Cálculo de promedios o sumas

Es análogo a Monte Carlo.

Ejemplo: se quiere calcular $P = \frac{1}{N} \sum_{j=1}^{N} a_j$

Tenemos $g : \{1, \dots, N\} \longrightarrow \mathbb{R}$ donde $g(j) = a_j$

$$P = \sum_{j=1}^{N} g(j) \cdot \frac{1}{N} = \sum_{j=1}^{N} g(j) \, P(X=j)$$

donde $X \sim \mathcal{U}\{1, 2, \dots, N\}$ (uniforme discreta)

Luego $P = E[g(X)]$

Usando la Ley Fuerte de los grandes números, podemos estimar $P = E[g(X)]$ generando una muestra:

$$X_1, X_2, \ldots \; X_M, \quad (\text{con } M < N)$$

aplicamos g: $g(X_1), g(X_2), \ldots, g(X_M)$

y promediamos: $\frac{1}{M}\left(g(X_1) + g(X_2) + \cdots + g(X_M)\right)$.

Si queremos estimar una suma:

$$S = \sum_{j=1}^{50000} e^{\cos(j)}$$

$$S = 50.000 \cdot \boxed{\frac{1}{50000} \sum_{j=1}^{50000} e^{\cos(j)}}$$

$\hookrightarrow$ Monte Carlo

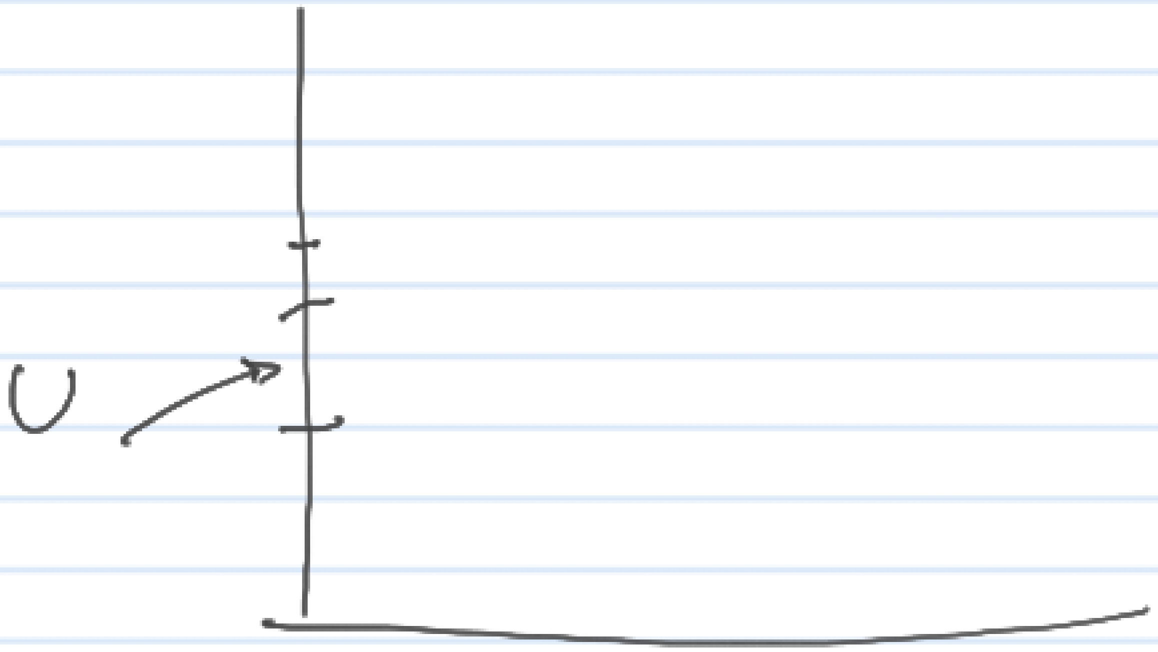
Podríamos elegir una muestra de tamaño $M = 1000$ para estimar el promedio.

$$S \sim 50000 \cdot \left(\frac{1}{1000} \sum_{j=1}^{1000} e^{\cos(X_j)} \right)$$

$X_1, \dots, X_{1000}$ es una muestra de $X \sim \mathcal{U}\{1 \dots, 50000\}$

Generación de $X \sim Geom(p)$; $0<p<1$

$X \in \{1, 2, 3, \dots, \}$ $\qquad P(X=j) = (1-p)^{j-1} \cdot p$



Método de la T. I.

```
def GeomX(p):
    prob = p
    j = 1
    F = p
    U = random()
    while U ≥ F:
        prob *= (1-p)
        F += prob , j += 1
    return j
```

El algoritmo devuelve $X=j$ si

$$F(j-1) \leq U < F(j)$$; esto es equivalente

$$1-F(j) < 1-U \leq 1-F(j-1)$$

$$1-F(j) = P(X>j) = (1-p)^j$$

$$(1-p)^j < 1-U \leq (1-p)^{j-1}$$

$$j \log(1-p) < \log(1-U) \leq (j-1)\log(1-p)$$

$$\boxed{\log(1-p)<0}$$

$$\boxed{j-1 \leq \frac{\log(1-U)}{\log(1-p)} < j}$$ $\longrightarrow$ en este caso se devuelve j.

El algoritmo devuelve j si

$$j = \underbrace{\left\lfloor \frac{\log(1-U)}{\log(1-p)} \right\rfloor + 1}_{\text{esto es lo que devuelve}}$$

```
def geomX(p):
    U = random()
    return int(log(1-U) / log(1-p)) + 1
```

$X=10$ $\longrightarrow$ es lo mismo que generar 10 Bernoulli con resultado 0 0 0 0 0 0 0 0 0 1

Variable Bernoulli

$X \sim B(p)$

$P(X=1) = p; \quad P(X=0) = (1-p)$

```
def BernoulliX(p):
    U = random()
    if U < p:
        return 1
    return 0
```

Otra posibilidad es generar una secuencia de $Y \sim Geom(p)$

Ejemplo: $Y_1 = 2 \quad Y_2 = 4 \quad Y_3 = 1$; es equivalente a

$X_1 = 0 \quad X_2 = 1 \quad X_3 = X_4 = X_5 = 0, \; X_6 = 1, \quad X_7 = 1$

0 1 0 0 0 1 1

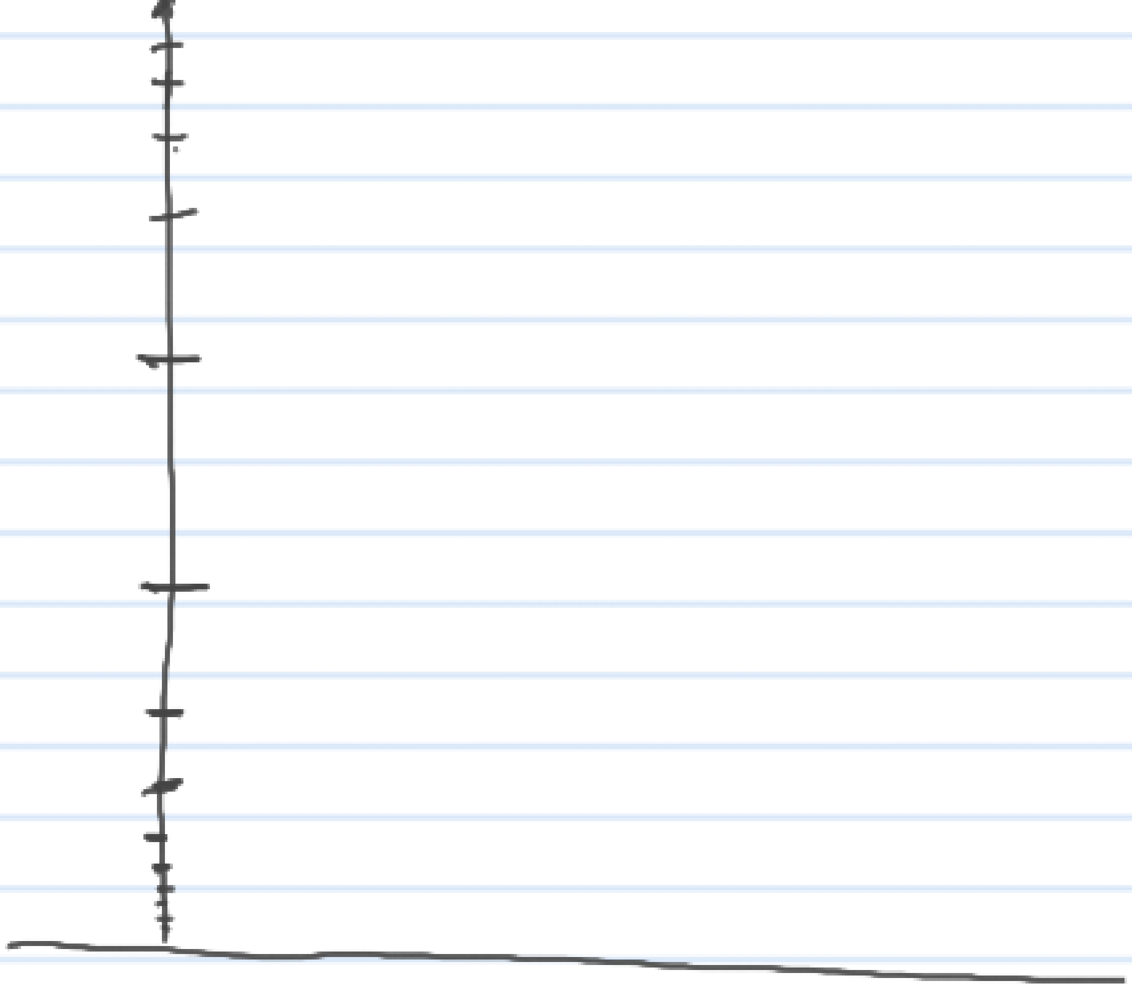
Variable $X \sim \mathcal{P}(\lambda)$ (Poisson con parámetro λ)

$X \in \{0, 1, 2, \dots \}$; $P(X=j) = e^{-\lambda} \cdot \dfrac{\lambda^j}{j!}$

Método de transformada inversa

```
def PoissonX ( L ):
    prob = exp(-L)
    F    = prob
    j    = 0
    U    = random( )
    while U ≥ F:
        prob *= L/(j+1)
        F += prob
        j += 1
    return j
```

("L = λ")



El valor más probable es $\lfloor \lambda \rfloor$.

```
def Poisson(lamda):
    p = exp(-lamda);     F = p
    for j in range(1, int(lamda) + 1):
        p *= lamda / j
        F += p
    U = random()
    if U >= F:
        j = int(lamda) + 1
        while U >= F:
            p *= lamda / j; F += p
            j += 1
        return j - 1
    else:
```

```
        j = int(lamda)
        while U < F:
            F -= p;    p *= j/lamda
            j -= 1
        return j+1
```

¿En qué mejora?

En el primer algoritmo, el número de comparaciones es 1+ valor generado.

$$E[\text{número de comparaciones}] = 1 + E[X] = 1 + \lambda$$

En el segundo algoritmo, el número de comparaciones es

$$1 + \text{distancia}(\text{valor generado}, \lambda) = 1 + |X-\lambda|$$

Si $X > 10$, podemos aproximar $X \sim N(\lambda, \sqrt{\lambda})$

Entonces $\dfrac{X-\lambda}{\sqrt{\lambda}} \sim N(0,1)$

$$\text{Luego } E[\text{número de comparaciones}] \sim 1 + \overbrace{E\left[\left|\frac{X-\lambda}{\sqrt{\lambda}}\right|\right]}^{E[|Z|];\ Z\sim N(0,1)} \cdot \sqrt{\lambda}$$

$$\mathbb{E}[\underbrace{|Z|}_{g(Z)}] = \int_{-\infty}^{\infty} \underbrace{|x|}_{g(x)} \cdot \underbrace{\frac{e^{-x^2/2}}{\sqrt{2\pi}}}_{f(x)} \, dx = \frac{2}{\sqrt{2\pi}} \underbrace{\int_0^{\infty} x\, e^{-x^2/2} \, dx}_{\int_0^{\infty} e^{-u} du = 1}$$

$u = x^2/2 \, ; \quad du = 2x \cdot \frac{1}{2} \, dx = x\, dx$

$$\mathbb{E}[|Z|] = \sqrt{\frac{2}{\pi}} \cong 0{,}798$$

$$\mathbb{E}[\text{nº de comparaciones}] \sim 1 + 0{,}798 \times \sqrt{\lambda}$$

Variable Binomial. $X \sim Bin(n,p)$

$$X \in \{0,1,2,\dots,n\} \; ; \quad P(X=j) = \binom{n}{j} p^j (1-p)^{n-j}$$

Fórmula recursiva para $P(X=j)$ es:

$$P(X=0) = (1-p)^n$$

$$P(X=j+1) = \binom{n}{j+1} p^{j+1} (1-p)^{n-(j+1)}$$

$$\binom{n}{j+1} = \frac{n!}{(j+1)!\,(n-(j+1))!} = \frac{n!}{j!\,(n-j)!} \cdot \frac{(n-j)}{(j+1)} = \frac{n-j}{j+1} \binom{n}{j}$$

$$P(X=j+1) = \frac{n-j}{j+1} \cdot \frac{p}{(1-p)} \, P(X=j)$$

$\longrightarrow$ no depende de j

```
def BinomialX(n, p):
        c = p/(1-p)
        prob = (1-p)**n
        F = prob
        j = 0
        U = random()
        while U ≥ F:
            prob *= c*(n-j)/(j+1)
            F += prob
            j += 1
        return j
```

Este algoritmo se puede mejorar preguntando primero por las probabilidades más altas:

$$E[\text{comparaciones}] = 1 + \underbrace{E\left[\left|\frac{X - np}{\sqrt{np(1-p)}}\right|\right]}_{\sim 0{,}798} \sqrt{np(1-p)}$$

<u>Otra forma</u>: $E[\text{Número de comparaciones}] = 1 + n \cdot p$ prob de éxito

Si $X \sim Bin(n, p) \Rightarrow Y = n - X \sim Bin(n, 1-p)$

Si generamos Y;

```
def BinY(n, q)
    ⋮
    return n - Y
```

número de comp $\sim 1 + n(1-p)$