

Análisis de Accidentes de Tráfico en Madrid (2019-2023)

Pau Garcia Orengo, Vanessa Machordom Torres, Ditmar Estrada Bernuy

2025-11-01

Contents

1	Configuración del Entorno	1
1.1	Instalación de Paquetes	1
1.2	Carga de Librerías	1
2	Carga y Preparación de Datos	2
2.1	Importación del Dataset	2
2.2	Exploración Inicial del Dataset	2
2.2.1	Estructura de Variables	2
2.2.2	Resumen Estadístico de Variables Numéricas	3
2.2.3	Variables Categóricas Principales	4
2.2.4	Distribución de la Variable Objetivo: Lesividad	4
2.2.5	Calidad de Datos: Variables con Datos Faltantes	5
2.3	Limpieza de Datos	6
2.3.1	Normalización de Valores Faltantes	6
2.3.2	Ánalisis de Valores Faltantes	6
2.4	Ingeniería de Características	8
2.4.1	Variables Temporales	8
2.4.2	Franjas Horarias y Períodos	8
2.4.3	Variables Meteorológicas Derivadas	8
2.4.4	Clasificación de Gravedad de Lesiones	9
2.5	Resumen del Dataset Procesado	9
3	Ánalisis General del Conjunto de Datos	11
3.1	Distribución por Tipo de Accidente	11
3.2	Distribución por Distrito	12
3.3	Distribución por Tipo de Vehículo	13
3.4	Gravedad según Tipo de Accidente	14
3.5	Distribución Temporal Básica	15
3.5.1	Accidentes por Día de la Semana	15
3.5.2	Accidentes por Mes	16

4 Análisis Meteorológico	17
4.1 Accidentes por Condición Meteorológica (Totales)	17
4.2 Media de Accidentes Diarios por Condición Meteorológica	18
4.3 Accidentes según Categoría de Temperatura	19
4.4 Accidentes con y sin Lluvia	20
5 Análisis Temporal	21
5.1 Distribución por Año	21
5.2 Distribución por Hora del Día	22
6 Análisis Temporal Avanzado	23
6.1 Configuración para Análisis de Días Festivos	23
6.2 Media de Accidentes por Día de la Semana	23
6.3 Obtención de Días Festivos	26
6.4 Clasificación de Días Festivos vs No Festivos	27
6.5 Comparativa: Festivo vs No Festivo por Día de la Semana	27
6.6 Comparativa General: Festivo vs No Festivo	29
6.7 Resumen Estadístico: Impacto de Días Festivos	30
7 Análisis Espacial	32
7.1 Preparación de Datos Geoespaciales	32
7.2 Mapa de Calor de Densidad de Accidentes	33
7.3 Mapa de Burbujas por Distrito	34
8 Análisis de Factores Humanos	36
8.1 Distribución por Rango de Edad y Sexo	36
9 Análisis Multivariado	36
9.1 Matriz de Correlaciones entre Variables Meteorológicas	36
10 Análisis Avanzado: Clustering Geográfico	37
10.1 Identificación de Zonas de Alta Concentración de Accidentes	37
11 Conclusiones	38
11.1 Resumen de Hallazgos Clave	38
11.2 Ruta Metodológica Seguida	38

1 Configuración del Entorno

1.1 Instalación de Paquetes

```
# Ejecutar solo si los paquetes no están instalados
install.packages("corrplot")
install.packages("sf")
install.packages("leaflet")
install.packages(c("sf", "dplyr", "ggplot2", "scales"))
```

1.2 Carga de Librerías

```
# Manipulación y visualización de datos
library(dplyr)
library(ggplot2)
library(lubridate)
library(stringr)
library(tidyr)
library(scales)

# Análisis espacial
library(sf)
library(leaflet)

# Análisis estadístico
library(corrplot)
library(cluster)
library(factoextra)

library(knitr)
library(kableExtra)
```

2 Carga y Preparación de Datos

2.1 Importación del Dataset

```
# Cargar el archivo CSV con datos de accidentes y meteorología
accidents_clean_data <- read.csv("../data/processed/accidentes_madrid_con_weather.csv",
                                    header = TRUE,
                                    sep = ",")
```

2.2 Exploración Inicial del Dataset

```
# Dimensiones del dataset
cat("### RESUMEN GENERAL DEL DATASET ###\n\n")

## ### RESUMEN GENERAL DEL DATASET ###

cat(sprintf(" Dimensiones: %s observaciones x %s variables\n",
            format(nrow(accidents_clean_data), big.mark = ","),
            ncol(accidents_clean_data)))
```

```
## Dimensions: 221,910 observaciones × 17 variables
```

```
cat(sprintf(" Período: %s a %s\n",
            min(as.Date(accidents_clean_data$time)),
            max(as.Date(accidents_clean_data$time))))
```

```
## Período: 2019-01-01 a 2023-12-31
```

2.2.1 Estructura de Variables

```
# Crear tabla resumen de estructura de variables
estructura_df <- data.frame(
  Variable = names(accidents_clean_data),
  Tipo = sapply(accidents_clean_data, function(x) class(x)[1]),
  NAs = sapply(accidents_clean_data, function(x) sum(is.na(x))),
  Porcentaje_NA = sapply(accidents_clean_data,
    function(x) round(mean(is.na(x)) * 100, 1)),
  Valores_unicos = sapply(accidents_clean_data,
    function(x) length(unique(x)))
)

# Ordenar por porcentaje de NAs descendente
estructura_df <- estructura_df %>%
  arrange(desc(Porcentaje_NA))

# Mostrar tabla con formato profesional
estructura_df %>%
  kable(format = "latex",
        booktabs = TRUE,
        caption = "Estructura y características de las variables del dataset",
        col.names = c("Variable", "Tipo", "NAs", "Pct NA", "Valores únicos"),
        align = c("l", "c", "r", "r", "r")) %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"),
                font_size = 8,
                full_width = FALSE) %>%
  row_spec(0, bold = TRUE)
```

Table 1: Estructura y características de las variables del dataset

	Variable	Tipo	NAs	Pct NA	Valores únicos
wx_temperature	wx_temperature	numeric	92427	41.7	414
wx_wind_speed	wx_wind_speed	numeric	92427	41.7	452
wx_precipitation	wx_precipitation	numeric	92427	41.7	67
localizacion	localizacion	character	0	0.0	50365
distrito	distrito	character	0	0.0	22
tipo_accidente	tipo_accidente	character	0	0.0	15
estado_meteorol_gico	estado_meteorol_gico	character	0	0.0	9
tipo_vehiculo	tipo_vehiculo	character	0	0.0	42
tipo_persona	tipo_persona	character	0	0.0	5
rango_edad	rango_edad	character	0	0.0	19
sexo	sexo	character	0	0.0	4
lesividad	lesividad	character	0	0.0	11
coordenada_x_utm	coordenada_x_utm	integer	43	0.0	82066
coordenada_y_utm	coordenada_y_utm	numeric	43	0.0	89105
positiva_alcohol	positiva_alcohol	character	0	0.0	4
positiva_droga	positiva_droga	integer	5	0.0	3
time	time	character	0	0.0	85127

2.2.2 Resumen Estadístico de Variables Numéricas

```
# Seleccionar solo variables numéricas
vars_numericas <- accidents_clean_data %>%
  select(where(is.numeric))

# Crear resumen estadístico
if(ncol(vars_numericas) > 0) {
  resumen_num <- data.frame(
    Variable = names(vars_numericas),
    Media = sapply(vars_numericas, mean, na.rm = TRUE),
    Mediana = sapply(vars_numericas, median, na.rm = TRUE),
    Desv_Est = sapply(vars_numericas, sd, na.rm = TRUE),
    Minimo = sapply(vars_numericas, min, na.rm = TRUE),
    Maximo = sapply(vars_numericas, max, na.rm = TRUE)
  )

  resumen_num %>%
    mutate(across(where(is.numeric), ~round(., 2))) %>%
    kable(format = "latex",
      booktabs = TRUE,
      caption = "Estadísticas descriptivas de variables numéricas",
      col.names = c("Variable", "Media", "Mediana", "Desv. Est.",
                  "Mínimo", "Máximo"),
      align = c("l", rep("r", 5))) %>%
    kable_styling(latex_options = c("striped", "HOLD_position"),
      font_size = 9,
      full_width = FALSE) %>%
    row_spec(0, bold = TRUE)
}
```

Table 2: Estadísticas descriptivas de variables numéricas

	Variable	Media	Mediana	Desv. Est.	Mínimo	Máximo
wx_temperature	wx_temperature	1.369000e+01	1.210000e+01	8.210000e+00	-6.4	3.790000e+01
wx_wind_speed	wx_wind_speed	2.140000e+00	1.850000e+00	1.300000e+00	0.0	9.450000e+00
wx_precipitation	wx_precipitation	6.000000e-02	0.000000e+00	5.000000e-01	0.0	2.740000e+01
coordenada_x_utm	coordenada_x_utm	4.106837e+08	4.415069e+08	1.080808e+08	0.0	4.554839e+08
coordenada_y_utm	coordenada_y_utm	4.160790e+09	4.474482e+09	1.086421e+09	0.0	4.495569e+09
positiva_droga	positiva_droga	0.000000e+00	0.000000e+00	6.000000e-02	0.0	1.000000e+00

2.2.3 Variables Categóricas Principales

```
# Identificar variables categóricas con pocos valores únicos
vars_categoricas <- accidents_clean_data %>%
  select(where(is.character)) %>%
  select(where(~length(unique(.)) <= 20))

# Crear resumen de frecuencias para variables categóricas clave
if(ncol(vars_categoricas) > 0) {
  cat("\n**Variables categóricas identificadas:**\n")
  cat(paste("-", names(vars_categoricas), collapse = "\n"))
}

## 
## **Variables categóricas identificadas:** 
## - tipo_accidente
## - estado_meteorológico
## - tipo_persona
## - rango_edad
## - sexo
## - lesividad
## - positiva_alcohol
```

2.2.4 Distribución de la Variable Objetivo: Lesividad

```
# Análisis detallado de lesividad
lesividad_freq <- accidents_clean_data %>%
  filter(!is.na(lesividad)) %>%
  count(lesividad) %>%
  mutate(Porcentaje = round(n / sum(n) * 100, 2)) %>%
  arrange(desc(n))

lesividad_freq %>%
  kable(format = "latex",
        booktabs = TRUE,
        caption = "Distribución de tipos de lesividad en accidentes",
        col.names = c("Tipo de Lesividad", "Frecuencia", "Porcentaje"),
        align = c("l", "r", "r")) %>%
  kable_styling(latex_options = c("striped", "HOLD_position"),
               font_size = 9,
               full_width = FALSE) %>%
```

```
row_spec(0, bold = TRUE) %>%
column_spec(1, width = "8cm")
```

Table 3: Distribución de tipos de lesividad en accidentes

Tipo de Lesividad	Frecuencia	Porcentaje
No se registró	100566	45.32
Sin asistencia sanitaria	66555	29.99
Asistencia sanitaria sólo en el lugar del accidente	28707	12.94
Ingreso inferior o igual a 24 horas	8768	3.95
Atención en urgencias sin posterior ingreso	6357	2.86
Asistencia sanitaria inmediata en centro de salud o mutua	5478	2.47
Asistencia sanitaria ambulatoria con posterioridad	2757	1.24
Ingreso superior a 24 horas	2562	1.15
Fallecido 24 horas	144	0.06
Se desconoce	11	0.00
	5	0.00

2.2.5 Calidad de Datos: Variables con Datos Faltantes

```
# Resumen visual de datos faltantes
missing_data <- estructura_df %>%
  filter(Porcentaje_NA > 0) %>%
  select(Variable, NAs, Porcentaje_NA) %>%
  arrange(desc(Porcentaje_NA))

if(nrow(missing_data) > 0) {
  missing_data %>%
    head(15) %>%
    kable(format = "latex",
          booktabs = TRUE,
          caption = "Top 15 variables con mayor proporción de datos faltantes",
          col.names = c("Variable", "Cantidad NAs", "Pct Faltante"),
          align = c("l", "r", "r")) %>%
    kable_styling(latex_options = c("striped", "HOLD_position"),
                  font_size = 9,
                  full_width = FALSE) %>%
    row_spec(0, bold = TRUE)
} else {
  cat(" No se encontraron datos faltantes en el dataset.\n")
}
```

Table 4: Top 15 variables con mayor proporción de datos faltantes

	Variable	Cantidad NAs	Pct Faltante
	wx_temperature	wx_temperature	92427
	wx_wind_speed	wx_wind_speed	92427
	wx_precipitation	wx_precipitation	92427

2.3 Limpieza de Datos

2.3.1 Normalización de Valores Faltantes

```
# Estandarizar valores faltantes y limpiar espacios en blanco
accidents_clean_data <- accidents_clean_data %>%
  # Eliminar espacios en blanco en columnas de texto
  mutate(across(where(is.character), ~str_trim(.))) %>%

# Reemplazar valores problemáticos por NA
mutate(across(where(is.character),
  ~case_when(
    . %in% c("", "NA", "N/A", "No se registró", "No se registro",
    "null", "NULL", "Sin dato", "Desconocido") ~ NA_character_,
    TRUE ~ .
  )))

```

2.3.2 Análisis de Valores Faltantes

```
# Calcular porcentaje de valores faltantes por columna
missing_summary <- accidents_clean_data %>%
  summarise(across(everything(), ~mean(is.na(.)) * 100)) %>%
  pivot_longer(cols = everything(), names_to = "columna", values_to = "porcentaje_na") %>%
  arrange(desc(porcentaje_na))

# Mostrar resumen de valores faltantes con formato ajustado
missing_summary %>%
  kable(format = "latex", booktabs = TRUE,
        digits = 2,
        col.names = c("Columna", "% NA"),
        caption = "Porcentaje de valores faltantes por columna") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"),
                font_size = 9,
                full_width = FALSE)

```

Table 5: Porcentaje de valores faltantes por columna

Columna	% NA
lesividad	45.32
wx_temperature	41.65
wx_wind_speed	41.65
wx_precipitation	41.65
rango_edad	11.01
estado_meteorol_gico	10.70
sexo	10.66
tipo_vehiculo	0.43
positiva_alcohol	0.35
coordenada_x_utm	0.02
coordenada_y_utm	0.02
distrito	0.01
tipo_accidente	0.00
tipo_persona	0.00
positiva_droga	0.00
localizacion	0.00
time	0.00

```
# Contar valores únicos en columnas categóricas
valores_unicos <- sapply(accidents_clean_data[, sapply(accidents_clean_data, is.character)],
                           function(x) length(unique(x)))

data.frame(
  Columna = names(valores_unicos),
  Valores_Unicos = valores_unicos
) %>%
  kable(format = "latex", booktabs = TRUE,
        caption = "Valores únicos en columnas categóricas") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"),
                font_size = 9,
                full_width = FALSE)
```

Table 6: Valores únicos en columnas categóricas

	Columna	Valores_Unicos
localizacion	localizacion	50365
distrito	distrito	22
tipo_accidente	tipo_accidente	14
estado_meteorol_gico	estado_meteorol_gico	8
tipo_vehiculo	tipo_vehiculo	41
tipo_persona	tipo_persona	4
rango_edad	rango_edad	18
sexo	sexo	3
lesividad	lesividad	10
positiva_alcohol	positiva_alcohol	3
time	time	85127

2.4 Ingeniería de Características

2.4.1 Variables Temporales

```
# Extraer componentes de fecha y tiempo
accidents_clean_data$fecha <- as.Date(accidents_clean_data$time)
accidents_clean_data$anio <- year(accidents_clean_data$fecha)
accidents_clean_data$hora <- hour(accidents_clean_data$time)
accidents_clean_data$dia_semana <- wday(accidents_clean_data$time, label = TRUE, abbr = FALSE)
accidents_clean_data$mes <- month(accidents_clean_data$time, label = TRUE)

# Crear variable de estación del año
accidents_clean_data$estacion <- case_when(
  accidents_clean_data$mes %in% c("diciembre", "enero", "febrero") ~ "Invierno",
  accidents_clean_data$mes %in% c("marzo", "abril", "mayo") ~ "Primavera",
  accidents_clean_data$mes %in% c("junio", "julio", "agosto") ~ "Verano",
  accidents_clean_data$mes %in% c("septiembre", "octubre", "noviembre") ~ "Otoño"
)
```

2.4.2 Franjas Horarias y Períodos

```
# Clasificar accidentes por franja horaria
accidents_clean_data$franja_horaria <- case_when(
  accidents_clean_data$hora >= 6 & accidents_clean_data$hora < 12 ~ "Mañana",
  accidents_clean_data$hora >= 12 & accidents_clean_data$hora < 18 ~ "Tarde",
  accidents_clean_data$hora >= 18 & accidents_clean_data$hora < 24 ~ "Noche",
  TRUE ~ "Madrugada"
)

# Clasificar fin de semana vs entre semana
accidents_clean_data$es_fin_de_semana <- ifelse(
  accidents_clean_data$dia_semana %in% c("sábado", "domingo"), "Sí", "No"
)
```

2.4.3 Variables Meteorológicas Derivadas

```
# Categorizar temperatura en rangos
accidents_clean_data$categoría_temp <- case_when(
  accidents_clean_data$wx_temperature < 5 ~
    "Frío (< 5°C)",
  accidents_clean_data$wx_temperature >= 5 &
    accidents_clean_data$wx_temperature < 15 ~
    "Templado (5-15°C)",
  accidents_clean_data$wx_temperature >= 15 &
    accidents_clean_data$wx_temperature < 25 ~
    "Cálido (15-25°C)",
  accidents_clean_data$wx_temperature >= 25 ~
    "Caluroso (> 25°C)"
)

# Crear indicador binario de precipitación
accidents_clean_data$hubo_lluvia <- ifelse(
```

```

accidents_clean_data$wx_precipitation > 0,
"Sí",
"No"
)

```

2.4.4 Clasificación de Gravedad de Lesiones

```

# Crear variable de gravedad simplificada
accidents_clean_data <- accidents_clean_data %>%
  mutate(gravedad_binaria = case_when(
    # Casos leves
    grepl("sólo en el lugar|Sin asistencia|ambulatoria",
          lesividad,
          ignore.case = TRUE) ~ "Leve",
    
    # Casos graves
    grepl("Ingreso|urgencias|centro de salud|mutua|Fallecido",
          lesividad,
          ignore.case = TRUE) ~ "Grave",
    
    # Casos sin información
    lesividad == "Se desconoce" |
      is.na(lesividad) ~ "Desconocido",
    
    # Otros casos
    TRUE ~ "Otro"
  ))

```

2.5 Resumen del Dataset Procesado

```

# Estructura general del dataset
cat("Estructura del dataset procesado:\n")

```

```
## Estructura del dataset procesado:
```

```

cat(sprintf("Dimensiones: %d filas x %d columnas\n",
            nrow(accidents_clean_data),
            ncol(accidents_clean_data)))

```

```
## Dimensiones: 221910 filas x 28 columnas
```

```

# Estadísticas descriptivas - solo mostrar las primeras variables numéricas
cat("\nEstadísticas descriptivas (primeras variables numéricas):\n")

```

```

## 
## Estadísticas descriptivas (primeras variables numéricas):

```

```

numericas_idx <- which(sapply(accidents_clean_data, is.numeric))[1:min(5, sum(sapply(accidents_clean_data,
summary(accidents_clean_data[, numericas_idx]))

```

```

## wx_temperature  wx_wind_speed  wx_precipitation  coordenada_x_utm
## Min.   :-6.40    Min.   :0.000    Min.   : 0.000    Min.   :      0
## 1st Qu.: 7.30    1st Qu.:1.150    1st Qu.: 0.000    1st Qu.:439335254
## Median :12.10    Median :1.850    Median : 0.000    Median :441506946
## Mean   :13.69    Mean   :2.145    Mean   : 0.061    Mean   :410683724
## 3rd Qu.:19.80    3rd Qu.:2.900    3rd Qu.: 0.000    3rd Qu.:443925885
## Max.   :37.90    Max.   :9.450    Max.   :27.400    Max.   :455483886
## NA's    :92427   NA's    :92427   NA's    :92427   NA's    :43

## coordenada_y_utm
## Min.   :0.000e+00
## 1st Qu.:4.471e+09
## Median :4.474e+09
## Mean   :4.161e+09
## 3rd Qu.:4.477e+09
## Max.   :4.496e+09
## NA's    :43

```

```

# Conteo de valores faltantes por variable
na_counts <- sapply(accidents_clean_data, function(x) sum(is.na(x)))
data.frame(
  Variable = names(na_counts),
  NA_Count = na_counts
) %>%
  filter(NA_Count > 0) %>%
  kable(format = "latex", booktabs = TRUE,
        caption = "Conteo de valores faltantes",
        col.names = c("Variable", "NAs")) %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"),
                font_size = 8,
                full_width = FALSE)

```

Table 7: Conteo de valores faltantes

	Variable	NAs
wx_temperature	wx_temperature	92427
wx_wind_speed	wx_wind_speed	92427
wx_precipitation	wx_precipitation	92427
distrito	distrito	13
tipo_accidente	tipo_accidente	9
estado_meteorol_gico	estado_meteorol_gico	23750
tipo_vehiculo	tipo_vehiculo	949
tipo_persona	tipo_persona	8
rango_edad	rango_edad	24425
sexo	sexo	23664
lesividad	lesividad	100571
coordenada_x_utm	coordenada_x_utm	43
coordenada_y_utm	coordenada_y_utm	43
positiva_alcohol	positiva_alcohol	787
positiva_droga	positiva_droga	5
estacion	estacion	221910
categoria_temp	categoria_temp	92427
hubo_lluvia	hubo_lluvia	92427

```

# Primeras filas del dataset
head(accidents_clean_data, 5) %>%
  select(1:6) %>%

```

```

kable(format = "latex", booktabs = TRUE,
      caption = "Primeras filas del dataset (primeras 6 columnas)") %>%
kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"),
              font_size = 7,
              full_width = FALSE)

```

Table 8: Primeras filas del dataset (primeras 6 columnas)

wx_temperature	wx_wind_speed	wx_precipitation	localizacion	distrito	tipo_accidente
-1.0	1.02	0	CALL. ALBERTO AGUILERA, 1	CENTRO	Colisión lateral
-1.0	1.02	0	CALL. ALBERTO AGUILERA, 1	CENTRO	Colisión lateral
1.9	0.73	0	PASEO. SANTA MARIA DE LA CABEZA / PLAZA. ELIPTICA	CARABANCHEL	Alcance
1.9	0.73	0	PASEO. SANTA MARIA DE LA CABEZA / PLAZA. ELIPTICA	CARABANCHEL	Alcance
1.9	0.73	0	PASEO. SANTA MARIA DE LA CABEZA / PLAZA. ELIPTICA	CARABANCHEL	Alcance

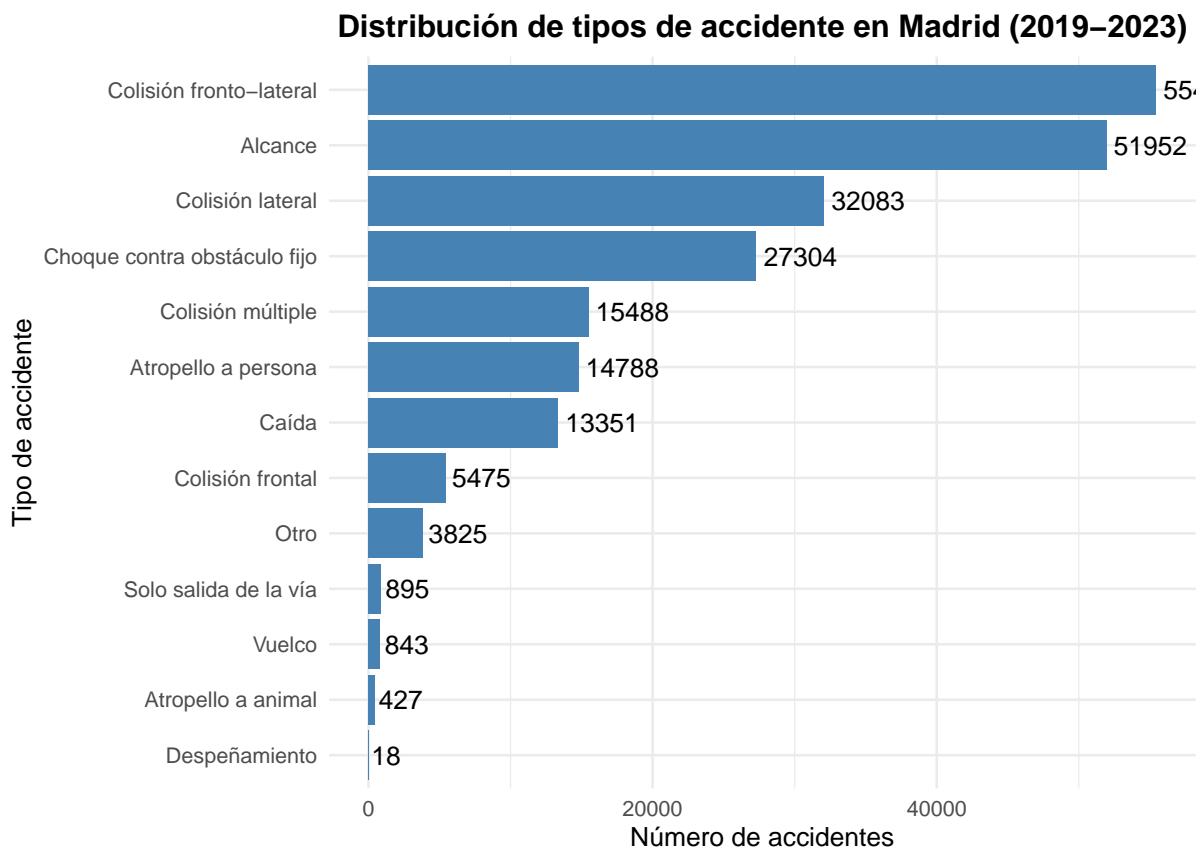
3 Análisis General del Conjunto de Datos

3.1 Distribución por Tipo de Accidente

```

accidents_clean_data %>%
  filter(!is.na(tipo_accidente)) %>%
  count(tipo_accidente) %>%
  mutate(tipo_accidente = reorder(tipo_accidente, n)) %>%
  ggplot(aes(x = tipo_accidente, y = n)) +
  geom_col(fill = "steelblue") +
  geom_text(aes(label = n), hjust = -0.1) +
  coord_flip() +
  labs(
    title = "Distribución de tipos de accidente en Madrid (2019-2023)",
    x = "Tipo de accidente",
    y = "Número de accidentes"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.text.y = element_text(size = 9)
  )

```



3.2 Distribución por Distrito

```
# Calcular total de accidentes por distrito
accidentes_por_distrito <- accidents_clean_data %>%
  filter(!is.na(distrito)) %>%
  group_by(distrito) %>%
  summarise(total_accidentes = n()) %>%
  arrange(desc(total_accidentes))

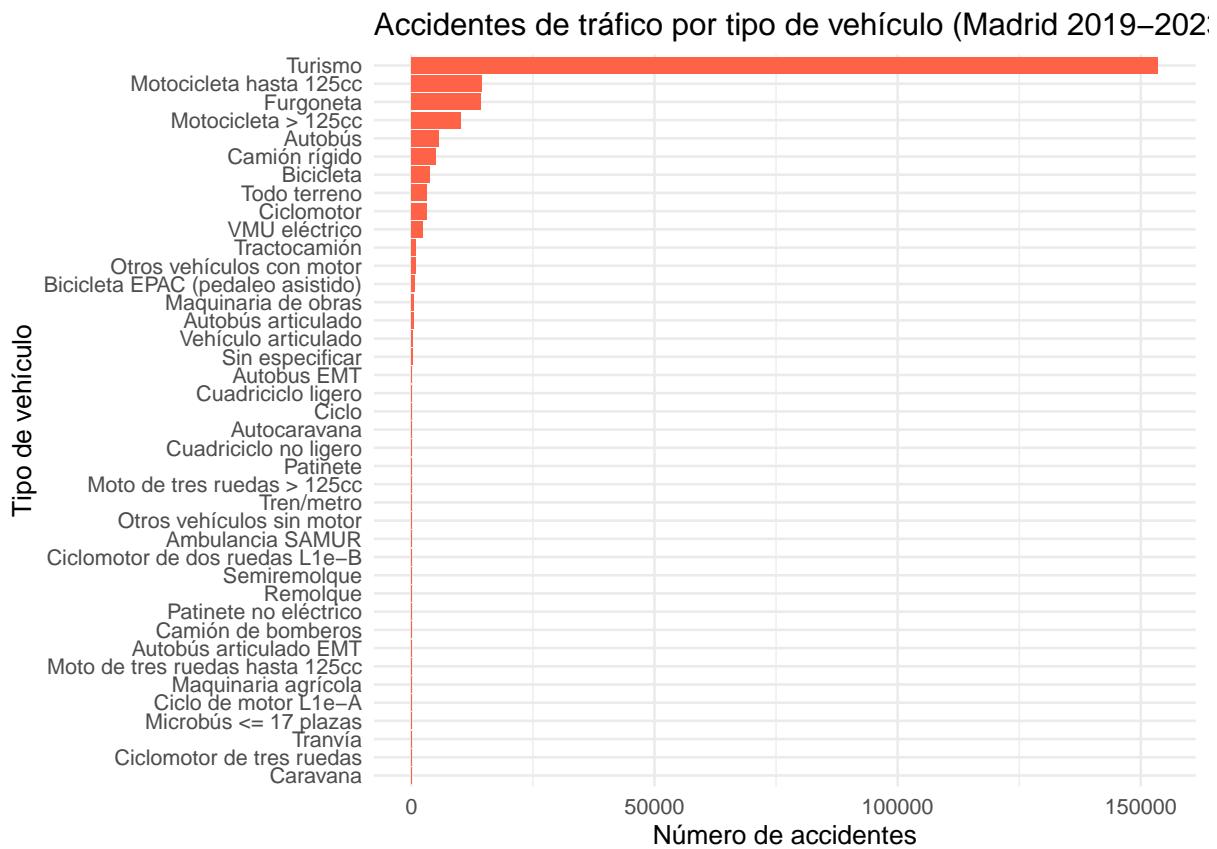
# Visualización
ggplot(accidentes_por_distrito,
       aes(x = reorder(distrito, total_accidentes), y = total_accidentes, fill = distrito)) +
  geom_col() +
  geom_text(aes(label = total_accidentes), vjust = 0.5, hjust = -0.1) +
  coord_flip() +
  labs(
    title = "Accidentes de tráfico por distrito (Madrid 2019–2023)",
    x = "Distrito",
    y = "Número de accidentes"
) +
  theme_minimal() +
  theme(legend.position = "none")
```



3.3 Distribución por Tipo de Vehículo

```
# Calcular accidentes por tipo de vehículo
accidentes_por_tipo_vehiculo <- accidents_clean_data %>%
  filter(!is.na(tipo_vehiculo)) %>%
  group_by(tipo_vehiculo) %>%
  summarise(total_accidentes = n()) %>%
  arrange(desc(total_accidentes))

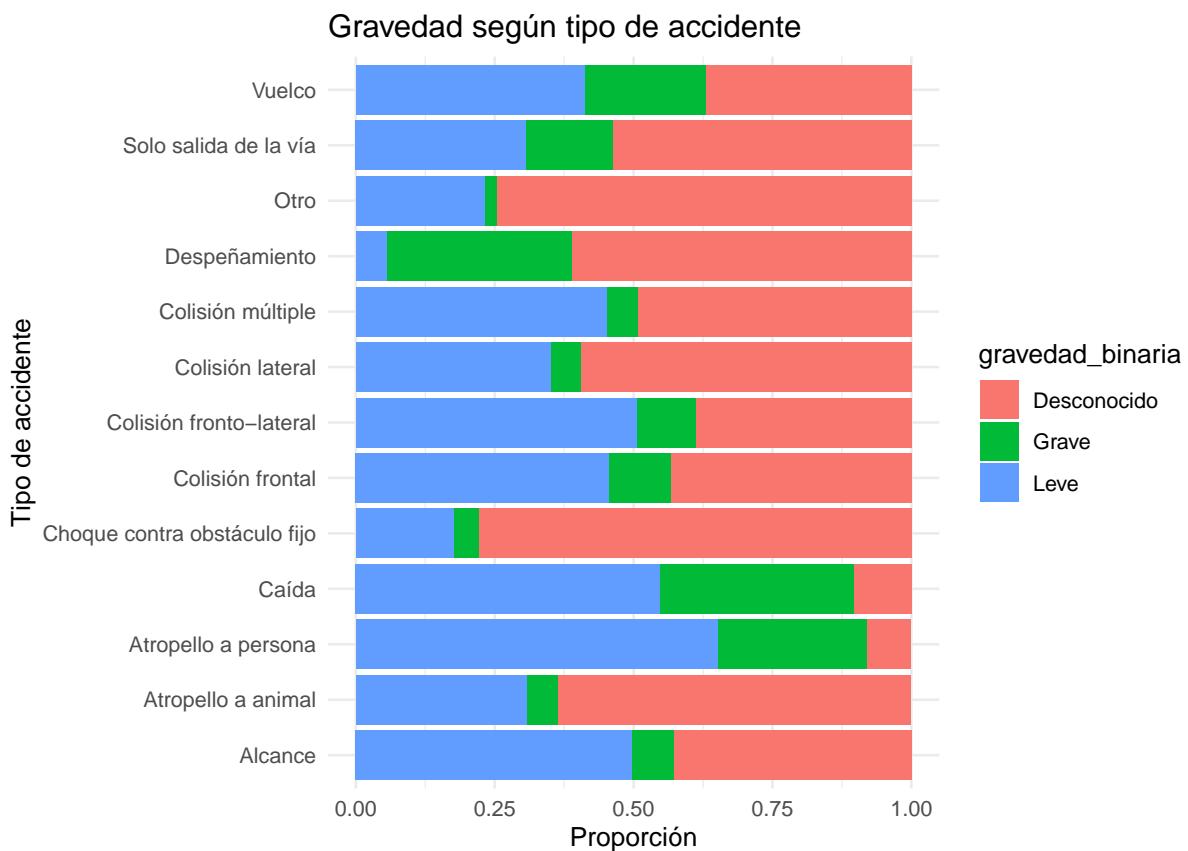
# Visualización
ggplot(accidentes_por_tipo_vehiculo,
       aes(x = reorder(tipo_vehiculo, total_accidentes), y = total_accidentes)) +
  geom_col(fill = "tomato") +
  coord_flip() +
  labs(
    title = "Accidentes de tráfico por tipo de vehículo (Madrid 2019–2023)",
    x = "Tipo de vehículo",
    y = "Número de accidentes"
) +
  theme_minimal()
```



3.4 Gravedad según Tipo de Accidente

```
# Filtrar datos sin NA en tipo de accidente
accidents_clean_data_sin_na <- accidents_clean_data %>%
  filter(!is.na(tipo_accidente))

# Visualización de proporción de gravedad por tipo de accidente
ggplot(accidents_clean_data_sin_na, aes(x = tipo_accidente, fill = gravedad_binaria)) +
  geom_bar(position = "fill") +
  coord_flip() +
  labs(
    title = "Gravedad según tipo de accidente",
    x = "Tipo de accidente",
    y = "Proporción"
  ) +
  theme_minimal()
```

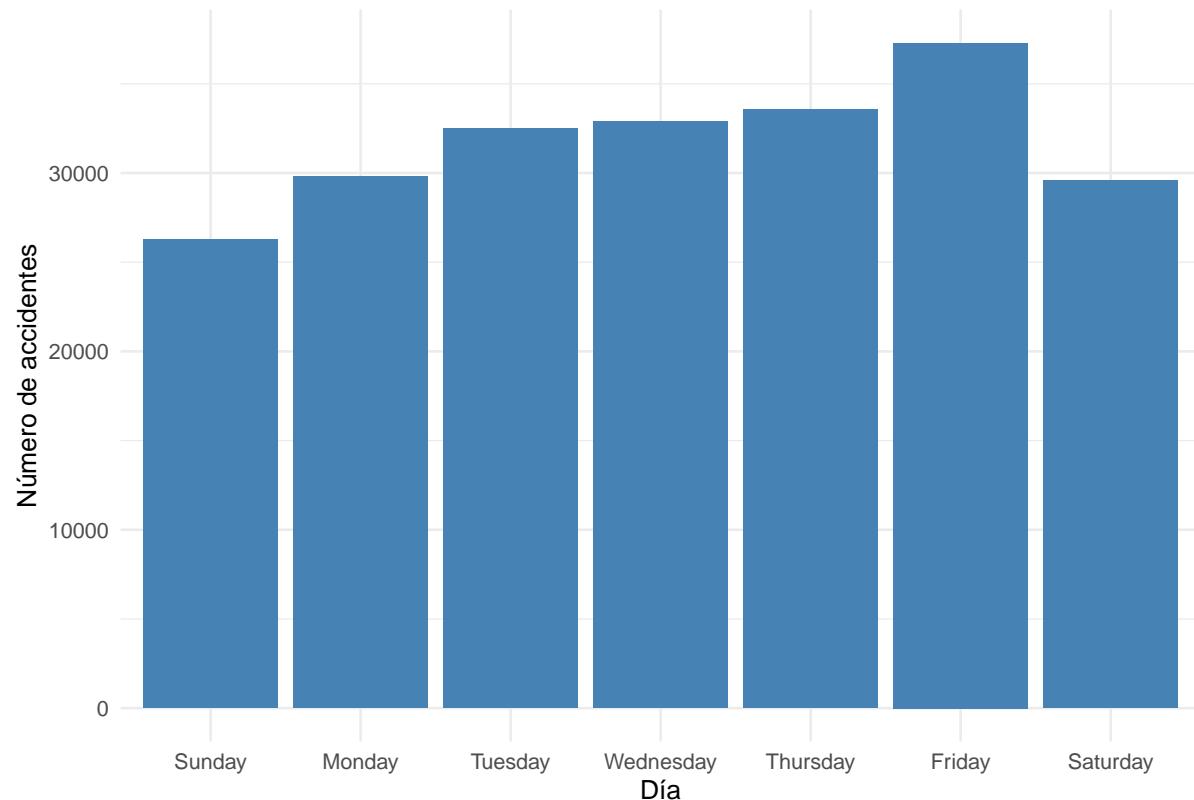


3.5 Distribución Temporal Básica

3.5.1 Accidentes por Día de la Semana

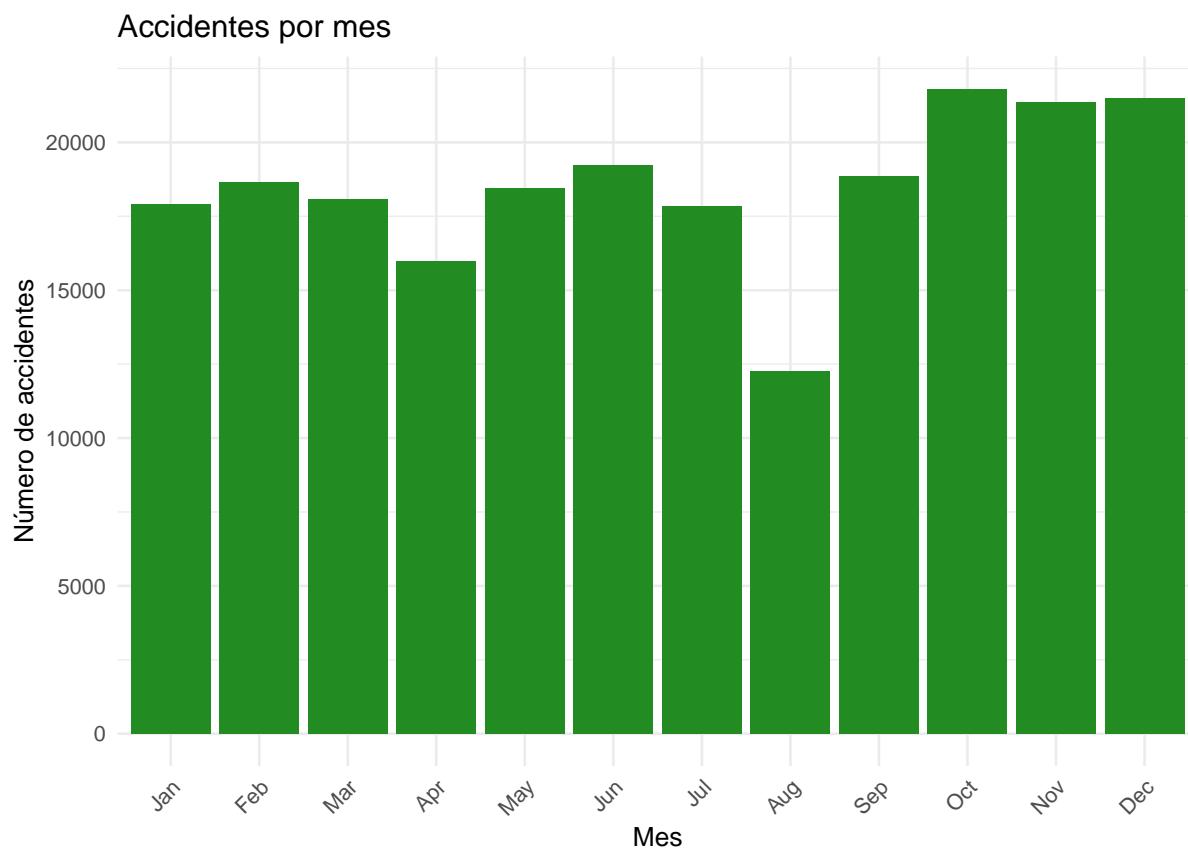
```
ggplot(accidents_clean_data, aes(x = dia_semana)) +
  geom_bar(fill = "steelblue") +
  labs(
    title = "Accidentes por día de la semana",
    x = "Día",
    y = "Número de accidentes"
  ) +
  theme_minimal()
```

Accidentes por día de la semana



3.5.2 Accidentes por Mes

```
ggplot(accidents_clean_data, aes(x = mes)) +
  geom_bar(fill = "forestgreen") +
  labs(
    title = "Accidentes por mes",
    x = "Mes",
    y = "Número de accidentes"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

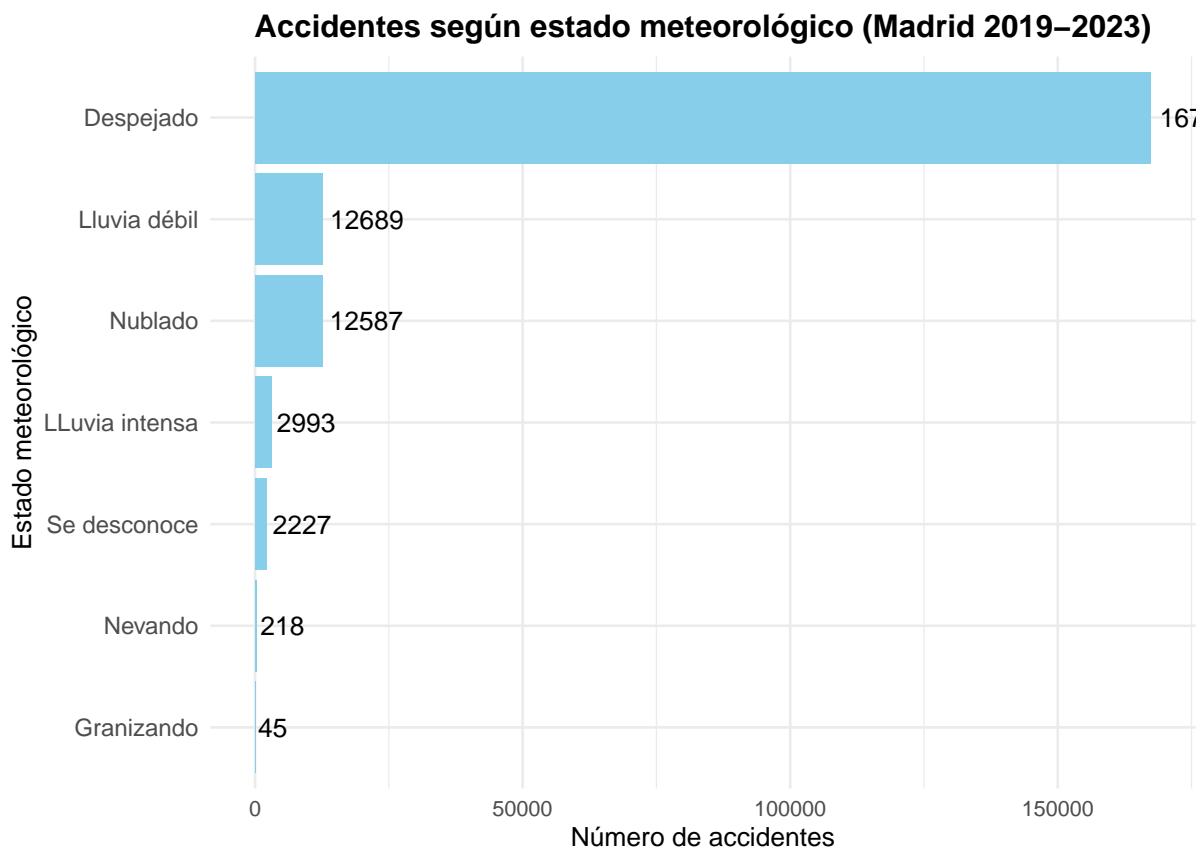


4 Análisis Meteorológico

4.1 Accidentes por Condición Meteorológica (Totales)

```
# Contar accidentes por estado meteorológico
accidentes_clima <- accidents_clean_data %>%
  filter(!is.na(estado_meteorológico)) %>%
  count(estado_meteorológico) %>%
  arrange(desc(n))

# Visualización
ggplot(accidentes_clima, aes(x = reorder(estado_meteorológico, n), y = n)) +
  geom_col(fill = "skyblue") +
  geom_text(aes(label = n), hjust = -0.1) +
  coord_flip() +
  labs(
    title = "Accidentes según estado meteorológico (Madrid 2019-2023)",
    x = "Estado meteorológico",
    y = "Número de accidentes"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.text.y = element_text(size = 10)
  )
```

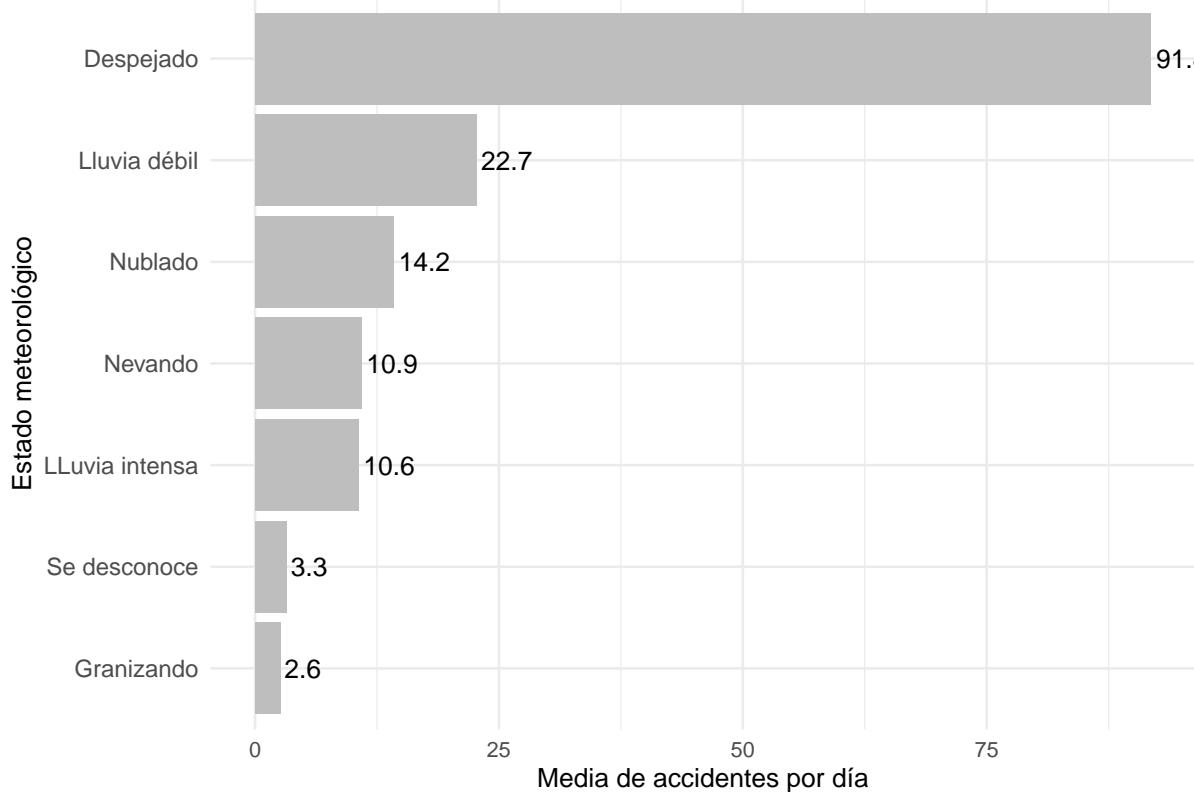


4.2 Media de Accidentes Diarios por Condición Meteorológica

```
# Calcular media de accidentes por día según condición meteorológica
accidentes_clima <- accidents_clean_data %>%
  filter(!is.na(estado_meteorológico)) %>%
  mutate(fecha = as.Date(time)) %>%
  group_by(estado_meteorológico, fecha) %>%
  summarise(accidentes_dia = n(), .groups = 'drop') %>%
  group_by(estado_meteorológico) %>%
  summarise(media = mean(accidentes_dia)) %>%
  arrange(desc(media))

# Visualización
ggplot(accidentes_clima, aes(x = reorder(estado_meteorológico, media), y = media)) +
  geom_col(fill = "grey") +
  geom_text(aes(label = round(media, 1)), hjust = -0.1) +
  coord_flip() +
  labs(
    title = "Media de accidentes por día según estado meteorológico (Madrid 2019–2023)",
    x = "Estado meteorológico",
    y = "Media de accidentes por día"
  ) +
  theme_minimal() +
  theme(
    plot.title = element_text(face = "bold", hjust = 0.5),
    axis.text.y = element_text(size = 10)
  )
```

Media de accidentes por día según estado meteorológico (Madrid 2019–2

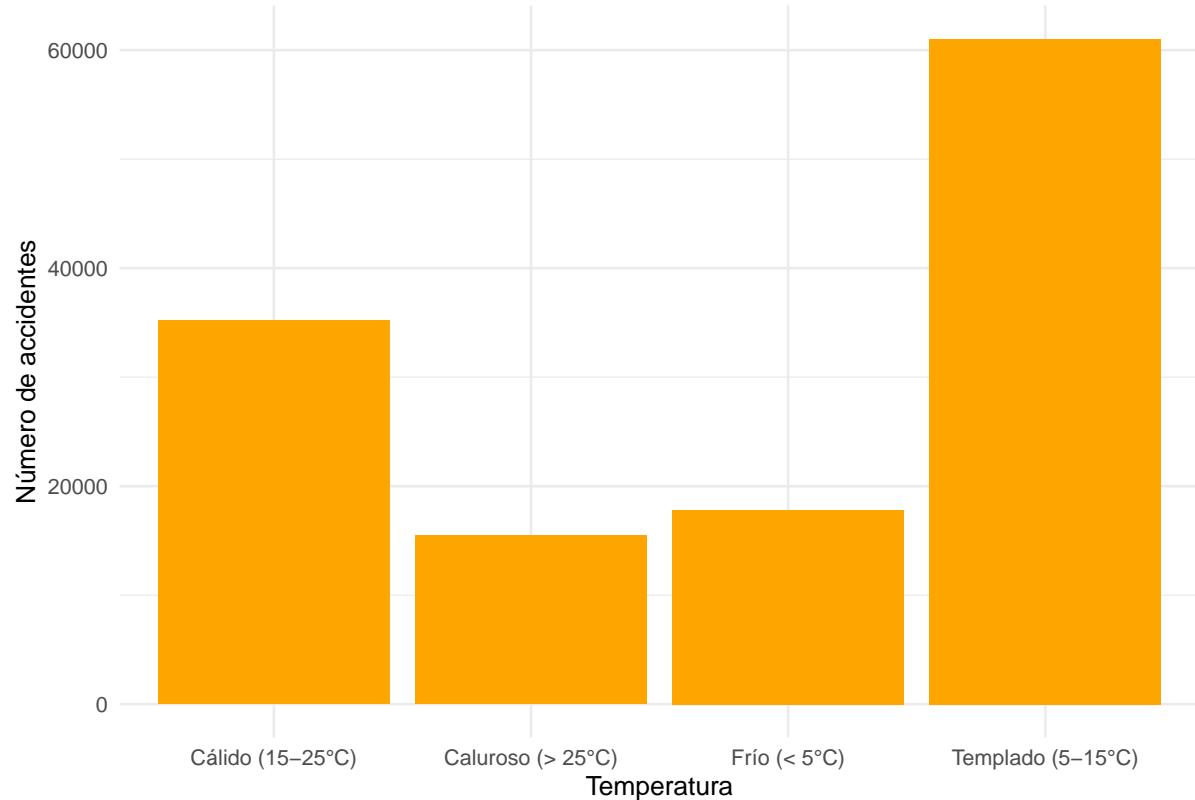


4.3 Accidentes según Categoría de Temperatura

```
# Filtrar datos con temperatura categorizada
accidents_clean_data_temp <- accidents_clean_data %>%
  filter(!is.na(categoría_temp))

# Visualización
ggplot(accidents_clean_data_temp, aes(x = categoría_temp)) +
  geom_bar(fill = "orange") +
  labs(
    title = "Accidentes según categoría de temperatura",
    x = "Temperatura",
    y = "Número de accidentes"
  ) +
  theme_minimal()
```

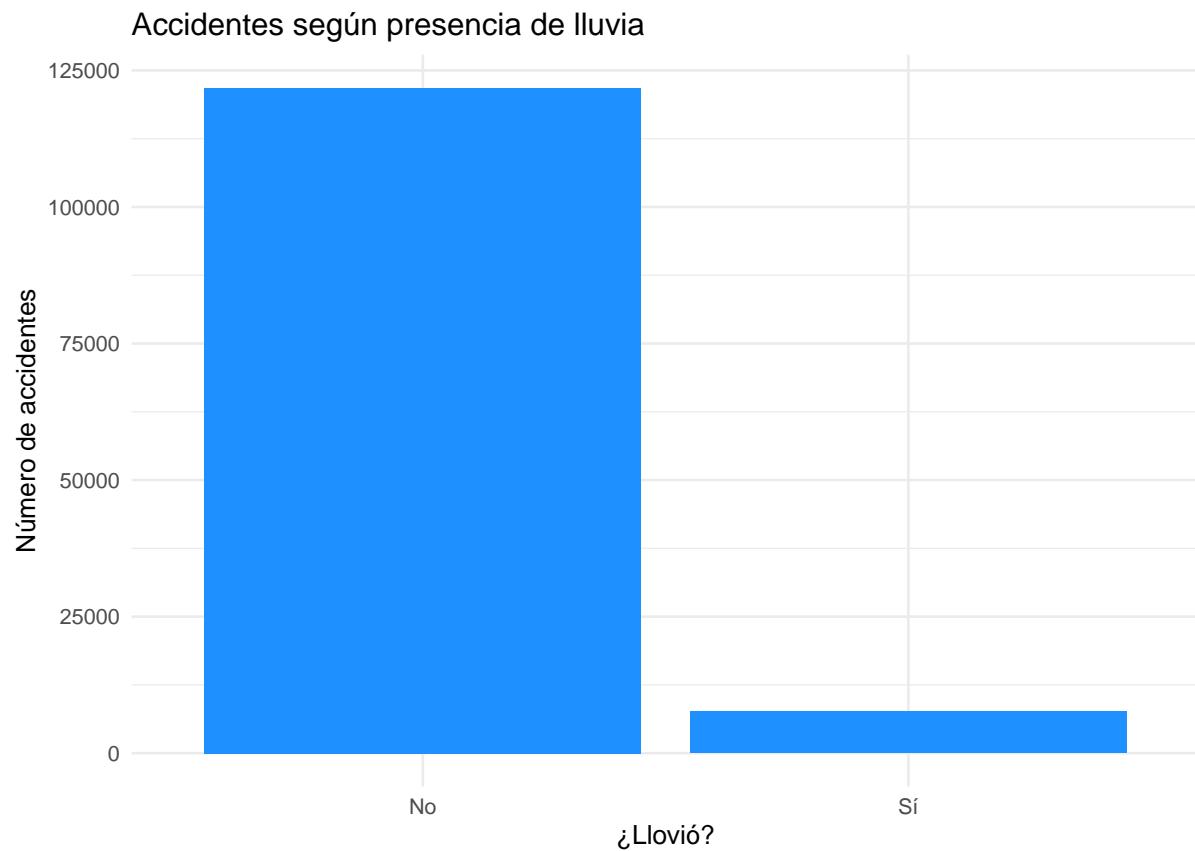
Accidentes según categoría de temperatura



4.4 Accidentes con y sin Lluvia

```
# Filtrar datos con información de lluvia
accidents_clean_data_lluvia <- accidents_clean_data %>%
  filter(!is.na(hubo_lluvia))

# Visualización
ggplot(accidents_clean_data_lluvia, aes(x = hubo_lluvia)) +
  geom_bar(fill = "dodgerblue") +
  labs(
    title = "Accidentes según presencia de lluvia",
    x = "¿Llovió?",
    y = "Número de accidentes"
  ) +
  theme_minimal()
```

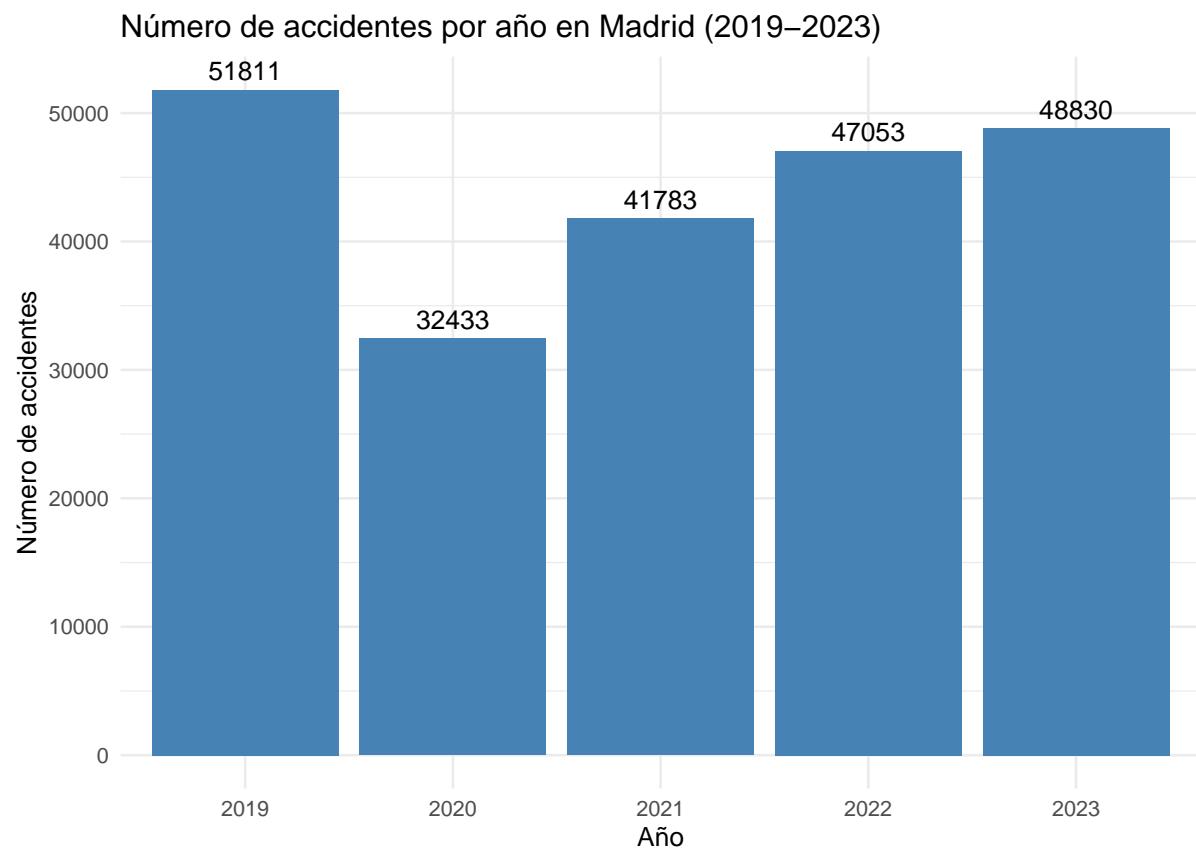


5 Análisis Temporal

5.1 Distribución por Año

```
# Calcular accidentes por año
accidentes_por_anio <- accidents_clean_data %>%
  group_by(anio) %>%
  summarise(total_accidentes = n())

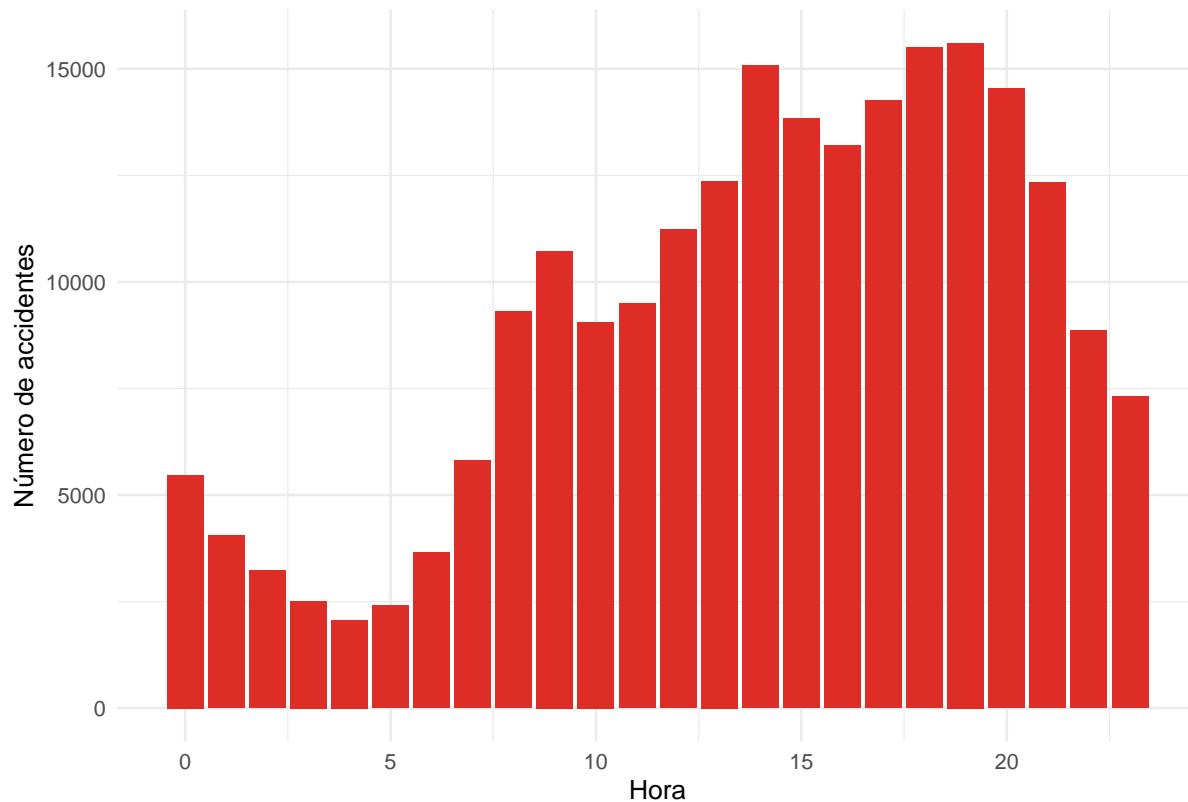
# Visualización
ggplot(accidentes_por_anio, aes(x = factor(anio), y = total_accidentes)) +
  geom_col(fill = "steelblue") +
  geom_text(aes(label = total_accidentes), vjust = -0.5, size = 4) +
  labs(
    title = "Número de accidentes por año en Madrid (2019-2023)",
    x = "Año",
    y = "Número de accidentes"
  ) +
  theme_minimal()
```



5.2 Distribución por Hora del Día

```
ggplot(accidents_clean_data, aes(x = hora)) +  
  geom_bar(fill = "#DE2D27") +  
  labs(  
    title = "Distribución de accidentes por hora del día",  
    x = "Hora",  
    y = "Número de accidentes"  
) +  
  theme_minimal()
```

Distribución de accidentes por hora del día



6 Análisis Temporal Avanzado

6.1 Configuración para Análisis de Días Festivos

```
# Parámetros para análisis temporal con festivos
TZ_LOCAL <- "Europe/Madrid"
COUNTRY <- "ES"

# Asegurar que tenemos la librería necesaria
if (!require("jsonlite")) install.packages("jsonlite")
library(jsonlite)
```

6.2 Media de Accidentes por Día de la Semana

```
# Convertir fecha a formato Date
fecha_date <- as.Date(accidents_clean_data$time, tz = TZ_LOCAL)

# Agregar accidentes por fecha
daily <- as.data.frame(table(fecha_date))
names(daily) <- c("fecha", "accidents")
daily$fecha <- as.Date(daily$fecha)

# Obtener día de la semana
wd <- as.POSIXlt(daily$fecha)$wday
```

```

map_es <- c("domingo", "lunes", "martes", "miércoles", "jueves", "viernes", "sábado")
lvl_es <- c("lunes", "martes", "miércoles", "jueves", "viernes", "sábado", "domingo")
daily$day <- factor(map_es[wd + 1], levels = lvl_es, ordered = TRUE)

# Calcular estadísticas por día de la semana
sum_by <- aggregate(accidents ~ day, data = daily, sum)
n_by <- aggregate(fecha ~ day, data = daily, function(x) length(unique(x)))
names(n_by)[2] <- "n_días"

# Calcular media: total de accidentes / número de días de ese tipo
stats <- merge(sum_by, n_by, by = "day", all = TRUE)
stats$mean_acc <- stats$accidents / stats$n_días

# Mostrar estadísticas con formato ajustado
stats %>%
  kable(format = "latex", booktabs = TRUE,
        digits = 2,
        col.names = c("Día", "Total accidentes", "Nº días", "Media"),
        caption = "Estadísticas por día de la semana") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"),
                font_size = 10,
                full_width = FALSE)

```

Table 9: Estadísticas por día de la semana

Día	Total accidentes	Nº días	Media
domingo	26266	261	100.64
jueves	33563	261	128.59
lunes	29795	260	114.60
martes	32493	261	124.49
miércoles	32900	261	126.05
sábado	29595	261	113.39
viernes	37298	261	142.90

```

# Visualización de media de accidentes por día
ggplot(stats, aes(x = day, y = mean_acc)) +
  geom_col(fill = "grey70", color = "grey30") +
  geom_text(aes(label = sprintf("%.2f", mean_acc)), vjust = -0.4, size = 3.5) +
  scale_y_continuous(expand = expansion(mult = c(0, 0.08))) +
  labs(
    title = "Media de accidentes por día de la semana (2019-2023)",
    x = "Día de la semana",
    y = "Accidentes por día de la semana (media)"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold", size = 16, hjust = 0.5),
    panel.grid.minor = element_blank()
  )

```

Media de accidentes por día de la semana (2019–2023)

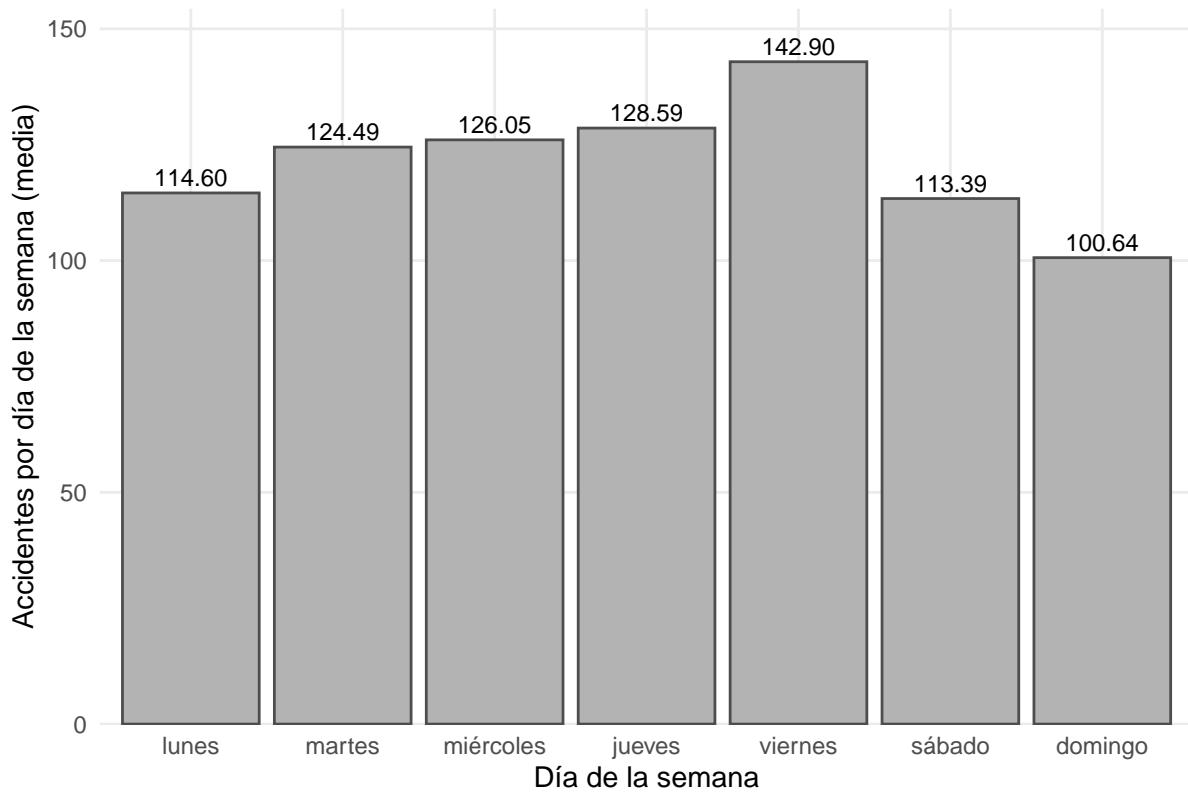


Figure 1: Media de accidentes por día de la semana

6.3 Obtención de Días Festivos

```
# Función para obtener festivos desde API Nager.Date
get_holidays <- function(year, country = COUNTRY) {
  url <- paste0("https://date.nager.at/api/v3/PublicHolidays/", year, "/", country)
  json <- try(jsonlite::fromJSON(url), silent = TRUE)

  if (inherits(json, "try-error") || length(json) == 0) {
    warning("No se pudieron obtener festivos para ", year)
    return(data.frame(date = as.Date(character()), stringsAsFactors = FALSE))
  }

  data.frame(
    date = as.Date(json$date),
    localName = json$localName,
    name = json$name,
    fixed = json$fixed,
    global = json$global,
    stringsAsFactors = FALSE
  )
}

# Obtener años únicos del dataset
years <- sort(unique(format(as.Date(accidents_clean_data$time, tz = TZ_LOCAL), "%Y")))

# Obtener festivos para todos los años
hols_list <- lapply(years, get_holidays)
hols <- if (length(hols_list)) do.call(rbind, hols_list) else data.frame(date = as.Date(character()))
hols_days <- unique(hols$date)

# Mostrar festivos obtenidos
cat("Total de días festivos identificados:", length(hols_days), "\n\n")

## Total de días festivos identificados: 150

# Tabla con primeros festivos
head(hols[, c("date", "localName")], 10) %>%
  kable(format = "latex", booktabs = TRUE,
        col.names = c("Fecha", "Nombre"),
        caption = "Primeros días festivos identificados") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"),
                font_size = 10,
                full_width = FALSE)
```

Table 10: Primeros días festivos identificados

Fecha	Nombre
2019-01-01	Año Nuevo
2019-01-06	Día de Reyes / Epifanía del Señor
2019-02-28	Día de Andalucía
2019-03-01	Dia de les Illes Balears
2019-03-19	San José
2019-04-18	Jueves Santo
2019-04-19	Viernes Santo
2019-04-22	Lunes de Pascua
2019-04-23	Día de Castilla y León
2019-04-23	San Jorge (Día de Aragón)

6.4 Clasificación de Días Festivos vs No Festivos

```
# Agregar columna de festivo al dataset diario
daily$holiday <- factor(
  ifelse(daily$fecha %in% hols_days, "Festivo (ES)", "No festivo"),
  levels = c("No festivo", "Festivo (ES)")
)

# Resumen de días festivos vs no festivos
tabla_festivos <- as.data.frame(table(daily$holiday))
names(tabla_festivos) <- c("Tipo_dia", "Frecuencia")

tabla_festivos %>%
  kable(format = "latex", booktabs = TRUE,
        caption = "Distribución de días en el período") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"),
                font_size = 10,
                full_width = FALSE)
```

Table 11: Distribución de días en el período

Tipo_dia	Frecuencia
No festivo	1676
Festivo (ES)	150

6.5 Comparativa: Festivo vs No Festivo por Día de la Semana

```
# Calcular media diaria por día de la semana y condición de festivo
mean_by <- aggregate(accidents ~ day + holiday, data = daily, FUN = mean, na.rm = TRUE)
names(mean_by)[names(mean_by) == "accidents"] <- "mean_acc"

# Contar número de días en cada combinación
n_by <- aggregate(fecha ~ day + holiday, data = daily, function(x) length(unique(x)))
names(n_by)[names(n_by) == "fecha"] <- "n_dias"
```

```

# Combinar estadísticas
comp_mean <- merge(mean_by, n_by, by = c("day", "holiday"), all = TRUE)
comp_mean <- comp_mean[order(comp_mean$day, comp_mean$holiday), ]

# Mostrar tabla comparativa con formato ajustado
comp_mean %>%
  kable(format = "latex", booktabs = TRUE,
        digits = 2,
        col.names = c("Día", "Tipo", "Media acc.", "Nº días"),
        caption = "Media de accidentes por día de la semana y condición de festivo") %>%
  kable_styling(latex_options = c("striped", "scale_down", "HOLD_position"),
                font_size = 9,
                full_width = FALSE)

```

Table 12: Media de accidentes por día de la semana y condición de festivo

	Día	Tipo	Media acc.	Nº días
6	lunes	No festivo	116.33	232
5	lunes	Festivo (ES)	100.25	28
8	martes	No festivo	124.95	240
7	martes	Festivo (ES)	119.24	21
10	miércoles	No festivo	126.26	243
9	miércoles	Festivo (ES)	123.28	18
4	jueves	No festivo	131.54	235
3	jueves	Festivo (ES)	101.96	26
14	viernes	No festivo	146.19	236
13	viernes	Festivo (ES)	111.84	25
12	sábado	No festivo	114.31	247
11	sábado	Festivo (ES)	97.14	14
2	domingo	No festivo	99.84	243
1	domingo	Festivo (ES)	111.39	18

```

# Visualización comparativa por día de la semana
pd <- position_dodge(width = 0.8)

ggplot(comp_mean, aes(x = day, y = mean_acc, fill = holiday)) +
  geom_col(position = pd, color = "grey30") +
  geom_text(
    aes(label = sprintf("%.2f", mean_acc)),
    position = pd,
    vjust = -0.35,
    size = 3.3,
    na.rm = TRUE
  ) +
  scale_y_continuous(expand = expansion(mult = c(0, 0.10))) +
  scale_fill_manual(values = c("#9ecae1", "#de2d26")) +
  labs(
    title = "Media de accidentes diarios - Festivo vs. No festivo (2019–2023)",
    subtitle = "Para cada día: total de accidentes / número de días de ese tipo",
    x = "Día de la semana",
    y = "Media de accidentes diarios",
    fill = ""
  ) +
  theme_minimal(base_size = 12) +
  theme(

```

```

    plot.title = element_text(face = "bold", size = 16, hjust = 0.5),
    panel.grid.minor = element_blank()
)

```

dia de accidentes diarios – Festivo vs. No festivo (2019–2023)

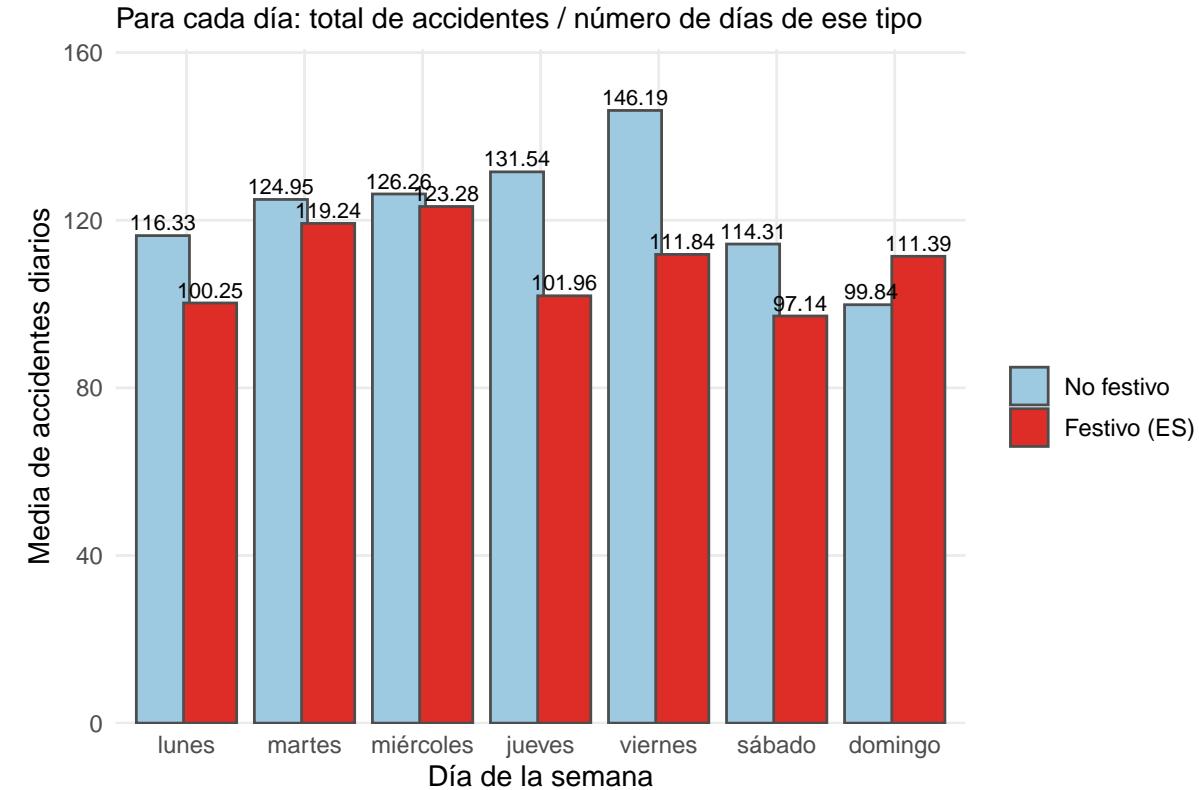


Figure 2: Comparativa de accidentes: festivo vs no festivo por día de la semana

6.6 Comparativa General: Festivo vs No Festivo

```

# Calcular media diaria por condición (festivo vs no festivo)
mean_h <- aggregate(accidents ~ holiday, data = daily, FUN = mean, na.rm = TRUE)
names(mean_h)[2] <- "mean_acc"

# Contar número de días en cada grupo
n_h <- aggregate(fecha ~ holiday, data = daily, function(x) length(unique(x)))
names(n_h)[2] <- "n_dias"

# Combinar estadísticas
mean_h <- merge(mean_h, n_h, by = "holiday", all = TRUE)

# Mostrar estadísticas con formato ajustado
mean_h %>%
  kable(format = "latex", booktabs = TRUE,
        digits = 2,
        col.names = c("Tipo día", "Media accidentes", "Nº días"),
        caption = "Estadísticas generales festivo vs no festivo") %>%

```

```
kable_styling(latex_options = c("striped", "HOLD_position"),
             font_size = 10,
             full_width = FALSE)
```

Table 13: Estadísticas generales festivo vs no festivo

Tipo día	Media accidentes	Nº días
Festivo (ES)	108.95	150
No festivo	122.65	1676

```
# Visualización comparativa general
ggplot(mean_h, aes(x = holiday, y = mean_acc, fill = holiday)) +
  geom_col(color = "grey30", width = 0.7, show.legend = FALSE) +
  geom_text(aes(label = sprintf("%.2f", mean_acc)), vjust = -0.35, size = 3.6) +
  scale_y_continuous(expand = expansion(mult = c(0, 0.10))) +
  scale_fill_manual(values = c("#9ecae1", "#de2d26")) +
  labs(
    title = "Media de accidentes - Festivo vs. No festivo (2019-2023)",
    subtitle = "Total de accidentes por condición / número de días de esa condición",
    x = "",
    y = "Media de accidentes"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    plot.title = element_text(face = "bold", size = 16, hjust = 0.5),
    panel.grid.minor = element_blank()
  )
```

6.7 Resumen Estadístico: Impacto de Días Festivos

```
# Calcular proporción de accidentes en festivos
totales <- aggregate(accidents ~ holiday, data = daily, FUN = sum)
prop_holiday <- with(totales, accidents[holiday == "Festivo (ES)"] / sum(accidents))

# Mostrar resumen
cat("\n==== RESUMEN DE ANÁLISIS TEMPORAL ===\n")

## 
## === RESUMEN DE ANÁLISIS TEMPORAL ===

cat("Total de accidentes por condición:\n")

## Total de accidentes por condición:

print(totales)

##      holiday accidents
## 1  No festivo     205568
## 2 Festivo (ES)     16342
```

Media de accidentes – Festivo vs. No festivo (2019–2023)

Total de accidentes por condición / número de días de esa condición

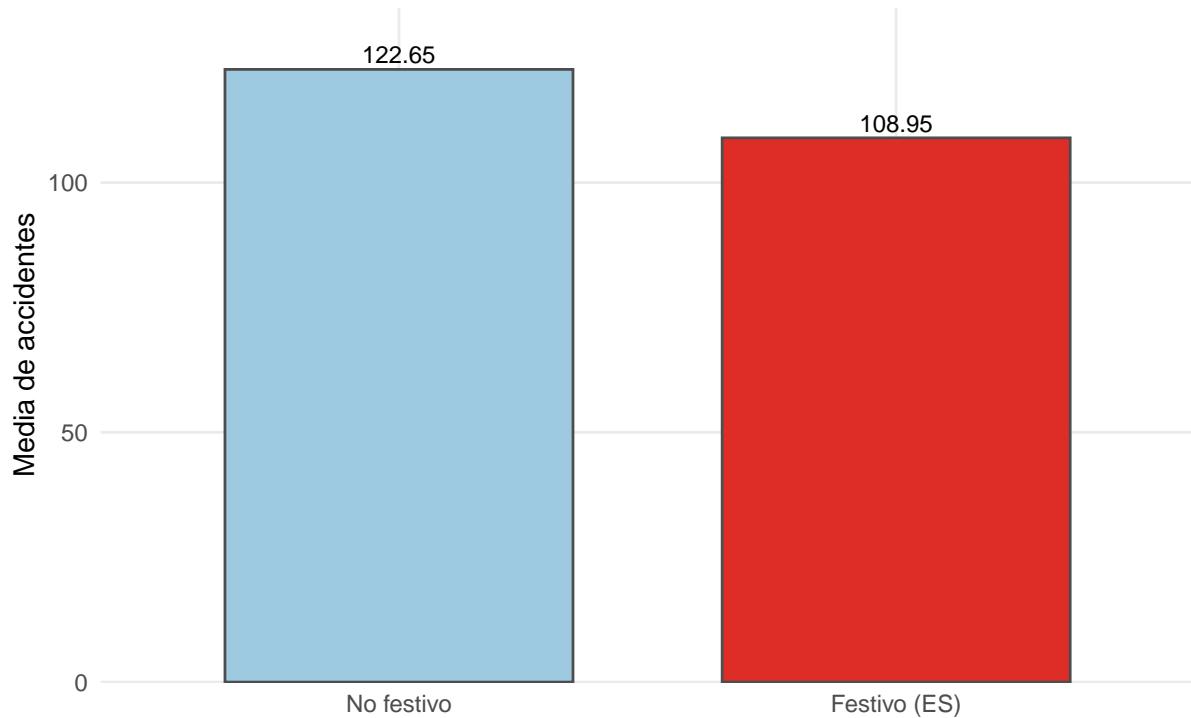


Figure 3: Comparativa general: media de accidentes en días festivos vs no festivos

```

cat("\n Estadísticas clave:\n")

##
## Estadísticas clave:

cat(sprintf("- Proporción de accidentes en días festivos: %.2f%%\n", 100 * prop_holiday))

## - Proporción de accidentes en días festivos: 7.36%

cat(sprintf("- Media en días festivos: %.2f accidentes/día\n",
            mean_h$mean_acc[mean_h$holiday == "Festivo (ES)"]))

## - Media en días festivos: 108.95 accidentes/día

cat(sprintf("- Media en días no festivos: %.2f accidentes/día\n",
            mean_h$mean_acc[mean_h$holiday == "No festivo"]))

## - Media en días no festivos: 122.65 accidentes/día

# Calcular diferencia porcentual
diff_pct <- ((mean_h$mean_acc[mean_h$holiday == "Festivo (ES)"] /
               mean_h$mean_acc[mean_h$holiday == "No festivo"]) - 1) * 100
cat(sprintf("- Diferencia: %.2f%% %s accidentes en festivos\n",
            abs(diff_pct),
            ifelse(diff_pct > 0, "más", "menos")))

## - Diferencia: 11.18% menos accidentes en festivos

```

7 Análisis Espacial

7.1 Preparación de Datos Geoespaciales

```

# Filtrar y normalizar coordenadas UTM
accidents_clean_data_filtered <- accidents_clean_data %>%
  filter(!is.na(coordenada_x_utm) & !is.na(coordenada_y_utm)) %>%
  mutate(
    coordenada_x_utm = as.numeric(coordenada_x_utm) / 1000,
    coordenada_y_utm = as.numeric(coordenada_y_utm) / 1000
  ) %>%
# Filtrar coordenadas válidas para el área metropolitana de Madrid (UTM zona 30N)
filter(
  coordenada_x_utm >= 430000 & coordenada_x_utm <= 450000,
  coordenada_y_utm >= 4465000 & coordenada_y_utm <= 4485000
)

# Convertir a objeto espacial SF con proyección UTM
accidentes_sf <- st_as_sf(accidents_clean_data_filtered,
                           coords = c("coordenada_x_utm", "coordenada_y_utm"),
                           crs = 25830)

# Transformar a sistema WGS84 (latitud/longitud)
accidentes_sf <- st_transform(accidentes_sf, crs = 4326)

```

7.2 Mapa de Calor de Densidad de Accidentes

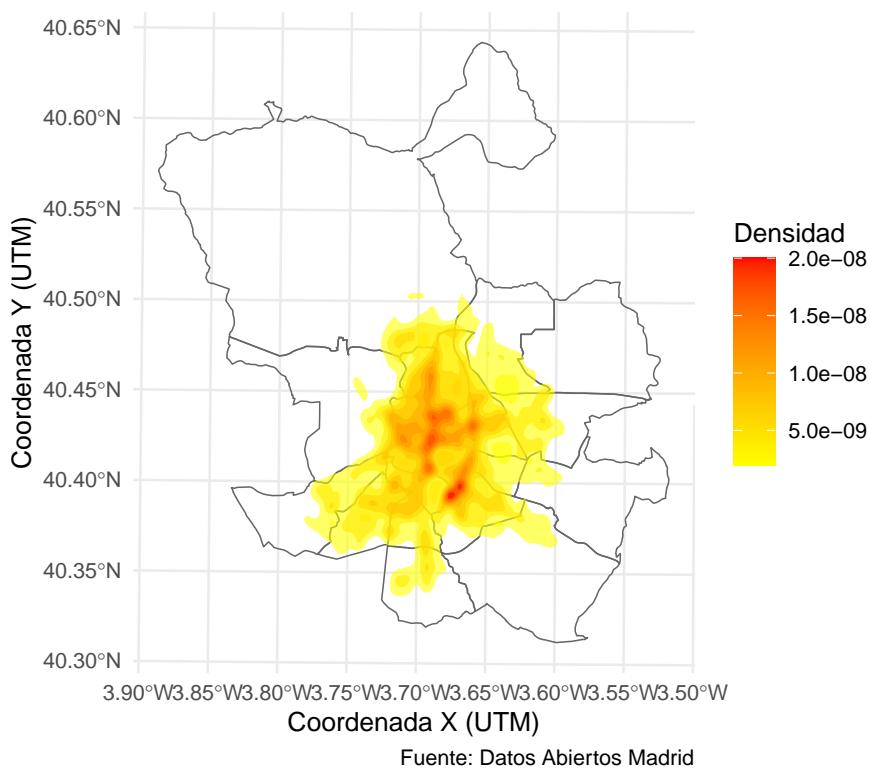
```
# Cargar shapefile de los distritos de Madrid
madrid_sf <- st_read("../data/raw/distritos/DISTRITOS.shp")

## Reading layer `DISTRITOS` from data source
##   `/Users/ditmarestradabernuy/Yachay/UPV/Data Science/DASproject/data/raw/distritos/DISTRITOS.shp'
##   using driver `ESRI Shapefile'
## Simple feature collection with 21 features and 11 fields
## Geometry type: POLYGON
## Dimension:      XY
## Bounding box:  xmin: 424753.5 ymin: 4462566 xmax: 456033.3 ymax: 4499366
## Projected CRS: ETRS89 / UTM zone 30N

# Crear mapa de calor superpuesto sobre límites de distritos
ggplot() +
  # Capa base: límites administrativos de distritos
  geom_sf(data = madrid_sf, fill = NA, color = "gray40", size = 0.3) +
  # Capa de densidad: mapa de calor de accidentes
  stat_density_2d(
    data = accidents_clean_data_filtered,
    aes(x = coordenada_x_utm, y = coordenada_y_utm, fill = ..level..),
    geom = "polygon",
    alpha = 0.6
  ) +
  # Escala de color de amarillo (baja densidad) a rojo (alta densidad)
  scale_fill_gradient(low = "yellow", high = "red", name = "Densidad") +
  # Títulos y etiquetas
  labs(
    title = "Mapa de calor de accidentes en Madrid",
    subtitle = "Superpuesto sobre distritos (2019-2023)",
    x = "Coordenada X (UTM)",
    y = "Coordenada Y (UTM)",
    caption = "Fuente: Datos Abiertos Madrid"
  ) +
  # Estilo visual
  theme_minimal() +
  theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 10),
    legend.position = "right"
  )
```

Mapa de calor de accidentes en Madrid

Superpuesto sobre distritos (2019–2023)



7.3 Mapa de Burbujas por Distrito

```
# Normalizar nombres de distrito para evitar problemas de coincidencia
madrid_sf$NOMBRE <- toupper(trimws(madrid_sf$NOMBRE))
accidents_clean_data_filtered$distrito <- toupper(trimws(accidents_clean_data_filtered$distrito))

# Calcular centroides geométricos de cada distrito
centroides_sf <- st_point_on_surface(madrid_sf)
coords <- st_coordinates(centroides_sf)

# Contar accidentes por distrito
accidentes_distrito <- accidents_clean_data_filtered %>%
  count(distrito, name = "n_accidentes")

# Preparar datos para visualización con coordenadas de centroides
datos_mapa <- data.frame(
  distrito = madrid_sf$NOMBRE,
  x = coords[,1],
  y = coords[,2]
) %>%
  left_join(accidentes_distrito, by = c("distrito" = "distrito")) %>%
  mutate(n_accidentes = ifelse(is.na(n_accidentes), 0, n_accidentes))

# Crear mapa con burbujas proporcionales al número de accidentes
ggplot() +
  # Capa base: polígonos de distritos
  geom_sf(data = madrid_sf, fill = "gray95", color = "gray50", size = 0.3) +
```

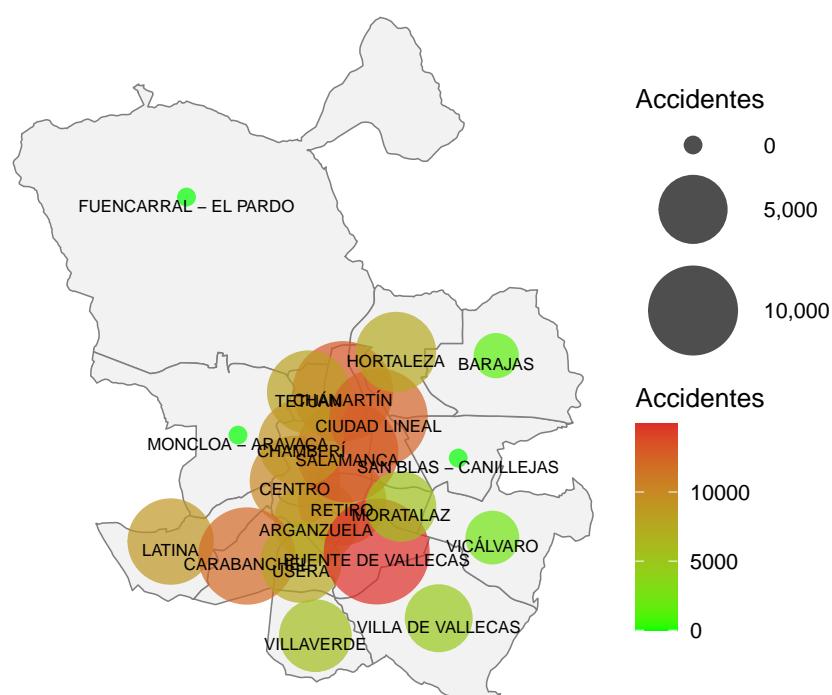
```

# Burbujas: tamaño proporcional al número de accidentes
geom_point(data = datos_mapa,
            aes(x = x, y = y, size = n_accidentes, color = n_accidentes),
            alpha = 0.7) +
            
# Escalas de color y tamaño
scale_color_gradient(low = "green", high = "#de2d27", name = "Accidentes") +
scale_size_continuous(range = c(3, 20), name = "Accidentes", labels = comma) +
            
# Etiquetas de nombre de distrito
geom_text(data = datos_mapa,
            aes(x = x, y = y, label = distrito),
            size = 2.5, color = "black", vjust = 1.2) +
            
# Títulos y metadatos
labs(
    title = "Cantidad de accidentes por distrito",
    subtitle = "Madrid 2019-2023",
    caption = "Fuente: Datos Abiertos Madrid"
) +
            
# Estilo visual limpio
theme_void() +
theme(
    plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
    plot.subtitle = element_text(hjust = 0.5, size = 10),
    legend.position = "right"
)

```

Cantidad de accidentes por distrito

Madrid 2019-2023

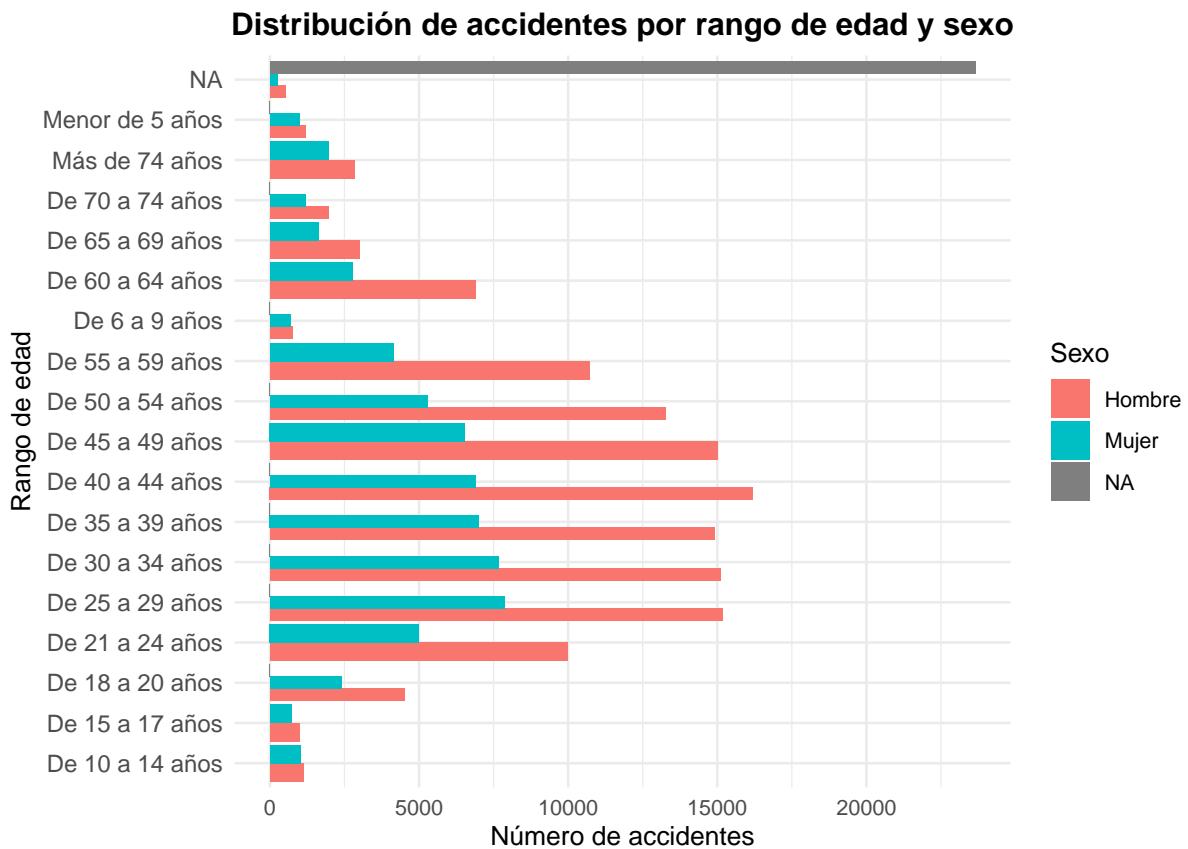


Fuente: Datos Abiertos Madrid

8 Análisis de Factores Humanos

8.1 Distribución por Rango de Edad y Sexo

```
ggplot(accidents_clean_data, aes(x = rango_edad, fill = sexo)) +  
  geom_bar(position = "dodge") +  
  coord_flip() +  
  labs(  
    title = "Distribución de accidentes por rango de edad y sexo",  
    x = "Rango de edad",  
    y = "Número de accidentes",  
    fill = "Sexo"  
) +  
  theme_minimal() +  
  theme(  
    plot.title = element_text(face = "bold", hjust = 0.5),  
    axis.text.y = element_text(size = 10)  
)
```



9 Análisis Multivariado

9.1 Matriz de Correlaciones entre Variables Meteorológicas

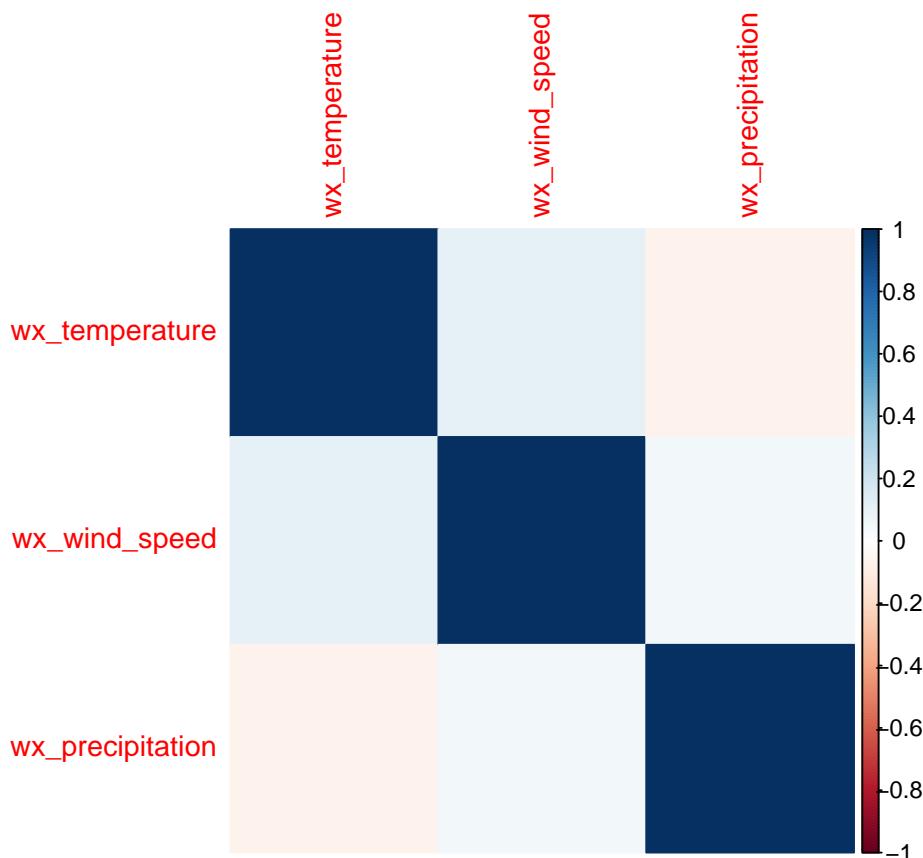
```

# Seleccionar únicamente variables meteorológicas numéricas
numericas <- accidents_clean_data[, c("wx_temperature", "wx_wind_speed", "wx_precipitation")]

# Calcular matriz de correlación
cor_matrix <- cor(numericas, use = "complete.obs")

# Visualizar matriz de correlaciones
corrplot(cor_matrix, method = "color")

```



10 Análisis Avanzado: Clustering Geográfico

10.1 Identificación de Zonas de Alta Concentración de Accidentes

```

# Preparar dataset con solo coordenadas espaciales
datos_geo <- accidents_clean_data_filtered[, c("coordenada_x_utm", "coordenada_y_utm")]

# Aplicar algoritmo K-means con 5 clusters
set.seed(123)
km_result <- kmeans(datos_geo, centers = 5, nstart = 25)

# Añadir identificador de cluster al dataset
accidents_clean_data_filtered$cluster_zona <- as.factor(km_result$cluster)

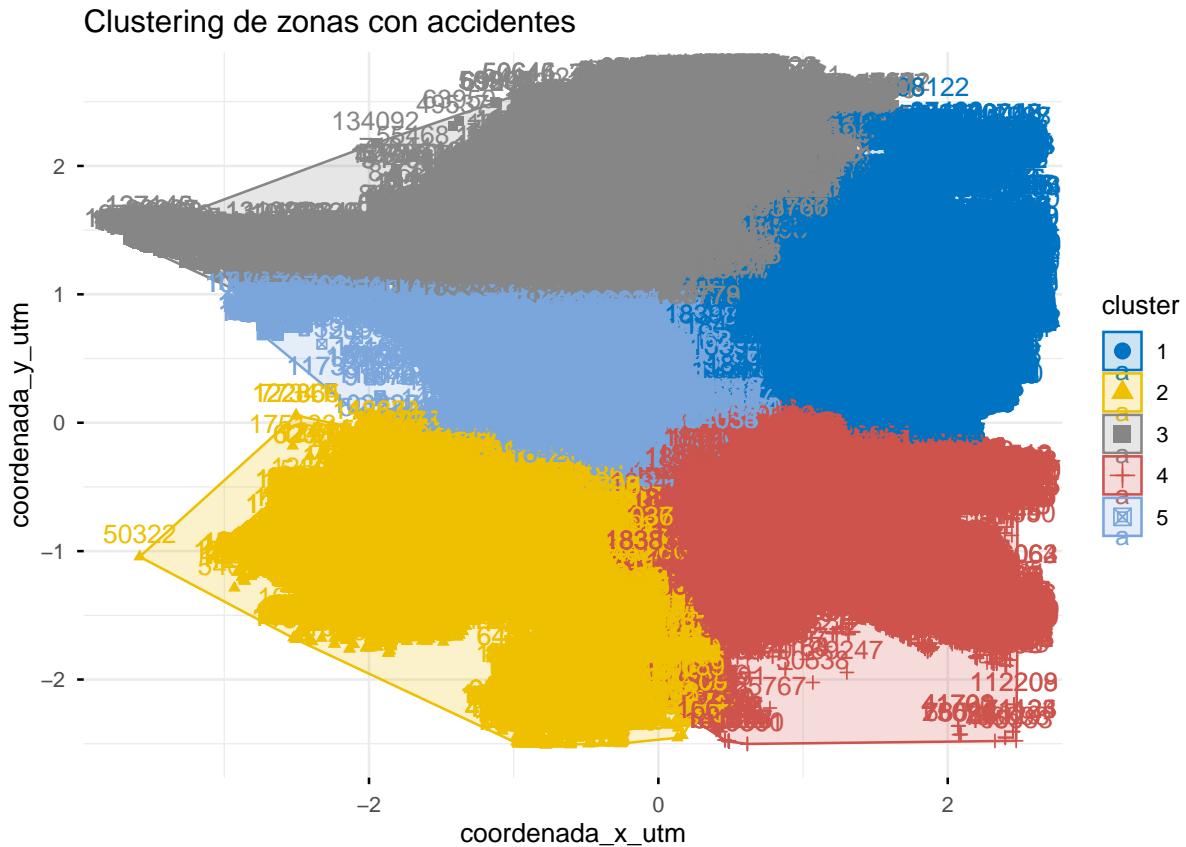
# Visualizar clusters identificados
fviz_cluster(km_result, data = datos_geo,

```

```

palette = "jco",
ggtheme = theme_minimal(),
main = "Clustering de zonas con accidentes")

```



11 Conclusiones

11.1 Resumen de Hallazgos Clave

- Análisis temporal:** Se identificaron patrones claros en la distribución de accidentes por hora, día de la semana y mes.
- Factores meteorológicos:** Las condiciones climáticas muestran relación con la frecuencia de accidentes.
- Análisis espacial:** Ciertos distritos presentan mayor concentración de accidentes.
- Clustering geográfico:** Se identificaron 5 zonas principales con características similares de siniestralidad.

11.2 Ruta Metodológica Seguida

- Carga y limpieza de datos:** Tratamiento de valores faltantes y normalización.
- Análisis univariante:** Exploración de cada variable por separado.
- Análisis bivariante:** Relaciones entre clima, tiempo y accidentes.
- Análisis temporal y espacial:** Identificación de patrones espacio-temporales.
- Análisis multivariado:** Correlaciones y clustering.

Información del Análisis

- **Fuente de datos:** [Kaggle](#)
- **Período analizado:** 2019-2023
- **Herramientas:** R 4.5.1
- **Fecha de generación del informe:** 2025-11-01