Universidad Nacional del Altiplano
Facultad de Ingeniería Estadística e Informática
Docente: Fred Torres Cruz
Autor : edilberto wilson mamani emanuel

Trabajo Encargado - libreria r

Mi tareaa

```
# ============================================================================
Características principales:

Función gcl() - Implementa el generador de congruencia lineal con validación de parámetr
Análisis estadístico completo - Media, mediana, desviación estándar, etc.
Visualización avanzada - 4 tipos de gráficos usando ggplot2
Pruebas de aleatoriedad - Prueba de rachas y autocorrelación
Comparación de parámetros - Incluye ejemplos con diferentes configuraciones
Modo interactivo - Función para experimentar con tus propios parámetros

Cómo usar en RStudio:

Copia todo el código en un nuevo script de R
Ejecuta el código completo - generará automáticamente los análisis
Para experimentar con tus parámetros, ejecuta: experimentar_gcl()

Parámetros incluidos:

Original: a=5, c=3, m=16 (como tu ejemplo)
Tipo C++: Parámetros estándar de C++
Park-Miller: Generador clásico sin incremento
Personalizado: Para que pruebes otros valores

============================================================================

# Limpiar el entorno de trabajo
rm(list = ls())

# Cargar librerías necesarias
if (!require(ggplot2)) install.packages("ggplot2")
if (!require(gridExtra)) install.packages("gridExtra")
library(ggplot2)
library(gridExtra)

# ============================================================================
```

```
# FUNCIÓN PRINCIPAL DEL GENERADOR DE CONGRUENCIA LINEAL
# ============================================================================

gcl <- function(semilla, a, c, m, n) {
  # Validación de parámetros
  if (a <= 0 || c < 0 || m <= 0 || n <= 0) {
    stop("Los parámetros deben ser positivos")
  }

  if (semilla < 0 || semilla >= m) {
    stop("La semilla debe estar entre 0 y m-1")
  }

  # Vector para almacenar los números generados
  numeros <- numeric(n)

  # Valor inicial
  x_actual <- semilla

  # Generar secuencia usando la fórmula: X(n+1) = (a*X(n) + c) mod m
  for (i in 1:n) {
    x_actual <- (a * x_actual + c) %% m
    numeros[i] <- x_actual
  }

  return(numeros)
}

# ============================================================================
# FUNCIÓN PARA ANÁLISIS ESTADÍSTICO
# ============================================================================

analizar_gcl <- function(numeros) {
  cat("=== ANÁLISIS ESTADÍSTICO ===\n")
  cat("Números generados:", length(numeros), "\n")
  cat("Media:", round(mean(numeros), 4), "\n")
  cat("Mediana:", round(median(numeros), 4), "\n")
  cat("Desviación estándar:", round(sd(numeros), 4), "\n")
  cat("Varianza:", round(var(numeros), 4), "\n")
  cat("Mínimo:", min(numeros), "\n")
  cat("Máximo:", max(numeros), "\n")
  cat("Rango:", max(numeros) - min(numeros), "\n")
  cat("Valores únicos:", length(unique(numeros)), "\n")
  cat("Proporción de valores únicos:", round(length(unique(numeros))/length(numeros), 4)
}
```

```
# ============================================================================
# FUNCIÓN PARA VISUALIZACIÓN AVANZADA
# ============================================================================

visualizar_gcl <- function(numeros, titulo = "Generador de Congruencia Lineal") {
  # Crear data frame para ggplot
  df <- data.frame(
    indice = 1:length(numeros),
    valor = numeros,
    normalizado = numeros / max(numeros)
  )

  # Gráfico 1: Serie temporal
  p1 <- ggplot(df, aes(x = indice, y = valor)) +
    geom_line(color = "blue", size = 1) +
    geom_point(color = "red", size = 2) +
    labs(title = paste(titulo, "- Serie Temporal"),
         x = "Índice", y = "Valor") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))

  # Gráfico 2: Histograma
  p2 <- ggplot(df, aes(x = valor)) +
    geom_histogram(bins = 15, fill = "lightblue", color = "black", alpha = 0.7) +
    labs(title = "Distribución de Frecuencias",
         x = "Valor", y = "Frecuencia") +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))

  # Gráfico 3: Diagrama de dispersión (valores consecutivos)
  if (length(numeros) > 1) {
    df_lag <- data.frame(
      x_n = numeros[-length(numeros)],
      x_n1 = numeros[-1]
    )

    p3 <- ggplot(df_lag, aes(x = x_n, y = x_n1)) +
      geom_point(color = "darkgreen", size = 2, alpha = 0.7) +
      labs(title = "Diagrama de Dispersión (X_n vs X_n+1)",
           x = "X_n", y = "X_n+1") +
      theme_minimal() +
      theme(plot.title = element_text(hjust = 0.5))
  } else {
    p3 <- ggplot() + theme_void()
```

```
  }

  # Gráfico 4: Valores normalizados
  p4 <- ggplot(df, aes(x = indice, y = normalizado)) +
    geom_line(color = "purple", size = 1) +
    geom_point(color = "orange", size = 1.5) +
    labs(title = "Valores Normalizados [0,1]",
         x = "Índice", y = "Valor Normalizado") +
    ylim(0, 1) +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))

  # Combinar gráficos
  grid.arrange(p1, p2, p3, p4, ncol = 2)
}

# ============================================================================
# FUNCIÓN PARA PRUEBAS DE ALEATORIEDAD
# ============================================================================

pruebas_aleatoriedad <- function(numeros) {
  cat("\n=== PRUEBAS DE ALEATORIEDAD ===\n")

  # Prueba de rachas (runs test)
  if (require(randtests, quietly = TRUE)) {
    runs_result <- runs.test(numeros)
    cat("Prueba de rachas - p-valor:", round(runs_result$p.value, 4), "\n")
    cat("Interpretación:", ifelse(runs_result$p.value > 0.05, "Aleatorio", "No aleatorio
  }

  # Autocorrelación de lag-1
  if (length(numeros) > 1) {
    autocorr <- cor(numeros[-length(numeros)], numeros[-1])
    cat("Autocorrelación lag-1:", round(autocorr, 4), "\n")
    cat("Interpretación:", ifelse(abs(autocorr) < 0.1, "Baja correlación", "Alta correla
  }
}

# ============================================================================
# EJEMPLOS DE USO
# ============================================================================

cat("=== GENERADOR DE CONGRUENCIA LINEAL ===\n\n")

# Parámetros del ejemplo original
```

```
semilla <- 7
a <- 5
c <- 3
m <- 16
n <- 10

cat("Parámetros utilizados:\n")
cat("Semilla (X0):", semilla, "\n")
cat("Multiplicador (a):", a, "\n")
cat("Incremento (c):", c, "\n")
cat("Módulo (m):", m, "\n")
cat("Cantidad de números (n):", n, "\n\n")

# Generar números
numeros <- gcl(semilla, a, c, m, n)

# Mostrar resultados
cat("Números generados:\n")
print(numeros)

# Análisis estadístico
analizar_gcl(numeros)

# Pruebas de aleatoriedad
if (!require(randtests)) {
  cat("\nPara ejecutar pruebas de aleatoriedad, instale el paquete 'randtests':\n")
  cat("install.packages('randtests')\n")
} else {
  pruebas_aleatoriedad(numeros)
}

# Visualización
cat("\nGenerando gráficos...\n")
visualizar_gcl(numeros, "Ejemplo GCL (a=5, c=3, m=16)")

# ============================================================================
# EJEMPLOS ADICIONALES CON DIFERENTES PARÁMETROS
# ============================================================================

cat("\n=== COMPARACIÓN CON DIFERENTES PARÁMETROS ===\n")

# Conjunto de parámetros para comparar
parametros <- list(
  list(a = 5, c = 3, m = 16, nombre = "Original"),
  list(a = 1103515245, c = 12345, m = 2^31, nombre = "Tipo C++"),
```

```
  list(a = 16807, c = 0, m = 2^31 - 1, nombre = "Park-Miller"),
  list(a = 9, c = 1, m = 64, nombre = "Personalizado")
)

# Generar y comparar
for (i in 1:length(parametros)) {
  param <- parametros[[i]]
  cat("\n--- Generador", param$nombre, "---\n")
  cat("a =", param$a, ", c =", param$c, ", m =", param$m, "\n")

  # Generar más números para mejor análisis
  nums <- gcl(semilla, param$a, param$c, param$m, 100)

  cat("Primeros 10 números:", nums[1:10], "\n")
  cat("Valores únicos:", length(unique(nums)), "de", length(nums), "\n")
  cat("Proporción única:", round(length(unique(nums))/length(nums), 3), "\n")
}

# ============================================================================
# FUNCIÓN INTERACTIVA PARA EXPERIMENTAR
# ============================================================================

experimentar_gcl <- function() {
  cat("\n=== MODO EXPERIMENTAL ===\n")
  cat("Ingrese sus parámetros personalizados:\n")

  # Solicitar parámetros al usuario
  semilla <- as.numeric(readline("Semilla (X0): "))
  a <- as.numeric(readline("Multiplicador (a): "))
  c <- as.numeric(readline("Incremento (c): "))
  m <- as.numeric(readline("Módulo (m): "))
  n <- as.numeric(readline("Cantidad de números (n): "))

  # Generar y analizar
  tryCatch({
    numeros <- gcl(semilla, a, c, m, n)
    cat("\nNúmeros generados:\n")
    print(numeros)
    analizar_gcl(numeros)
    visualizar_gcl(numeros, paste("GCL Personalizado (a=",a,", c=",c,", m=",m,")", sep="
  }, error = function(e) {
    cat("Error:", e$message, "\n")
  })
}
```

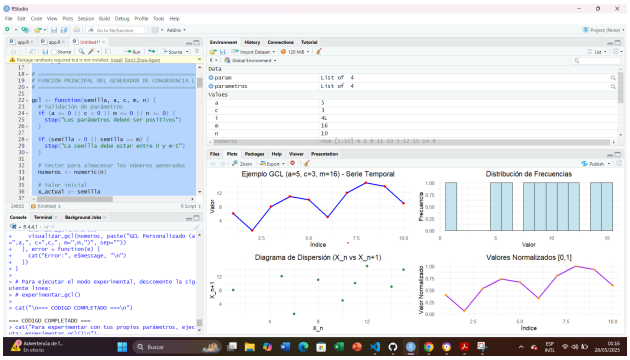

Figura 1: Enter Caption

```
# Para ejecutar el modo experimental, descomente la siguiente línea:
# experimentar_gcl()

cat("\n=== CÓDIGO COMPLETADO ===\n")
cat("Para experimentar con tus propios parámetros, ejecuta: experimentar_gcl()\n")
```