Simulación del Sistema de Canje de Caramelos y Chupetines

Henry Ccoarite Dueñas

30 de octubre de 2025

Descripción del problema

Se tienen **9 personas** en un grupo, cada una comienza con **2 caramelos**, por lo que existen en total **18 dulces**. Los dulces pueden ser de tres tipos distintos:

$$A,\ B,\ C$$

El objetivo es que el grupo, mediante intercambios y reglas de canje, logre que cada integrante tenga **exactamente un chupetín**, es decir:

$$9 chupetines en total.$$

Parámetros iniciales

- Número de integrantes: $n = 9$.
- Tipos de dulces: A, B, C.
- Dulces totales iniciales: 18.
- Distribución inicial: cada persona recibe 2 dulces (pueden repetirse).
- Inicialmente, nadie tiene chupetines.

Reglas del sistema

Regla R1: Si el grupo puede formar un trío (A, B, C), se canjea por **1 chupetín**.

$$A + B + C \rightarrow$$

Regla R2: Si se pueden formar dos tríos $(2A, 2B, 2C)$, se canjean por **2 chupetines + 1 caramelo aleatorio adicional**.

$$2A + 2B + 2C \rightarrow 2 + (A/B/C)$$

Regla R3: Si una persona devuelve un chupetín, recibe **3 caramelos aleatorios**.

$$\rightarrow A + B + C$$

Condiciones de finalización:

- El proceso continúa hasta que se alcancen **9 chupetines**.
- Todos los dulces se comparten entre las 9 personas.
- Las reglas se aplican en orden de prioridad: R2 → R1 → R3.

Código en R

```
set.seed(123)
personas <- paste("P",1:9,sep="")
tipos <- c("A","B","C")
inventario <- list(c("A","B"),c("A","C"),c("B","C"),
                   c("A","B"),c("A","C"),c("B","C"),
                   c("A","B"),c("A","C"),c("B","C"))
chupetines <- rep(0,9); iter <- 0

mostrar_estado <- function(iter,inv,ch){
  cat("\n==== Iteraci n",iter,"====\n")
  for(i in 1:9){
    caramelos <- if(length(inv[[i]])>0) paste(inv[[i]],collapse=" ") else
       "Ninguno"
    cat(sprintf("%s: %-10s | Chupetines: %d\n",personas[i],caramelos,ch[i
       ]))
  }
  cat("Totales -> Caramelos:",sum(sapply(inv,length)),
      "| Chupetines:",sum(ch),"\n")
}

regla_R1 <- function(inv,ch){
  todos <- unlist(inv)
  if(all(c("A","B","C")%in%todos)){
    for(t in c("A","B","C"))
      for(i in 1:9){
        pos<-match(t,inv[[i]])
        if(!is.na(pos)){inv[[i]]<-inv[[i]][-pos];break}
      }
    i<-which.min(ch); ch[i]<-ch[i]+1
    cat("\ n   R1 aplicada -> +1 chupet n a",personas[i],"\n")
  } else cat("\  n     R1 no aplicada\n")
  list(inv=inv,ch=ch)
}

regla_R2 <- function(inv,ch){
  counts<-table(unlist(inv))
  if(all(counts>=2)){
    for(t in c("A","B","C")){
      faltan<-2
      for(i in 1:9){
        if(faltan==0)break
        pos<-match(t,inv[[i]])
        if(!is.na(pos)){inv[[i]]<-inv[[i]][-pos];faltan<-faltan-1}
      }}
    for(k in 1:2){i<-which.min(ch);ch[i]<-ch[i]+1}
    t<-sample(c("A","B","C"),1);i<-sample(1:9,1)
    inv[[i]]<-c(inv[[i]],t)
    cat("\ n   R2 aplicada -> +2 chupetines y +1",t,"a",personas[i],"\n")
  } else cat("\  n     R2 no aplicada\n")
```

[width=0.5]image3.png

Figura 1: image3.png

```
  list(inv=inv,ch=ch)
}

regla_R3 <- function(inv,ch){
  i<-which.max(ch)
  if(ch[i]>0){
    ch[i]<-ch[i]-1
    nuevos<-sample(c("A","B","C"),3,rep=TRUE)
    inv[[i]]<-c(inv[[i]],nuevos)
    cat("\ n   R3 aplicada:",personas[i],"recibi ",paste(nuevos,collapse=
       " "),"\n")
  } else cat("\  n     R3 no aplicada\n")
  list(inv=inv,ch=ch)
}

mostrar_estado(iter,inventario,chupetines)
while(sum(chupetines)<9){
  iter<-iter+1
  total<-sum(sapply(inventario,length))
  if(total>=6 && all(table(unlist(inventario))>=2))
    res<-regla_R2(inventario,chupetines)
  else if(all(c("A","B","C")%in%unlist(inventario)))
    res<-regla_R1(inventario,chupetines)
  else if(sum(chupetines)>0)
    res<-regla_R3(inventario,chupetines)
  else {cat("\ n   No hay reglas aplicables\n"); break}
  inventario<-res$inv; chupetines<-res$ch
  mostrar_estado(iter,inventario,chupetines)
}
cat("\ n   9 chupetines alcanzados en",iter,"iteraciones\n")
```

Análisis y cumplimiento

- El modelo cumple con todos los parámetros del problema original.
- Se inicia con 9 personas y 18 dulces distribuidos.
- Se aplican las tres reglas de transformación y devolución.
- El proceso termina automáticamente al alcanzar los 9 chupetines.
- Los canjes y distribuciones son aleatorios pero controlados.
- El código es determinista gracias a la semilla `set.seed(123)`.

Conclusión

El algoritmo logra la meta de distribuir los recursos siguiendo las reglas definidas, garantizando un resultado coherente y reproducible. Cada integrante termina con un chupetín, cumpliendo las condiciones del sistema de intercambio.