```
# S es la cantidad de dinero que está en juego
# k es la cantidad de dinero inicial del jugador A
# p es la probabilidad de que el jugador A gane en cada paso
# Esta función simula el juego hasta que el jugador A
# se quede con todo el dinero, o quede en bancarrota.
GamblersRuin <- function(S, k, p, graf=FALSE) {
  if(graf) G <- c(k)
  dinero <- k
  while(dinero != 0 && dinero != S){
    r <- sample(2, 1, prob=c(1 - p, p)) # Valor aleatorio entre 1 y 2.
    # print(dinero) Para ver un proceso en particular, descomentar esta linea
    if(r > 1){
      dinero <- dinero + 1
    }
    else {
      dinero <- dinero - 1
    }
    if(graf) G <- append(G, dinero)
  }
  if(graf){
    pdf("gamblersruin.pdf")
    plot(G, type='l', xlab="Tiempo", ylab="Dinero")
    dev.off()
  }
  return(dinero == S)
}

# rep es la cantidad de repeticiones que se quieren simular
# S es la cantidad de dinero que está en juego
# k es la cantidad de dinero inicial del jugador A
# p es la probabilidad de que A gane en cada paso
# Esta funcion simplemente simula GamblersRuin 'rep' veces, y
# obtiene información empírica basada en sus resultados.
PruebaEmpirica <- function(rep, S, k, p) {
  acum <- 0
  for (i in 1:rep) {
    s <- GamblersRuin(S, k, p)
    if(s) acum <- acum + 1
  }
  return(acum / rep)
}

# Algunas pruebas interesantes

# Si le pasamos el parámetro 'graf' en TRUE, la función nos va a devolver un
# lindo gráfico de la cantidad de dinero del jugador A en función del tiempo.
# El gráfico queda guardado en 'gamblersruin.pdf', en la carpeta actual.
GamblersRuin(50, 25, 0.49, graf=TRUE)

# En este, empezamos con 7 dólares, y hay 10 en juego, nuestra probabilidad de g
anar
# cada ronda es de 0.42. A pesar de la clara ventaja inicial, las probabilidades
# son más grandes de quedar en la ruina que de ganar.
PruebaEmpirica(200, 4, 2, 0.42)
# Resultado teórico: 0.3541127

# En este, empezamos con 4 dólares, y solo necesitamos uno más para ganar (hay 5
 en
# juego), pero nuestra chance de ganar cada ronda es de 0.3. Aún a pesar de la g
ran
```

```
# ventaja, es más probable quedar en la ruina que ganar.
PruebaEmpirica(300, 5, 4, 0.3)
# Resultado teórico: 0.4201884

# Este es particularmente divertido, empezamos con 50 dólares, y necesitamos lle
gar a 100,
# y la probabilidad de ganar en cada partida es de 0.495, es decir, el juego est
á muy cerca
# de ser justo.
# Advertencia: Esta simulación tarda mucho.
PruebaEmpirica(300, 100, 50, 0.495)
# Resultado teórico: 0.2689349

# Resulta fascinante que estas simulaciones, puestas como juegos en un casino,
# podrían resultar exageramente atractivas, a pesar de la tan baja probabilidad
# de ser 'ganados'.
```

```r
expBernoulli <- function(p) {
  return(sample(0:1, 1, prob=c(1 - p, p)))
}

procesoBernoulli <- function(n, p, acum=FALSE) {
  result <- replicate(n, 0)
  for (i in 1:n) {
    result[i] <- expBernoulli(p)
  }
  if(acum) result <- cumsum(result)
  return(result)
}

pruebaEmpirica <- function(rep, n, p, acum=FALSE) {
  datos <- matrix(nrow=rep, ncol=n)
  for (i in 1:rep) {
    datos[i,] <- procesoBernoulli(n, p, acum)
  }
  result <- list(esperanza=replicate(n, 0), variancia=replicate(n, 0))
  for (i in 1:n) {
    # Procesar fila por fila y obtener espranza y variancia
    result$esperanza[i] = mean(datos[,i])
    result$variancia[i] = var(datos[,i])
  }
  return(result)
}

# 500 simulaciones de los procesos, con 10 repeticiones
# del experimento, y con 0.3 probabilidad de éxito.
print("Esperanza y Variancia de E_n")
pruebaEmpirica(500, 10, 0.3, acum=FALSE)
print("Esperanza y Variancia de S_n")
pruebaEmpirica(500, 10, 0.3, acum=TRUE)

# Otras pruebas realizadas jugando con los parámetros.
# Todos los valores resultaron muy similares, salvo por
# las últimas pruebas, dado su bajo número de repeticiones.
pruebaEmpirica(5000, 10, 0.3)
pruebaEmpirica(500, 100, 0.3)
pruebaEmpirica(10, 100, 0.3)
pruebaEmpirica(10, 3, 0.3)

# En el proceso S_n, que es el número de éxitos en el proceso Bernoulli,
# tenemos que la esperanza(S_n) = np, y que variancia(S_n) = npq.
```

```
zigZag <- function(n, p, acum=FALSE, graf=FALSE, plotSuffix='') {
  if(graf) G <- c(0)
  pos <- 0
  for (i in 1:n) {
    val <- sample(c(-1, 1), 1, prob=c(1 - p, p))
    pos <- pos + val
    if(graf){
      if(acum) G <- append(G, pos)
      else     G <- append(G, val)
    }
  }
  if(graf){
    pdf(paste("zigZag", plotSuffix, ".pdf", sep=''))
    if(acum){
      plot(G, type='l', xlab="Tiempo", ylab="Posición")
    }
    else {
      plot(G, type='b', xlab="Tiempo", ylab="Incremento")
    }
    dev.off()
  }
}

# Primero simulamos D_n, con pocos puntos para obtener un plot bonito
zigZag(10, 0.51, graf=TRUE, plotSuffix='Normal')

# Luego simulamos la posición de la partícula, con muchos puntos para notar
# qué tan lejos puede irse del estado inicial (0), y también, para tener
# un plot bonito.
zigZag(500, 0.51, graf=TRUE, acum=TRUE, plotSuffix='Acum')
```

```
library("markovchain")

# Retorna TRUE/FALSE dependiendo de si el vector es un
# vector de probabilidad válido de 4 elementos.
validarFila <- function(v) {
  if(length(v) != 4) return(FALSE)
  for (i in 1:length(v)) {
    if(v[i] < 0 || v[i] > 1) return(FALSE)
  }
  if(sum(v) != 1) return(FALSE)
  return(TRUE)
}

# Realiza la simulación con los parámetros especificados,
# y retorna la cantidad de veces que visitó cada estado como
# una tabla.
simulacionMarkov <- function(tMatrix, initial, rep) {
  # Check initial (ejercicio a)
  if(!validarFila(initial)) return(FALSE)

  # Check transitionMatrix (ejericio b)
  filas <- nrow(tMatrix)
  if(filas != 4) return(FALSE)
  for (i in 1:filas) {
    if(!validarFila(tMatrix[i,])) return(FALSE)
  }

  # Hacer la simulación (ejercicio c)
  estados <- c("A", "B", "C", "D")
  MC <- new("markovchain", states=estados, transitionMatrix=tMatrix)
  init <- sample(estados, 1, prob=initial)

  if(length(absorbingStates(MC)) > 0) return(FALSE) # No debe tener estados abso
rbentes
  lista <- rmarkovchain(n=rep, object=MC, t0=init)

  listaPlot <- replicate(length(lista), 0)
  for (i in 1:length(lista)) {
    listaPlot[i] <- match(lista[i], estados)
  }

  # Gráfico (enunciado)
  pdf("markovchain.pdf")
  plot(listaPlot, yaxt="n", type='b')
  axis(2, at=1:4, labels=c("A", "B", "C", "D"))
  dev.off();

  return(table(lista))
}

simulacionMarkov(matrix(c( 0 , .25, .5 , .25,
                           .75,  0 , .12, .13,
                           .1 , .1 ,  0 , .8,
                           .2 , .2 , .6 ,  0), byrow=TRUE, nrow=4), c(1/8, 2/8, 1
/8, 4/8), 20)
```