Catedra 1

Dataset utilizado

Se requiere descargar el siguiente dataset para que funcione todo lo presentado en la catedra 1

https://github.com/ggalanc/Programacion_R/blob/main/Catedra_1/student_habits_performance.csv

Resumen del proyecto

Desarrollo de una plataforma web interactiva que permita la exploración y visualización de datos mediante la generación automática de gráficos personalizados, a partir de las variables contenidas en cualquier dataset cargado por el usuario. La herramienta estará diseñada para reconocer y adaptar dinámicamente su funcionamiento según la estructura del dataset. Esto facilitará el análisis de información sin requerir conocimientos técnicos avanzados, promoviendo una comprensión visual efectiva de los datos y apoyando la toma de decisiones basada en evidencia.

Inicio del Proyecto

Definición del Problema

El proyecto surge a partir de la necesidad de facilitar el análisis visual de grandes volúmenes de datos de forma accesible, intuitiva y sin requerimientos técnicos avanzados por parte del usuario. En muchos contextos educativos, investigativos y organizacionales, los usuarios cuentan con datasets extensos pero carecen de herramientas especializadas o habilidades analíticas para extraer valor de esta información.

Objetivo General

Desarrollar una plataforma web interactiva que permita a cualquier usuario cargar un dataset y, a partir de su estructura, generar gráficos personalizados de forma automática. Esta solución permitirá explorar visualmente los datos, detectar patrones y apoyar la toma de decisiones basadas en evidencia.

Justificación Técnica

Considerando que los datasets pueden ser de gran tamaño, el proyecto contempla desde su inicio la implementación de los servicios en la nube. Esto permitirá:

- Escalabilidad del procesamiento.
- Almacenamiento eficiente de grandes volúmenes de datos,
- Disponibilidad continua del servicio.

Recursos utilizados

- **Servicios en la nube (AWS)**
  - Instancia de Linux con limitaciones debido a la cuenta utilizada posee restricciones (1CPU, 1GB de ram)
- **Lenguaje de programación**
  - R + Shiny para desarrollo del servidor y la interfaz web.

Librerías de R utilizadas

- **library(shiny)**
  - Permite a los usuarios crear interfaces web dinámicas sin necesidad de escribir HTML, CSS o JavaScript directamente.
  - Utiliza el modelo de programación reactiva, lo que significa que los componentes se actualizan automáticamente cuando los datos cambian.
  - Ideal para construir paneles, visualizaciones de datos interactivas y prototipos de análisis.
- **library(ggplot2)**

  - Permite construir gráficos complejos a partir de capas lógicas (`geom_point`, `geom_line`, `geom_col`, etc.).
  - Cada gráfico se construye con: un conjunto de datos (`data`), un mapeo estético (`aes()`) para asignar variables a ejes, color, tamaño, etc. una o más capas geométricas (`geom_*`)

- **library(readr)**
  - Para leer datos tabulares (CSV, TSV, etc.) rápidamente y de forma segura.
  - Esencial cuando se trabaja con grandes volúmenes de datos en apps `shiny`.

- **library(rlang)**
  - Paquete base para manipular expresiones, símbolos, entornos y funciones de forma programática.
  - Permite trabajar con funciones reactivas que dependen de entradas del usuario, haciendo que las columnas seleccionadas en tiempo real sean interpretadas correctamente por `ggplot2`

- **library(colourpicker)**

  Proporciona una interfaz gráfica interactiva para seleccionar colores dentro de una aplicación `Shiny`. A través del widget `colourInput()`, los usuarios pueden escoger colores desde un selector visual (similar al selector de color en editores gráficos), escribir un valor hexadecimal manualmente, o elegir entre una paleta predefinida.

- **library(ggthemes)**

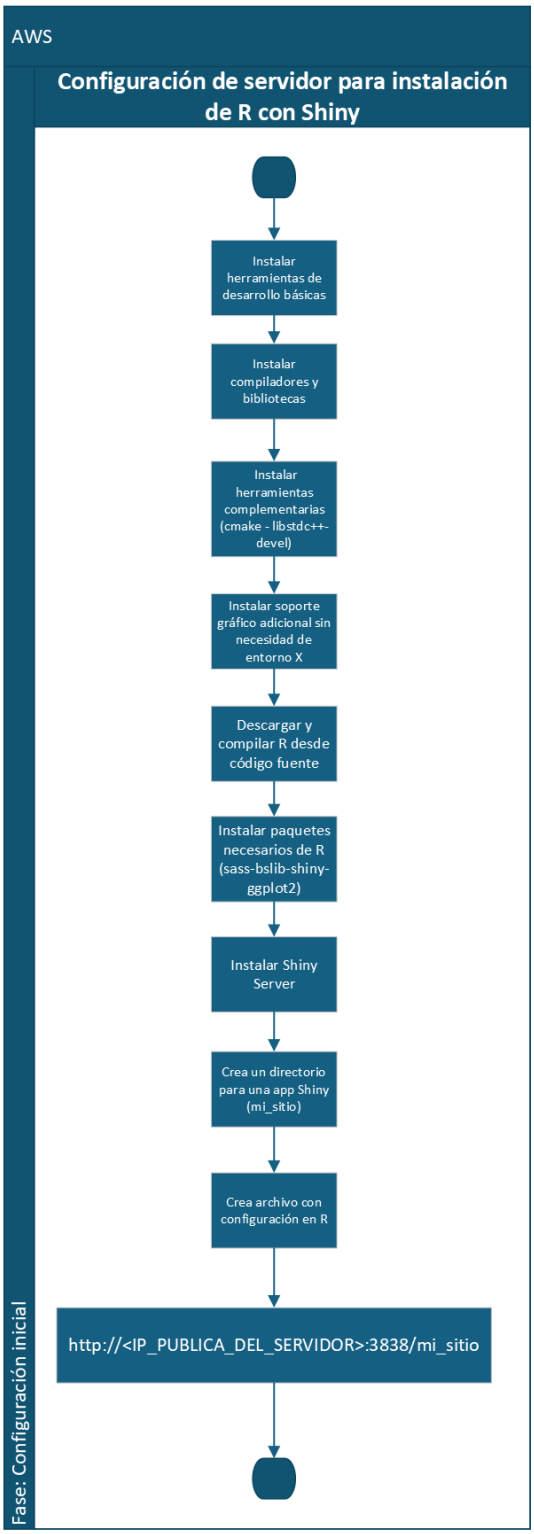
  Es una extensión para `ggplot2` que ofrece una amplia colección de temas adicionales y estilos gráficos preconfigurados. Estos temas permiten cambiar la estética general del gráfico con una sola línea de código.

- **library(scales)**

  Complementa `ggplot2` al proporcionar funciones para transformar, formatear y mapear escalas de datos. Esto incluye el formateo de ejes (por ejemplo, porcentajes, monedas, fechas), el ajuste de rangos y la normalización de datos.

Alcance inicial

Crear una versión funcional capaz de recibir datasets tabulares (CSV), reconocer su estructura automáticamente, y ofrecer visualizaciones básicas personalizadas según las variables identificadas.


AWS
Configuración de servidor para instalación de R con Shiny
Instalar herramientas de desarrollo básicas
Instalar compiladores y bibliotecas
Instalar herramientas complementarias (cmake - libstdc++-devel)
Instalar soporte gráfico adicional sin necesidad de entorno X
Descargar y compilar R desde código fuente
Instalar paquetes necesarios de R (sass-bslib-shiny-ggplot2)
Instalar Shiny Server
Crea un directorio para una app Shiny (mi_sitio)
Crea archivo con configuración en R
http://<IP_PUBLICA_DEL_SERVIDOR>:3838/mi_sitio
Fase: Configuración inicial

Configuración de servidor

Instalar herramientas de desarrollo y dependencias

Las siguientes herramientas permiten compilar R y sus paquetes desde el código fuente.

```
sudo yum groupinstall "Development Tools" -y

sudo yum install -y \
  gcc gcc-c++ gcc-gfortran \                      # Compiladores C, C++, Fortran
  readline-devel cairo-devel libX11-devel \       # Interfaces gráficas y lectura línea de co
  libXt-devel zlib-devel bzip2-devel xz-devel \   # Bibliotecas de compresión
  libcurl-devel openssl-devel wget tar make       # Descargas seguras y construcción de paque

sudo yum install -y cmake libstdc++-devel

# Soporte adicional para gráficos sin X11
sudo yum install -y \
  libpng-devel libjpeg-turbo-devel freetype-devel \
  fontconfig-devel mesa-libGL-devel mesa-libGLU-devel \
  cairo cairo-devel
```

Descargar y compilar R desde código fuente

```
cd /tmp
wget https://cran.r-project.org/src/base/R-4/R-4.3.2.tar.gz
tar -xzf R-4.3.2.tar.gz
cd R-4.3.2

# Configurar para uso compartido de bibliotecas y BLAS/LAPACK
./configure --enable-R-shlib --with-blas --with-lapack

# Compilar e instalar
make
sudo make install
```

Instalar paquetes necesarios de R

```
sudo su - -c "R -e \"install.packages('sass', repos = 'https://cran.rstudio.com')\""
sudo su - -c "R -e \"install.packages('bslib', repos = 'https://cran.rstudio.com')\""
sudo su - -c "R -e \"install.packages('shiny', repos = 'https://cran.rstudio.com')\""
```

```
sudo su - -c "R -e \"install.packages(c('ggplot2', 'readr', 'rlang'), repos = 'https://cran.
```

Instalar Shiny Server

```
cd /tmp
wget https://download3.rstudio.org/centos7/x86_64/shiny-server-1.5.20.1002-x86_64.rpm
sudo yum install -y --nogpgcheck shiny-server-1.5.20.1002-x86_64.rpm

# Verifica que esté activo
sudo systemctl status shiny-server
```

Crear aplicación de Shiny

```
# Creación de directorio

sudo mkdir -p /srv/shiny-server/mi_sitio

# Cambio de permisos de directorio para acceder al sitio via WEB

sudo chown -R shiny:shiny /srv/shiny-server/mi_sitio
```

Creación de archivo con configuración en R para el sitio

```
sudo vi /srv/shiny-server/mi_sitio/app.R
```

Ver aplicación de Shiny via WEB

```
#Ingresar a través del navegador

http://<IP_PUBLICA_DEL_SERVIDOR>:3838/mi_sitio
```

Características de la Aplicación Web

- Carga de archivos CSV.
- Interfaz dinámica para aplicar filtros automáticos según el tipo de dato.
- Selección de variables X, Y y opción de facetado.
- Visualización simultánea de hasta 3 gráficos:

  - Puntos
  - Barras
  - Líneas
  - Boxplot
  - Violin
  - Barras agrupadas
  - Facet Grid
  - Histograma
  - Densidad
- Descarga de gráficos generados
- Resumen estadístico automático del dataset.
- Sección de ayuda con descripción de los tipos de gráfico.

Codifciación de R

El archivo app.R se le agrega el siguiente código:

```
# Carga las bibliotecas necesarias
library(shiny)        # Para crear aplicaciones web interactivas
library(ggplot2)      # Para la generación de gráficos
library(readr)        # Para leer archivos CSV
library(rlang)        # Para la manipulación de expresiones
library(colourpicker) # Permite seleccionar colores desde la interfaz
```

```
Adjuntando el paquete: 'colourpicker'

The following object is masked from 'package:shiny':

    runExample
```

```
library(ggthemes)     # Temas adicionales para ggplot2
library(scales)       # Para mejorar la escala en los gráficos
```

```
Adjuntando el paquete: 'scales'

The following object is masked from 'package:readr':

    col_factor
```

```r
# Configura opciones de Shiny
options(shiny.usecairo = TRUE)                       # Mejora la calidad del renderizado gráfi
options(shiny.maxRequestSize = 200*1024^2)           # Permite subir archivos de hasta 200 MB

# Función personalizada para leer CSV con distintos posibles formatos
read_csv_auto <- function(file) {
  tryCatch({
    read_delim(file, delim = ",", locale = locale(encoding = "UTF-8"), show_col_types = FALSE
  }, error = function(e1) {
    tryCatch({
      read_delim(file, delim = ",", locale = locale(encoding = "ISO-8859-1"), show_col_types
    }, error = function(e2) {
      read_delim(file, delim = ";", locale = locale(encoding = "ISO-8859-1"), show_col_types
    })
  })
}

# Interfaz de usuario
ui <- fluidPage(
  # Título de la aplicación
  titlePanel("Visualizador interactivo de dataset"),
  # Diseño general con barra lateral y panel principal
  sidebarLayout(
    sidebarPanel(
      fileInput("csvfile", "Sube tu archivo CSV", accept = ".csv"), # Carga del archivo CSV
      uiOutput("filters_ui"),    # Filtros dinámicos por variable
      uiOutput("xcol_ui"),       # Selector de columna X
      uiOutput("ycol_ui"),       # Selector de columna Y
      selectInput("chartType", "Tipo de gráfico",
                  choices = c("Puntos" = "point", "Barras" = "col", "Líneas" = "line", "Boxpl
      colourInput("colorInput", "Color del gráfico", value = "#2C3E50"), # Selector de color
      selectInput("chartType2", "Segundo gráfico", choices = c("Histograma" = "hist", "Boxplo
      uiOutput("ycol2_ui"),
      selectInput("chartType3", "Tercer gráfico", choices = c("Densidad" = "density", "Líneas
      uiOutput("ycol3_ui"),
```

```
      uiOutput("facet_ui"),
      downloadButton("descargar", "Descargar gráfico") # Botón de descarga
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Vista previa", tableOutput("preview")),  # Tabla de muestra del dataset
        tabPanel("Gráfico",
                 fluidRow(
                   column(4, plotOutput("grafico")),
                   column(4, plotOutput("grafico2")),
                   column(4, plotOutput("grafico3"))
                 )),
        tabPanel("Estadísticas", verbatimTextOutput("summary")), # Estadísticas descriptivas
        tabPanel("Ayuda",  # Instrucciones sobre gráficos
                 h4("¿Cuál gráfico elegir?"),
                 tags$ul(
                   tags$li(strong("Puntos"), ": Muestra la relación entre dos variables numér
                   tags$li(strong("Barras"), ": Representa cantidades por categoría (X catego
                   tags$li(strong("Líneas"), ": Útil para datos temporales o secuenciales.")
                   tags$li(strong("Boxplot"), ": Compara distribuciones y detecta valores ati
                   tags$li(strong("Violin"), ": Visualiza distribución completa + mediana, út
                   tags$li(strong("Barras agrupadas"), ": Comparación entre categorías cruzad
                   tags$li(strong("Facet Grid"), ": Divide el gráfico en subgráficos por valo
                 ),
                 p("Elige las columnas adecuadas según el gráfico para obtener mejores result
        )
      )
    )
  )
)



# Define la lógica del servidor de Shiny
server <- function(input, output, session) {

  # Lectura reactiva del dataset crudo desde el archivo CSV cargado
  raw_dataset <- reactive({
    req(input$csvfile)  # Asegura que el archivo haya sido cargado
    df <- read_csv_auto(input$csvfile$datapath)  # Lee el archivo usando la función personali
    names(df) <- make.names(names(df))  # Asegura que los nombres de columnas sean válidos
    return(df)
```

```
})

# Genera la interfaz dinámica para los filtros según el tipo de datos
output$filters_ui <- renderUI({
  req(raw_dataset())
  df <- raw_dataset()
  ui_list <- list()
  for (col in names(df)) {
    if (is.numeric(df[[col]])) {
      rng <- range(df[[col]], na.rm = TRUE)
      ui_list[[col]] <- sliderInput(paste0("filter_", col), paste("Filtrar", col),
                                    min = rng[1], max = rng[2], value = rng)
    } else if (is.character(df[[col]]) || is.factor(df[[col]])) {
      choices <- unique(df[[col]])
      if (length(choices) <= 20) {
        ui_list[[col]] <- selectInput(paste0("filter_", col), paste("Filtrar", col),
                                      choices = choices, selected = choices, multiple = TRU
      }
    }
  }
  return(ui_list)
})

# Aplica los filtros seleccionados por el usuario sobre el dataset original
dataset <- reactive({
  df <- raw_dataset()
  for (col in names(df)) {
    id <- paste0("filter_", col)
    if (!is.null(input[[id]])) {
      if (is.numeric(df[[col]]) && length(input[[id]]) == 2) {
        df <- df[df[[col]] >= input[[id]][1] & df[[col]] <= input[[id]][2], ]
      } else if (is.character(df[[col]]) || is.factor(df[[col]])) {
        df <- df[df[[col]] %in% input[[id]], ]
      }
    }
  }
  return(df)
})

# UI dinámica para seleccionar la variable X del gráfico
output$xcol_ui <- renderUI({
  req(dataset())
```

```
  selectInput("xcol", "Columna X", choices = names(dataset()))
})

# UI para variable Y numérica para el primer gráfico
output$ycol_ui <- renderUI({
  req(dataset())
  num_cols <- names(dataset())[sapply(dataset(), is.numeric)]
  selectInput("ycol", "Columna Y (numérica)", choices = num_cols)
})

# UI para variable Y del segundo gráfico
output$ycol2_ui <- renderUI({
  req(dataset())
  num_cols <- names(dataset())[sapply(dataset(), is.numeric)]
  selectInput("ycol2", "Columna Y para gráfico 2", choices = num_cols)
})

# UI para variable Y del tercer gráfico
output$ycol3_ui <- renderUI({
  req(dataset())
  num_cols <- names(dataset())[sapply(dataset(), is.numeric)]
  selectInput("ycol3", "Columna Y para gráfico 3", choices = num_cols)
})

# UI para seleccionar la variable de facetado
output$facet_ui <- renderUI({
  req(dataset())
  df <- dataset()
  cat_cols <- names(df)[sapply(df, function(x) is.character(x) || is.factor(x))]
  choices <- if (length(cat_cols) > 0) c("Seleccione una variable" = "", cat_cols) else c("
  selectInput("facetCol", "Variable para Facetado (opcional)", choices = choices)
})

# Renderiza el segundo gráfico (Histograma o Boxplot)
output$grafico2 <- renderPlot({
  req(input$xcol, input$ycol2, input$chartType2)
  df <- dataset()
  p <- ggplot(df, aes_string(x = input$xcol, y = input$ycol2))
  if (input$chartType2 == "hist") {
    p <- ggplot(df, aes_string(x = input$ycol2)) +
      geom_histogram(fill = input$colorInput, bins = 30, color = "white")
  } else {
```

```
    p <- p + geom_boxplot(fill = input$colorInput)
  }
  if (!is.null(input$facetCol) && input$facetCol != "") {
    p <- p + facet_wrap(as.formula(paste("~", input$facetCol)))
  }
  p + theme_minimal() + labs(title = paste("Gráfico 2:", input$chartType2))
})

# Renderiza el tercer gráfico (Densidad o Líneas)
output$grafico3 <- renderPlot({
  req(input$xcol, input$ycol3, input$chartType3)
  df <- dataset()
  p <- ggplot(df, aes_string(x = input$xcol, y = input$ycol3))
  if (input$chartType3 == "density") {
    p <- ggplot(df, aes_string(x = input$ycol3)) +
      geom_density(fill = input$colorInput, alpha = 0.6, color = "#333333")
  } else {
    p <- p + geom_line(color = input$colorInput, linewidth = 1.2)
  }
  if (!is.null(input$facetCol) && input$facetCol != "") {
    p <- p + facet_wrap(as.formula(paste("~", input$facetCol)))
  }
  p + theme_minimal() + labs(title = paste("Gráfico 3:", input$chartType3))
})

# Muestra una vista previa de las primeras 5 filas del dataset
output$preview <- renderTable({
  head(dataset(), 5)
})

# Renderiza el gráfico principal en la pestaña "Gráfico"
output$grafico <- renderPlot({
  req(input$xcol, input$ycol, input$chartType)
  gg <- ggplot(dataset(), aes_string(x = input$xcol, y = input$ycol))
  tipo <- input$chartType
  if (tipo == "col") {
    gg <- gg + geom_col(fill = input$colorInput)
  } else if (tipo == "line") {
    gg <- gg + geom_line(color = input$colorInput, linewidth = 1.2)
  } else if (tipo == "boxplot") {
    gg <- gg + geom_boxplot(outlier.color = "red", fill = input$colorInput)
  } else if (tipo == "violin") {
```

```
    gg <- gg + geom_violin(fill = input$colorInput, color = "black")
  } else if (tipo == "dodgebar") {
    gg <- gg + geom_bar(aes_string(fill = input$xcol), position = "dodge", stat = "identity
  } else if (tipo == "facet") {
    gg <- gg + geom_point(color = input$colorInput)
    if (!is.null(input$facetCol) && input$facetCol != "") {
      gg <- gg + facet_wrap(as.formula(paste("~", input$facetCol)))
    }
  } else {
    gg <- gg + geom_point(size = 3, color = input$colorInput)
  }
  gg + theme_minimal()
})

# Genera un gráfico base reutilizable para descargar
grafico_base <- reactive({
  req(input$xcol, input$ycol, input$chartType)
  df <- dataset()
  tipo <- input$chartType
  gg <- ggplot(df, aes_string(x = input$xcol, y = input$ycol))
  if (tipo == "col") {
    gg <- gg + geom_col(fill = input$colorInput)
  } else if (tipo == "line") {
    gg <- gg + geom_line(color = input$colorInput, linewidth = 1.2)
  } else if (tipo == "boxplot") {
    gg <- gg + geom_boxplot(outlier.color = "red", fill = input$colorInput)
  } else if (tipo == "violin") {
    gg <- gg + geom_violin(fill = input$colorInput, color = "black")
  } else if (tipo == "dodgebar") {
    gg <- gg + geom_bar(aes_string(fill = input$xcol), position = "dodge", stat = "identity
  } else if (tipo == "facet") {
    gg <- gg + geom_point(color = input$colorInput)
    if (!is.null(input$facetCol) && input$facetCol != "") {
      gg <- gg + facet_wrap(as.formula(paste("~", input$facetCol)))
    }
  } else {
    gg <- gg + geom_point(size = 3, color = input$colorInput)
  }
  gg + theme_minimal()
})

# Permite descargar el gráfico en formato PNG
```

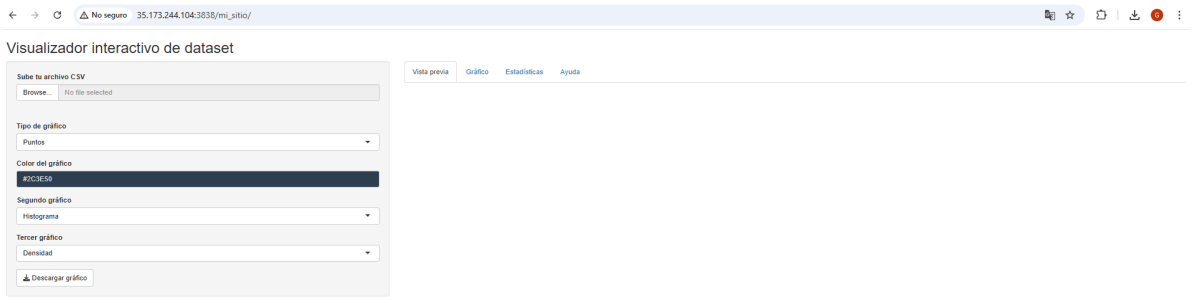
```
  output$descargar <- downloadHandler(
    filename = function() {
      paste0("grafico_", input$chartType, ".png")
    },
    content = function(file) {
      ggsave(file, plot = grafico_base(), width = 7, height = 5)
    }
  )

  # Muestra un resumen estadístico del dataset
  output$summary <- renderPrint({
    summary(dataset())
  })
}

# Ejecuta la aplicación Shiny
shinyApp(ui, server)
```

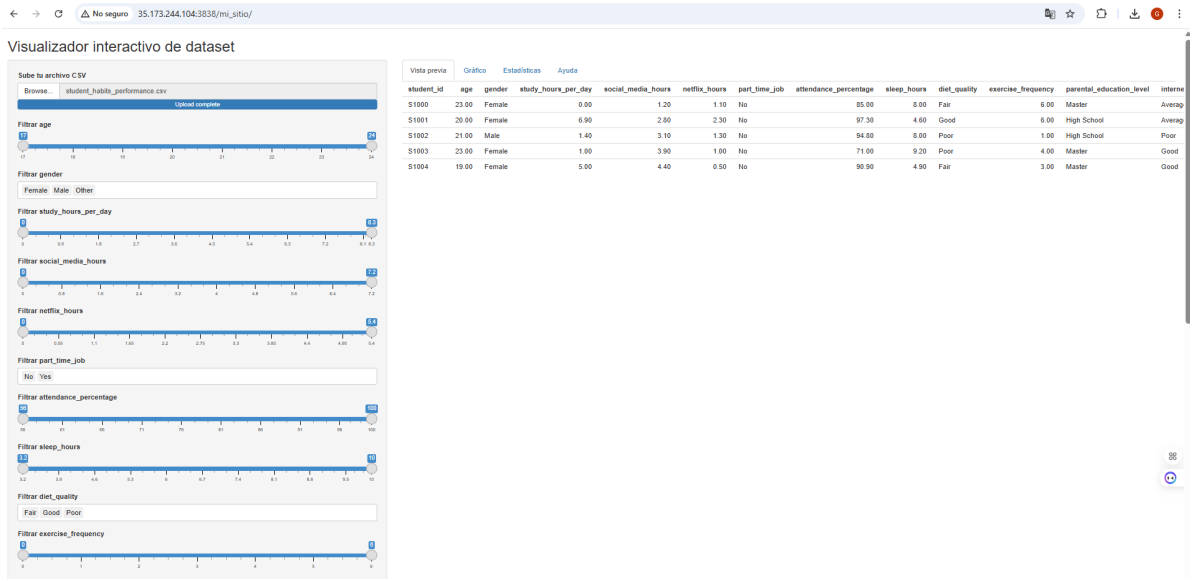
Imagenes de ejemplo del servidor

Visualización sin dataset cargado



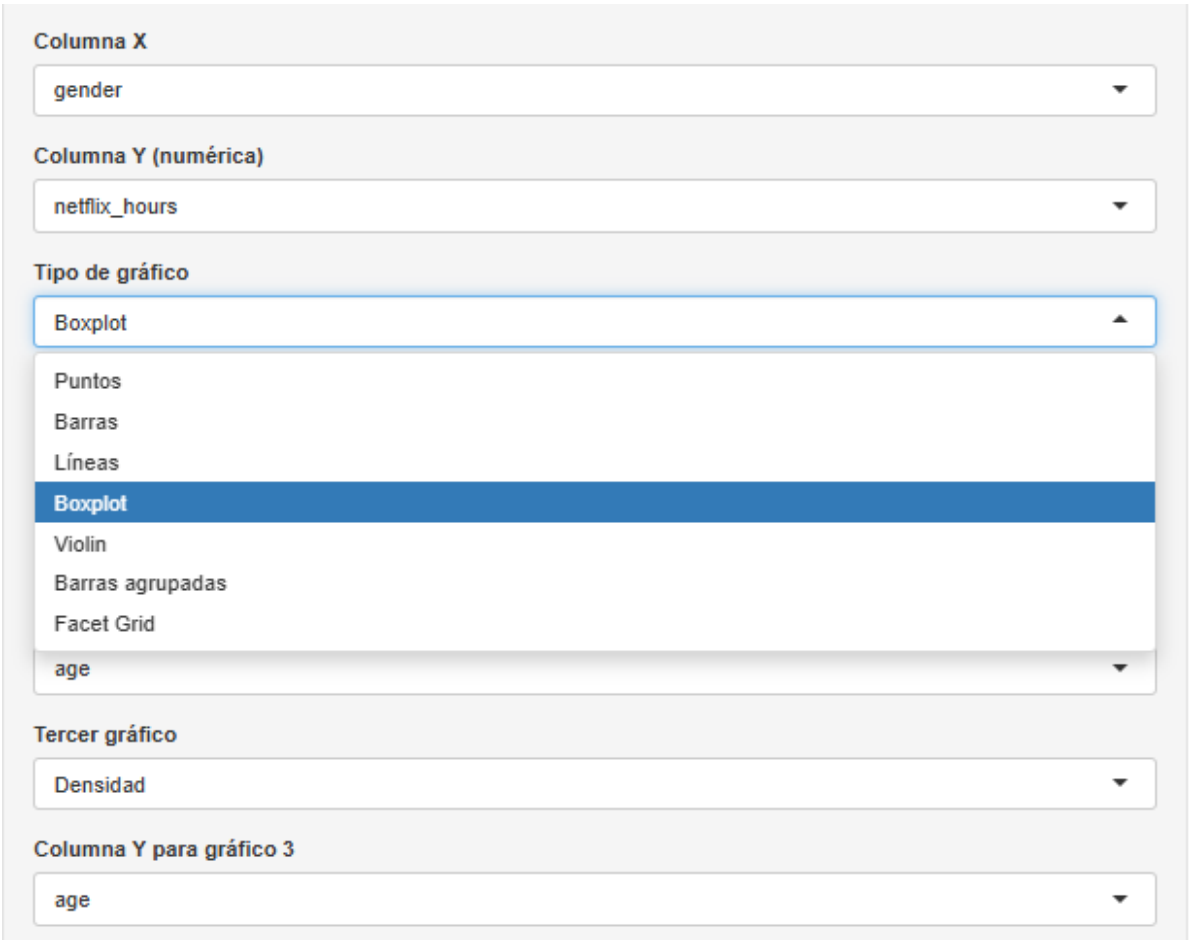
Visualización con dataset cargado

Al cargar el dataset muestra un ejemplo con las primeras cinco filas del dataset. Además, se carga de manera automática un filtro para ir seleccionado.



Selección de Gráficos

Dentro de las opciones agregadas a través de R, generamos una opción de listar para escoger los gráficos que sean necesarios para analizar la información, se utilizaron 6 tipos de gráficos para poder ser seleccionados.

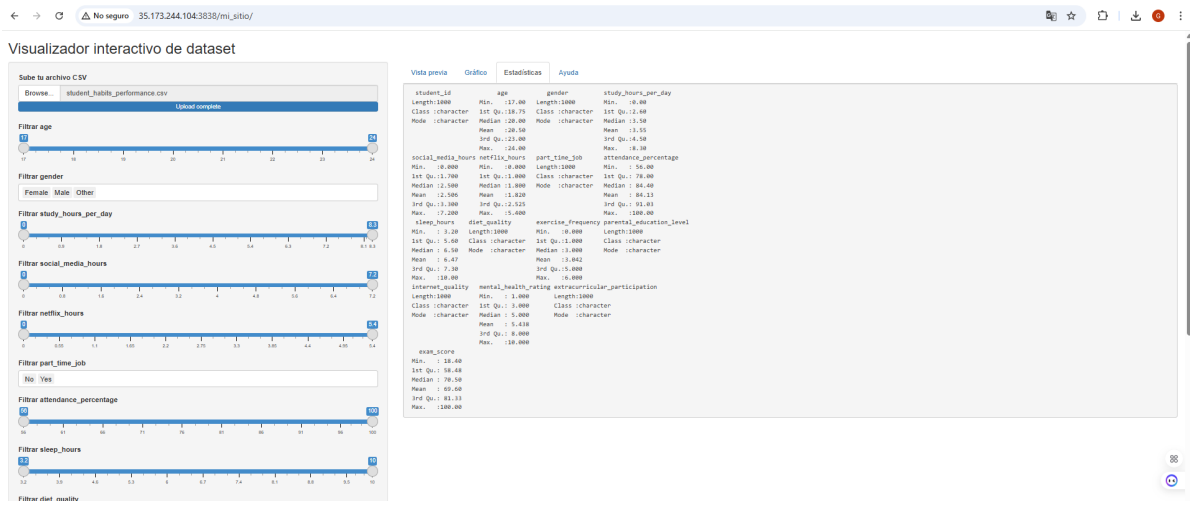


Visualización de gráficos

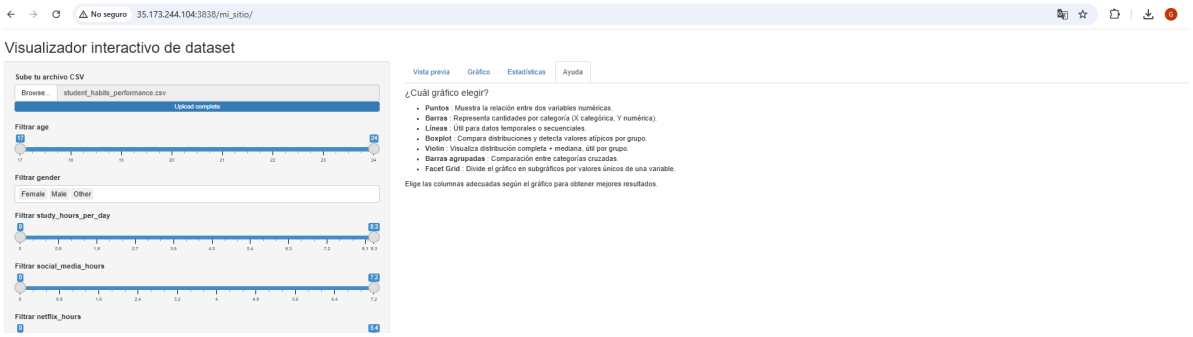
En la opción de visualización se programo en R para que en Shinny aparecieran 3 gráficos los cuales pueden ser modificados de acuerdo con el filtro demostrado en el ítem anterior.

Estadísiticas

Se agrega una sección con las estadísticas del dataset cargado



Opción ayuda



Análisis de dataset

Introducción del dataset

El análisis se basa en el dataset `student_habits_performance.csv`, que contiene información sobre 1.000 estudiantes universitarios, incluyendo datos personales, hábitos diarios y puntajes de exámenes. El objetivo principal es identificar patrones que expliquen diferencias de rendimiento académico.

Problema:

¿Cómo influyen los hábitos diarios de estudio, sueño y redes sociales en el rendimiento académico de los estudiantes?

Hipótesis:

Los estudiantes que dedican más horas al estudio y menos tiempo a redes sociales obtienen mejores resultados en el examen final.

Gráfico 1

Horas de estudio y puntaje de examen

Este gráfico de dispersión muestra una fuerte correlación positiva entre las horas de estudio y el puntaje en el examen. A medida que aumentan las horas dedicadas al estudio diario, también lo hace el rendimiento académico.

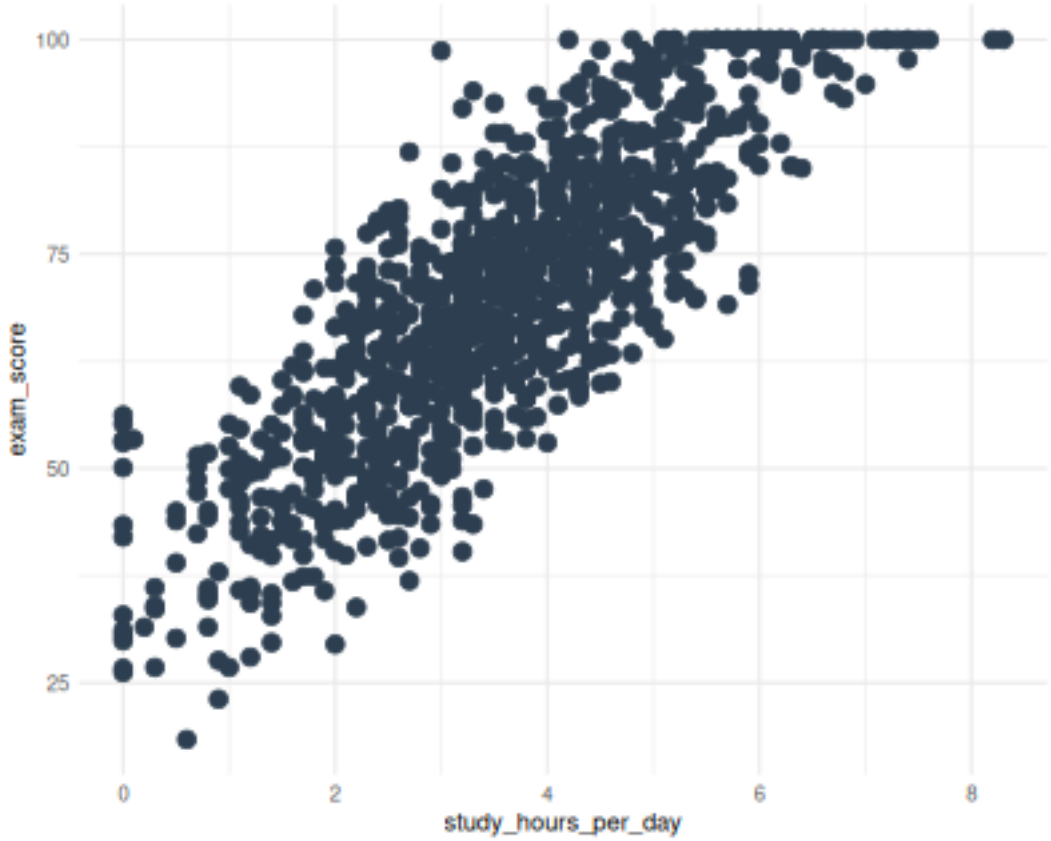
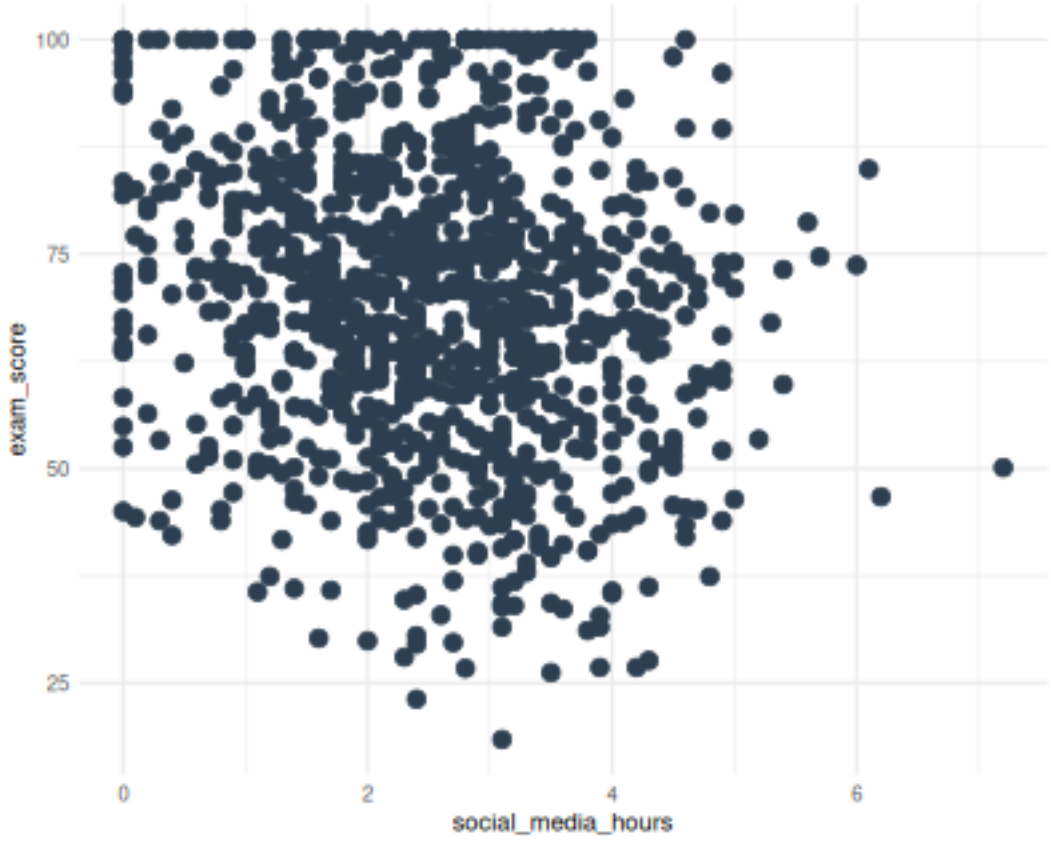


Gráfico 2

Horas de redes sociales y puntaje de examen

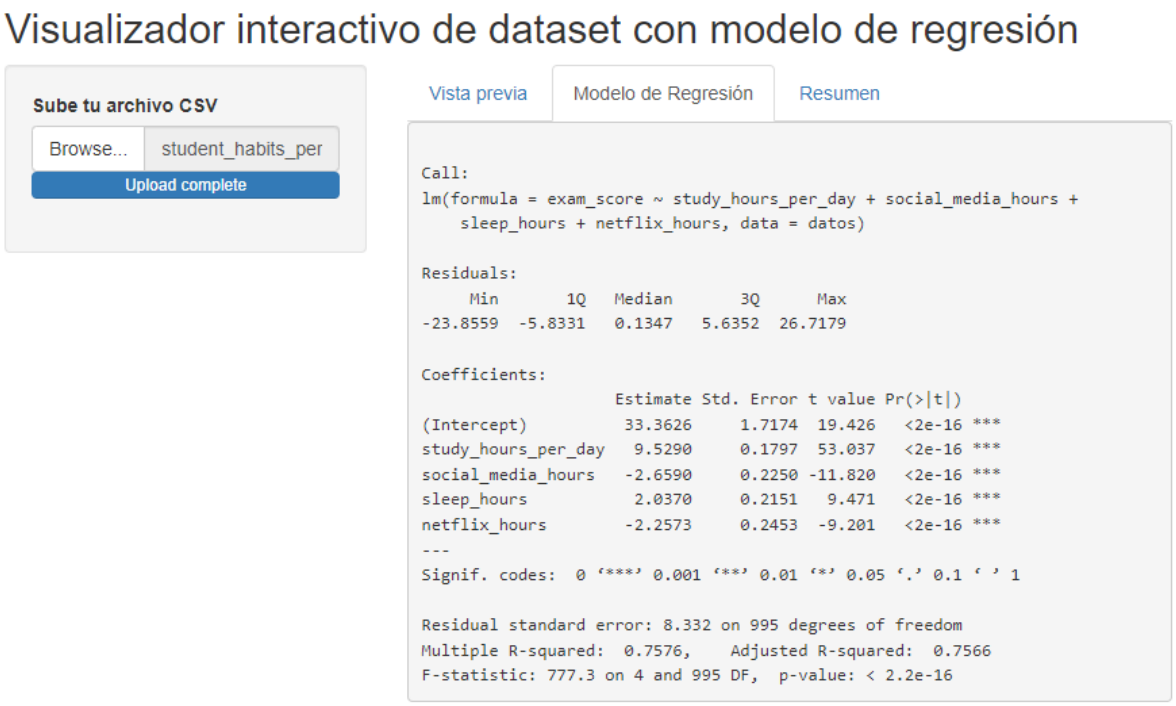
Existe una tendencia negativa débil entre las horas en redes sociales y el desempeño en el examen. Aunque no es una correlación fuerte, se sugiere que el uso excesivo puede afectar el rendimiento académico.



Recomendación basada en los datos

Para mejorar el rendimiento académico, es más efectivo aumentar las horas de estudio que simplemente reducir el uso de redes sociales, aunque limitar este último puede ayudar a evitar distracciones.

Modelo de Regresión lineal




```r
# Carga las bibliotecas necesarias
library(shiny)
library(ggplot2)
library(readr)
library(rlang)
library(colourpicker)
library(ggthemes)
library(scales)

# Opciones globales de Shiny
options(shiny.usecairo = TRUE)
options(shiny.maxRequestSize = 200*1024^2)

# Función para intentar leer CSV con distintos formatos
read_csv_auto <- function(file) {
  tryCatch({
    read_delim(file, delim = ",", locale = locale(encoding = "UTF-8"), show_col_types = FALSE
  }, error = function(e1) {
    tryCatch({
```

```
      read_delim(file, delim = ",", locale = locale(encoding = "ISO-8859-1"), show_col_types
    }, error = function(e2) {
      read_delim(file, delim = ";", locale = locale(encoding = "ISO-8859-1"), show_col_types
    })
  })
}

# Interfaz de usuario
ui <- fluidPage(
  titlePanel("Visualizador interactivo de dataset con modelo de regresión"),
  sidebarLayout(
    sidebarPanel(
      fileInput("csvfile", "Sube tu archivo CSV", accept = ".csv")
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Vista previa", tableOutput("preview")),
        tabPanel("Modelo de Regresión", verbatimTextOutput("regresion")),
        tabPanel("Resumen", verbatimTextOutput("summary"))
      )
    )
  )
)

# Lógica del servidor
server <- function(input, output, session) {
  raw_dataset <- reactive({
    req(input$csvfile)
    df <- read_csv_auto(input$csvfile$datapath)
    names(df) <- make.names(names(df))
    return(df)
  })

  dataset <- reactive({
    raw_dataset()
  })

  # Modelo de regresión lineal múltiple
modelo_regresion <- reactive({
  datos <- dataset()
  columnas_requeridas <- c("exam_score", "study_hours_per_day", "social_media_hours", "sleep
```

```
  if (!all(columnas_requeridas %in% names(datos))) {
    return(NULL)
  }

  lm(exam_score ~ study_hours_per_day + social_media_hours + sleep_hours + netflix_hours, da
})

  # Mostrar el resumen del modelo
output$regresion <- renderPrint({
  modelo <- modelo_regresion()
  if (is.null(modelo)) {
    cat("No se puede ajustar el modelo: faltan columnas necesarias en el dataset.\n")
  } else {
    summary(modelo)
  }
})

  output$preview <- renderTable({
    head(dataset(), 5)
  })

  output$summary <- renderPrint({
    summary(dataset())
  })
}

# Ejecutar la aplicación
shinyApp(ui, server)
```

Se construyó un modelo de regresión lineal para predecir el **puntaje del examen** (`exam_score`) en función de cuatro variables predictoras:

- **Horas de estudio por día** (`study_hours_per_day`)
- **Horas en redes sociales** (`social_media_hours`)
- **Horas de sueño** (`sleep_hours`)
- **Horas viendo Netflix** (`netflix_hours`)

Resultados del modelo

- **R^2 ajustado**: 0.7566
  El modelo explica aproximadamente el 75.66% de la variabilidad en el puntaje del examen, lo que indica un buen poder predictivo.
- **Estadístico F**: 777.3, con un p-valor $<$ 2.2e-16
  Este resultado es altamente significativo, lo que indica que el modelo en su conjunto es útil para explicar el resultado.

Interpretación de los coeficientes

Variable	Coeficiente	Interpretación
(Intercepto)	33.3626	Valor esperado del puntaje del examen cuando todas las variables independientes son cero.
Horas de estudio	+9.5290	Por cada hora adicional de estudio al día, el puntaje del examen aumenta en promedio 9.53 puntos.
Horas en redes	-2.6590	Por cada hora adicional en redes sociales, el puntaje disminuye en promedio 2.66 puntos.
Horas de sueño	+2.0370	Dormir más se asocia positivamente con el puntaje del examen (cada hora extra suma 2.04 puntos).
Horas de Netflix	-2.2753	Por cada hora adicional viendo Netflix, el puntaje baja en promedio 2.28 puntos.

Conclusión

El modelo sugiere que estudiar más y dormir mejor están positivamente relacionados con un mejor desempeño en los exámenes, mientras que dedicar más tiempo a redes sociales o ver Netflix impacta negativamente.