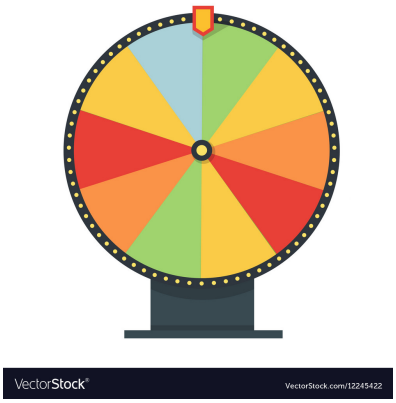
EST-24107: Simulación

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022 — Números aleatorios.
Objetivo: Que veremos.
Lectura recomendada: Capítulo 3 de [2]. Capítulo 2 de [1].

source Rubinstein, RobertR, Ross.

1. INTRODUCCIÓN

Es posible, que cuando pensamos en generar números aleatorios, idealizamos con lanzar una moneda, un dado, una baraja o una rueda giratoria estilo *Jeopardy!*.



En nuestra computadora, los `números pseudo-aleatorios` son secuencias generadas de manera determinista de tal forma que *parecen* ser variables uniformes independientes. Es decir, parecen ser

$$x_i \overset{\text{iid}}{\sim} \mathsf{U}(0,1)\,. \tag{1}$$

El procedimiento mas común es utilizar una semilla x_0 y calcular recursivamente valores x_n con $n \geq 1$ por medio de

$$x_n = ax_{n-1} \mod m\,, \tag{2}$$

donde a y m son enteros positivos.

Nota que x_n es un valor entre $0, 1, \ldots, m-1$. Llamamos a la cantidad x_n/m un número pseudo-aleatorio. Esto nos da un valor en el intervalo $(0, 1)$.

Las constantes a y m se escogen de tal forma que:

1. Para cualquier punto inicial, la secuencia *parece* ser un secuencia de números aleatorios uniformes.
2. Para cualquier punto inicial, el tiempo estimado para ver una repetición es muy largo.
3. Se puede calcular la secuencia eficientemente.

La constante m está asociada al periodo de la secuencia. Por ejemplo, podemos utilizar

$$x_n = 3x_{n-1} \quad \text{mód } 5\,, \tag{3}$$

para generar la secuencia a partir de $x_0 = 3$,

```
x0 ← 3; a ← 3; m ← 5;
x ← x0;
for (jj in 2:10){
  x[jj] ← (a * x[jj-1]) %% m
}
x
```

```
[1] 3 4 2 1 3 4 2 1 3 4
```

Si cambiamos los valores podemos conseguir un periodo mas largo y por lo tanto un mayor colección de números aleatorios.

```
x0 ← 3; a ← 2; m ← 11;
x ← x0;
for (jj in 2:20){
  x[jj] ← (a * x[jj-1]) %% m
}
x
```

```
[1]  3  6  1  2  4  8  5 10  9  7  3  6  1  2  4  8  5 10  9  7
```

Usualmente m se escoge como un `número primo` de longitud igual al máximo número representable en una computadora.

> Por ejemplo, en una máquina de 32-bits se ha visto que $m = 2^{31} - 1$ y $a = 7^5 = 16,807$ funcionan bien.

Los lenguajes de programación tienen funciones para generar números aleatorios. Por ejemplo, en `Matlab` el enfoque es cómputo numérico por lo tanto el generador de aleatorios uniformes es la opción estándar.

```
rand
```

```
0.4616637206465595
```

El lenguaje de `python` es multi-propósito. Por lo tanto, no es una opción *natural* y se llaman módulos especializados para generar números aleatorios. El módulo para generar números aleatorios tiene cierto nivel de compatibilidad con otros lenguajes.

```
import numpy as np
np.random.random()
```

```
0.9820617713830841
```

Por último, `R` es un lenguaje que se originó en la comunidad estadística. Por lo tanto, se la generación de números aleatorios requiere de la distribución de interés.

```
runif(1)
```

```
[1] 0.2955
```

2. CONCLUSIONES

REFERENCIAS

[1] C. Robert and G. Casella. *Introducing Monte Carlo Methods with R*. Springer New York, New York, NY, 2010. 1

[2] S. M. Ross. *Simulation*. Academic Press, Amsterdam, 5th edition edition, 2013. 1