Práctica Nº 8
Modelo de urnas

Alumno: José Adrián García Fuentes
Profesor: Satu Elisa Schaeffer
Fecha: 20/abril/2021

1. Introducción

El modelo de la urna es uno de los elementos de mayor uso en la aplicación de probabilidades estadísticas ya que tiende a ser un concepto que permite la facilidad de comprender gráficamente las distintas formas en las cuales puede aplicarse dicho modelo, un modelo de urnas es aquel que trata de simular un contenedor con elementos para calcular la probabilidad de extraer un elemento de la jornada, alguna característica del elemento, por ejemplo, podemos tener una urna con bolas de un tamaño y bolas de otro donde se desea conocer cuál es la probabilidad de una bola de menor tamaño. En la práctica se simula un sistema en donde se abordan los fenómenos de coalescencia y fragmentación de partículas, donde las partículas se unen y se descomponen para formar cúmulos, estos fenómenos son de gran utilidad al realizar análisis en munchas áreas como en física y química. Esto puede servir en la práctica de laboratorio como para lograr predecir qué cantidad de partículas quedaran atrapadas en un filtro de cierta apertura de poro, por ejemplo, supongamos que tenemos alguna solución y deseamos filtrado una determinada partícula una de las características más relevantes de dicha partícula es su tamaño se cuenta con una red que sólo captura las partículas de tamaño que se especifique, mediante el procedimiento solicitado por el uso del modelo de urnas aplicando el principio para este modelo generaríamos un número de enteros distribuido de tal manera que se agrupen al tamaño de los cúmulos originalmente iniciado considerando estas condiciones se procederá a realizar la simulación del sistema.

2. Objetivo

- Graficar el porcentaje que se logra filtrar en cada interación.

3. Resultados

Para la simulación se toman en cuenta dos parámetros principales, que son el número de partículas $n \in \{16k, 32k, 64k, 128k\}$ y el número de cumulos $k = 1000$ [1]. La metodología empleada se realizó a través de Rstudio [2] llevando a cabo los pasos señalados en la práctica 8: modelo de urna [1], a partir del código en el repositorio de Schaeffer [3], se realizarón modificaciones, los resultados de la experimentación los podemos ver en la figura **??** donde el eje vertical nos indica el porcentaje de las partículas que se logran filtrar y en el eje horizontal la iteración.

```
library(testit) # para pruebas, recuerda instalar antes de usar

k <- 1000
```

```
vectorn <- c(16*k,32*k,64*k,128*k)

for (replica in 1:30) {
  basefiltrados <- c()
  for (n in vectorn) {

    originales <- rnorm(k)
    cumulos <- originales - min(originales) + 1
    cumulos <- round(n * cumulos / sum(cumulos))
    assert(min(cumulos) > 0)
    diferencia <- n - sum(cumulos)
    if (diferencia > 0) {
      for (i in 1:diferencia) {
        p <- sample(1:k, 1)
        cumulos[p] <- cumulos[p] + 1
      }
    } else if (diferencia < 0) {
      for (i in 1:-diferencia) {
        p <- sample(1:k, 1)
        if (cumulos[p] > 1) {
          cumulos[p] <- cumulos[p] - 1
        }
      }
    }

    assert(length(cumulos[cumulos == 0]) == 0) # que no haya vacios
    assert(sum(cumulos) == n)
    c <- median(cumulos) # tamanio critico de cumulos
    d <- sd(cumulos) / 4 # factor arbitrario para suavizar la curva


    rotura <- function(x) {
      return (1 / (1 + exp((c - x) / d)))
    }


    union <- function(x) {
      return (exp(-x / c))
    }


    romperse <- function(tam, cuantos) {
      romper <- round(rotura(tam) * cuantos) # independientes
      resultado <- rep(tam, cuantos - romper) # los demas
      if (romper > 0) {
        for (cumulo in 1:romper) { # agregar las rotas
          t <- 1
          if (tam > 2) { # sample no jala con un solo valor
            t <- sample(1:(tam-1), 1)
          }
          resultado <- c(resultado, t, tam - t)
        }
      }
      assert(sum(resultado) == tam * cuantos) # no hubo perdidas
      return(resultado)
    }


    unirse <- function(tam, cuantos) {
      unir <- round(union(tam) * cuantos) # independientes
      if (unir > 0) {
        division <- c(rep(-tam, unir), rep(tam, cuantos - unir))
```

```
66          assert(sum(abs(division)) == tam * cuantos)
67          return(division)
68        } else {
69          return(rep(tam, cuantos))
70        }
71      }
72
73
74      freq <- as.data.frame(table(cumulos))
75      names(freq) <- c("tam", "num")
76      freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
77      duracion <- 50
78      digitos <- floor(log(duracion, 10)) + 1
79      filtrado <- c()
80      for (paso in 1:duracion) {
81        assert(sum(cumulos) == n)
82        cumulos <- integer()
83        for (i in 1:dim(freq)[1]) { # fase de rotura
84          urna <- freq[i,]
85          if (urna$tam > 1) { # no tiene caso romper si no se puede
86            cumulos <- c(cumulos, romperse(urna$tam, urna$num))
87          } else {
88            cumulos <- c(cumulos, rep(1, urna$num))
89          }
90        }
91        assert(sum(cumulos) == n)
92        assert(length(cumulos[cumulos == 0]) == 0) # que no haya
      vacios
93        freq <- as.data.frame(table(cumulos)) # actualizar urnas
94        names(freq) <- c("tam", "num")
95        freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
96        assert(sum(freq$num * freq$tam) == n)
97        cumulos <- integer()
98        for (i in 1:dim(freq)[1]) { # fase de union
99          urna <- freq[i,]
100         cumulos <- c(cumulos, unirse(urna$tam, urna$num))
101       }
102       assert(sum(abs(cumulos)) == n)
103       assert(length(cumulos[cumulos == 0]) == 0) # que no haya
      vacios
104       juntarse <- -cumulos[cumulos < 0]
105       cumulos <- cumulos[cumulos > 0]
106       assert(sum(cumulos) + sum(juntarse) == n)
107       nt <- length(juntarse)
108       if (nt > 0) {
109         if (nt > 1) {
110           juntarse <- sample(juntarse)
111           for (i in 1:floor(nt / 2) ) {
112             cumulos <- c(cumulos, juntarse[2*i-1] + juntarse[2*i])
113           }
114         }
115         if (nt %% 2 == 1) {
116           cumulos <- c(cumulos, juntarse[nt])
117         }
118       }
119       assert(sum(cumulos) == n)
120       freq <- as.data.frame(table(cumulos))
121       names(freq) <- c("tam", "num")
122       freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
123       assert(sum(freq$num * freq$tam) == n)
124       tl <- paste(paso, "", sep="")
125       while (nchar(tl) < digitos) {
```

```
        tl <- paste("0", tl, sep="")
      }
      png(paste("p8_ct", tl, ".png", sep=""), width=300, height
    =300)
      tope <- 50 * ceiling(max(cumulos) / 50)
      hist(cumulos, breaks=seq(0, tope, 50),
           main=paste("Paso", paso, "con ambos fen\u{00f3}menos"),
    freq=FALSE,
           ylim=c(0, 0.05), xlab="Tama\u{00f1}o", ylab="Frecuencia
    relativa")
      graphics.off()
      h <- cumulos[cumulos > c]
      filtrado[paso] <- sum(h) / n
    }
    basefiltrados <- cbind(basefiltrados,filtrado)
  }

  colnames(basefiltrados) <- vectorn

png(paste("p8_", replica, ".png", sep=""), width=300, height=300)
 plot(basefiltrados[,1], type = "l", col= "red", ylim=c(min(
    basefiltrados), max(basefiltrados)),
       main = paste("Replica", replica), xlab = "Iteraciones", ylab
     = "Porcentaje de filtraciones")
  lines(basefiltrados[,2], type = "l", col= "blue")
  lines(basefiltrados[,3], type = "l", col= "black")
  lines(basefiltrados[,4], type = "l", col= "green")
  print(replica)
}
```

4. Reto 1

Determina si algún intervalo de iteraciones en el que el filtrado alcance un óptimo. Realiza réplicas para determinar si el momento en el cual se alcanza el máximo tiene un comportamiento sistemático. Incluye visualizaciones para justificar las conclusiones.

```
library(testit) # para pruebas, recuerda instalar antes de usar

k <- 1000
vectorn <- c(16*k,32*k,64*k,128*k)

for (replica in 1:30) {
  basefiltrados <- c()
  for (n in vectorn) {

    originales <- rnorm(k)
    cumulos <- originales - min(originales) + 1
    cumulos <- round(n * cumulos / sum(cumulos))
    assert(min(cumulos) > 0)
    diferencia <- n - sum(cumulos)
    if (diferencia > 0) {
      for (i in 1:diferencia) {
        p <- sample(1:k, 1)
        cumulos[p] <- cumulos[p] + 1
      }
    } else if (diferencia < 0) {
      for (i in 1:-diferencia) {
        p <- sample(1:k, 1)
        if (cumulos[p] > 1) {
```

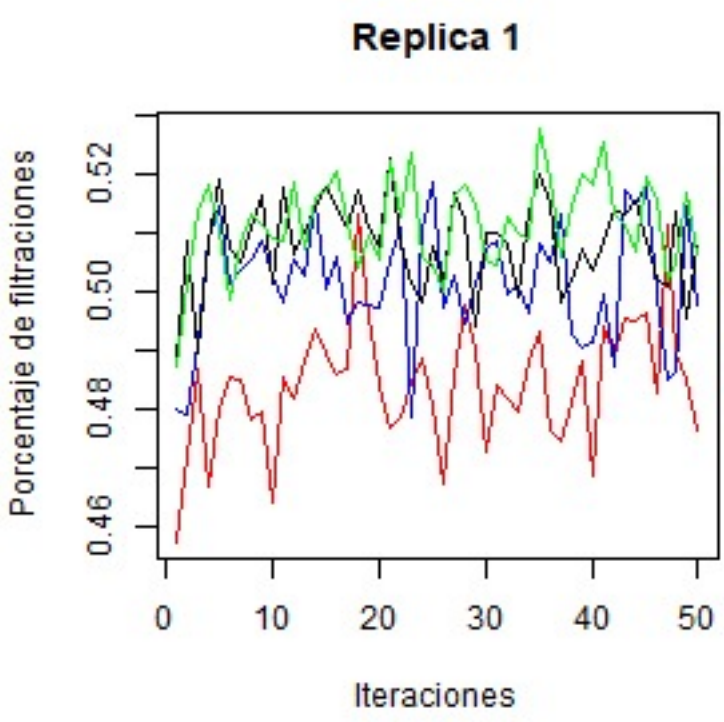

Figura 1: Grafico del porcentaje filtrado en cada interación.

```
24            cumulos[p] <- cumulos[p] - 1
25          }
26        }
27      }
28
29      assert(length(cumulos[cumulos == 0]) == 0) # que no haya vacios
30      assert(sum(cumulos) == n)
31      c <- median(cumulos) -  sd(cumulos)# tamanio critico de cumulos
32      d <- sd(cumulos) / 4 # factor arbitrario para suavizar la curva
33
34
35      rotura <- function(x) {
36        return (1 / (1 + exp((c - x) / d)))
37      }
38
39
40      union <- function(x) {
41        return (exp(-x / c))
42      }
43
44
45      romperse <- function(tam, cuantos) {
46        romper <- round(rotura(tam) * cuantos) # independientes
47        resultado <- rep(tam, cuantos - romper) # los demas
48        if (romper > 0) {
49          for (cumulo in 1:romper) { # agregar las rotas
```

```
          t <- 1
          if (tam > 2) { # sample no jala con un solo valor
            t <- sample(1:(tam-1), 1)
          }
          resultado <- c(resultado, t, tam - t)
        }
      }
      assert(sum(resultado) == tam * cuantos) # no hubo perdidas
      return(resultado)
    }


    unirse <- function(tam, cuantos) {
      unir <- round(union(tam) * cuantos) # independientes
      if (unir > 0) {
        division <- c(rep(-tam, unir), rep(tam, cuantos - unir))
        assert(sum(abs(division)) == tam * cuantos)
        return(division)
      } else {
        return(rep(tam, cuantos))
      }
    }


    freq <- as.data.frame(table(cumulos))
    names(freq) <- c("tam", "num")
    freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
    duracion <- 50
    digitos <- floor(log(duracion, 10)) + 1
    filtrado <- c()
    for (paso in 1:duracion) {
      assert(sum(cumulos) == n)
      cumulos <- integer()
      for (i in 1:dim(freq)[1]) { # fase de rotura
        urna <- freq[i,]
        if (urna$tam > 1) { # no tiene caso romper si no se puede
          cumulos <- c(cumulos, romperse(urna$tam, urna$num))
        } else {
          cumulos <- c(cumulos, rep(1, urna$num))
        }
      }
      assert(sum(cumulos) == n)
      assert(length(cumulos[cumulos == 0]) == 0) # que no haya
    vacios
      freq <- as.data.frame(table(cumulos)) # actualizar urnas
      names(freq) <- c("tam", "num")
      freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
      assert(sum(freq$num * freq$tam) == n)
      cumulos <- integer()
      for (i in 1:dim(freq)[1]) { # fase de union
        urna <- freq[i,]
        cumulos <- c(cumulos, unirse(urna$tam, urna$num))
      }
      assert(sum(abs(cumulos)) == n)
      assert(length(cumulos[cumulos == 0]) == 0) # que no haya
    vacios
      juntarse <- -cumulos[cumulos < 0]
      cumulos <- cumulos[cumulos > 0]
      assert(sum(cumulos) + sum(juntarse) == n)
      nt <- length(juntarse)
      if (nt > 0) {
        if (nt > 1) {
```

```
          juntarse <- sample(juntarse)
          for (i in 1:floor(nt / 2) ) {
            cumulos <- c(cumulos, juntarse[2*i-1] + juntarse[2*i])
          }
        }
        if (nt %% 2 == 1) {
          cumulos <- c(cumulos, juntarse[nt])
        }
      }
      assert(sum(cumulos) == n)
      freq <- as.data.frame(table(cumulos))
      names(freq) <- c("tam", "num")
      freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
      assert(sum(freq$num * freq$tam) == n)
      tl <- paste(paso, "", sep="")
      while (nchar(tl) < digitos) {
        tl <- paste("0", tl, sep="")
      }
      png(paste("p8_ct", tl, ".png", sep=""), width=300, height=300)
      tope <- 50 * ceiling(max(cumulos) / 50)
      hist(cumulos, breaks=seq(0, tope, 50),
           main=paste("Paso", paso, "con ambos fen\u{00f3}menos"), freq=FALSE,
           ylim=c(0, 0.05), xlab="Tama\u{00f1}o", ylab="Frecuencia relativa")
      graphics.off()
      h <- cumulos[cumulos > c]
      filtrado[paso] <- sum(h) / n
    }
    basefiltrados <- cbind(basefiltrados,filtrado)
  }

  colnames(basefiltrados) <- vectorn

  png(paste("p8_", replica, ".png", sep=""), width=300, height=300)
  plot(basefiltrados[,1], type = "l", col= "red", ylim=c(min(basefiltrados), max(basefiltrados)),
       main = paste("Replica", replica), xlab = "Iteraciones", ylab = "Porcentaje de filtraciones")
  lines(basefiltrados[,2], type = "l", col= "blue")
  lines(basefiltrados[,3], type = "l", col= "black")
  lines(basefiltrados[,4], type = "l", col= "green")
  print(replica)
}
```

5. Reto 2

Determina cómo los resultados de la tarea y del primer reto dependen del valor de C. ¿Qué todo cambia y cómo si C ya no se asigna como la mediana inicial sino a un valor menor o mayor?

```
library(testit) # para pruebas, recuerda instalar antes de usar

k <- 1000
vectorn <- c(16*k,32*k,64*k,128*k)

for (replica in 1:30) {
  basefiltrados <- c()
  for (n in vectorn) {
```

```
9
10    originales <- rnorm(k)
11    cumulos <- originales - min(originales) + 1
12    cumulos <- round(n * cumulos / sum(cumulos))
13    assert(min(cumulos) > 0)
14    diferencia <- n - sum(cumulos)
15    if (diferencia > 0) {
16      for (i in 1:diferencia) {
17        p <- sample(1:k, 1)
18        cumulos[p] <- cumulos[p] + 1
19      }
20    } else if (diferencia < 0) {
21      for (i in 1:-diferencia) {
22        p <- sample(1:k, 1)
23        if (cumulos[p] > 1) {
24          cumulos[p] <- cumulos[p] - 1
25        }
26      }
27    }
28
29    assert(length(cumulos[cumulos == 0]) == 0) # que no haya vacios
30    assert(sum(cumulos) == n)
31    c <- median(cumulos) +  sd(cumulos)# tamanio critico de cumulos
32    d <- sd(cumulos) / 4 # factor arbitrario para suavizar la curva
33
34
35    rotura <- function(x) {
36      return (1 / (1 + exp((c - x) / d)))
37    }
38
39
40    union <- function(x) {
41      return (exp(-x / c))
42    }
43
44
45    romperse <- function(tam, cuantos) {
46      romper <- round(rotura(tam) * cuantos) # independientes
47      resultado <- rep(tam, cuantos - romper) # los demas
48      if (romper > 0) {
49        for (cumulo in 1:romper) { # agregar las rotas
50          t <- 1
51          if (tam > 2) { # sample no jala con un solo valor
52            t <- sample(1:(tam-1), 1)
53          }
54          resultado <- c(resultado, t, tam - t)
55        }
56      }
57      assert(sum(resultado) == tam * cuantos) # no hubo perdidas
58      return(resultado)
59    }
60
61
62    unirse <- function(tam, cuantos) {
63      unir <- round(union(tam) * cuantos) # independientes
64      if (unir > 0) {
65        division <- c(rep(-tam, unir), rep(tam, cuantos - unir))
66        assert(sum(abs(division)) == tam * cuantos)
67        return(division)
68      } else {
69        return(rep(tam, cuantos))
70      }
```

```
71      }
72
73
74      freq <- as.data.frame(table(cumulos))
75      names(freq) <- c("tam", "num")
76      freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
77      duracion <- 50
78      digitos <- floor(log(duracion, 10)) + 1
79      filtrado <- c()
80      for (paso in 1:duracion) {
81        assert(sum(cumulos) == n)
82        cumulos <- integer()
83        for (i in 1:dim(freq)[1]) { # fase de rotura
84          urna <- freq[i,]
85          if (urna$tam > 1) { # no tiene caso romper si no se puede
86            cumulos <- c(cumulos, romperse(urna$tam, urna$num))
87          } else {
88            cumulos <- c(cumulos, rep(1, urna$num))
89          }
90        }
91        assert(sum(cumulos) == n)
92        assert(length(cumulos[cumulos == 0]) == 0) # que no haya
      vacios
93        freq <- as.data.frame(table(cumulos)) # actualizar urnas
94        names(freq) <- c("tam", "num")
95        freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
96        assert(sum(freq$num * freq$tam) == n)
97        cumulos <- integer()
98        for (i in 1:dim(freq)[1]) { # fase de union
99          urna <- freq[i,]
100         cumulos <- c(cumulos, unirse(urna$tam, urna$num))
101       }
102       assert(sum(abs(cumulos)) == n)
103       assert(length(cumulos[cumulos == 0]) == 0) # que no haya
      vacios
104       juntarse <- -cumulos[cumulos < 0]
105       cumulos <- cumulos[cumulos > 0]
106       assert(sum(cumulos) + sum(juntarse) == n)
107       nt <- length(juntarse)
108       if (nt > 0) {
109         if (nt > 1) {
110           juntarse <- sample(juntarse)
111           for (i in 1:floor(nt / 2) ) {
112             cumulos <- c(cumulos, juntarse[2*i-1] + juntarse[2*i])
113           }
114         }
115         if (nt %% 2 == 1) {
116           cumulos <- c(cumulos, juntarse[nt])
117         }
118       }
119       assert(sum(cumulos) == n)
120       freq <- as.data.frame(table(cumulos))
121       names(freq) <- c("tam", "num")
122       freq$tam <- as.numeric(levels(freq$tam))[freq$tam]
123       assert(sum(freq$num * freq$tam) == n)
124       tl <- paste(paso, "", sep="")
125       while (nchar(tl) < digitos) {
126         tl <- paste("0", tl, sep="")
127       }
128       png(paste("p8_ct", tl, ".png", sep=""), width=300, height
      =300)
129       tope <- 50 * ceiling(max(cumulos) / 50)
```

```
130       hist(cumulos, breaks=seq(0, tope, 50),
131            main=paste("Paso", paso, "con ambos fen\u{00f3}menos"),
      freq=FALSE,
132            ylim=c(0, 0.05), xlab="Tama\u{00f1}o", ylab="Frecuencia
      relativa")
133       graphics.off()
134       h <- cumulos[cumulos > c]
135       filtrado[paso] <- sum(h) / n
136     }
137     basefiltrados <- cbind(basefiltrados,filtrado)
138   }
139
140   colnames(basefiltrados) <- vectorn
141
142   png(paste("p8_", replica, ".png", sep=""), width=300, height=300)
143   plot(basefiltrados[,1], type = "l", col= "red", ylim=c(min(
        basefiltrados), max(basefiltrados)),
144        main = paste("Replica", replica), xlab = "Iteraciones", ylab
       = "Porcentaje de filtraciones")
145   lines(basefiltrados[,2], type = "l", col= "blue")
146   lines(basefiltrados[,3], type = "l", col= "black")
147   lines(basefiltrados[,4], type = "l", col= "green")
148   print(replica)
149 }
```

6. Conclusión

Como se observa en la figura **??** el valor más alto de la curva se encuentra en los cúmulos principales y a medida que avanza la interación se observa que el filtrado avanza por tanto los cúmulos más grandes se filtran con mayor facilidad pero tardar en formarse que los cúmulos pequeños.

Referencias

[1] E. Schaeffer, "Práctica 8: modelo de urnas," abril 2021. https://elisa.dyndns-web.com/teaching/comp/par/p8.html.

[2] J. J. Allaire, "Rstudio," abril 2021. https://rstudio.com.

[3] E. Schaeffer, "Práctica 8: modelo de urnas," ABRIL 2021. https://github.com/fuentesadrian/Simulation/tree/master/UrnModel.