Informe de Análisis de Datos: Estudio del Acoso en las Comunas de Santiago

Programación con R | Master Data Science | Universidad de las Américas (UDLA.)

- Guido Alejandro Ríos Ciaffaroni
- Eduardo Alex Opazo Diaz

[Ver video 001]

[Ver video 002]

[Ver video 003]

1. Introducción General

1.1. Contextualización del Problema

El acoso, en sus diversas manifestaciones, representa una de las formas más persistentes y normalizadas de violencia en los entornos urbanos contemporáneos. En el contexto chileno, y particularmente en la Región Metropolitana, el acoso callejero, laboral, escolar y digital ha adquirido creciente visibilidad como un problema social que afecta la calidad de vida y la seguridad de miles de personas, especialmente mujeres, niñas y disidencias sexuales. Las comunas de Santiago, al concentrar una alta densidad poblacional, movilidad urbana y diversidad sociocultural, se convierten en espacios donde estas formas de acoso se expresan con particular intensidad y variabilidad.

La creciente preocupación ciudadana, junto con los movimientos feministas y las políticas públicas orientadas a la equidad de género, han impulsado el interés por comprender cómo, dónde y a quiénes afecta el acoso, así como las respuestas institucionales y comunitarias frente a este fenómeno. Sin embargo, aún persiste una falta de información detallada y sistematizada a nivel comunal que permita caracterizar de manera precisa la magnitud y las dinámicas del acoso en el territorio de Santiago de Chile.

1.2. Propósito del Estudio

El propósito de este estudio es identificar, analizar y comprender las distintas formas de acoso que se manifiestan en las comunas de Santiago en los últimos 6 meses, con el fin de evidenciar sus patrones territoriales, características sociodemográficas asociadas, y sus efectos en la vida cotidiana de las personas afectadas. A través de un enfoque territorial y multidimensional, se busca generar información relevante que sirva de base para el diseño e implementación de políticas públicas, estrategias preventivas y acciones de sensibilización orientadas a reducir la incidencia del acoso y promover entornos urbanos más seguros, inclusivos y equitativos. Se considerarán variables como sexo, edad y comuna de residencia. A través del uso de R y técnicas de análisis de datos, se busca identificar patrones, diferencias significativas y posibles factores asociados al fenómeno.

1.3. Objetivos General: Analizar el acoso en las distintas comunas de Santiago. A través de un enfoque mixto, que combina fuentes estadísticas, análisis espacial y percepciones ciudadanas, se busca aportar a la comprensión integral del fenómeno y contribuir al diseño de estrategias de prevención, intervención y reparación adecuadas al contexto local de las comunas de Santiago en los últimos 6 meses.

Específicos:

- Identificar sus principales características

- Identificar factores asociados y diferencias territoriales
- Describir la distribución de acoso según sexo, edad y comuna.
- Evaluar si existe diferencia significativa en la proporción de casos entre hombres y mujeres.
- Modelar la probabilidad de sufrir acoso en función de las variables disponibles.
- Proporcionar visualizaciones claras y relevantes para la interpretación de los resultados.

2. Definición del Problema / Hipótesis

2.1. Pregunta de Investigación ¿Existe una diferencia significativa en la proporción de hombres y mujeres que han reportado haber sufrido acoso en los últimos 6 meses en las comunas de Santiago?

2.2. Hipótesis Estadística Hipótesis Nula (H0): No existe una diferencia significativa en la proporción de casos de acoso entre hombres y mujeres.

Hipótesis Alternativa (H1): Sí existe una diferencia significativa en la proporción de casos de acoso entre hombres y mujeres.

3. Descripción del Conjunto de Datos

Variable Tipo Descripción

- Sexo Cualitativa nominal Género de la persona (Hombre/Mujer)
- Edad Cuantitativa continua Edad de la persona en años completos
- Comuna Cualitativa nominal Comuna de residencia dentro de Santiago
- Acoso Cualitativa binaria Variable indicadora si la persona ha sufrido acoso en los últimos 6 meses (Sí/No)

3.1. Dimensión de los Datos Se especificará la cantidad total de registros, porcentaje por categoría (sexo, comuna) y distribución de la variable respuesta (acoso).

3.2. Exploración de Datos Faltantes e Irregularidades Se realizará una inspección para identificar:

Datos faltantes en variables clave.

Outliers en la variable edad.

Posibles inconsistencias (ej: comunas no válidas).

4. Preprocesamiento de Datos

4.1. Carga y Conversión de Variables Utilizando tidyverse, se realizará la carga de datos y la conversión de variables a factores según corresponda.

4.2. Tratamiento de Datos Faltantes Se definirán criterios de exclusión o imputación si fuese necesario.

4.3. Creación de Variables Derivadas

- Agrupación de edades en rangos.
- Variables indicadoras si se requieren para modelado.

4.4. Resumen de Preprocesamiento Se incluirá un flujo esquemático de limpieza, transformación y preparación del dataset.

5. Análisis Exploratorio de Datos (EDA)

5.1. Distribución Demográfica

- Gráficos de barras para la distribución de sexo.
- Histograma de edades global y por sexo.

5.2. Incidencia de Acoso por Sexo

- Proporción de casos de acoso por género.
- Comparación gráfica (barras, proporciones).

5.3. Distribución Geográfica del Acoso

- Frecuencia de casos por comuna.
- Visualización con gráficos de barras o mapas de calor.

5.4. Relaciones Cruzadas

- Boxplots de edad por condición de acoso.
- Tablas de contingencia cruzando sexo, comuna y acoso.

6. Modelado de Datos

6.1. Selección del Modelo Se utilizará un modelo de regresión logística binaria, dado que la variable respuesta (acoso) es dicotómica.

6.2. Especificación del Modelo acoso ~ sexo + edad + comuna

6.3. Evaluación del Modelo

- Estimación de Odds Ratios.
- Prueba de significancia para coeficientes.
- Evaluación de ajuste y predicción.

6.4. Visualización de Resultados

- Gráficos de efectos marginales.

- Probabilidades predichas para interpretación práctica.

7. Interpretación de Resultados

7.1. Discusión de la Hipótesis

- Conclusión respecto a la existencia de diferencias significativas en la proporción de acoso entre hombres y mujeres.
- Impacto de la edad y comuna en la probabilidad de acoso.

7.2. Implicancias Sociales

- Reflexión sobre los resultados en el contexto de la ciudad de Santiago.
- Posibles recomendaciones para prevención e intervención.

8. Código R y Repositorio

8.1. Estructura

Además, se incluyen utilidades como wget, unzip, ca-certificates y gnupg2 para la gestión de paquetes y repositorios seguros. Se habilitan módulos de Apache como php* y rewrite, fundamentales para el funcionamiento de frameworks y CMS (como WordPress o Laravel).

Finalmente, el script realiza una limpieza de paquetes innecesarios (apt autoremove) y reinicia los servicios de Apache y MySQL/MariaDB para aplicar los cambios. Este procedimiento deja el sistema listo para desplegar aplicaciones web en PHP con soporte de base de datos.

```
#!/bin/bash

################################################################################
##################################

# Actualización del sistema: se actualiza la lista de paquetes disponibles y se aplican actualizaciones

################################################################################
##################################

sudo apt update -y

sudo apt upgrade -y

################################################################################
##################################
```

Instalación de herramientas útiles para monitoreo, administración del sistema, y utilidades varias

##

apt-get install -y apg # Generador de contraseñas aleatorias seguras

apt-get install -y atop # Monitor avanzado de uso de recursos del sistema

apt-get install -y bmon # Monitor de ancho de banda en tiempo real

apt-get install -y byobu # Entorno de terminal con paneles y multiplexación

apt-get install -y ccze # Colorea logs para facilitar la lectura

apt-get install -y cmatrix # Animación de texto al estilo "The Matrix"

apt-get install -y console-setup # Configura el teclado y la consola del sistema

apt-get install -y console-setup-linux # Archivos específicos de Linux para setup de consola

apt-get install -y cron # Herramienta para ejecutar tareas programadas

apt-get install -y cron-daemon-common # Archivos comunes del servicio cron

apt-get install -y gawk # Versión GNU de AWK para procesamiento de texto

apt-get install -y gettext-base # Herramienta para traducción y localización de software

apt-get install -y htop # Monitor interactivo de procesos del sistema

apt-get install -y iproute2 # Herramientas modernas para configuración de red

apt-get install -y jp2a # Convierte imágenes JPEG a arte ASCII

apt-get install -y kbd # Configuración de teclado (mapas de teclas, etc.)

apt-get install -y keyboard-configuration # Configura el teclado para consola y X

apt-get install -y libatm1t64 # Soporte para conexiones ATM (Asynchronous Transfer Mode)

apt-get install -y libbpf1 # Biblioteca para trabajar con eBPF

apt-get install -y libconfuse-common # Archivos comunes para libConfuse (parser de config)

apt-get install -y libconfuse2 # Biblioteca para parsear archivos de configuración

apt-get install -y libevent-core-2.1-7t64 # Biblioteca para programación basada en eventos

apt-get install -y libfribidi0 # Soporte para texto bidireccional (ej. árabe, hebreo)

apt-get install -y libio-pty-perl # Permite manejar pseudo-terminales en Perl

```
apt-get install -y libipc-run-perl          # Ejecuta procesos y comunica entre ellos en Perl
apt-get install -y libmnl0                  # Biblioteca minimalista para Netlink (interfaz kernel)
apt-get install -y libncurses6              # Biblioteca para interfaces de texto enriquecido
apt-get install -y libnewt0.52              # Biblioteca para diálogos en consola (TUI)
apt-get install -y libnl-3-200              # Biblioteca para manipular netlink en C
apt-get install -y libnl-genl-3-200         # Extensión para mensajes genéricos Netlink
apt-get install -y libnl-route-3-200        # Ruta y manejo de red vía netlink
apt-get install -y libsigsegv2              # Manejo de errores por violación de segmento (debugging)
apt-get install -y libslang2                # Biblioteca para interfaces en modo texto
apt-get install -y libtime-duration-perl    # Manejo de duraciones de tiempo en Perl
apt-get install -y libtimedate-perl         # Manejo de fechas y tiempos en Perl
apt-get install -y libtirpc-common          # Archivos comunes para RPC (Remote Procedure Call)
apt-get install -y libtirpc3t64             # Implementación moderna de RPC para Linux
apt-get install -y liburing2                # Biblioteca para syscalls asincrónicos (io_uring)
apt-get install -y libutempter0             # Maneja entradas en utmp al abrir pseudoterminales
apt-get install -y libxtables12             # Biblioteca base para iptables
apt-get install -y moreutils                # Conjunto de herramientas útiles para la línea de comandos
apt-get install -y pastebinit               # Envia texto o archivos a servicios pastebin desde terminal
apt-get install -y plocate                  # Reemplazo moderno y rápido de `locate`
apt-get install -y python3-newt             # Binding de Python para libnewt (interfaz TUI)
apt-get install -y python3-psutil           # Acceso a información de procesos y sistema en Python
apt-get install -y python3-typing-extensions  # Extensiones de tipado para Python 3
apt-get install -y python3-urwid            # Biblioteca para interfaces de usuario en texto (Python)
apt-get install -y python3-wcwidth          # Manejo de caracteres anchos en consola
apt-get install -y run-one                  # Evita que se ejecuten múltiples instancias del mismo
comando
apt-get install -y speedometer              # Visualiza velocidad de red o disco en consola
apt-get install -y tmux                     # Multiplexor de terminal, permite múltiples sesiones
apt-get install -y tree                     # Muestra estructura de directorios como árbol
```

```
apt-get install -y xkb-data             # Datos de definición de teclados para X
apt-get install -y hollywood            # Simula actividad de hacker en la terminal

################################################################################
#################################

# Instalación de herramientas básicas comunes para administración de sistemas y redes

################################################################################
#################################
sudo apt install curl -y                # Herramienta de transferencia de datos por URL
sudo apt install git -y                 # Sistema de control de versiones distribuido
sudo apt install unzip -y               # Utilidad para descomprimir archivos ZIP
sudo apt install ufw -y                 # Firewall simple para administrar reglas de red
sudo apt install mailutils -y           # Utilidades para enviar correo desde la terminal
sudo apt install tmux -y                # Multiplexor de terminal (repetido, ya instalado arriba)
sudo apt install vim -y                 # Editor de texto mejorado basado en vi
sudo apt install vsftpd -y              # Servidor FTP seguro y ligero

################################################################################
#################################

# Configuración de locale del sistema (idioma y codificación de caracteres)

################################################################################
#################################
sudo apt install locales -y             # Instalación de paquetes de localización
sudo dpkg-reconfigure locales           # Configuración interactiva de locales
#97. en_US.UTF-8 UTF-8                  # Seleccionar esta opción al configurar locales
sudo locale-gen                         # Genera los archivos de locales
sudo update-locale LANG=en_US.UTF-8     # Establece el locale por defecto del sistema
#sudo update-locale LANG=es_CL.UTF-8 LANGUAGE=es_CL.UTF-8  # Alternativa para español
de Chile

source /etc/default/locale              # Carga las variables de entorno del locale
```

Esperado:

LANG=en_US.UTF-8

LANGUAGE=en_US.UTF-8

LC_ALL=

##
##################################

Configuracion de Servidor

Este script automatiza la activación de extensiones críticas de PHP 8.3 en un sistema Ubuntu, editando directamente los archivos php.ini para los modos Apache, CLI y FPM. Utiliza el comando sed para descomentar (activar) extensiones como curl, gd, mbstring, mysqli y pdo_mysql, esenciales para la funcionalidad de muchas aplicaciones web como WordPress o Laravel.

Posteriormente, reinicia el servicio Apache para que los cambios en la configuración surtan efecto. También modifica el archivo principal de configuración de Apache (apache2.conf) para permitir el uso de archivos .htaccess, cambiando la directiva AllowOverride de None a All, lo que es fundamental para la reescritura de URLs y otras configuraciones personalizadas.

El script finaliza ajustando permisos sobre los directorios del servidor web. Establece como propietario al usuario www-data (el que ejecuta Apache) y configura los permisos de escritura necesarios en /var/www/html/wordpress/ y /var/www/, lo que garantiza que Apache tenga acceso adecuado. Finalmente, se habilita el módulo rewrite de Apache para permitir reglas de redirección y reescritura de URLs. Este script deja listo el entorno para alojar sitios web dinámicos.

#!/usr/bin/env bash

Script para instalar y configurar PHP 8.3 con extensiones comunes para WordPress en Ubuntu 24.04 LTS

PHPVERSION=8.3

##
##################################

set -e # Detiene el script si ocurre un error en cualquier línea

echo "==> Actualizando índices de paquetes..."

```
sudo apt update

echo "==> Instalando PHP 8.3 y módulos requeridos..."
sudo apt install php8.3 -y                      # PHP base
sudo apt install libapache2-mod-php8.3 -y       # Integración de PHP con Apache
sudo apt install php8.3-mysql -y                # Conector de PHP con MySQL
sudo apt install php8.3-cli -y                  # Interfaz de línea de comandos para PHP
sudo apt install php8.3-curl -y                 # Soporte para transferencias cURL
sudo apt install php8.3-gd -y                   # Librería gráfica para manipulación de imágenes
sudo apt install php8.3-mbstring -y             # Soporte para cadenas multibyte
sudo apt install php8.3-xml -y                  # Manejo de XML
# Nota: desde PHP 8.0 el módulo xmlrpc fue eliminado del core.
# sudo apt install php8.3-xmlrpc -y             # Puede no estar disponible
sudo apt install php8.3-zip -y                  # Soporte para archivos comprimidos ZIP
sudo apt install php8.3-bcmath -y               # Operaciones matemáticas de precisión arbitraria
sudo apt install php8.3-intl -y                 # Soporte para internacionalización
sudo apt install php8.3-soap -y                 # Soporte para servicios web SOAP

echo "==> Reiniciando Apache para cargar el módulo PHP 8.3..."
service apache2 restart

echo "------------------------------------------------------------"
echo "PHP instalado. Comprobando versión y módulos cargados:"
php -v                                          # Mostrar versión de PHP
echo
php -m | grep -E 'curl|gd|mbstring|mysql|xml|zip|bcmath|intl|soap'   # Mostrar módulos cargados
echo
echo "¡Listo! PHP 8.3 y sus extensiones están operativos."
```

Instalación adicional de extensiones útiles en un solo paso (algunas ya instaladas antes)

sudo apt install php8.3-{fpm,cli,common,mysql,curl,gd,imagick,mbstring,xml,zip,bcmath,intl,soap,exif,fileinfo,opcache} -y

##

Configuración de parámetros de PHP para optimizar el entorno WordPress

##

##

1. Límites de tiempo, memoria y carga

##

sudo sed -i 's/^max_execution_time = .*/max_execution_time = 300/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^max_execution_time = .*/max_execution_time = 300/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^max_input_time = .*/max_input_time = 600/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^max_input_time = .*/max_input_time = 600/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^memory_limit = .*/memory_limit = 512M/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^memory_limit = .*/memory_limit = 512M/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^post_max_size = .*/post_max_size = 256M/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^post_max_size = .*/post_max_size = 256M/' /etc/php/${PHPVERSION}/cli/php.ini

```
sudo sed -i 's/^upload_max_filesize = .*/upload_max_filesize = 256M/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^upload_max_filesize = .*/upload_max_filesize = 256M/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^max_file_uploads = .*/max_file_uploads = 50/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^max_file_uploads = .*/max_file_uploads = 50/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^max_input_vars = .*/max_input_vars = 3000/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^max_input_vars = .*/max_input_vars = 3000/' /etc/php/${PHPVERSION}/cli/php.ini


#########################################################################
#####

# 2. Zona horaria (ajusta la configuración regional para PHP)

#########################################################################
#####

sudo sed -i 's~^;*date.timezone =.*~date.timezone = America/Santiago~' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's~^;*date.timezone =.*~date.timezone = America/Santiago~' /etc/php/${PHPVERSION}/cli/php.ini


#########################################################################
#####

# 3. Seguridad básica

#########################################################################
#####

sudo sed -i 's/^;*expose_php = .*/expose_php = Off/' /etc/php/${PHPVERSION}/apache2/php.ini  # Oculta versión de PHP en cabeceras

sudo sed -i 's/^;*expose_php = .*/expose_php = Off/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^display_errors = .*/display_errors = Off/' /etc/php/${PHPVERSION}/apache2/php.ini  # Oculta errores al usuario
```

```
sudo sed -i 's/^display_errors = .*/display_errors = Off/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^;*log_errors = .*/log_errors = On/' /etc/php/${PHPVERSION}/apache2/php.ini  # Habilita registro de errores

sudo sed -i 's/^;*log_errors = .*/log_errors = On/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's~^;*error_log = .*~error_log = /var/log/php_errors.log~' /etc/php/${PHPVERSION}/apache2/php.ini # Archivo de log de errores

sudo sed -i 's~^;*error_log = .*~error_log = /var/log/php_errors.log~' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^;*disable_functions = .*/disable_functions = exec,passthru,shell_exec,system,proc_open,popen/' \
                                        /etc/php/${PHPVERSION}/apache2/php.ini  # Deshabilita funciones peligrosas

sudo sed -i 's/^;*disable_functions = .*/disable_functions = exec,passthru,shell_exec,system,proc_open,popen/' \
                                        /etc/php/${PHPVERSION}/cli/php.ini


################################################################################

# 4. Activación de extensiones necesarias si estuvieran comentadas

################################################################################

for EXT in curl ftp fileinfo gd mbstring mysqli zip xml intl bcmath soap exif imagick opcache; do
  sudo sed -i "s/^;extension=${EXT}/extension=${EXT}/" /etc/php/${PHPVERSION}/apache2/php.ini
  sudo sed -i "s/^;extension=${EXT}/extension=${EXT}/" /etc/php/${PHPVERSION}/cli/php.ini
done


################################################################################

# 5. Optimización de OPcache y realpath cache (rendimiento)
```

```
################################################################################
#####

# OPcache mejora el rendimiento de PHP al mantener código precompilado en memoria

sudo sed -i 's/^;*opcache.enable=.*/opcache.enable=1/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^;*opcache.memory_consumption=.*/opcache.memory_consumption=128/' \
                                        /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^;*opcache.interned_strings_buffer=.*/opcache.interned_strings_buffer=16/' \
                                        /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^;*opcache.max_accelerated_files=.*/opcache.max_accelerated_files=10000/' \
                                        /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^;*opcache.revalidate_freq=.*/opcache.revalidate_freq=60/' \
                                        /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^;*opcache.validate_timestamps=.*/opcache.validate_timestamps=1/' \
                                        /etc/php/${PHPVERSION}/apache2/php.ini


# realpath_cache mejora el acceso a archivos al almacenar rutas absolutas

sudo sed -i 's/^;*realpath_cache_size = .*/realpath_cache_size = 4096k/' /etc/php/${PHPVERSION}/apache2/php.ini

sudo sed -i 's/^;*realpath_cache_ttl = .*/realpath_cache_ttl = 120/' /etc/php/${PHPVERSION}/apache2/php.ini


# Replicación de configuración de rendimiento para CLI

sudo sed -i 's/^;*opcache.enable=.*/opcache.enable=1/' /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^;*opcache.memory_consumption=.*/opcache.memory_consumption=128/' \
                                        /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^;*opcache.interned_strings_buffer=.*/opcache.interned_strings_buffer=16/' \
                                        /etc/php/${PHPVERSION}/cli/php.ini

sudo sed -i 's/^;*opcache.max_accelerated_files=.*/opcache.max_accelerated_files=10000/' \
                                        /etc/php/${PHPVERSION}/cli/php.ini
```

```
sudo sed -i 's/^;*opcache.revalidate_freq=.*/opcache.revalidate_freq=60/' \
                                        /etc/php/${PHPVERSION}/cli/php.ini
sudo sed -i 's/^;*opcache.validate_timestamps=.*/opcache.validate_timestamps=1/' \
                                        /etc/php/${PHPVERSION}/cli/php.ini
sudo sed -i 's/^;*realpath_cache_size = .*/realpath_cache_size = 4096k/' /etc/php/${PHPVERSION}/cli/php.ini
sudo sed -i 's/^;*realpath_cache_ttl = .*/realpath_cache_ttl = 120/' /etc/php/${PHPVERSION}/cli/php.ini

################################################################################
# 6. Reiniciar Apache para aplicar los cambios de configuración
################################################################################
sudo service apache2 restart

echo "------------------------------------------------------------"
echo "Parámetros PHP ajustados y Apache reiniciado. ¡Listo para WordPress!"

#!/bin/bash

################################################################################
# Instala Apache2 y herramientas adicionales para autenticación básica y monitoreo
sudo apt install -y apache2
sudo apt install -y apache2-utils

# Reinicia el servicio Apache para aplicar los cambios
service apache2 restart
################################################################################
```

```
#########################################################################################
#####################################
# Copia el archivo de configuración personalizado de WordPress al directorio de sitios disponibles de Apache
sudo cp wordpress.conf /etc/apache2/sites-available/

# Crea el directorio donde se alojará el sitio WordPress
sudo mkdir -p /var/www/wordpress

# Asigna la propiedad del directorio a Apache (usuario www-data)
sudo chown -R www-data:www-data /var/www/wordpress

# Establece permisos: 755 para directorios
find /var/www/wordpress -type d -exec chmod 755 {} \;

# Establece permisos: 644 para archivos
find /var/www/wordpress -type f -exec chmod 644 {} \;

# Habilita el módulo headers, necesario para manejar encabezados HTTP
sudo a2enmod headers

# Habilita los módulos expires y deflate (control de caché y compresión)
sudo a2enmod expires deflate

# Habilita el módulo rewrite, necesario para URLs amigables de WordPress
sudo a2enmod rewrite

# Activa el sitio de WordPress definido en wordpress.conf
```

sudo a2ensite wordpress

Desactiva el sitio por defecto (opcional, elimina "It works")

sudo a2dissite 000-default

Reinicia Apache para aplicar cambios en sitios y módulos

sudo service apache2 restart

##
###################################

Instalacion de R en el servidor

Este script automatiza la instalación de R, RStudio Desktop y RStudio Server en Ubuntu. Inicia con la actualización del sistema y la instalación de utilitarios esenciales como locales, gnupg y software-properties-common. Luego, incorpora bibliotecas de desarrollo requeridas para compilar paquetes R con dependencias en C/C++, gráficos y red (libcurl, libssl, libxml2, entre otras).

Posteriormente, agrega el repositorio oficial de R (cloud.r-project.org) y su clave GPG, asegurando la instalación desde fuentes confiables. Luego instala r-base, que incluye el intérprete de R y herramientas principales.

El script continúa descargando e instalando RStudio Desktop y RStudio Server, ambos mediante archivos .deb. Asegura permisos de ejecución y resuelve dependencias con apt install -f.

Finalmente, dentro de una sesión no interactiva de R, instala los paquetes tidyverse y ggplot2, ampliamente utilizados para análisis de datos y visualización.

Este script permite desplegar un entorno completo de análisis estadístico y científico sobre Ubuntu, listo para usuarios locales y remotos, con todas las dependencias necesarias para un trabajo fluido con R y RStudio.

#!/bin/bash

Script para instalar R, RStudio Desktop y RStudio Server en Ubuntu

Actualiza la lista de paquetes disponibles

sudo apt update

```
# Actualiza todos los paquetes instalados a su última versión
sudo apt upgrade -y

# Instala soporte para localización de idiomas (idiomas, codificaciones)
sudo apt install -y locales

# Instala herramientas necesarias para la gestión de claves y repositorios externos
sudo apt install -y dirmngr
sudo apt install gnupg -y
sudo apt install apt-transport-https -y
sudo apt install ca-certificates -y
sudo apt install software-properties-common -y

# Instala bibliotecas de desarrollo necesarias para compilar paquetes R que dependen de componentes gráficos, red o cifrado

# Conectividad y seguridad
sudo apt install libcurl4-openssl-dev -y
sudo apt install libssl-dev -y
sudo apt install libxml2-dev -y

# Requisitos para visualización, fuentes y gráficos
sudo apt install libfontconfig1-dev -y
sudo apt install libharfbuzz-dev -y
sudo apt install libfribidi-dev -y
sudo apt install libfreetype6-dev -y
sudo apt install libpng-dev -y
sudo apt install libtiff5-dev -y
sudo apt install libjpeg-dev -y
```

```
# (Repetidos, aunque inofensivos; pueden eliminarse si se desea optimizar)
sudo apt install libcurl4-openssl-dev -y
sudo apt install libssl-dev -y
sudo apt install libxml2-dev -y
sudo apt install libfontconfig1-dev -y
sudo apt install libharfbuzz-dev -y
sudo apt install libfribidi-dev -y
sudo apt install libfreetype6-dev -y
sudo apt install libpng-dev -y
sudo apt install libtiff5-dev -y
sudo apt install libjpeg-dev -y

# Herramientas esenciales de compilación (make, gcc, etc.)
sudo apt install build-essential -y

# Herramienta para descargar archivos desde internet
sudo apt install wget -y

# Agrega la clave pública del repositorio de R (CRAN)
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 51716619E084DAB9

# Agrega el repositorio oficial de R para Ubuntu 24.04 "Noble"
sudo add-apt-repository "deb https://cloud.r-project.org/bin/linux/ubuntu noble-cran40/"

# Actualiza la lista de paquetes para incluir los del nuevo repositorio de R
sudo apt update

# Instala la base del lenguaje R
```

```
sudo apt install -y r-base


# Descarga RStudio Desktop (versión específica para Ubuntu 22.04 compatible con muchos entornos)

wget https://download1.rstudio.org/electron/jammy/amd64/rstudio-2025.05.0-496-amd64.deb


# Da permisos de ejecución total al archivo .deb descargado (no estrictamente necesario, pero puede facilitar instalación)

chmod 777 rstudio-2025.05.0-496-amd64.deb


# Instala RStudio Desktop desde el archivo .deb

sudo dpkg -i rstudio-*.deb


# Corrige dependencias faltantes que podrían haber surgido durante la instalación

sudo apt install -f


# Descarga RStudio Server (versión compatible con Ubuntu 22.04)

wget https://download2.rstudio.org/server/jammy/amd64/rstudio-server-2024.04.0-735-amd64.deb


# Instala RStudio Server directamente desde el archivo .deb

sudo apt install ./rstudio-server-2024.04.0-735-amd64.deb


# Abre R desde consola sin guardar la sesión y ejecuta la instalación de paquetes esenciales

sudo R --no-save <<EOF

install.packages("tidyverse", repos="https://cloud.r-project.org")  # Conjunto de paquetes para ciencia de datos

install.packages("ggplot2", repos="https://cloud.r-project.org")    # Paquete de visualización

EOF
```

Configuracion R

Este script tiene como objetivo preparar un entorno completo de análisis de datos y visualización en R mediante la instalación de paquetes clave. Comienza con quarto, herramienta fundamental para la generación de reportes reproducibles y dinámicos en múltiples formatos (HTML, PDF, Word).

Se instala el metapaquete tidyverse, que agrupa herramientas esenciales como ggplot2, dplyr, tidyr, readr, purrr y tibble, facilitando tareas de importación, manipulación, visualización y organización de datos. Para análisis exploratorio rápido, se incorpora skimr, que ofrece estadísticas descriptivas enriquecidas.

Se refuerza el enfoque visual con ggplot2, base de la gramática de gráficos en R, junto con GGally, que extiende sus capacidades para crear gráficos multivariantes como matrices de dispersión. plotly añade interactividad a los gráficos, permitiendo zoom, tooltips y exportaciones dinámicas.

Además, se incluye shiny, un framework para construir aplicaciones web interactivas directamente en R, ideal para compartir análisis en línea. Finalmente, scatterplot3d y rgl permiten crear gráficos tridimensionales estáticos e interactivos, ampliando las capacidades visuales para datos multivariantes. En conjunto, este script configura un entorno robusto y versátil para ciencia de datos en R.

```
##############################################################################

#instalacion que quarto

install.packages("quarto")

##############################################################################


##############################################################################

# Manipulación, exploración y visualización de datos.

# ggplot2 (gráficos),

# dplyr (manipulación de datos),

# tidyr (reorganización de datos),

# readr (lectura de archivos),

# purrr (programación funcional),

# tibble (estructuras de datos mejoradas).

install.packages("tidyverse")

##############################################################################
```

```
#############################################################################
#resumen estadístico rápido
# La función principal skim() ofrece una alternativa enriquecida a summary()
install.packages("skimr")
#############################################################################

#############################################################################
# el sistema más popular en R para la creación de gráficos.
# Se basa en la gramática de los gráficos (Grammar of Graphics)
# Es el estándar en visualización de datos en R.
install.packages("ggplot2")
#############################################################################

#############################################################################
# extensión de ggplot2 que facilita la creación de gráficos multivariantes
# Gráficos de pares (ggpairs()),
install.packages("GGally")
#############################################################################
# paquete que convierte gráficos de ggplot2 o crea desde cero gráficos
# Permite:
# Zoom,
# Desplazamiento,
# Tooltips (etiquetas flotantes),
# Exportación interactiva.
install.packages("plotly")
#############################################################################

#############################################################################
```

Instala shiny, un framework de R para construir aplicaciones web interactivas.

install.packages("shiny")

##

install.packages("scatterplot3d")

install.packages("rgl")

Prueba de R

Este script tiene como objetivo realizar una inspección preliminar de un conjunto de datos en R. Inicia cargando bibliotecas esenciales: tidyverse (para manipulación y visualización), skimr (para resumen estadístico), GGally (para gráficos multivariantes) y dplyr (para manejo eficiente de datos).

El archivo Data.csv se importa mediante read.csv, indicando que los datos están codificados en UTF-8, separados por comas y contienen encabezados. Una vez cargado, se ejecutan dos funciones clave para el análisis exploratorio inicial:

head(datos): muestra las primeras filas del dataset, permitiendo una vista rápida del contenido y posibles anomalías.

glimpse(datos): proporciona una descripción compacta de la estructura del conjunto de datos, incluyendo nombres de columnas, tipos de datos y ejemplos de valores.

Este proceso es fundamental para verificar la correcta carga del archivo, entender la naturaleza de las variables y preparar futuras etapas de limpieza, análisis o modelado. En resumen, el script establece una base sólida para el análisis estadístico, asegurando comprensión y control del contenido desde el inicio.

library(tidyverse)

library(skimr)

library(GGally)

library(dplyr)

datos <- read.csv("Data.csv", header = TRUE, sep = "," , encoding = "UTF-8") # Carga de Datos

Ver las primeras filas

head(datos)

```
# Ver la estructura de los datos

glimpse(datos)
```

Exploracion preliminar de Datos

Este script tiene como objetivo realizar un examen preliminar y enriquecer un conjunto de datos en R para facilitar su análisis. Inicia cargando la librería dplyr y leyendo un archivo CSV separado por punto y coma (read.csv2), validando que contenga registros (stopifnot).

Luego convierte variables codificadas (como género, comuna y tipo de violencia) a formato numérico, y utiliza mutate con case_when para traducir dichos códigos en textos descriptivos. Así, se generan nuevas columnas que identifican el género de la víctima y del agresor (Hombre, Mujer u Otro), el nombre de la comuna correspondiente a cada código y una descripción textual del tipo de violencia, que abarca dimensiones simbólicas, psicológicas, sexuales, físicas y estructurales.

Este proceso transforma datos crudos en una versión más comprensible y analíticamente útil. Finalmente, los datos enriquecidos se guardan en un nuevo archivo (Data_modificado.csv), listo para visualización, estadística descriptiva o modelado. En resumen, el script mejora la legibilidad del dataset, facilitando su análisis posterior con herramientas estadísticas o gráficas.

```
##############################################################################
#

# Cargar librerías

library(dplyr)


# Leer el archivo correctamente (con separador ";")

datos <- read.csv2("Data.csv", encoding = "UTF-8", stringsAsFactors = FALSE)


# Verificar que el archivo tenga datos

stopifnot(nrow(datos) > 0)


# Convertir columnas a numéricas desde texto

datos$Genero.Victima <- as.numeric(as.character(datos$Genero.Victima))

datos$Genero.Agresor <- as.numeric(as.character(datos$Genero.Agresor))

datos$Comuna <- as.numeric(as.character(datos$Comuna))

datos$Tipo <- as.numeric(as.character(datos$Tipo))
```

```
# Agregar columnas con texto interpretado para Género de Víctima y Agresor
datos <- datos %>%
  mutate(
    Nombre_Genero_Victima_Texto = case_when(
      Genero.Victima == 0 ~ "Hombre",
      Genero.Victima == 1 ~ "Mujer",
      Genero.Victima == 2 ~ "Otro",
      TRUE ~ NA_character_
    ),
    Nombre_Genero_Agresor_Texto = case_when(
      Genero.Agresor == 0 ~ "Hombre",
      Genero.Agresor == 1 ~ "Mujer",
      Genero.Agresor == 2 ~ "Otro",
      TRUE ~ NA_character_
    )
  )

# Agregar columna con nombre de comuna
datos <- datos %>%
  mutate(Nombre_Comuna = case_when(
    Comuna == 13101 ~ "Santiago",
    Comuna == 13102 ~ "Cerrillos",
    Comuna == 13103 ~ "Cerro Navia",
    Comuna == 13104 ~ "Conchalí",
    Comuna == 13105 ~ "El Bosque",
    Comuna == 13106 ~ "Estación Central",
    Comuna == 13107 ~ "Huechuraba",
```

Comuna == 13108 ~ "Independencia",

Comuna == 13109 ~ "La Cisterna",

Comuna == 13110 ~ "La Florida",

Comuna == 13111 ~ "La Granja",

Comuna == 13112 ~ "La Pintana",

Comuna == 13113 ~ "La Reina",

Comuna == 13114 ~ "Las Condes",

Comuna == 13115 ~ "Lo Barnechea",

Comuna == 13116 ~ "Lo Espejo",

Comuna == 13117 ~ "Lo Prado",

Comuna == 13118 ~ "Macul",

Comuna == 13119 ~ "Maipú",

Comuna == 13120 ~ "Ñuñoa",

Comuna == 13121 ~ "Pedro Aguirre Cerda",

Comuna == 13122 ~ "Peñalolén",

Comuna == 13123 ~ "Providencia",

Comuna == 13124 ~ "Pudahuel",

Comuna == 13125 ~ "Quilicura",

Comuna == 13126 ~ "Quinta Normal",

Comuna == 13127 ~ "Recoleta",

Comuna == 13128 ~ "Renca",

Comuna == 13129 ~ "San Joaquín",

Comuna == 13130 ~ "San Miguel",

Comuna == 13131 ~ "San Ramón",

Comuna == 13132 ~ "Vitacura",

Comuna == 13201 ~ "Puente Alto",

Comuna == 13202 ~ "Pirque",

Comuna == 13203 ~ "San José de Maipo",

Comuna == 13301 ~ "Colina",

```
    Comuna == 13302 ~ "Lampa",
    Comuna == 13303 ~ "Tiltil",
    Comuna == 13401 ~ "San Bernardo",
    Comuna == 13402 ~ "Buin",
    Comuna == 13403 ~ "Calera de Tango",
    Comuna == 13404 ~ "Paine",
    Comuna == 13501 ~ "Melipilla",
    Comuna == 13502 ~ "Alhué",
    Comuna == 13503 ~ "Curacaví",
    Comuna == 13504 ~ "María Pinto",
    Comuna == 13505 ~ "San Pedro",
    Comuna == 13601 ~ "Talagante",
    Comuna == 13602 ~ "El Monte",
    Comuna == 13603 ~ "Isla de Maipo",
    Comuna == 13604 ~ "Padre Hurtado",
    Comuna == 13605 ~ "Peñaflor",
    TRUE ~ NA_character_
  ))



# Agregar columna con nombre de Violencia
datos <- datos %>%
  mutate(Nombre_Violencia = case_when(
    Tipo == 1 ~ "Lenguaje discriminatorio (sexista, racista, homofóbico, etc.)",
    Tipo == 2 ~ "Chistes, burlas o memes ofensivos sobre género o identidad",
    Tipo == 3 ~ "Estereotipos de género o roles impuestos",
    Tipo == 4 ~ "Interrupciones o invisibilización en espacios públicos o laborales",
    Tipo == 5 ~ "Expectativas sociales rígidas sobre cómo debe ser un hombre/mujer/persona",
    Tipo == 6 ~ "Cosificación o reducción de una persona a su apariencia física",
```

Tipo == 7 ~ "Falta de representación o inclusión en medios, política o educación",

Tipo == 8 ~ "Apropiación de ideas u opiniones en entornos profesionales",

Tipo == 9 ~ "Descalificación constante o humillación verbal",

Tipo == 10 ~ "Manipulación emocional o gaslighting",

Tipo == 11 ~ "Aislamiento social o familiar intencionado",

Tipo == 12 ~ "Amenazas verbales (directas o indirectas)",

Tipo == 13 ~ "Culpar a la víctima por el conflicto o por la violencia sufrida",

Tipo == 14 ~ "Desvalorización de logros o capacidades personales",

Tipo == 15 ~ "Fomento de inseguridades o dependencia emocional",

Tipo == 16 ~ "Uso de hijos/as u otras personas como chantaje emocional",

Tipo == 17 ~ "Control o prohibición del uso del dinero propio",

Tipo == 18 ~ "Negarle acceso a estudios, trabajo o recursos económicos",

Tipo == 19 ~ "Retención o robo de bienes personales o dinero",

Tipo == 20 ~ "Endeudamiento forzoso a nombre de la víctima",

Tipo == 21 ~ "Destrucción de objetos personales o simbólicos",

Tipo == 22 ~ "Negación de recursos básicos (ropa, medicamentos, alimentos)",

Tipo == 23 ~ "Comentarios sexuales no deseados",

Tipo == 24 ~ "Miradas o gestos lascivos",

Tipo == 25 ~ "Tocamientos sin consentimiento",

Tipo == 26 ~ "Presión o chantaje para relaciones sexuales",

Tipo == 27 ~ "Relaciones sexuales sin consentimiento pleno",

Tipo == 28 ~ "Envío o difusión de imágenes sexuales sin consentimiento",

Tipo == 29 ~ "Exhibicionismo o voyeurismo sin consentimiento",

Tipo == 30 ~ "Acoso sexual en el trabajo o espacios educativos",

Tipo == 31 ~ "Violación sexual (con o sin uso de fuerza física)",

Tipo == 32 ~ "Explotación sexual o trata de personas",

Tipo == 33 ~ "Empujones o sacudidas leves",

Tipo == 34 ~ "Bofetadas, tirones de cabello o rasguños",

Tipo == 35 ~ "Golpes con objetos, puñetazos, patadas",

```
  Tipo == 36 ~ "Lesiones físicas moderadas o graves",
  Tipo == 37 ~ "Tortura física o castigos corporales continuados",
  Tipo == 38 ~ "Privación de atención médica, alimentos o sueño",
  Tipo == 39 ~ "Encierro o inmovilización forzada",
  Tipo == 40 ~ "Intentos de estrangulamiento o ahogo",
  Tipo == 41 ~ "Amenazas de muerte o daño a terceros",
  Tipo == 42 ~ "Persecución o acoso persistente (stalking)",
  Tipo == 43 ~ "Intentos de homicidio o feminicidio",
  Tipo == 44 ~ "Homicidio por razones de género, orientación o identidad",
  Tipo == 45 ~ "Suicidio inducido por violencia prolongada o sistemática",
  Tipo == 46 ~ "Negación de acceso a justicia por prejuicio o discriminación",
  Tipo == 47 ~ "Trato deshumanizante en servicios públicos o de salud",
  Tipo == 48 ~ "Criminalización o patologización de la identidad de género",
  Tipo == 49 ~ "Violencia policial, carcelaria o estatal por razón de identidad",
  Tipo == 50 ~ "Falta de políticas o protección efectiva frente a la violencia",

  TRUE ~ NA_character_
 ))

# Guardar el archivo modificado
write.csv2(datos, "Data_modificado.csv", row.names = FALSE)
##############################################################################
#
```

Cambiar Datos

Este script tiene como propósito realizar una limpieza textual de un conjunto de datos previamente procesado (Data_modificado.csv) utilizando R. Comienza cargando las librerías readr y dplyr, y luego importa los datos asegurando una correcta codificación en UTF-8.

Antes de proceder, se valida la existencia de tres columnas clave: Nombre_Comuna, Comuna.1 y Nombre_Violencia. Esto garantiza que las operaciones siguientes se apliquen sobre campos existentes y relevantes.

El núcleo del script es la función limpiar_texto, la cual reemplaza caracteres especiales como tildes (í, ó, é, ú) y la letra ñ por sus equivalentes sin diacríticos. Esto es útil para evitar problemas en análisis posteriores o al trabajar con sistemas que no manejan bien caracteres acentuados.

La función se aplica a las columnas mencionadas, normalizando su contenido textual. Finalmente, el dataset limpio se guarda como Data_modificado_sin_tildes.csv, conservando la codificación UTF-8. Este archivo resulta ideal para análisis estadístico, visualizaciones o procesos automatizados que requieren datos consistentes y sin caracteres especiales.

```
################################################################################
#

# Cargar librerías

library(readr)

library(dplyr)


# Leer archivo

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)


# Verificar que las columnas existen

cols_necesarias <- c("Nombre_Comuna", "Comuna.1", "Nombre_Violencia")

stopifnot(all(cols_necesarias %in% names(datos)))


# Función de limpieza

limpiar_texto <- function(texto) {

  texto <- gsub("Ñ", "N", texto)

  texto <- gsub("ñ", "n", texto)

  texto <- gsub("í", "i", texto)

  texto <- gsub("ó", "o", texto)

  texto <- gsub("é", "e", texto)

  texto <- gsub("ú", "u", texto)

  return(texto)

}
```

```
# Aplicar limpieza a las columnas relevantes
datos$Nombre_Comuna    <- limpiar_texto(datos$Nombre_Comuna)
datos$Comuna.1         <- limpiar_texto(datos$Comuna.1)
datos$Nombre_Violencia  <- limpiar_texto(datos$Nombre_Violencia)


# Guardar archivo limpio
write.csv2(datos, "Data_modificado.csv", fileEncoding = "UTF-8", row.names = FALSE)
```

##
#

001 Grafico Preliminar

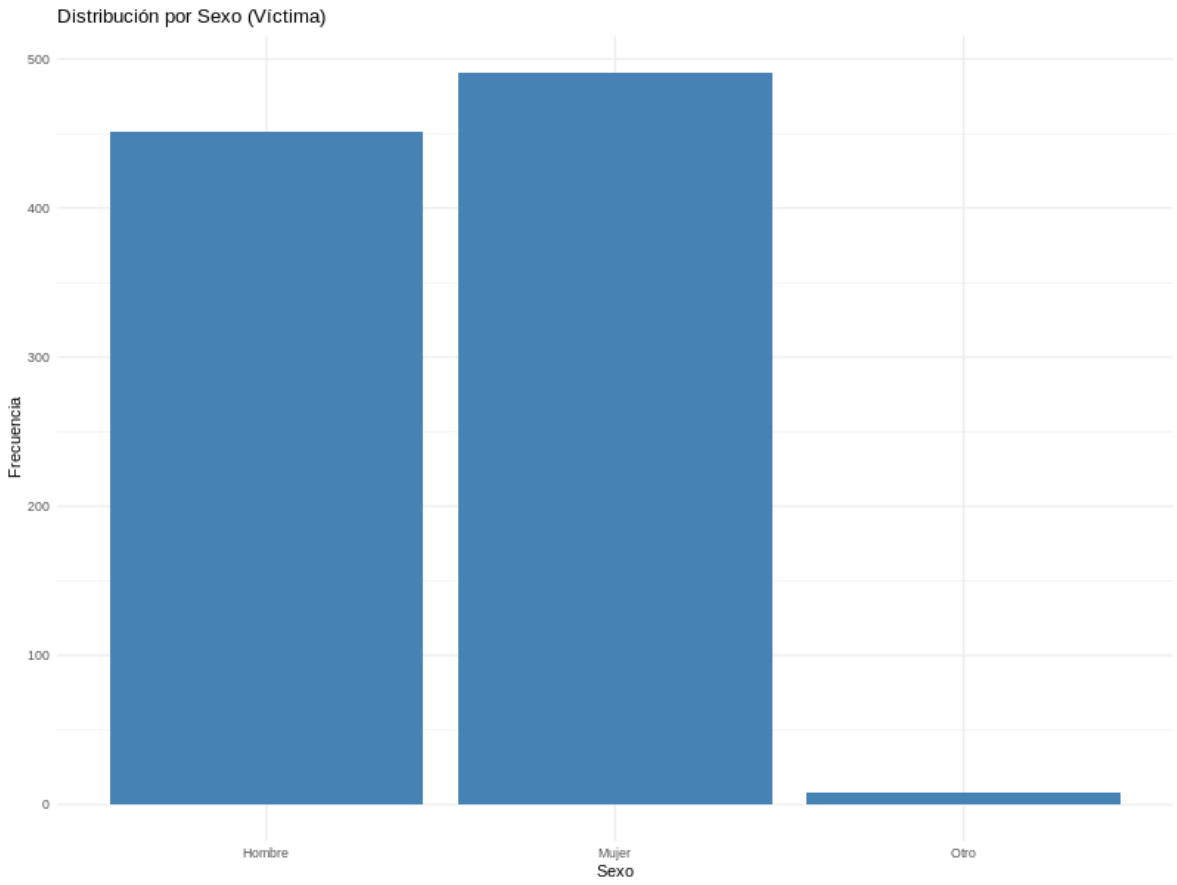
El código en R realiza una visualización básica de la distribución de víctimas según su género. Primero, carga un archivo CSV llamado "Data_modificado.csv", asegurándose de que los textos se lean correctamente y sin convertir cadenas a factores automáticamente. Luego, transforma la columna Genero.Victima en un factor con tres niveles etiquetados como "Hombre", "Mujer" y "Otro", asignando etiquetas más descriptivas a los valores codificados 0, 1 y 2.

Posteriormente, crea un gráfico de barras usando ggplot2, donde el eje x representa el género de las víctimas y el eje y la frecuencia de cada categoría. Las barras se rellenan con un color azul ("steelblue") y se aplica un estilo minimalista con theme_minimal(). Finalmente, el gráfico se guarda como una imagen PNG con dimensiones de 800 por 600 píxeles bajo el nombre "001_genero_victima.png".

##
#

```
# Leer el archivo CSV modificado
datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)


# Convertir columna Genero.Victima a factor con etiquetas
datos$Genero.Victima <- factor(datos$Genero.Victima,
                               levels = c(0, 1, 2),
                               labels = c("Hombre", "Mujer", "Otro"))


# Crear gráfico y guardar como imagen PNG
```

```
png("001_genero_victima.png", width = 800, height = 600)

ggplot(datos, aes(x = Genero.Victima)) +
  geom_bar(fill = "steelblue") +
  labs(title = "Distribución por Sexo (Víctima)",
       x = "Sexo",
       y = "Frecuencia") +
  theme_minimal()

dev.off()

################################################################################
#
```



```
# Cargar librerías
library(ggplot2)
library(dplyr)
library(readr)

# Leer archivo
datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)

# Convertir fecha
datos$Fecha <- as.POSIXct(datos$Fecha, format = "%d-%m-%Y %H:%M", tz = "UTC")

# Filtrar filas con valores clave no faltantes
datos <- datos %>% filter(!is.na(Nombre_Genero_Victima_Texto), !is.na(Edad), !is.na(Nombre_Comuna))

# 1. Barras por género de la víctima
graf1 <- ggplot(datos, aes(x = Nombre_Genero_Victima_Texto)) +
  geom_bar(fill = "steelblue") +
  labs(title = "Casos por género de la víctima", x = "Género", y = "Cantidad")
ggsave("001_grafico_genero_victima.png", plot = graf1, width = 6, height = 4, dpi = 300)

# 2. Barras agrupadas víctima vs agresor
graf2 <- ggplot(datos, aes(x = Nombre_Genero_Victima_Texto, fill = Nombre_Genero_Agresor_Texto)) +
  geom_bar(position = "dodge") +
  labs(title = "Víctima vs Agresor por género", x = "Víctima", fill = "Agresor")
ggsave("001_grafico_victima_agresor.png", plot = graf2, width = 6, height = 4, dpi = 300)

# 3. Barras por comuna
graf3 <- ggplot(datos, aes(x = Nombre_Comuna)) +
```

```
  geom_bar(fill = "darkorange") +
  labs(title = "Casos por comuna", x = "Comuna", y = "Cantidad") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
ggsave("001_grafico_comuna.png", plot = graf3, width = 8, height = 5, dpi = 300)

# 4. Dispersión Edad vs Comuna
graf4 <- ggplot(datos, aes(x = Edad, y = Nombre_Comuna, color = Nombre_Genero_Victima_Texto)) +
  geom_point(size = 2) +
  labs(title = "Edad de víctimas por comuna", x = "Edad", y = "Comuna")
ggsave("001_grafico_edad_vs_comuna.png", plot = graf4, width = 6, height = 5, dpi = 300)

# 5. Pastel por género víctima
tabla_victima <- datos %>%
  group_by(Nombre_Genero_Victima_Texto) %>%
  summarise(Frecuencia = n())

graf5 <- ggplot(tabla_victima, aes(x = "", y = Frecuencia, fill = Nombre_Genero_Victima_Texto)) +
  geom_col() +
  coord_polar("y", start = 0) +
  labs(title = "Proporción por género de víctima", fill = "Género")
ggsave("001_grafico_pastel_genero.png", plot = graf5, width = 5, height = 5, dpi = 300)

# 6. Histograma de casos por fecha
graf6 <- ggplot(datos, aes(x = Fecha)) +
  geom_histogram(binwidth = 86400, fill = "purple") +
  labs(title = "Distribución temporal de casos", x = "Fecha", y = "Número de casos")
ggsave("001_grafico_casos_por_fecha.png", plot = graf6, width = 7, height = 4, dpi = 300)
```

Gráfico de barras: Casos por género de la víctima Código: graf1

Eje X: Género de la víctima (Nombre_Genero_Victima_Texto).

Eje Y: Cantidad de casos registrados.

Descripción: Representa cuántos casos se registraron para cada género de víctima. Se visualiza fácilmente cuál es el género más afectado.

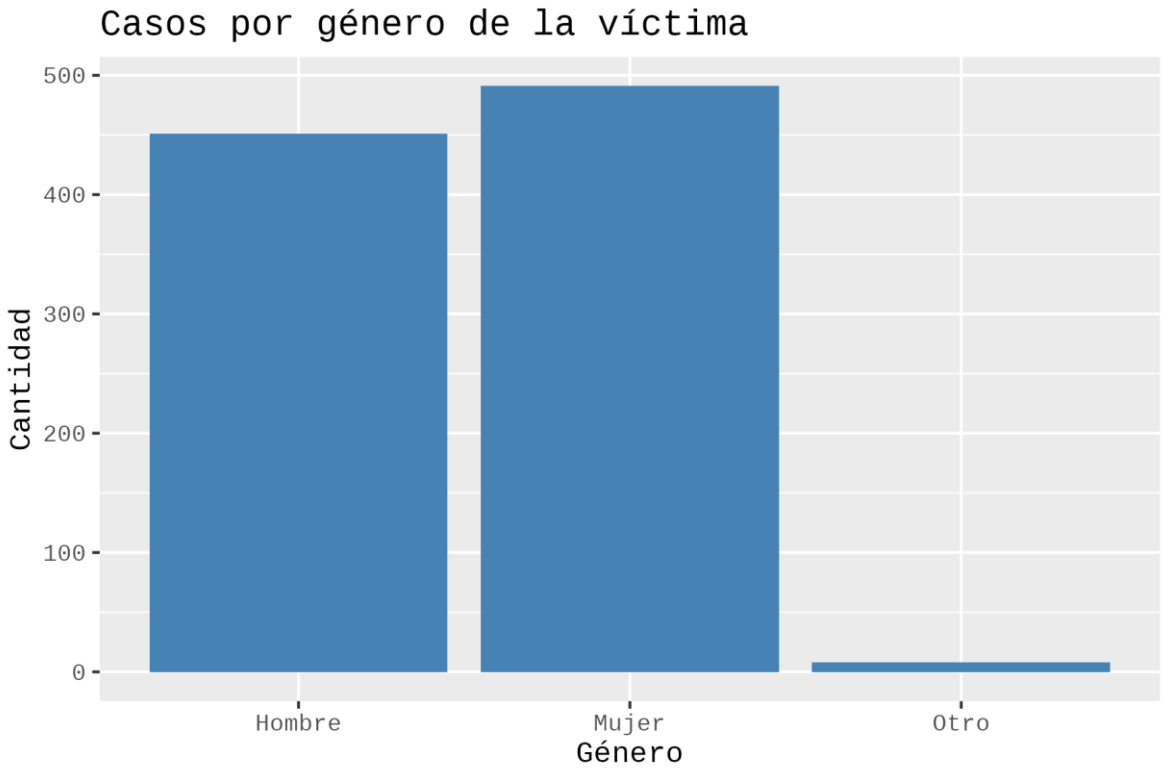
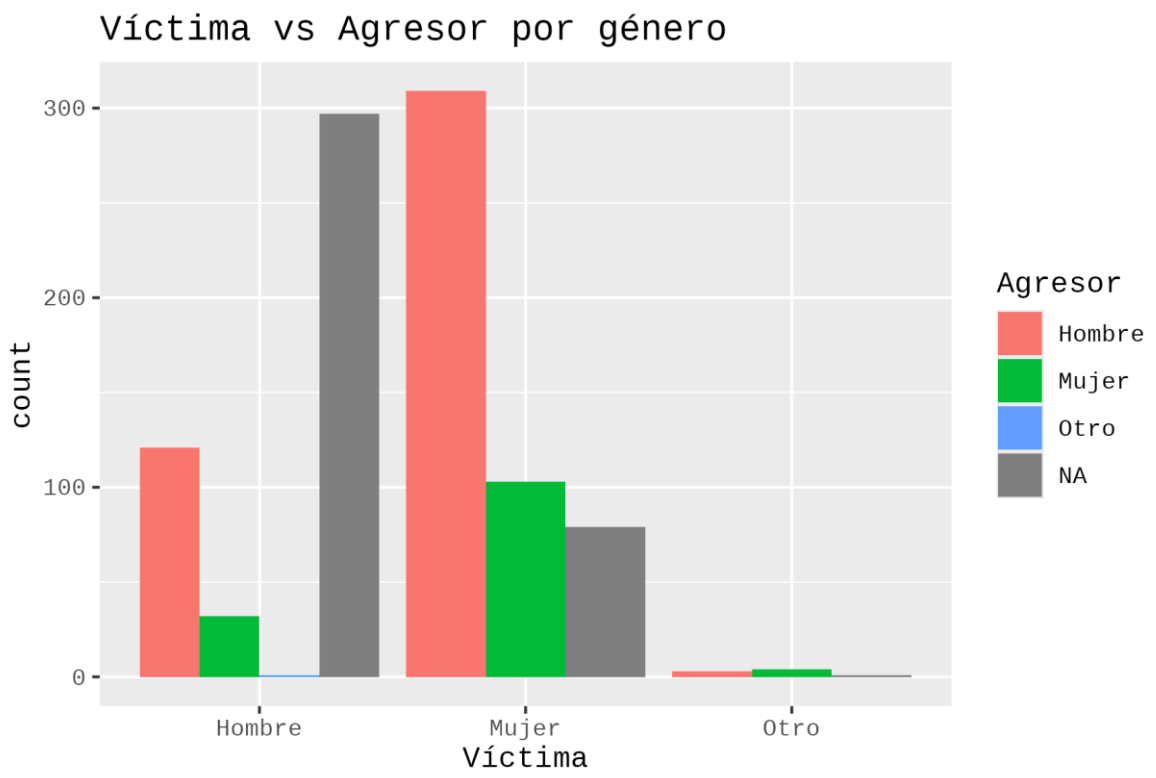


Gráfico de barras agrupadas: Víctima vs Agresor por género Código: graf2

Eje X: Género de la víctima.

Colores (fill): Género del agresor (Nombre_Genero_Agresor_Texto).

Descripción: Muestra cómo se distribuyen los agresores por género, según el género de la víctima. Útil para analizar relaciones de género en los casos de violencia.

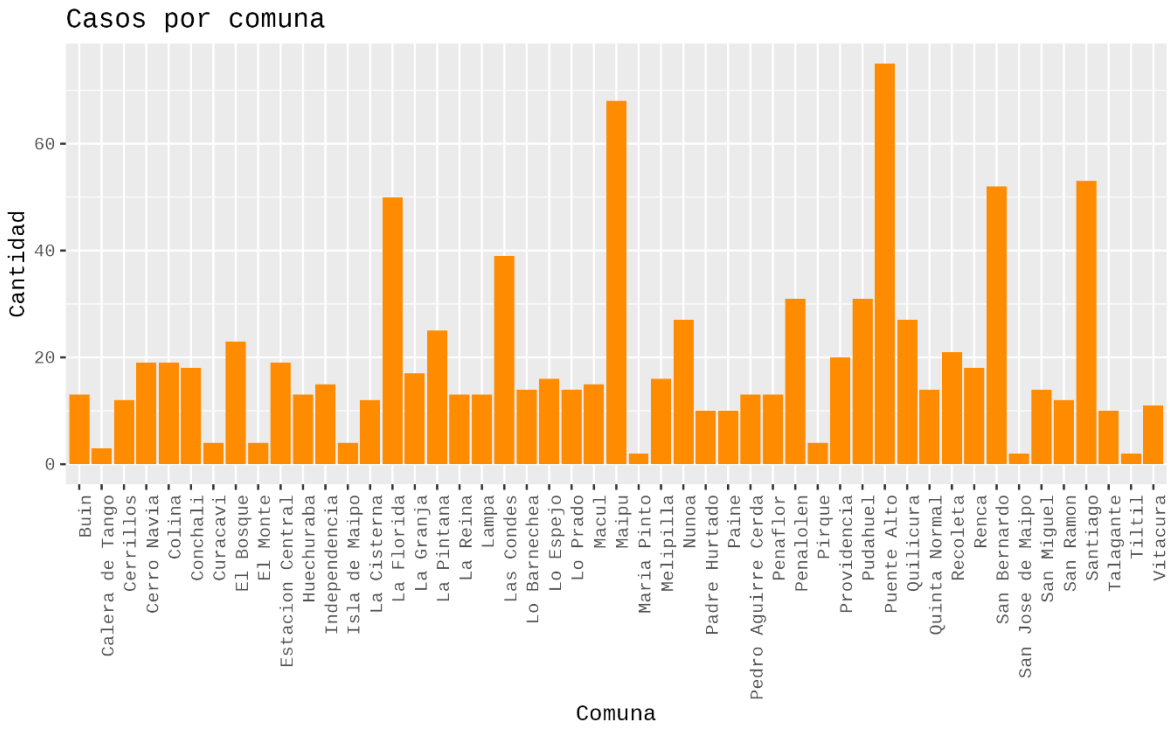


Código: graf3

Eje X: Nombre de la comuna (Nombre_Comuna).

Eje Y: Frecuencia de casos.

Descripción: Permite identificar en qué comunas se reportan más casos. Las etiquetas están giradas para facilitar la lectura.



Código: graf4

Eje X: Edad de la víctima.

Eje Y: Comuna del hecho.

Colores: Género de la víctima.

Descripción: Muestra cómo se distribuyen las edades de las víctimas según la comuna, diferenciando los géneros con colores. Útil para análisis cruzado geográfico-etario.

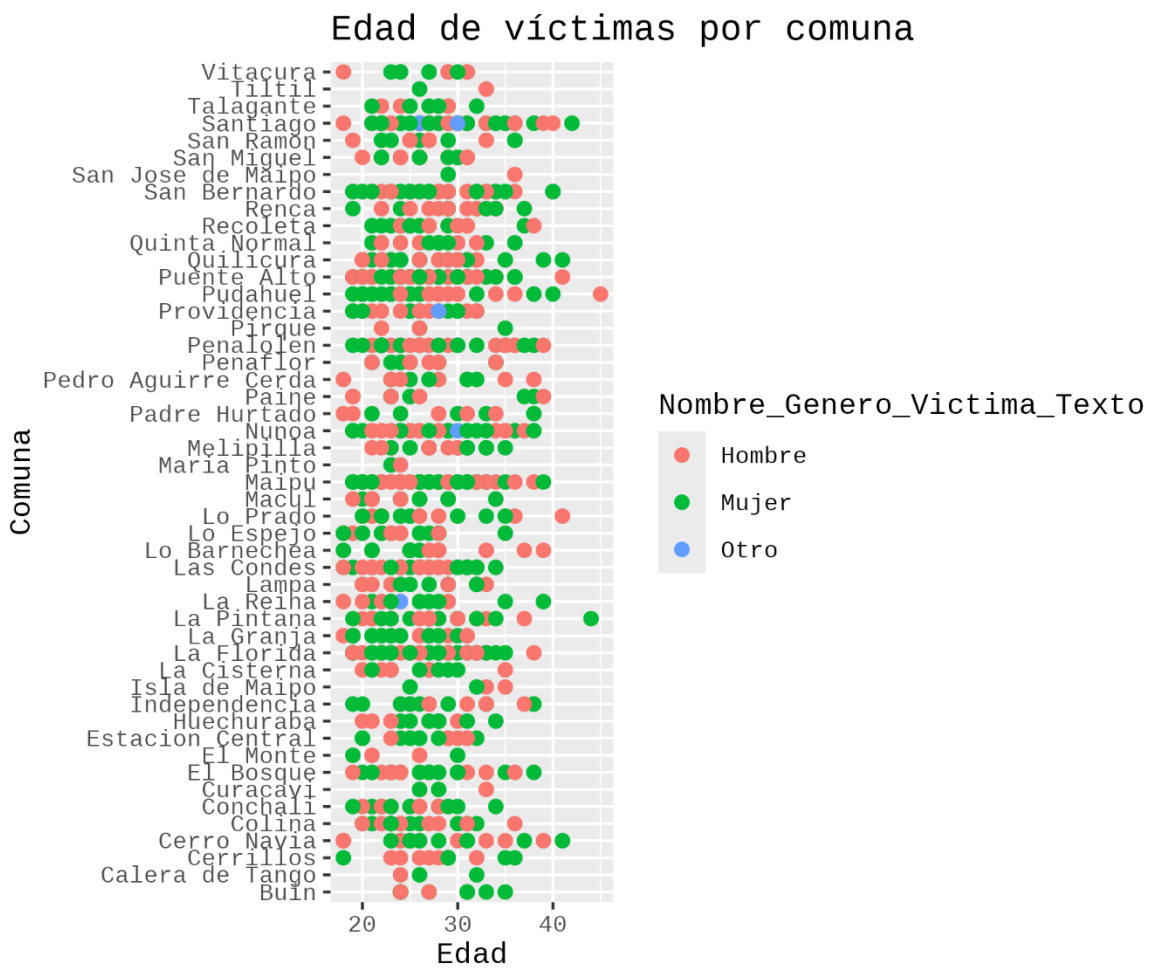
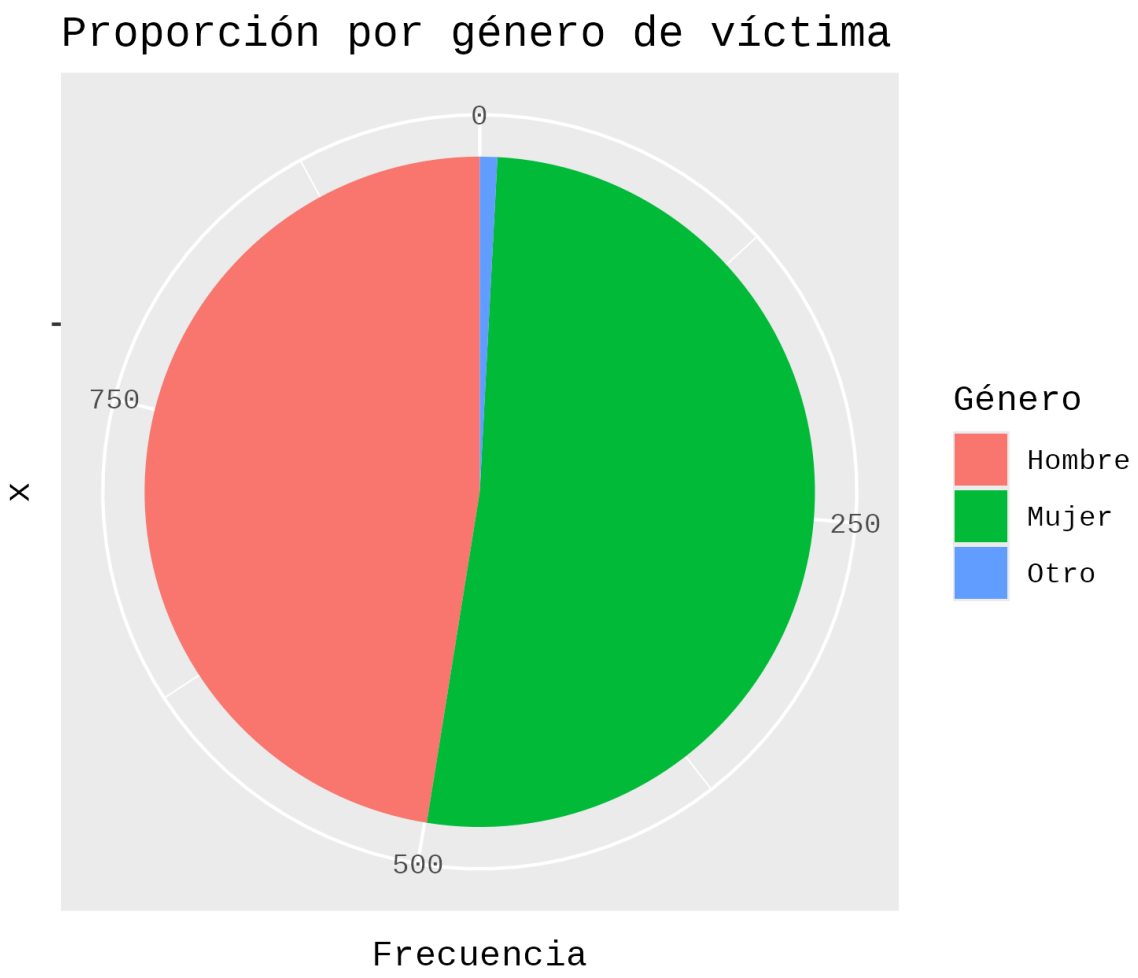


Gráfico de pastel: Proporción por género de víctima Código: graf5

Segmentos: Frecuencia de casos por género de víctima.

Descripción: Presenta visualmente la proporción de casos entre los distintos géneros. Complementa al gráfico de barras (graf1) pero con énfasis porcentual.

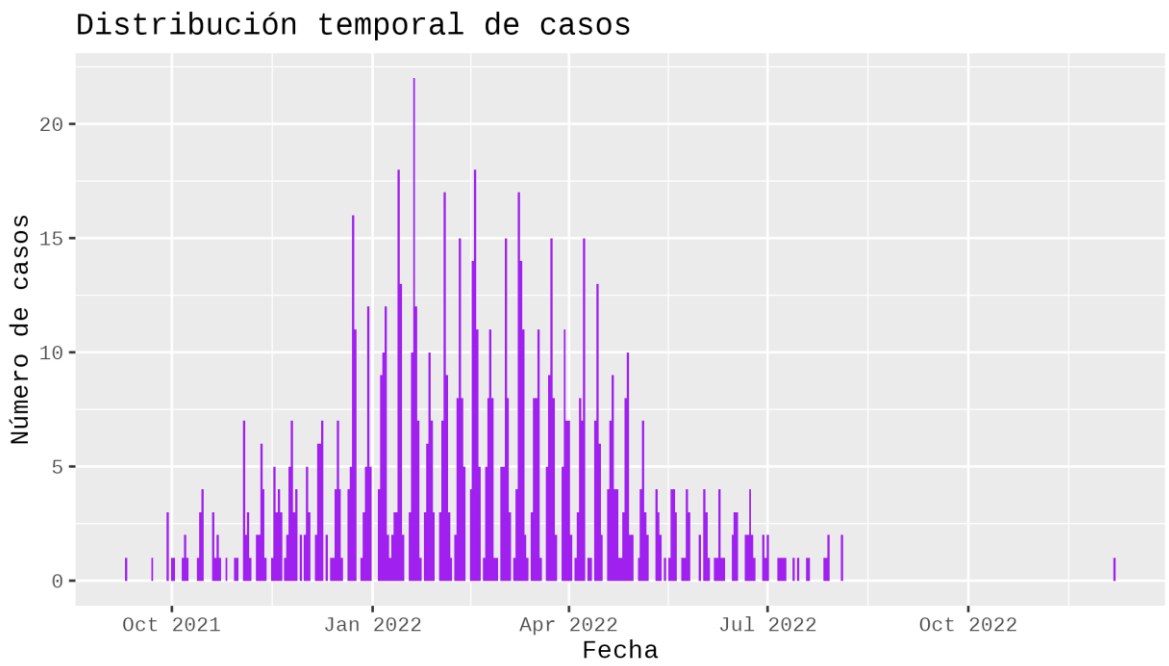


Histograma: Distribución temporal de casos Código: graf6

Eje X: Fecha del incidente (convertida a formato POSIXct).

Eje Y: Número de casos por día.

Descripción: Muestra cómo se distribuyen los casos en el tiempo. El binwidth de 86400 segundos (1 día) permite observar la frecuencia diaria de reportes.



002 Distribucion de registros por comuna

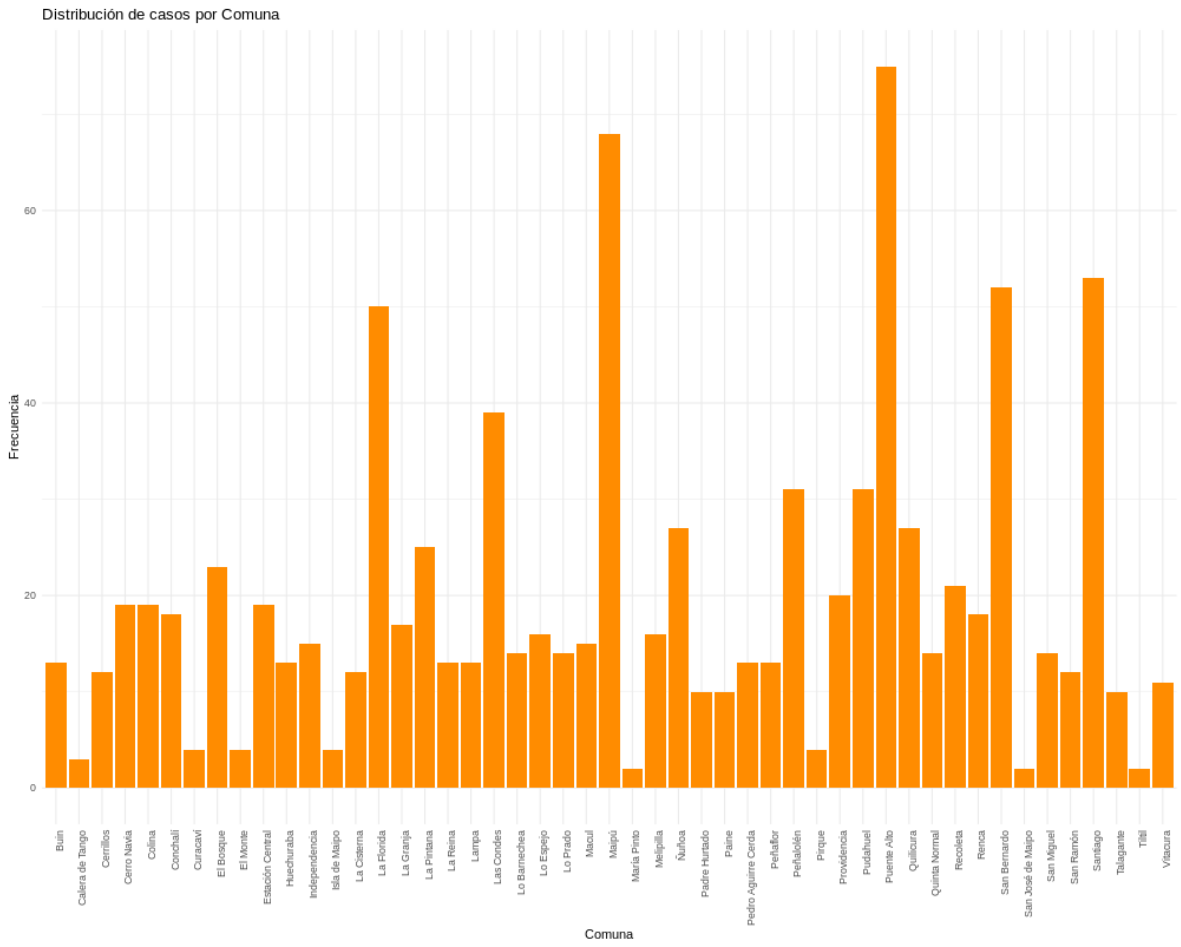
Este código en R genera un gráfico de barras que muestra la distribución de casos por comuna a partir de un archivo CSV llamado "Data_modificado.csv". Primero, se carga el archivo especificando la codificación UTF-8 y evitando que las cadenas se conviertan automáticamente en factores. Luego, utilizando ggplot2, se crea un gráfico donde el eje x representa los nombres de las comunas (Nombre_Comuna) y el eje y la cantidad de registros (frecuencia de casos) en cada una.

Las barras del gráfico se rellenan con color naranja oscuro ("darkorange") y se aplica un estilo visual limpio mediante theme_minimal(). Además, para mejorar la legibilidad del eje x, los nombres de las comunas se rotan 90 grados. Finalmente, el gráfico se guarda como una imagen PNG con dimensiones de 1000 por 800 píxeles bajo el nombre "002_distribucion_comunas.png".

```
#############################################################################
#

# Leer el archivo CSV modificado

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)


# Crear gráfico y guardar como imagen PNG

png("002_distribucion_comunas.png", width = 1000, height = 800)


ggplot(datos, aes(x = Nombre_Comuna)) +
```

```
 geom_bar(fill = "darkorange") +
 labs(title = "Distribución de casos por Comuna",
    x = "Comuna",
    y = "Frecuencia") +
 theme_minimal() +
 theme(axis.text.x = element_text(angle = 90, hjust = 1))  # Rotar nombres

dev.off()
```

##
#



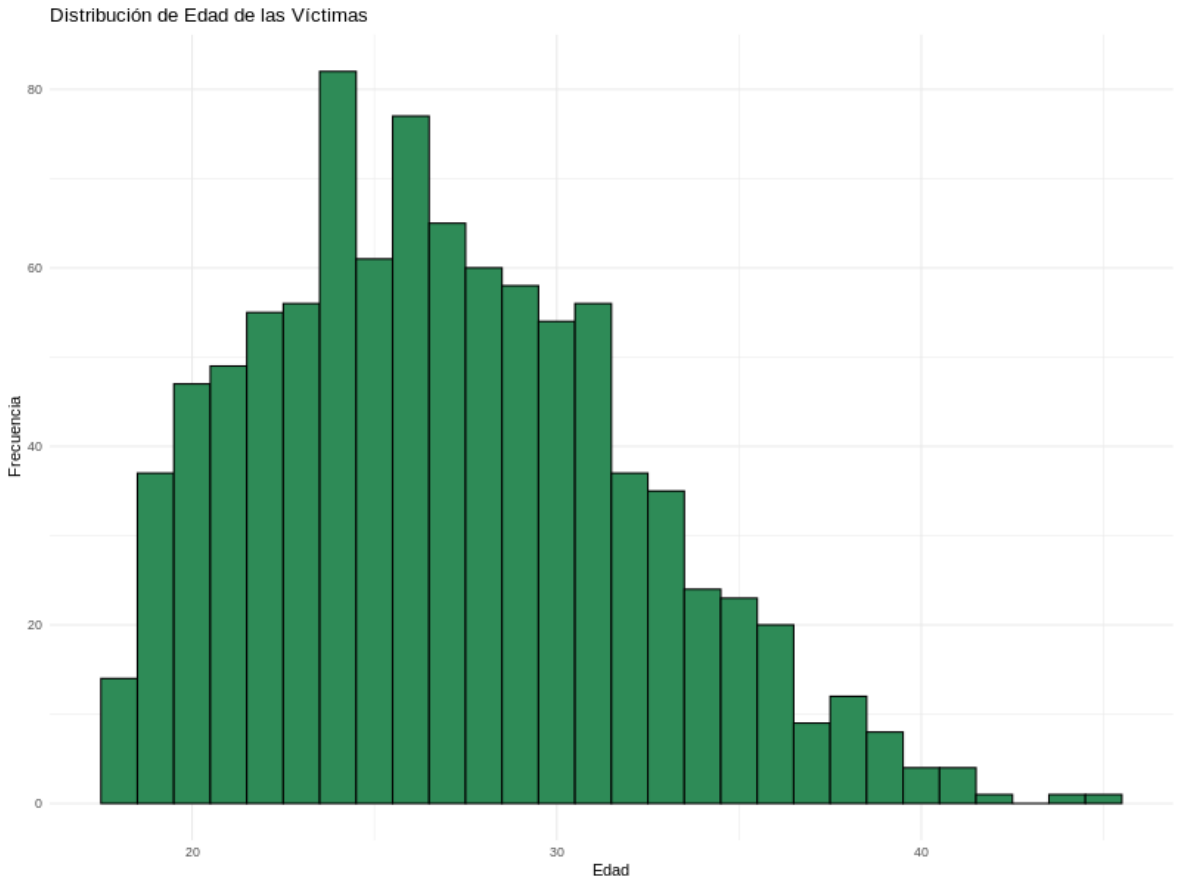
003 Dsitribucion de las edades de las victimas

Este código en R genera un histograma que muestra la distribución de edades de las víctimas utilizando datos del archivo "Data_modificado.csv". Primero, se carga el archivo asegurando una correcta codificación de caracteres y evitando la conversión automática de cadenas a

factores. Luego, se transforma explícitamente la columna Edad a formato numérico, garantizando que se puedan realizar operaciones estadísticas y gráficas sobre ella.

Con ggplot2, se construye un histograma donde el eje x representa las edades y el eje y la frecuencia de aparición de cada valor. Se utiliza un ancho de bin de 1 año para una mayor precisión y detalle, coloreando las barras de verde marino ("seagreen") con bordes negros. Se aplica un tema minimalista para una presentación clara. Finalmente, el gráfico se guarda como imagen PNG de 800 por 600 píxeles con el nombre "003_edad.png".

```
################################################################################
#


# Leer el archivo CSV modificado

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)


# Asegurarse de que la columna Edad sea numérica

datos$Edad <- as.numeric(as.character(datos$Edad))


# Crear gráfico y guardar como imagen PNG

png("003_edad.png", width = 800, height = 600)


ggplot(datos, aes(x = Edad)) +

 geom_histogram(binwidth = 1, fill = "seagreen", color = "black") +

 labs(title = "Distribución de Edad de las Víctimas",

     x = "Edad",

     y = "Frecuencia") +

 theme_minimal()


dev.off()

################################################################################
#
```



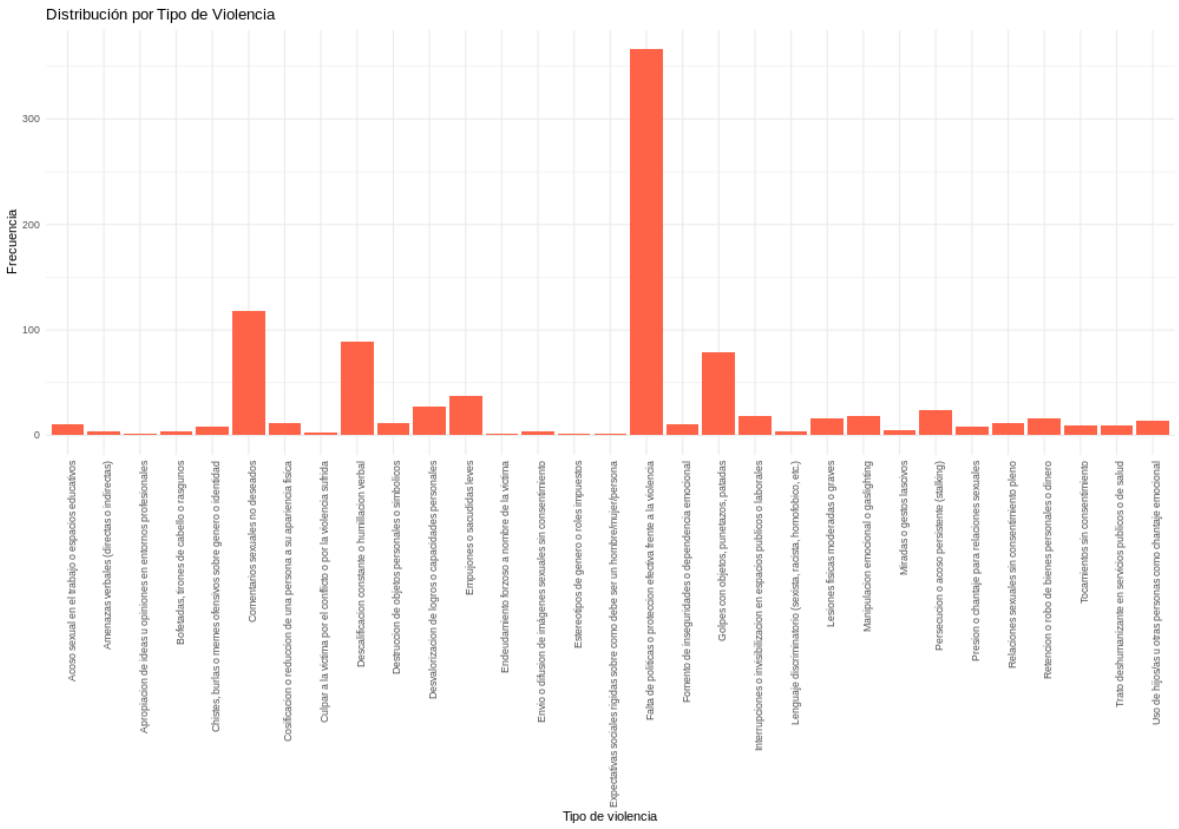
004 Distribución de los distintos tipos de violencia

Este código en R genera un gráfico de barras que representa la distribución de los distintos tipos de violencia registrados en el archivo "Data_modificado.csv". Primero, se lee el archivo con codificación UTF-8 y sin convertir cadenas automáticamente en factores. Luego, se asume que la columna que contiene los tipos de violencia se llama Nombre_Violencia (debe ajustarse si el nombre difiere en los datos reales).

Utilizando ggplot2, se crea un gráfico donde el eje x muestra las distintas categorías de violencia y el eje y su frecuencia. Las barras se visualizan en color rojo tomate ("tomato"), y se aplica un diseño minimalista con theme_minimal(). Para facilitar la lectura, las etiquetas del eje x se rotan 90 grados. Finalmente, el gráfico se guarda como imagen PNG de 1000 por 700 píxeles bajo el nombre "004_registro_violencia.png".

```
##############################################################################
#


# Leer el archivo CSV modificado

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)
```

```
# Verificar que la columna de tipos de violencia existe
# Puedes ajustar el nombre exacto si es diferente
# Supongamos que la columna se llama "Nombre_Violencia"
# Si no existe, reemplazar por el nombre real, por ejemplo: datos$Tipo

# Crear gráfico y guardar como imagen PNG
png("004_registro_violencia.png", width = 1000, height = 700)

ggplot(datos, aes(x = Nombre_Violencia)) +
  geom_bar(fill = "tomato") +
  labs(title = "Distribución por Tipo de Violencia",
       x = "Tipo de violencia",
       y = "Frecuencia") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

dev.off()
################################################################################
#
```



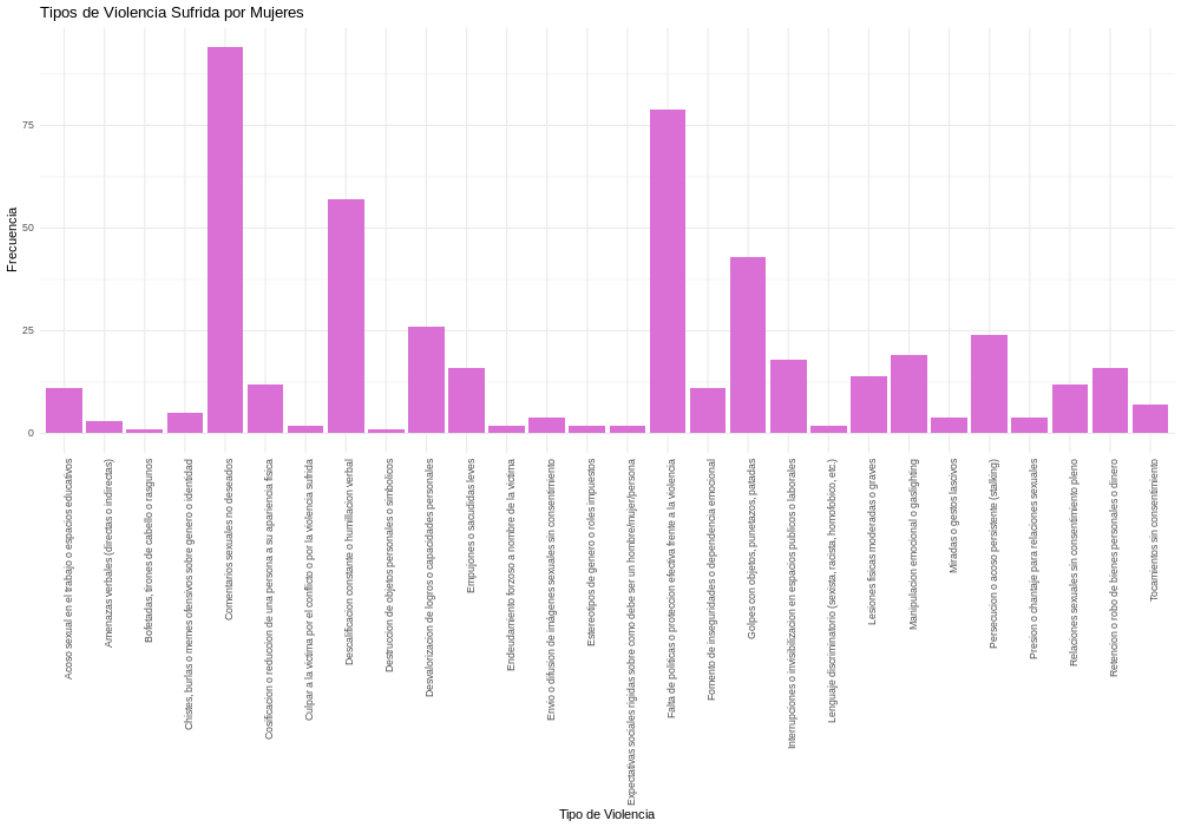
005 Violencia cometida hacia las mujeres

Este código en R genera un gráfico de barras que muestra los tipos de violencia sufridos exclusivamente por mujeres, utilizando datos del archivo "Data_modificado.csv". Primero, se carga el archivo asegurando la correcta codificación de caracteres y sin convertir cadenas a factores automáticamente. Luego, se filtran los registros donde la columna Genero.Victima tiene el valor 1, que representa a mujeres, y se guarda ese subconjunto en datos_mujeres.

Con ggplot2, se crea un gráfico de barras donde el eje x representa los distintos tipos de violencia (Nombre_Violencia) y el eje y su frecuencia. Las barras se muestran en color orquídea ("orchid") y se aplica un diseño limpio usando theme_minimal(). Para mejorar la legibilidad, las etiquetas del eje x se rotan 90 grados. Finalmente, el gráfico se guarda como imagen PNG de 1000 por 700 píxeles con el nombre "005_violencia_hacia_mujeres.png".

```
##############################################################################
#


# Leer el archivo CSV modificado

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)


# Filtrar solo los casos donde la víctima es mujer (Genero.Victima == 1)
```

```
datos_mujeres <- datos %>%
  filter(Genero.Victima == 1)

# Crear gráfico y guardar como imagen PNG
png("005_violencia_hacia_mujeres.png", width = 1000, height = 700)

ggplot(datos_mujeres, aes(x = Nombre_Violencia)) +
  geom_bar(fill = "orchid") +
  labs(title = "Tipos de Violencia Sufrida por Mujeres",
       x = "Tipo de Violencia",
       y = "Frecuencia") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

dev.off()
################################################################################
#
```



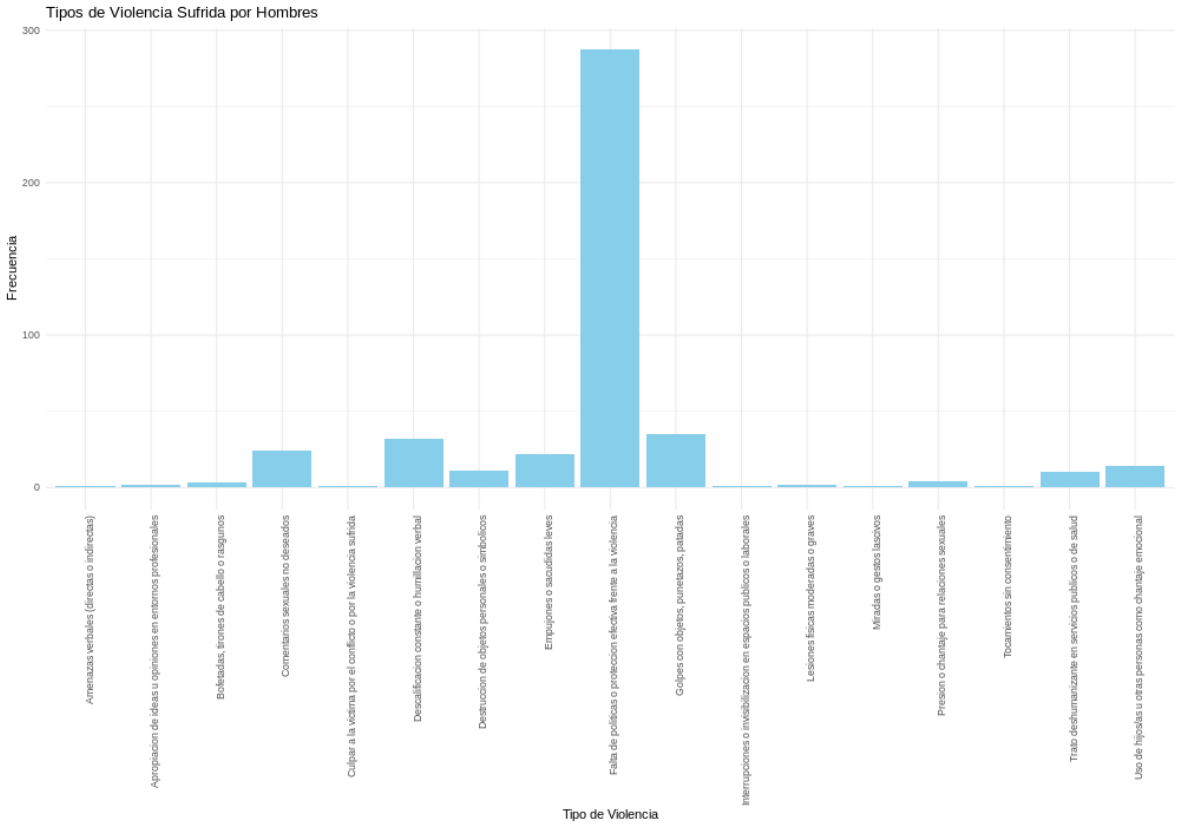
006 Violencia cometida hacia las hombres

Este código en R genera un gráfico de barras que representa los tipos de violencia sufridos exclusivamente por hombres, utilizando información contenida en el archivo "Data_modificado.csv". En primer lugar, se carga el archivo especificando la codificación UTF-8 y evitando la conversión automática de cadenas en factores. A continuación, se filtran los registros donde la columna Genero.Victima es igual a 0, que corresponde a víctimas de sexo masculino, y se guarda el resultado en datos_hombres.

Usando ggplot2, se construye un gráfico de barras donde el eje x muestra los distintos tipos de violencia (Nombre_Violencia) y el eje y la cantidad de casos correspondientes. Las barras se colorean en azul claro ("skyblue"), y se utiliza un tema minimalista para una visualización clara y estética. Para facilitar la lectura, los nombres de las categorías en el eje x se rotan 90 grados. Finalmente, el gráfico se exporta como una imagen PNG de 1000 por 700 píxeles bajo el nombre "006_violencia_hacia_hombres.png".

```
################################################################################
#


# Leer el archivo CSV modificado

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)
```

```
# Filtrar solo los casos donde la víctima es hombre (Genero.Victima == 0)
datos_hombres <- datos %>%
  filter(Genero.Victima == 0)

# Crear gráfico y guardar como imagen PNG
png("006_violencia_hacia_hombres.png", width = 1000, height = 700)

ggplot(datos_hombres, aes(x = Nombre_Violencia)) +
  geom_bar(fill = "skyblue") +
  labs(title = "Tipos de Violencia Sufrida por Hombres",
       x = "Tipo de Violencia",
       y = "Frecuencia") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

dev.off()
################################################################################
#
```



007 Comparacion de los actos violentos comentidos hacia hombres y mujeres

Este código en R genera un gráfico comparativo que muestra los tipos de violencia sufridos por hombres y mujeres, utilizando datos del archivo "Data_modificado.csv". Primero, se carga el archivo con codificación UTF-8 y sin transformar automáticamente las cadenas en factores. Luego, la columna Genero.Victima se convierte en un factor con dos niveles: "Hombre" (0) y "Mujer" (1), para facilitar la interpretación.

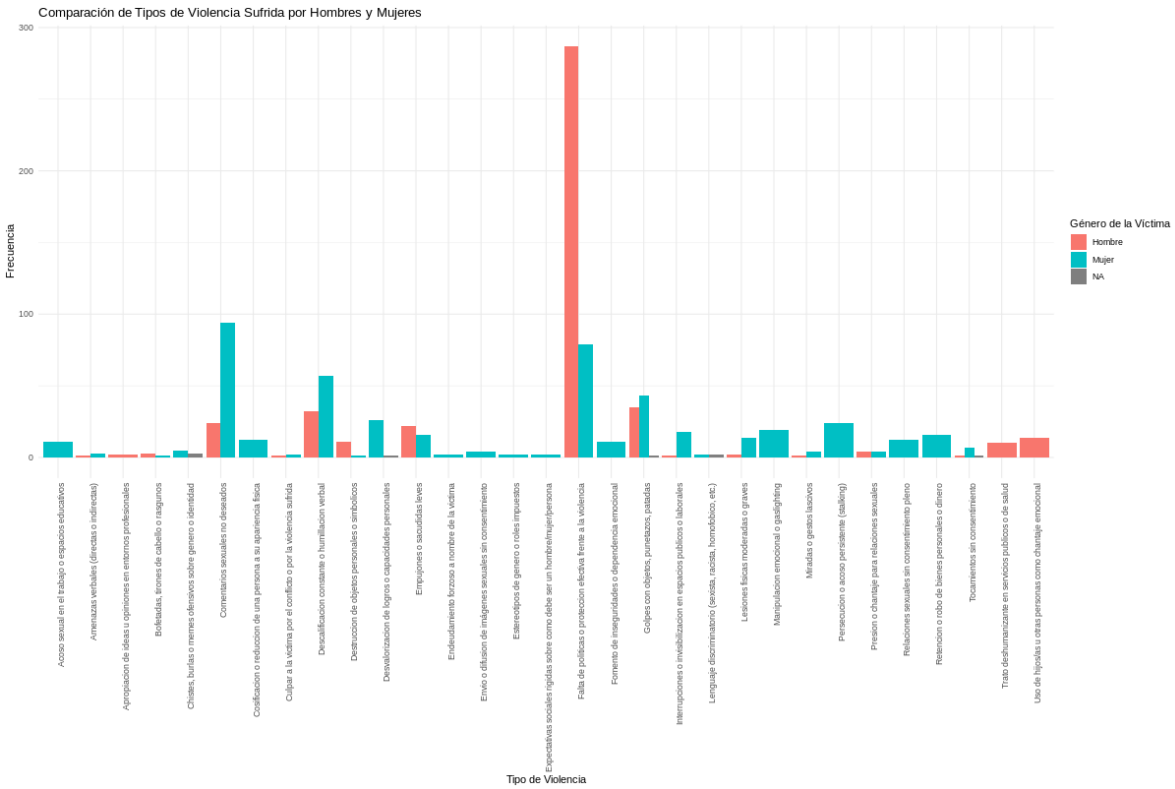
Usando ggplot2, se construye un gráfico de barras agrupadas donde el eje x representa los distintos tipos de violencia (Nombre_Violencia) y el eje y la frecuencia de casos. Las barras se agrupan por género utilizando el argumento position = "dodge", lo que permite una comparación visual directa entre hombres y mujeres para cada tipo de violencia. Cada grupo se colorea según el género de la víctima, y se añade una leyenda explicativa. El diseño se estiliza con un tema minimalista, y se rotan las etiquetas del eje x para mejorar la legibilidad. Finalmente, el gráfico se exporta como imagen PNG de 1200 por 800 píxeles bajo el nombre "007_violencia_hombres_mujeres.png".

##
#

Leer el archivo CSV modificado

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)

```
# Asegurar que la columna Genero.Victima esté en formato factor
datos$Genero.Victima <- factor(datos$Genero.Victima,
                               levels = c(0, 1),
                               labels = c("Hombre", "Mujer"))

# Crear gráfico y guardar como imagen PNG
png("007_violencia_hombres_mujeres.png", width = 1200, height = 800)

ggplot(datos, aes(x = Nombre_Violencia, fill = Genero.Victima)) +
  geom_bar(position = "dodge") +
  labs(title = "Comparación de Tipos de Violencia Sufrida por Hombres y Mujeres",
       x = "Tipo de Violencia",
       y = "Frecuencia",
       fill = "Género de la Víctima") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

dev.off()
################################################################################
#
```



008 Ingreso de los registros por meses y genero

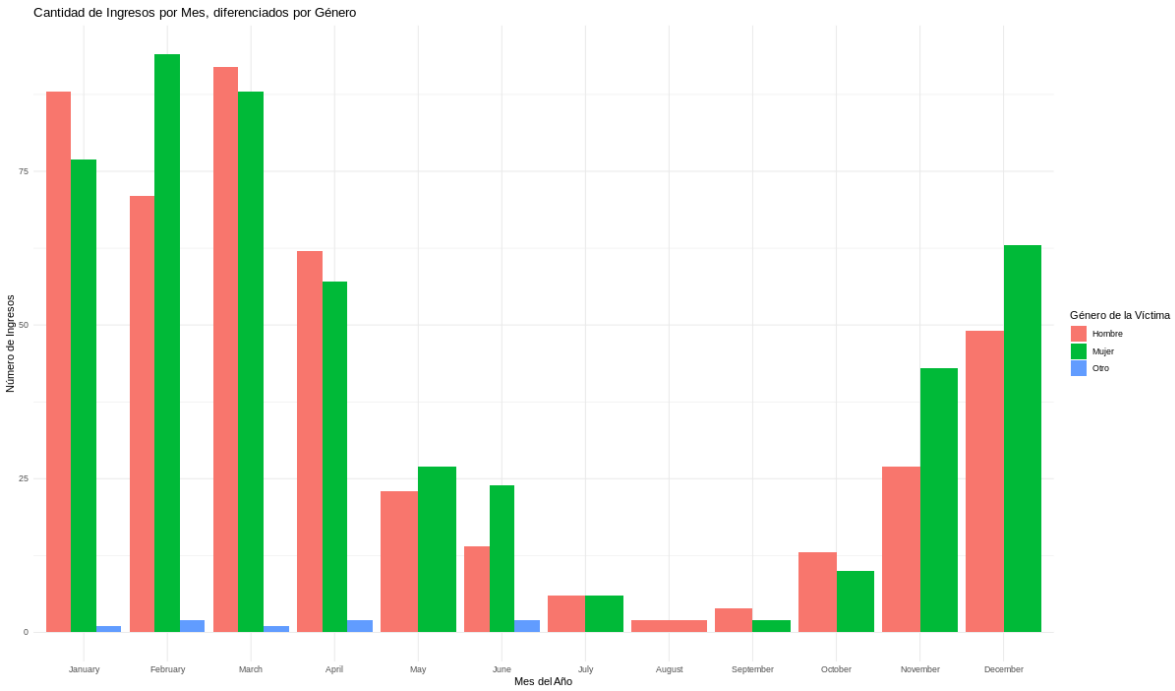
Este código en R genera un gráfico de barras que muestra la cantidad de ingresos registrados por mes, diferenciados por género de la víctima. Primero, se carga el archivo "Data_modificado.csv" con codificación UTF-8, evitando convertir automáticamente cadenas en factores. Luego, se convierte la columna Fecha, que incluye fecha y hora en formato "dd-mm-aaaa hh:mm", a un objeto de fecha y hora (POSIXct) utilizando la función dmy_hm() de lubridate.

Posteriormente, se extrae el nombre del mes de cada fecha y se guarda en la nueva columna Mes, definiendo los niveles como los nombres completos de los meses en orden cronológico. La variable Genero.Victima se transforma en un factor con tres niveles etiquetados como "Hombre", "Mujer" y "Otro".

Finalmente, se utiliza ggplot2 para crear un gráfico de barras agrupadas (position = "dodge"), donde el eje x representa los meses del año y el eje y el número de ingresos. Las barras se colorean según el género de la víctima, y se incluye una leyenda explicativa. El gráfico se estiliza con un tema minimalista y se guarda como imagen PNG de 1200 por 700 píxeles bajo el nombre "008_Ingreso_PorMesesYGenero.png".

```
################################################################################
#

library(lubridate)
```

```
# Leer el archivo CSV
datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)

# Convertir formato de fecha con hora: "09-09-2021 14:09"
datos$Fecha <- dmy_hm(datos$Fecha)  # convierte a objeto POSIXct con hora

# Extraer nombre del mes como texto ordenado
datos$Mes <- factor(month(datos$Fecha, label = TRUE, abbr = FALSE),
                    levels = month.name, ordered = TRUE)

# Convertir Genero.Victima a factor con etiquetas
datos$Genero.Victima <- factor(datos$Genero.Victima,
                               levels = c(0, 1, 2),
                               labels = c("Hombre", "Mujer", "Otro"))

# Crear gráfico y guardar imagen
png("008_Ingreso_PorMesesYGenero.png", width = 1200, height = 700)

ggplot(datos, aes(x = Mes, fill = Genero.Victima)) +
  geom_bar(position = "dodge") +
  labs(title = "Cantidad de Ingresos por Mes, diferenciados por Género",
       x = "Mes del Año",
       y = "Número de Ingresos",
       fill = "Género de la Víctima") +
  theme_minimal()

dev.off()
################################################################################
#
```



009 Ingreso de Registros Diferenciados por Mes y Genero

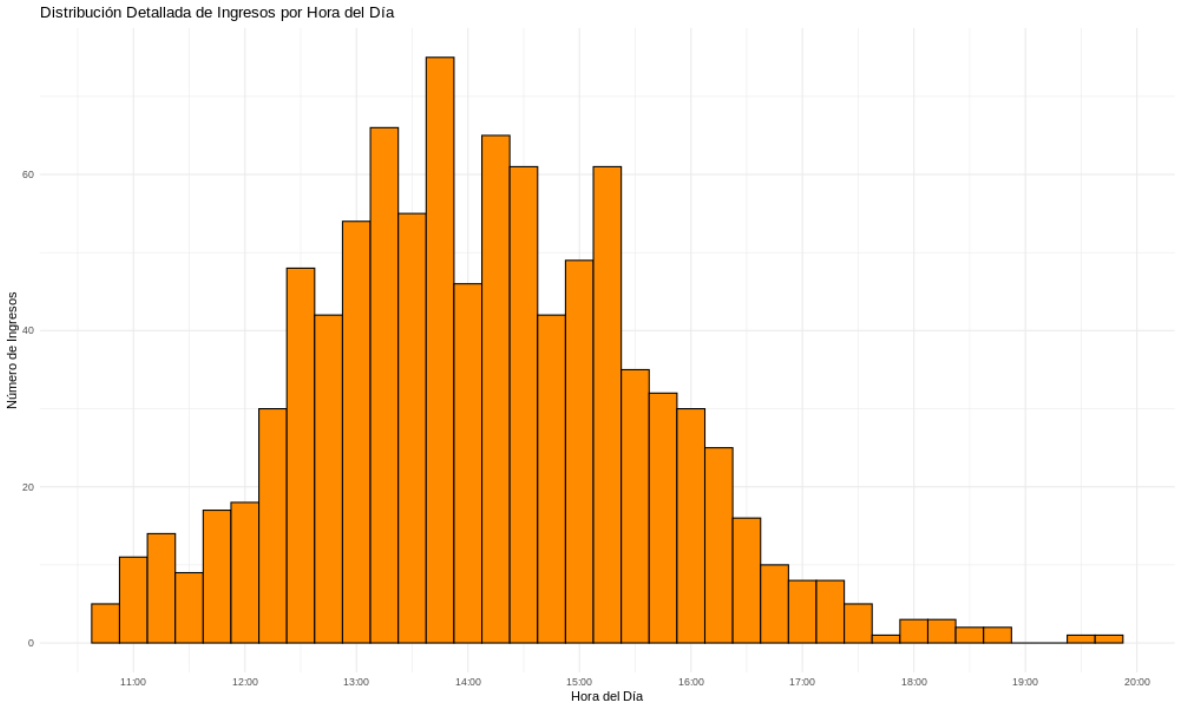
Este código en R genera un histograma que muestra la distribución detallada de ingresos por hora del día, utilizando información temporal contenida en el archivo "Data_modificado.csv". Primero, se carga el archivo con codificación UTF-8, sin convertir cadenas en factores automáticamente. Luego, la columna Fecha, que incluye fecha y hora en formato "dd-mm-aaaa hh:mm", se convierte a un objeto POSIXct mediante la función dmy_hm() de lubridate.

A partir de esta fecha, se calcula una nueva variable Hora_decimal, que representa la hora del ingreso en formato decimal (por ejemplo, 14.25 para las 14:15). Usando ggplot2, se construye un histograma donde el eje x representa las horas del día y el eje y la cantidad de ingresos en cada franja horaria. Las barras tienen un ancho de bin de 0.25, equivalente a intervalos de 15 minutos, y se colorean en naranja oscuro con borde negro.

El eje x se ajusta para mostrar las horas completas del día, de 0:00 a 23:00. El gráfico se estiliza con un tema minimalista y se guarda como imagen PNG de 1000 por 600 píxeles bajo el nombre "009_ingresos_Horario.png".

```
################################################################################
#


# Leer archivo CSV

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)


# Convertir Fecha y calcular hora decimal (ej: 14.25 = 14:15)
```

```
datos$Fecha <- dmy_hm(datos$Fecha)
datos$Hora_decimal <- hour(datos$Fecha) + minute(datos$Fecha) / 60

# Crear gráfico y guardar imagen PNG
png("009_ingresos_Horario.png", width = 1000, height = 600)

ggplot(datos, aes(x = Hora_decimal)) +
  geom_histogram(binwidth = 0.25, fill = "darkorange", color = "black") +  # cada 15 minutos
  scale_x_continuous(breaks = seq(0, 23.75, by = 1),
                     labels = paste0(seq(0, 23), ":00")) +
  labs(title = "Distribución Detallada de Ingresos por Hora del Día",
       x = "Hora del Día",
       y = "Número de Ingresos") +
  theme_minimal()

dev.off()
################################################################################
#
```



010 Funcion Graficar Violencia de genero sobre genero

Este código en R define una función llamada graficar_violencia_por_genero que permite generar gráficos de barras para visualizar los tipos de violencia según combinaciones específicas de género de la víctima y del agresor, usando datos de un archivo CSV. La función acepta cuatro argumentos: el archivo CSV, el género de la víctima (genero_victima), el género del agresor (genero_agresor), y el nombre del archivo de salida para guardar el gráfico.

Primero, se cargan los datos y se convierten las columnas Genero.Victima y Genero.Agresor a valores numéricos. Luego, se filtran los registros que coincidan con los géneros indicados. Si no hay datos que cumplan con esos criterios, la función devuelve un mensaje y termina sin crear el gráfico.

En caso contrario, se genera un gráfico de barras con ggplot2 que muestra la frecuencia de los distintos tipos de violencia (Nombre_Violencia) en la combinación de géneros especificada, y se guarda como imagen PNG. El gráfico se estiliza con colores y un diseño minimalista.

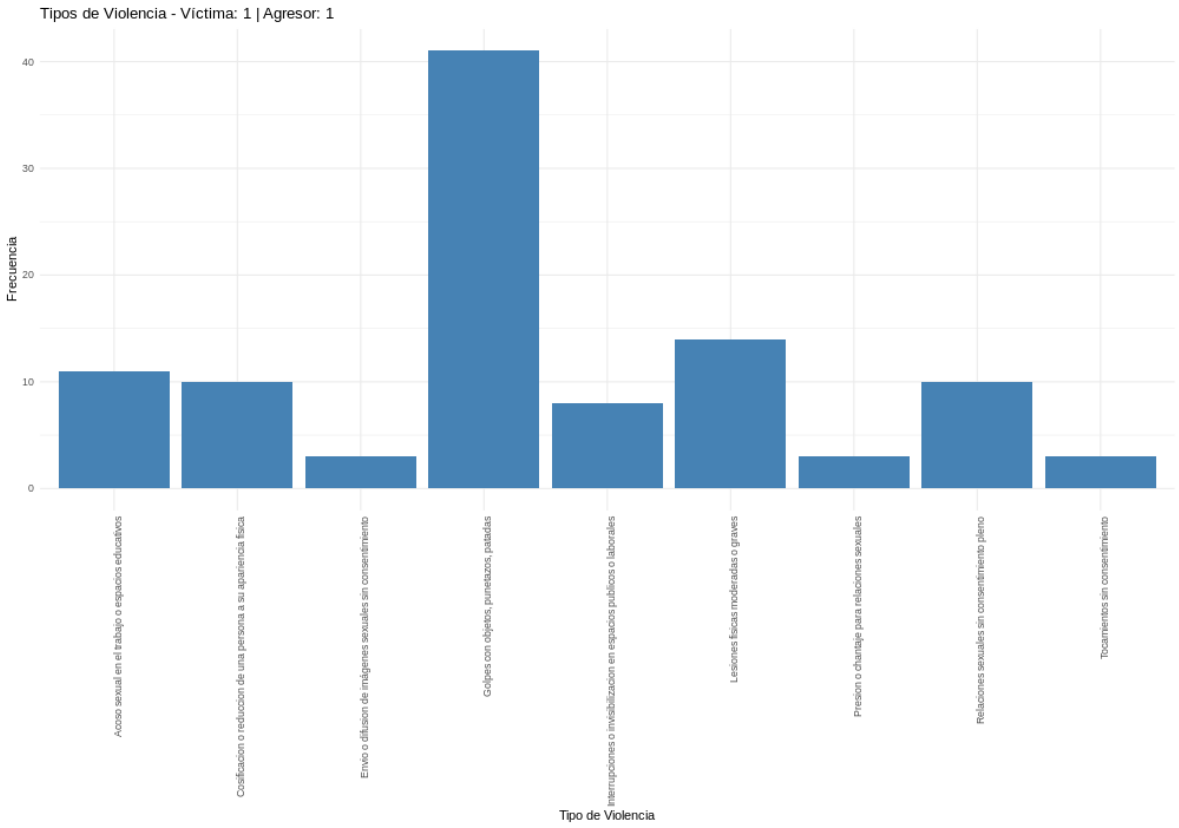
Finalmente, se ejecuta la función cuatro veces para visualizar los casos de violencia mujer a mujer, mujer a hombre, hombre a mujer y hombre a hombre, guardando cada gráfico con un nombre representativo.

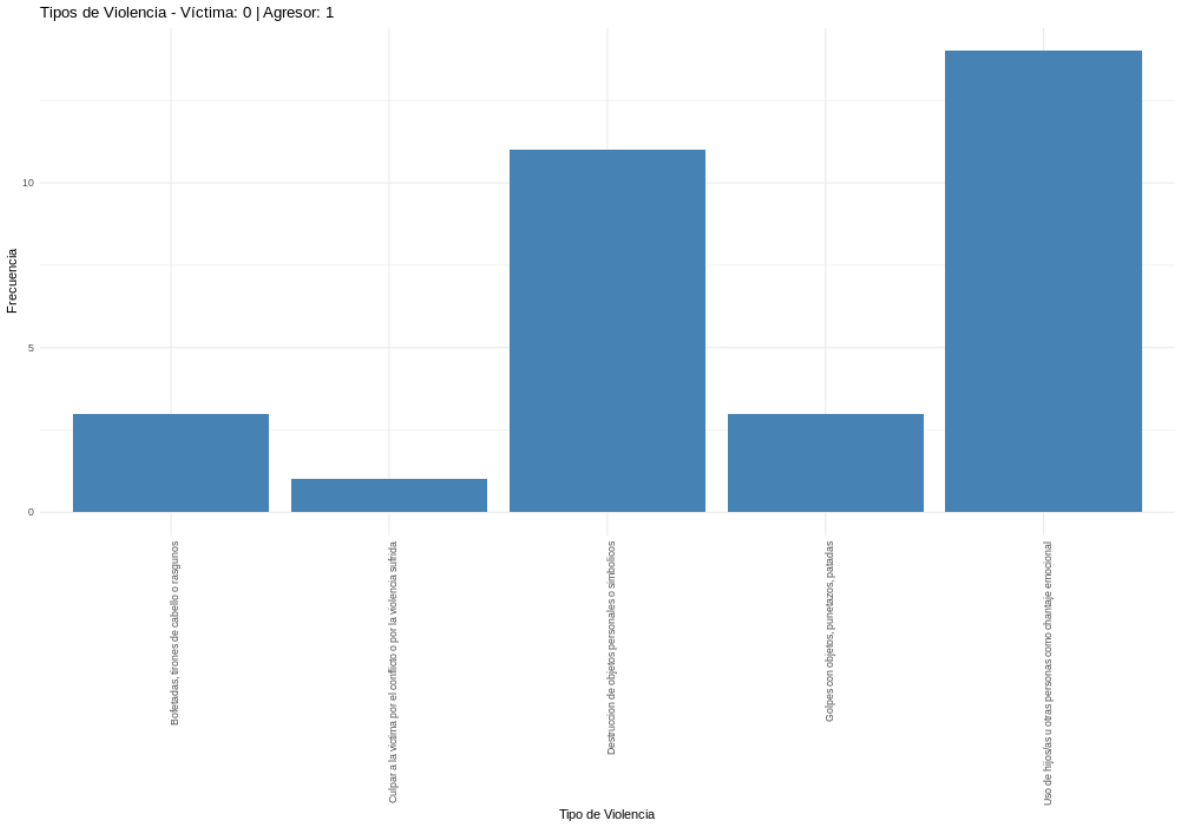
```
################################################################################
#

################################################################################
#

# Funcion Graficar Violencia de genero sobre genero
```

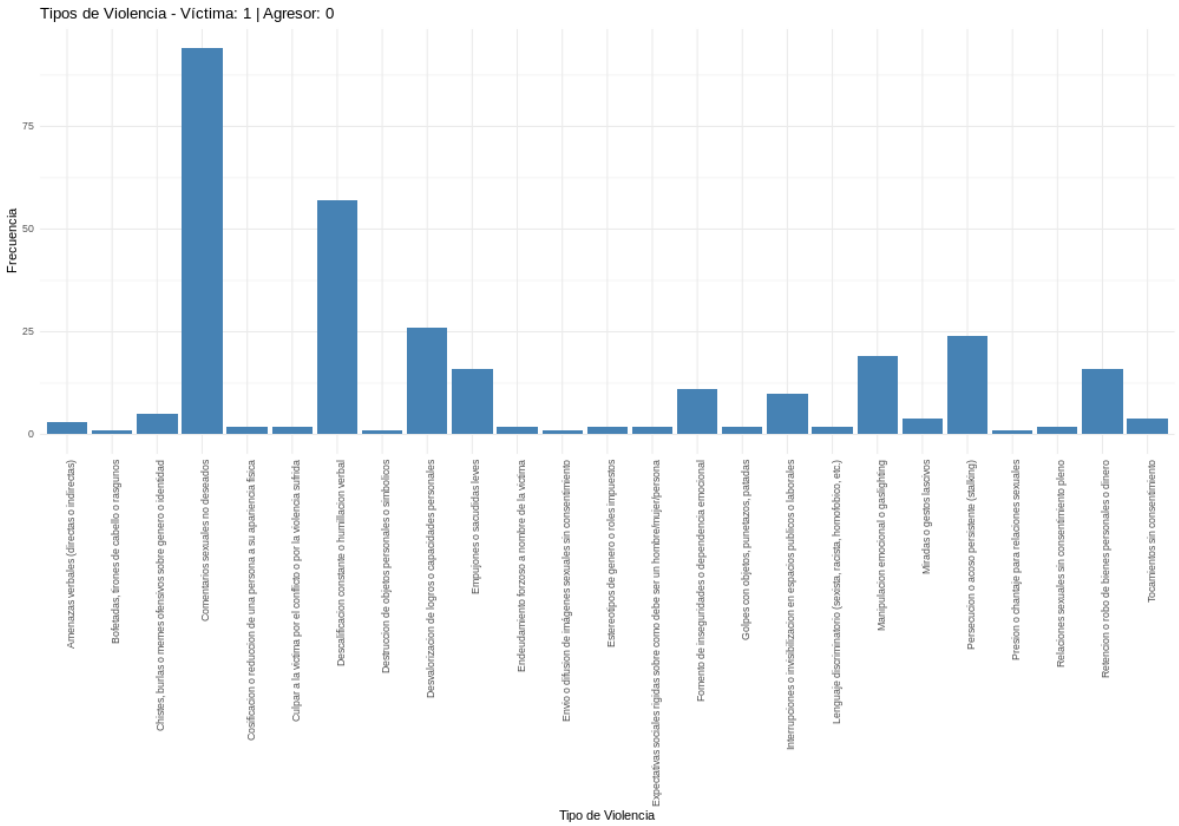
```
graficar_violencia_por_genero <- function(archivo_csv, genero_victima = 1, genero_agresor = 1,
salida = "grafico.png") {
 library(ggplot2)
 library(readr)
 library(dplyr)

 # Leer archivo
 datos <- read.csv2(archivo_csv, encoding = "UTF-8", stringsAsFactors = FALSE)

 # Convertir a numérico
 datos$Genero.Victima <- as.numeric(as.character(datos$Genero.Victima))
 datos$Genero.Agresor <- as.numeric(as.character(datos$Genero.Agresor))

 # Filtrar
 datos_filtrados <- datos %>%
  filter(Genero.Victima == genero_victima,
       Genero.Agresor == genero_agresor)

 # Si no hay datos, no crear el gráfico
 if (nrow(datos_filtrados) == 0) {
  message("No se encontraron registros para esos géneros.")
  return(NULL)
 }

 # Intentar crear gráfico
 tryCatch({
  png(salida, width = 1000, height = 700)
  print(
   ggplot(datos_filtrados, aes(x = Nombre_Violencia)) +
```

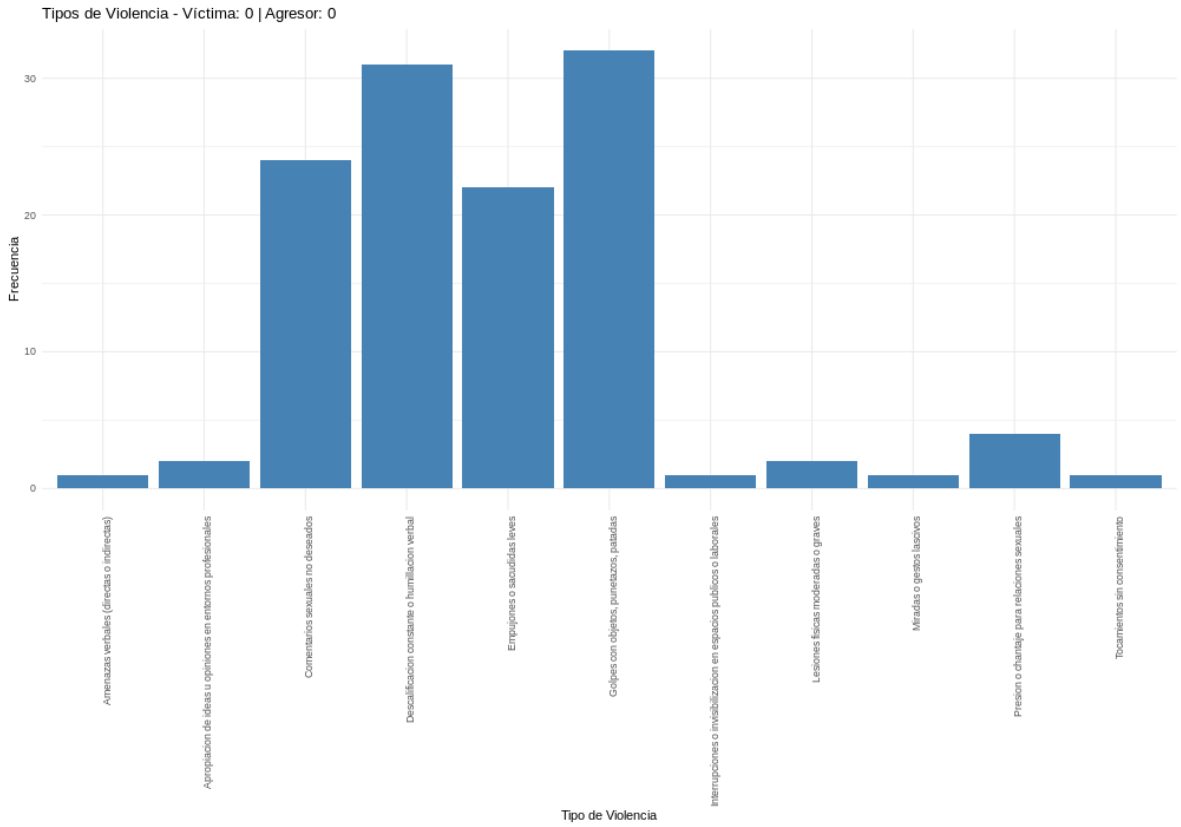
```
    geom_bar(fill = "steelblue") +
    labs(
     title = paste("Tipos de Violencia - Víctima:", genero_victima, "| Agresor:", genero_agresor),
     x = "Tipo de Violencia",
     y = "Frecuencia"
    ) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
  )
  dev.off()
  message(paste(" Gráfico guardado en:", salida))
 }, error = function(e) {
  message(" Error al generar el gráfico: ", e$message)
  dev.off()  # cierra dispositivo en caso de error
 })
}
################################################################################
#

graficar_violencia_por_genero("Data_modificado.csv", 1, 1, "010_mujeres_sobre_mujeres.png")
graficar_violencia_por_genero("Data_modificado.csv", 0, 1, "010_mujeres_sobre_hombres.png")
graficar_violencia_por_genero("Data_modificado.csv", 1, 0, "010_hombres_sobre_mujeres.png")
graficar_violencia_por_genero("Data_modificado.csv", 0, 0, "010_hombres_sobre_hombres.png")

################################################################################
##
```

##
##


Tipos de Violencia - Víctima: 1 | Agresor: 1
Frecuencia
0
10
20
30
40
Acoso sexual en el trabajo o espacios educativos
Cosificacion o reduccion de una persona a su apariencia fisica
Envio o difusion de imágenes sexuales sin consentimiento
Golpes con objetos, puñetazos, patadas
Interrupciones o invisibilizacion en espacios publicos o laborales
Lesiones fisicas moderadas o graves
Presion o chantaje para relaciones sexuales
Relaciones sexuales sin consentimiento pleno
Tocamientos sin consentimiento
Tipo de Violencia


Tipos de Violencia - Víctima: 0 | Agresor: 1
Frecuencia
0
5
10
Bofetadas, tirones de cabello o rasgunos
Culpar a la víctima por el conflicto o por la violencia sufrida
Destruccion de objetos personales o simbolicos
Golpes con objetos, punetazos, patadas
Uso de hijos/as u otras personas como chantaje emocional
Tipo de Violencia


Tipos de Violencia - Víctima: 1 | Agresor: 0
Frecuencia
0
25
50
75
Amenazas verbales (directas o indirectas)
Bofetadas, tirones de cabello o rasgunos
Chistes, burlas o memes ofensivos sobre genero o identidad
Comentarios sexuales no deseados
Cosificacion o reduccion de una persona a su apariencia fisica
Culpar a la víctima por el conflicto o por la violencia sufrida
Descalificacion constante o humillacion verbal
Destruccion de objetos personales o simbolicos
Desvalorizacion de logros o capacidades personales
Empujones o sacudidas leves
Endeudamiento forzoso a nombre de la víctima
Envio o difusion de imágenes sexuales sin consentimiento
Estereotipos de genero o roles impuestos
Expectativas sociales rigidas sobre como debe ser un hombre/mujer/persona
Fomento de inseguridades o dependencia emocional
Golpes con objetos, punetazos, patadas
Interrupciones o invisibilizacion en espacios publicos o laborales
Lenguaje discriminatorio (sexista, racista, homofobico, etc.)
Manipulacion emocional o gaslighting
Miradas o gestos lasivos
Persecucion o acoso persistente (stalking)
Presion o chantaje para relaciones sexuales
Relaciones sexuales sin consentimiento pleno
Retencion o robo de bienes personales o dinero
Tocamientos sin consentimiento
Tipo de Violencia



011 Función para Graficar tipo de violencia por edad, comuna y género de la víctima

Este código en R define una función llamada graficar_violencia_por_edad_comuna_y_genero, que genera un gráfico de barras para visualizar los tipos de violencia registrados en una comuna específica, para una edad y género determinados. Primero, se cargan los datos desde un archivo CSV y se convierten las columnas relevantes (Genero.Victima, Edad, Nombre_Comuna) a formatos adecuados para su análisis: numéricos y texto en minúsculas.

Luego, se filtran los registros que coincidan exactamente con la edad, la comuna y el género de la víctima especificados como argumentos. Si no existen registros que cumplan estas condiciones, la función informa al usuario y no genera ningún gráfico.

Si existen datos, se crea un gráfico de barras con ggplot2, donde el eje x representa los distintos tipos de violencia (Nombre_Violencia) y el eje y la frecuencia de cada tipo. Las barras se colorean en rojo tomate y se aplica un tema visual limpio. Finalmente, el gráfico se guarda como una imagen PNG.

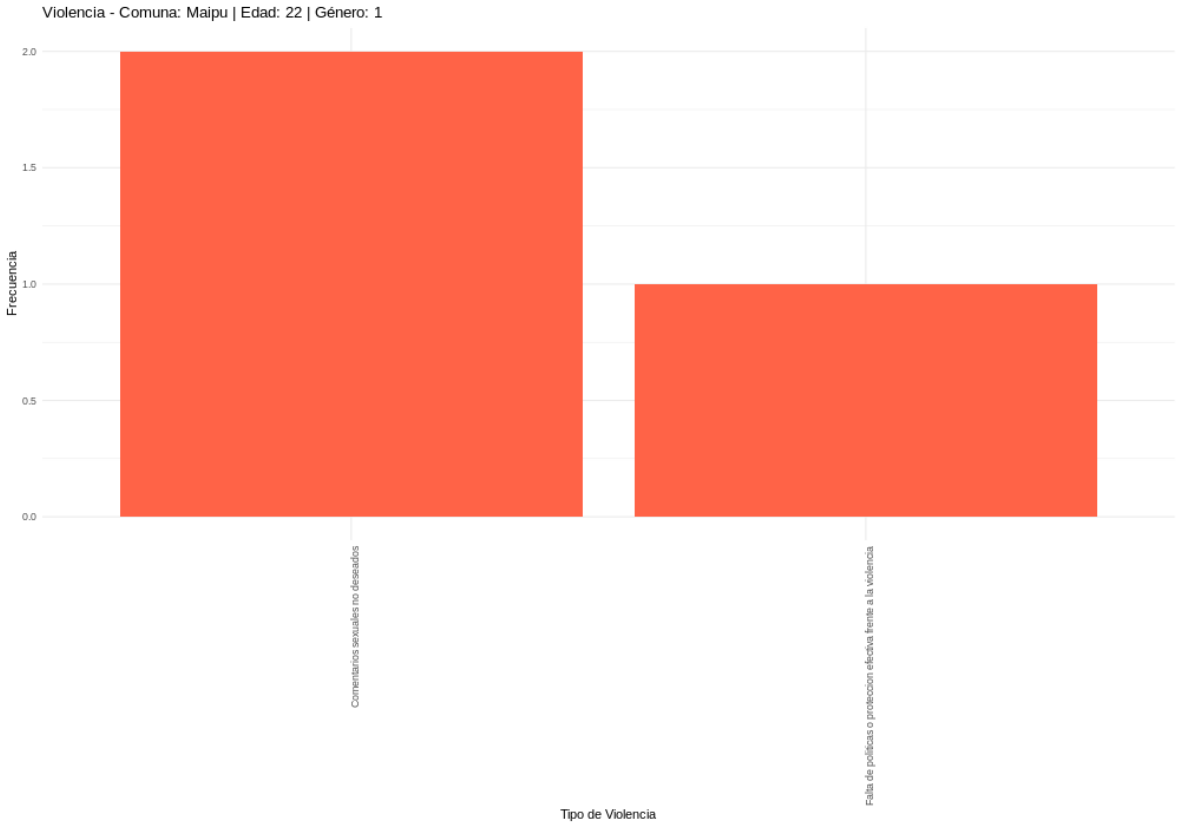
Como ejemplo de uso, se ejecuta la función para una mujer de 22 años en la comuna de Maipú, generando un archivo llamado "011_violencia_maipu_mujer_22.png".

```
################################################################################
#

# Función: Graficar tipo de violencia por edad, comuna y género de la víctima
```

```
graficar_violencia_por_edad_comuna_y_genero <- function(archivo_csv, edad_objetivo, nombre_comuna, genero_victima, salida = "violencia_edad_comuna_genero.png") {
 library(ggplot2)
 library(readr)
 library(dplyr)

 # Leer el archivo
 datos <- read.csv2(archivo_csv, encoding = "UTF-8", stringsAsFactors = FALSE)

 # Convertir columnas a formatos adecuados
 datos$Genero.Victima <- as.numeric(as.character(datos$Genero.Victima))
 datos$Edad <- as.numeric(as.character(datos$Edad))
 datos$Nombre_Comuna <- tolower(datos$Nombre_Comuna)
 nombre_comuna <- tolower(nombre_comuna)

 # Filtrar por edad, comuna y género
 datos_filtrados <- datos %>%
  filter(Edad == edad_objetivo,
      Nombre_Comuna == nombre_comuna,
      Genero.Victima == genero_victima)

 # Validar si hay datos
 if (nrow(datos_filtrados) == 0) {
  message(" No hay registros para la comuna '", nombre_comuna, "', edad ", edad_objetivo, " y género ", genero_victima, ".")
  return(NULL)
 }

 # Crear gráfico
 tryCatch({
```

```
  png(salida, width = 1000, height = 700)
  print(
   ggplot(datos_filtrados, aes(x = Nombre_Violencia)) +
    geom_bar(fill = "tomato") +
    labs(
     title = paste("Violencia - Comuna:", tools::toTitleCase(nombre_comuna),
                   "| Edad:", edad_objetivo,
                   "| Género:", genero_victima),
     x = "Tipo de Violencia",
     y = "Frecuencia"
    ) +
    theme_minimal() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
  )
  dev.off()
  message(paste(" Gráfico guardado en:", salida))
 }, error = function(e) {
  message(" Error al generar el gráfico: ", e$message)
  dev.off()
 })
}
################################################################################
#


# Ejemplo de uso: hombre (0), edad 30, comuna "maipu"
genero <- 1
genero_texto <- ifelse(genero == 1, "mujer", "hombre")
comuna <- "maipu"
edad <- 22
```

```
graficar_violencia_por_edad_comuna_y_genero(
  archivo_csv = "Data_modificado.csv",
  edad_objetivo = edad,
  nombre_comuna = comuna,
  genero_victima = genero,
  salida = paste0("011_violencia_", comuna, "_", genero_texto, "_", edad, ".png")
)
```



012 Función para graficar en 3D: Edad, Comuna y Género de la Víctima

Este código en R define una función llamada graficar_violencia_3d, que genera un gráfico tridimensional para visualizar la relación entre la edad de la víctima, la comuna (codificada numéricamente) y su género, a partir de los datos contenidos en un archivo CSV. Se utilizan las librerías scatterplot3d, readr y dplyr.

Primero, se cargan los datos desde el archivo indicado. Luego, se verifica que las columnas esenciales (Edad, Genero.Victima y Nombre_Comuna) estén presentes. A continuación, se preprocesan los datos: la comuna se codifica numéricamente usando factor(), y se filtran los registros incompletos o con valores faltantes.

Si se especifica un valor para genero_victima, los datos se filtran para incluir solo registros correspondientes a ese género. Si no hay datos tras el filtrado, se muestra un mensaje y no se genera el gráfico.

Cuando hay datos válidos, se crea un gráfico 3D con scatterplot3d, donde:

el eje X representa la edad,

el eje Y representa el código de la comuna,

el eje Z representa el género de la víctima.

Los puntos se colorean en azul y el gráfico se guarda como imagen PNG. La función se ejecuta tres veces: para todas las víctimas, solo mujeres y solo hombres.

```
##############################################################################
#
# Función para graficar en 3D: Edad, Comuna y Género de la Víctima
graficar_violencia_3d <- function(archivo_csv, genero_victima = NULL, salida = "grafico_3d.png") {
 # Cargar librerías necesarias
 #if (!requireNamespace("scatterplot3d", quietly = TRUE)) install.packages("scatterplot3d")
 #library(scatterplot3d)
 #library(readr)
 #library(dplyr)

 # Leer archivo CSV
 datos <- read.csv2(archivo_csv, encoding = "UTF-8", stringsAsFactors = FALSE)

 # Validar columnas necesarias
 if (!all(c("Edad", "Genero.Victima", "Nombre_Comuna") %in% names(datos))) {
  stop("El archivo debe contener las columnas: Edad, Genero.Victima y Nombre_Comuna.")
 }

 # Preprocesar y codificar comuna
 datos <- datos %>%
  mutate(
```

```
  Edad = as.numeric(Edad),
  Genero.Victima = as.numeric(Genero.Victima),
  Comuna_Cod = as.numeric(factor(Nombre_Comuna))
 ) %>%
 filter(!is.na(Edad), !is.na(Genero.Victima), !is.na(Comuna_Cod))

# Aplicar filtro de género si se especifica
if (!is.null(genero_victima)) {
 datos <- datos %>% filter(Genero.Victima == genero_victima)
}

# Verificar que hay datos
if (nrow(datos) == 0) {
 message(" No hay datos para el criterio especificado.")
 return(NULL)
}

# Generar gráfico y guardar como imagen PNG
tryCatch({
 png(salida, width = 1000, height = 800)
 scatterplot3d(
  x = datos$Edad,
  y = datos$Comuna_Cod,
  z = datos$Genero.Victima,
  pch = 16,
  color = "steelblue",
  xlab = "Edad",
  ylab = "Comuna (codificada)",
  zlab = "Género de la Víctima",
```

```
    main = "Gráfico 3D: Edad, Comuna, Género de la Víctima"
  )
  dev.off()
  message(paste(" Gráfico guardado como:", salida))
 }, error = function(e) {
  dev.off()
  message(" Error al generar el gráfico:", e$message)
 })
}
################################################################################
#

# Todas las víctimas
graficar_violencia_3d("Data_modificado.csv", salida = "012_violencia_3d_todas.png")

# Solo mujeres (1)
graficar_violencia_3d("Data_modificado.csv", genero_victima = 1, salida =
"012_violencia_3d_mujeres.png")

# Solo hombres (0)
graficar_violencia_3d("Data_modificado.csv", genero_victima = 0, salida =
"012_violencia_3d_hombres.png")
```

Gráfico 3D: Edad, Comuna, Género de la Víctima

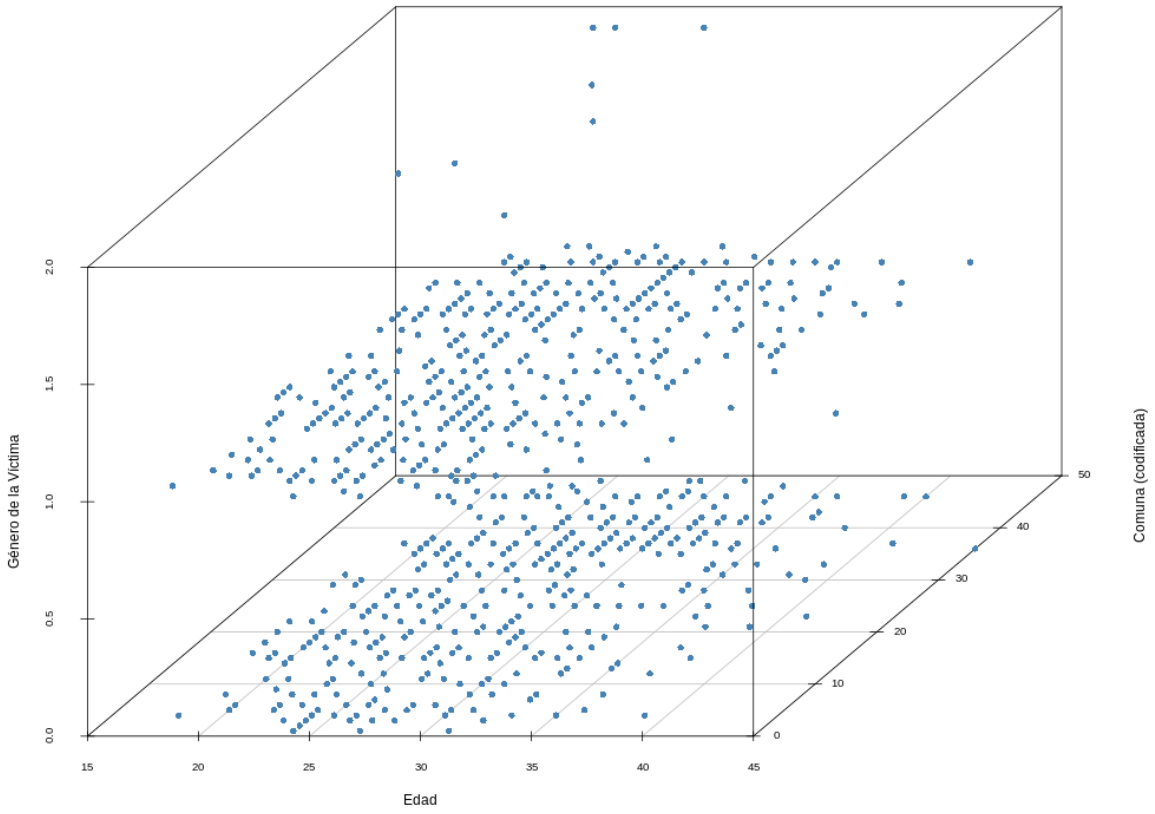
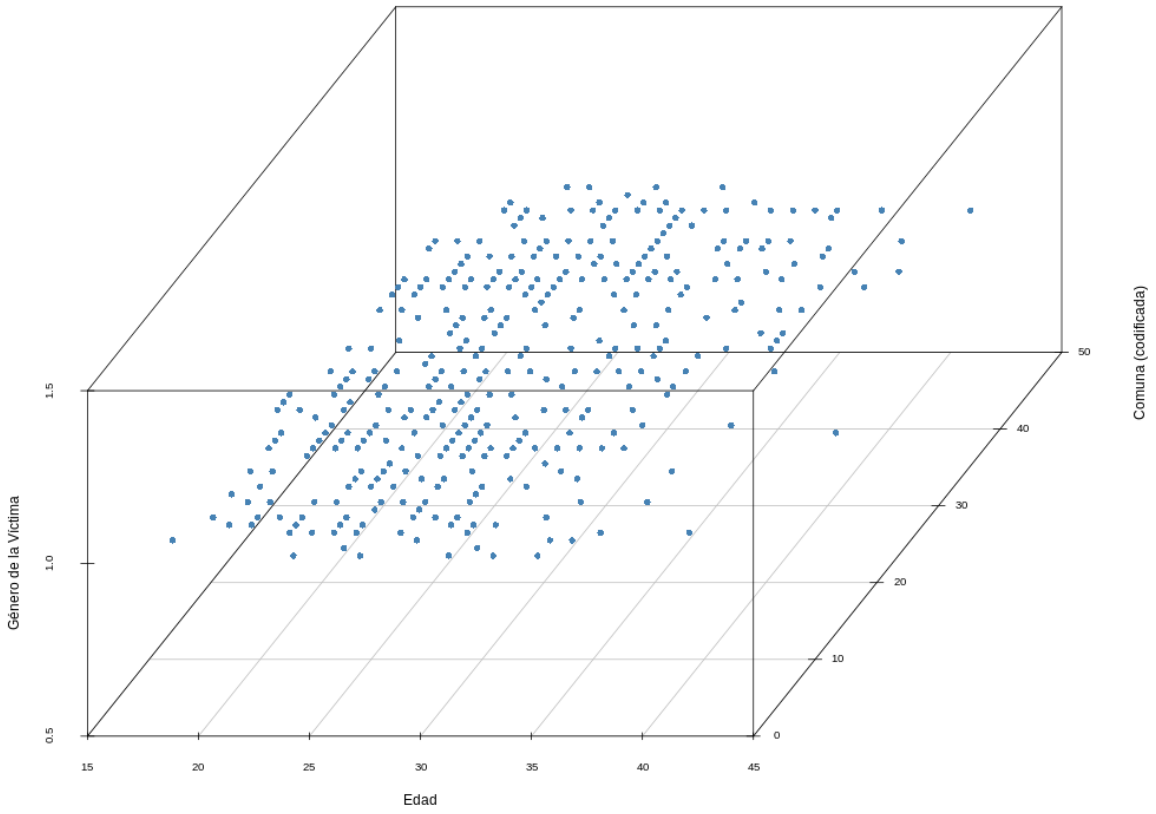
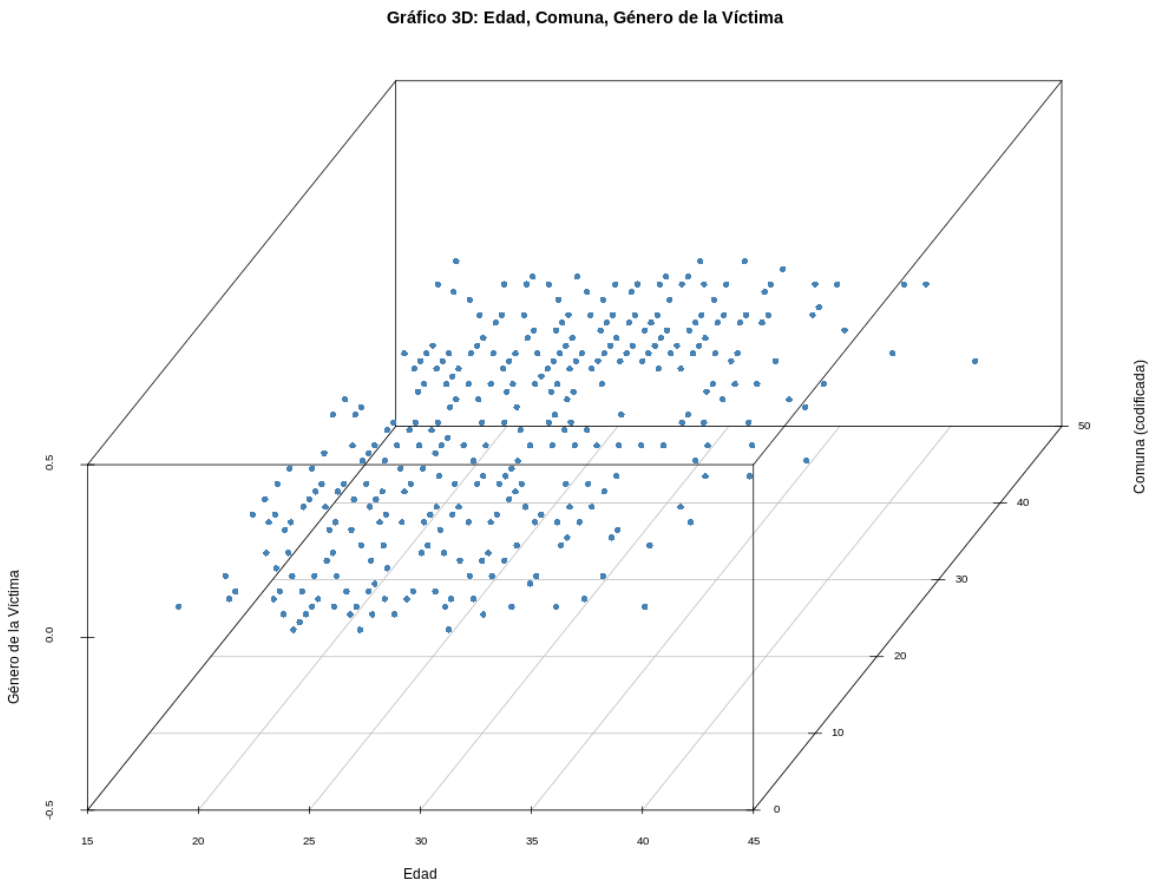


Gráfico 3D: Edad, Comuna, Género de la Víctima





El siguiente generar un grafico mediante una funcion el cual analizar visualmente datos relacionados con casos de violencia, cargados desde un archivo CSV real. Utiliza las librerías ggplot2, dplyr y readr para procesar y visualizar los datos de forma estructurada. En una primera etapa, el script importa los datos desde un archivo denominado Data_modificado.csv, convirtiendo la columna Fecha al formato fecha-hora POSIXct. Luego, realiza una limpieza de los datos, filtrando aquellos registros que tienen valores faltantes en columnas claves como el género de la víctima, la edad y la comuna. Posteriormente, se definen seis funciones para generar distintos tipos de gráficos: Barras por género de la víctima, que muestra la frecuencia de casos según el género. Barras agrupadas víctima-agresor, que compara el género de víctimas y agresores. Casos por comuna, para identificar la distribución geográfica. Dispersión edad-comuna, que cruza edad, comuna y género de víctima. Gráfico de pastel, que representa la proporción por género de víctima. Histograma temporal, que muestra cómo varían los casos en el tiempo. Cada gráfico se guarda automáticamente en formato PNG con alta resolución, facilitando su uso en informes o presentaciones. En conjunto, este código permite visualizar de forma clara y eficiente los patrones y distribuciones en los datos de violencia registrados.

```
# === 1. Cargar datos reales ===

datos <- read.csv2("Data_modificado.csv", encoding = "UTF-8", stringsAsFactors = FALSE)
```

```
# Convertir columna Fecha a formato de fecha
datos$Fecha <- as.POSIXct(datos$Fecha, format = "%d-%m-%Y %H:%M", tz = "UTC")

# Eliminar registros con valores faltantes clave
datos <- datos %>% filter(!is.na(Nombre_Genero_Victima_Texto), !is.na(Edad), !is.na(Nombre_Comuna))

# === 2. Funciones de gráficos ===

grafico_genero_victima <- function(df) {
  g <- ggplot(df, aes(x = Nombre_Genero_Victima_Texto)) +
    geom_bar(fill = "steelblue") +
    labs(title = "Casos por género de la víctima", x = "Género", y = "Cantidad")
  ggsave("012_grafico_genero_victima.png", plot = g, width = 6, height = 4, dpi = 300)
}

grafico_genero_cruzado <- function(df) {
  g <- ggplot(df, aes(x = Nombre_Genero_Victima_Texto, fill = Nombre_Genero_Agresor_Texto)) +
    geom_bar(position = "dodge") +
    labs(title = "Víctima vs Agresor por género", x = "Género víctima", fill = "Género agresor")
  ggsave("012_grafico_genero_cruzado.png", plot = g, width = 6, height = 4, dpi = 300)
}

grafico_comuna <- function(df) {
  g <- ggplot(df, aes(x = Nombre_Comuna)) +
    geom_bar(fill = "darkorange") +
    labs(title = "Casos por comuna", x = "Comuna", y = "Cantidad") +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
  ggsave("012_grafico_comuna.png", plot = g, width = 8, height = 5, dpi = 300)
```

```
}

grafico_edad_comuna <- function(df) {
  g <- ggplot(df, aes(x = Edad, y = Nombre_Comuna, color = Nombre_Genero_Victima_Texto)) +
    geom_point(size = 2) +
    labs(title = "Edad de víctimas por comuna", x = "Edad", y = "Comuna")
  ggsave("012_grafico_edad_comuna.png", plot = g, width = 6, height = 5, dpi = 300)
}

grafico_pastel_tipo <- function(df) {
  df_summary <- df %>%
    group_by(Nombre_Genero_Victima_Texto) %>%
    summarise(Frecuencia = n())

  g <- ggplot(df_summary, aes(x = "", y = Frecuencia, fill = Nombre_Genero_Victima_Texto)) +
    geom_col() +
    coord_polar("y", start = 0) +
    labs(title = "Proporción por género de víctima", fill = "Género")
  ggsave("012_grafico_pastel_tipo.png", plot = g, width = 5, height = 5, dpi = 300)
}

grafico_tiempo <- function(df) {
  g <- ggplot(df, aes(x = Fecha)) +
    geom_histogram(binwidth = 86400, fill = "purple") +
    labs(title = "Distribución temporal de casos", x = "Fecha", y = "Cantidad")
  ggsave("012_grafico_tiempo.png", plot = g, width = 7, height = 4, dpi = 300)
}

# === 3. Ejecutar y guardar gráficos ===
```

grafico_genero_victima(datos)

grafico_genero_cruzado(datos)

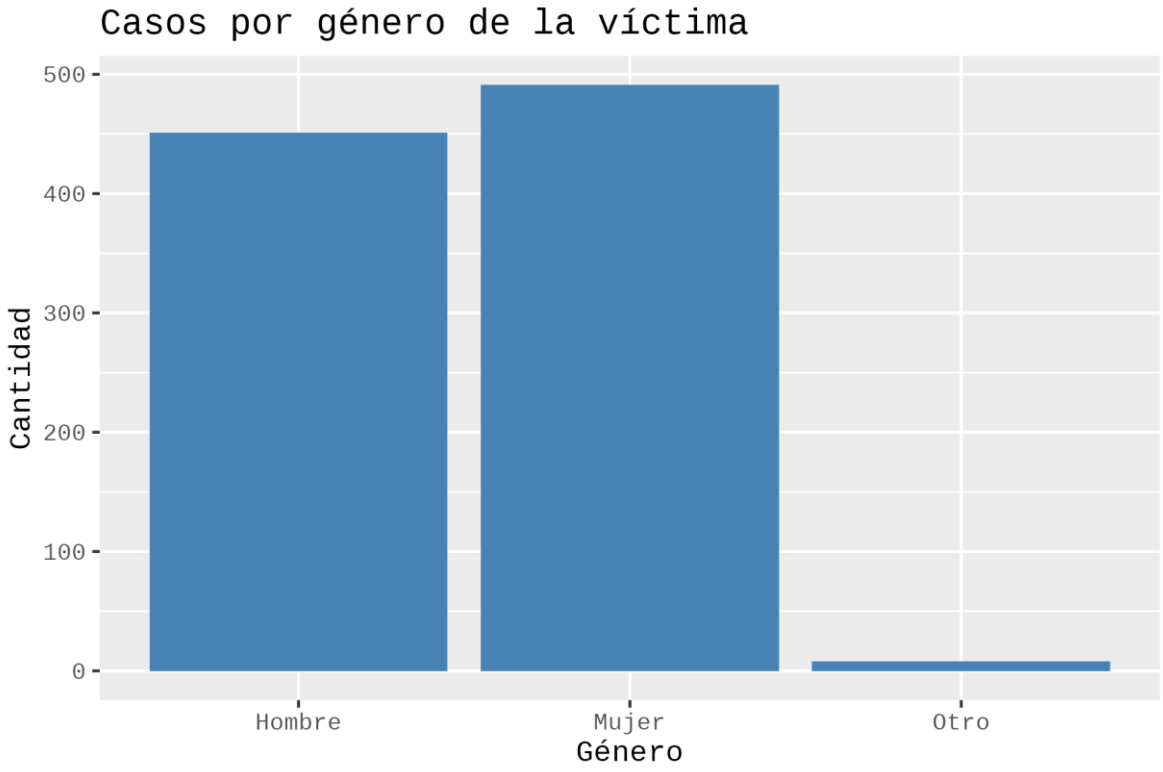
grafico_comuna(datos)

grafico_edad_comuna(datos)

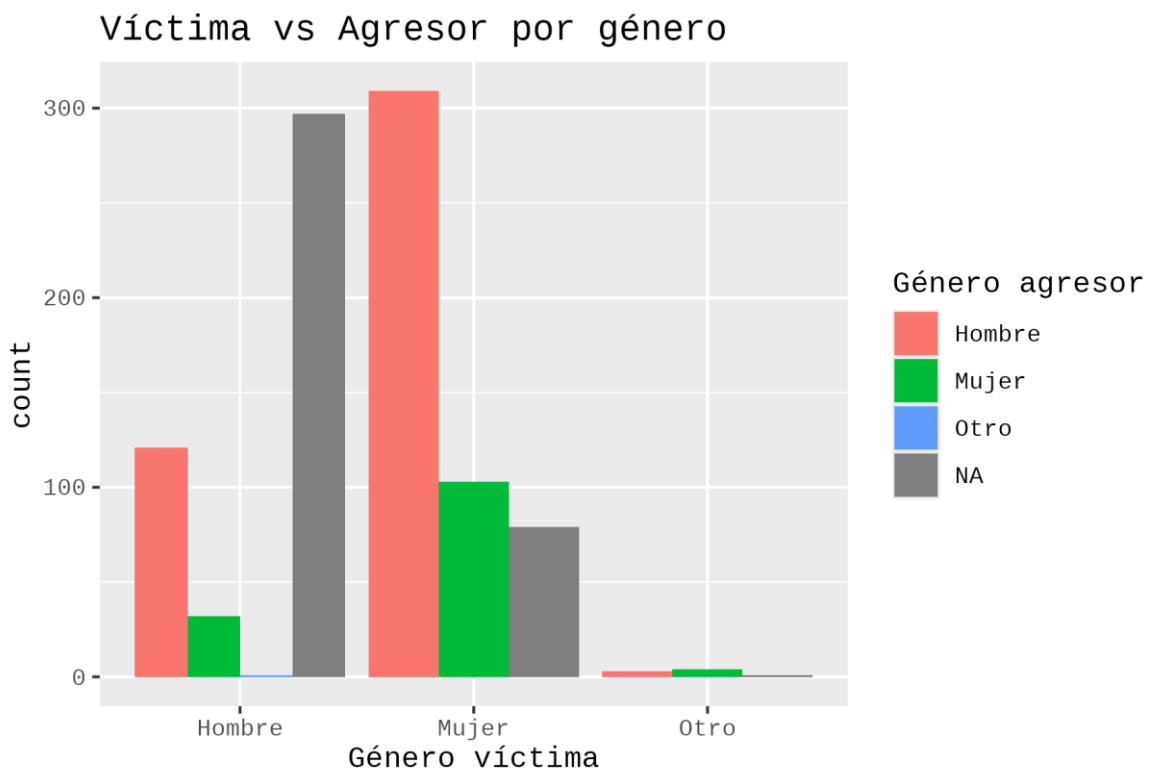
grafico_pastel_tipo(datos)

grafico_tiempo(datos)

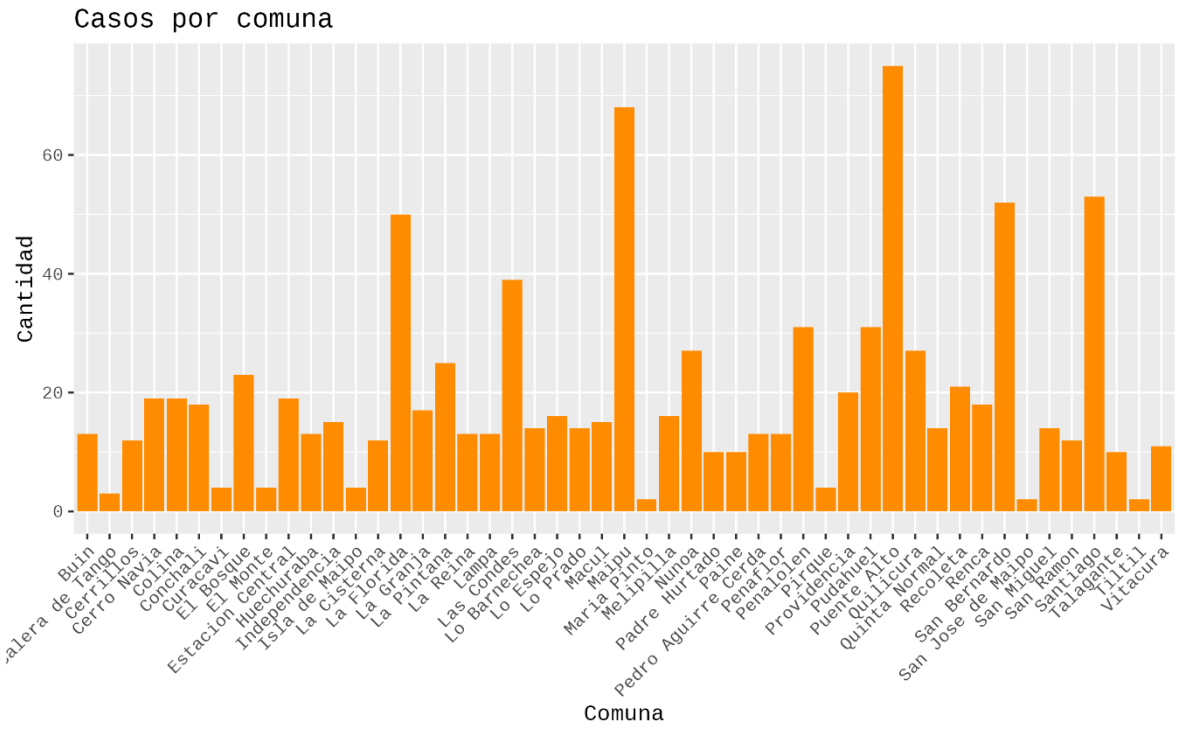
Esta función en R crea un gráfico de barras que muestra la cantidad de casos según el género de la víctima. Utiliza ggplot2 para visualizar los datos con barras en color azul y etiquetas descriptivas. El gráfico se guarda automáticamente como una imagen PNG llamada 012_grafico_genero_victima.png con alta resolución.



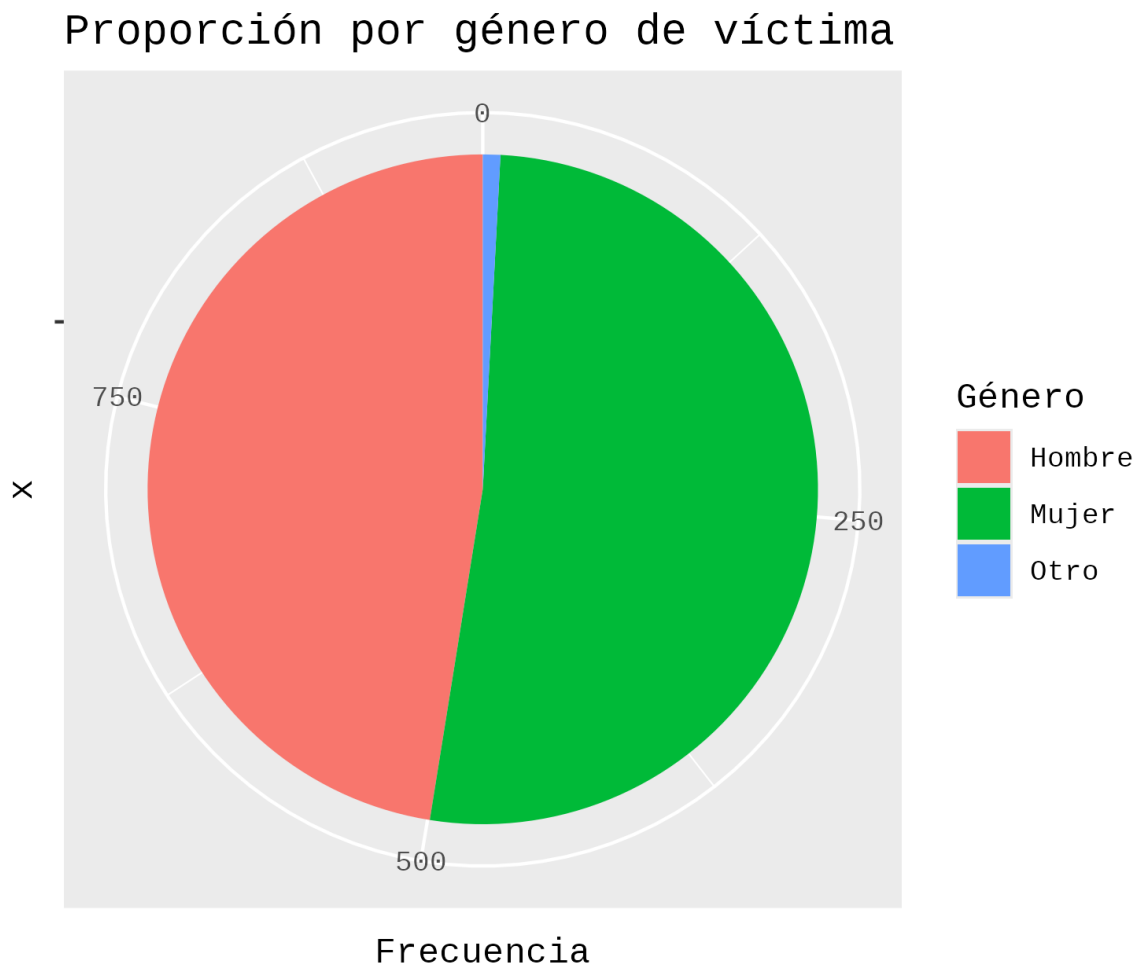
Esta función genera un gráfico de barras agrupadas que compara el género de la víctima con el del agresor. Cada barra representa la cantidad de casos por género de víctima, diferenciados por colores según el género del agresor. El gráfico se guarda como imagen PNG con el nombre 012_grafico_genero_cruzado.png en alta resolución.



Esta función genera un gráfico de barras que muestra la cantidad de casos registrados por comuna. Utiliza barras de color naranja y rota las etiquetas del eje X para facilitar su lectura. El objetivo es visualizar la distribución geográfica de los casos en el dataset. El gráfico se guarda como archivo PNG con el nombre 012_grafico_comuna.png en alta resolución.

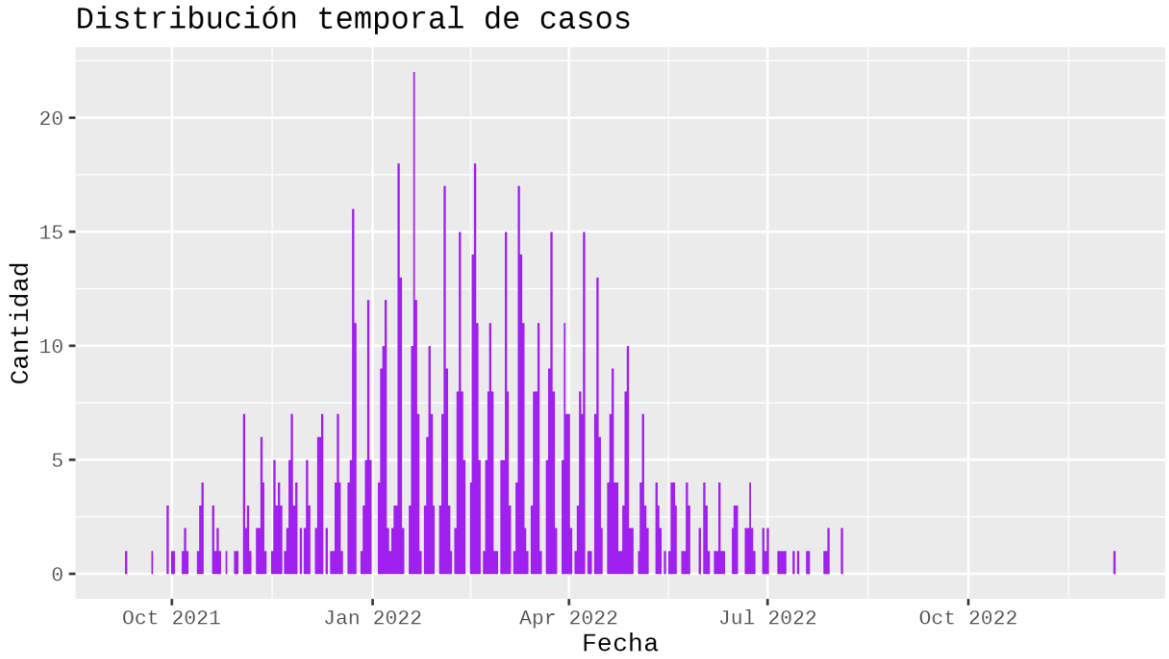


Esta función crea un gráfico de pastel que muestra la proporción de casos según el género de la víctima. Agrupa y resume los datos por género, y luego utiliza ggplot2 con coordenadas polares para representar visualmente la distribución porcentual. El gráfico se guarda como imagen PNG con el nombre 012_grafico_pastel_tipo.png en alta resolución.



Esta función genera un histograma que muestra la distribución temporal de los casos registrados, utilizando un intervalo de un día (86.400 segundos) como ancho de banda. El gráfico permite visualizar la frecuencia de casos a lo largo del tiempo y se guarda como imagen PNG con el nombre 012_grafico_tiempo.png en alta

resolución.


Distribución temporal de casos
Cantidad
20
15
10
5
0
Oct 2021
Jan 2022
Apr 2022
Jul 2022
Oct 2022
Fecha