

# Análisis de Datos Exploratorio: Accidentes viales EE. UU. (Código)

Nombre: Maricela Flores Manqui.

Profesor Cátedra: Ricardo Crespo Vergara.

Profesor Laboratorio: Claudio Álvarez Soto.

Ayudante: Jimmy Fierro Herrera.

Curso: Econometría espacial.

Fecha: 24/10/2023.



```
##INTALACION LIBRERIAS - CARGAR DATOS-----
```

```
```{r}
```

```
# Cargar la librerias
```

```
install.packages("ggplot2")
```

```
install.packages("lubridate")
```

```
install.packages("dplyr")
```

```
install.packages("vcd")
```

```
install.packages("reshape")
```

```
install.packages("leaflet")
```

```
library(dplyr)
```

```
library(vcd)
```

```
library(ggplot2)
```

```
library(reshape)
```

```
library(leaflet)
```

```
library(lubridate)
```

```
##Cargar datos
```

```
datos = read.csv("D:/ECONOMETRIA/US_Accidents_March23.csv")
```

```
##filtro por estado
```

```
filtro1 = datos[datos$State == "OR",]
```

```
##filtro para columbia
```

```
filtro2 = filtro1[filtro1$County == "Columbia",]
```



```
##seleccion de variables a usar
```

```
data <- data.frame(ID = filtro2$ID,  
  
  severity = filtro2$Severity,  
  
  latitud = filtro2$Start_Lat,  
  
  longitud = filtro2$Start_Lng,  
  
  distancia = filtro2$Distance.mi.,  
  
  hora_inicio = filtro2$Start_Time,  
  
  condado = filtro2$County,  
  
  visibilidad = filtro2$Visibility.mi.,  
  
  velocidad_viento = filtro2$Wind_Speed.mph.,  
  
  precipitacion = filtro2$Precipitation.in.,  
  
  cond_meteo = filtro2$Weather_Condition,  
  
  crossing = filtro2$Crossing,  
  
  give_way = filtro2$Give_Way,  
  
  stop = filtro2$Stop)
```

```
```
```

```
## UNIVARIADO
```

```
##MAPA-----
```

```
```{r}
```

```
# Crear un mapa
```

```
mapa <- leaflet(data) %>%
```

```
  addTiles() %>% # Agregar mosaicos base
```



```
addCircleMarkers(  
  
  lat = data$latitud,  
  
  lng = data$longitud,  
  
  label = data$severity,  
  
  popup = data$severity  
  
)  
  
# Visualizar el mapa  
  
mapa  
````  
  
###HISTOGRAMAS CONTINUAS-----  
  
````{r}  
  
# Función para generar histogramas para variables continuas  
  
histogramas_para_variables <- function(data) {  
  
  # Seleccionar solo las columnas con variables continuas  
  
  variables_continuas <- data[, sapply(data, is.numeric)]  
  
  # Crear una lista de histogramas  
  
  histogram_list <- list()  
  
  # Generar un histograma para cada variable continua  
  
  for (col in names(variables_continuas)) {  
  
    hist_data <- variables_continuas[, col]  
  
    hist_title <- paste("Histograma de", col)  
  
    # Crear el histograma y almacenarlo en la lista
```



```
hist_obj <- hist(hist_data, main = hist_title, xlab = col)

histogram_list[[col]] <- hist_obj

}

return(histogram_list)

}

# Uso de la función con datos NA tratados con media y mediana

resultados_histogramas <- histogramas_para_variables(datos_media)

resultados_histogramas <- histogramas_para_variables(datos_mediana)

...

###DENSIDAD CONTINUAS-----

```{r}

# Función para generar gráficos de densidad para variables continuas

density_plots_para_variables <- function(data) {

  # Seleccionar solo las columnas con variables continuas

  variables_continuas <- data[, sapply(data, is.numeric)]

  # Crear una lista de gráficos de densidad

  density_plot_list <- list()

  # Generar un gráfico de densidad para cada variable continua

  for (col in names(variables_continuas)) {

    density_data <- variables_continuas[, col]
```



```
density_title <- paste("Gráfico de Densidad de", col)

# Crear el gráfico de densidad y almacenarlo en la lista

density_plot_obj <- density(density_data)

# Visualizar el gráfico de densidad

plot(density_plot_obj, main = density_title, xlab = col)

density_plot_list[[col]] <- density_plot_obj

}

return(density_plot_list)

}

# Uso de la función con datos NA tratados con media y mediana

resultados_density_plots <- density_plots_para_variables(datos_media)

resultados_density_plots <- density_plots_para_variables(datos_mediana)

...

### QQPLOT CONTINUAS-----

```{r}

# Función para generar Q-Q plots para variables continuas

qqplots_para_variables <- function(data) {

  # Seleccionar solo las columnas con variables continuas

  variables_continuas <- data[, sapply(data, is.numeric)]

  # Crear una lista de Q-Q plots

  qqplot_list <- list()
```



```
# Generar un Q-Q plot para cada variable continua

for (col in names(variables_continuas)) {

  qq_data <- variables_continuas[, col]

  qq_title <- paste("Q-Q Plot de", col)

  # Crear el Q-Q plot y almacenarlo en la lista

  qqplot_obj <- qqnorm(qq_data)

  qqline(qq_data, col = 2) # Agregar línea de referencia

  # Configuración del gráfico para evitar superposición de títulos de ejes

  title(main = NULL) # Eliminar el título principal

  title(xlab = NULL, ylab = NULL) # Eliminar títulos de ejes

  mtext(paste("Cuantiles teóricos de", col), side = 1, line = 2)

  mtext(paste("Cuantiles observados de", col), side = 2, line = 2)

  qqplot_list[[col]] <- qqplot_obj

}

return(qqplot_list)

}

# Uso de la función con datos NA tratados con media y mediana

resultados_qqplots <- qqplots_para_variables(datos_media)

resultados_qqplots <- qqplots_para_variables(datos_mediana)

...

### GRAFICOS DE BARRA CATEGORICAS-----
```



```
```{r}

# Función para generar gráficos de barras para variables categóricas

barplots_para_variables_categoricas <- function(data) {

  # Seleccionar todas las columnas categóricas excepto la primera

  variables_categoricas <- data[, -1] # -1 para omitir la primera columna

  # Crear una lista de gráficos de barras

  barplot_list <- list()

  # Generar un gráfico de barras para cada variable categórica

  for (col in names(variables_categoricas)) {

    bar_data <- variables_categoricas[, col]

    bar_title <- paste("Gráfico de Barras de", col)

    # Crear el gráfico de barras y almacenarlo en la lista

    bar_freq <- table(bar_data)

    bar_width <- 0.8 # Ancho de las barras

    barplot_obj <- barplot(bar_freq, main = bar_title, ylim = c(0, max(bar_freq) * 1.2), beside = TRUE,
width = bar_width)

    # Configuración del gráfico para evitar superposición de títulos de ejes

    title(main = NULL) # Eliminar el título principal

    title(xlab = NULL, ylab = NULL) # Eliminar títulos de ejes

    # Rotar las etiquetas de las categorías en el eje X (ángulo de 45 grados)

    axis(1, at = barplot_obj, labels = names(bar_freq), las = 2, cex.axis = 0.7, srt = 45)

    # Agregar etiquetas de frecuencia sobre las barras
```





```
text(barplot_obj, bar_freq, labels = bar_freq, pos = 3, cex = 0.7)

barplot_list[[col]] <- barplot_obj

}

return(barplot_list)

}

# Uso de la función con variables categóricas

resultados_barplots <- barplots_para_variables_categoricas(categoricas)

...

###OUTLIERS-----

```{r}

# # Crear una copia de la base de datos original en una variable auxiliar

datos_media <- data.frame(severity = datos_media$severity,

                           latitud = datos_media$latitud,

                           longitud = datos_media$longitud,

                           distancia = datos_media$distancia,

                           visibilidad = datos_media$visibilidad,

                           velocidad_viento = datos_media$velocidad_viento,

                           precipitacion = datos_media$precipitacion)

datos_mediana <- data.frame(severity = datos_mediana$severity,
```



```
latitud = datos_mediana$latitud,  
longitud = datos_mediana$longitud,  
distancia = datos_mediana$distancia,  
visibilidad = datos_mediana$visibilidad,  
velocidad_viento = datos_mediana$velocidad_viento,  
precipitacion = datos_mediana$precipitacion)
```

```
datos_media_original <- datos_media  
datos_mediana_original <- datos_mediana  
  
#  
# Función para reemplazar outliers por estadísticos  
  
replace_outliers_with_statistic <- function(x, method = "median", threshold = 2) {  
  
  q1 <- quantile(x, 0.25, na.rm = TRUE)  
  q3 <- quantile(x, 0.75, na.rm = TRUE)  
  iqr <- q3 - q1  
  lower_limit <- q1 - threshold * iqr  
  upper_limit <- q3 + threshold * iqr  
  
  if (method == "min") {  
    x[x < lower_limit] <- min(x, na.rm = TRUE)  
  } else if (method == "max") {  
    x[x > upper_limit] <- max(x, na.rm = TRUE)  
  }  
}
```



```
} else if (method == "median") {  
  
  x[x < lower_limit] <- median(x, na.rm = TRUE)  
  
  x[x > upper_limit] <- median(x, na.rm = TRUE)  
  
} else if (method == "p25") {  
  
  x[x < lower_limit] <- quantile(x, 0.25, na.rm = TRUE)  
  
} else if (method == "p75") {  
  
  x[x > upper_limit] <- quantile(x, 0.75, na.rm = TRUE)  
  
}  
  
return(x)  
  
}  
  
# Lista de variables a procesar NA tratados con media(excluyendo las 4 primeras)  
  
variables_a_procesar <- colnames(datos_media)[4:ncol(datos_media)]  
  
# Iterar a través de las variables y reemplazar outliers en una nueva variable con la mediana  
  
datos_media_media <- datos_media  
  
for (var in variables_a_procesar) {  
  
  datos_media_media[[var]] <- replace_outliers_with_statistic(datos_media[[var]], method =  
  "median")  
  
}  
  
# Se realiza mismo procedimiento anterior
```



```
# Lista de variables a procesar NA tratados con media(excluyendo las 4 primeras) OTL por valor minimo
```

```
variables_a_procesar <- colnames(datos_media)[4:ncol(datos_media)]
```

```
datos_media_min <- datos_media
```

```
for (var in variables_a_procesar) {
```

```
  datos_media_min[[var]] <- replace_outliers_with_statistic(datos_media[[var]], method = "min")
```

```
}
```

```
# Lista de variables a procesar NA tratados con media(excluyendo las 4 primeras) OTL por valor maximo
```

```
variables_a_procesar <- colnames(datos_media)[4:ncol(datos_media)]
```

```
datos_media_max <- datos_media
```

```
for (var in variables_a_procesar) {
```

```
  datos_media_max[[var]] <- replace_outliers_with_statistic(datos_media[[var]], method = "max")
```

```
}
```

```
# Lista de variables a procesar NA tratados con media(excluyendo las 4 primeras) OTL por percentil25
```

```
variables_a_procesar <- colnames(datos_media)[4:ncol(datos_media)]
```

```
datos_media_p25 <- datos_media
```

```
for (var in variables_a_procesar) {
```

```
  datos_media_p25[[var]] <- replace_outliers_with_statistic(datos_media[[var]], method = "p25")
```

```
}
```



```
# Lista de variables a procesar NA tratados con media(excluyendo las 4 primeras) OTL por percentil 75
```

```
variables_a_procesar <- colnames(datos_media)[4:ncol(datos_media)]
```

```
datos_media_p75 <- datos_media
```

```
for (var in variables_a_procesar) {
```

```
  datos_media_p75[[var]] <- replace_outliers_with_statistic(datos_media[[var]], method = "p75")
```

```
}
```

```
# Lista de variables a procesar NA tratados con mediana(excluyendo las 4 primeras) OTL por valor minimo
```

```
variables_a_procesar <- colnames(datos_mediana)[4:ncol(datos_mediana)]
```

```
datos_mediana_min <- datos_mediana
```

```
for (var in variables_a_procesar) {
```

```
  datos_mediana_min[[var]] <- replace_outliers_with_statistic(datos_mediana[[var]], method = "min")
```

```
}
```

```
# Lista de variables a procesar NA tratados con mediana(excluyendo las 4 primeras) OTL por valor maximo
```

```
variables_a_procesar <- colnames(datos_mediana)[4:ncol(datos_mediana)]
```

```
datos_mediana_max <- datos_mediana
```

```
for (var in variables_a_procesar) {
```

```
  datos_mediana_max[[var]] <- replace_outliers_with_statistic(datos_mediana[[var]], method = "max")
```



```
}

# Lista de variables a procesar NA tratados con mediana(excluyendo las 4 primeras) OTL por media

variables_a_procesar <- colnames(datos_mediana)[4:ncol(datos_mediana)]

datos_mediana_media <- datos_mediana

for (var in variables_a_procesar) {

  datos_mediana_media[[var]] <- replace_outliers_with_statistic(datos_mediana[[var]], method =
"median")

}

# Lista de variables a procesar NA tratados con mediana(excluyendo las 4 primeras) OTL por
percentil 25

variables_a_procesar <- colnames(datos_mediana)[4:ncol(datos_mediana)]

datos_mediana_p25 <- datos_mediana

for (var in variables_a_procesar) {

  datos_mediana_p25[[var]] <- replace_outliers_with_statistic(datos_mediana[[var]], method =
"p25")

}

# Lista de variables a procesar NA tratados con mediana(excluyendo las 4 primeras) OTL por
percentil 75

variables_a_procesar <- colnames(datos_mediana)[4:ncol(datos_mediana)]

datos_mediana_p75 <- datos_mediana

for (var in variables_a_procesar) {

  datos_mediana_p75[[var]] <- replace_outliers_with_statistic(datos_mediana[[var]], method =
"p75")

}
```



```
}

###mejor correlacion con datos transformados

conti_uni <- datos_media

for (var in variables_a_procesar) {

  datos_media_max[[var]] <- replace_outliers_with_statistic(datos_media[[var]], method = "max")

}

...

###CORRELACION CON Y SIN OUTLIERS TRATADOS-----

```{r}

# Función para crear un heatmap de correlación

crear_heatmap_correlacion <- function(matriz_correlacion, titulo = "Heatmap de Correlación") {

  # Convertir la matriz de correlación en un formato adecuado para ggplot

  cor_matrix_melted <- melt(matriz_correlacion)

  # Crear el heatmap utilizando ggplot2

  ggplot(cor_matrix_melted, aes(X1, X2, fill = value)) +

    geom_tile() +

    geom_text(aes(label = round(value, 2)), color = "black", size = 4) +

    scale_fill_gradient2(low = "#11AAAA", mid = 'white', high = "red") +
```



```
labs(title = titulo) +  
  
theme_minimal() +  
  
theme(  
  
  axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),  
  
  axis.text.y = element_text(vjust = 0.5),  
  
  plot.title = element_text(hjust = 0.5)  
  
)  
}  
  
####MEDIA  
  
# Uso de la funcion de matriz de correlacion agregando titulo OTL sin tratar  
  
matriz_correlacion <- cor(datos_media)  
  
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL sin  
tratar")  
  
  
  
  
  
  
  
  
  
  
#Uso de la funcion de matriz de correlacion agregando titulo OTL con maximo  
  
matriz_correlacion <- cor(datos_media_max)  
  
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL max")  
  
  
  
  
  
  
  
  
  
  
#Uso de la funcion de matriz de correlacion agregando titulo OTL con minimo  
  
matriz_correlacion <- cor(datos_media_min)  
  
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL min")
```





#Uso de la funcion de matriz de correlacion agregando titulo OTL con percentil 25

```
matriz_correlacion <- cor(datos_media_p25)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL p25")
```

#Uso de la funcion de matriz de correlacion agregando titulo OTL con percentil 75

```
matriz_correlacion <- cor(datos_media_p75)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL p75")
```

#Uso de la funcion de matriz de correlacion agregando titulo OTL con media

```
matriz_correlacion <- cor(datos_media_media)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL media")
```

####MEDIANA

#Uso de la funcion de matriz de correlacion agregando titulo OTL sin tratar

```
matriz_correlacion <- cor(datos_mediana_original)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL sin tratar")
```

#Uso de la funcion de matriz de correlacion agregando titulo OTL con media

```
matriz_correlacion <- cor(datos_mediana_media)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL media")
```



```
#Uso de la funcion de matriz de correlacion agregando titulo OTL con maximo

matriz_correlacion <- cor(datos_mediana_max)

crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL max")


#Uso de la funcion de matriz de correlacion agregando titulo OTL con minimo

matriz_correlacion <- cor(datos_mediana_min)

crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL min")


#Uso de la funcion de matriz de correlacion agregando titulo OTL con percentil 25

matriz_correlacion <- cor(datos_mediana_p25)

crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL p25")


#Uso de la funcion de matriz de correlacion agregando titulo OTL con percentil 75

matriz_correlacion <- cor(datos_mediana_p75)

crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL p75")


...


### GRAFICOS CON OUTLIERS TRATADOS

###BOXPLOT-----

```{r}

# Función para generar boxplots para variables continuas (excluyendo las tres primeras variables)

boxplots_para_variables <- function(data) {
```

```
# Seleccionar solo las columnas con variables continuas (omitir las tres primeras variables)
```

```
variables_continuas <- data[, -(1:3)][, sapply(data[,-(1:3)], is.numeric)]
```

```
# Crear una lista de boxplots
```

```
boxplot_list <- list()
```

```
# Generar un boxplot para cada variable continua
```

```
for (col in names(variables_continuas)) {
```

```
  boxplot_data <- variables_continuas[, col]
```

```
  boxplot_title <- paste("Boxplot de", col)
```

```
# Crear el boxplot y almacenarlo en la lista
```

```
boxplot_obj <- boxplot(boxplot_data, main = boxplot_title, ylab = col)
```

```
boxplot_list[[col]] <- boxplot_obj
```

```
}
```

```
return(boxplot_list)
```

```
}
```

```
####son 10 ya que previo se analizaron sin el tratamiento de OTL
```

```
# Uso de la función con datos NA por media y OTL por maximo
```

```
resultados_boxplots <- boxplots_para_variables(datos_media_max)
```



```
# Uso de la función con datos NA por media y OTL por minimo

resultados_boxplots <- boxplots_para_variables(datos_media_min)

#

# Uso de la función con datos NA por media y OTL por media

# resultados_boxplots <- boxplots_para_variables(datos_media_media)

#

# Uso de la función con datos NA por media y OTL por percentil 25

# resultados_boxplots <- boxplots_para_variables(datos_media_p25)

#

# Uso de la función con datos NA por media y OTL por percentil 75

# resultados_boxplots <- boxplots_para_variables(datos_media_p75)


# Uso de la función con datos NA por mediana y OTL por maximo

resultados_boxplots <- boxplots_para_variables(datos_mediana_max)


# Uso de la función con datos NA por mediana y OTL por minimo

resultados_boxplots <- boxplots_para_variables(datos_mediana_min)


# Uso de la función con datos NA por mediana y OTL por media

# resultados_boxplots <- boxplots_para_variables(datos_mediana_media)

#
```



```
# Uso de la función con datos NA por mediana y OTL por percentil 25

# resultados_boxplots <- boxplots_para_variables(datos_mediana_p25)

#

# Uso de la función con datos NA por mediana y OTL por percentil 75

# resultados_boxplots <- boxplots_para_variables(datos_mediana_p75)

...

###DENSIDAD-----

```{r}

# Función para generar gráficos de densidad para variables continuas (excluyendo las tres primeras
variables)

density_plots_para_variables <- function(data) {

  # Seleccionar solo las columnas con variables continuas (omitir las tres primeras variables)

  variables_continuas <- data[, -(1:3)][, sapply(data[,-(1:3)], is.numeric)]

  # Crear una lista de gráficos de densidad

  density_plot_list <- list()

  # Generar un gráfico de densidad para cada variable continua

  for (col in names(variables_continuas)) {

    density_data <- variables_continuas[, col]

    density_title <- paste("Gráfico de Densidad de", col)
```



```
# Crear el gráfico de densidad y almacenarlo en la lista

density_plot_obj <- density(density_data)


# Visualizar el gráfico de densidad

plot(density_plot_obj, main = density_title, xlab = col)


density_plot_list[[col]] <- density_plot_obj
}


return(density_plot_list)
}


#Uso de la función con datos NA por media y OTL por maximo

resultados_density_plots <- density_plots_para_variables(datos_media_max)


# Uso de la función con datos NA por media y OTL por minimo

resultados_density_plots <- density_plots_para_variables(datos_media_min)


# Uso de la función con datos NA por media y OTL por media

# resultados_density_plots <- density_plots_para_variables(datos_media_media)

#

# Uso de la función con datos NA por media y OTL por percentil 25
```



```
# resultados_density_plots <- density_plots_para_variables(datos_media_p25)

#

# Uso de la función con datos NA por media y OTL por mpercentil 75

# resultados_density_plots <- density_plots_para_variables(datos_media_p75)


# Uso de la función con datos NA por mediana y OTL por maximo

resultados_density_plots <- density_plots_para_variables(datos_mediana_max)


# Uso de la función con datos NA por mediana y OTL por minimo

resultados_density_plots <- density_plots_para_variables(datos_mediana_min)


# Uso de la función con datos NA por mediana y OTL por media

# resultados_density_plots <- density_plots_para_variables(datos_mediana_media)

#

# Uso de la función con datos NA por mediana y OTL por percentil 25

# resultados_density_plots <- density_plots_para_variables(datos_mediana_p25)

#

# Uso de la función con datos NA por mediana y OTL por percentil 75

# resultados_density_plots <- density_plots_para_variables(datos_mediana_p75)

...

###TRANSFORMACION DE DATOS-----

``{r}
```



```
find_best_transformation <- function(data, variable, target, methods) {  
  
  # Inicializar variables para almacenar los resultados  
  
  best_transformation <- NULL  
  
  best_corr <- -Inf  
  
  # Calcular la correlación con la variable original  
  
  original_corr <- cor(data[[variable]], data[[target]], use = "complete.obs")  
  
  # Iterar a través de los métodos de transformación  
  
  for (method in methods) {  
  
    transformed_variable <- switch(  
  
      method,  
  
      "name_logp" = log1p(data[[variable]]),  
  
      "name_expm" = expm1(data[[variable]]),  
  
      "name_sqrt" = sqrt(data[[variable]]),  
  
      "name_sq" = data[[variable]]^2,  
  
      "iname_inv" = 1 / data[[variable]]  
  
    )  
  
    # Verificar si la desviación estándar es cero  
  
    if (sd(transformed_variable) > 0) {  
  
      # Calcular la correlación con la variable transformada  
  
      transformed_corr <- cor(transformed_variable, data[[target]], use = "complete.obs")
```





```
# Actualizar la mejor transformación si se encuentra una correlación mayor

if (transformed_corr > best_corr) {

  best_corr <- transformed_corr

  best_transformation <- method

}

}

}

# Devolver el método de transformación con la correlación más alta

return(list(transformation = best_transformation, correlation = best_corr))

}

# Crear una lista de doce bases de datos

bases_de_datos <- list(

  datos_media,

  datos_media_max,

  datos_media_media,

  datos_media_min,

  datos_media_p25,

  datos_media_p75,

  datos_mediana,

  datos_mediana_max,

  datos_mediana_media,
```



```
datos_mediana_min,
```

```
datos_mediana_p25,
```

```
datos_mediana_p75)
```

```
# Seleccionar las variables objetivo y a transformar independientes
```

```
target_variable <- 'severity'
```

```
variable_to_transform <- 'visibilidad'
```

```
# Definir los métodos de transformación disponibles
```

```
transformation_methods <- c("name_logp", "name_expm", "name_sqrt", "name_sq", "iname_inv")
```

```
# Lista para variables con observaciones igual a 0
```

```
#transformation_methods <- c("name_expm", "name_sqrt", "name_sq")
```

```
# Inicializar un vector para almacenar los resultados
```

```
resultados <- vector("list", length(bases_de_datos))
```

```
# Iterar a través de las bases de datos y calcular correlaciones y transformaciones
```

```
for (i in 1:length(bases_de_datos)) {
```

```
  continuas_transform <- bases_de_datos[[i]]
```

```
# Encontrar la mejor transformación
```



```
result <- find_best_transformation(continuas_transform, variable_to_transform, target_variable,
transformation_methods)
```

```
# Almacenar los resultados en la lista
```

```
resultados[[i]] <- list(
```

```
  BaseDatos = paste("Base de datos", i),
```

```
  MejorTransformacion = result$transformation,
```

```
  Correlacion = result$correlation
```

```
)
```

```
}
```

```
# Imprimir los resultados
```

```
for (i in 1:length(resultados)) {
```

```
  cat("Resultados para", resultados[[i]]$BaseDatos, ":\n")
```

```
  cat("Mejor transformación:", resultados[[i]]$MejorTransformacion, "\n")
```

```
  cat("Correlación con la mejor transformación:", resultados[[i]]$Correlacion, "\n\n")
```

```
}
```

```
...
```

```
### GRAFICOS CON TRANSFORMACION DE DATOS
```

```
### APLICAR TRANSFORMACION-----
```

```
```{r}
```



```
##Transformacion para datos tratados con NA por media y OTL sin tratar

transformed_md <- datos_media

transformed_md[["distancia"]] <- sqrt(transformed_md[["distancia"]])

transformed_md[["visibilidad"]] <- expm1(transformed_md[["visibilidad"]])

transformed_md[["velocidad_viento"]] <-sqrt(transformed_md[["velocidad_viento"]])

transformed_md[["precipitacion"]] <- sqrt(transformed_md[["precipitacion"]])

##Transformacion para datos tratados con NA por media y OTL con maximo

transformed_md_mx <- datos_media_max

transformed_md_mx[["distancia"]] <- sqrt(transformed_md_mx[["distancia"]])

transformed_md_mx[["visibilidad"]] <- expm1(transformed_md_mx[["visibilidad"]])

transformed_md_mx[["velocidad_viento"]] <-sqrt(transformed_md_mx[["velocidad_viento"]])

transformed_md_mx[["precipitacion"]] <-sqrt(transformed_md_mx[["precipitacion"]])

####Transformacion para datos tratados con NA por media y OTL con minimo

transformed_md_mn <- datos_media_min

transformed_md_mn[["distancia"]] <- sqrt(transformed_md_mn[["distancia"]])

transformed_md_mn[["velocidad_viento"]] <-sqrt(transformed_md_mn[["velocidad_viento"]])

####Transformacion para datos tratados con NA por media y OTL con media

transformed_md_md <- datos_media_media

transformed_md_md[["distancia"]] <- sqrt(transformed_md_md[["distancia"]])

transformed_md_md[["visibilidad"]] <- sqrt(transformed_md_md[["visibilidad"]])

transformed_md_md[["velocidad_viento"]] <-sqrt(transformed_md_md[["velocidad_viento"]])

transformed_md_md[["precipitacion"]] <-sqrt(transformed_md_md[["precipitacion"]])
```

```
####Transformacion para datos tratados con NA por media y OTL con percentil 25

transformed_md_p2 <- datos_media_p25

transformed_md_p2[["distancia"]] <- sqrt(transformed_md_p2[["distancia"]])

transformed_md_p2[["velocidad_viento"]] <-sqrt(transformed_md_p2[["velocidad_viento"]])

transformed_md_p2[["precipitacion"]] <-sqrt(transformed_md_p2[["precipitacion"]])

####Transformacion para datos tratados con NA por media y OTL con percentil 75

transformed_md_p7 <- datos_media_p75

transformed_md_p7[["distancia"]] <- sqrt(transformed_md_p7[["distancia"]])

transformed_md_p7[["visibilidad"]] <- expm1(transformed_md_p7[["visibilidad"]])

transformed_md_p7[["velocidad_viento"]] <-sqrt(transformed_md_p7[["velocidad_viento"]])


####Transformacion para datos tratados con NA por mediana y OTL sin tratar

transformed_mn <- datos_mediana

transformed_mn[["distancia"]] <- sqrt(transformed_mn[["distancia"]])

transformed_mn[["visibilidad"]] <- expm1(transformed_mn[["visibilidad"]])

transformed_mn[["velocidad_viento"]] <-sqrt(transformed_mn[["velocidad_viento"]])

transformed_mn[["precipitacion"]] <-sqrt(transformed_mn[["precipitacion"]])

####Transformacion para datos tratados con NA por mediana y OTL con maximo

transformed_mn_mx <- datos_mediana_max

transformed_mn_mx[["distancia"]] <- sqrt(transformed_mn_mx[["distancia"]])

transformed_mn_mx[["visibilidad"]] <- expm1(transformed_mn_mx[["visibilidad"]])

transformed_mn_mx[["velocidad_viento"]] <-sqrt(transformed_mn_mx[["velocidad_viento"]])
```



```
transformed_mn_mx[["precipitacion"]] <- sqrt(transformed_mn_mx[["precipitacion"]])

####Transformacion para datos tratados con NA por mediana y OTL con minimo

transformed_mn_mn <- datos_mediana_min

transformed_mn_mn[["distancia"]] <- sqrt(transformed_mn_mn[["distancia"]])

transformed_mn_mn[["velocidad_viento"]] <- sqrt(transformed_mn_mn[["velocidad_viento"]])

####Transformacion para datos tratados con NA por mediana y OTL con media

transformed_mn_md <- datos_mediana_media

transformed_mn_md[["distancia"]] <- sqrt(transformed_mn_md[["distancia"]])

transformed_mn_md[["visibilidad"]] <- expm1(transformed_mn_md[["visibilidad"]])

transformed_mn_md[["velocidad_viento"]] <- sqrt(transformed_mn_md[["velocidad_viento"]])

transformed_mn_md[["precipitacion"]] <- sqrt(transformed_mn_md[["precipitacion"]])

####Transformacion para datos tratados con NA por mediana y OTL con percentil 25

transformed_mn_p25 <- datos_mediana_p25

transformed_mn_p25[["distancia"]] <- sqrt(transformed_mn_p25[["distancia"]])

transformed_mn_p25[["velocidad_viento"]] <- sqrt(transformed_mn_p25[["velocidad_viento"]])

transformed_mn_p25[["precipitacion"]] <- sqrt(transformed_mn_p25[["precipitacion"]])

####Transformacion para datos tratados con NA por mediana y OTL con percentil 75

transformed_mn_p7 <- datos_mediana_p75

transformed_mn_p7[["distancia"]] <- sqrt(transformed_mn_p7[["distancia"]])

transformed_mn_p7[["visibilidad"]] <- expm1(transformed_mn_p7[["visibilidad"]])

transformed_mn_p7[["velocidad_viento"]] <- sqrt(transformed_mn_p7[["velocidad_viento"]])

####Transformacion para datos tratados con NA por mediana y OTL con
```



```
###mejor correlacion continuas con transformacion

conti_uni <- conti_uni

conti_uni[["distancia"]] <- sqrt(conti_uni[["distancia"]])

conti_uni[["visibilidad"]] <- expm1(conti_uni[["visibilidad"]])

conti_uni[["velocidad_viento"]] <-sqrt(conti_uni[["velocidad_viento"]])

conti_uni[["precipitacion"]] <-sqrt(conti_uni[["precipitacion"]])

```

### BOXPLOT-----

```{r}

# Función para generar boxplots para variables continuas (excluyendo las tres primeras variables)

boxplots_para_variables <- function(data) {

  # Seleccionar solo las columnas con variables continuas (omitir las tres primeras variables)

  variables_continuas <- data[, -(1:3)][, sapply(data[,-(1:3)], is.numeric)]

  # Crear una lista de boxplots

  boxplot_list <- list()

  # Generar un boxplot para cada variable continua

  for (col in names(variables_continuas)) {

    boxplot_data <- variables_continuas[, col]
```



```
boxplot_title <- paste("Boxplot de", col)

# Crear el boxplot y almacenarlo en la lista

boxplot_obj <- boxplot(boxplot_data, main = boxplot_title, ylab = col)

boxplot_list[[col]] <- boxplot_obj

}

return(boxplot_list)

}

# Uso de la función con datos NA por media y OTL sin tratar

resultados_boxplots <- boxplots_para_variables(transformed_md)

## Uso de la función con datos NA por media y OTL con media

# resultados_boxplots <- boxplots_para_variables(transformed_md_md)

## Uso de la función con datos NA por media y OTL con minimo

resultados_boxplots <- boxplots_para_variables(transformed_md_mn)

## Uso de la función con datos NA por media y OTL con maximo

resultados_boxplots <- boxplots_para_variables(transformed_md_mx)

## Uso de la función con datos NA por media y OTL con percentil 25
```





```
# resultados_boxplots <- boxplots_para_variables(transformed_md_p2)

#

## Uso de la función con datos NA por media y OTL con percentil75

# resultados_boxplots <- boxplots_para_variables(transformed_md_p7)


## Uso de la función con datos NA por mediana y OTL sin tratar

# resultados_boxplots <- boxplots_para_variables(transformed_mn)


## Uso de la función con datos NA por mediana y OTL con media

# resultados_boxplots <- boxplots_para_variables(transformed_mn_md)


## Uso de la función con datos NA por mediana y OTL con minimo

# resultados_boxplots <- boxplots_para_variables(transformed_mn_mn)


## Uso de la función con datos NA por mediana y OTL con maximo

# resultados_boxplots <- boxplots_para_variables(transformed_mn_mx)


## Uso de la función con datos NA por mediana y OTL con percentil 25

# resultados_boxplots <- boxplots_para_variables(transformed_mn_p25)

#

## Uso de la función con datos NA por mediana y OTL con percentil 75

# resultados_boxplots <- boxplots_para_variables(transformed_mn_p7)
```



```

...

### DENSIDAD-----

```{r}

# Función para generar gráficos de densidad para variables continuas (excluyendo las tres primeras
variables)

density_plots_para_variables <- function(data) {

  # Seleccionar solo las columnas con variables continuas (omitir las tres primeras variables)

  variables_continuas <- data[, -(1:3)][, sapply(data[,-(1:3)], is.numeric)]

  # Crear una lista de gráficos de densidad

  density_plot_list <- list()

  # Generar un gráfico de densidad para cada variable continua

  for (col in names(variables_continuas)) {

    density_data <- variables_continuas[, col]

    density_title <- paste("Gráfico de Densidad de", col)

    # Crear el gráfico de densidad y almacenarlo en la lista

    density_plot_obj <- density(density_data)

    # Visualizar el gráfico de densidad
  }
}
```



```
plot(density_plot_obj, main = density_title, xlab = col)

density_plot_list[[col]] <- density_plot_obj

}

return(density_plot_list)

}

# Uso de la función con datos tratados con NA por media y OTL sin tratar
resultados_density_plots <- density_plots_para_variables(transformed_md)

# Uso de la función con datos tratados con NA por media y OTL con media
# resultados_density_plots <- density_plots_para_variables(transformed_md_md)

# Uso de la función con datos tratados con NA por media y OTL con minimo
resultados_density_plots <- density_plots_para_variables(transformed_md_mn)

# Uso de la función con datos tratados con NA por media y OTL con maximo
resultados_density_plots <- density_plots_para_variables(transformed_md_mx)

# Uso de la función con datos tratados con NA por media y OTL con percentil 25
# resultados_density_plots <- density_plots_para_variables(transformed_md_p2)

#
```



```
# Uso de la función con datos tratados con NA por media y OTL con percentil 75

# resultados_density_plots <- density_plots_para_variables(transformed_md_p7)


# Uso de la función con datos tratados con NA por mediana y OTL csin tratra

# resultados_density_plots <- density_plots_para_variables(transformed_mn)


# Uso de la función con datos tratados con NA por mediana y OTL con media

# resultados_density_plots <- density_plots_para_variables(transformed_mn_md)


# Uso de la función con datos tratados con NA por mediana y OTL con minimo

# resultados_density_plots <- density_plots_para_variables(transformed_mn_mn)

#

# Uso de la función con datos tratados con NA por mediana y OTL con maximo

# resultados_density_plots <- density_plots_para_variables(transformed_mn_mx)


# Uso de la función con datos tratados con NA por mediana y OTL con percentil 25

# resultados_density_plots <- density_plots_para_variables(transformed_mn_p25)

#

# Uso de la función con datos tratados con NA por mediana y OTL con percentil 75

# resultados_density_plots <- density_plots_para_variables(transformed_mn_p7)

...

##BIVARIADO
```



```
####CORRELACION CONTINUAS CN TRANSFO-----

```{r}

# Función para crear un heatmap de correlación

crear_heatmap_correlacion <- function(matriz_correlacion, titulo = "Heatmap de Correlación") {

  # Convertir la matriz de correlación en un formato adecuado para ggplot

  cor_matrix_melted <- melt(matriz_correlacion)


  # Crear el heatmap utilizando ggplot2

  ggplot(cor_matrix_melted, aes(X1, X2, fill = value)) +

    geom_tile() +

    geom_text(aes(label = round(value, 2)), color = "black", size = 4) +

    scale_fill_gradient2(low = "#11AAAA", mid = 'white', high = "red") +

    labs(title = titulo) +

    theme_minimal() +

    theme(

      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),

      axis.text.y = element_text(vjust = 0.5),

      plot.title = element_text(hjust = 0.5)

    )

  }

####MEDIA

# Uso de la funcion de matriz de correlacion agregando titulo OTL sin tratar
```



```
matriz_correlacion <- cor(transformed_md)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL sin  
tratar")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por maximo
```

```
matriz_correlacion <- cor(transformed_md_mx)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL max")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por minimo
```

```
matriz_correlacion <- cor(transformed_md_mn)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL min")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por per 25
```

```
matriz_correlacion <- cor(transformed_md_p2)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL p25")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por per 75
```

```
matriz_correlacion <- cor(transformed_md_p7)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL p75")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por media
```

```
matriz_correlacion <- cor(transformed_md_md)
```



```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA media y OTL media")
```

```
####MEDIANA
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL sin tratar
```

```
matriz_correlacion <- cor(transformed_mn)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL sin  
tratar")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por media
```

```
matriz_correlacion <- cor(transformed_mn_md)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL  
media")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por maxima
```

```
matriz_correlacion <- cor(transformed_mn_mx)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL max")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por minimo
```

```
matriz_correlacion <- cor(transformed_mn_mn)
```

```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL min")
```

```
# Uso de la funcion de matriz de correlacion agregando titulo OTL por per 25
```

```
matriz_correlacion <- cor(transformed_mn_p25)
```



```
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL p25")

# Uso de la funcion de matriz de correlacion agregando titulo OTL por per 75

matriz_correlacion <- cor(transformed_mn_p7)

crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación NA mediana y OTL p75")

...

###MATRIZ DE CORRELACION CATEGORICAS—

```{r}

# Función para calcular la tabla de contingencia y realizar la prueba de chi-cuadrado

calculate_chi_squared <- function(data, categorical_var) {

  # Crear la tabla de contingencia

  tabla_contingencia <- table(data[, categorical_var], data$severity)

  # Realizar la prueba de chi-cuadrado

  chi_square_result <- chisq.test(tabla_contingencia)

  # Mostrar la tabla de contingencia

  print(tabla_contingencia)

  # Mostrar los resultados de la prueba de chi-cuadrado

  print(chi_square_result)

  return(chi_square_result)

}
```





```
# Función para iterar a través de las variables categóricas y calcular chi-cuadrado
```

```
calculate_chi_squared_for_categoricals <- function(data) {
```

```
  categorical_vars <- setdiff(names(data), "severity")
```

```
  results <- list()
```

```
  for (var in categorical_vars) {
```

```
    cat(paste("Variable Categórica:", var, "\n"))
```

```
    results[[var]] <- calculate_chi_squared(data, var)
```

```
    cat("\n")
```

```
  }
```

```
  return(results)
```

```
}
```

```
results <- calculate_chi_squared_for_categoricals(categoricas)
```

```
---
```

```
### DENSIDAD CONTINUAS-----
```

```
```{r}
```



# Función para crear gráficos de densidad para variables numéricas (excluyendo las tres primeras variables)

```
create_density_plots <- function(data) {  
  
  numeric_vars <- sapply(data, is.numeric)  
  
  numeric_data <- data[, numeric_vars]  
  
  # Excluir las tres primeras variables  
  
  numeric_data <- numeric_data[, -c(1, 2, 3)]  
  
  # Crear un vector para almacenar los resultados  
  
  density_plot_results <- list()  
  
  for (var in colnames(numeric_data)) {  
  
    plot <- ggplot(data, aes_string(x = var, fill = "severity")) +  
  
      geom_density(alpha = 0.5) +  
  
      labs(title = paste("Densidad de", var, "por Severity"))  
  
    print(plot)  
  
  }  
  
}  
  
create_density_plots(transformed_md) # NA por Media y OTL sin tratar  
  
create_density_plots(transformed_md_mn) # NA por Media y OTL con minimo  
  
create_density_plots(transformed_md_mx) # NA por Media y OTL maximo  
  
create_density_plots(transformed_mn) # NA por Mediaana y OTL sin tratar  
  
create_density_plots(transformed_mn_mn) # NA por Mediana y OTL minimo  
  
create_density_plots(transformed_mn_mx) # NA por Mediana y OTL maximo
```



```
```\n\n###QQPLOT CONTINUAS-----\n\n```\n{r}\n\n# Función para crear gráficos QQ para variables numéricas (excluyendo las tres primeras variables)\n\ncreate_qq_plots <- function(data) {\n\n  numeric_vars <- sapply(data, is.numeric)\n\n  numeric_data <- data[, numeric_vars]\n\n  # Excluir las tres primeras variables\n\n  numeric_data <- numeric_data[, -c(1, 2, 3)]\n\n  qq_plot_results <- list()\n\n  for (var in colnames(numeric_data)) {\n\n    qq_data <- numeric_data[, var]\n\n    qq_plot <- qqnorm(qq_data, main = paste("QQ Plot for", var))\n\n    qq_line <- qqline(qq_data)\n\n    qq_plot_results[[var]] <- list(qq_plot, qq_line)\n\n  }\n\n  return(qq_plot_results)\n\n}\n\nqq_plots <- create_qq_plots(transformed_md) # datos con NA por media y OTL sin tratar\n\nqq_plots <- create_qq_plots(transformed_md_mn)# datos con NA por media y OTL con minimo
```



```
qq_plots <- create_qq_plots(transformed_md_mx)# datos con NA por media y OTL con maximo

qq_plots <- create_qq_plots(transformed_mn)# datos con NA por media y OTL sin tratar

qq_plots <- create_qq_plots(transformed_mn_mn)# datos con NA por media y OTL con minimo

qq_plots <- create_qq_plots(transformed_mn_mx)# datos con NA por media y OTL con maximo
```

```
```
```

```
### BARRAS CATEGORICAS-----
```

```
```{r}
```

```
# Crear un gráfico de barras con horas del día y severidad en el eje X
```

```
ggplot(categoricas, aes(x = hour_of_day, fill = factor(severity))) +
```

```
  geom_bar(position = "dodge") +
```

```
  labs(title = "Accidentes por hora del dia y severidad",
```

```
    x = "hora del dia y severidad",
```

```
    y = "Number of Accidents",
```

```
    fill = "Severity") +
```

```
  scale_fill_brewer(palette = "Set1") +
```

```
  theme_minimal() +
```

```
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
# Crear un gráfico de barras con condicion meteorologica y severidad en el eje X
```

```
ggplot(categoricas, aes(x = cond_meteo, fill = factor(severity))) +
```

```
  geom_bar(position = "dodge") +
```



```
labs(title = "Accidentes por condicion meteorologica y severidad",
```

```
  x = "condicion meteorologica y severidad",
```

```
  y = "Number of Accidents",
```

```
  fill = "Severity") +
```

```
scale_fill_brewer(palette = "Set1") +
```

```
theme_minimal() +
```

```
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

# Crear un gráfico de barras con presencia de cruce y severidad y severidad en el eje X

```
ggplot(categoricas, aes(x = crossing, fill = factor(severity))) +
```

```
geom_bar(position = "dodge") +
```

```
labs(title = "Accidentes por presencia de cruce y severidad y severidad",
```

```
  x = "Hour of Day and Severity",
```

```
  y = "presencia de cruce y severidad y severidad",
```

```
  fill = "Severity") +
```

```
scale_fill_brewer(palette = "Set1") +
```

```
theme_minimal() +
```

```
theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

# Crear un gráfico de barras con presencia de ceda el paso y severidad en el eje X

```
ggplot(categoricas, aes(x = give_way, fill = factor(severity))) +
```

```
geom_bar(position = "dodge") +
```



```
labs(title = "Accidentes por presencia de ceda el paso y severidad",  
      x = "Hour of Day and Severity",  
      y = "presencia de ceda el paso y severidad",  
      fill = "Severity") +  
scale_fill_brewer(palette = "Set1") +  
theme_minimal() +  
theme(axis.text.x = element_text(angle = 45, hjust = 1))  
  
# Crear un gráfico de barras con presencia de señalética pare y severidad en el eje X  
ggplot(categoricas, aes(x = stop, fill = factor(severity))) +  
geom_bar(position = "dodge") +  
labs(title = "Accidentes por presencia de señalética pare y severidad",  
      x = "Hour of Day and Severity",  
      y = "presencia de señalética pare y severidad",  
      fill = "Severity") +  
scale_fill_brewer(palette = "Set1") +  
theme_minimal() +  
theme(axis.text.x = element_text(angle = 45, hjust = 1))  
...  
  
### HOT ENCONDIG  
```{r}  
  
# Función para realizar one-hot encoding
```



```
one_hot_encoding <- function(data, variable) {  
  
  # Asegurarse de que la variable sea de tipo factor  
  
  data[[variable]] <- as.factor(data[[variable]])  
  
  # Realizar one-hot encoding utilizando la función 'model.matrix'  
  
  one_hot_encoded <- model.matrix(~ data[[variable]] - 1, data = data)  
  
  # Obtener los nombres de las categorías originales  
  
  categories <- levels(data[[variable]])  
  
  # Cambiar los nombres de las columnas generadas  
  
  colnames(one_hot_encoded) <- paste(variable, categories, sep = "_")  
  
  # Eliminar la variable categórica original  
  
  data[[variable]] <- NULL  
  
  # Combinar el conjunto de datos original con las nuevas columnas one-hot encoded  
  
  data <- cbind(data, one_hot_encoded)  
  
  
  return(data)  
}  
  
  
# Realizar one-hot encoding en la variable "crossing"  
  
categ_1 <- one_hot_encoding(categoricas, "crossing")  
  
# Almacenar unicamente variable con presencia de cruce y ID  
  
categ_1 <- data.frame(ID = categ_1$ID, crossing_true = categ_1$crossing_True)  
  
# Realizar one-hot encoding en la variable "give_way"
```



```
categ_2 <- one_hot_encoding(categoricas, "give_way")

# Almacenar unicamente variable con presencia de ceda el paso y ID

categ_2 <- data.frame(ID = categ_2$ID, give_way_true = categ_2$give_way_True)

# Realizar one-hot encoding en la variable "stop"

categ_3 <- one_hot_encoding(categoricas, "stop")

# Almacenar unicamente variable con presencia de pare y ID

categ_3 <- data.frame(ID = categ_3$ID, stop_true = categ_3$stop_True)

...

###MAPPING

```{r}

# Definir el mapeo "cond_meteo"

mapeo <- c("Clear" = 2,

           "Cloudy" = 1,

           "Cloudy / Windy" = 1,

           "Fair" = 2,

           "Fog" = 1,

           "Haze" = 1,

           "Heavy Rain" = 3,

           "Snow" = 4,

           "T-Storm" = 3,

           "Thunder in the Vicinity" = 3,
```





"Light Freezing Fog" = 1,

"Light Rain" = "3",

"Light Rain / Windy" = 3,

"Light Snow" = 4,

"Mostly Cloudy" = 1,

"N/A Precipitation" = 1,

"Overcast" = 1,

"Partly Cloudy" = 1,

"Rain" = 2,

"Scattered Clouds" = 1,

"Wintry Mix" = 1)

# Realizar el mapeo y reemplazar los valores en el campo

```
map <- categoricas %>%
```

```
  mutate(cond_meteo = ifelse(cond_meteo %in% names(mapeo), mapeo[cond_meteo], cond_meteo))
```

```
map$cond_meteo <- as.numeric(map$cond_meteo)
```

# Definir el mapeo variable "hour\_of\_day"

```
mapeo1 <- c("0" = "0",
```

```
  "1" = "0",
```



"2" = "0",

"3" = "0",

"4" = "0",

"5" = "0",

"6" = "1",

"7" = "1",

"8" = "1",

"9" = "1",

"10" = "1",

"11" = "1",

"12" = "2",

"13" = "2",

"14" = "2",

"15" = "2",

"16" = "2",

"17" = "2",

"18" = "2",

"19" = "2",

"20" = "2",

"21" = "3",

"22" = "3",

"23" = "3")



```
# Realizar el mapeo y reemplazar los valores en el campo
```

```
map <- map %>%
```

```
  mutate(hour_of_day = ifelse(hour_of_day %in% names(mapeo1), mapeo1[hour_of_day],  
    hour_of_day))
```

```
map$hour_of_day <- as.numeric(map$hour_of_day)
```

```
# Almacenar unicamente variables codificadas de condicion climatica, hora del dia y ID
```

```
map <- data.frame(ID = map$ID,  
  hour_of_day = map$hour_of_day,  
  cond_meteo = map$cond_meteo)
```

```
#Union de data procedente de hot encoding y mapping
```

```
categ_uni <- merge(map, categ_1, by="ID", all=FALSE)
```

```
categ_uni <- merge(categ_uni, categ_2, by="ID", all=FALSE)
```

```
categ_uni <- merge(categ_uni, categ_3, by="ID", all=FALSE)
```

```
#Eliminar variables categoricas de la union anterior
```

```
categ_uni <- data_frame(ID = categ_uni$ID,  
  hour_of_day = categ_uni$hour_of_day,  
  cond_meteo = categ_uni$cond_meteo,  
  crossing_true = categ_uni$crossing_true,  
  stop_true = categ_uni$stop_true,
```



```
give_way_true = categ_uni$give_way_true)

...

##UNION MEJOR CORRELACION CONTINUAS Y CATEGORICAS TRATADAS

```{r}

##union de variables categoricas codificadas y continuas

data_posible <- merge(categ_uni, conti_uni, by="ID", all=FALSE)

# Función para crear un heatmap de correlación

crear_heatmap_correlacion <- function(matriz_correlacion, titulo = "Heatmap de Correlación") {

  # Convertir la matriz de correlación en un formato adecuado para ggplot

  cor_matrix_melted <- melt(matriz_correlacion)

  # Crear el heatmap utilizando ggplot2

  ggplot(cor_matrix_melted, aes(X1, X2, fill = value)) +

    geom_tile() +

    geom_text(aes(label = round(value, 2)), color = "black", size = 4) +

    scale_fill_gradient2(low = "#11AAAA", mid = 'white', high = "red") +

    labs(title = titulo) +

    theme_minimal() +

    theme(

      axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),

      axis.text.y = element_text(vjust = 0.5),

      plot.title = element_text(hjust = 0.5)
```



```
)  
}  
  
# Correlacion de variables dependiente y independientes continuas y categoricas codificadasomite el  
campo ID  
  
matriz_correlacion <- cor(data_posible[, -1])  
  
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación dependiente v/s  
independientes")  
  
# Crear la matriz de correlación excluyendo los valores NA  
  
matriz_correlacion <- cor(data_posible[, -1], use = "pairwise.complete.obs")  
  
# Crear el heatmap de correlación con la matriz resultante  
  
crear_heatmap_correlacion(matriz_correlacion, "Heatmap de Correlación dependiente v/s  
independientes")  
---
```

