EXAMEN
PROGRAMACIÓN EN R

Mabel Herrera / Gerardo Galán

10 de julio de 2025

Dataset utilizado

Se requiere descargar el siguiente dataset para que funcione todo lo presentado en nuestro exámen

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.
- **library(vroom)**

  Es una herramienta altamente eficiente para leer archivos de texto delimitados (como CSV, TSV, etc.). Está diseñado para ser más rápido y eficiente en memoria que funciones tradicionales como `read.csv()`

- **library(crayon)**

  Se utiliza para **agregar colores y estilos al texto impreso en la consola**. Es especialmente útil para mejorar la legibilidad de mensajes, errores o logs cuando se ejecutan scripts, pruebas o aplicaciones en la consola R.

- **library(shinycssloaders)**

  Permite **agregar indicadores de carga (spinners)** a los elementos de salida (`output`) en aplicaciones **Shiny**, para mejorar la experiencia del usuario mientras se renderizan gráficos, tablas, textos, etc.

- **library(httr)**

  Se utiliza para realizar peticiones HTTP (GET, POST, PUT, DELETE, etc.), lo que permite conectar el código R con APIs web, enviar datos, recibir respuestas y trabajar con servicios RESTful

- **library(jsonlite)**

  Permite trabajar con datos en formato **JSON** (JavaScript Object Notation), que es ampliamente utilizado en APIs, almacenamiento de configuración, y estructuras de datos jerárquicas.

- **library(testthat)**

  Es el estándar para realizar pruebas unitarias y de integración en R. Es ampliamente utilizado para asegurar que funciones y aplicaciones (como paquetes o apps Shiny) funcionen correctamente y sin errores.

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.

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 comandos
  libXt-devel zlib-devel bzip2-devel xz-devel \   # Bibliotecas de compresión
  libcurl-devel openssl-devel wget tar make       # Descargas seguras y construcción de paquetes

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.rstudio.c
```

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 de gráficos:
  - Puntos
  - Barras
  - Líneas
  - Boxplot
  - Violin
  - Barras agrupadas
  - Facet Grid
  - Histograma
  - Densidad
- Descarga de gráficos generados
- Resumen estadístico automático del dataset.
- Análisis con IA del dataset cargado

Codifciación de R

Se crea un archivo llamado app.R el cual se carga en el servidor. Se le agrega el siguiente código:

```
# app.R - Aplicación Shiny integrada con análisis IA local y por API

# ========================================================
# LIBRERÍAS Y CONFIGURACIÓN GLOBAL
# Carga de paquetes necesarios para toda la aplicación.
# ========================================================
library(shiny)
```

```
Warning: package 'shiny' was built under R version 4.5.1
```

```
library(ggplot2)
library(readr)
library(rlang)
library(colourpicker)
```

```
Adjuntando el paquete: 'colourpicker'

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

    runExample
```

```
library(ggthemes)
library(scales)
```

```
Adjuntando el paquete: 'scales'

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

    col_factor
```

```
library(vroom)
```

```
Adjuntando el paquete: 'vroom'
```

```
The following object is masked from 'package:scales':

    col_factor

The following objects are masked from 'package:readr':

    as.col_spec, col_character, col_date, col_datetime, col_double,
    col_factor, col_guess, col_integer, col_logical, col_number,
    col_skip, col_time, cols, cols_condense, cols_only, date_names,
    date_names_lang, date_names_langs, default_locale, fwf_cols,
    fwf_empty, fwf_positions, fwf_widths, locale, output_column,
    problems, spec
```

```
library(crayon)
```

```
Adjuntando el paquete: 'crayon'

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

    chr

The following object is masked from 'package:ggplot2':

    %+%
```

```
library(shinycssloaders)
library(httr)
library(jsonlite)
```

```
Adjuntando el paquete: 'jsonlite'

The following objects are masked from 'package:rlang':

    flatten, unbox

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

    validate
```

```
library(testthat)
```

```
Warning: package 'testthat' was built under R version 4.5.1

Adjuntando el paquete: 'testthat'
```

```
The following object is masked from 'package:vroom':

    matches

The following objects are masked from 'package:rlang':

    is_false, is_null, is_true

The following objects are masked from 'package:readr':

    edition_get, local_edition
```

```r
#CONFIGURACIONES GLOBALES
options(shiny.usecairo = TRUE) # Mejor calidad de renderizado de gráficos
options(shiny.maxRequestSize = 1024*1024^2) # Aumentar el tamaño máximo de archivo que puede soportar (1 GB)


# ========================================================
# FUNCIONES AUXILIARES
# ========================================================
# contiene funciones reutilizables: una para limpiar texto ANSI de errores (limpiar_ansi) y otra para cargar archivos CSV detectando automáticamente el delimitador (read_csv_auto).

limpiar_ansi <- function(msg) {
  gsub("\\033\\[[0-9;]*[a-zA-Z]", "", msg)
}

# Carga archivos CSV detectando automáticamente el delimitador correcto
read_csv_auto <- function(file) {
  delimitadores <- c(",", ";", "    ", "|")
  for (delim in delimitadores) {
    try({
      df <- vroom(file, delim = delim, locale = locale(encoding = "UTF-8"), progress = FALSE)
      if (ncol(df) > 1) return(df)
    }, silent = TRUE)
  }
  for (delim in delimitadores) {
    try({
      df <- read_delim(file, delim = delim, locale = locale(encoding = "UTF-8"), show_col_types = FALSE, skip_empty_rows = TRUE)
      if (ncol(df) > 1) return(df)
    }, silent = TRUE)
  }

# Notifica al usuario si falla

  showNotification("Error al leer el archivo.", type = "error", duration = 10)
  return(NULL)
}

# ========================================================
# INTERFAZ DE USUARIO (UI)
# Define todos los componentes visuales e interacciones.
# ========================================================
```

```r
ui <- fluidPage(
  titlePanel("Visualizador interactivo con análisis IA"),
  sidebarLayout(
    sidebarPanel(
      fileInput("csvfile", "Sube tu archivo CSV", accept = ".csv"),
      passwordInput("api_key_input", "Clave API de OpenAI (local)", value = ""),
      uiOutput("filters_ui"),
      uiOutput("xcol_ui"),
      uiOutput("ycol_ui"),

# Selector de tipo de gráfico

      selectInput("chartType", "Tipo de gráfico",
                  choices = c("Puntos" = "point", "Barras" = "col", "Líneas" = "line",
                   ↪  "Boxplot" = "boxplot", "Violin" = "violin", "Barras agrupadas" =
                   ↪  "dodgebar", "Facet Grid" = "facet")),
      colourInput("colorInput", "Color del gráfico", value = "#2C3E50"),
      uiOutput("facet_ui"),
      downloadButton("descargar", "Descargar gráfico")
    ),
    mainPanel(
      tabsetPanel(
        tabPanel("Vista previa", tableOutput("preview")),
        tabPanel("Gráfico", withSpinner(plotOutput("grafico"), type = 6)),
        tabPanel("Estadísticas", verbatimTextOutput("summary")),
        tabPanel("Análisis IA",
          tabsetPanel(
            tabPanel("Análisis Local", verbatimTextOutput("analisis_local")),
            tabPanel("Análisis por API",
                     textAreaInput("user_question", "Tu pregunta al modelo", placeholder =
                      ↪  "Ej: ¿Qué patrones encuentras en las variables numéricas?", rows =
                      ↪  3),
                     actionButton("api_button", "Enviar a API para análisis"),
                     verbatimTextOutput("analisis_api")
            ),
            tabPanel("Elegir análisis",
              selectInput("tipo_analisis", "Tipo de análisis",
                          choices = c("Media y Desviación" = "media_sd", "Valores únicos" =
                           ↪  "unicos")),
              verbatimTextOutput("analisis_manual")
            )
          )
        )
      )
    )
  )
)

# ======================================================
# FUNCIÓNES DEL SERVIDOR
# Contiene toda la lógica reactiva: carga de datos, aplicación de filtros, generación de
 ↪  gráficos, análisis locales, conexión con API y estadísticas.
# ======================================================

server <- function(input, output, session) {
```

```
# Carga de archivo original y estandariza nombres de columnas

  raw_dataset <- reactive({
    req(input$csvfile)
    df <- read_csv_auto(input$csvfile$datapath)
    names(df) <- make.names(names(df))
    return(df)
  })

# Aplica filtros dinámicos sobre el dataset cargado

  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)
  })


#  Filtros dinámicos para variables según su tipo

  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 = TRUE)
        }
      }
    }
    return(ui_list)
  })


# Selección de columnas X e Y para gráficos

  output$xcol_ui <- renderUI({
    req(dataset())
    selectInput("xcol", "Columna X", choices = names(dataset()))
  })
```

```
  output$ycol_ui <- renderUI({
    req(dataset())
    num_cols <- names(dataset())[sapply(dataset(), is.numeric)]
    selectInput("ycol", "Columna Y (numérica)", choices = num_cols)
  })

  output$facet_ui <- renderUI({
    req(dataset())
    df <- dataset()
    cat_cols <- names(df)[sapply(df, function(x) is.character(x) || is.factor(x))]
    selectInput("facetCol", "Variable para Facetado (opcional)", choices = c("Ninguno" = "",
     ↪  cat_cols))
  })


# Visualización gráfica con ggplot2

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

# Vista previa y estadísticas descriptivas

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

# Descargar gráfico generado

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

```
# Análisis automático local (media y desviación estándar)

  output$analisis_local <- renderPrint({
    req(dataset())
    df <- dataset()
    resumen <- lapply(df[sapply(df, is.numeric)], function(col) {
      c(Media = mean(col, na.rm = TRUE), SD = sd(col, na.rm = TRUE))
    })
    print(resumen)
  })

# Análisis mediante API externa (OpenRouter / GPT)

  output$analisis_api <- renderPrint({
    input$api_button
    isolate({
      df <- head(dataset(), 100)
      json_data <- toJSON(df, dataframe = "columns", pretty = TRUE)

      api_key <- input$api_key_input
      print(api_key)
      if (is.null(api_key) || api_key == "") {
        showNotification(" Debes ingresar tu clave API de OpenRouter.", type = "error",
        ↪  duration = 10)
        return("No se envió la solicitud. Falta la clave API.")
      }

      url <- "https://openrouter.ai/api/v1/chat/completions"
      headers <- httr::add_headers(
        "Authorization" = paste("Bearer", api_key),
        "Content-Type" = "application/json",
        "HTTP-Referer" = "https://localhost",   # Cambia esto si publicas tu app
        "X-Title" = "App Shiny con OpenRouter"
      )

      mensaje <- list(
        model = "openai/gpt-3.5-turbo-0613",
        messages = list(
          list(role = "system", content = "Hola consulta lo que necesites."),
          list(role = "user", content = paste(
            "Analiza el siguiente dataset en base a la siguiente consulta:\n\n",
            input$user_question, "\n\nDatos:\n", json_data
          ))
        )
      )

      res <- tryCatch({
        httr::POST(url, headers, body = mensaje, encode = "json")
      }, error = function(e) {
        return(paste("Error al conectar con la API:", e$message))
      })

      if (inherits(res, "response")) {
        parsed <- content(res, as = "parsed", encoding = "UTF-8")
        if (!is.null(parsed$error)) {
          cat(" Error API:", parsed$error$message)
        } else {
```

```
          cat(parsed$choices[[1]]$message$content)
        }
      } else {
        print(res)
      }
    })
  })

# Análisis manual (valores únicos, estadísticas)

  output$analisis_manual <- renderPrint({
    df <- dataset()
    if (input$tipo_analisis == "media_sd") {
      apply(df[sapply(df, is.numeric)], 2, function(col) c(Media = mean(col, na.rm = TRUE),
      ↪  SD = sd(col, na.rm = TRUE)))
    } else if (input$tipo_analisis == "unicos") {
      sapply(df, function(col) length(unique(col)))
    }
  })
}

# Inicialización de la aplicación Shiny

shinyApp(ui, server)
```

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.

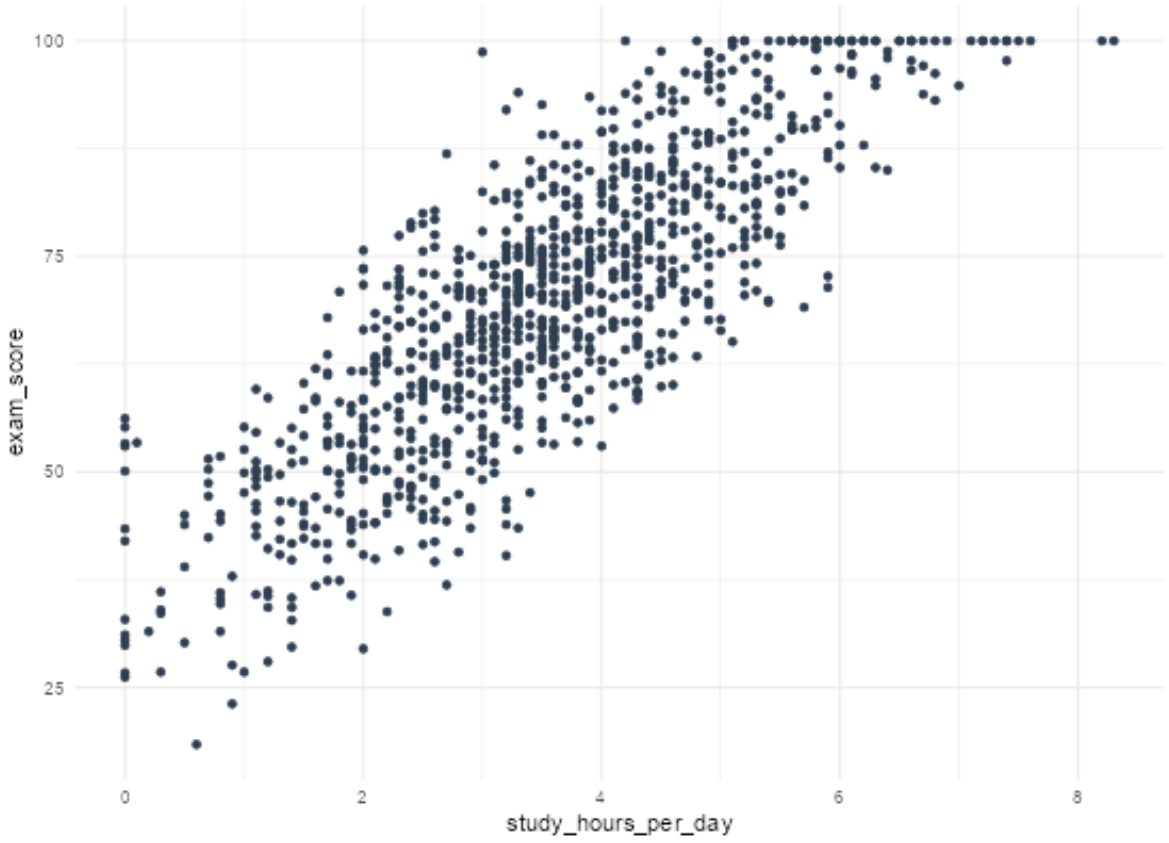
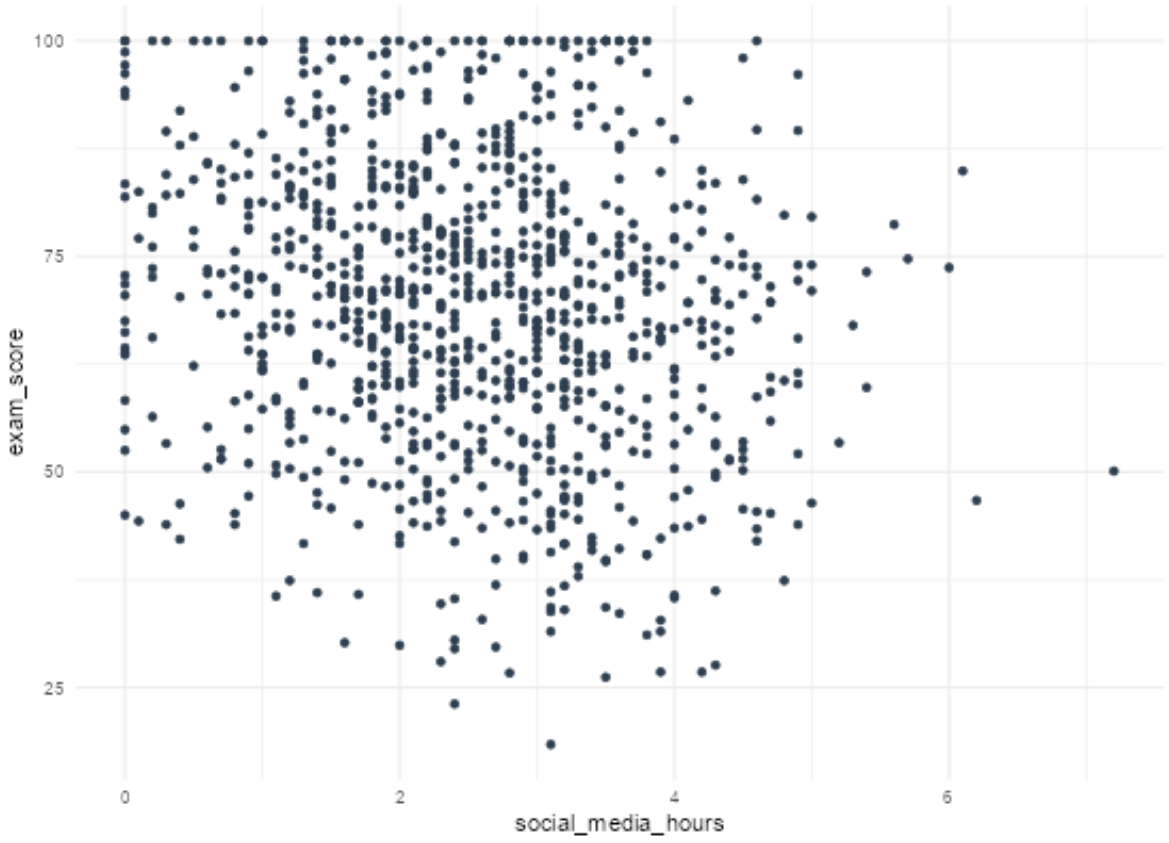


Gráfico 2

Horas de redes sociales y puntaje de examen

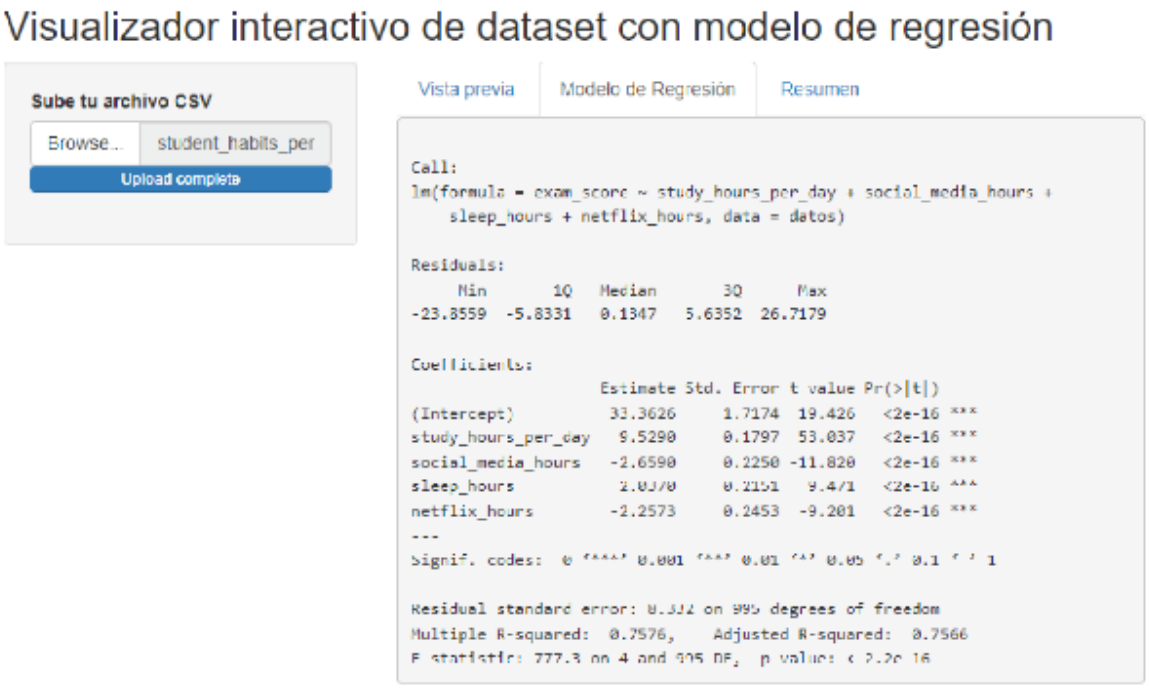
Existe una tendencia negativa débil entre las horas en redes sociales y el desempeño en el examen. Aunque no es una correlación fuerte, se sugiere que el uso excesivo puede afectar el rendimiento académico.



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




```
# 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, skip_empty_rows = TRUE)
```

```
  }, error = function(e1) {
    tryCatch({
      read_delim(file, delim = ",", locale = locale(encoding = "ISO-8859-1"), show_col_types
      ↪  = FALSE, skip_empty_rows = TRUE)
    }, error = function(e2) {
      read_delim(file, delim = ";", locale = locale(encoding = "ISO-8859-1"), show_col_types
      ↪  = FALSE, skip_empty_rows = TRUE)
    })
  })
}

# 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_hours", "netflix_hours")

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

  lm(exam_score ~ study_hours_per_day + social_media_hours + sleep_hours + netflix_hours,
   ↪  data = datos)
})

  # 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.

Variable	Coeficiente	Interpretación
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.

Pruebas con Testthat

Prueba función read_csv_auto()

prueba unitaria con `testthat`, y sirve para verificar automáticamente que la función `read_csv_auto()` funciona correctamente.

```
library(testthat)
source("app_shiny_ia_integrada.R_ACTUALIZADO.R")

# Crear archivo de prueba temporal
archivo_prueba <- tempfile(fileext = ".csv")
writeLines("col1,col2\n1,a\n2,b", archivo_prueba)

test_that("read_csv_auto carga correctamente el CSV", {
  df <- read_csv_auto(archivo_prueba)
  expect_s3_class(df, "data.frame")
  expect_equal(ncol(df), 2)
  expect_equal(nrow(df), 2)
})
```

```
Test passed
```

Prueba con archivo inexistente

Esta prueba confirma que la función responde adecuadamente cuando se intenta leer un archivo que no existe. Es útil para prevenir errores silenciosos o fallas no controladas en producción.

```
library(testthat)
source("app_shiny_ia_integrada.R_ACTUALIZADO.R")

test_that("read_csv_auto lanza error con archivo inexistente", {
  expect_error(read_csv_auto("no_existe.csv"))
})
```

```
Test passed
```

Prueba para generación de gráfico con `ggplot`

Este test verifica que una función de graficación personalizada (`grafico_puntos`) devuelve un objeto de clase `"ggplot"`, lo cual indica que la visualización está siendo generada correctamente.

```r
test_that("grafico_puntos devuelve un objeto ggplot", {
  grafico_puntos <- function(df) {
    ggplot(df, aes(x, y)) + geom_point()
  }

  df <- data.frame(x = 1:5, y = c(3, 5, 2, 8, 7))
  p <- grafico_puntos(df)

  expect_s3_class(p, "ggplot")
})
```

```
Test passed
```