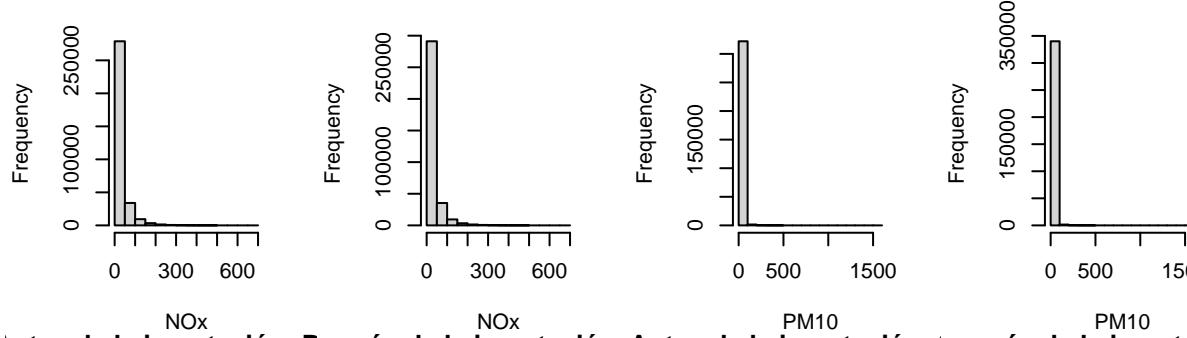
# Comprobar la distribución de los contaminantes antes y después de la imputación
par(mfrow = c(2, 4))
for (contaminante in contaminantes) {
  # Histograma antes de la imputación
  hist(df[[contaminante]], main = paste("Antes de la imputación:", contaminante), xlab = contaminante)

  # Histograma después de la imputación
  hist(datos_limpios[[contaminante]], main = paste("Después de la imputación:", contaminante), xlab = c
}
```

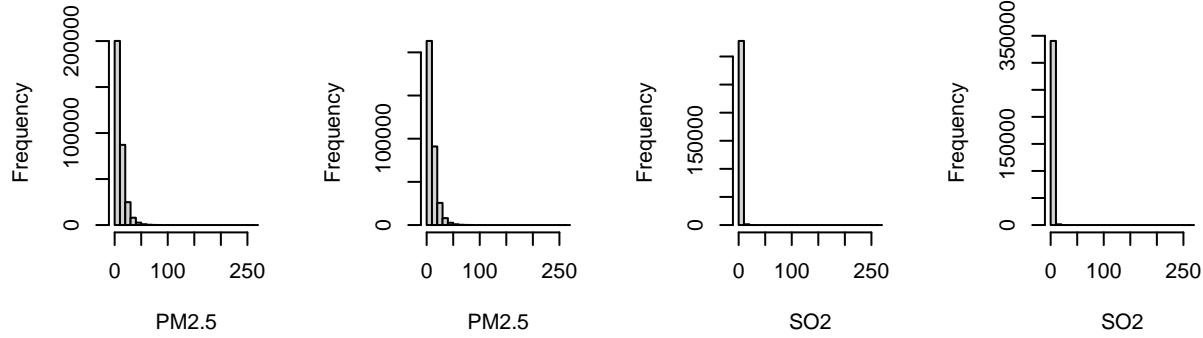
Antes de la imputación: después de la imputación



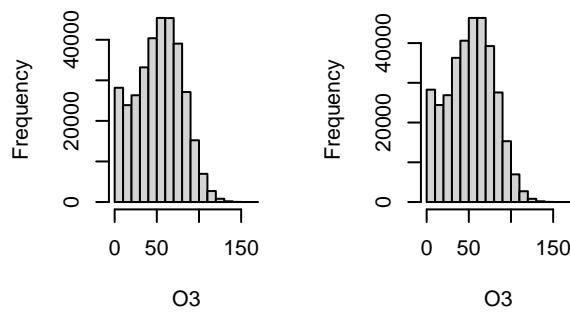
Antes de la imputación: después de la imputación



Antes de la imputación: después de la imputación Antes de la imputación: después de la imputación



Antes de la imputación: después de la imputación



```
# Quitamos la dana para que se vea mejor
```

```
df_sin_DANA <- subset(df, FechaHora < as.POSIXct("2024-11-17 18:00:00"))
```

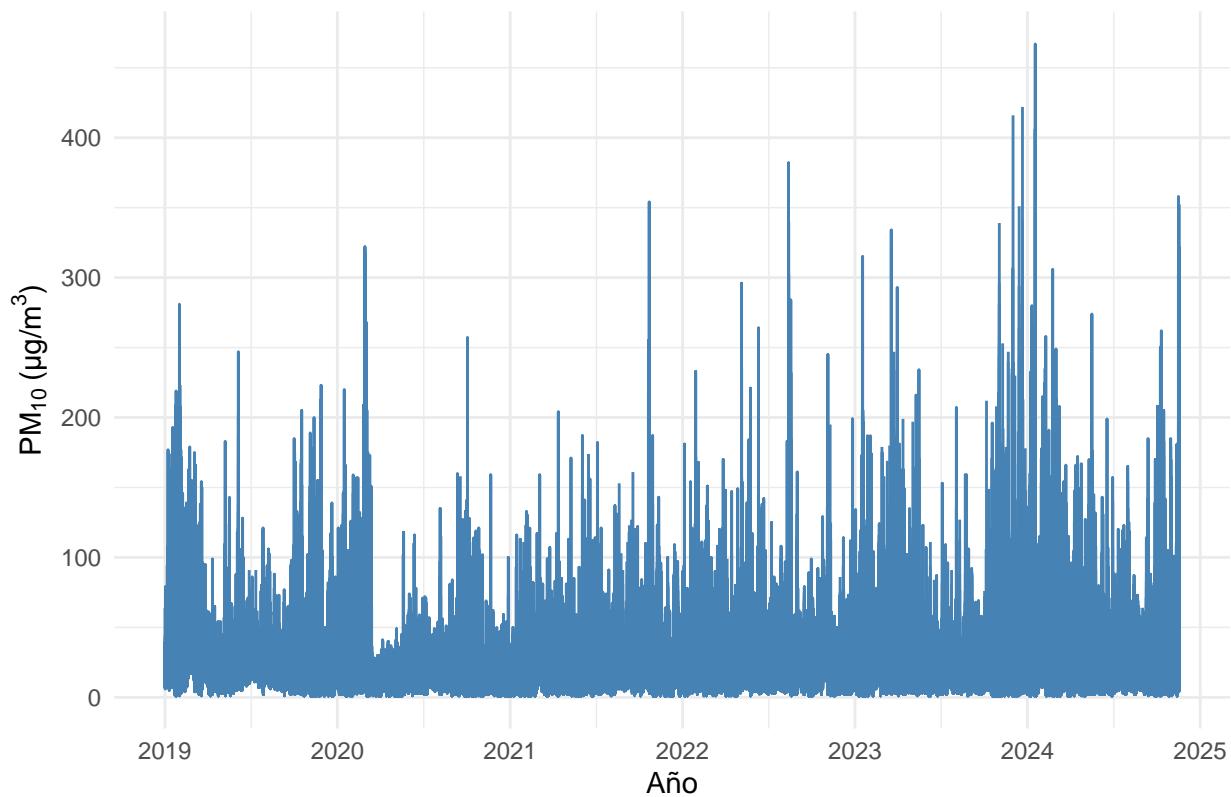
```
datos_limpios_sin_DANA <- subset(datos_limpios, FechaHora < as.POSIXct("2024-11-17 18:00:00"))
```

```

ggplot(df_sin_DANA, aes(x = FechaHora, y = PM10, group = 1)) +
  geom_line(color = "steelblue") +
  scale_x_datetime(
    date_breaks = "1 year",
    date_labels = "%Y"
  ) +
  labs(
    title = "Evolución temporal de PM10",
    x      = "Año",
    y      = expression(PM[10]^"(μg/m^3*)")
  ) +
  theme_minimal()

```

Evolución temporal de PM10

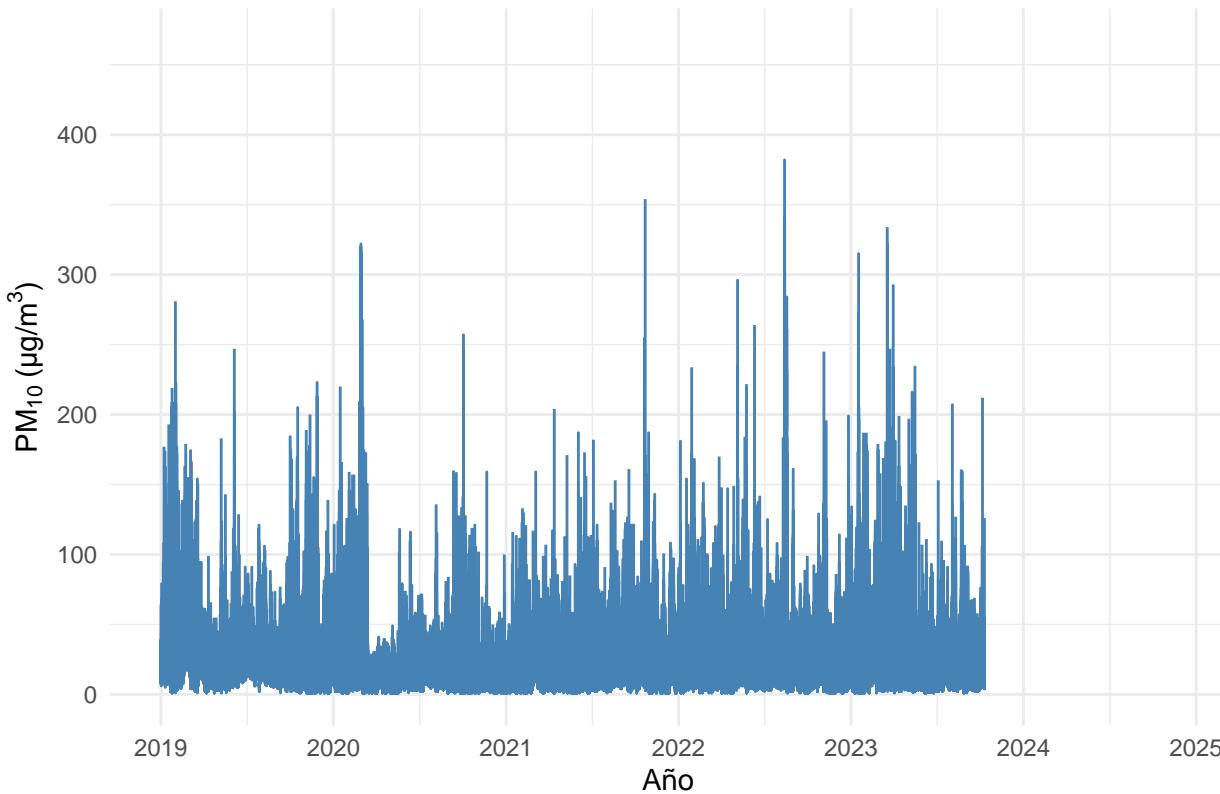


```

# imputados
ggplot(datos_limpios_sin_DANA, aes(x = FechaHora, y = PM10, group = 1)) +
  geom_line(color = "steelblue") +
  scale_x_datetime(
    date_breaks = "1 year",
    date_labels = "%Y"
  ) +
  labs(
    title = "Evolución temporal de PM10 con imputados",
    x      = "Año",
    y      = expression(PM[10]^"(μg/m^3*)")
  ) +
  theme_minimal()

```

Evolución temporal de PM10 con imputados



Observamos que no hay casi cambio entre ellos, los cambios son irrelevantes. Vamos a verlo un poco más en detalle, por ejemplo viendo como afecta a la media anual.

```
df_sin_DANA <- df_sin_DANA %>%
  mutate(
    hora = hour(FechaHora))

datos_limpios_sin_DANA <- datos_limpios_sin_DANA %>%
  mutate(
    hora = hour(FechaHora))

df_hourly <- df_sin_DANA %>%
  group_by(hora) %>%
  summarise(media_PM10 = mean(PM10, na.rm = TRUE)) %>%
  mutate(Hora = as.numeric(as.character(hora))) %>%
  arrange(Hora)

ggplot(df_hourly, aes(x = hora, y = media_PM10)) +
  geom_area(fill = "#FF6E6E", alpha = 0.4) +
  geom_line(color = "#B22222", size = 1.2) +
  geom_point(color = "#B22222", size = 3) +
  scale_x_continuous(breaks = 0:23, limits = c(0, 23)) +
  labs(
    title      = "Hourly evolution of the average PM10 concentration",
    subtitle   = "Average values from all stations in Valencia",
    x         = "Hour of the day",
```

```

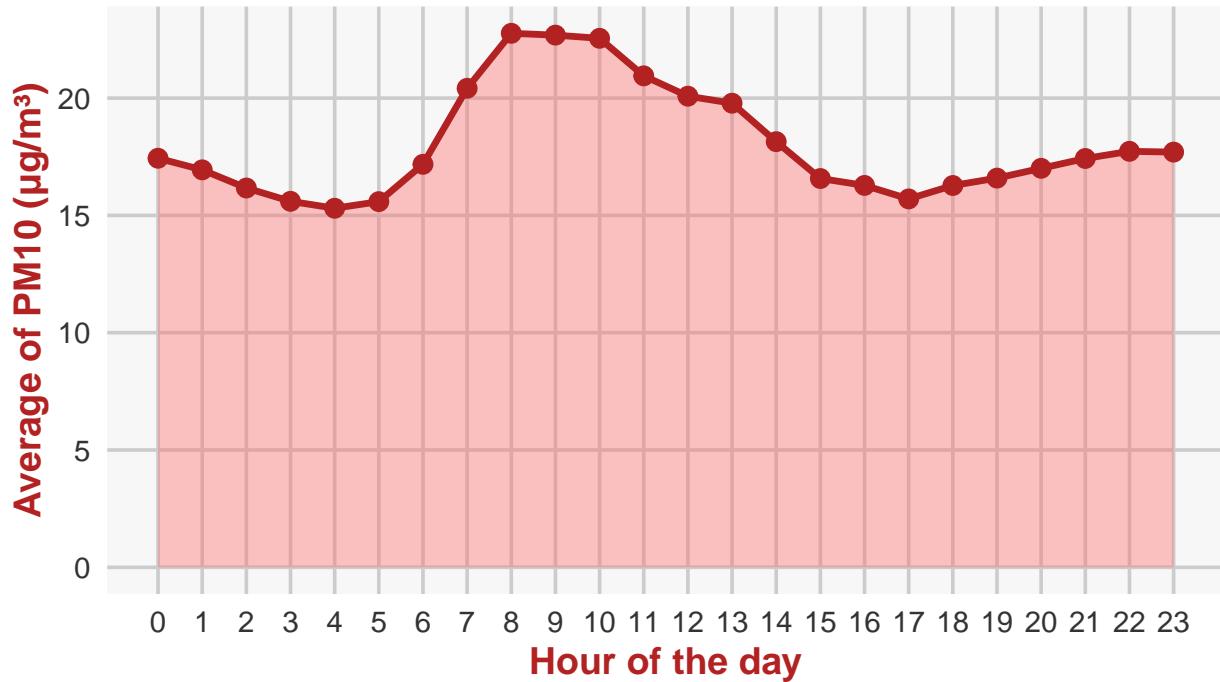
y      = "Average of PM10 ( $\mu\text{g}/\text{m}^3$ )",
caption = "Source: your air pollution data"
) +
theme_minimal(base_size = 14) +
theme(
  plot.background    = element_rect(fill = "white", color = NA),
  panel.background  = element_rect(fill = "#f7f7f7", color = NA),
  panel.grid.major   = element_line(color = "gray80"),
  panel.grid.minor   = element_blank(),
  axis.title         = element_text(face = "bold", color = "#B22222"),
  axis.text          = element_text(color = "gray20"),
  plot.title         = element_text(face = "bold", size = 16, hjust = 0.5),
  plot.subtitle      = element_text(color = "gray40", hjust = 0.5),
  plot.caption        = element_text(size = 8, color = "gray60")
)

```

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.

Hourly evolution of the average PM10 concentration

Average values from all stations in Valencia



Source: your air pollution data

```

#IMPUTADOS
df_hourly_limpio <- datos_limpios_sin_DANA %>%
  group_by(hora) %>%
  summarise(media_PM10 = mean(PM10, na.rm = TRUE)) %>%

```

```

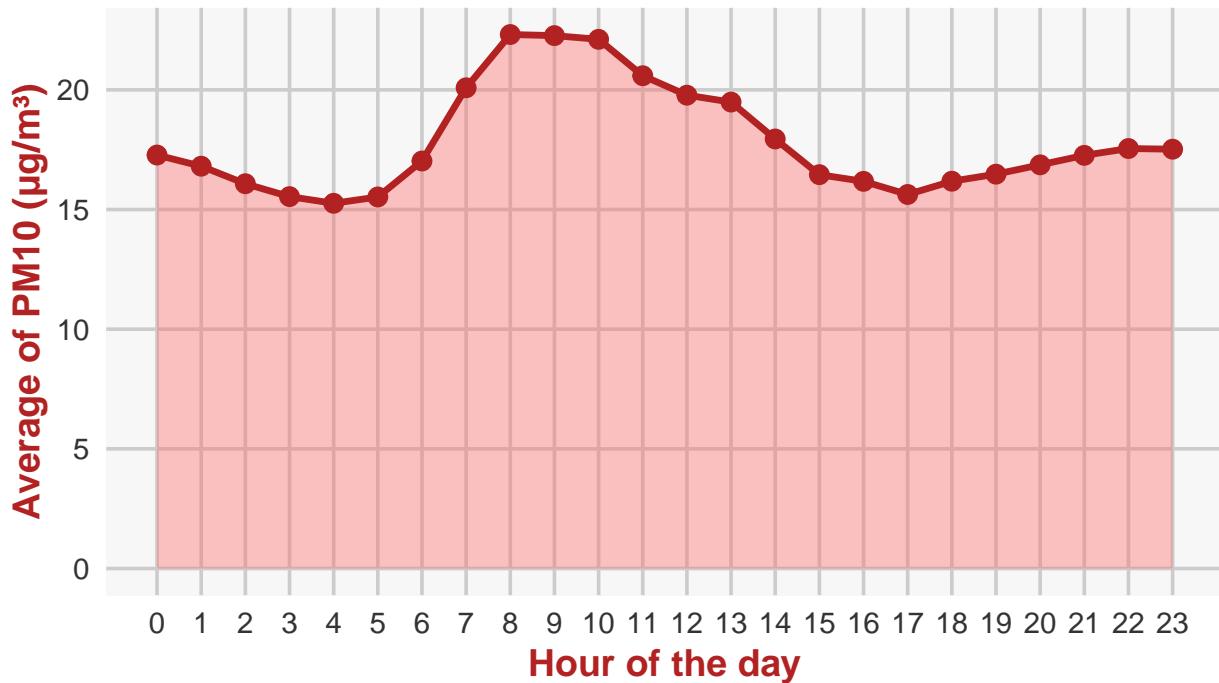
mutate(Hora = as.numeric(as.character(hora))) %>%
arrange(Hora)

ggplot(df_hourly_limpio, aes(x = hora, y = media_PM10)) +
  geom_area(fill = "#FF6E6E", alpha = 0.4) +
  geom_line(color = "#B22222", size = 1.2) +
  geom_point(color = "#B22222", size = 3) +
  scale_x_continuous(breaks = 0:23, limits = c(0, 23)) +
  labs(
    title      = "Hourly evolution of the average PM10 concentration IMPUTED",
    subtitle   = "Average values from all stations in Valencia",
    x          = "Hour of the day",
    y          = "Average of PM10 (µg/m³)",
    caption    = "Source: your air pollution data"
  ) +
  theme_minimal(base_size = 14) +
  theme(
    plot.background = element_rect(fill = "white", color = NA),
    panel.background = element_rect(fill = "#f7f7f7", color = NA),
    panel.grid.major = element_line(color = "gray80"),
    panel.grid.minor = element_blank(),
    axis.title       = element_text(face = "bold", color = "#B22222"),
    axis.text         = element_text(color = "gray20"),
    plot.title        = element_text(face = "bold", size = 16, hjust = 0.5),
    plot.subtitle     = element_text(color = "gray40", hjust = 0.5),
    plot.caption      = element_text(size = 8, color = "gray60")
  )
)

```

Hourly evolution of the average PM10 concentration IMPUTI

Average values from all stations in Valencia



Source: your air pollution data

Observamos una muy ligera disminución, pero muy insignificante. Podemos concluir que la imputación ha sido correcta.

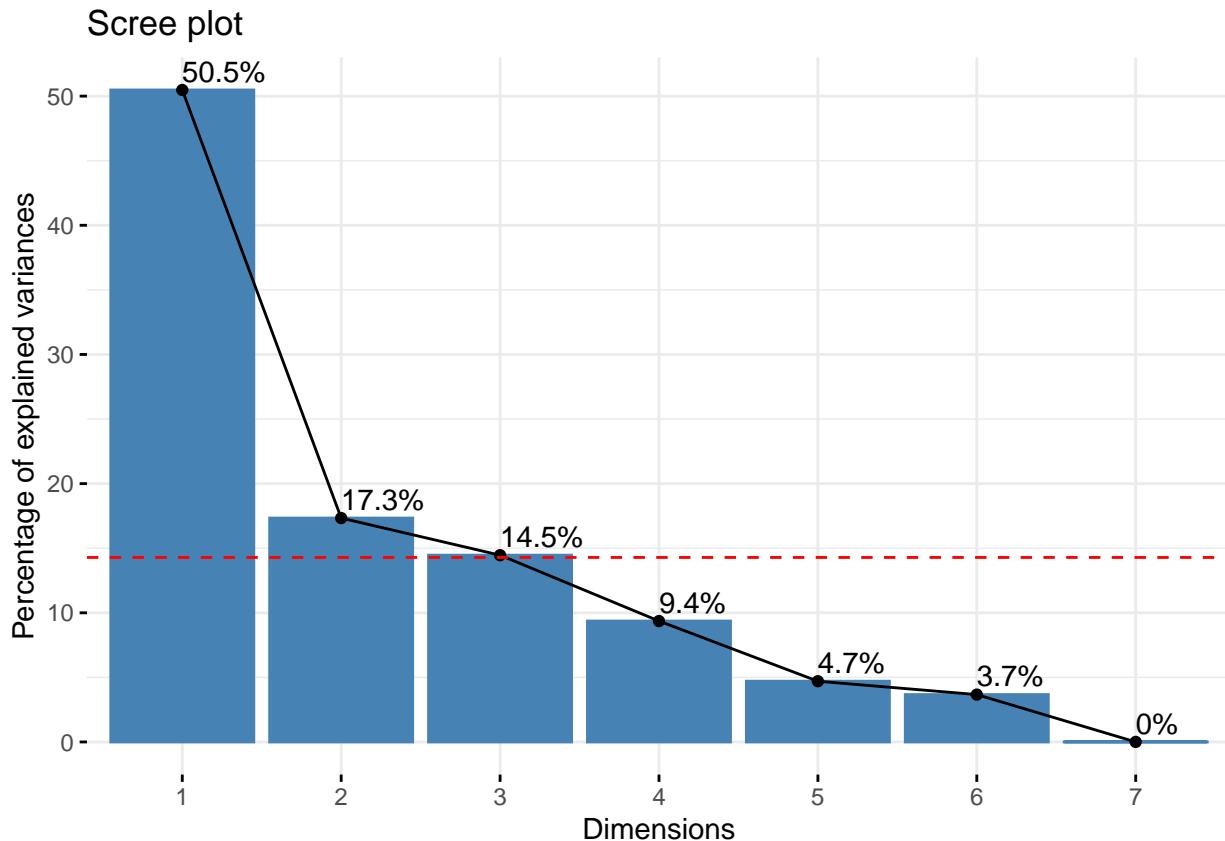
Comportamiento entre contaminantes

Para ver patrones entre contaminantes vamos a realizar, en primer lugar, un PCA. Dado que hemos imputado los faltantes, podemos usar cualquier modelo del PCA.

```
set.seed(100)
df_pca_20k <- datos_limpios %>%
  filter(year(FechaHora) == 2023) %>%
  sample_n(20000) %>%
  select(Origen, NO, NO2, NOx, O3, PM10, `PM2.5`, SO2)

res.pca <- PCA(df_pca_20k,
                 scale.unit = TRUE,
                 ncp = 10,
                 graph = FALSE,
                 quali.sup = 'Origen')

# Visualizar varianza explicada
fviz_eig(res.pca, addlabels = TRUE) +
  geom_hline(yintercept = 100 / res.pca$call$ncp, linetype = 2, color = "red")
```

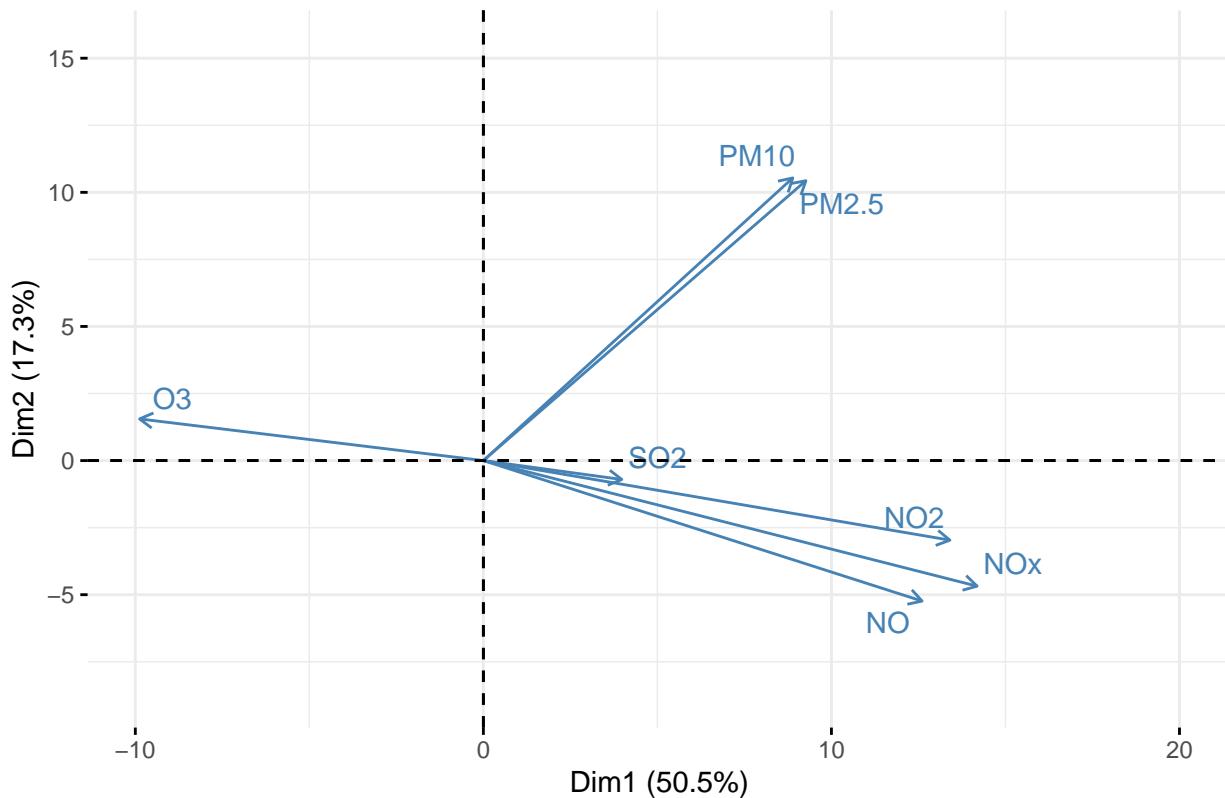


```
eig.val <- get_eigenvalue(res.pca)
```

Observamos que la CP1 explica un 50.5% de la variabilidad, la CP2 explica un 17.7% y la CP3 un 14.3%. Dado este gráfico podemos tomar 2 o 3 componentes principales. Primero visualizaremos la CP1 y CP2

```
set.seed(100)
fviz_pca_biplot(
  res.pca,
  axes      = c(1, 2),
  habillage = "Origen",
  addEllipses = FALSE,
  repel     = TRUE,
  label     = "var",
  invisible = "ind",
  pointsize = 1.5
) +
  labs(
    title = "PCA Biplot: Dim1 vs Dim2",
  )
```

PCA Biplot: Dim1 vs Dim2



Como esperábamos, las variables *PM10* y *PM2.5* están fuertemente correlacionadas tanto en la componente 1 como en la 2. Estas dos variables también están relacionadas con *NO*, *NO₂*, y *NO_x* en la dimensión 1, y con *SO₂* también pero mucho menos significante.

Explicación para dimensión 1:

- Las variables *PM10* y *PM2.5* están fuertemente correlacionadas entre sí.
- Las *NO*, *NO₂*, y *NO_x* también están fuertemente correlacionadas entre sí.
- La variable *SO₂* no parece verse muy afectada por la componente principal 1.
- Las variables *PM10*, *PM2.5*, *NO*, *NO₂*, *NO_x* y *SO₂* están correlacionadas pero *SO₂* menos significativamente.
- La variable *O₃* está negativamente correlacionada con el resto de variables.

Estas relaciones están justificadas, ya que tienen fuentes generadoras en común, por ejemplo la combustión en los automóviles. La relación con tráfico, la veremos más adelante.

La relación de *O₃* está justificada ya que reacciona con *NO* en procesos fotoquímicos para generar *NO₂*, por lo que cuanto más *NO* y *NO_x* menos *O₃*.

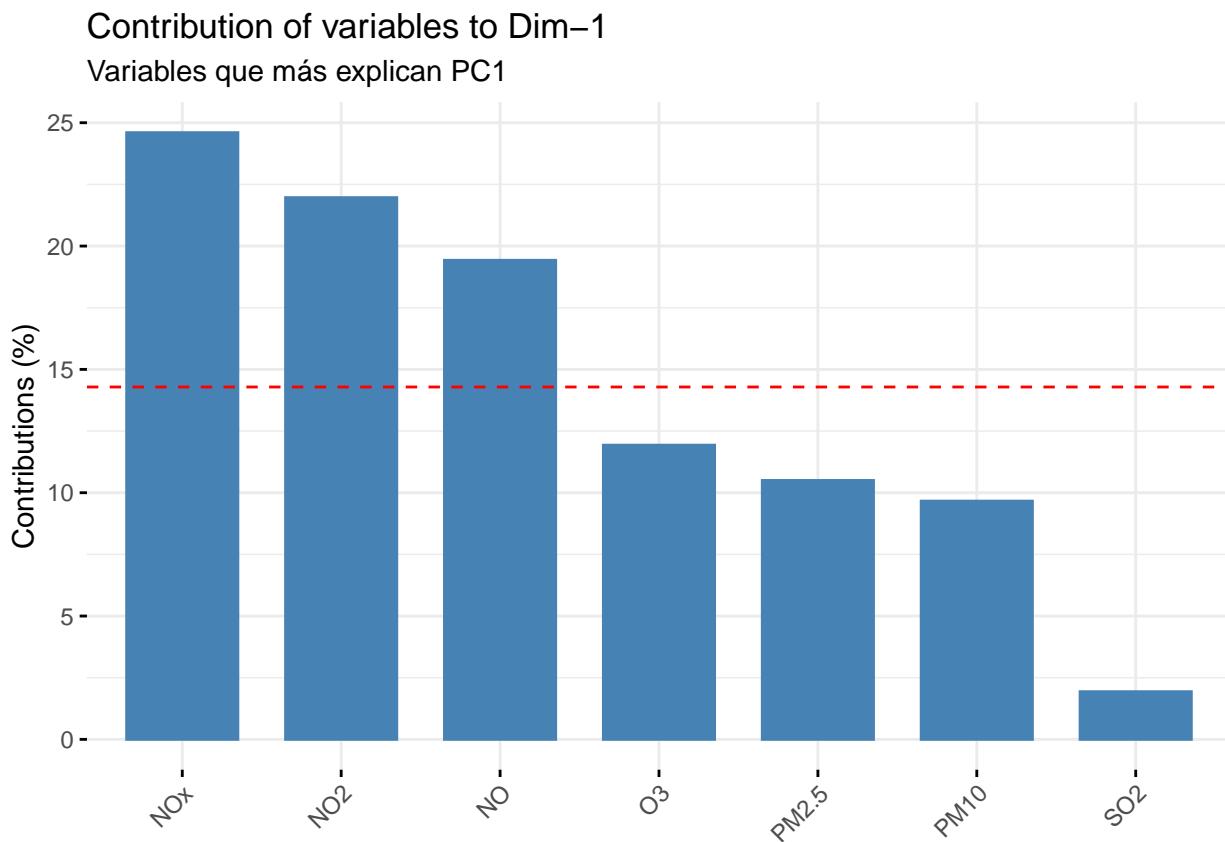
Explicación dimensión 2:

- Los contaminantes *O₃* y *SO₂* no se ven casi afectados por esta componente, pero tienen una relación positiva.
- Las *NO*, *NO₂*, y *NO_x* también están fuertemente correlacionadas entre sí.
- Las variables *PM10* y *PM2.5* están fuertemente correlacionadas entre sí.
- Los contaminantes *PM10* y *PM2.5* están negativamente correlacionados con *NO*, *NO₂*, y *NO_x*.

En este caso sucede al revés que en la primera componente, lo cual podría deberse a muchos factores como por ejemplo, la meteorología ya que NO , NO_2 y NOx parece ser que se ve afectada por la temperatura segun hemos visto, pero se confirmara más adelante en el estudio con meteorología.

Además se observa en las siguientes gráficas.

```
fviz_contrib(
  res.pca,
  choice = "var",
  axes   = 1,
  top    = 7
) +
  labs(subtitle = "Variables que más explican PC1")
```

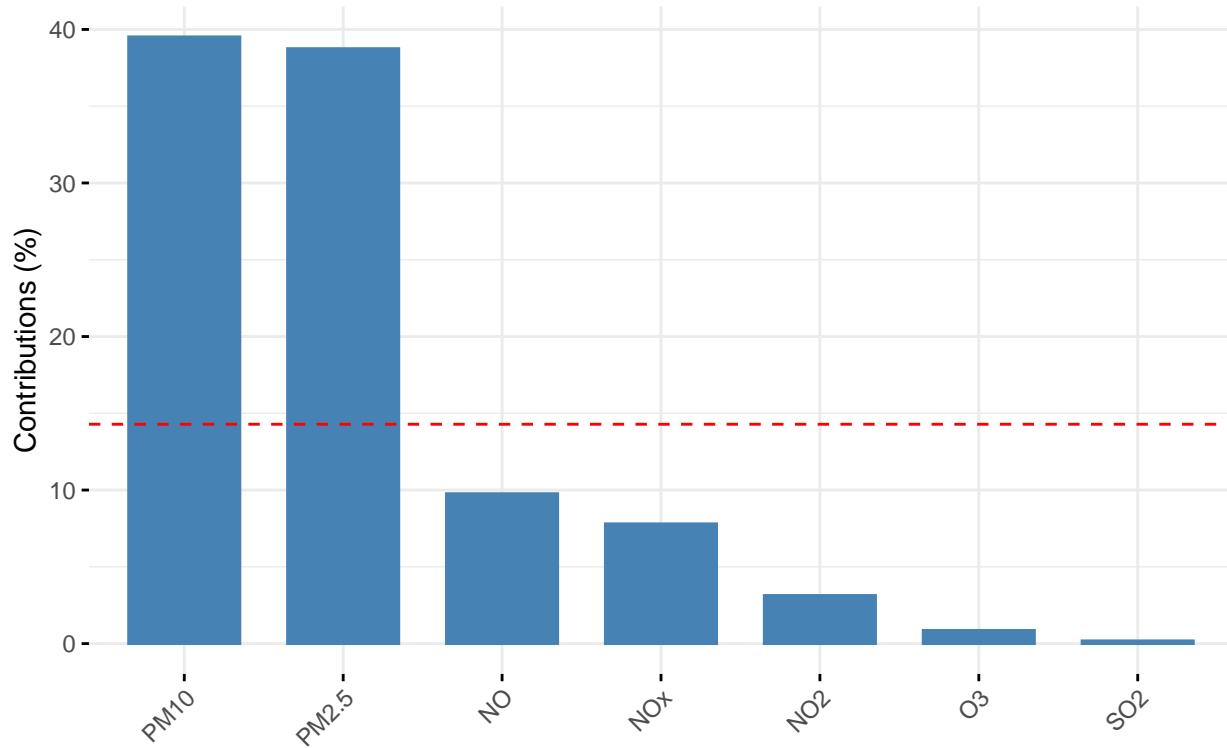


Las variables NO , NO_2 , y NOx son las que más contribuyen a la primera componente. Los contaminantes $PM10$ y $PM2.5$ tambien contribuyen pero menos.

```
fviz_contrib(
  res.pca,
  choice = "var",
  axes   = 2,
  top    = 7
) +
  labs(subtitle = "Variables que más explican PC2")
```

Contribution of variables to Dim–2

Variables que más explican PC2

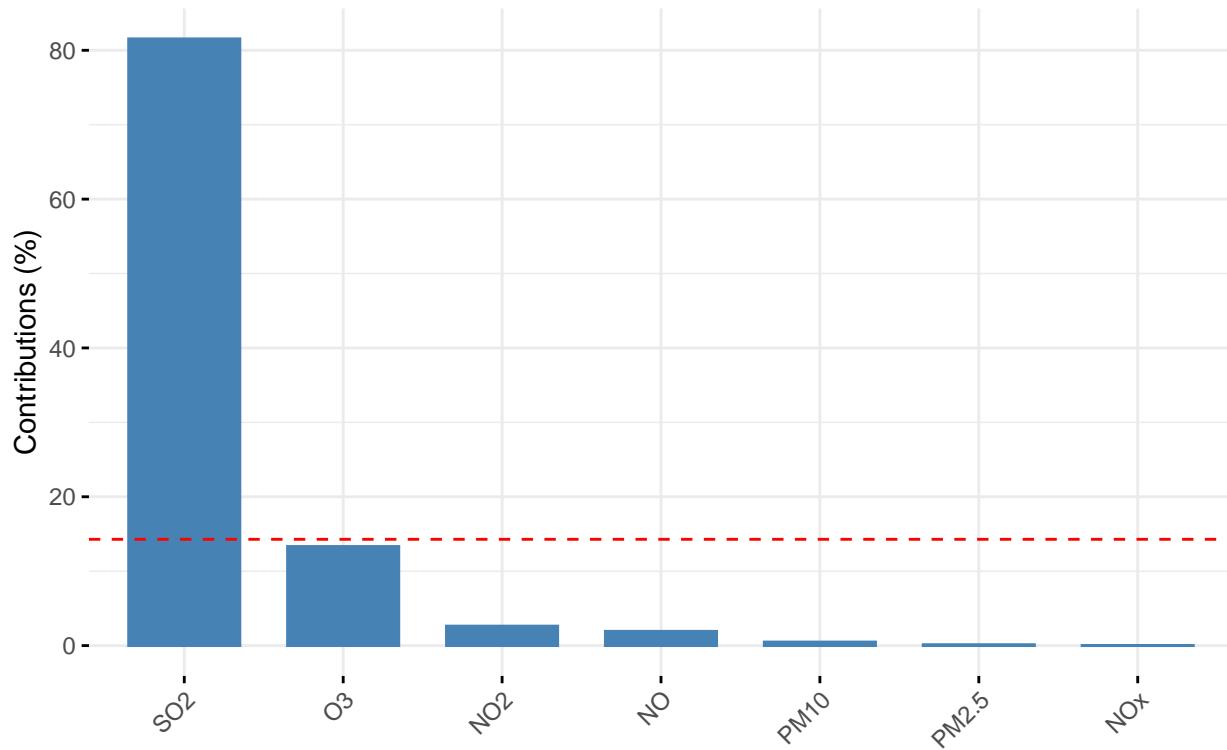


En la segunda componente las variables *PM10* y *OM2.5* son las que más contribuyen con diferencia.

```
fviz_contrib(  
  res.pca,  
  choice = "var",  
  axes    = 3,  
  top     = 7  
) +  
  labs(subtitle = "Variables que más explican PC3")
```

Contribution of variables to Dim-3

Variables que más explican PC3

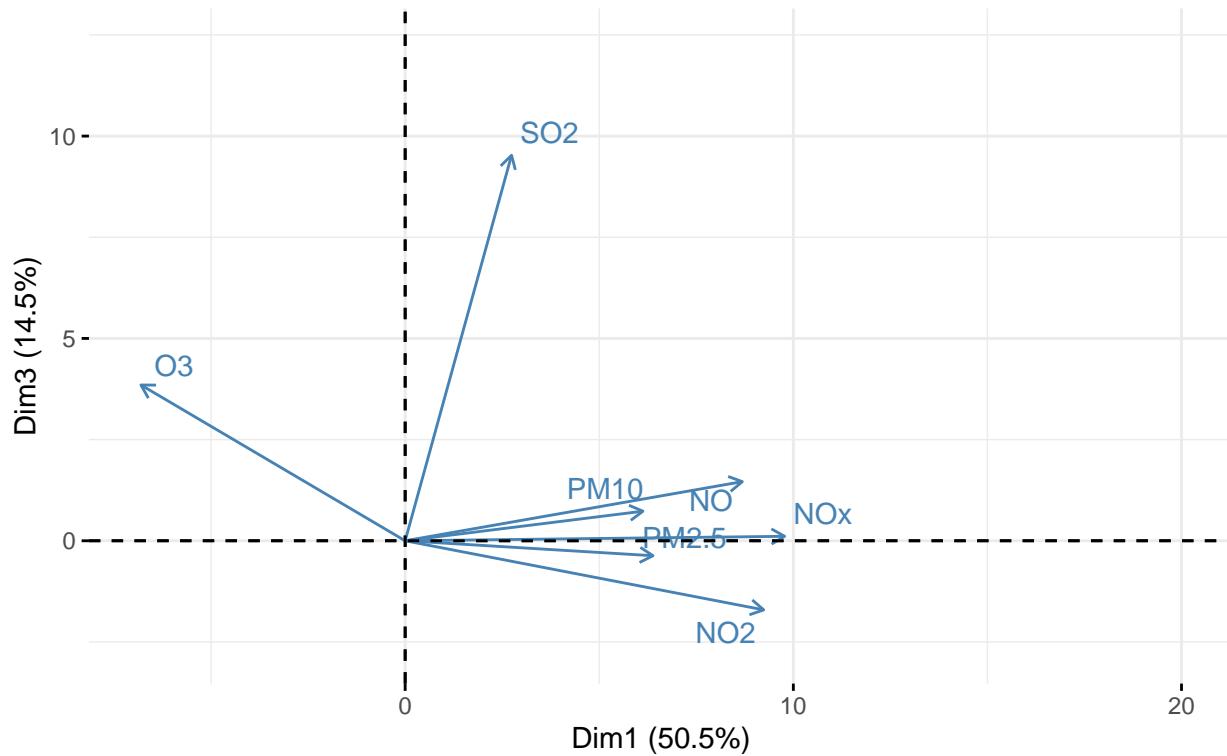


En la tercera componente la variable SO_2 es la que más contribuye con mucha diferencia. Vamos a ver el biplot para CP1 y CP3.

```
set.seed(100)
fviz_pca_biplot(
  res.pca,
  axes      = c(1, 3),
  habillage = "Origen",
  addEllipses = FALSE,
  repel     = TRUE,
  label     = "var",
  invisible = "ind",
  pointsize = 1.5
) +
  labs(
    title = "PCA Biplot: Dim1 vs Dim3",
    subtitle = "Relación de la tercera componente con el principal"
)
```

PCA Biplot: Dim1 vs Dim3

Relación de la tercera componente con el principal

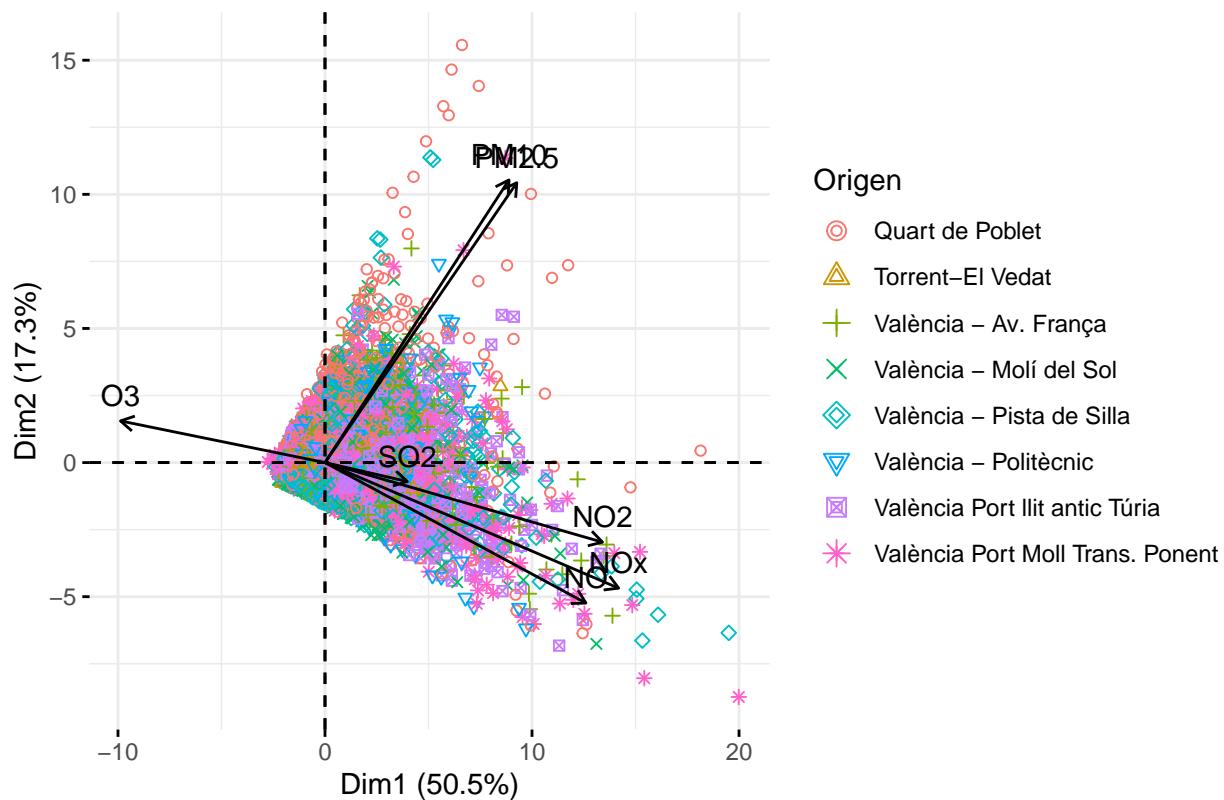


Como habíamos visto, la variable *SO₂* tiene un loading alto en la dimensión 3, por lo que explica bastante la variaildad en esa componente.

Ahora vamos a añadir los puntos de origen.

```
set.seed(100)
fviz_pca_biplot(
  res.pca,
  habillage      = "Origen",
  addEllipses    = FALSE,
  repel          = FALSE,
  col.var        = "black",
  pointsize      = 1.5,
  label = "var"
)
```

PCA – Biplot



Se ve que los puntos están muy concentrados, pero con ligeras diferencias.

Vamos a hacer ahora el PCA agrupando las estaciones por su localización geográfica.

- Puerto: València - Av. França, València Port Ilit antic Túria, València Port Moll Trans. Ponent
- Extrarradio: Quart de Poblet, Torrent-El Vedat
- València - Molí del Sol
- València - Pista de Silla
- València - Politècnic

```
df_pca_grp <- datos_limpios %>%
  filter(year(FechaHora) == 2023) %>%
  sample_n(20000) %>%
  mutate(
    Grupo = case_when(
      Origen %in% c(
        "València – Av. França",
        "València Port Ilit antic Túria",
        "València Port Moll Trans. Ponent"
      ) ~ "Puerto",
      Origen %in% c("Quart de Poblet", "Torrent-El Vedat") ~ "Extrarradio",
      Origen == c("València – Molí del Sol", "València – Pista de Silla", "València – Politècnic") ~ "C"
      TRUE
    )
  )
```

```

) %>%
filter(!is.na(Grupo)) %>%
select(Grupo, NO, NO2, NOx, O3, PM10, `PM2.5`, SO2)

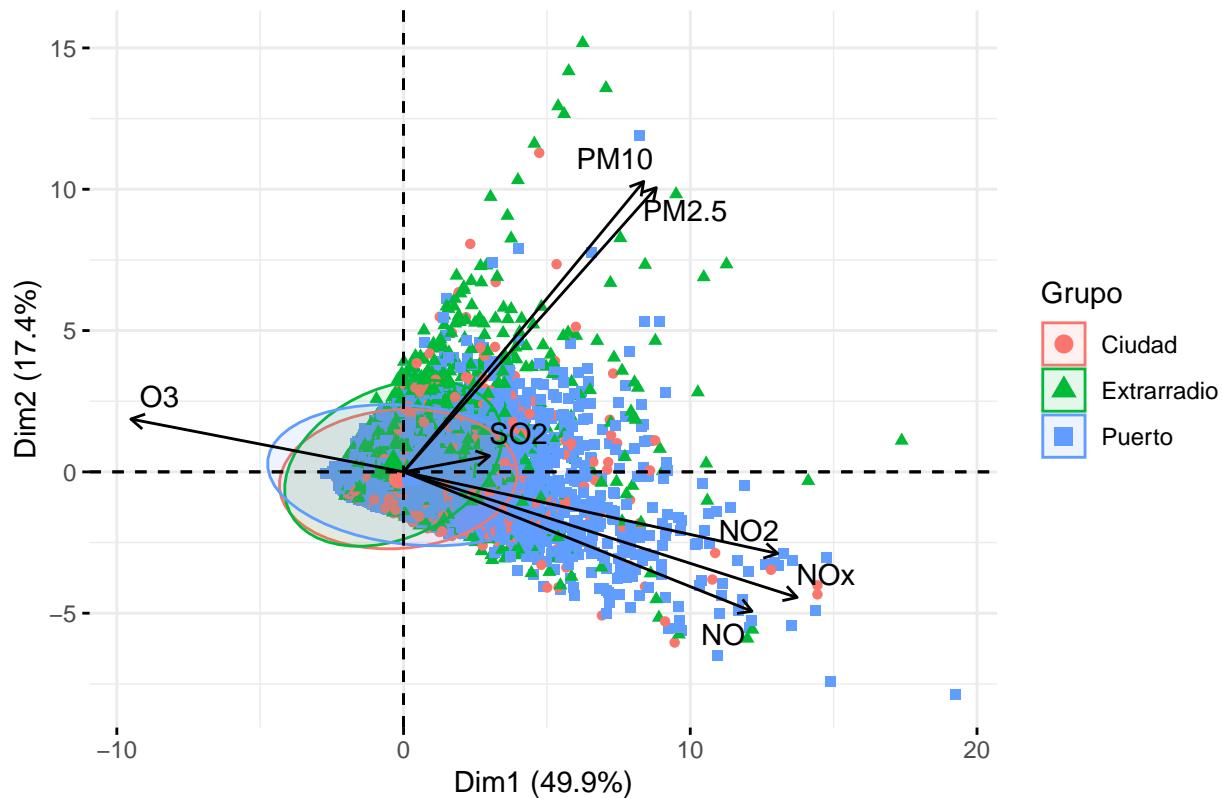
## Warning: There were 2 warnings in 'mutate()'.
## The first warning was:
## i In argument: 'Grupo = case_when(...)'.
## Caused by warning in '==.default':
## ! longer object length is not a multiple of shorter object length
## i Run 'dplyr::last_dplyr_warnings()' to see the 1 remaining warning.

# 2) PCA (Grupo como variable cuali.sup en la primera columna)
res.pca_grp <- PCA(
  df_pca_grp,
  quali.sup = 1,
  scale.unit = TRUE,
  graph      = FALSE
)

# 4) Biplot PC1 vs PC2 coloreado por Grupo
fviz_pca_biplot(
  res.pca_grp,
  axes      = c(1, 2),
  habillage = "Grupo",
  addEllipses = TRUE,
  ellipse.level = 0.95,
  col.var    = "black",
  repel      = TRUE,
  label      = "var"
)

```

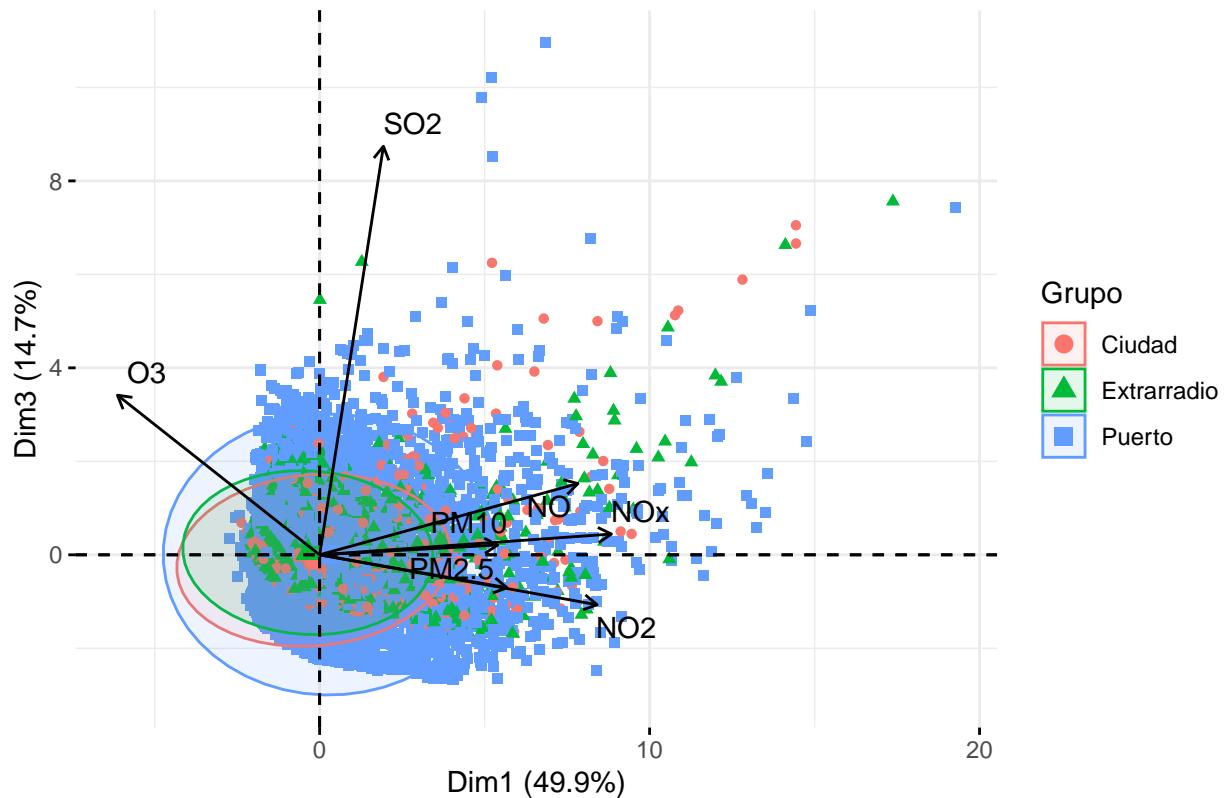
PCA – Biplot



Observamos más o menos lo mismo que antes. Las zonas de ‘puertos’ tienden a irse hacia la parte positiva de la PC1, lo que indica que tienden a generar más *PM10*, *PM2.5*, *NO*, *NO₂*, *NO_x* y *SO₂*. Las zonas en el extrarradio tienden a generar más *PM10* y *PM2.5*. Pero en general todos están distribuidos similarmente.

```
set.seed(100)
fviz_pca_biplot(
  res.pca_grp,
  axes      = c(1, 3),
  habillage = "Grupo",
  addEllipses = TRUE,
  ellipse.level = 0.95,
  col.var   = "black",
  repel     = TRUE,
  label     = "var"
)
```

PCA – Biplot



Parece ser que:

- La CP1 explica la combustión de fosiles, como tráfico, puerto,....
- La CP2 parece explicar algo de meteorología
- La CP3 no le vemos explicación.

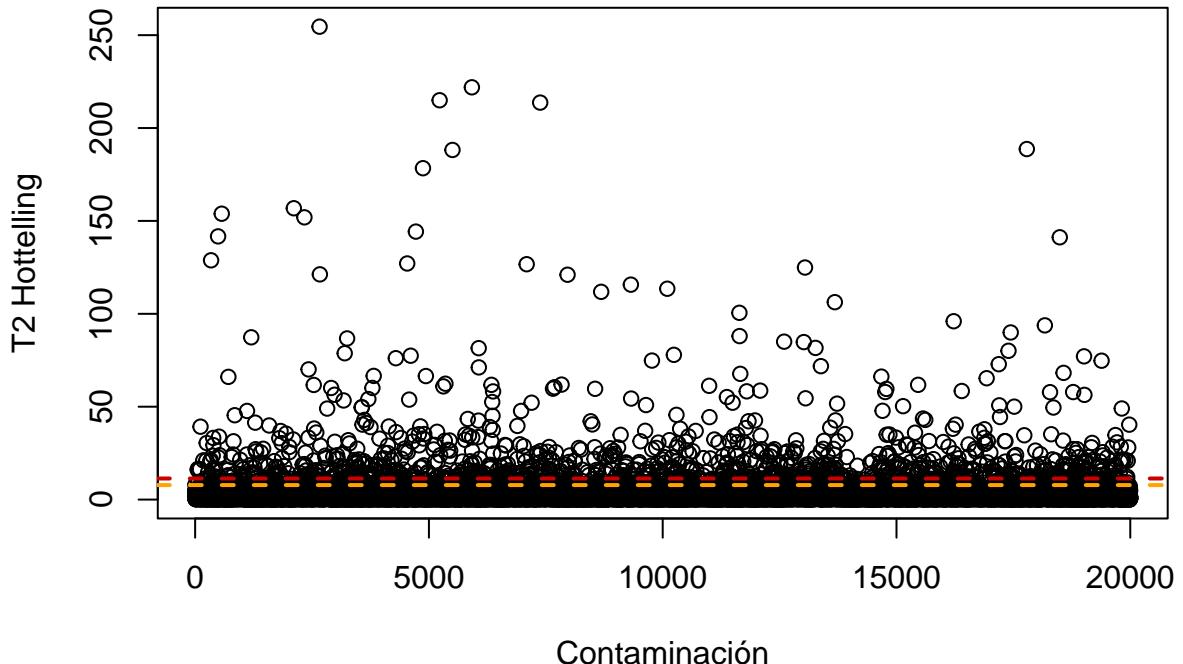
Vamos a validar el modelo.

```
set.seed(100)
K <- 3

res.pca <- PCA(df_pca_20k,
                 scale.unit = TRUE,
                 graph = FALSE,
                 ncp = K,
                 quali.sup = which(names(df_pca_20k) == "Origen"))

misScores = res.pca$ind$coord[,1:K]
miT2 = colSums(t(misScores)**2)/eig.val[1:K,1])
I = nrow(df_pca_20k)
F95 = K*(I**2 - 1)/(I*(I - K)) * qf(0.95, K, I-K)
F99 = K*(I**2 - 1)/(I*(I - K)) * qf(0.99, K, I-K)

plot(1:length(miT2), miT2, type = "p", xlab = "Contaminación", ylab = "T2 Hottelling")
abline(h = F95, col = "orange", lty = 2, lwd = 2)
abline(h = F99, col = "red3", lty = 2, lwd = 2)
```



Observamos que la gran mayoría de los puntos se encuentra por debajo de esta línea, lo que indica que esas combinaciones de contaminantes se consideran normales dentro de la variabilidad esperada. En cambio, hay un número razonable de observaciones que superan el umbral: esos valores atípicos señalan episodios extremos o inusuales de contaminación, ya sea por picos puntuales de emisión o por condiciones meteorológicas excepcionales.

```
idx_max_anomalia <- which.max(miT2)

# Mostrar el valor de  $T^2$  más alto
miT2[idx_max_anomalia]

##      2661
## 254.5608

# Mostrar la fila correspondiente en los datos originales
df[idx_max_anomalia, ]

## # A tibble: 1 x 9
##   FechaHora        Origen     NO    NO2    NOx    O3   PM10  PM2.5    SO2
##   <dttm>        <fct>    <int> <int> <int> <int> <int> <int>
## 1 2023-04-22 12:00:00 Quart de Poblet 1      9     10    98     38    21     4
```

Clustering

Ahora vamos a hacer un clustering.

```
# 1) Agregar la concentración media de cada contaminante por estación
df_sto <- datos_limpios %>%
  group_by(Origen) %>%
  summarise(across(
```

```

c(NO, NO2, NOx, O3, PM10, `PM2.5`, SO2),
~ mean(.x, na.rm = TRUE)
)) %>%
column_to_rownames("Origen")

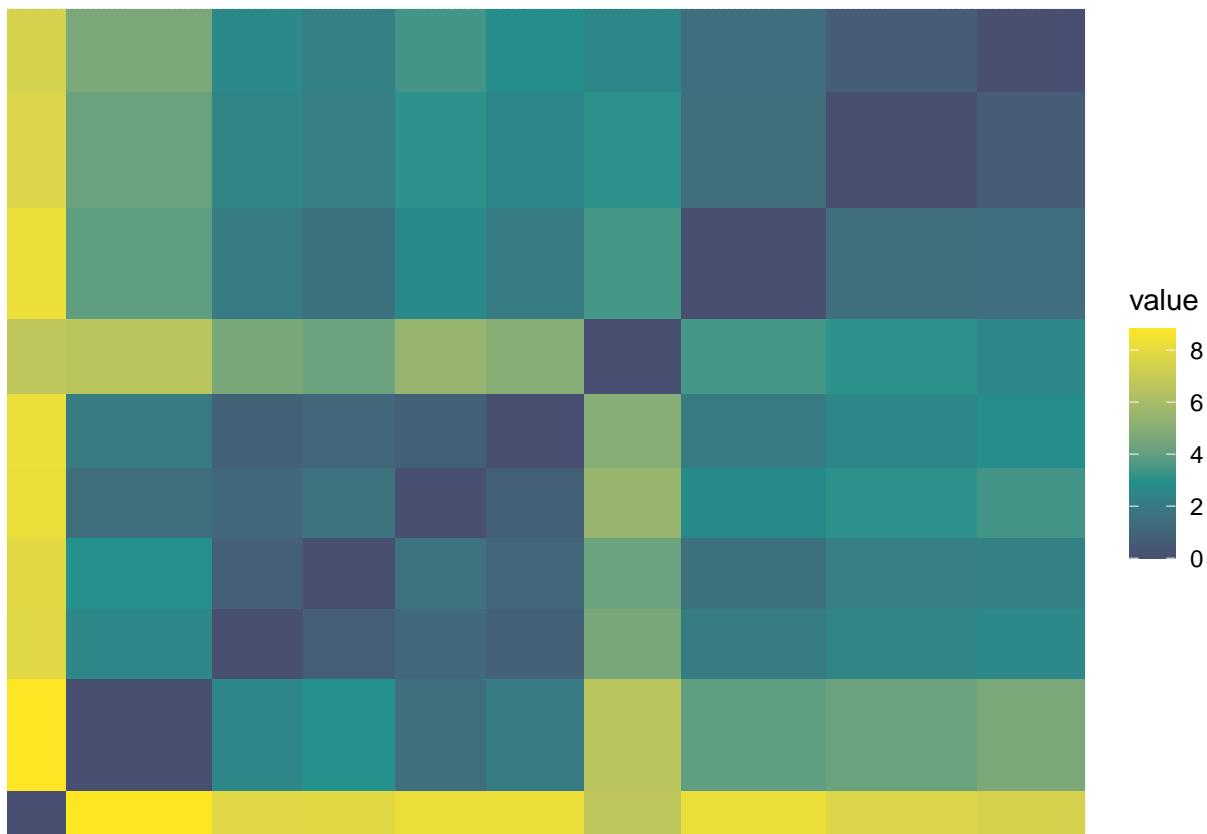
# 2) Escalar los datos
df_scaled <- scale(df_sta)

# Convertir matriz a data frame antes de usar sample_n
vars_sample <- as.data.frame(df_scaled) %>% sample_n(200, replace = TRUE)

# Calcular la matriz de distancias (opcional: usar scale() si hace falta)
midist <- get_dist(scale(vars_sample), method = "euclidean")

# Visualizar la matriz
fviz_dist(midist,
          show_labels = FALSE,
          lab_size = 0.3,
          gradient = list(low = "#440154", mid = "#21908C", high = "#FDE725"))

```



```

# 1) Medias de contaminantes por estación
df_sta <- datos_limpios %>%
  group_by(Origen) %>%
  summarise(
    NO      = mean(NO,     na.rm = TRUE),
    NO2     = mean(NO2,    na.rm = TRUE),

```

```

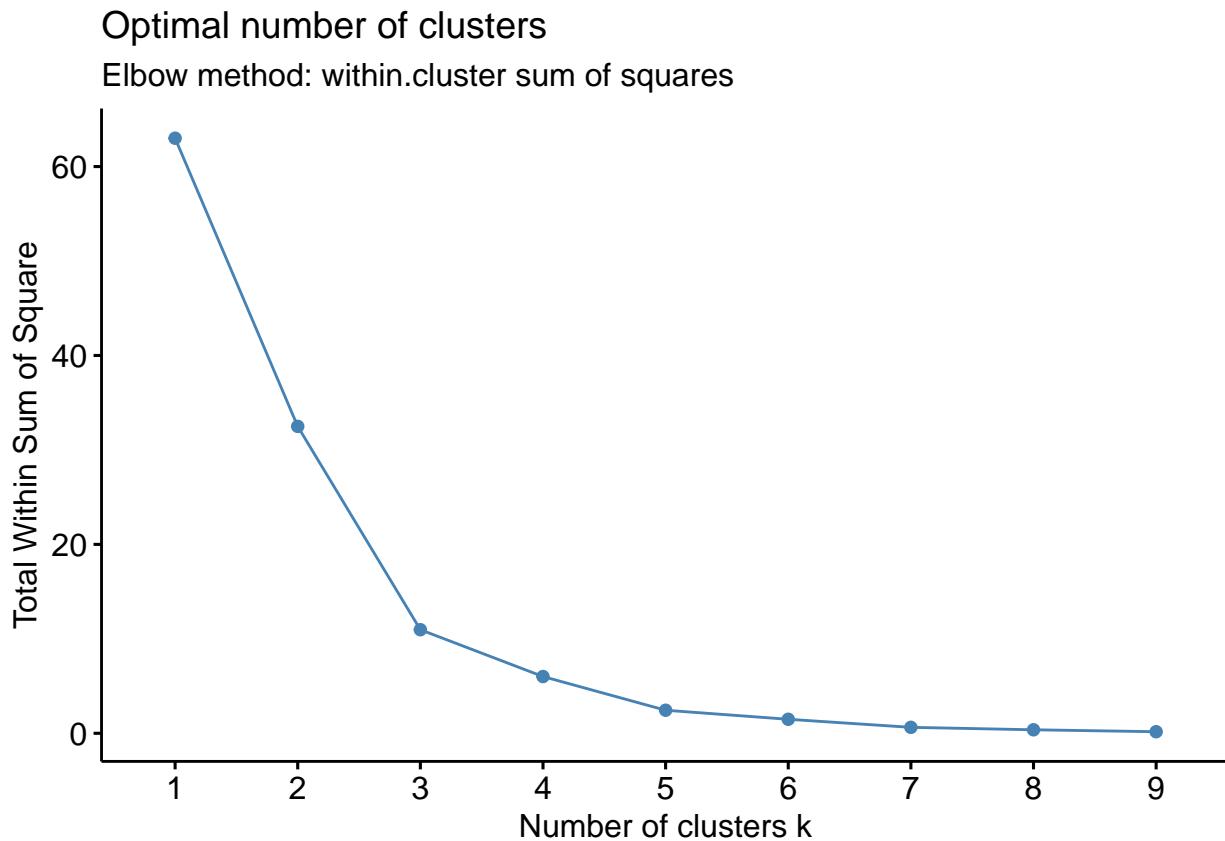
NOx    = mean(NOx,     na.rm = TRUE),
O3     = mean(O3,      na.rm = TRUE),
PM10   = mean(PM10,    na.rm = TRUE),
PM2.5  = mean(`PM2.5`, na.rm = TRUE),
S02    = mean(S02,     na.rm = TRUE)
) %>%
column_to_rownames("Origen")

# 2) Escalado (media=0, sd=1)
df_scaled <- scale(df_st)

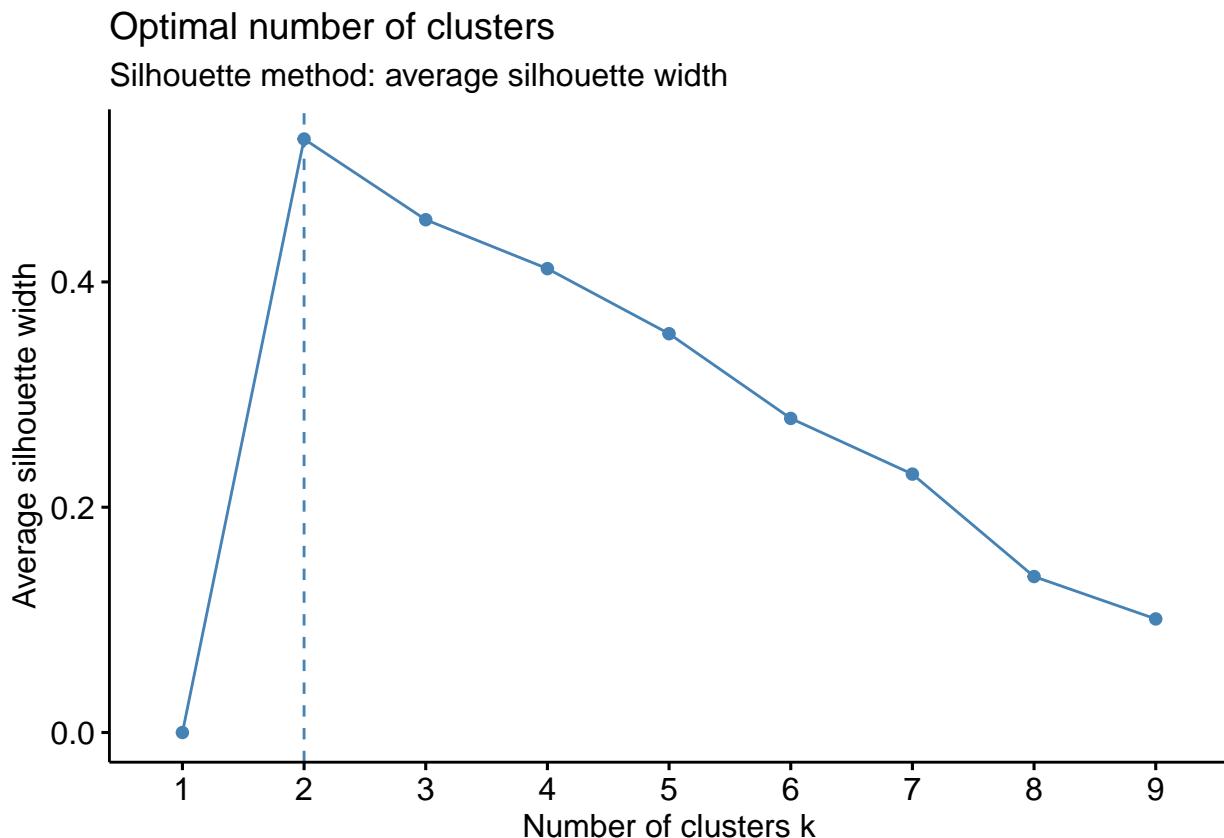
# 3) Rango de k a probar (n estaciones - 1, o hasta 10)
n_stations <- nrow(df_scaled)
max_k       <- min(10, n_stations - 1)

# 4a) Elbow method (WSS)
fviz_nbclust(
  df_scaled,
  FUNcluster = kmeans,
  method      = "wss",
  k.max       = max_k,
  nstart      = 25
) +
  labs(subtitle = "Elbow method: within-cluster sum of squares")

```



```
# 4b) Silhouette method
fviz_nbclust(
  df_scaled,
  FUNcluster = kmeans,
  method      = "silhouette",
  k.max       = max_k,
  nstart      = 25
) +
  labs(subtitle = "Silhouette method: average silhouette width")
```



```
# 5) Clustering jerárquico (distancia Euclídea + Ward)
dist_mat <- dist(df_scaled, method = "euclidean")
hc        <- hclust(dist_mat,     method = "ward.D2")

# 6) Elegir k (por ejemplo 3, según los gráficos anteriores)
k_opt <- 3

# 7) Cortar el árbol y extraer clusters
station_clusters <- cutree(hc, k = k_opt)

# 8) Data.frame de etiquetas por estación
station_clusters_df <- data.frame(
  Origen  = names(station_clusters),
  cluster = factor(station_clusters)
)
```

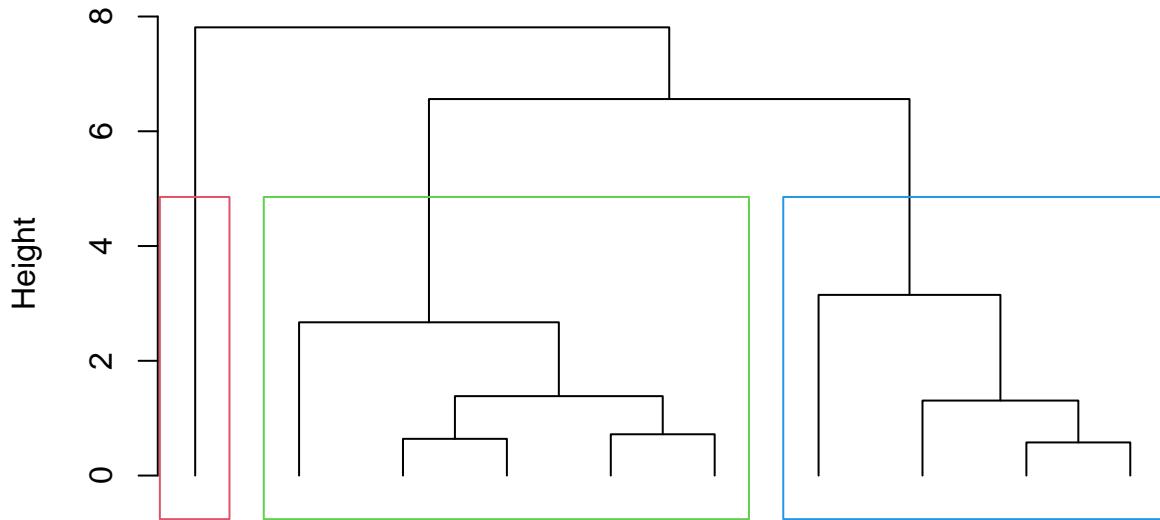
```

# 9) Unir etiquetas al dataset original
datos_clustered <- datos_limpios %>%
  left_join(station_clusters_df, by = "Origen")

# 10) Dendrograma con cortes marcados
plot(hc,
      labels = FALSE,
      hang    = -1,
      main   = "Dendrograma: Ward + Euclidiana")
rect.hclust(hc, k = k_opt, border = 2:(k_opt+1))

```

Dendrograma: Ward + Euclidiana



```

dist_mat
hclust (*, "ward.D2")

```

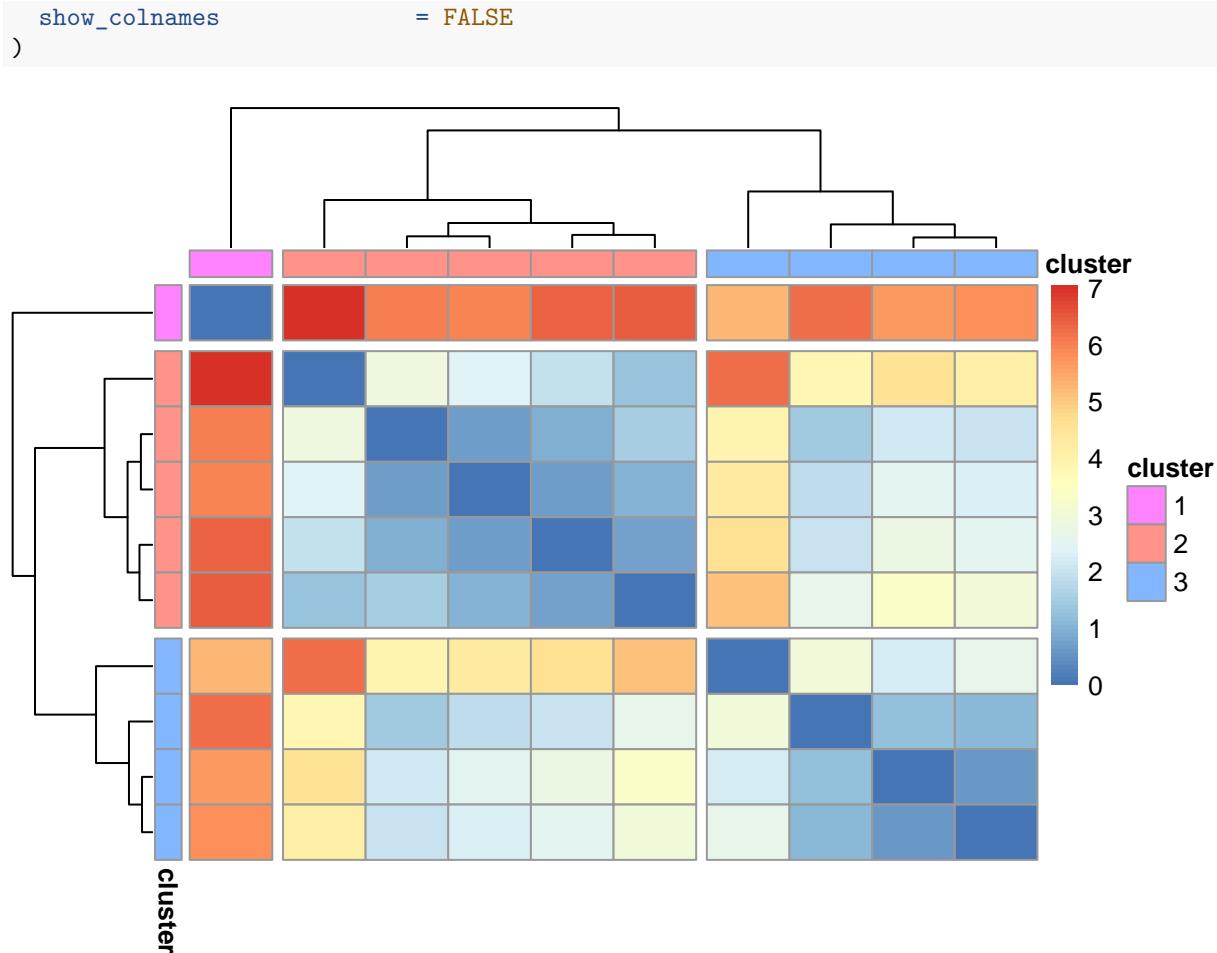
```

rownames(station_clusters_df) <- NULL

annotation <- station_clusters_df %>%
  column_to_rownames("Origen")

# Llamada a pheatmap
pheatmap(
  as.matrix(dist_mat),
  clustering_distance_rows = dist_mat,
  clustering_distance_cols = dist_mat,
  clustering_method        = "ward.D2",
  cutree_rows               = k_opt,
  cutree_cols               = k_opt,
  annotation_row            = annotation,
  annotation_col            = annotation,
  show_rownames             = FALSE,

```

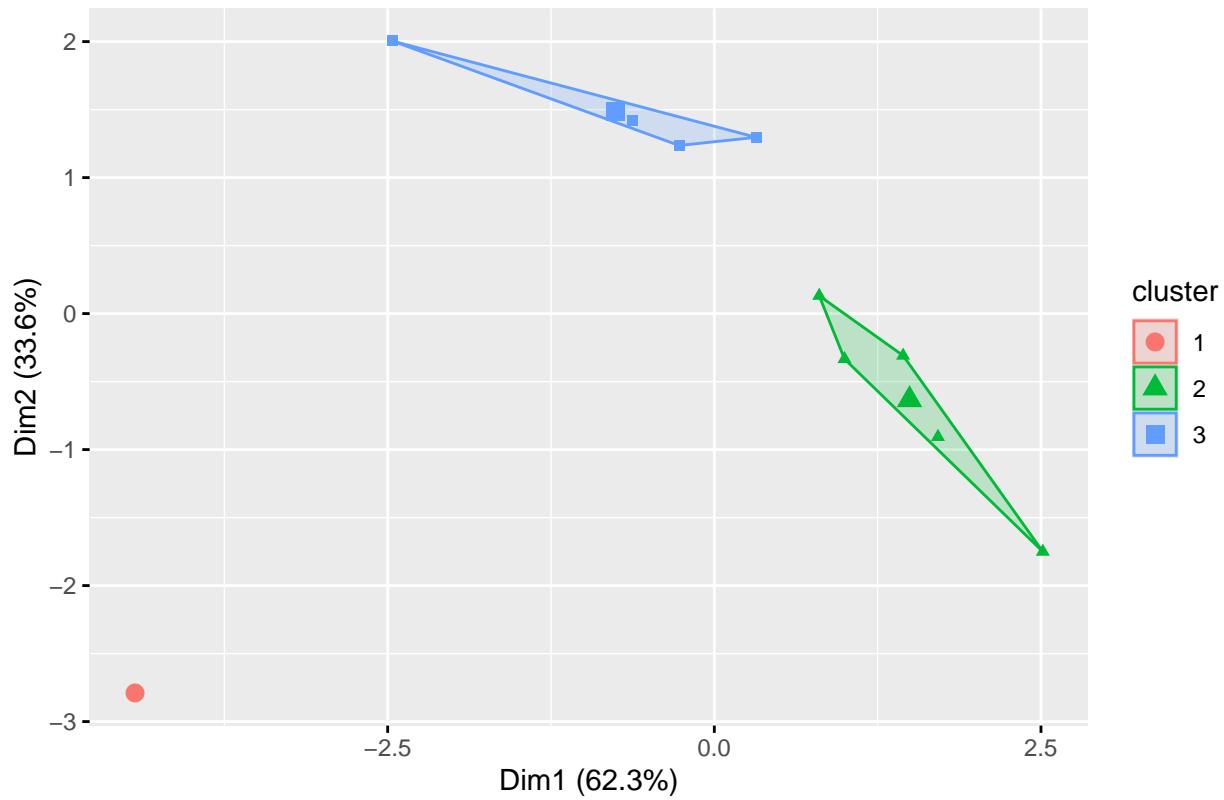


```

# 11) Proyección PCA coloreada por cluster
fviz_cluster(
  list(data    = df_scaled,
       cluster = station_clusters),
  geom      = "point",
  ellipse   = TRUE,
  show.clust.cent = TRUE
) +
  labs(title = paste0("PCA de estaciones (k = ", k_opt, ")"))

```

PCA de estaciones (k = 3)



```
# 12) Centros de cada cluster en las unidades originales
centers_unscaled <- df_stations
centers_unscaled$cluster <- station_clusters[rownames(centers_unscaled)]
centers_summary <- centers_unscaled %>%
  as.data.frame() %>%
  group_by(cluster) %>%
  summarise(
    NO      = mean(NO),
    NO2     = mean(NO2),
    NOx     = mean(NOx),
    O3      = mean(O3),
    PM10    = mean(PM10),
    PM2.5   = mean(PM2.5),
    S02     = mean(S02)
  )
  print(centers_summary)

## # A tibble: 3 x 8
##   cluster    NO    NO2    NOx    O3    PM10  PM2.5    S02
##   <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1       1 10.3  19.1  34.7  25.5 129.   25.3  1.06
## 2       2  4.33  13.9  20.3  53.6  17.5  10.6  3.61
## 3       3 12.6  24.2  43.4  46.4  21.2  10.6  3.39
```

Hemos seleccionado 3 clusters ya que creemos que es la mejor combinación, ya que el 3 tiene el coeficiente de silueta 2º más alto y por el metodo del codo vemos que en el cluster 3/4 se observa la diferencia.

```

station_clusters_df %>%
  arrange(cluster) %>%
  group_by(cluster) %>%
  summarise(
    estaciones = paste(Origen, collapse = ", ")
  )

## # A tibble: 3 x 2
##   cluster estaciones
##   <fct>   <chr>
## 1 1       Massanassa_UM
## 2 2       Quart de Poblet, Torrent-El Vedat, València - Av. França, València - ~
## 3 3       Sedavi UM, València - Pista de Silla, València Port llit antic Túria,~

station_clusters_df

## # A tibble: 10 x 2
##   Origen cluster
##   <chr>   <dbl>
## 1 Massanassa_UM     1
## 2 Quart de Poblet     2
## 3 Sedavi UM         3
## 4 Torrent-El Vedat     2
## 5 València - Av. França     2
## 6 València - Molí del Sol     2
## 7 València - Pista de Silla     3
## 8 València - Politécnic     2
## 9 València Port llit antic Túria     3
## 10 València Port Moll Trans. Ponent    3

```

Aquí vemos en qué clusters se ha asignado cada estación de medición.