LABORATORIO 2

Sentencias iterativas y personalización CSS

Sistema de control de calidad en laboratorio químico

Escuela de Informática

Información del laboratorio
Curso: EIY403 - Introducción al análisis de datos para otras carreras
Valor: 10 % de la nota final
Modalidad: Grupal (2-3 estudiantes) - Desarrollo en clase
Duración: 90 minutos
Entregables: Archivo .Rmd + archivo .html + archivo .css
Dataset: `datos_laboratorio_quimico.txt` (proporcionado)
Entrega: Correo institucional a `jordy.alfaro.brenes@una.cr`
Asunto: `Lab2_NombreEquipo_Integrante1_Integrante2`

1. Introducción y objetivos

Este laboratorio tiene dos componentes principales que evalúan competencias complementarias:

1. **Personalización visual (30 %):** Creación de un archivo CSS personalizado con ayuda de IA
2. **Sentencias iterativas (70 %):** Implementación de un sistema de control de calidad químico usando if/else, for y while

Contexto del laboratorio:
Trabajarán como analistas de control de calidad en una planta química industrial. Recibirán datos de 100 muestras analizadas durante una semana y deberán implementar un sistema automatizado que clasifique las muestras, identifique problemas y genere reportes usando sentencias iterativas en R.

2. Ejercicio 1: Personalización visual con CSS (30 puntos)

2.1. Paso 1: Selección de temática (5 minutos)

Ejercicio 1A: Definir temática del equipo

Cada equipo debe seleccionar UNA temática para su diseño CSS:

- **Científico futurista:** Colores neón, tipografías modernas
- **Laboratorio vintage:** Tonos sépia, estilo clásico
- **Minimalista industrial:** Grises, líneas limpias
- **Químico colorido:** Paleta inspirada en elementos químicos
- **Dark mode profesional:** Fondo oscuro, acentos brillantes
- **Retro científico:** Colores de los años 80, estilo retro-futurista

Anoten su selección: _____

2.2. Paso 2: Generación del CSS con IA (15 minutos)

Ejercicio 1B: Crear archivo CSS personalizado

a) Use ChatGPT, Claude o similar con este prompt base (adapte según su temática):
"Crea un archivo CSS para R Markdown con tema [SU TEMÁTICA] que incluya:

- *Paleta de colores coherente con el tema*
- *Estilos para títulos h1, h2, h3 con colores temáticos*
- *Cajas personalizadas para ejercicios y ejemplos*
- *Colores para bloques de código y resultados*
- *Mantener legibilidad académica y profesional*
- *Incluir estilos para tablas y listas*

"
b) Copie el CSS generado y guárdelo como `estilo_[nombreequipo].css`
c) Agregue esta línea al encabezado YAML de su R Markdown:

```
css: "estilo_[nombreequipo].css"
```

2.3. Paso 3: Integración y prueba (10 minutos)

Ejercicio 1C: Verificar funcionamiento del CSS

a) Compile su R Markdown y verifique que el CSS se aplique correctamente
b) Si hay errores, solicite a la IA que corrija el CSS
c) Asegúrense de que el texto siga siendo legible y profesional
Resultado esperado: Un documento R Markdown con diseño personalizado coherente con su temática seleccionada.

3. Ejercicio 2: Sistema de control de calidad químico (70 puntos)

Dataset proporcionado:
El archivo `datos_laboratorio_quimico.txt` contiene 100 muestras con las siguientes variables:

- `muestra_id`: Identificador (M001-M100)
- `ph`: Valores de pH (1.0 - 14.0)
- `concentracion_ppm`: Concentración en partes por millón (10-500)
- `temperatura_C`: Temperatura de análisis en grados Celsius (18-35)
- `operador`: Técnico que realizó la medición (Ana, Carlos, María, José)
- `turno`: Turno de trabajo (Mañana, Tarde, Noche)
- `dia`: Día de la semana (Lunes-Viernes)

3.1. Parte A: Carga y exploración de datos (10 puntos)

Ejercicio 2A: Importación y verificación inicial

Complete el siguiente código:

```
# Cargar librerias necesarias
library(knitr)
library(ggplot2)

# Cargar datos
datos <- read.csv("datos_laboratorio_quimico.txt",
                  sep = "\t", header = TRUE)

# COMPLETE: Verificar la carga de datos
str(_____)  # Ver estructura
head(_____, ___)  # Primeras 6 filas
nrow(_____)  # Confirmar 100 muestras

# COMPLETE: Resumen estadistico basico
summary(datos$_____)  # pH
summary(datos$_____)  # concentracion_ppm
summary(datos$_____)  # temperatura_C
```

3.2. Parte B: Clasificación con if/else (15 puntos)

Ejercicio 2B: Crear funciones de clasificación

Implemente las siguientes funciones usando **if/else**:

```
# Funcion para clasificar pH
clasificar_ph <- function(ph) {
  if (ph < 6.0) {
    return("Acido")
  } else if (ph >= 6.0 && ph <= 8.0) {
    return(_______)  # COMPLETE: pH neutro
  } else {
    return(_______)  # COMPLETE: pH basico
  }
}

# Funcion para evaluar concentracion
evaluar_concentracion <- function(conc) {
  if (_____ < 50) {         # COMPLETE: condicion baja
    return("Baja")
  } else if (conc >= 50 && conc <= 200) {
    return("Normal")
  } else if (_____ <= 350) { # COMPLETE: condicion alta
    return("Alta")
  } else {
    return("Critica")
  }
}

# Funcion para verificar temperatura
verificar_temperatura <- function(temp) {
  # COMPLETE: Implementar clasificacion
  # Rango seguro: 20-30 C
  # Fuera de rango: "Revisar"
  # Dentro de rango: "OK"
}
```

3.3. Parte C: Procesamiento masivo con for (20 puntos)

Ejercicio 2C: Analizar todas las muestras con bucle for

Use un bucle **for** para procesar todas las muestras:

```
# Crear vectores para almacenar resultados
muestras_aprobadas <- 0
muestras_rechazadas <- 0
problemas_encontrados <- c()

# COMPLETE: Bucle for para procesar todas las muestras
for (i in 1:nrow(_____)) {
  # Extraer datos de la muestra actual
  muestra_actual <- datos[i, ]
  ph_actual <- muestra_actual$ph
  conc_actual <- muestra_actual$concentracion_ppm
  temp_actual <- muestra_actual$temperatura_C

  # Aplicar clasificaciones
  tipo_ph <- clasificar_ph(_____)
  tipo_conc <- evaluar_concentracion(_____)
  estado_temp <- verificar_temperatura(_____)

  # COMPLETE: Logica de aprobacion/rechazo
  if (tipo_ph == "Neutro" && tipo_conc == "Normal" && estado_temp ==
      "OK") {
    __________ <- __________ + 1  # COMPLETE: incrementar aprobadas
    cat("Muestra", muestra_actual$muestra_id, ": APROBADA\n")
  } else {
    __________ <- __________ + 1  # COMPLETE: incrementar rechazadas
    problema <- paste("Muestra", muestra_actual$muestra_id,
                      "- pH:", tipo_ph, "Conc:", tipo_conc, "Temp:",
                         estado_temp)
    __________ <- c(__________, problema)  # COMPLETE: agregar
       problema
    cat("Muestra", muestra_actual$muestra_id, ": RECHAZADA\n")
  }
}

# Mostrar resumen
cat("\n=== RESUMEN DE PROCESAMIENTO ===\n")
cat("Muestras aprobadas:", muestras_aprobadas, "\n")
cat("Muestras rechazadas:", muestras_rechazadas, "\n")
```

3.4. Parte D: Análisis por operador con while (15 puntos)

Ejercicio 2D: Identificar operador con más errores

Use un bucle **while** para encontrar al operador con más muestras rechazadas:

```
# Lista de operadores unicos
operadores <- unique(datos$operador)
max_errores <- 0
operador_problema <- ""

# COMPLETE: Bucle while para revisar cada operador
i <- 1
while (i <= length(_____)) {
  operador_actual <- operadores[i]

  # Contar errores de este operador
  errores_operador <- 0

  # COMPLETE: Bucle for anidado para contar errores
  for (j in 1:nrow(datos)) {
    if (datos$operador[j] == _____ ) {  # COMPLETE: operador actual
      # Verificar si la muestra fallo algun criterio
      ph_muestra <- datos$ph[j]
      conc_muestra <- datos$concentracion_ppm[j]
      temp_muestra <- datos$temperatura_C[j]

      tipo_ph <- clasificar_ph(ph_muestra)
      tipo_conc <- evaluar_concentracion(conc_muestra)
      estado_temp <- verificar_temperatura(temp_muestra)

      # COMPLETE: Si falla algun criterio, incrementar errores
      if (tipo_ph != "Neutro" || tipo_conc != "Normal" || estado_
          temp != "OK") {
        __________ <- __________ + 1
      }
    }
  }

  cat("Operador", operador_actual, ": ", errores_operador, "errores\
      n")

  # COMPLETE: Actualizar maximo si es necesario
  if (errores_operador > _____) {
    max_errores <- __________
    operador_problema <- __________
  }

  i <- i + 1  # IMPORTANTE: incrementar contador
}

cat("\n=== OPERADOR CON MAS ERRORES ===\n")
cat("Operador:", operador_problema, "\n")
cat("Total de errores:", max_errores, "\n")
```

3.5. Parte E: Reporte final integrado (10 puntos)

Ejercicio 2E: Generar reporte ejecutivo

Complete el código para crear un reporte final:

```
# COMPLETE: Calcular estadisticas finales
porcentaje_aprobacion <- round((muestras_aprobadas / nrow(datos)) *
    100, 1)
total_muestras <- nrow(datos)

# Crear reporte
cat("\n" , rep("=", 50), "\n")
cat("          REPORTE DE CONTROL DE CALIDAD\n")
cat(rep("=", 50), "\n")
cat("Total de muestras analizadas:", _____, "\n")  # COMPLETE
cat("Muestras aprobadas:", _____, "\n")            # COMPLETE
cat("Muestras rechazadas:", _____, "\n")           # COMPLETE
cat("Porcentaje de aprobacion:", _____, "%\n")     # COMPLETE
cat("Operador con mas problemas:", _____, "\n")    # COMPLETE

# COMPLETE: Evaluar estado general del laboratorio
if (porcentaje_aprobacion >= 80) {
  cat("ESTADO GENERAL: EXCELENTE\n")
} else if (porcentaje_aprobacion >= 60) {
  cat("ESTADO GENERAL: _____\n")  # COMPLETE: Aceptable
} else {
  cat("ESTADO GENERAL: _____\n")  # COMPLETE: Critico
}

cat(rep("=", 50), "\n")

# Mostrar primeros 5 problemas encontrados
cat("PRIMEROS 5 PROBLEMAS IDENTIFICADOS:\n")
for (i in 1:min(5, length(problemas_encontrados))) {
  cat(i, ".", problemas_encontrados[i], "\n")
}
```

4. Especificaciones de entrega

Archivos a entregar:

1. Archivo principal: `laboratorio2_[nombreequipo].Rmd`
2. CSS personalizado: `estilo_[nombreequipo].css`
3. HTML compilado: `laboratorio2_[nombreequipo].html`
Configuración obligatoria del YAML:

```
---
title: "Laboratorio 2: Control de Calidad con [SU TEMATICA]"
author: "Equipo [nombre] - [Integrante1], [Integrante2], [
    Integrante3]"
date: "`r Sys.Date()`"
output:
  html_document:
    toc: true
    toc_float: true
    number_sections: true
    theme: default
    highlight: tango
    css: "estilo_[nombreequipo].css"
---
```

Correo de entrega:

- Destinatario: `jordy.alfaro.brenes@una.cr`
- Asunto: `Lab2_NombreEquipo_Integrante1_Integrante2`
- Adjuntar los 3 archivos obligatorios
- En el cuerpo del correo: lista de integrantes y temática CSS elegida

5. Criterios de evaluación

Criterio	Puntos	Descripción
CSS Personalizado	30	
- Temática coherente	10	Diseño consistente con tema elegido
- Integración técnica	15	CSS funciona correctamente
- Legibilidad profesional	5	Mantiene estándares académicos
Sentencias Iterativas	70	
- Parte A: Carga de datos	10	Importación y verificación correcta
- Parte B: Funciones if/else	15	Clasificaciones lógicas apropiadas
- Parte C: Bucle for	20	Procesamiento masivo funcional
- Parte D: Bucle while	15	Búsqueda con condiciones correcta
- Parte E: Reporte integrado	10	Sistema completo funcionando
Total	**100**	

6. Recursos de apoyo

Referencias del notebook de fundamentos:

Clasificación con if/else: Revise los ejemplos de clasificación de pH
Bucles for: Consulte el procesamiento de listas de elementos
Bucles while: Vea los ejemplos de búsqueda y conteo
Funciones combinadas: Observe la integración de múltiples estructuras

Prompts útiles para CSS:

- "Corrige este CSS para que sea más legible"
- "Agrega estilos para tablas en tema [su temática]"
- "Mejora los colores manteniendo contraste profesional"

Este laboratorio integra creatividad visual con programación aplicada, desarrollando competencias técnicas y estéticas para análisis de datos profesional en contextos químicos industriales.