USO Y RESPALDO DE SCRIPTS PRIORITARIOS PARA LA VIGILANCIA Y PRONOSTICO AMBIENTAL ATMOSFERICO

Instructivo: IN-DMA-005

Versión: 01

SUBDIRECCIÓN DE EVALUACIÓN DEL AMBIENTE ATMOSFÉRICO – DIRECCIÓN DE METEOROLOGÍA Y EVALUACIÓN AMBIENTAL ATMOSFÉRICA

Elaborado por:	**Firma:**
Jhojan Pool Rojas Quincho Subdirector Subdirección de Evaluación del Ambiente Atmosférico	Firma Digital Senamhi Firmado digitalmente por ROJAS QUINCHO Jhojan Pool FAU 20131366028 hard Motivo: Soy el autor del documento Fecha: 20.12.2023 15:58:13 -05:00
Elvis Anthony Medina Dionicio Analista en Vigilancia Atmosférica Global Subdirección de Evaluación del Ambiente Atmosférico	Firma Digital Senamhi Firmado digitalmente por MEDINA DIONICIO Elvis Anthony FAU 20131366028 soft Motivo: Soy el autor del documento Fecha: 20.12.2023 15:56:21 -05:00
Hanns Kevin Gómez Muñoz Asistente Ambiental Subdirección de Evaluación del Ambiente Atmosférico	Firma Digital Senamhi Firmado digitalmente por GÓMEZ MUÑOZ Hanns Kevin FAU 20131366028 soft Motivo: Soy el autor del documento Fecha: 20.12.2023 14:57:34 -05:00
Revisado por:	**Firma:**
Sonia del Carmen Huamán Lozano Directora Unidad de Modernización y Gestión de la Calidad	Firma Digital Senamhi Firmado digitalmente por HUAMAN LOZANO Sonia Del Carmen FAU 20131366028 hard Motivo: Soy el autor del documento Fecha: 20.12.2023 16:42:11 -05:00

Aprobado por:	Firma:
Jhojan Pool Rojas Quincho Director Dirección de Meteorología y Evaluación Ambiental Atmosférica	Firma Digital Firmado digitalmente por ROJAS QUINCHO Jhojan Pool FAU 20131366028 hard Motivo: Soy el autor del documento Fecha: 21.12.2023 10:30:25 -05:00

ÍNDICE

1. OBJETIVO

Definir las actividades para el uso y respaldo de los sripts prioritarios para la vigilancia y pronóstico ambiental atmosférico de la Subdirección de Evaluación del Ambiente Atmosférico (SEA) del Servicio Nacional de Meteorología e Hidrología del Perú (SENAMHI).

2. DESARROLLO

El presente documento, se centrará en detallar el uso scripts y definir las actividades que permitan efectuar copias de respaldo y resguardo de sripts prioritarios para el desarrollo de actividades de los productos de la SEA.

2.1. Pronóstico de calidad del aire

2.1.1. Gráfica de 24 horas

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
library(ggplot2)
library(openair)
library(latticeExtra)
library(directlabels)
library(dplyr)
library(scales)
library(readxl)
library(egg)
library(gtable)
library(gridExtra)
library(grid)
library(tidyverse)
```

Luego se realiza la descarga la información de los contaminates del aire para la elaboración de las gráficas. La siguiente línea de código selecciona la carpeta de trabajo, descarga un archivo exel.csv del googledrive y formatea la fecha. Asimismo, identifica los valores menores a 0 y se les asigna la etiqueta de NA.

```
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO")

id <- "1QLn-aHHDzXapJqbPL4Bce4hBK3MyxbFK"
datos = read_delim(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
                   col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

datoscont = datos %>% openair::selectByDate(.,start = as.character(Sys.Date()-2),end = as.character(Sys.Date()))
cambio_na = function(x){ifelse(x<0,NA,x)}
df_sin_na = data.frame(sapply(datoscont,cambio_na))
df_sin_na$date = datoscont$date
datoscont = df_sin_na
```

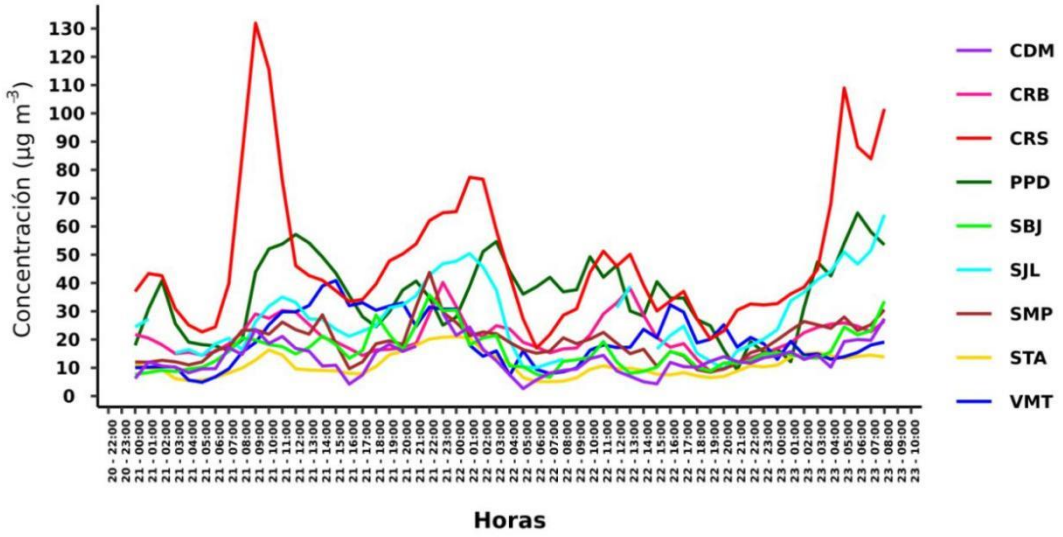
La siguiente gráfica presenta los promedios horarios de las últimas 65 horas. El código elabora la gráfica y la guarda en el directorio seleccionado en formato .jpg con el nombre **PM25HORAS**.

```
PM25Horas <- ggplot(data = datoscont,aes(x=date))+
  geom_line(aes(y=PPD,colour="PPD"),size=0.6)+
  geom_line(aes(y=CRB,colour="CRB"),size=0.6)+
```

```
 geom_line(aes(y=STA,colour="STA"),size=0.6)+
 geom_line(aes(y=VMT,colour="VMT"),size=0.6)+
 geom_line(aes(y=SJL,colour="SJL"),size=0.6)+
 geom_line(aes(y=SMP,colour="SMP"),size=0.6)+
 geom_line(aes(y=CRS,colour="CRS"),size=0.6)+
 geom_line(aes(y=SBJ,colour="SBJ"),size=0.6)+
 geom_line(aes(y=CDM,colour="CDM"),size=0.6)+
 labs(x="\nHoras", y=quickText("Concentración (µg/m3)"))+
 scale_x_datetime(date_breaks = "1 hour", labels = date_format("%d - %H:%M"))+
 scale_y_continuous(breaks = seq(0,200,10))+
 scale_colour_manual("",values=c("PPD"="darkgreen","CRB"="deeppink","SMP"="brown","CRS"="red",
                     "STA"="gold","SJL"="cyan","CDM"="purple","SBJ"="green","VMT"="blue"))+ # "CDM"="purple"
#scale_colour_manual("",values=estac_order)+#,
#labels = estac_order) +
 theme_classic()+
 theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.5)),
       axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
 theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
 theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
 theme(legend.text=element_text(size=7,face = "bold"))+
 theme(legend.margin = margin(1,1,1,1))+geom_vline(xintercept = 09-00:00)+
 theme(legend.position = "right")
PM25Horas
ggsave(filename = "PM25HORAS.jpg", plot =PM25Horas, width = 15.0, height = 7.75, dpi = 1000, units = "cm")
```

El resultado se presenta a continuación:

Figura N° 1. Promedio horario del $PM_{2.5}$



2.1.2. Gráfica de promedio movil de 24 horas

Para la ejecución del script es necesario el cálculo de promedios moviles de 24 horas, para lo cual es necesario utilizar la función **rollingMean ()** del paquete Openair, como se muestra a continuación:

```
df_sin_na = data.frame(sapply(datos,cambio_na))
df_sin_na$date = datos$date
datos = df_sin_na

PM25_24h <- rollingMean(datos, pollutant = "CDM", new.name = "CDM24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "SBJ", new.name = "SBJ24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "CDM", new.name = "CDM24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "STA", new.name = "STA24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "VMT", new.name = "VMT24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "CRS", new.name = "CRS24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "SJL", new.name = "SJL24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "SMP", new.name = "SMP24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "CRB", new.name = "CRB24h", width=24, data.thresh = 75, align = "right")
PM25_24h <-rollingMean(PM25_24h, pollutant = "PPD", new.name = "PPD24h", width=24, data.thresh = 75, align = "right")
PM25_24h_dia <-selectByDate(PM25_24h, start = as.character(Sys.Date()-2),end = as.character(Sys.Date())) #ACTUALIZAR F
ECHA
```

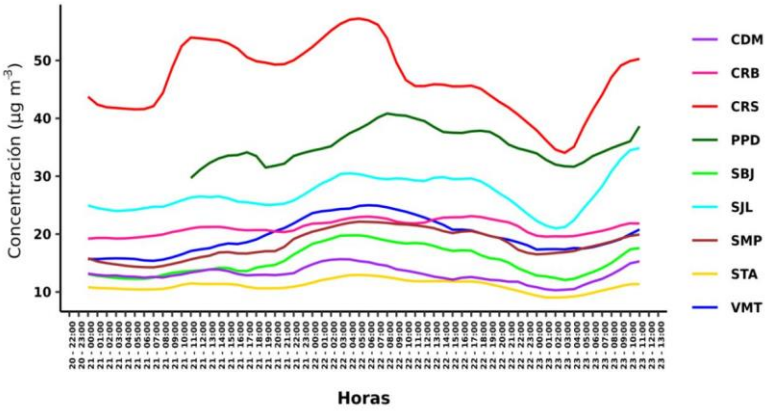
La siguiente gráfica presenta los promedios moviles de 24 horas. El código elabora la gráfica y la guarda en el directorio seleccionado en formato .jpg con el nombre **PM25movil**.

```
datoscont2<-PM25_24h_dia
PM25Horas <- ggplot(data = datoscont2,aes(x=date))+geom_line(aes(y=PPD24h,colour="PPD"),size=0.6)+
 geom_line(aes(y=STA24h,colour="STA"),size=0.6)+geom_line(aes(y=VMT24h,colour="VMT"),size=0.6)+geom_line(aes(y=
 CRB24h,colour="CRB"),size=0.6)+geom_line(aes(y=SJL24h,colour="SJL"),size=0.6)+
 geom_line(aes(y=SMP24h,colour="SMP"),size=0.6)+geom_line(aes(y=CRS24h,colour="CRS"),size=0.6)+
 geom_line(aes(y=SBJ24h,colour="SBJ"),size=0.6)+geom_line(aes(y=CDM24h,colour="CDM"),size=0.6)+
 labs(x="\nHoras", y=quickText("Concentración (µg/m3)"))+scale_x_datetime(date_breaks = "1 hour", labels = date_format("%
 d - %H:%M"))+
 #scale_y_continuous(breaks = seq(0,200,10))+
 scale_colour_manual("",values=c("PPD"="darkgreen","CRB"="deeppink","SMP"="brown","CRS"="red",
                     "STA"="gold","SJL"="cyan","CDM"="purple","SBJ"="green","VMT"="blue"))+ #"CDM"="purple"
 theme_classic()+
 theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.5)),
       axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
 theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
 theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
 theme(legend.text=element_text(size=7,face = "bold"))+
 theme(legend.margin = margin(1,1,1,1))
PM25Horas

ggsave(filename = "PM25movil.jpg", plot =PM25Horas, width = 15.0, height = 7.75, dpi = 1000, units
 = "cm")
```

El resultado se presenta a continuación:

Figura N° 2. Promedio movil de 24 horas del $PM_{2.5}$



2.1.3. Estados de Calidad del Aire

Para la ejecución del script es necesario la determinación de los Estados de Calidad del Aire, según la Agencia de Protección Ambiental de los Estados Unidos (EPA). En ese sentido, las siguientes lineas de código con los promedios moviles de la base de datos "datoscont2", clasifica los datos con las etiquetas "Bueno", "Moderado","Insalubre GS" e "Insalubre", por cada estación y crea una base de datos llamada **PM25_INCA**.

```
PM25_INCA<-datoscont2

PM25_INCA$AQI_CDM <- ifelse(PM25_INCA$CDM24h<12,"Bueno",
                    ifelse(PM25_INCA$CDM24h>12.0001 & PM25_INCA$CDM24h<35.4,"Moderado",
                        ifelse(PM25_INCA$CDM24h>35.40001 & PM25_INCA$CDM24h<55.4,"Insalubre GS","Insalubre")))

PM25_INCA$AQI_SBJ <- ifelse(PM25_INCA$SBJ24h<12,"Bueno",
                    ifelse(PM25_INCA$SBJ24h>12.0001 & PM25_INCA$SBJ24h<35.4,"Moderado",
                        ifelse(PM25_INCA$SBJ24h>35.40001 &  PM25_INCA$SBJ24h<55.4,"Insalubre GS","Insalubre")))

PM25_INCA$AQI_STA <- ifelse(PM25_INCA$STA24h<12,"Bueno",
                    ifelse(PM25_INCA$STA24h>12.0001 &  PM25_INCA$STA24h<35.4,"Moderado",
                        ifelse(PM25_INCA$STA24h>35.40001 &  PM25_INCA$STA24h<55.4,"Insalubre GS","Insalubre")))

PM25_INCA$AQI_VMT <- ifelse(PM25_INCA$VMT24h<12,"Bueno",
                    ifelse(PM25_INCA$VMT24h>12.00001 & PM25_INCA$VMT24h<35.4,"Moderado",
                        ifelse(PM25_INCA$VMT24h>35.40001 & PM25_INCA$VMT24h<55.4,"Insalubre GS","Insalubre")))


PM25_INCA$AQI_CRS <- ifelse(PM25_INCA$CRS24h<12,"Bueno",
                    ifelse(PM25_INCA$CRS24h>12.0001 & PM25_INCA$CRS24h<35.4,"Moderado",
                        ifelse(PM25_INCA$CRS24h>35.40001 & PM25_INCA$CRS24h<55.4,"Insalubre GS","Insalubre")))

PM25_INCA$AQI_SJL <- ifelse(PM25_INCA$SJL24h<12,"Bueno",
                    ifelse(PM25_INCA$SJL24h>12.0001 & PM25_INCA$SJL24h<35.4,"Moderado",
                        ifelse(PM25_INCA$SJL24h>35.40001 & PM25_INCA$SJL24h<55.4,"Insalubre GS","Insalubre")))

PM25_INCA$AQI_SMP <- ifelse(PM25_INCA$SMP24h<12,"Bueno",
                    ifelse(PM25_INCA$SMP24h>12.0001 & PM25_INCA$SMP24h<35.4,"Moderado",
                        ifelse(PM25_INCA$SMP24h>35.40001&  PM25_INCA$SMP24h<55.4,"Insalubre GS","Insalubre")))

PM25_INCA$AQI_CRB <- ifelse(PM25_INCA$CRB24h<12,"Bueno",
                    ifelse(PM25_INCA$CRB24h>12.0001 & PM25_INCA$CRB24h<35.4,"Moderado",
                        ifelse(PM25_INCA$CRB24h>35.40001&  PM25_INCA$CRB24h<55.4,"Insalubre GS","Insalubre")))

PM25_INCA$AQI_PPD <- ifelse(PM25_INCA$PPD24h<12,"Bueno",
                    ifelse(PM25_INCA$PPD24h>12.0001 & PM25_INCA$PPD24h<35.4,"Moderado",
                        ifelse(PM25_INCA$PPD24h>35.40001&  PM25_INCA$PPD24h<55.4,"Insalubre GS","Insalubre")))
```

Luego de la creación del Dataframe **PM25_INCA,** se crean gŕaficas por cada estación con las siguientes líneas de código:

```
PM25_CDM <-ggplot(data = PM25_INCA, aes(x=date, y=CDM24h, fill=AQI_CDM))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,60,10),limits = c(0,60))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="red"))+ theme (p
anel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_CDM
```

```
PM25_SBJ <-ggplot(data = PM25_INCA, aes(x=date, y=SBJ24h, fill=AQI_SBJ))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,80,10),limits = c(0,50))+
```

```
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="red"))+ theme (p
anel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_SBJ

PM25_STA <-ggplot(data = PM25_INCA, aes(x=date, y=STA24h, fill=AQI_STA))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,100,10),limits = c(0,50))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="red"))+ theme (p
anel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_STA

PM25_VMT <-ggplot(data = PM25_INCA, aes(x=date, y=VMT24h, fill=AQI_VMT))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,120,10),limits = c(0,50))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="red"))+ theme (p
anel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_VMT

PM25_CRS <-ggplot(data = PM25_INCA, aes(x=date, y=CRS24h, fill=AQI_CRS))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,100,10),limits = c(0,70))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="red"))+ theme (p
anel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_CRS


PM25_SJL <-ggplot(data = PM25_INCA, aes(x=date, y=SJL24h, fill=AQI_SJL))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,100,10),limits = c(0,50))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="red"))+ theme (p
anel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_SJL

PM25_SMP <-ggplot(data = PM25_INCA, aes(x=date, y=SMP24h, fill=AQI_SMP))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,80,10),limits = c(0,50))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="red"))+ theme (p
anel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_SMP

PM25_CRB <-ggplot(data = PM25_INCA, aes(x=date, y=CRB24h, fill=AQI_CRB))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,100,10),limits = c(0,60))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="orange"))+ theme
 (panel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_CRB
```

```
PM25_PPD <-ggplot(data = PM25_INCA, aes(x=date, y=PPD24h, fill=AQI_PPD))+
  geom_bar(color="Black",stat = "identity", position = "dodge")+scale_x_datetime(date_breaks = "1 hour", labels = date_format(
"%d - %H:%M"))+scale_y_continuous(breaks = seq(0,100,10),limits = c(0,50))+
  scale_fill_manual(values=c("Bueno"="green", "Moderado"="yellow","Insalubre GS"="orange","Insalubre"="orange"))+ theme
 (panel.background = element_rect (fill = 'white'))+ theme_bw()+#theme_linedraw()+
  theme(axis.title.x=element_blank())+ theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
                                              axis.text.y = element_text(face="bold", colour="black", size=rel(1), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(1)))+theme(legend.position="none")
PM25_PPD
```

Finalmente con la función grid.arrange () se elabora un panelPlot como se indica a continuación:

```
PM25_AML_LE<-grid.arrange(PM25_PPD,PM25_CRB,PM25_SMP,PM25_CRS, #PM25_SJL,
                          ncol=2
                          ,bottom = textGrob("Horas"))
ggsave(filename = "PM25_INCA_ESTE.jpg", plot =PM25_AML_LE, width = 40, height = 15, dpi = 1000, units = "cm")
```

```
PM25_AML_LS<-grid.arrange(PM25_STA,PM25_SBJ,PM25_CDM,PM25_VMT,
                          ncol=2,bottom = textGrob("Horas"))
ggsave(filename = "PM25_INCA_SUR.jpg", plot =PM25_AML_LS, width = 40, height = 15, dpi = 1000, units = "cm")
```

El resultado se presenta a continuación:

Figura N° 3. Estados de Calidad del Aire Lima Este

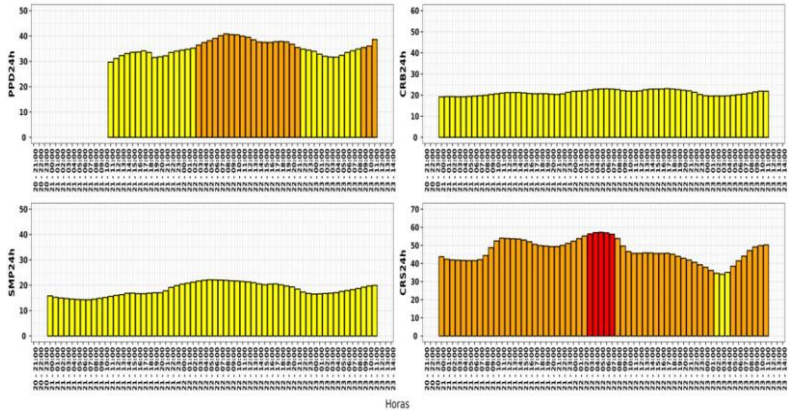
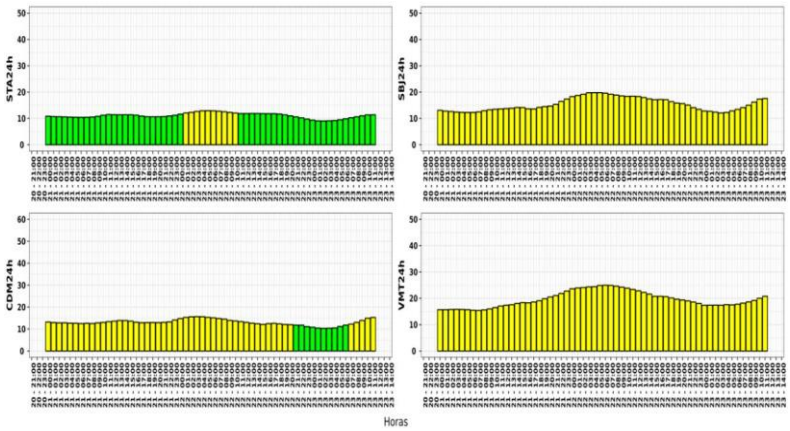


Figura N° 4. Estados de Calidad del Aire Lima Sur



2.1.4. Tránsito vehicular

Para la generación de la gráfica de tránsito vehicular es necesario la instalación de las siguientes librerias en Python:

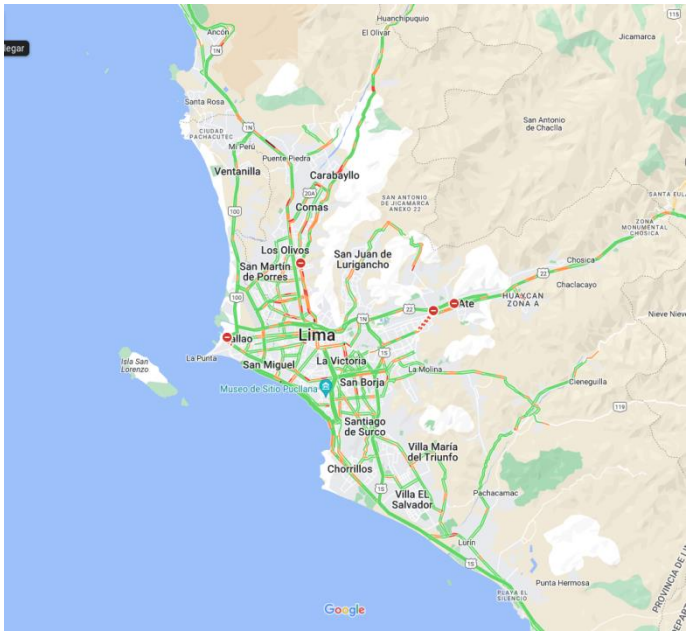
```
import warnings,os,pickle,glob,time,datetime,traceback
warnings.filterwarnings("ignore")
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver import Remote
from PIL import Image
```

A continuación se presentan unas líneas de código en pyhton del script “get_traffic.py”que capturan imágenes del transito vehicuar para Líma Metropolitana de GoogleTraficc. La imagen resultante, esta en formato .png y su nombre es asignado con el prefijo”**Trafico”, seguido del año, mes, dia y la hora de captura: ejemplo Trafico_2023-09-26_17 hrs.png.**

```
def crop_trafic(files):
 for i in files:
  im = Image.open(i)
  left,top,right,bottom=400,0,im.size[0]-400,im.size[1] #600
  im=im.crop((left, top, right, bottom))
  im.save(i)

pathd=f"~/Escritorio/SEA/SERVIDOR/PANTERA/scripts/python"
url="https://www.google.com.pe/maps/@-12.0288429,-77.012588,11z/data=!5m1!1e1"
paths=f"~/Escritorio/SEA/SERVIDOR/PANTERA/scripts/python/trafico/"
now=datetime.datetime.now()
option=webdriver.FirefoxOptions()
option.add_argument('--headless')
option.add_argument('--private')
#browser=webdriver.Firefox(executable_path=f'{paths}/driver_firefox',
 #                        firefox_binary=f'{paths}/firefox/firefox',
  #                       options=option,)
browser = webdriver.Firefox ()
browser.get(url)
browser.maximize_window()
time.sleep(1)
afile=f"trafico/Trafico_{now.strftime('%Y-%m-%d')}_{now.strftime('%-H')} hrs.png"
browser.save_screenshot(afile)
browser.close()
browser.quit()
files = glob.glob(afile)
try:
 crop_trafic(files)
except Exception:
   traceback.print_exc()
   pass
```

Figura N° 5. Imagen de google Traficc



Luego de la captura de la imagen, con el script en Python “traffic.py” se realiza lo siguiente:

- ✓ Filtrar por colores la imagen (verde, naranja, rojo, marrón) y se les asigna los nombres, Rapido, Moderado, Lento y Muy Lento.
- ✓ Georrefenciación de la imagen filtrada.
- ✓ Aplicación de una mascara con un shapefile del Aŕea Metropolitana de Lima y Callao (AMLC), el cual corta la imagen en zonas.
- ✓ Calculo del porcentaje de los rangos Rapido, Moderado, Lento y Muy Lento por zona.
- ✓ Creación de un dataframe
- ✓ Elaboración de la figura de tránsito vehicular.

```
import cv2 as cv
import xarray as xr
import numpy as np
import pandas as pd
import geopandas as gp
import rioxarray,glob,re
from shapely.geometry import mapping
import pylab as plt
import cartopy.crs as ccrs
import cartopy,os,re #,utm
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER, LATITUDE_FORMATTER
from pyproj import Proj
from cartopy.io.shapereader import Reader
import matplotlib.dates as mdates
from cartopy.feature import ShapelyFeature

pathd0="/content/drive/MyDrive/SERVIDOR/PANTERA/briefing/"
pathd2=f"{pathd0}/TRAFICO/TRAFICO_AGOSTO/SETIEMBRE"
fname1=f"/content/drive/MyDrive/SERVIDOR/PANTERA/shapes/LIMA_CALLAO.shp"

shp = gp.read_file(fname1, crs="epsg:4326")
#from PIL import Image
#pathd0="/home/sea"
def bias_correction(aa):
 err=(100-sum(aa))/len(aa)
```

```
 for j,i in enumerate(aa):
  aa[j]=i+err
 return aa

lat2=-11.79335567915411
lat1=-12.257414228133253
lon1=-77.20553538038394
lon2=-76.81964061961604
l_muylento = np.array([46,4,4])
u_muylento = np.array([255,41,41])
l_lento = np.array([242,60,50])
u_lento = np.array([249,158,153])
l_rapido = np.array([86,205,86])
u_rapido = np.array([116,226,121])
l_moderado = np.array([255,151,77])
u_moderado = np.array([255,223,200])

def calc_perc(img):
 img = cv.imread(img)
 hsv = cv.cvtColor(img, cv.COLOR_BGR2RGB)

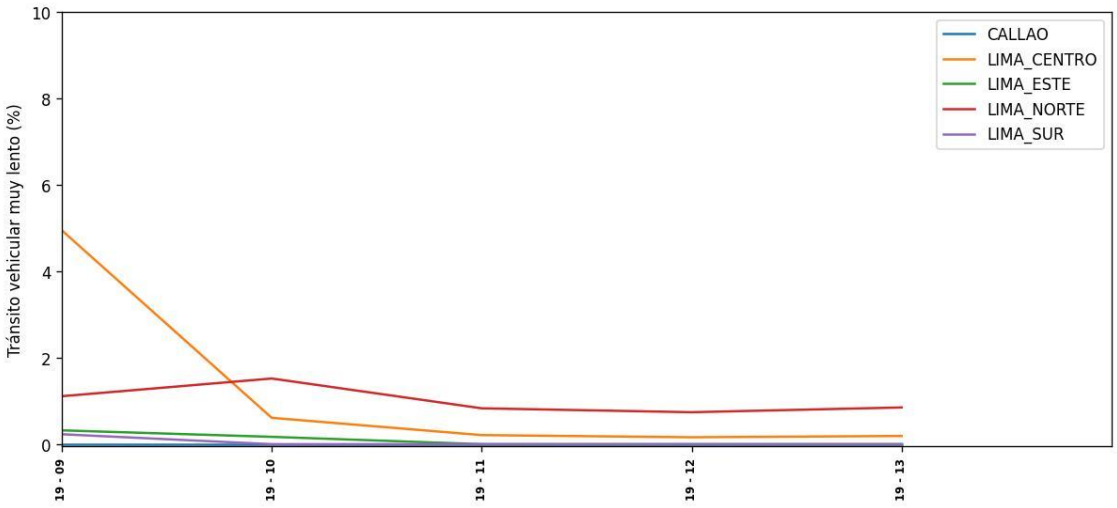
 mask1 = cv.inRange(hsv, l_rapido, u_rapido)
 mask2 = cv.inRange(hsv, l_moderado, u_moderado)
 mask4 = cv.inRange(hsv, l_muylento, u_muylento)
 mask3 = cv.inRange(hsv, l_lento, u_lento)
 mask=mask1+mask2+mask3+mask4
 lats=np.linspace(lat2,lat1,mask.shape[0])
 lons=np.linspace(lon1,lon2,mask.shape[1])
 dss=[xr.DataArray(data=i,dims=["lat","lon"],coords=dict(lon=lons,lat=lats)) for i in [mask,mask1,mask2,mask3,mask4]]
# shp = gp.read_file(fname1, crs="epsg:4326")
 dst=dss[0].rio.set_spatial_dims(x_dim="lon", y_dim="lat")
 dst.rio.write_crs("epsg:4326", inplace=True)
 dsf={}
 for j,k in enumerate(shp.layer.tolist()):
  shp2=shp.loc[[j]]
  dsy=dst.rio.clip(shp2.geometry.apply(mapping),shp2.crs,drop=False)
  ds2=[]
  for i in dss[1:]:
   ds0=i.rio.set_spatial_dims(x_dim="lon",y_dim="lat")
   ds0.rio.write_crs("epsg:4326",inplace=True)
   dsx=ds0.rio.clip(shp2.geometry.apply(mapping),shp2.crs,drop=False)
   ds2+=[dsx.where(dsx<=0,1).sum().values/dsy.where(dsy<=0,1).sum().values*100]
  dsf[k]=bias_correction(ds2)
 return dsf

def make_plot(df0,z):
 dd=[str(i).split(":")[0].split("-")[-1].replace("_"," - ") for i in df0.index]
 fig,ax=plt.subplots(figsize=(12,5))
 df0['date2']=dd #pd.date_range(start=str(df0.index[0]),periods=len(df0),freq='H') #pd.to_datetime(df0.index,format='%Y-%m-
%d_%H')
 df0=df0.set_index("date2")
#df0.index.astype(str)
 df0.plot(ax=ax)
#ax.xaxis.set_major_formatter(mdates.DateFormatter('%d - %H'))
 ax.set_xlabel("")
#print([df0.index[0],df0.index[-1]])
 ax.set_xticks(range(len(dd)))
 ax.set_xlim([0,len(dd)])
 ax.set_ylabel(f"Tránsito vehicular {z.replace('_',' ')} (%)")
#ax.xaxis.set_major_locator(mdates.HourLocator(interval=10))
#ax.tick_params(axis='x', labelrotation=90)
 ax.set_xticklabels(dd,rotation=90,ha='center') #ax.get_xticklabels()
#print(ax.get_xticklabels())
# ax.set_title(f'{z}')
 bb=30
 if z=="muy_lento":
  bb=10
 ax.set_ylim([-0.05,bb])
 plt.xticks(rotation=90,ha='center',size=7,weight = 'bold')
 plt.savefig(f"{pathd0}/TV_{z}.jpg",dpi=120,bbox_inches='tight')
 plt.close(fig)
```

```
imgs=glob.glob(f'{pathd2}/Traf*.png')
imgs.sort()
print(imgs)
#print(imgs)
dff={}
for i in imgs:
 id=re.split(" |o_",i)[-2]
 dff[id]=calc_perc(i)

zz=list(dff[list(dff.keys())[0]].keys())#[0]
columns=['rapido','moderado','lento','muy_lento']
dfl=[]
for z in zz:
 df=pd.DataFrame(columns=columns)
 for i in dff.keys():
  df.loc[i]=dff[i][z]
 df=df.round(2).reset_index().rename(columns={'index':'date'})
 df['day']=df[['date']].applymap(lambda x:x.split("_")[-1])
 df['date']=df[['date']].applymap(lambda x:x.split("_")[0])
 df.day=pd.to_numeric(df.day)
 df['day']=df[['day']].applymap(lambda x:f"{x:02d}")
 df['date']=df.date+"_"+df.day
 del df['day']
 df=df.sort_values(by="date")
 df.date==pd.to_datetime(df.date,format='%Y-%m-%d_%H')
 df=df.set_index("date")
 dfl+=[df]
for i in columns[-2:]:
 dfs=[j[i] for j in dfl]
 aa=pd.concat(dfs,1)
 aa.columns=zz
 make_plot(aa,i)
```

Figura N° 6. Tránsito Vehicular



2.1.5. Avisos meteorológicos

A continuación se presentan unas líneas de código en pyhton del script “Avisos_meteorologicos.py”que capturan imágenes de los avisos meteorológicos vigentes para el AMLC de la página web del SENAMHI (https://www.senamhi.gob.pe/?&p=aviso-meteorologico).

```
import warnings,os,pickle,glob,time,datetime,traceback
warnings.filterwarnings("ignore")
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver import Remote
from PIL import Image

pathd=f"~/Escritorio/SEA/SERVIDOR/PANTERA/scripts/"
paths=f"~/Escritorio/SEA/SERVIDOR/PANTERA/scripts/"
#xvig="//td[@class='vigente' and @tabindex='0']/a"
#xvig="//a[contains(text(),'(vigente)')]"
xvig="//a[contains(text(),'(vigente)') or contains(text(),'(emitido)')]"
take1="//h2[@class='desaparecerHR']/span[contains(@class,'tres') or contains(@class,'dos')]"
xpanel=".ico_arrow_circulo"
xframe="//iframe[contains(@id,'#v-pills-tab-1-')]"
url="https://www.senamhi.gob.pe/?&p=aviso-meteorologico"

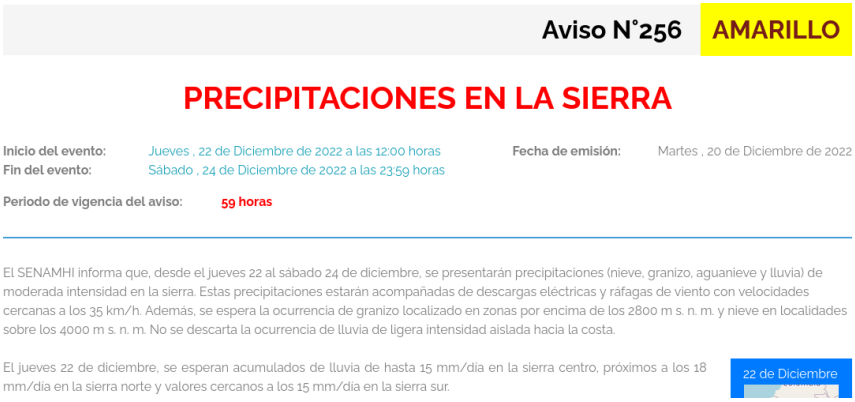
def crop_map(files,k,left,top,right,bottom):
 for i in files:
  im = Image.open(i)
  if k>1:
   bottom=im.size[1]
  im = im.crop((left, top, right, bottom))
  im.save(i)

def get_warns0(browser,k):
 browser.switch_to.default_content()
 aa=browser.find_elements_by_xpath(take1)[-1]
 browser.execute_script("arguments[0].scrollIntoView();",aa)
 browser.save_screenshot(f"warn{k}_1.png")
 for i in range(10):
  browser.find_element_by_tag_name('body').send_keys(Keys.DOWN)
 time.sleep(5)
 idx=browser.find_elements_by_xpath(xframe)
 print("frame")
 print(idx)
 idx=idx[-1]
 browser.switch_to_frame(idx)
 browser.find_element(By.CSS_SELECTOR,xpanel).click()
 time.sleep(1.5)
 browser.save_screenshot(f"warn{k}_2.png")

files = glob.glob(f"{pathd}/warn*.png")
for i in files:
 os.remove(i)
option=webdriver.FirefoxOptions()
option.add_argument('--headless')
option.add_argument('--private')
try:
 #browser=webdriver.Firefox(executable_path=f'{paths}/driver_firefox',
                           #firefox_binary=f'{paths}/firefox/firefox',
                           #options=option,)
 browser = webdriver.Firefox ()
 browser.get(url)
 browser.maximize_window()
 time.sleep(7)
 lala=browser.find_elements_by_xpath(xvig)
 print(lala)
 links=[i.get_attribute("href") for i in lala]
 k=0
 for j in links:
  k+=1
  browser.get(j)
  time.sleep(5)
  get_warns0(browser,k)
 browser.close()
 browser.quit()
```

```
domain=[]
domain+=[[125,0,1225,500]] #[[170,0,1255,500]]
domain+=[[330,0,910,1400]]
for i in [1,2]:
 files = glob.glob(f"{pathd}/warn*_{i}.png")
 crop_map(files,i,*domain[i-1])
except Exception:
  traceback.print_exc()
  pass
print("finished")
```

Figura N° 7. Avisos meorológicos vigentes



2.1.6. Creación de la presentación del briefing ambienal atmosférico

Para la creación de la presentación del briefing ambiental atmosférico se ejecuta el código en bash denominado Briefing_ppt.sh, el cual ejecuta un script en Python llamado "**main_script.py"**, los cuales se muestrán a continuación:

Código en bash:

```
#!/bin/bash
date
kill -9 `ps aux |grep driver_ |grep firefox | awk '{ print $2 }'`
kill -9 `ps aux |grep firefox | awk '{ print $2 }'`
source ~/anaconda3/etc/profile.d/conda.sh
cd ~/briefing
~/scripts/bash/get_fig_ftp.sh &&
conda activate py37
python ~/scripts/python/main_script.py
aa=`LC_ALL=es_ES.utf8 date +%d' DE '%^B`
cc=`LC_ALL=es_ES.utf8 date +%d' DE '%^B -d "-1 day"`
bb="\"BRIEFING SEA ${aa}.pptx\""
dd="\"BRIEFING SEA ${cc}.pptx\""
echo /home/emedina/briefing/$bb
ftp -p ftp.senamhi.gob.pe <<EOF
lcd /home/emedina/briefing
cd SEA/briefing
del ${dd}
put ${bb}
EOF
echo "done!"
```

Código en Python "main_script.py":

El script en Python llamado "**main_script.py"** importa en una plantilla de PowertPoint las imágenes de los avisos meteorológicos, imágenes satelitales de la cobertura nubosa, imágenes de la temperatura superficial del mar, imágenes de las salidas del

WRF (temperatura, humedad relativa, líneas de corriente y de la altura de la capa límete planetaria), imágenes del tránsito vehicular, imágenes de calidad del aire (promedio horario, promedio movil de 24 horas y estados de calidad del aire), imágenes del WRF-CHEM. Asimismo, actualiza la fecha de la diapositiva.

```
import warnings,os,pickle,glob,time,traceback
warnings.filterwarnings("ignore")
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.action_chains import ActionChains
from selenium.webdriver import Remote
import xarray as xr
import pandas as pd
import numpy as np
import cartopy.crs as ccrs
import cartopy
import pylab as plt
#import matplotlib.pyplot as plt
from cartopy.mpl.gridliner import LONGITUDE_FORMATTER, LATITUDE_FORMATTER
import matplotlib.ticker as mticker
import matplotlib
from matplotlib.axes import Axes
import metpy.calc as mpcalc
from cartopy.io.shapereader import Reader
from cartopy.feature import ShapelyFeature
from pptx import Presentation
from pptx.util import Inches
import datetime
import wget
from PIL import Image
import matplotlib.colors as colors
from scipy.interpolate import interp2d
#import win32com.client
import locale
locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')
def l360(lon1):
 return (360 + (lon1 % 360)) % 360

def crop_goes(files):
 for i in files:
  im = Image.open(i)
  left,top,right,bottom=500,0,im.size[0],im.size[1]-50
  im=im.crop((left, top, right, bottom))
  im.save(i)

def crop_sst():
 im = Image.open("sst2.png")
 left,top,right,bottom=40,20,im.size[0]-60,im.size[1]-30
 im=im.crop((left, top, right, bottom))
 im.save("sst2.png")

def put_imgs(slide,imgs,j):
 cc = 10/100; cc2 = 8/100; cc3 = 94/100
 if j is not False:
  imgs=[i for i in imgs if j in i]
 slide.shapes.add_picture(imgs[0], Inches(0.1), Inches(0.1),width=Inches(3.2))
 slide.shapes.add_picture(imgs[1], Inches(0.1), Inches(3.8),width=Inches(3.2))
 pic=slide.shapes.add_picture(imgs[2], Inches(3.4), Inches(0.4),width=Inches(5.5))
 aa=pic.height.inches;bb=pic.width.inches
 pic = pic._element
 pic.getparent().remove(pic)
 pic=slide.shapes.add_picture(imgs[2], Inches(3.4), Inches(0.6),width=Inches(4.5))
 aa2=pic.height.inches;bb2=pic.width.inches
 pic = pic._element
 pic.getparent().remove(pic)
```

```
 pic=slide.shapes.add_picture(imgs[2], Inches(3.6), Inches(0.6),width=Inches(4.7),height=Inches(aa2)*(1-cc3))
 pic.crop_bottom = cc3
 pic=slide.shapes.add_picture(imgs[2], Inches(3.4), Inches(1.),width=Inches(5.5),height=Inches(aa)*(1-cc2))
 pic.crop_top = cc2
 pic=slide.shapes.add_picture(imgs[3], Inches(8.4), Inches(0.6),width=Inches(4.7),height=Inches(aa2)*(1-cc3))
 pic.crop_bottom = cc3
 pic=slide.shapes.add_picture(imgs[3], Inches(8.4), Inches(1.),width=Inches(5.5)*(1-cc),height=Inches(aa)*(1-cc2))
 pic.crop_top = cc2
 pic.crop_left = cc #crop_left #/ pic.width.inches

def move_slide(slides, slide, new_idx):
 slides._sldIdLst.insert(new_idx, slides._sldIdLst[slides.index(slide)])

def new_slide(pst):
 pst.slides.add_slide(pst.slide_layouts[0])
 slide = pst.slides[-1]
 shapes=slide.shapes
 for shape in shapes:
  try:
#   print(shape.shape_type)
   shapes.element.remove(shape.element)
  except:
   sp = shape.element
   sp.getparent().remove(sp)
#  for placeholder in shapes.placeholders:
#   sp = placeholder._sp
#   sp.getparent().remove(sp)
 return slide

pathd=f"/home/emedina/briefing"
#cmd="gdown --folder 1QUbq2r2OyBYF8VC4zqsPIDmJRLj8ItmJ |grep .csv>out.log"
#os.system(cmd)
#links=pd.read_csv(f"{pathd}/briefing/out.log",header=None,sep=" ").loc[:,2].tolist()
#for i in links:
# gdown.download(id=i)

files = glob.glob(f"{pathd}/sst*.png")
for i in files:
 os.remove(i)
now=datetime.datetime.now()
url="http://www.imarpe.gob.pe/ftp/enso/imagenes/rw_s3_atsm_CALLAO.png"
file_name = wget.download(url,"sst2.png")
crop_sst()
dd=0; xx=0
locale.setlocale(locale.LC_TIME, 'en_US.UTF-8')
while xx==0:
 dd+=1
 try:
  yes=now-datetime.timedelta(days=dd)
  yes=yes.strftime("%d%b%Y").upper()
  print(yes)
  url=f"https://www.senamhi.gob.pe/usr/dms/dato_tsm/diario/tsm_a_peru/{yes}_ostia_tsm_a_peru.png"
  file_name = wget.download(url,"sst1.png")
  xx=1
 except Exception:
   traceback.print_exc()
   pass

locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')
files = glob.glob(f"{pathd}/cloud*.jpg")
for i in files:
 os.remove(i)
now=datetime.datetime.now()
yes=now-datetime.timedelta(days=1)
k=0
for i in [yes,now]:
 for j in [12,17]:
  try:
   k+=1
#   print(i.strftime("%Y%m%d"))
   site_url = f'https://www.senamhi.gob.pe/usr/sat/G16D/Lim/LimC02CP{i.strftime("%Y%m%d")}{j}00.jpg'
```

```
#   print(site_url)
   file_name = wget.download(site_url,f"cloud{k}.jpg")
  except:
   try:
    site_url = f'https://www.senamhi.gob.pe/usr/sat/G16D/Lim/LimC02CP{i.strftime("%Y%m%d")}{j-1}50.jpg'
    file_name = wget.download(site_url,f"cloud{k}.jpg")
   except:
    try:
     site_url = f'https://www.senamhi.gob.pe/usr/sat/G16D/Lim/LimC02CP{i.strftime("%Y%m%d")}{j-1}40.jpg'
     file_name = wget.download(site_url,f"cloud{k}.jpg")
    except:
     try:
      site_url = f'https://www.senamhi.gob.pe/usr/sat/G16D/Lim/LimC02CP{i.strftime("%Y%m%d")}{j-1}30.jpg'
      file_name = wget.download(site_url,f"cloud{k}.jpg")
     except:
      try:
       site_url = f'https://www.senamhi.gob.pe/usr/sat/G16D/Lim/LimC02CP{i.strftime("%Y%m%d")}{j-2}00.jpg'
       file_name = wget.download(site_url,f"cloud{k}.jpg")
      except Exception:
       traceback.print_exc()
       print(site_url)
       pass

files = glob.glob(f"{pathd}/cloud*.jpg")
crop_goes(files)

pst = Presentation(f'{pathd}/briefing_template.pptx')
tom=datetime.datetime.now()+datetime.timedelta(days=1)
try:
 slide = pst.slides[-2]
 shapes = slide.shapes
 paragraph = shapes[0].text_frame.paragraphs[0]
 paragraph.runs[2].text=f'PARA EL {tom.strftime("%#d DE %B").upper()}'
 paragraph.runs[3].text=f"
 paragraph.runs[4].text=f"
 slide.shapes.add_picture('prom_pm2.5_lima.2.png',Inches(6),Inches(0.1),width=Inches(7))
except Exception:
  traceback.print_exc()
  pass
##################################
try:
 files = glob.glob(f"{pathd}/*PM25*.jpg")
 files.sort()
 print(files)
 slides=pst.slides
 for i,j in zip(files,range(-6,-2)):
  if j<-3:
   slides[j].shapes.add_picture(i, Inches(0.6), Inches(1),width=Inches(12))
  else:
   slides[j].shapes.add_picture(i, Inches(0.2), Inches(1),width=Inches(13))
except Exception:
  traceback.print_exc()
  pass
#####################################
slide = pst.slides[0]
shapes = slide.shapes
#for shape in shapes:
#    print(shape.shape_type)
paragraph = shapes[5].text_frame.paragraphs[0]
#print(paragraph.text)
paragraph.runs[0].text=f'PRONÓSTICO DE LA CALIDAD DEL AIRE PARA EL '
#tom=datetime.datetime.now()+datetime.timedelta(days=1)
paragraph.runs[1].text=f'{tom.strftime("%#d DE %B").upper()}'
paragraph.runs[2].text=f"
for i in [1,3]:
 try:
  slide=new_slide(pst)
  left_img = f'cloud{i}.jpg'
  right_img = f'cloud{i+1}.jpg'
  top_pic = slide.shapes.add_picture(left_img, Inches(0.5), Inches(0.7),width=Inches(6))
  bottom_pic = slide.shapes.add_picture(right_img, Inches(6.9), Inches(0.7),width=Inches(6))
```

```
 except Exception:
   traceback.print_exc()
   pass
slide=new_slide(pst)
left_img = f'sst1.png'
right_img = f'sst2.png'
top_pic = slide.shapes.add_picture(left_img, Inches(0), Inches(0.7),width=Inches(8))
bottom_pic = slide.shapes.add_picture(right_img, Inches(6.2), Inches(1.9),width=Inches(7))
files = glob.glob(f"{pathd}/warn*.png")
try:
 for i in range(int(len(files)/2)):
  slide=new_slide(pst)
  left_img = f'warn{i+1}_1.png'
  right_img = f'warn{i+1}_2.png'
  top_pic = slide.shapes.add_picture(left_img, Inches(0.1), Inches(0.6),width=Inches(8.1))
  bottom_pic = slide.shapes.add_picture(right_img, Inches(8.3), Inches(0.6),width=Inches(5))
except Exception:
   traceback.print_exc()
   pass
slides=pst.slides
n=len(slides)
slide=slides[1]
move_slide(slides,slide,n)

locale.setlocale(locale.LC_TIME, 'en_US.UTF-8')
now=datetime.datetime.now()
past=now-datetime.timedelta(days=2)
for k in ["RelHum","Streamlines","Temperature","PBL"]:
# imgs=[]
 try:
  for j in ["07","13","19"]:
   imgs=[]
   for i in range(3):
    dtx0=past+datetime.timedelta(days=i)
    dtx=dtx0.strftime("%Y%m%d")
    dtx2=dtx0.strftime("%d%b%Y").upper()
    imgs+=[f'{pathd}/FIG_WRF1km/{dtx}/{k}_{dtx2}_{j}hr.png']
   dtx0=past+datetime.timedelta(days=3)
   dtx2=dtx0.strftime("%d%b%Y").upper()
   imgs+=[f'{pathd}/FIG_WRF1km/{dtx}/{k}_{dtx2}_{j}hr.png']
   slide=new_slide(pst)
   put_imgs(slide,imgs,j)
   print(imgs)
  imgs=[]
  for i in range(3):
    dtx0=past+datetime.timedelta(days=i)
    dtx=dtx0.strftime("%Y%m%d")
    dtx2=dtx0.strftime("%d%b%Y").upper()
    imgs+=[f'{pathd}/FIG_WRF1km/{dtx}/prom{k}_{dtx2}.png']
  dtx0=past+datetime.timedelta(days=3)
  dtx2=dtx0.strftime("%d%b%Y").upper()
  imgs+=[f'{pathd}/FIG_WRF1km/{dtx}/prom{k}_{dtx2}.png']
  slide=new_slide(pst)
  put_imgs(slide,imgs,False)
 except Exception:
   traceback.print_exc()
   pass
slides=pst.slides
n=len(slides)
slide=slides[1]
move_slide(slides,slide,n)

locale.setlocale(locale.LC_TIME, 'es_ES.UTF-8')
now=datetime.datetime.now()
yes=datetime.datetime.now()-datetime.timedelta(days=1)
imgs = glob.glob(f"{pathd}/TV_*.jpg")
for i in range(2):
 try:
  slides=pst.slides
  slide=slides[1]
  slide.shapes.add_picture(imgs[i], Inches(.8), Inches(1.5),width=Inches(12))
```

```
#  move_slide(slides,slide,n)
 except Exception:
#   move_slide(slides,slide,n)
   traceback.print_exc()
   pass
 move_slide(slides,slide,n)
for i in range(7):
 slides=pst.slides
 n=len(slides)
 slide=slides[1]
 move_slide(slides,slide,n)

tt=f'{pathd}/BRIEFING SEA {now.strftime("%#d DE %B").upper()}.pptx'
#tt=f'{pathd}/BRIEFING_SEA_{now.strftime("%#d_DE_%B").upper()}.pptx'
pst.save(tt)
print(tt)
print("finished")
```

2.2. Vigilancia de la calidad del aire

2.2.1. Post de Calidad del Aire

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
library(ggplot2)
library(openair)
library(latticeExtra)
library(directlabels)
library(dplyr)
library(scales)
library(readxl)
library(egg)
library(gtable)
library(gridExtra)
library(grid)
library(readxl)
library(readr)
library(googledrive)
library(data.table)
```

Luego se realiza la descarga la información de los contaminates del aire para la elaboración de las gráficas. La siguiente línea de código selecciona la carpeta de trabajo, descarga un archivo exel.csv del googledrive y formatea la fecha. El script identifica los valores menores a 0 y se les asigna la etiqueta de NA. Asimismo, el script crea un Dataframe llamado DATA de los últimos 27 días:

```
id="1QLn-aHHDzXapJqbPL4Bce4hBK3MyxbFK"
POST_DATA = read_delim(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
                       col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

cambio_na = function(x){ifelse(x<0,NA,x)}

df_cambio = data.frame(sapply(POST_DATA,cambio_na))

df_cambio$date = POST_DATA$date

POST_DATA = df_cambio

DATA<-timeAverage(POST_DATA, avg.time="day", statistic="mean",data.thresh = 75)
as.Date(DATA$date)

DATA<-mutate(DATA, Buena=12, Moderada=35, InsalubreGS=55,Insalubre=70)
n<-nrow(DATA)
```

```
x<-n-27
DATA<-DATA[x:n-1,]
```

Las siguientes líneas de código elaboran una gráfica por cada estación de calidad del aire para el $PM_{2.5}$:

```
VMT_POST <- ggplot(data = DATA,aes(x=date))+
  geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=VMT,colour="VMT"),linewidth=0.5)+
  geom_point(aes(y=VMT),color="blue",size=1)+geom_hline(yintercept = 50,linewidth=0.5,color="red")+
  xlab("días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,290,10),expand = c(0,0))+
  scale_colour_manual("",values=c("VMT"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'VILLA MARIA DEL TRIUNFO'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))+
  theme(plot.caption = element_text(size = rel(0.8),vjust=-10,hjust = -10,face = "bold",color = "black"))
VMT_POST
```

```
SJL_POST <- ggplot(data = DATA,aes(x=date))+
  geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=SJL,colour="SJL"),size=0.5)+
  geom_point(aes(y=SJL),color="blue",size=1)+geom_hline(yintercept = 50,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,80,10),expand = c(0,0))+
  scale_colour_manual("",values=c("SJL"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'SAN JUAN DE LURIGANCHO'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
SJL_POST
```

```
STA_POST <- ggplot(data = DATA,aes(x=date))+
  geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=STA,colour="STA"),size=0.5)+
  geom_point(aes(y=STA),color="blue",size=1)+geom_hline(yintercept = 50,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,80,10),expand = c(0,0))+
  scale_colour_manual("",values=c("STA"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'SANTA ANITA'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
STA_POST
```

```
SBJ_POST <- ggplot(data = DATA,aes(x=date))+
  geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
```

```
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=SBJ,colour="SBJ"),size=0.5)+
  geom_point(aes(y=SBJ),color="blue",size=1)+geom_hline(yintercept = 50,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,80,10),expand = c(0,0))+
  scale_colour_manual("",values=c("SBJ"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
        axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACÓN 'SAN BORJA'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
SBJ_POST
```

```
SMP_POST <- ggplot(data = DATA,aes(x=date))+
  geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=SMP,colour="SMP"),size=0.5)+
  geom_point(aes(y=SMP),color="blue",size=1)+geom_hline(yintercept = 50,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0,0))+
  scale_y_continuous(breaks = seq(0,80,10),expand = c(0,0))+
  scale_colour_manual("",values=c("SMP"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
        axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'SAN MARTÍN DE PORRES'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
SMP_POST
```

```
CRB_POST <- ggplot(data = DATA,aes(x=date))+
  geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=CRB,colour="CRB"),size=0.5)+
  geom_point(aes(y=CRB),color="blue",size=1)+geom_hline(yintercept = 50,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0,0))+
  scale_y_continuous(breaks = seq(0,80,10),expand = c(0,0))+
  scale_colour_manual("",values=c("CRB"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
        axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'CARABAYLLO'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
CRB_POST
```

```
PPD_POST <- ggplot(data = DATA,aes(x=date))+
  geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=PPD,colour="PPD"),size=0.5)+
  geom_point(aes(y=PPD),color="blue",size=1)+
  geom_hline(yintercept = 50,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0,0))+
  scale_y_continuous(breaks = seq(0,80,10),expand = c(0,0))+
  scale_colour_manual("",values=c("PPD"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
```

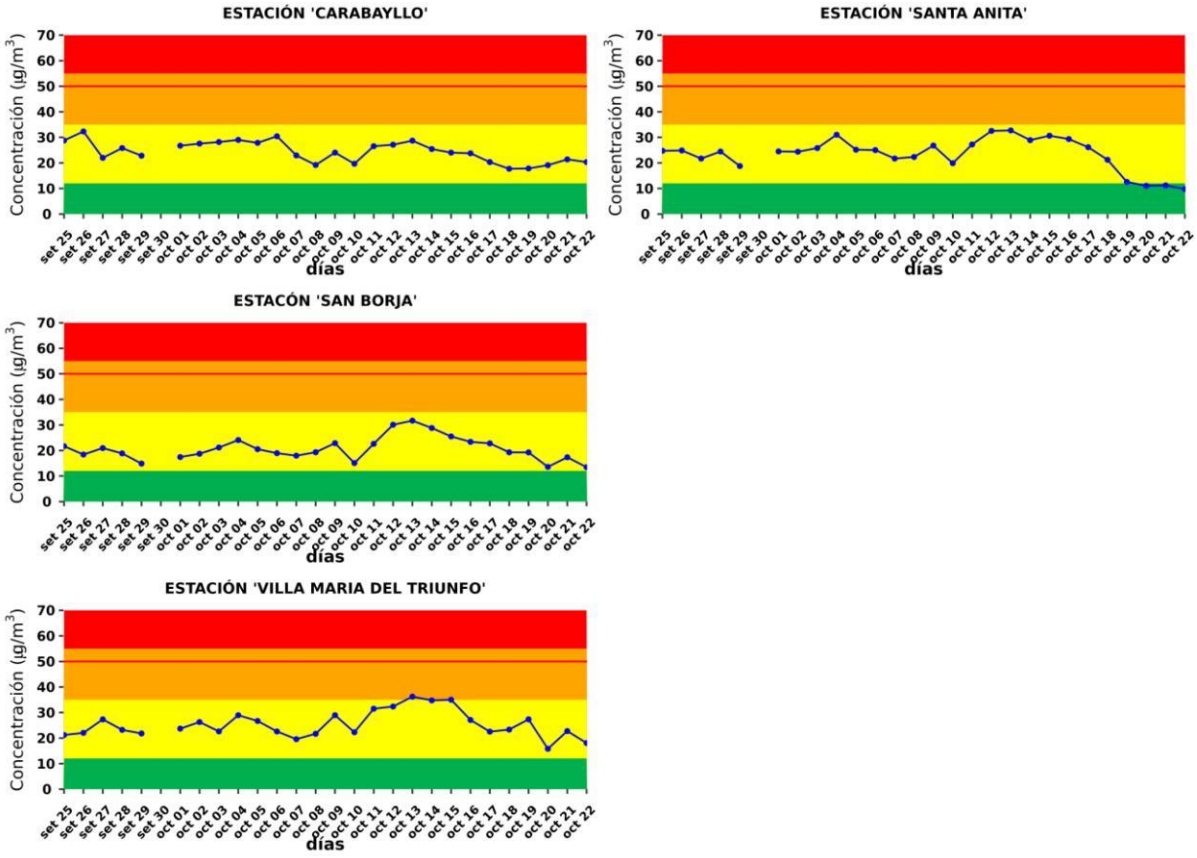
```
      axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'PUENTE PIEDRA'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
PPD_POST
```

Las siguientes líneas de código elaboran un panelPlot y la guarda con el nombre ESTADOS_POST_25.jpg.

```
ESTADOS<-grid.arrange(CRB_POST,SBJ_POST, VMT_POST,STA_POST, layout_matrix = rbind(c(1, 4),
                                                                                   c(2, NA),
                                                                                   c(3,NA)))
ggsave(filename = "ESTADOS_POST_25.jpg", plot =ESTADOS, width = 25.0, height = 18, dpi = 1000, units = "cm")

write.csv(DATA, "DATOS_PM25.csv")
```

Figura N° 8. Post de Calidad del Aire $PM_{2.5}$



Las siguientes líneas de código elaboran una gŕafica por cada estación de calidad del aire para el PM_{10}:

```
id="10YFfnPnW2yfL-_jPg8DIF-UxjOBQ4xS0"
POST_DATA_PM10 = read_delim(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
                    col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

#cambio_na = function(x){ifelse(x<0,NA,x)}

df_cambio = data.frame(sapply(POST_DATA_PM10,cambio_na))

df_cambio$date = POST_DATA_PM10$date

POST_DATA_PM10 = df_cambio

DATA<-timeAverage(POST_DATA_PM10, avg.time="day", statistic="mean",data.thresh = 75)
as.Date(DATA$date)

DATA<-mutate(DATA, Buena=54, Moderada=154, InsalubreGS=180,Insalubre=290)
n<-nrow(DATA)
x<-n-27
DATA<-DATA[x:n-1,]
```

```
VMT_POST <- ggplot(data = DATA,aes(x=date))+
  #geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=VMT,colour="VMT"),size=0.5)+
  geom_point(aes(y=VMT),color="blue",size=1)+geom_hline(yintercept = 100,size=0.5,color="red")+
  xlab("días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,210,30),expand = c(0,0))+
  scale_colour_manual("",values=c("VMT"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'VILLA MARIA DEL TRIUNFO'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))+
  theme(plot.caption = element_text(size = rel(0.8),vjust=-10,hjust = -10,face = "bold",color = "black"))
VMT_POST
```

```
SJL_POST <- ggplot(data = DATA,aes(x=date))+
  #geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=SJL,colour="SJL"),size=0.5)+
  geom_point(aes(y=SJL),color="blue",size=1)+geom_hline(yintercept = 100,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,210,30),expand = c(0,0))+
  scale_colour_manual("",values=c("SJL"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'SAN JUAN DE LURIGANCHO'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
SJL_POST
```

```
STA_POST <- ggplot(data = DATA,aes(x=date))+
  #geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=STA,colour="STA"),size=0.5)+
  geom_point(aes(y=STA),color="blue",size=1)+geom_hline(yintercept = 100,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,210,30),expand = c(0,0))+
  scale_colour_manual("",values=c("STA"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'SANTA ANITA'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
STA_POST
```

```
SBJ_POST <- ggplot(data = DATA,aes(x=date))+
  #geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=SBJ,colour="SBJ"),size=0.5)+
  geom_point(aes(y=SBJ),color="blue",size=1)+geom_hline(yintercept = 100,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0, 0))+
  scale_y_continuous(breaks = seq(0,210,30),expand = c(0,0))+
  scale_colour_manual("",values=c("SBJ"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'SAN BORJA'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
SBJ_POST
```

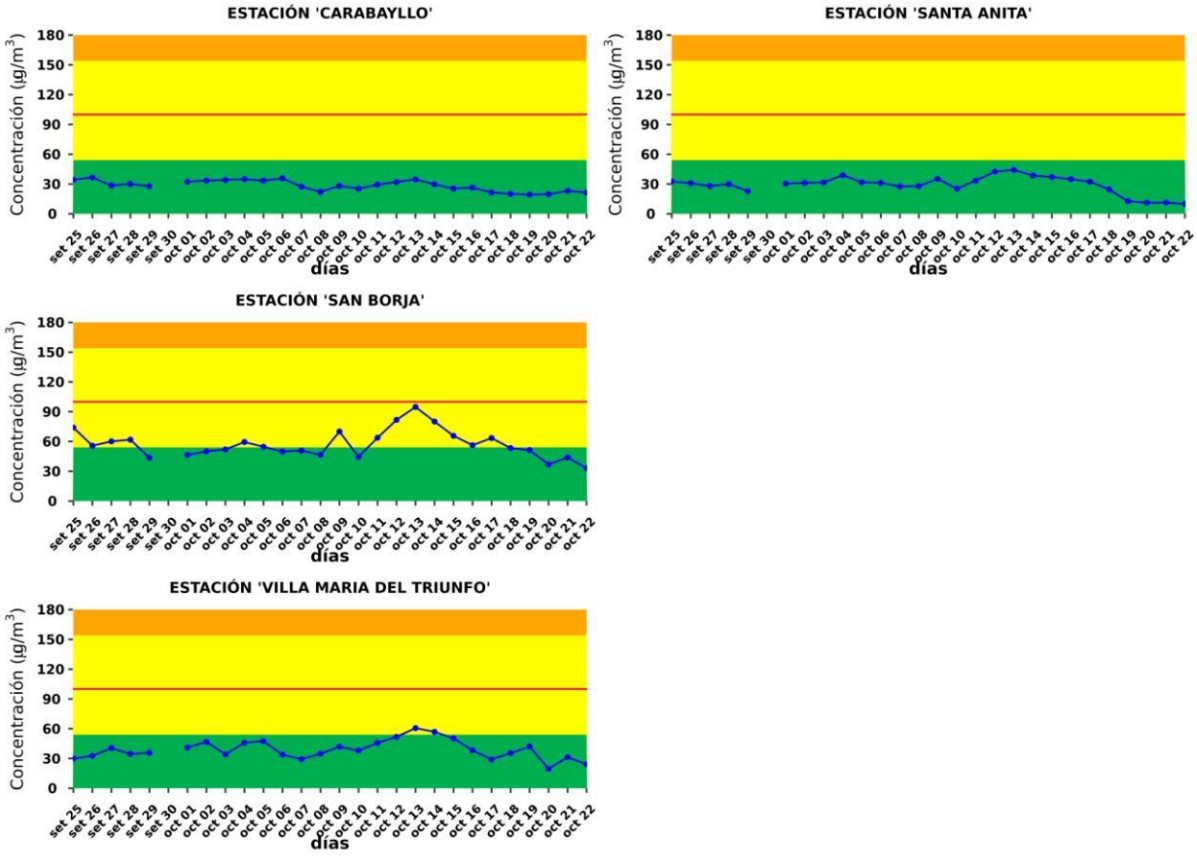
```
CRB_POST <- ggplot(data = DATA,aes(x=date))+
  #geom_area(aes(y=Insalubre),fill="red")+
  geom_area(aes(y=InsalubreGS),fill="orange")+
  geom_area(aes(y=Moderada),fill="yellow")+
  geom_area(aes(y=Buena),fill="#00b050")+
  geom_line(aes(y=CRB,colour="CRB"),size=0.5)+
  geom_point(aes(y=CRB),color="blue",size=1)+geom_hline(yintercept = 100,size=0.5,color="red")+
  labs(x="días")+ylab(expression(Concentración* " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_datetime(date_breaks = "1 day", labels = date_format("%b %d"),expand = c(0,0))+
  scale_y_continuous(breaks = seq(0,210,30),expand = c(0,0))+
  scale_colour_manual("",values=c("CRB"="blue"))+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8),hjust = 0.8,angle = 45,vjust = 0.6),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.9)))+
  theme(legend.position = "none")+ggtitle("ESTACIÓN 'CARABAYLLO'")+
  theme(plot.title = element_text(size = rel(0.8),vjust=2,hjust=0.5,face="bold",color = "black"))
CRB_POST
```

Las siguientes líneas de código elaboran un panelPlot y la guarda con el nombre **ESTADOS_POST_10.jpg.**

```
ESTADOS_PM10<-grid.arrange(CRB_POST,SBJ_POST, VMT_POST,STA_POST, layout_matrix = rbind(c(1, 4),
                                                                                       c(2, NA),
                                                                                       c(3,NA)))

ggsave(filename = "ESTADOS_POST_PM10.jpg", plot =ESTADOS_PM10, width = 25.0, height = 18, dpi = 1000, units = "cm
")
```

Figura N° 9. Post de Calidad del Aire PM_{10}



2.2.2. Gráfica de variables meteorológicas y contaminantes del aire

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
rm (list = ls())
library(openair)
library(ggplot2)
library(tidyverse)
library(magrittr)
library(reshape2)
library(directlabels)
library(latticeExtra)
library(gridExtra)
library(grid)
```

Luego se realiza la descarga de la información de los contaminantes del aire y variables meteorologicas, a través del siguiente código.

```
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO")

# lectura de datos

ids = c("10YFfnPnW2yfL-_jPg8DIF-UxjOBQ4xS0",
        "1QLn-aHHDzXapJqbPL4Bce4hBK3MyxbFK",
        "179wE4h1h_C0PMjJFRwHpAMOFluYRuVSR",
        "1DaoSkSJb9vM9rTEpoRlgL14Y-Xr_7cQl",
        "1tEgyepcu9esZ_JlIxUzd3kvN54jh9KVU")

open <- function(id){

  tabla = read_delim(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
                     col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

  tabla = tabla %>% openair::selectByDate(.,start = as.character(Sys.Date()-4),end = as.character(Sys.Date()-1))
  tabla = tabla %>% openair::timeAverage(.,avg.time = "day",statistic = "mean",data.thresh = 75)
}

multiple_csv <- sapply(ids, open)

estaciones = c("SBJ","VMT","STA","CRB")

# Creacion de tablas vacias

tablas = list()

gen = function() {
  df <- data.frame(matrix(ncol = 5, nrow = nrow(multiple_csv[[1]])))
  #proporcionar nombres de columna
  colnames (df) <- c ('PM1O','PM25', 'HUMR', 'TEMP', 'VV')
  return(df)
}

for (k in 1:length(estaciones)) {
  tablas[[k]] = gen()
}

# Creacion de tablas de variables por estacion

for (i in 1:length(estaciones)) {

  for (j in 1:length(multiple_csv)) {
    var = multiple_csv[[j]] %>% select(.,estaciones[i])
    tablas[[i]][j] = var
  }

}
```

Finalmente se elabora la gráfica y se guarda en formato .jpg con el nombre **Post_ayuda.jpg.**

```
#Creación de gráfica

fechas = multiple_csv[[1]][1] %>% pull(.)

variable_names <- c("PM10"="PM10 (ug/m3)",
                    "PM2.5"="PM2.5 (ug/m3)",
                    "HUMR" = "%" ,"TEMP"="°C",
                    "VV" = "m/s")


variable_labeller <- function(variable,value){
  return(variable_names[value])
}
```

```
graficar = function(n,nombre){

  datos = tablas[[n]] %>% mutate(date = fechas) %>% melt(.,id.vars="date")
  g1 = ggplot(datos,aes(x = date,y = value,fill=variable)) +
    geom_bar(stat = "identity",position = "dodge")+
    facet_grid(variable ~., scales = 'free',switch = "y",
               labeller = as_labeller(variable_labeller)
    ) +
    scale_x_datetime(date_breaks = "1 day",date_labels = "%d - %m") +
    guides(color = guide_legend(title = "Variable",title.theme = element_text(face = "bold",hjust = 0.5)))+
    xlab("") +
    ylab("") +
    geom_text(aes(y = value, #ymax = value,
                  label = round(value,1)),size=4,vjust=2, hjust=0.5,col="white",fontface="bold")+
    ggtitle(paste0("Estación ",nombre)) +
    scale_fill_manual(values = c("brown","red","blue","orange","darkgreen"),labels = c("PM10","PM2.5", "Humedad Relativa","
Temperatura","Velocidad de viento")) +
    theme_bw()+
    theme (axis.text.x = element_text(angle=90,face="bold", colour="black", size=rel(0.9)),
           axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6),
           legend.position = "none")

  return(g1)

}

graficas = list()

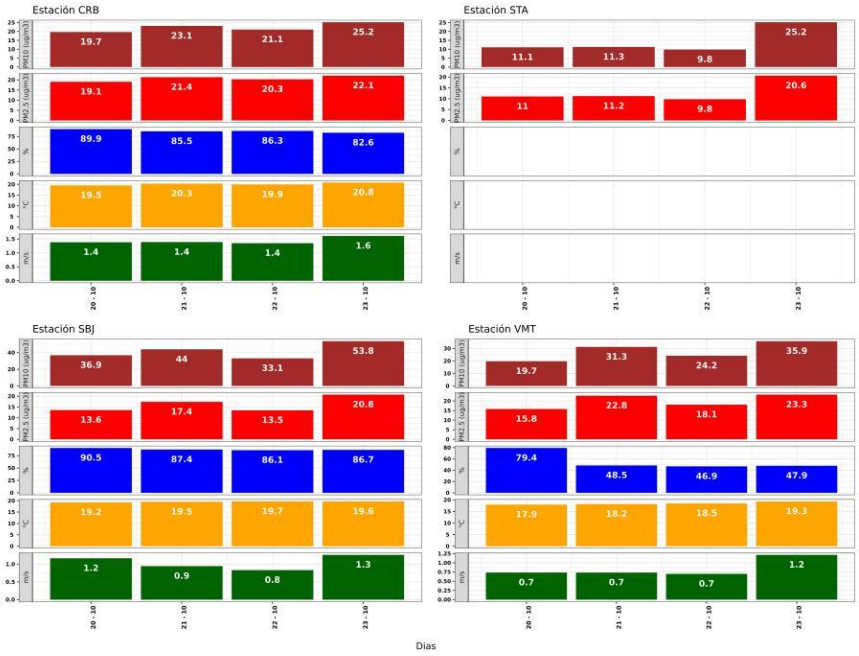
for (i in 1:length(tablas)) {

  graficas[[i]] = graficar(i,estaciones[i])

}


PM25_AML_NE<-grid.arrange(graficas[[which(estaciones=="CRB")]],
                          graficas[[which(estaciones=="STA")]],
                          graficas[[which(estaciones=="SBJ")]],
                          graficas[[which(estaciones=="VMT")]],
                          ncol=2,
                          bottom = textGrob("Dias"))
```

Figura N° 10. Post de ayuda



2.2.3. Dashboard interactivo para la vigilancia de la calidad del aire

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
library(flexdashboard)
library(ggplot2)
library(openair)
library(latticeExtra)
library(directlabels)
library(dplyr)
library(scales)
library(readxl)
library(egg)
library(gtable)
library(gridExtra)
library(grid)
library(plotly)
library(magick)
library(readxl)
library(readr)
library(googledrive)
library(data.table)
library(gganimate)
library(gifski)
library(gapminder)
library(highcharter)
library(gt)
library(knitr)
library(tidyverse)
library(DT)
library(kableExtra)
library(leaflet)
```

Luego se realiza la descarga de la información de los contaminantes del aire, a través del siguiente código.

```
#id ="12Zwp805UkzrzFjGhPmWMUaK4q0b-NjmG"
id ="1QLn-aHHDzXapJqbPL4Bce4hBK3MyxbFK"
#id ="1YDofVc230I0t4dParlIwqeGgg15ewL08"


POST_DATA = read_delim(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
                       col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

cambio_na = function(x){ifelse(x<0,NA,x)}

df_cambio = data.frame(sapply(POST_DATA,cambio_na))

df_cambio$date = POST_DATA$date

POST_DATA = df_cambio

DATA<-timeAverage(POST_DATA, avg.time="day", statistic="mean",data.thresh = 15)
DATA<-mutate(DATA, Buena=12, Moderada=35, InsalubreGS=55,Insalubre=100)
n<-nrow(DATA)
x<-n-19
DATA<-DATA[x:n-1,]
DATA<-mutate(DATA,num = c(1:20))


catgpm25 <- data.frame(xstart = c(0,12,35,55), xend = c(12,35,55,100),
                       col = letters[1:4])

catgpm10 <- data.frame(xstart = c(0,54,154,254), xend = c(54,154,254,290),
```

```
                col = letters[1:4])



s <- Sys.Date()-20
e <- Sys.Date()-1
etiquetas = format(seq(from=s, to=e, by=1),"%d/%m")


id ="10YFfnPnW2yfL-_jPg8DIF-UxjOBQ4xS0"

#id ="1uHG6Jbq68hncWJSHJthM56rYwQj4D_Ht"
POST_DATA_PM10 = read_delim(sprintf("https://docs.google.com/uc?id=%s&export=download", id),
                col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

cambio_na = function(x){ifelse(x<0,NA,x)}

df_cambio = data.frame(sapply(POST_DATA_PM10,cambio_na))

df_cambio$date = POST_DATA_PM10$date

POST_DATA_PM10 = df_cambio

DATA2<-timeAverage(POST_DATA_PM10, avg.time="day", statistic="mean",data.thresh = 15)
DATA2<-mutate(POST_DATA_PM10, Buena=12, Moderada=35, InsalubreGS=55,Insalubre=100)
n<-nrow(DATA2)
x<-n-19
DATA2<-DATA2[x:n-1,]
DATA2<-mutate(DATA2,num = c(1:20))


catgpm25 <- data.frame(xstart = c(0,12,35,55), xend = c(12,35,55,100),
                col = letters[1:4])

catgpm10 <- data.frame(xstart = c(0,54,154,254), xend = c(54,154,254,290),
                col = letters[1:4])

s <- Sys.Date()-20
e <- Sys.Date()-1
etiquetas = format(seq(from=s, to=e, by=1),"%d\n%b")
```

Finalmente se elabora el dasboard con el siguiente código:

```
Column {data-width=350}
-----------------------------------------------------------------------
### CARABAYLLO (CRB)

```{r}
CRB<- ggplot() +
  geom_rect(data=catgpm25, aes(ymin = xstart,
                               ymax = xend,
                               xmin = - Inf,
                               xmax = Inf,
                               fill= col), alpha = 0.8) +
  geom_line(data= DATA, aes(x= num, y = CRB), colour = "blue", size= 2)+
  geom_point(data= DATA,aes(x=num,y=CRB),color="blue",size=8)+
  geom_hline(yintercept = 50,size=0.5,color="red") +
  theme_bw() +
  scale_y_continuous(breaks = seq(10, 90, 20),
                     limits = c(0,100),
                     expand = c(0,0)) +
  scale_x_continuous(breaks = seq(1, nrow(DATA), 1),
                     expand = c(0.01, 0.01),
                     name = "Fecha",
                     labels = etiquetas)+
  xlab("\ndías")+ylab(expression(Concentración * " (" * mu * "g/m" ^ 3 *")"))+
  theme(axis.text.x = element_text(size = 50, color = "black", face = "bold"),
        axis.text.y = element_text(size = 80, color = "black", face = "bold",
```
```

```
                          margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
      panel.grid.major = element_line(colour = "black"),
      panel.grid.minor = element_blank(),
      axis.title.y = element_text(size=80),
      axis.title.x = element_text(size=60),
      title = element_text(size = 24, face = "bold", color = "black"),
      legend.position = "none") +
  scale_fill_manual(values = c("#4eb400", "#f7e400", "#f88700", "#d8001d", "#998cff"))+
  transition_manual(num,cumulative = TRUE)
CRB<-animate(CRB,  height = 1500, width =2500)
CRB
anim_save("CRB_PM25.gif",CRB)

```

SANTA ANITA (STA)

```{r}
STA<- ggplot() +
  geom_rect(data=catgpm25, aes(ymin = xstart,
                           ymax = xend,
                           xmin = - Inf,
                           xmax = Inf,
                           fill= col), alpha = 0.8) +
  geom_line(data= DATA, aes(x= num, y = STA), colour = "blue", size= 2)+
  geom_point(data= DATA,aes(x=num,y=STA),color="blue",size=8)+
  geom_hline(yintercept = 50,size=0.5,color="red") +
  theme_bw() +
  scale_y_continuous(breaks = seq(10, 90, 20),
                     limits = c(0,100),
                     expand = c(0,0)) +
  scale_x_continuous(breaks = seq(1, nrow(DATA), 1),
                     expand = c(0.01, 0.01),
                     name = "Fecha",
                     labels = etiquetas)+
  xlab("\ndías")+ylab(expression(Concentración * " (" * mu * "g/m" ^ 3 *")"))+
  theme(axis.text.x = element_text(size = 50, color = "black", face = "bold"),
        axis.text.y = element_text(size = 80, color = "black", face = "bold",
                                   margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
        panel.grid.major = element_line(colour = "black"),
        panel.grid.minor = element_blank(),
        axis.title.y = element_text(size=80),
        axis.title.x = element_text(size=60),
        title = element_text(size = 24, face = "bold", color = "black"),
        legend.position = "none") +
  scale_fill_manual(values = c("#4eb400", "#f7e400", "#f88700", "#d8001d", "#998cff"))+
  transition_manual(num,cumulative = TRUE)
STA<-animate(STA,  height = 1500, width =2500)
STA

```
Column {data-width=350}

PUENTE PIEDRA (PPD)

```{r}
PPD<- ggplot() +
  geom_rect(data=catgpm25, aes(ymin = xstart,
                           ymax = xend,
                           xmin = - Inf,
                           xmax = Inf,
                           fill= col), alpha = 0.8) +
  geom_line(data= DATA, aes(x= num, y = PPD), colour = "blue", size= 2)+
  geom_point(data= DATA,aes(x=num,y=PPD),color="blue",size=8)+
  geom_hline(yintercept = 50,size=0.5,color="red") +
  theme_bw() +
  scale_y_continuous(breaks = seq(10, 90, 20),
                     limits = c(0,100),
                     expand = c(0,0)) +
  scale_x_continuous(breaks = seq(1, nrow(DATA), 1),
```

```
                expand = c(0.01, 0.01),
                name = "Fecha",
                labels = etiquetas)+
  xlab("\ndías")+ylab(expression(Concentración * " (" * mu * "g/m" ^ 3 *")"))+
  theme(axis.text.x = element_text(size = 50, color = "black", face = "bold"),
      axis.text.y = element_text(size = 80, color = "black", face = "bold",
                        margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
      panel.grid.major = element_line(colour = "black"),
      panel.grid.minor = element_blank(),
      axis.title.y = element_text(size=80),
      axis.title.x = element_text(size=60),
      title = element_text(size = 24, face = "bold", color = "black"),
      legend.position = "none") +
  scale_fill_manual(values = c("#4eb400", "#f7e400", "#f88700", "#d8001d", "#998cff"))+
  transition_manual(num,cumulative = TRUE)
PPD<-animate(PPD,  height = 1500, width =2500)
PPD
```

VILLA MARÍA DEL TRIUNFO

```{r}
VMT<- ggplot() +
  geom_rect(data=catgpm25, aes(ymin = xstart,
                        ymax = xend,
                        xmin = - Inf,
                        xmax = Inf,
                        fill= col), alpha = 0.8) +
  geom_line(data= DATA, aes(x= num, y = VMT), colour = "blue", size= 2)+
  geom_point(data= DATA,aes(x=num,y=VMT),color="blue",size=8)+
  geom_hline(yintercept = 50,size=0.5,color="red") +
  theme_bw() +
  scale_y_continuous(breaks = seq(10, 90, 20),
                limits = c(0,100),
                expand = c(0,0)) +
  scale_x_continuous(breaks = seq(1, nrow(DATA), 1),
                expand = c(0.01, 0.01),
                name = "Fecha",
                labels = etiquetas)+ylab(expression(Concentración * " (" * mu * "g/m" ^ 3 *")"))+
  theme(axis.text.x = element_text(size = 50, color = "black", face = "bold"),
      axis.text.y = element_text(size = 50, color = "black", face = "bold",
                        margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
      panel.grid.major = element_line(colour = "black"),
      panel.grid.minor = element_blank(),
      axis.title.y = element_text(size=80),
      axis.title.x = element_text(size=60),
      title = element_text(size = 24, face = "bold", color = "black"),
      legend.position = "none") +
  scale_fill_manual(values = c("#4eb400", "#f7e400", "#f88700", "#d8001d", "#998cff"))+
  transition_manual(num,cumulative = TRUE)
VMT<-animate(VMT,  height = 1500, width =2500)
VMT

anim_save("VMT_PM25.gif",VMT)
```

Figura N° 11. Dasboard Vigilancia de Calidad del Aire

2.3. Atención de incendios urbanos

2.3.1. Generación de inputs meteorológicos

En la vigilancia de incendios urbanos es necesario la información meteorológica para analizar la dispersión de los contaminantes a travéz de modelos de dispersión de contaminantes atmosféricos. En ese sentido se ha desarrollado un script en bash que utiliza las salidas del modelo WRF en formato (.wrfout) para generar los inputs meteorológicos (.pfl y .sfc) para las corridas con el modelo de dispersión. A continuación se presenta el código:

```
#! /bin/bash
#
# Envio de scrip: ./preparaDATOS.sh -11.912972 -77.074584 20220120 8
#               ./preparaDATOS.sh [Latitud] [longitud] [fecha: YYYYmmdd] [hora local: 0 a 23] [numero de horas pronostico]>& hola.log

cd /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/MMIFv342

if [ $# -eq 5 ]
then

lati=$1   # -11.912972
loni=$2   # -77.074584
ymd=$3    #`date +%Y%m%d`
hr=$4
nhp=${5:-10}

#if [ ${hr} -lt 10 ]
if [ ${hr} -lt 0 ]
then
  hr=0${hr}
  yr=$( date +%Y --date "${ymd} -1 day" )
  mo=$( date +%m --date "${ymd} -1 day" )
  dy=$( date +%d --date "${ymd} -1 day" )
  ymd2=${yr}${mo}${dy}
else
  ymd2=${ymd}
fi

yr=`echo ${ymd} | awk '{print substr($1,1,4)}'`
mo=`echo ${ymd} | awk '{print substr($1,5,2)}'`
dy=`echo ${ymd} | awk '{print substr($1,7,2)}'`

yr2=$( date +%Y --date "${ymd} +1 day" )
mo2=$( date +%m --date "${ymd} +1 day" )
dy2=$( date +%d --date "${ymd} +1 day" )
```

```
ymd3=$(date -d"${hr} ${yr}-${mo}-${dy} +${nhp} hours" +"%Y %m %d %H" )

# Inicio de nameslit

echo "# START and STOP can be either of the forms below, or YYYY-MM-DD_HH:mm:ss."              >  mmif.inp
echo "start      ${yr} ${mo} ${dy} ${hr} ; start time in LST for TimeZone, hour-ending format"   >> mmif.inp
echo "stop       ${ymd3} ; end   time in LST for TimeZone, hour-ending format"                 >> mmif.inp
echo                                                                                          >> mmif.inp
echo TimeZone   -5   ! default is zero, i.e. GMT-00                                           >> mmif.inp
echo grid       IJ -5,-5 -5,-5   ! default                                                    >> mmif.inp
echo                                                                                          >> mmif.inp
echo "layers top 20 40 80 160 320 640 1200 2000 3000 4000   ! FLM CALMET Guidance (2009)"     >> mmif.inp
echo stability  GOLDER      ! default                                                         >> mmif.inp
echo CLOUDCOVER ANGEVINE    ! default                                                         >> mmif.inp
echo CALSCI_MIXHT WRF       ! default                                                         >> mmif.inp
echo aer_mixht     WRF  ! default                                                           >> mmif.inp
echo "aer_min_mixht 1.0  ! default (same as AERMET)"                                        >> mmif.inp
echo "aer_min_obuk  1.0  ! default (same as AERMET)"                                        >> mmif.inp
echo "aer_min_speed 0.0  ! default (following Apr 2018 MMIF Guidance)"                      >> mmif.inp
echo                                                                                          >> mmif.inp
echo "# See the Users Guide for the OUTPUT keyword details"                                 >> mmif.inp
echo POINT  latlon     ${lati}   ${loni}  -5 ! LUGAR                                        >> mmif.inp
echo AER_layers       0       0          ! write only 2m and 10m data                       >> mmif.inp
echo Output aermod     useful   PUNTO_MMIFv342.info.txt                                     >> mmif.inp
echo Output aermod     sfc      PUNTO_MMIFv342.sfc                                         >> mmif.inp
echo Output aermod     PFL      PUNTO_MMIFv342.pfl                                         >> mmif.inp
echo                                                                                          >> mmif.inp
echo Output calpuff    useful   calmet.info.txt                                            >> mmif.inp
echo Output calpuff    calmet   calmet.met                                                 >> mmif.inp
echo Output calpuff    terrain  terrain.grd                                               >> mmif.inp
echo                                                                                          >> mmif.inp
echo Output calpuffv6  useful   calmetv6.info.txt                                          >> mmif.inp
echo Output calpuffv6  calmet   calmetv6.met                                               >> mmif.inp
echo Output calpuffv6  aux      calmetv6.aux                                               >> mmif.inp
echo Output calpuffv6  terrain  terrainv6.grd                                              >> mmif.inp
echo                                                                                          >> mmif.inp
echo "# INPUT gives filenames of either MM5 or WRF files"                                   >> mmif.inp
#echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_00:00:00
          >> mmif.inp
#echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_01:00:00
          >> mmif.inp
#echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_02:00:00
          >> mmif.inp
#echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_03:00:00
          >> mmif.inp
#echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_04:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_05:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_06:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_07:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_08:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_09:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_10:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_11:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_12:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_13:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_14:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_15:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_16:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_17:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_18:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_19:00:00
          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_20:00:00
          >> mmif.inp
```

```
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_21:00:00
        >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_22:00:00
        >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr}-${mo}-${dy}_23:00:00
        >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_00:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_01:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_02:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_03:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_04:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_05:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_06:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_07:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_08:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_09:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_10:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_11:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_12:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_13:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_14:00:
00          >> mmif.inp
echo INPUT /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/WRF1k/${ymd2}00/wrfout_d03_${yr2}-${mo2}-${dy2}_15:00:
00          >> mmif.inp

# Fin de namelist ---------------------------------
wine mmif.exe
cp -f /home/sea/Escritorio/SEA/INCENDIOS/INCENDIOS_URBANOS/MMIFv342/PUNTO_MMIFv342.* /home/sea/Escritorio/SEA/INCENDI
OS/INCENDIOS_URBANOS/AERMOD
#cp -f /home/sea/Escritorio/SEA/SERVIDOR/PANTERA/scripts/bash/aermod_exe/AERMOD_2/OLORES/MMIFv342/calmet*        /home/sea/E
scritorio/SEA/SERVIDOR/PANTERA/scripts/bash/aermod_exe/AERMOD_2/OLORES/DATOS
#cp -f /home/sea/Escritorio/SEA/SERVIDOR/PANTERA/scripts/bash/aermod_exe/AERMOD_2/OLORES/MMIFv342/terrain* /home/sea/Escrito
rio/SEA/SERVIDOR/PANTERA/scripts/bash/aermod_exe/AERMOD_2/OLORES/DATOS

else
  echo Ingresar 5 variables a script
  echo "./preparaDATOS.sh  LATI LONI YYYYMMMDD HRL NHP"
fi

#Ubicacion: ftp://ftp.senamhi.gob.pe/SEA/MMIF/
```

2.4. Boletin de Calidad del Aire

2.4.1. Gráficas del comportamiento diario de las variables metereológicas

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

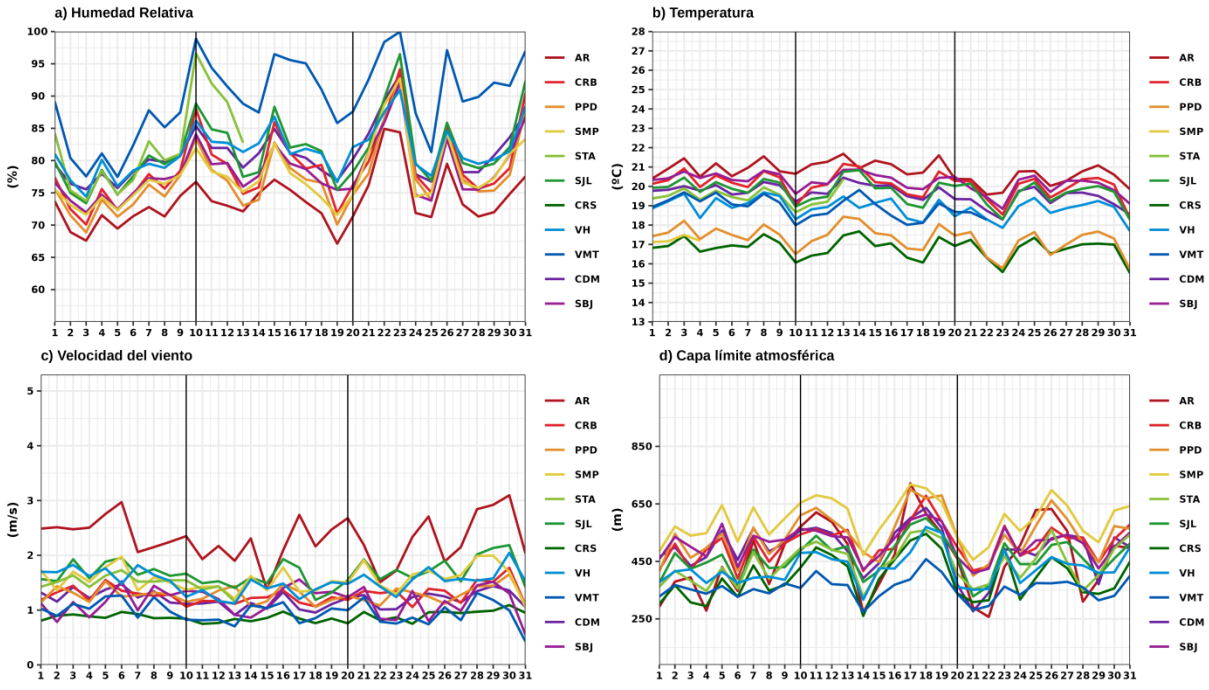
```
library(ggplot2)
library(openair)
library(latticeExtra)
library(directlabels)
library(reshape2)
library(gridExtra)
library(readxl)
library(readr)
library(tidyverse)
```

```
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/Agosto__/Boletin de AMLC/")
#C:/Users/hgomez/Documents/proyectosKVN/plt/shapefile/shapefile
#TEMPERATURA

t=read.csv('Temp_Ago.csv',sep =';')
t$date=as.POSIXct(strptime(t$date, format = "%d/%m/%Y %H:%M","UTC"))
t=timeAverage(t,avg.time = 'day',statistic = 'mean')
t$day=format(t$date,'%d')
t$day=as.integer(t$day)
t
t=t[,c(2:13)] #elegir las estaciones con lo que se va a trabajar
#agregado para linux
df <- pivot_longer(t, cols = -day, names_to = "variable", values_to = "value")
#para windows
#df=melt(t,id.vars = c("day","date"))
# ,'CRS' ,'darkgreen'

fraktal <- c('#af141f','#e2242d','#e38f2d','#e3ca3c','#89c138','#1e9734','darkgreen','#018fd7','#0159b2','#6a24a1','#991d94')
#'#1e9734'

plott=ggplot(df,aes(x=day,y=value,group=variable))+geom_line(aes(color=variable),linewidth=0.8)+
  scale_color_manual(values = fraktal,limits=c('AR','CRB','PPD','SMP','STA','SJL','CRS','VH','VMT','CDM','SBJ')) +
  theme_bw()+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8)),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=2, size=rel(0.8)))+
  theme(axis.title.x = element_text(face="bold", vjust=2, size=rel(0.8)))+
  theme(legend.text=element_text(size=7,face = "bold"))+
  theme(legend.margin = margin(1,1,1,1))+
  labs(y="(ºC)")+theme(legend.title=element_blank())+theme(legend.position='right')+
  theme(legend.key = element_rect(color = NA, fill = NA))+theme(axis.title.x = element_blank())+
  scale_x_continuous(breaks=seq(1,31,1),expand = c(0,0))+
  scale_y_continuous(limits = c(13,28),breaks = seq(13,28,1),expand = c(0,0))+
  ggtitle("b) Temperatura") + theme (plot.title = element_text(family="Arial",
                                     size=rel(0.9), #Tama?o relativo de la letra del t?tulo
                                     vjust=2, #Justificaci?n vertical, para separarlo del gr?fico
                                     face="bold", #Letra negrilla. Otras posibilidades "plain", "italic", "bold" y "bold.italic"
                                     color="black", #Color del texto
                                     lineheight=1.5,hjust = 0.0))+
  geom_vline(xintercept = c(10,20), colour ="black",lwd=0.4)+guides(color=guide_legend(ncol=1))

plott

#HR

hr=read.csv('HR_Ago.csv',sep = ';')
hr$date=as.POSIXct(strptime(hr$date, format = "%d/%m/%Y %H:%M","UTC"))
hr=timeAverage(hr,avg.time = 'day',statistic = 'mean')
hr$day=format(hr$date,'%d')
hr$day=as.integer(hr$day)
hr
hr=hr[,c(2:12)]

df <- pivot_longer(hr, cols = -day, names_to = "variable", values_to = "value")

#df=melt(hr,id.vars = 'day')

plothr2=ggplot(df,aes(x=day,y=value,group=variable))+geom_line(aes(color=variable),linewidth=0.8)+
  scale_color_manual(values = fraktal,limits=c('AR','CRB','PPD','SMP','STA','SJL','CRS','VH','VMT','CDM','SBJ')) +
  theme_bw()+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8)),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=2, size=rel(0.8)))+
  theme(axis.title.x = element_text(face="bold", vjust=2, size=rel(0.8)))+
  theme(legend.text=element_text(size=7,face = "bold"))+
  theme(legend.margin = margin(1,1,1,1))+
  labs(y="(%)")+theme(legend.title=element_blank())+theme(legend.position='right')+
  theme(legend.key = element_rect(color = NA, fill = NA))+theme(axis.title.x = element_blank())+
  scale_x_continuous(breaks=seq(1,31,1),expand = c(0,0))+
  scale_y_continuous(limits = c(55,100),breaks=seq(60,100,5),expand = c(0,0))+
  ggtitle("a) Humedad Relativa") + theme (plot.title = element_text(family="Arial",
```

```
                                    size=rel(0.9), #Tama?o relativo de la letra del t?tulo
                                    vjust=2, #Justificaci?n vertical, para separarlo del gr?fico
                                    face="bold", #Letra negrilla. Otras posibilidades "plain", "italic", "bold" y "bold.italic"
                                    color="black", #Color del texto
                                    lineheight=1.5,hjust = 0.0))+
  geom_vline(xintercept = c(10,20), colour ="black",lwd=0.4)+guides(color=guide_legend(ncol=1))
plothr2

#Vv

vv=read.csv('Viento_Ago.csv',sep =';')
vv$date=as.POSIXct(strptime(vv$date, format = "%d/%m/%Y %H:%M","UTC"))
vv=timeAverage(vv,avg.time = 'day',statistic = 'mean')
vv$day=format(vv$date,'%d')
vv$day=as.integer(vv$day)
vv
vv=vv[,c(2:13)]
vv
df <- pivot_longer(vv, cols = -day, names_to = "variable", values_to = "value")

#df=melt(vv,id.vars = 'day')

plotvv=ggplot(df,aes(x=day,y=value,group=variable))+geom_line(aes(color=variable),linewidth=0.8)+
  scale_color_manual(values = fraktal,limits=c('AR','CRB','PPD','SMP','STA','SJL','CRS','VH','VMT','CDM','SBJ')) +
  theme_bw()+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8)),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=2, size=rel(0.8)))+
  theme(axis.title.x = element_text(face="bold", vjust=2, size=rel(0.8)))+
  theme(legend.text=element_text(size=7,face = "bold"))+
  theme(legend.margin = margin(1,1,1,1))+
  labs(y="(m/s)")+theme(legend.title=element_blank())+theme(legend.position='right')+
  theme(legend.key = element_rect(color = NA, fill = NA))+theme(axis.title.x = element_blank())+
  scale_x_continuous(breaks=seq(1,31,1),expand = c(0,0))+
  scale_y_continuous(expand = c(0,0),limits = c(0,5.3))+
  ggtitle("c) Velocidad del viento") + theme (plot.title = element_text(family="Arial",
                                    size=rel(0.9), #Tama?o relativo de la letra del t?tulo
                                    vjust=2, #Justificaci?n vertical, para separarlo del gr?fico
                                    face="bold", #Letra negrilla. Otras posibilidades "plain", "italic", "bold" y "bold.italic"
                                    color="black", #Color del texto
                                    lineheight=1.5,hjust = 0.0))+
  geom_vline(xintercept = c(10,20), colour ="black",lwd=0.4)+guides(color=guide_legend(ncol=1))

plotvv

#bhl 2
bhl=read.csv('Pbl_Ago.csv',sep =';')
head(bhl)
#bhl<- mutate(bhl,est= ifelse(station == "ANTONIO RAIMONDI", "AR",
#                  ifelse(station == "CAMPO DE MARTE", "CDM",
#                  ifelse(station == "CARABAYLLO", "CRB",
#                  ifelse(station == "PUENTE PIEDRA","PPD",
#                  ifelse(station == "SAN BORJA", "SBJ",
#                  ifelse(station == "SAN JUAN DE LURIGANCHO", "SJL",
#                  ifelse(station == "SAN MARTIN DE PORRES", "SMP",
#                  ifelse(station == "SANTA ANITA", "STA",
#                  ifelse(station == "VILLA MARIA DEL TRIUNFO", "VMT", "VH"))))))))))
#vv$date=as.POSIXct(strptime(vv$date, format = "%d/%m/%Y %H:%M","UTC"))
bhl$date=as.POSIXct(strptime(bhl$date, format = "%d/%m/%Y %H:%M","UTC"))
head(bhl)
bhl=timeAverage(bhl,avg.time = 'day',statistic = 'mean')
bhl$day=format(bhl$date,'%d')
bhl$day=as.integer(bhl$day)
bhl
bhl=bhl[,c(2:13)]
bhl
df <- pivot_longer(bhl, cols = -day, names_to = "variable", values_to = "value")

#df=melt(bhl,id.vars = 'day')

plotbhl=ggplot(df,aes(x=day,y=value,group=variable))+geom_line(aes(color=variable),linewidth=0.8)+
```

```
scale_color_manual(values = fraktal,limits=c('AR','CRB','PPD','SMP','STA','SJL','CRS','VH','VMT','CDM','SBJ')) +
theme_bw()+
theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.8)),
      axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
theme(axis.title.y = element_text(face="bold", vjust=2, size=rel(0.8)))+
theme(axis.title.x = element_text(face="bold", vjust=2, size=rel(0.8)))+
theme(legend.text=element_text(size=7,face = "bold"))+
theme(legend.margin = margin(1,1,1,1))+
labs(y="(m)")+theme(legend.title=element_blank())+theme(legend.position='right')+
theme(legend.key = element_rect(color = NA, fill = NA))+theme(axis.title.x = element_blank())+
scale_x_continuous(breaks=seq(1,31,1),expand = c(0,0))+
scale_y_continuous(limits = c(90,1100),breaks = seq(50,1000,200),expand = c(0,0))+
ggtitle("d) Capa límite atmosférica ") + theme (plot.title = element_text(family="Arial",
                                size=rel(0.9), #Tama?o relativo de la letra del t?tulo
                                vjust=2, #Justificaci?n vertical, para separarlo del gr?fico
                                face="bold", #Letra negrilla. Otras posibilidades "plain", "italic", "bold" y "bold.italic"
                                color="black", #Color del texto
                                lineheight=1.5,hjust = 0.0))+
geom_vline(xintercept = c(10,20), colour ="black",lwd=0.4)+guides(color=guide_legend(ncol=1))

plotbhl

a = grid.arrange(plothr2,plott,plotvv,plotbhl,ncol=2)
ggsave(filename = "01Met_Sep.png", plot =a, width = 30, height = 17, dpi = 400, units = "cm")
```

Figura N° 12. Gráficas del comportamiento diario de varibles meteorológicas



2.4.2. Gráficas del comportamiento diario del material particulado

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

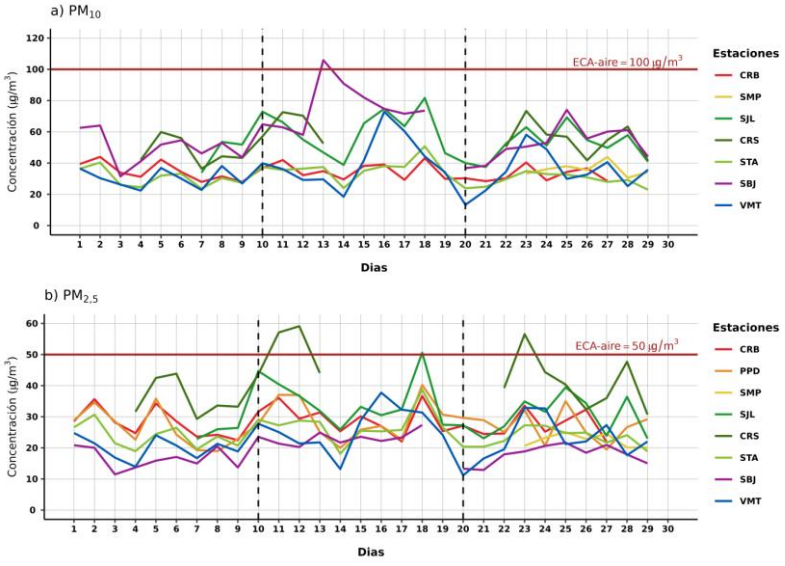
```
library(ggplot2)
library(openair)
library(latticeExtra)
library(directlabels)
library(reshape2)
library(gridExtra)
library(readxl)
library(readr)
library(tidyverse)
```

Luego se tiene que establecer la carpeta de trabajo, donde se encuentren los datos de PM_{10} y $PM_{2,5}$ en formato .csv. Posteriormente se importan y se ejecuta las siguientes líneas de código para la generación de las gráficas, se unen en un panelPlot y se guardan en formato .png con el nombre de **PM.png**.

```
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/BOLETIN_CALIDAD_AIRE")

####### PM10
PM10 = read_delim("PM10.csv",col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

PM10_DIA<-timeAverage(PM10, avg.time="day", statistic="mean", data.thresh=75)

PM10_DIA$dia=c(1:30) #CAMBIAR POR LA CANTIDAD DE DIAS QUE TIENE EL MES
head(PM10_DIA)
# PM10_DIA
#write.table(PM10_DIA, "PM10-RESUMEN.csv", sep = ";", na = "", row.names = F)
PM10_DIA=PM10_DIA[,c(2:9)]
frame=melt(PM10_DIA, id.vars = 'dia')
frame$variable = factor(frame$variable,levels = c("CRB","SMP","SJL","CRS","STA","SBJ","VMT"))

#resumen <- summary(PM10_DIA)

PM10_DIA_Grafico <- ggplot(frame,aes(x=dia,y=value))+
  geom_line(aes(colour=variable),size=0.8)+
  ggtitle(expression('a) PM'[10]))+
  labs(#title = expression('a) PM'[10]),
       x="\nDias",
       y=expression(Concentración * " (" * mu * "g/m" ^ 3 *")"))+
  scale_x_continuous(breaks = seq(1,30,1))+
  scale_y_continuous(breaks = seq(0,250,20),limits = c(0,120))+
  geom_hline(yintercept=100,colour="brown",size=0.8)+
  #scale_colour_brewer(palette = 'Dark2',breaks=c('CRB','SMP','STA','SJL','VMT','SBJ'))+
  theme_classic()+theme(legend.position='right', legend.direction = 'vertical')+
  theme (plot.title = element_text(face="bold", colour="black", size=rel(1)),
         axis.text.x = element_text(face="bold", colour="black", size=rel(0.8)),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
  theme(legend.title = element_text(size=rel(0.8),face = "bold"))+
  theme(legend.text=element_text(size=7,face = "bold"))+
  theme(legend.margin = margin(1,1,1,1)) +
  theme(panel.grid.major = element_line(color="gray",size = 0.2))+
  geom_vline(xintercept = c(10,20), colour ="gray1",lwd=0.6,linetype='dashed')+
  geom_text(data = NULL, x = 28, y = 105,
            #label = expression("ECA PM"[10] == 100 ~ mu*g/m^3),col="red",size=3)
            label = expression("ECA-aire" == 100 ~ mu*g/m^3),col="brown",size=3)+
  scale_color_manual(name="Estaciones",
                     values = c("CRB"="#e2242d","SMP"="#e3ca3c","STA"="#89c138","SJL"="#1e9734",
                                "CRS"="#3f7117",
                                "SBJ"="#991d94","VMT"="#0159b2"))
PM10_DIA_Grafico

####### PM2.5

PM25 = read_delim("PM25.csv",col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M")))

PM25_DIA<-timeAverage(PM25, avg.time="day", statistic="mean", data.thresh=75)
head(PM25_DIA)
```

```
PM25_DIA$dia<-c(1:30) #CAMBIAR POR LA CANTIDAD DE DIAS QUE TIENE EL MES
head(PM25_DIA)
# write.table(PM25_DIA, "PM25-RESUMEN.csv", sep = ";", na = "", row.names = F)
PM25_DIA=PM25_DIA[,c(2:10)]
frame=melt(PM25_DIA, id.vars = 'dia')

frame$variable = factor(frame$variable,levels = c("CRB","PPD","SMP","SJL","CRS","STA",
                                                  "SBJ","VMT"))

PM25_DIA_Grafico <- ggplot(frame,aes(x=dia,y=value))+
 geom_line(aes(colour=variable),size=0.8)+
 ggtitle(expression('b) PM'['2,5']))+
 labs(#title = expression('a) PM'[10]),
   x="\nDias",
   y=expression(Concentración * " (" * mu * "g/m" ^ 3 *")"))+
 scale_x_continuous(breaks = seq(1,30,1))+
 scale_y_continuous(breaks = seq(0,120,10), limits = c(0,60))+
 geom_hline(yintercept=50,colour="brown",size=0.8)+
 #scale_colour_brewer(palette = 'Dark2',breaks=c('CRB','PPD','SMP','SJL','STA','VMT','SBJ'))+
 theme_classic()+theme(legend.position='right', legend.direction = 'vertical')+ guides(fill = guide_legend(nrow = 1))+
 theme (plot.title = element_text(face="bold", colour="black", size=rel(1)),
       axis.text.x = element_text(face="bold", colour="black", size=rel(0.8)),
       axis.text.y = element_text(face="bold", colour="black", size=rel(0.8), hjust=0.6))+
 theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
 theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
 theme(legend.text=element_text(size=7,face = "bold"))+
 theme(legend.margin = margin(1,1,1,1)) +
 theme(legend.title = element_text(face = "bold",size=rel(0.8)))+
 theme(panel.grid.major = element_line(color="gray",size = 0.2))+
 geom_vline(xintercept = c(10,20), colour ="gray1",lwd=0.6,linetype='dashed')+
 geom_text(data = NULL, x = 28, y = 53,
         #label = expression("ECA PM"[2.5] == 50 ~ mu*g/m^3),col="red",size=3)
         label = expression("ECA-aire" == 50 ~ mu*g/m^3),col="brown",size=3)+
 scale_color_manual(name="Estaciones",
                values = c("CRB"="#e2242d","PPD"="#e38f2d","SMP"="#e3ca3c",
                       "STA"="#89c138","SJL"="#1e9734",
                       "CRS"="#3f7117","SBJ"="#991d94","VMT"="#0159b2"))

PM25_DIA_Grafico

PM_plot <- grid.arrange(PM10_DIA_Grafico, PM25_DIA_Grafico, ncol = 1)
ggsave(filename = "PM.png", plot =PM_plot,
     width = 22, height = 16, dpi = 1000, units = "cm")
```

Figura N° 13. Gráficas del comportamiento diario del material particulado



2.4.3. Estados de Calidad del Aire

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

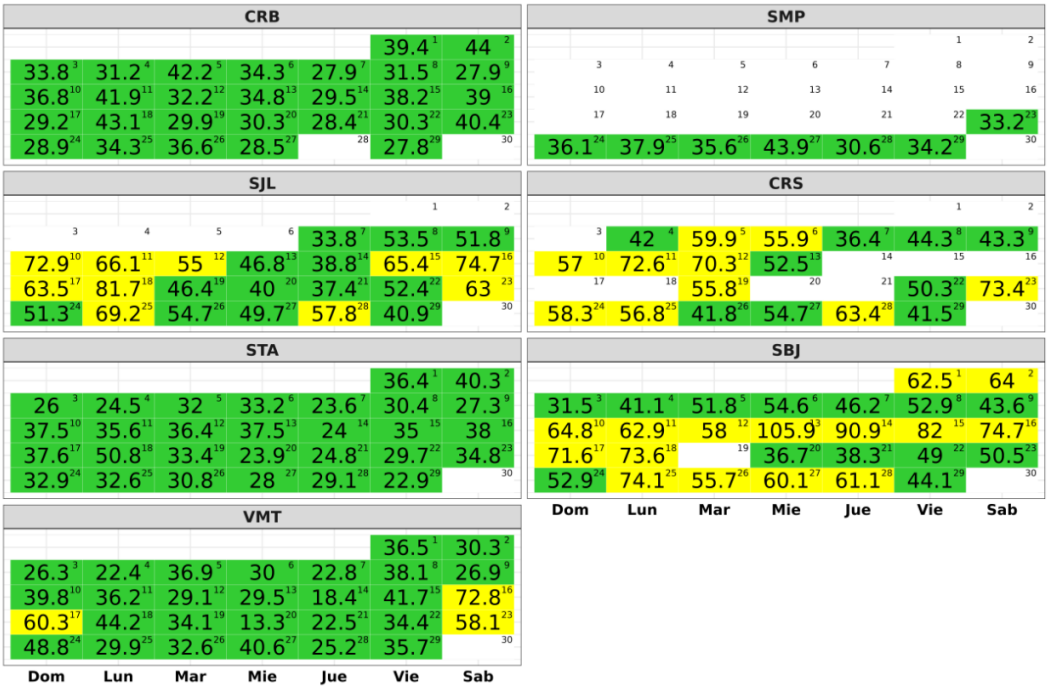
```
#install.packages("ggplot2", dependencies = T)
#install.packages("openair", dependencies = T)

rm (list = ls())

library(ggplot2)
library(openair)
library(latticeExtra)
library(directlabels)
library(gtable)
library(egg)
library(grid)
library(DT)
library(kableExtra)
library(htmltools)
library(readxl)
library(gridExtra)
library(lubridate)
library(reshape2)
library(lattice)
library(png)
```

Luego se tiene que establecer la carpeta de trabajo, donde se encuentren los datos de PM_{10} y $PM_{2,5}$ en formato .xlsx. Posteriormente se importan y se ejecuta las siguientes líneas de código para la generación de las gráficas y se guardan en formato .png con el nombre de **PM10.png** y **PM25.png.**

```
getwd()
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/BOLETIN_CALIDAD_AIRE")
datos<-'DATOS_SETIEMBRE_2023.xlsx'
lts<-lapply(excel_sheets (datos),read_excel,path=datos)
PM10<-lts[[1]]
PM25<-lts[[2]]

####PM10####

#ggplot
tabla = timeAverage(PM10,avg.time = "day",data.thresh = 75)
frame=reshape2::melt(tabla,id.vars="date")
frame$dia <- as.numeric(format(frame$date, "%d"))
frame$week = lubridate::week(frame$date)
frame$weekday = lubridate::wday(frame$date)
frame$weekday = factor(frame$weekday,labels = c("Dom","Lun","Mar","Mie","Jue","Vie","Sab"))
frame$variable = factor(frame$variable,levels = c("CRB","SMP","SJL","CRS","STA","SBJ","VMT"))


frame$AQI <- ifelse(frame$value<54.999,"Buena",
                    ifelse(frame$value>55.0001 & frame$value<154,"Moderada",
                           ifelse(frame$value>154.0001 & frame$value<254,"Insalubre para grupos sensibles","Insalubre")))

frame$AQI = ifelse(is.na(frame$AQI),"NA",frame$AQI)


# Configurar el tema del gráfico
theme_set(theme_bw())
```

```
# Crear el gráfico de calendario
calendar_pm10 =
  ggplot(frame, aes(x = weekday, y = week, fill = AQI)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(value,1)),size=7,color="black")+
  geom_text(aes(label=dia),size=3,nudge_x = 0.4,nudge_y = 0.3)+
  scale_fill_manual(values=c("Buena"="#33cc33", "Moderada"="yellow","NA"="white",
                             "Insalubre para grupos sensibles"="#ffc000","Insalubre"="red")) +
  scale_y_reverse()+
  labs(x = "", y = "", title = quickText(""),
       fill = "Valor") +
  facet_wrap(~ variable, ncol = 2) +
  theme(legend.position = "none",
        axis.text.x = element_text(face = "bold",size = 14,colour = "black"),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        #strip.background = element_blank(),
        strip.text = element_text(face="bold",size=15))

calendar_pm10
ggsave(filename = "PM10.png", plot =calendar_pm10, width = 36, height = 24, dpi = 1200, units = "cm")
```

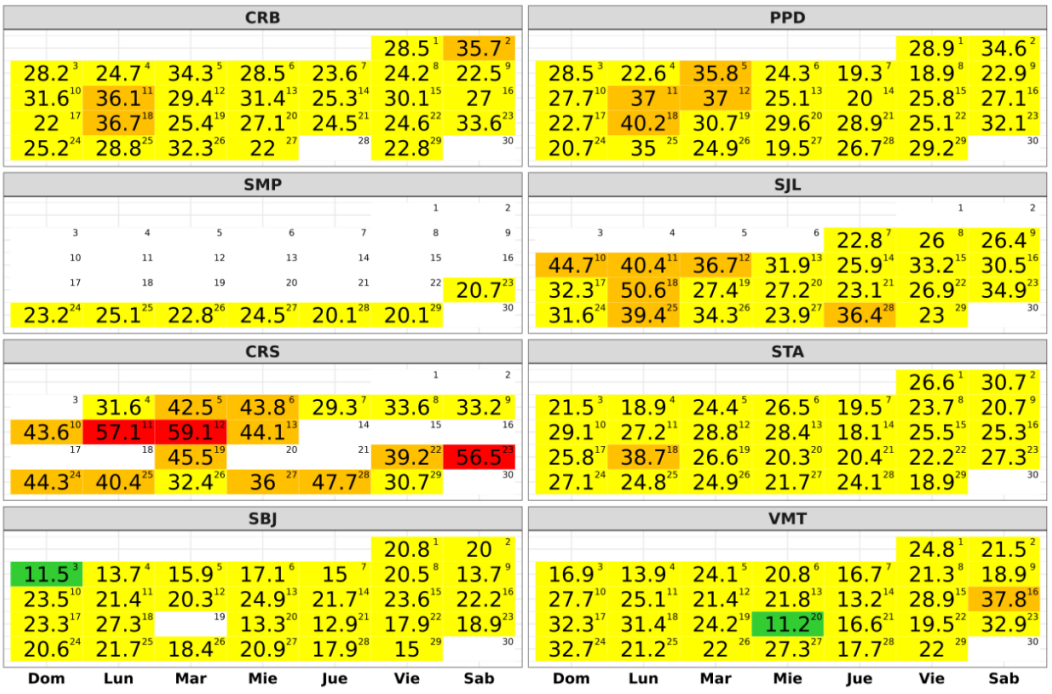
Figura N° 14. Estados de la Calidad del Aire para PM_{10}




```
####PM2.5####

 #ggplot
 tabla2 = timeAverage(PM25,avg.time = "day",data.thresh = 75)
df=reshape2::melt(tabla2,id.vars="date")
 df$dia <- as.numeric(format(df$date, "%d"))
 df$week = lubridate::week(df$date)
 df$weekday = lubridate::wday(df$date)
 df$weekday = factor(df$weekday,labels = c("Dom","Lun","Mar","Mie","Jue","Vie","Sab"))
 df$variable = factor(df$variable,levels = c("CRB","PPD","SMP","SJL","CRS","STA","SBJ","VMT"))

 df$AQI <- ifelse(df$value<12.1,"Buena",
                  ifelse(df$value>12.1001 & df$value<35.4,"Moderada",
                         ifelse(df$value>35.4001 & df$value<55.4,"Insalubre para grupos sensibles","Insalubre")))
```

```
df$AQI = ifelse(is.na(df$AQI),"NA",df$AQI)



# Configurar el tema del gr?fico
theme_set(theme_bw())


# Crear el gr?fico de calendario
calendar_pm25 =
  ggplot(df, aes(x = weekday, y = week, fill = AQI)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(value,1)),size=7,color="black")+
  geom_text(aes(label=dia),size=3,nudge_x = 0.4,nudge_y = 0.3)+
  scale_fill_manual(values=c("Buena"="#33cc33", "Moderada"="yellow","NA"="white",
                             "Insalubre para grupos sensibles"="#ffc000","Insalubre"="red")) +
  scale_y_reverse()+
  labs(x = "", y = "", title = quickText(""),
       fill = "Valor") +
  facet_wrap(~ variable, ncol = 2) +
  theme(legend.position = "none",
        axis.text.x = element_text(face = "bold",size = 14,colour = "black"),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        #strip.background = element_blank(),
        strip.text = element_text(face="bold",size=15))

calendar_pm25
```

Figura N° 15. Estados de la Calidad del Aire para $PM_{2,5}$



2.4.4. Gráfica de Gases

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
library(readxl)
library(zoo)
library(lubridate)
library(dplyr)
```

```
library(reshape2)
library(openair)
library(ggplot2)
library(grid)
library(gridExtra)
```

Luego se tiene que establecer la carpeta de trabajo, donde se encuentren los datos de CO, SO_2, NO_2 y O_3 en formato .xlsx. Posteriormente se importa y se ejecuta las siguientes líneas de código para la generación de las gráficas y se guardan en formato .jpg con el nombre de **GASES.jpg.**

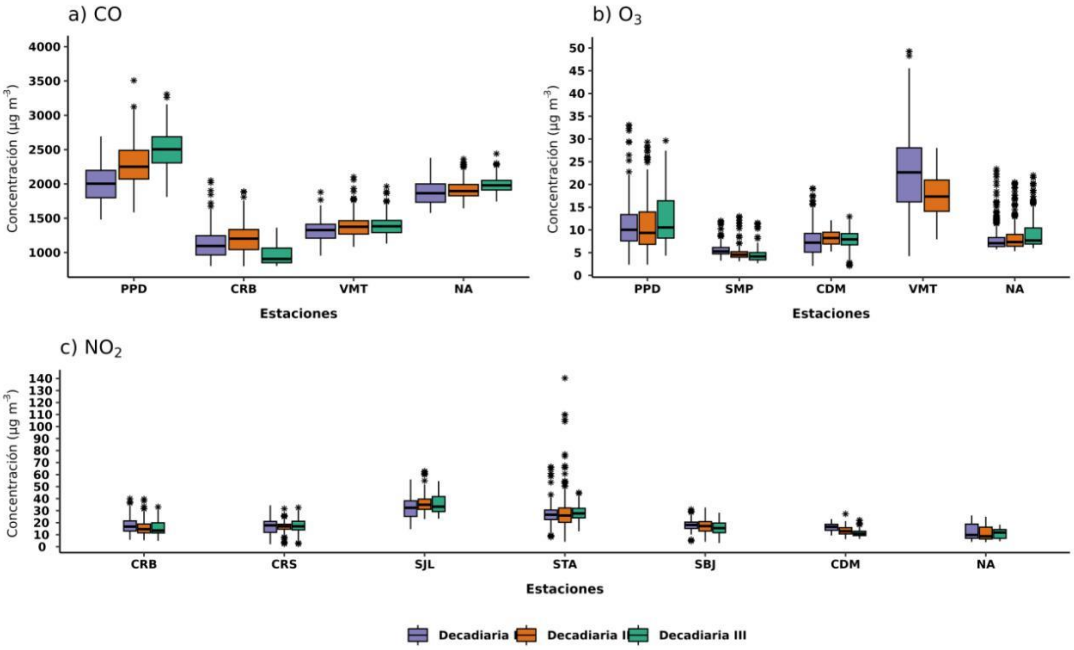
```
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/BOLETIN_CALIDAD_AIRE")

# Ruta al archivo Excel
ruta_excel <- "DATOS_SETIEMBRE_2023.xlsx"

# Obtener la lista de hojas en el archivo Excel
hojas = excel_sheets(ruta_excel)
#indices = which(hojas ==c("PM10","PM25"))
#hojas = hojas[-indices]

convNO2 = function(){

 datos = read_excel(ruta_excel, sheet = "NO2")
 columnas = names(datos)
 patron = "NO2_"
 indices = grep(patron,columnas)
 columnas_query = columnas[indices]
 tabla1 = datos %>% select(columnas_query)
 estaciones = names(tabla1)
 estaciones = gsub("NO2_", "", estaciones)
 names(tabla1) = estaciones

 convGAS = function(x){
  return(round(x*1.88,digits = 1))
 }

 tabla1 = lapply(tabla1,convGAS) %>% as.data.frame()
 tabla2 = datos %>% select(date)
 tabla = cbind(tabla2,tabla1)

 # Crear el nombre del archivo CSV
 nombre_csv <- paste0("NO2", ".csv")

 # Guardar los datos como un archivo CSV
 #return(tabla)
 write.csv(tabla,nombre_csv,row.names = F)

 cat("Se ha exportado el archivo", nombre_csv, "\n")

 #Graficar

 tablaG = tabla %>% mutate(date=as.POSIXct(date,format = "%d/%m/%Y %H:%M:%S","UTC"),
                           dia=format(date,'%d'),
                           hora=format(date,'%H'))

 tablaG = tablaG %>% mutate(decadiario = ifelse(dia<11,'Decadiaria I',
                                    ifelse(dia<21,'Decadiaria II','Decadiaria III')))

 tablaG=tablaG[,-1]
 tablaG=melt(tablaG,id.vars = c('dia','decadiario','hora'))
 tablaG$variable=factor(tablaG$variable,levels = c('CRB','CRS','SJL','STA','SBJ','CDM','VMT')) #colocar orden de acuerdo a l
as estaciones disponibles

 PLOTno2<-ggplot(tablaG,aes(x= variable , y = value, fill=decadiario))+
  geom_boxplot(alpha =0.9,width=0.3,outlier.shape = 8 ,outlier.size = 1,show.legend = T,
               colour="black",lwd=0.4, weight=.15)+
  ggtitle(expression('c) NO'[2]))+
```

```
  theme_classic()+scale_fill_brewer(palette="Dark2",direction = -1)+
  labs(x='\nEstaciones', y=quickText("Concentración (µg/m3)")) + theme(axis.title.x=element_blank())+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.9)),
        axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
  theme(legend.text=element_text(size=8,face = "bold"))+
  scale_y_continuous(breaks = seq(0,200,10))+theme(legend.position='bottom') +
  theme(legend.title=element_blank())

 #ggsave(filename = "NO2.tiff", plot =PLOTno2, width = 15.0, height = 8.3, dpi = 500, units = "cm")
 return(PLOTno2)
}

convCO = function(){

 datos = read_excel(ruta_excel, sheet = "CO")
 tabla1 = datos %>% select(!date)

 convGAS = function(x){
  return(round(x*1000*1.15, digits = 1))
 }

 tabla1 = lapply(tabla1,convGAS) %>% as.data.frame()
 tabla2 = datos %>% select(date)
 tabla = cbind(tabla2,tabla1)

 # Crear el nombre del archivo CSV
 nombre_csv <- paste0("CO", ".csv")

 # Guardar los datos como un archivo CSV
 #return(tabla)
 write.csv(tabla,nombre_csv,row.names = F)

 cat("Se ha exportado el archivo", nombre_csv, "\n")

 tablaG = tabla %>% mutate(date=as.POSIXct(date,format = "%d/%m/%Y %H:%M:%S","UTC"),
                           dia=format(date,'%d'),
                           hora=format(date,'%H'))

 tablaG = tablaG %>% mutate(decadiario = ifelse(dia<11,'Decadiaria I',
                                        ifelse(dia<21,'Decadiaria II','Decadiaria III')))

 tablaG=tablaG[,-1]
 tablaG=melt(tablaG,id.vars = c('dia','decadiario','hora'))
 tablaG$variable=factor(tablaG$variable,levels = c('PPD','CRB','SBJ','VMT')) #colocar orden de acuerdo a las estaciones dispo
nibles

 #Graficando

 PLOTco<-ggplot(tablaG,aes(x= variable , y = value, fill=decadiario))+
  geom_boxplot(alpha =0.9,width=0.9,outlier.shape = 8 ,outlier.size = 1,show.legend = T,
               colour="black",lwd=0.4, weight=.15)+
  ggtitle(expression('a) CO'))+
  theme_classic()+scale_fill_brewer(palette="Dark2",direction = -1)+
  labs(x='\nEstaciones', y=quickText("Concentración (µg/m3)")) + theme(axis.title.x=element_blank())+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.9)),
        axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.8)))+theme(legend.text=element_text(size=8,face = "bold
"))+
  scale_y_continuous(limits = c(800,4000),breaks = seq(0,30000,500))+theme(legend.position='bottom') +
  theme(legend.title=element_blank(),legend.position = "none")

}

convO3 = function(){

 datos = read_excel(ruta_excel, sheet = "O3")
 tabla1 = datos %>% select(!date)
```

```
convGAS = function(x){
  return(round(x*1.96, digits = 1))
}

tabla1 = lapply(tabla1,convGAS) %>% as.data.frame()
tabla2 = datos %>% select(date)
tabla = cbind(tabla2,tabla1)
estaciones = names(tabla1)

for (i in 1:length(estaciones)) {

  df = rollingMean(tabla, pollutant = estaciones[i],
                   new.name = paste0(estaciones[i],"_8h"),
                   width=8, data.thresh = 75, align = "right")
  df1 = df %>% select(last_col())

  tabla = cbind(tabla,df1)

}

tabla = tabla %>% select("date",paste0(estaciones,"_8h"))
names(tabla) = c("date",estaciones)

# Crear el nombre del archivo CSV

nombre_csv <- paste0("O3_8h", ".csv")

# Guardar los datos como un archivo CSV
write.csv(tabla,nombre_csv,row.names = F)

cat("Se ha exportado el archivo", nombre_csv, "\n")

#Graficar

tablaG = tabla %>% mutate(date=as.POSIXct(date,format = "%d/%m/%Y %H:%M:%S","UTC"),
                          dia=format(date,'%d'),
                          hora=format(date,'%H'))

tablaG = tablaG %>% mutate(decadiario = ifelse(dia<11,'Decadaria I',
                                               ifelse(dia<21,'Decadaria II','Decadaria III')))

tablaG=tablaG[,-1]
tablaG=melt(tablaG,id.vars = c('dia','decadiario','hora'))
tablaG$variable=factor(tablaG$variable,levels = c('PPD','SMP','CRS','SBJ','CDM','VMT')) #colocar orden de acuerdo a las est
aciones disponibles

#Graficando

PLOTO3<-ggplot(tablaG,aes(x= variable , y = value, fill=decadiario))+
  geom_boxplot(alpha =0.9,width=0.6,outlier.shape = 8 ,outlier.size = 1,show.legend = T,
               colour="black",lwd=0.4, weight=.15)+
  ggtitle(expression('b) O'[3]))+
  theme_classic()+scale_fill_brewer(palette="Dark2",direction = -1)+
  labs(x='\nEstaciones', y=quickText("Concentración (µg/m3)")) + theme(axis.title.x=element_blank())+
  theme (axis.text.x = element_text(face="bold", colour="black", size=rel(0.9)),
         axis.text.y = element_text(face="bold", colour="black", size=rel(0.9), hjust=0.6))+
  theme(axis.title.x = element_text(face="bold", vjust=1.5, size=rel(0.8)))+
  theme(axis.title.y = element_text(face="bold", vjust=1.5, size=rel(0.8)))+theme(legend.text=element_text(size=8,face = "bold
"))+
  scale_y_continuous(limits = c(2,50),breaks = seq(0,50,5))+theme(legend.position='bottom') +
  theme(legend.title=element_blank(),legend.position = "none")

#ggsave(filename = "O3.tiff", plot =PLOTO3, width = 15.0, height = 8.3, dpi = 500, units = "cm")
return(PLOTO3)
}

PLOTo3 = convO3()
PLOTno2 = convNO2()
PLOTco = convCO()
```

```
########## MULTIGRAFICOS

GASES<-grid.arrange(PLOTco,PLOTo3,ncol=2) #,widths = c(1, 1, 3))
GASES2<-grid.arrange(GASES,PLOTno2,nrow=2)
ggsave(filename = "GASES.jpg", plot =GASES2, width = 26, height = 16, dpi = 1000, units = "cm")
```

Figura N° 16. Gases



2.5. Boletin “Monitoreo de la atmósfera en el Observatorio de Vigilancia Atmosférica MARCAPOMACOCHA”

2.5.1. Gráficas de la Columna Total de Ozono

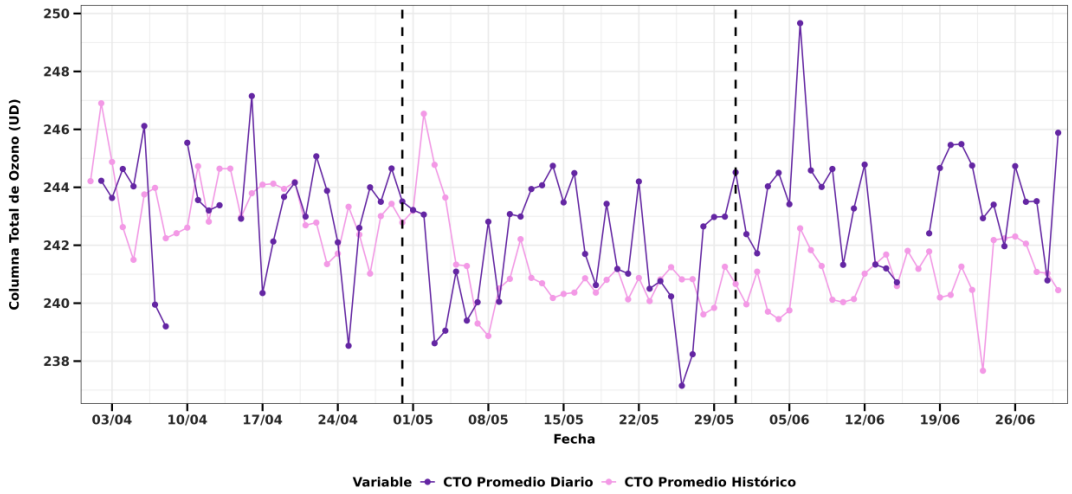
Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
library(ggplot2)
library(tidyverse)
library(lubridate)
library(latticeExtra)
library(openair)
library(patchwork)
library(arsenal)
library(hms)
library(plotly)
```

```
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/BOLETIN_MARCAPOMACOCHA")

##### GRAFICOS COLUMNA TOTAL DE OZONO #############

CTO1<- read.csv("CTO.csv", header = TRUE, sep = ";")

CTO1$date<-dmy(CTO1$date)
CTO1<- mutate(CTO1, mes= month(date))


#Estadistica descriptiva de los valores de CTO
```

```
my_controls <- tableby.control(
  test = T,
  total = T,
  numeric.test = "kwt", cat.test = "chisq",
  numeric.stats = c("meansd", "medianq1q3", "range", "Nmiss2", "sum"),
  cat.stats = c("countpct", "Nmiss2"),
  stats.labels = list(
    meansd = "Mean (SD)",
    medianq1q3 = "Median (Q1, Q3)",
    range = "Min - Max",
    Nmiss2 = "Missing",
    sum = "Suma"
  )
)

table_two <- tableby(mes ~ cto,
                     data = CTO1,
                     control = my_controls
)
summary(table_two,
        title = "Summary Statistic of CTO data"
)
#Grafica de serie de tiempo de CTO
CTO<- ggplot(data=CTO1) +
  geom_line(mapping= aes(x= date, y= cto_normal,color = "CTO Promedio Histórico")) +
  geom_point(aes(x= date, y= cto_normal,color = "CTO Promedio Histórico"), size = 1.6) +
  geom_line(aes(x= date, y= cto,color = "CTO Promedio Diario")) +
  geom_point(aes(x= date, y= cto,color = "CTO Promedio Diario"), size = 1.6) +
  #geom_point(aes(x= date, y= cto_normal),color = "red", size = 2) +
  geom_vline(xintercept = c(as.Date("2023-04-30"), as.Date("2023-05-31")),
             linetype = 2, size = 0.8) +
  # geom_hline(yintercept=220,colour="brown",size=0.8)+
  # geom_text(data = NULL, x = as.Date("2023-06-17"), y = 219,
  #           label = "Umbral-agujero de ozono = 220 UD",col="brown",size=3) +
  scale_x_date(date_breaks = "1 week",
               date_labels = "%d/%m",
               limits = c(min(CTO1$date), max = max(CTO1$date)), expand=c(0.01,0.01),
               name = "Fecha") +
  scale_y_continuous(name = "Columna Total de Ozono (UD)",n.breaks = 7)+ #,limits = c(218,250)) +
  scale_color_manual(name="Variable",
                     values = c("CTO Promedio Histórico"="#F399E6",
                                "CTO Promedio Diario"="#6426A3"))+
  theme_bw() +
  theme(legend.position = "bottom",legend.key.width = unit(0.5,"cm"),
        legend.title = element_text(face = "bold",size = 10),
        legend.text = element_text(face = "bold",size = 10),
        axis.title = element_text(size = 10, face = "bold", color = "black"),
        axis.title.x = element_text(margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
        axis.title.y = element_text(margin = unit(c(0.5, 0.5, 0.5, 0.5), "cm")),
        axis.text.x = element_text(size = 10, color = "#262626", face = "bold", margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
        axis.text.y = element_text(size = 10, color = "#262626", face = "bold", margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
        axis.ticks = element_line(size = 0.7, color = "black"),
        axis.ticks.length = unit(0.2, "cm"))
CTO
ggsave(file = "cto_diario.png",
       units = c("cm"),
       width = 30,
       height = 14,
       dpi = 600,
       CTO)
```

Figura N° 16. Columna Total de Ozono (UD)



2.5.2. Gráficas de Calendario del Ozono

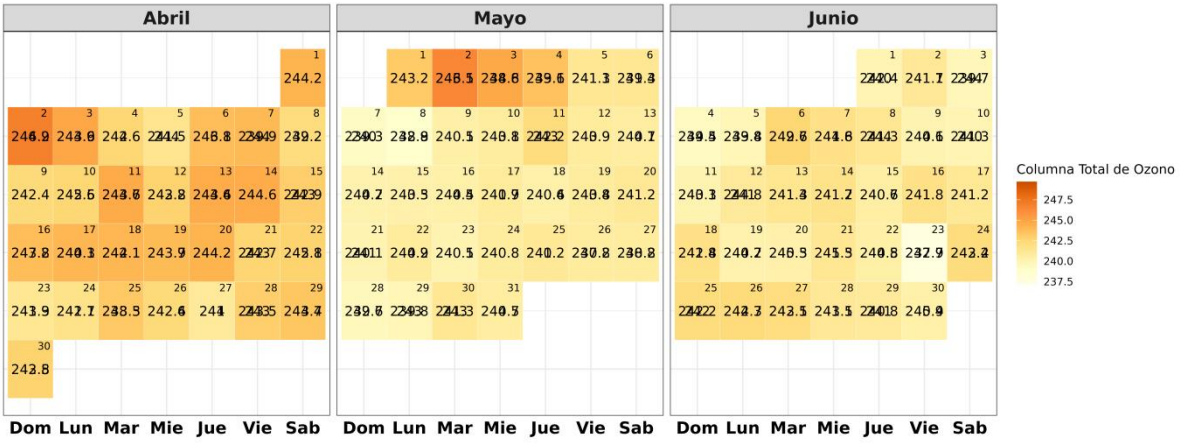
Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
rm (list = ls())
library(ggplot2)
library(openair)
library(latticeExtra)
library(directlabels)
library(gtable)
library(egg)
library(grid)
library(DT)
library(kableExtra)
library(htmltools)
library(readxl)
library(gridExtra)
library(lubridate)
library(reshape2)
library(lattice)
library(png)
library(tidyverse)
library(RColorBrewer)
```

```
getwd()
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/BOLETIN_MARCAPOMACOCHA")
datos_CTO<-read.csv("CTO.csv",sep = ";")
datos_CTO$date<-as.POSIXct(strptime(datos_CTO$date, format = "%d/%m/%Y","UTC"))
#datos_CTO = openair::import(file = "CTO.csv",
 #                  sep = ",",header.at = 1,date = "date",
  #                 date.format = "%d/%m/%Y")


#GRAFICA CALENDARIO DE OZONO

frame=reshape2::melt(datos_CTO,id.vars="date")
frame
frame$dia <- as.numeric(format(frame$date, "%d"))
frame$week = lubridate::week(frame$date)
frame$weekday = lubridate::wday(frame$date)
frame$weekday = factor(frame$weekday,labels = c("Dom","Lun","Mar","Mie","Jue","Vie","Sab"))
frame$month = lubridate::month(frame$date)
```

```
frame$month = factor(frame$month,labels = c("Abril","Mayo","Junio"))

frame = frame %>% rename(.,"CTO"="value")

frame$CTO = round(frame$CTO,1)

tabla1 = frame %>% filter(.,month=="Abril")
unique(tabla1$week)
tabla2 = frame %>% filter(.,month=="Mayo")
unique(tabla2$week)
tabla2 = tabla2 %>%
  mutate(week = if_else(week == 18, 13, week)) %>%
  mutate(week = if_else(week == 19, 14, week)) %>%
  mutate(week = if_else(week == 20, 15, week)) %>%
  mutate(week = if_else(week == 21, 16, week)) %>%
  mutate(week = if_else(week == 22, 17, week))

tabla3 = frame %>% filter(.,month=="Junio")
unique(tabla3$week)
tabla3 = tabla3 %>%
  mutate(week = if_else(week == 22, 13, week)) %>%
  mutate(week = if_else(week == 23, 14, week)) %>%
  mutate(week = if_else(week == 24, 15, week)) %>%
  mutate(week = if_else(week == 25, 16, week)) %>%
  mutate(week = if_else(week == 26, 17, week))


df = rbind(tabla1,tabla2,tabla3)


# Configurar el tema del gr?fico
theme_set(theme_bw())

color_palette <- c("#fffee5","#fffecc","#ffeda0", "#fed976","#feb24c", "#fd8d3c","#ec7014","#cc4c01")


color_na <- "white"

# Crear el gr?fico de calendario
calendar_cto =
  ggplot(df, aes(x = weekday, y = week, fill = CTO)) +
  geom_tile(color = "white") +
  geom_text(aes(label = round(CTO,1)),size=4,color="black") +
  geom_text(aes(label=dia),size=3,nudge_x = 0.3,nudge_y = 0.4) +
  scale_fill_gradientn(colors = color_palette,na.value = color_na) +
  scale_y_reverse()+
  labs(x = "", y = "", title = quickText(""),
       fill = "Columna Total de Ozono") +
  facet_wrap(~ month, ncol = 3) +
  theme(legend.position = "right",
        axis.text.x = element_text(face = "bold",size = 14,colour = "black"),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        #strip.background = element_blank(),
        strip.text = element_text(face="bold",size=15))

calendar_cto
ggsave(filename = "CTO_calendar.png", plot =calendar_cto, width = 36, height = 14, dpi = 1200, units = "cm")
```

Figura N° 17. Calendario de Ozono (UD)



2.5.3. Índice Ultravioleta Solar Máximo

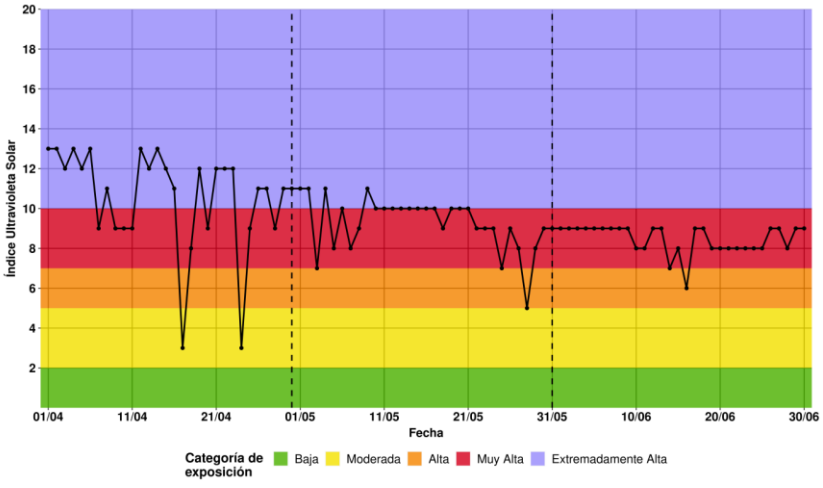
Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
rm (list = ls())
library(openair)
library(ggplot2)
library(tidyverse)
library(magrittr)
library(reshape2)
library(directlabels)
library(latticeExtra)
library(gridExtra)
library(grid)
```

```
#GRAFICA IUV MAX DIARIA CON CATEGORIAS DE EXPOSICION DE IUV
setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/BOLETIN_MARCAPOMACOCHA")

datos<-read.csv("RADIACION_MARCA.csv",sep = ",")
datos$date<-as.POSIXct(strptime(datos$date, format = "%d/%m/%Y %H:%M","UTC"))

datos = subset(datos,date >= as.POSIXct("2023-04-01 00:00",tz = "Etc/GMT") &
                 date <= as.POSIXct("2023-06-30 23:59",tz = "Etc/GMT"))

#datos_rolling$Indice_UV_Avg = as.numeric(datos_rolling$Indice_UV_Avg)
# datos_rolling$Indice_UV_Avg = ifelse(datos_rolling$Indice_UV_Avg>20,NA,datos_rolling$Indice_UV_Avg)
# datos_rolling$Indice_UV_Avg = ifelse(datos_rolling$Indice_UV_Avg<2,NA,datos_rolling$Indice_UV_Avg)
# datos_rolling$Indice_UV_Avg = ifelse(datos_rolling$Indice_UV_Avg==0,NA,datos_rolling$Indice_UV_Avg)

datos_ruv = datos %>% select(.,"date","UV_E_Avg","Indice_UV_Avg","SUV5_Avg")

#GRAFICA EVOLUCION DIARIA DEL IUV MAXIMO ####

datos_iuv = openair::rollingMean(datos_ruv,"Indice_UV_Avg",width = 30,new.name = "IUV_MM",data.thresh = 75)

datos_iuv = openair::timeAverage(datos_iuv,avg.time = "day",statistic = "max")

tabla = as.data.frame(datos_iuv) %>% select(.,date,IUV_MM) %>%
  mutate(IUV = round(IUV_MM,0))


tabla = tabla %>% mutate(num=c(1:nrow(tabla)))
```

```
s <- as.Date("2023-04-01",tz = "Etc/GMT")
e <- as.Date("2023-06-30",tz = "Etc/GMT")
etiquetas = format(seq(from=s, to=e, by=10),"%d/%m")


iuv_catg <- data.frame(xstart = c(0,2,5,7,10), xend = c(2,5,7,10,20),
                       Categoria = c("Baja","Moderada","Alta","Muy Alta","Extremadamente Alta"))

iuv_catg$Categoria = factor(iuv_catg$Categoria,levels = c("Baja","Moderada","Alta","Muy Alta","Extremadamente Alta"),
                            labels = c("Baja","Moderada","Alta","Muy Alta","Extremadamente Alta"))

IUV_MAX_DIARIO<- ggplot() +
  geom_rect(data=iuv_catg, aes(ymin = xstart,
                               ymax = xend,
                               xmin = - Inf,
                               xmax = Inf,
                               fill= Categoria), alpha = 0.8) +
  geom_line(data= tabla, aes(x= num, y = IUV), colour = "black", size = 0.9) +
  geom_point(data= tabla, aes(x= num, y = IUV), colour = "black", size = 1.6) +
  geom_vline(xintercept = c(30,61), linetype = 2, size = 0.8) +
  #geom_point(data= tabla,aes(x= num,y=IUV_MM),color="black",size=2) +
  scale_y_continuous(breaks = seq(2, 20, 2),
                     limits = c(0,20),
                     expand = c(0,0)) +
  scale_x_continuous(breaks = seq(1, nrow(tabla), 10),
                     expand = c(0.01, 0.01),
                     name = "Fecha",
                     labels = etiquetas)+
  xlab("Fecha")+ylab("Índice Ultravioleta Solar")+ labs(fill="Categoría de \nexposición")+
  #ggtitle("Comportamiento temporal del ?ndice de Radiaci?n Ultravioleta Solar (IUV)\nen el Observatorio de Vigilancia Atmos
f?rica Marcapomacocha")+
  theme(axis.text.x = element_text(size = 14, color = "black", face = "bold"),
        axis.text.y = element_text(size = 14, color = "black", face = "bold",
                                   margin = unit(c(0.1, 0.1, 0.1, 0.1), "cm")),
        panel.grid.major = element_line(colour = "black"),
        panel.grid.minor = element_blank(),
        axis.title.x = element_text(vjust = 0.1),
        axis.title = element_text(size = 14, face = "bold", color = "black"),
        title = element_text(size = 15, face = "bold", color = "black"),
        plot.title = element_text(hjust = 0.001,vjust = 2),
        legend.position = "bottom",
        legend.text = element_text(size = 14, color = "black"),
        plot.margin=unit(c(0.5,0.5,0.1,0.5), 'cm')) +
  scale_fill_manual(values = c("#4eb400", "#f7e400", "#f88700", "#d8001d", "#998cff"))
IUV_MAX_DIARIO

png("IUV_grafica.png",width = 34, height = 20, res = 1200, units = "cm")
```

Figura N° 18. Índice Ultravioleta Solar Máximo



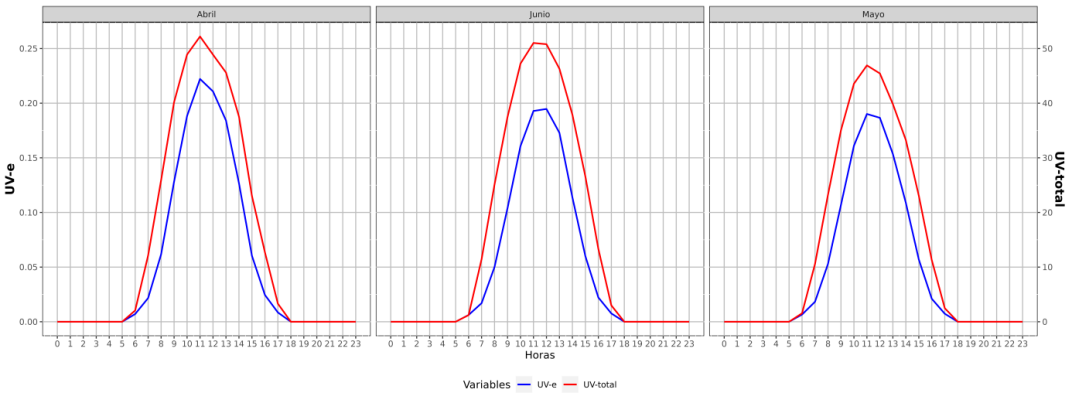
2.5.4. Radiación Ultravioleta Total y Eritemático

Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
rm (list = ls())
library(openair)
library(ggplot2)
library(tidyverse)
library(magrittr)
library(reshape2)
library(directlabels)
library(latticeExtra)
library(gridExtra)
library(grid)
```

```
 ####GRAFICA UV TOTAL Y UV ERITEMATICO ####

#Promedios horarios
datos_ruv_te = openair::timeAverage(datos_ruv,avg.time = "hour",statistic = "mean")

#library(readr)
#write.csv(datos_ruv_te,"DATOS_RUV.csv",row.names = F)
# datos_ruv_te <- read_delim("DATOS_RUV.csv",
#                      delim = ",", escape_double = FALSE, col_types = cols(date = col_datetime(format = "%d/%m/%Y %H:%M"
)),
#                      trim_ws = TRUE)
# View(datos_ruv_te)

ruv_mh = datos_ruv_te %>% select(date,UV_E_Avg,SUV5_Avg) %>% cutData(.,type="month")

ruv_mh$hour = lubridate::hour(ruv_mh$date)

ruv_mean_mes = aggregate(UV_E_Avg ~ hour + month, data = ruv_mh, mean)
ruv_mean_mes1 = aggregate(SUV5_Avg ~ hour + month, data = ruv_mh, mean)

ruv_mean_mes$SUV5_Avg = ruv_mean_mes1$SUV5_Avg

factor = max(ruv_mean_mes$UV_E_Avg)/max(ruv_mean_mes$SUV5_Avg)

#write.csv(ruv_mean_mes,"ruv_horario.csv",row.names = F)
ruv_mean_mes <- read_delim("ruv_horario.csv",
                           delim = ";", escape_double = FALSE, trim_ws = TRUE)
graf = ggplot(ruv_mean_mes,aes(x = hour)) +
  geom_line(aes(y = UV_E_Avg,color="UV-e"),size=0.8) +
  geom_line(aes(y = SUV5_Avg * 0.005,color="UV-total"), size=0.8) +
  scale_x_continuous(breaks=seq(0,23,1)) +
  scale_y_continuous(name = "UV-e",breaks = seq(0,0.3,0.05),
                     sec.axis = sec_axis(~.*(1/0.005), name = "UV-total")) +
  theme(legend.position = "bottom",
    axis.title.y = element_text(face="bold",color="black", size=13),
    axis.title.y.right = element_text(face="bold",color="black", size = 13),
    panel.background = element_rect(fill = "white",color="black"),
    panel.grid.major = element_line(color="gray"),
    strip.background = element_rect(fill = "lightgray",color="black")) +
  facet_wrap(~ month) +
  xlab("Horas")+
  scale_color_manual(name="Variables",
                     values = c("UV-e"="blue","UV-total"="red")) #+ theme_bw()

graf
ggsave(filename = "UV_e_total.png", plot =graf, width = 40, height = 15, dpi = 1000, units = "cm")
```

Figura N° 19. Radiación Ultravioleta Total y Eritemático



2.5.5. Tamaño de Partículas

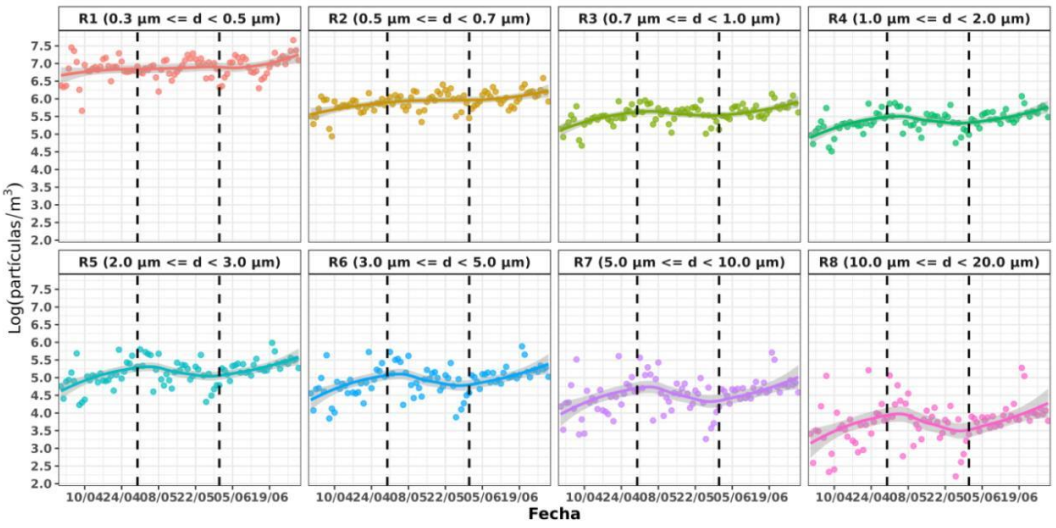
Para la ejecución del script es necesario la instalación de las siguientes librerias utilizando la función install.packages():

```
library(ggplot2)
library(tidyverse)
library(lubridate)
library(latticeExtra)
library(openair)
library(patchwork)
library(arsenal)
library(hms)
library(plotly)
```

```
###CONFIGURACIÓN DE LA CARPETA DE TRABAJO######

setwd("/home/sea/Escritorio/ELVIS/BACKUP 2023/2023/INSTRUCTIVO/BOLETIN_MARCAPOMACOCHA")
#GRAFICA PERFILADOR

perfilador<-read.csv("PERFILADOR.csv", sep = ";")
perfilador$date <- dmy_hm(perfilador$date)

perfilador2<- timeAverage(perfilador, avg.time = "day", statistic = "mean")

perfilador2 = selectByDate(perfilador2,start = "01/04/2023",end = "30/06/2023")

perfilador3<- perfilador2 %>% mutate(bin03log = log10(BIN03),
                                     bin05log = log10(BIN05),
                                     bin07log = log10(BIN07),
                                     bin1log = log10(BIN1),
                                     bin2log = log10(BIN2),
                                     bin3log = log10(BIN3),
                                     bin5log = log10(BIN5),
                                     bin10log = log10(BIN10)) %>%
  mutate(date = as.Date(date))

##exportar datos de perfilador 3 a excel para sacar los estad?sticos##
write.csv(perfilador3, "datos_perfilador_diario.csv",row.names = F)


##########
# GENERACION DE GRAFICA

data = perfilador3 %>% select(.,"date","bin03log","bin05log","bin07log","bin1log",
```

```
                    "bin2log","bin3log","bin5log","bin10log")

frame=reshape2::melt(data,id.vars="date")
frame
frame$variable = factor(frame$variable,labels = c("R1 (0.3 µm <= d < 0.5 µm)",
                                                   "R2 (0.5 µm <= d < 0.7 µm)",
                                                   "R3 (0.7 µm <= d < 1.0 µm)",
                                                   "R4 (1.0 µm <= d < 2.0 µm)",
                                                   "R5 (2.0 µm <= d < 3.0 µm)",
                                                   "R6 (3.0 µm <= d < 5.0 µm)",
                                                   "R7 (5.0 µm <= d < 10.0 µm)",
                                                   "R8 (10.0 µm <= d < 20.0 µm)"))
theme_set(theme_bw())

PF = ggplot(data = frame, aes(x=date,y = value)) +
  geom_point(aes(color = variable), size = 1.6, alpha = 0.7) +
  geom_smooth(aes(color = variable)) +
  xlab('Fecha') +
  ylab(expression(Log (partículas/m^3))) +
  geom_vline(xintercept = c(as.Date("2023-04-30"), as.Date("2023-05-31")), linetype = 2, size = 0.8)+
  scale_y_continuous(n.breaks = 10) +
  scale_x_date(date_breaks = "2 week", date_labels = "%d/%m", expand = c(0.01,0.01)) +
  theme(legend.position = "none",strip.text.x = element_text(face = "bold",size=10),
        axis.text = element_text(face = "bold",size = 9),axis.title = element_text(face = "bold",size = 12),
        strip.background.x = element_rect(color = "black",fill = "white")) +
  facet_wrap(~ variable, ncol = 4)
PF
ggsave(filename = "tamaño_particulas.jpg", plot =PF, width = 28.0, height = 14, dpi = 1000, units = "cm")
```

Figura N° 20. Tamaño de Partícula



2.6. Respaldos de scripts

2.6.1. Evaluación de prioridad de los Scripts

Como primer paso se debe evaluar la prioridad de cada script, para lo cual se hace uso de una matriz de priorización, que califica atributos y los condensa en un promedio ponderado. En esta matriz se considera el lenguaje de programación, la frecuencia de uso, complejidad, y el destino del producto. (Véase Tabla 1)

Tabla 1: Matriz de evaluación

Ítem	Fuente	Script	Descripción	Lenguaje	Frecuencia	Complejidad	Destino	Prioridad	Prioridad
1	Briefing SEA diario (REMCA)	C_AIRE_4.R	Gráficos de PM, gases, INCA para el briefing diario de la SEA	R	Diario	Moderado	Publico	3.125	**Alta**
2		Grafico_ayuda_post.R	Gráficos de ayuda para Post	R	Diario	Bajo	Interno SEA	2.375	**Media**
3		img2post.R	Gráficos para Post	R	Diario	Bajo	Interno SEA	2.375	**Media**
4	OVA Marcapomacocha	UV.R	Gráfico de comportamiento del IUV en el OVA Marcapomacocha	R	Diario	Moderado	Interno SEA	2.625	**Alta**
5	Boletín Calidad de Aire mensual	GASES_BOLETIN.R	Conversión de unidades y generación graficas	R	Mensual	Alto	Publico	3	**Alta**
6		PM_BOLETIN.R	Grafica de concentraciones de PM2.5 y PM10	R	Mensual	Moderado	Publico	2.75	**Alta**
7		NO2_AMLC.gee	Gráficos de NO2	GEE (JavaScript)	Mensual	Moderado	Publico	2.875	**Alta**
8		INCA_BOLETINES.R	Gráficos de INCA para boletín mensual de Calidad de Aire	R	Mensual	Moderado	Publico	2.75	**Alta**
9		01Met.R	Gráficos de información meteorológica para boletín mensual de Calidad de Aire	R	Mensual	Moderado	Publico	2.75	**Alta**

- El lenguaje de programación considera 3 categorías (Véase Tabla 2):

Tabla 2: puntuación de la variable lenguaje de programación

Lenguaje de programación	Puntaje
R y Python	1
Mathlab y Javascript (GEE)	2
Bash	3

- La frecuencia de uso está clasificada en 4 clases, debido a la periodicidad de emisión de productos. (Véase Tabla 3)

Tabla 3: puntuación de la variable frecuencia de uso

Frecuencia de uso	Puntaje
Semestral y/o Ocasional	1
Trimestral	2
Mensual	3
Diaria	4

- Con respecto a la complejidad, esta se debe puntuar del 1 al 5 entre muy bajo y muy alto respectivamente. Este valor es definido por el desarrollador del código. (Véase Tabla 4)

Tabla 4: puntuación de la variable complejidad

Complejidad	Puntaje
Muy Bajo	1
Bajo	2
Moderada	3
Alta	4
Muy Alta	5

- Con respecto al destino de los productos, estos se clasifican en tres. (Véase Tabla 5)

Tabla 5: puntuación de la variable destino del producto

Destino del Producto	Puntaje
Interno SEA	1
Interno SENAMHI	2
Público	3

Para el cálculo de la prioridad de cada script se reemplaza los valores obtenidos en la ecuación de prioridad, la cual se ha estimado que los pesos de cada variable se muestren tal como indica la tabla N°6, estos pesos se han establecido con el apoyo del equipo de la SEA.

$$Prioridad = \frac{1 * \text{Lenguaje de programacion} + 3 * \text{Frecuencia} + 2 * \text{Complejidad} + 2 * \text{Destino}}{8}$$

Tabla 6. Categoría de clasificación de la frecuencia de uso

Atributo	Puntaje
Frecuencia	3
Complejidad	2
Destino	2
Lenguaje	1

Finalmente, para poder categorizar el nivel de importancia que tiene cada script, se clasifica las salidas del promedio ponderado en 3 niveles: bajo, medio y alto (Véase Tabla 7), de los cuales, los Scripts que se encuentren en una prioridad media y alta, estarían considerados como elementos prioritarios para respaldar.

Tabla 7. Categoría de clasificación de promedios ponderados.

Categoría	Escala	Descripción
Bajo	0 -1.5	Mayor a 0 y menor igual a 1.5
Medio	1.5 - 2.5	Mayor a 1.5 y menor igual a 2.5
Alto	2.5 - 4	Mayor a 2.5 y menor igual a 4

2.6.2. Asignación de Carpetas

Una vez evaluado el script en función de la prioridad, se procede a asignar a una de las carpetas definidas. En caso se incorpore un nuevo producto, deberá crearse una nueva carpeta donde se pueda guardar sus respectivos Scripts.

Tabla 8. Carpetas definidas

N°	Nombre de la carpeta
1	Briefing SEA diario (REMCA)
2	OVA Marcapomacocha
3	Boletín Calidad de Aire mensual
4	Boletín OVA Marcapomacocha trimestral
5	Información satelital
6	Servicio prestado en exclusividad (DTM)
7	Preparación de Datos del Modelo WRF
8	Ploteo de salidas de modelo de dispersión

2.6.3. Actualización de copias

La actualización de copias se llevará a cabo al completar una actualización del script o un cambio significativo dentro del mismo. En caso de que se considere nuevos scripts para incluirlos dentro del respaldo se debe crear una nueva carpeta o sub carpeta con dicha temática.

Es importante recalcar que los scripts deben ser compartidos a la computadora central asignada para la ejecución de los Backups.

2.6.4. Documentación y registro de acciones de respaldo (Bitácora de cambios)

La documentación del registro de acciones de respaldo servirá para conocer la evolución de los scripts y el personal que desarrollo los cambios, así como también la fecha que fue desarrollado. Esta actualización se desarrollará mediante el llenado de la tabla 9 y solo posterior a la aprobación de los cambios por los miembros de la SEA, se trasladará hacia la computadora central para su respectivo respaldo.

Tabla 9. Bitácora de cambios

Ítem	Fuente	Script	Modificación	Personal/ Fecha	Modificación	Personal/Fecha
1	Briefing SEA diario (REMCA)	C_AIRE_4.R				
2		Grafico_ayuda_post.R				
3		img2post.R				
4	OVA Marcapomacocha	UV.R				
5	Boletín Calidad de Aire mensual	GASES_BOLETIN.R				
6		PM_BOLETIN.R				
7		NO2_AMLC.gee				
8		INCA_BOLETINES.R				
9		01Met.R				
10	Boletín OVA Marcapomacocha trimestral	calendarioVientos.R				
11		DeposAtmosferica.R				
12		perfilador_script_final.R				
13		purple_script_final.R				
14		02Met.R				
15		Ozono_Boletin_marca.R				
16		PM_Boletin_Marca.R				
17		Gráficos_UV_Marca.R				
18		IUV_Calendario_Marca				

2.6.5. Ejecución del respaldo

Rsync es la aplicación libre para sistemas tipo Unix y Microsoft Windows que ofrece transmisión eficiente de datos, que opera con datos comprimidos y cifrados. Esta herramienta necesita de los accesos SSH.

Requisitos:

- Tener instalada las herramientas Rsync en el sistema Local.
- Contar con accesos y permisos de escritura en la ubicación de destino en la red.
- Directorio local que contiene el programa y sus complementos.

2.6.5.1. Traslado de Scripts

Los scripts prioritarios que se han seleccionado para su respectivo respaldo, deben trasladarse de la computadora del usuario, lugar donde se realizaron las mejoras, hacia la computadora central a la carpeta designada para dicho proceso.

2.6.5.2. Pasos a seguir

a) Verificar el acceso a la ubicación en red donde se realiza la copia.
b) Definir las siguientes variables:

   i. $Directorio_Local: ruta global del directorio local a respaldar por ejemplo “/home/sea/SENAMHI/DirectorioS”.
   ii. $Destino_Red: Ruta global en red donde se almacenarán las copas de seguridad.

c) Ejecución de Rsync.
d) Se debe supervisar el copiado, así como cualquier archivo que esté siendo transferido, asimismo de debe verificar que no se presenten errores.
e) Verificar la copia.

3. TABLA HISTÓRICA DE CAMBIOS

Versión	Detalle de cambios
01	Versión inicial