Simulación y Análisis del Juego de los Dulces (Chupetines)

Alumno: Ronald Wilder Incacutipa Muñuico
Docente: Fred Cruz Torres

Universidad Nacional del Altiplano
29 de octubre de 2025

Resumen

Presentamos un modelo estocástico que simula un proceso de intercambio de dulces entre personas. Cada persona empieza con dos dulces (tipos `a`, `b`, `c`). Mediante préstamos momentáneos, si una persona llega a tener los tres tipos $\{a, b, c\}$ canjea esos tres por un chupetín y se retira del juego; además devuelve un dulce aleatorio al prestamista. El objetivo es simular el proceso y medir cuántos chupetines se obtienen en un número dado de interacciones; además calculamos automáticamente el número de iteraciones como $\left\lceil \frac{2N}{3} \right\rceil$.

Tipo de problema

Este problema es un **modelo estocástico discreto por agente** (agent-based) con dinámica de intercambio y retirada. Tiene componentes de:

- **Aleatoriedad intrínseca**: los tipos de dulces iniciales y las decisiones de préstamo/devuelta son aleatorias.
- **Interacciones locales**: los eventos (préstamo, devolución, canje) ocurren entre pares de agentes.
- **Dinámica absorbente**: un agente que canjea se retira (estado absorbente).

Se trata de un problema de simulación Monte Carlo — no buscamos una solución analítica cerrada, sino estudiar el comportamiento por simulaciones y reglas del juego.

Descripción de la dinámica (reglas)

1. Hay N personas; cada persona inicialmente recibe 2 dulces elegidos uniformemente en $\{a, b, c\}$ (con reemplazo).
2. En cada interacción se elige un prestatario p_1 y un prestamista p_2 entre los agentes activos.
3. p_2 presta *un dulce* a p_1 (ahora temporalmente p_1 tiene 3 dulces).
4. Si p_1 tiene los tres tipos $\{a, b, c\}$, canjea esos 3 por un chupetín y **se retira** (ya no participa). Además devuelve *un dulce aleatorio* a p_2 (puede ser distinto del que recibió).
5. Si p_1 no consigue los 3 tipos, devuelve el dulce prestado y no se retira.
6. El proceso se repite hasta que no haya más interacciones factibles o hasta un número máximo de iteraciones.

Observaciones y razón práctica de "siempre queda alguien"

En la práctica, debido a la estructura discreta y a la aleatoriedad de los préstamos y devoluciones, las simulaciones muestran con frecuencia que *no todos* los agentes logran canjear por chupetín: suele quedar al menos una persona sin chupetín. Esto ocurre por combinaciones y secuencias de préstamos que agotan las posibilidades locales antes de que ciertos agentes logren completar su trío. No afirmamos aquí una prueba matemática formal de «siempre» (eso depende de reglas exactas y condiciones iniciales), pero en la dinámica que definimos resulta altamente probable que algunos queden sin chupetín para valores típicos de N. Las simulaciones ayudan a medir esa probabilidad y la distribución de resultados.

Cálculo automático del número de iteraciones

Para fijar un tope razonable de trabajo, definimos el número de iteraciones usando la fórmula:

$$\text{iteraciones} \;=\; \left\lceil \frac{2\,N}{3} \right\rceil$$

(es decir, se toma el numero de dulces dividido por 3, y se redondea hacia arriba). Este criterio se usa como límite práctico de interacciones a simular.

Algoritmo (alto nivel)

1. Inicializar N agentes con 2 dulces aleatorios cada uno.

2. Calcular max_iter $= \left\lceil \frac{2N}{3} \right\rceil$.

3. Para cada iteración hasta max_iter:

   a) Elegir aleatoriamente un prestatario p_1 entre activos.

   b) Elegir aleatoriamente un prestamista $p_2 \neq p_1$ entre activos.

   c) p_2 presta 1 dulce a p_1; si p_1 consigue $\{a, b, c\}$ entonces canjea y se retira; devuelve un dulce aleatorio a p_2.

   d) Si no consigue los 3, devuelve el dulce prestado.

4. Reportar cuántos agentes consiguieron chupetín y el estado final.

Código en R

A continuación va el código R . Este código incorpora el cálculo automático de iteraciones, la aleatoriedad descrita y salida con la lista de personas que consiguieron chupetín.

```r
# Simulación del juego de los dulces (chupetines)
# Dinámica: cada persona tiene 2 dulces; puede recibir 1 prestado;
# si consigue {a,b,c} canjea y se retira; devuelve 1 dulce aleatorio.

simular_chupetines_doc <- function(N = 20, seed = NULL) {
  if (!is.null(seed)) set.seed(seed)
  tipos <- c("a", "b", "c")

  # Cálculo automático de iteraciones según la fórmula solicitada
  max_iter <- ceiling((2 * N) / 3)

  # Inicialización: cada persona con 2 dulces aleatorios
  personas <- replicate(N, sample(tipos, 2, replace = TRUE), simplify = FALSE)
  obtuvo_chupetin <- rep(FALSE, N)   # indicador de retiro por haber obtenido chupetín
  iter <- 0

  tiene_tres <- function(x) length(unique(x)) == 3

  # Bucle principal: max_iter interacciones
  for (k in 1:max_iter) {
    iter <- iter + 1

    # elegir prestatario y prestamista entre los activos
    activos <- which(!obtuvo_chupetin)
    if (length(activos) <= 1) break  # ya no hay pares posibles

    p1 <- sample(activos, 1)                 # prestatario
    p2 <- sample(setdiff(activos, p1), 1)    # prestamista

```

```r
    # si prestamista no tiene dulces, saltar (seguridad)
    if (length(personas[[p2]]) == 0) next

    # p2 presta 1 dulce a p1
    dulce_prestado <- sample(personas[[p2]], 1)
    personas[[p1]] <- c(personas[[p1]], dulce_prestado)

    # comprobar si p1 ahora tiene los 3 tipos
    if (tiene_tres(personas[[p1]])) {
      obtuvo_chupetin[p1] <- TRUE

      # Devuelve UN dulce aleatorio al prestamista (puede cambiar el tipo)
      # Aquí modelamos la devolución como: p1 se queda vacío y p2 recibe un dulce aleatorio
      if (length(personas[[p2]]) > 0) {
        # Reemplazamos uno de los dulces de p2 por un dulce aleatorio (simula devolución alea
        idx <- sample(seq_along(personas[[p2]]), 1)
        personas[[p2]][idx] <- sample(tipos, 1)
      } else {
        # Si p2 quedó vacío, le damos un dulce aleatorio (seguridad)
        personas[[p2]] <- sample(tipos, 1)
      }

      # p1 ya se retira: lo dejamos con vector vacío (no participa más)
      personas[[p1]] <- character(0)
    } else {
      # No consiguió completar: devuelve el dulce prestado (vuelve a 2)
      # Aseguramos que personas[[p1]] tenga exactamente 2 (si uno se repitió, recortamos)
      if (length(personas[[p1]]) > 2) {
        # mantener dos elementos (aleatorio) para volver a estado "2 dulces"
        personas[[p1]] <- sample(personas[[p1]], 2)
      }
    }
  }

  # Resultados
  total_chupetines <- sum(obtuvo_chupetin)
  cat("Simulación terminada\n")
  cat("Personas (N):", N, "\n")
  cat("Iteraciones usadas (ceil(2N/3)):", max_iter, "\n")
  cat("Chupetines obtenidos:", total_chupetines, "\n\n")
  cat("Índices de personas que obtuvieron chupetín:\n")
  print(which(obtuvo_chupetin))

  # Devolver lista con información útil
  list(
    N = N,
    iteraciones = max_iter,
    total_chupetines = total_chupetines,
    quienes = which(obtuvo_chupetin),
    personas_final = personas
  )
```

```r
}

# Ejemplo de uso:
# sim <- simular_chupetines_doc(20, seed = 123)
# Para ver variabilidad, ejecutar sin semilla o con semillas distintas.
```

Ejemplo de ejecución (salida típica)

```
Simulación terminada
Personas (N): 20
Iteraciones usadas (ceil(2N/3)): 14
Chupetines obtenidos: 8

Índices de personas que obtuvieron chupetín:
[1] 2 3 5 7 8 11 14 19
```

(Esta salida varía con la semilla; la lista anterior es solo ilustrativa.)

Conclusión

Hemos definido y simulado un modelo de intercambio de dulces donde los agentes pueden momentáneamente recibir un dulce para intentar completar los tres tipos y canjearlos por un chupetín. Usando la regla pragmática $\left\lceil \frac{2N}{3} \right\rceil$ para el número de iteraciones, el simulador produce resultados distintos en cada ejecución (aleatoriedad real). En la práctica observamos que suele quedar al menos una persona sin chupetín bajo la dinámica planteada, aunque la cantidad exacta depende de la realización aleatoria y del valor de N.

Extensión: Modelo Cooperativo por Grupo

En esta segunda parte extendemos el modelo individual al caso de un **grupo cooperativo**. En lugar de actuar de manera independiente, los N integrantes del grupo juntan todos sus dulces en una «bolsa común» y los administran colectivamente. La dinámica ahora incluye dos tipos de intercambios definidos por la empresa:

- **Intercambio tipo A**: 3 dulces distintos $\{a, b, c\}$ se canjean por un chupetín.
- **Intercambio tipo B**: 6 dulces (2 de cada tipo) se canjean por 2 chupetines y 1 dulce aleatorio adicional.

Además, si no es posible hacer ningún intercambio de los tipos anteriores pero el grupo posee chupetines, estos pueden canjearse de manera inversa:

$$1 \text{ chupetín} \longrightarrow 3 \text{ dulces aleatorios.}$$

El proceso continúa hasta que el número de chupetines es al menos igual al número de integrantes del grupo (todos consiguen uno) o hasta un máximo de iteraciones definido.

Código en R para el modelo cooperativo

```r
# Modelo cooperativo del juego de los dulces
# Un grupo trabaja junto compartiendo todos sus dulces.

simular_grupo_cooperativo <- function(n_personas = 5, max_iter = 200, seed = NULL) {
  if (!is.null(seed)) set.seed(seed)
  tipos <- c("A", "B", "C")

  # Dulces iniciales: 2 por persona
  dulces <- sample(tipos, n_personas * 2, replace = TRUE)
  chupetines <- 0
  iter <- 0

  cat("Simulación cooperativa: grupo de", n_personas, "personas\n\n")

  while (iter < max_iter) {
    iter <- iter + 1
    tabla <- table(dulces)

    # Intercambio tipo B: 6 dulces (2 de cada tipo)
    if (all(c("A","B","C") %in% names(tabla)) &&
        all(tabla[c("A","B","C")] >= 2)) {
      # Eliminar 6 dulces (2 de cada tipo)
      quitar <- c(
        rep("A", 2), rep("B", 2), rep("C", 2)
      )
```

```r
      for (q in quitar) {
        dulces <- dulces[-match(q, dulces)]
      }
      chupetines <- chupetines + 2
      # Dulce extra aleatorio
      dulces <- c(dulces, sample(tipos, 1))

    # Intercambio tipo A: 3 dulces distintos
    } else if (length(unique(dulces)) >= 3) {
      dulces <- dulces[-match(unique(dulces)[1:3], dulces)]
      chupetines <- chupetines + 1
      dulces <- c(dulces, sample(tipos, 1))

    # Si no se puede intercambiar pero hay chupetines
    } else if (chupetines > 0) {
      chupetines <- chupetines - 1
      dulces <- c(dulces, sample(tipos, 3, replace = TRUE))

    } else {
      break
    }

    # Mostrar progreso cada 10 iteraciones
    if (iter %% 10 == 0 || iter == 1) {
      cat(sprintf("Iter %d -> Dulces: %d | Chupetines: %d\n",
                  iter, length(dulces), chupetines))
    }

    # Condición de éxito
    if (chupetines >= n_personas) {
      cat("\nResultado final\n")
      cat("Iteraciones:", iter, "\n")
      cat("Dulces restantes:", length(dulces), "\n")
      cat("Chupetines totales:", chupetines, "\n")
      cat("El grupo logró que todos tengan su chupetín.\n")
      return(invisible(list(iter = iter, dulces = length(dulces), chupetines = chupetines)))
    }
  }

  cat("\nResultado final\n")
  cat("Iteraciones:", iter, "\n")
  cat("Dulces restantes:", length(dulces), "\n")
  cat("Chupetines totales:", chupetines, "\n")
  cat("El grupo no logró completar todos los chupetines.\n")

  invisible(list(iter = iter, dulces = length(dulces), chupetines = chupetines))
}

# Ejemplo:
# simular_grupo_cooperativo(5)
```

Ejemplo de ejecución (salida típica)

```
Simulación cooperativa: grupo de 5 personas

Iter 1 -> Dulces: 10 | Chupetines: 0
Iter 10 -> Dulces: 8 | Chupetines: 3
Iter 20 -> Dulces: 5 | Chupetines: 5

Resultado final
Iteraciones: 21
Dulces restantes: 5
Chupetines totales: 5
El grupo logró que todos tengan su chupetín.
```

Conclusión del modelo cooperativo

El modelo cooperativo demuestra que, mediante el uso compartido de los recursos (dulces), los grupos pueden alcanzar estados en los que todos sus miembros obtienen un chupetín. La aleatoriedad de los dulces iniciales y los resultados de los intercambios hace que el número de iteraciones necesarias varíe ampliamente entre ejecuciones. Este esquema es una aproximación a un sistema con redistribución estocástica de recursos donde la cooperación incrementa la probabilidad de éxito colectivo.