Modelos y simulación - Prácticos

FAMAF - UNC

Severino Di Giovanni

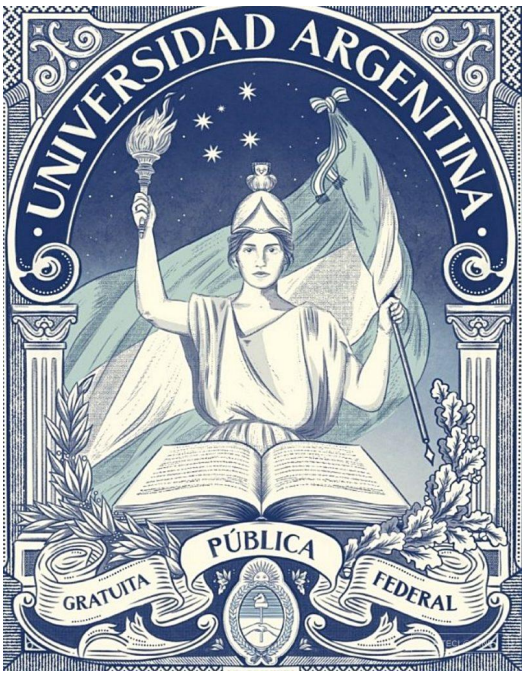

Figure 1: Severino Di Giovanni, el autor de este apunte. Un anarquista libertario, murió luchando por la libertad. Como él, otros miles han muerto para que nosotros gocemos de los derechos que tenemos. No te dejes engañar por los tristes pregoneros del egoísmo. Amá a tu prójimo y no olvides que si sus derechos se vulneran, los tuyos también. Ayudá a tu compañero de estudio, defendé tu universidad.

Contents

1 P1

(1) Un geólogo recoge 10 especímenes de A y 12 de B. 15 son elegidos al azar. De la función de probabilidad de masa del número de especímenes de tipo A elegidos.

Solución. Sea Ω el espacio muestral, con cada $\omega \in \Omega$ una de las posibles selecciones de 15 de entre los 22 especímenes. Claramente,

$$|\Omega| = \binom{22}{15}$$

Sea X la variable aleatoria para el número de especímenes de tipo A en una selección aleatoria de Ω. Claramente, para tener $X = k$, con $3 \leq k \leq 10$, la selección debe ser tal que k muestras sean de tipo A y $15 - k$ muestras sean de tipo B. Luego

$$P(X = k) = \frac{\binom{10}{k}\binom{12}{15-k}}{\binom{22}{15}}$$

(2) Sean X, Y independientes y exponenciales con parámetros λ, μ respectivamente. Computar $f_{X|Y}(x \mid y)$ y $P(X < Y)$.

(a) Recordemos que $f_{X|Y}(x \mid y) = \frac{f_{X,Y}(x,y)}{f_Y(y)}$. Como X, Y son independientes, su densidad conjunta es el producto de las densidades, y tenemos $f_{X|Y}(x \mid y) = (f_X(x) f_Y(y))/f_Y(y) = f_X(x)$.

(b) Veamos que

$$\begin{aligned} P(X < Y) &= \int_{\mathbb{R}} f_Y(y) \int_{-\infty}^{y} f_X(x)\, dx\, dy \\ &= \int_{\mathbb{R}} \mathcal{I}_{(0,\infty)} \cdot \mu e^{-\mu y} \int_{-\infty}^{y} \mathcal{I}_{(0,\infty)} \lambda e^{-\lambda x}\, dx\, dy \\ &= \int_0^{\infty} \mu e^{-\mu y} \int_0^{y} \lambda e^{-\lambda\, x} \end{aligned}$$

Ahora bien, con $w := -\lambda x$, $dw = -\lambda dx$, tenemos

$$\begin{aligned} \int_0^{y} \lambda e^{-\lambda x}\, dx &= -\int_0^{w(y)} e^{w}\, dw \\ &= -\int_0^{-\lambda y} e^{w}\, dw \\ &= -\left[e^{-\lambda y} - e^{0}\right] \\ &= 1 - e^{-\lambda y} \end{aligned}$$

Por lo tanto

$$
\begin{aligned}
P(X < Y) &= \int_0^\infty \mu e^{-\mu y} \left(1 - e^{-\lambda\ y}\right)\ dy \\
&= \int_0^\infty \mu e^{-\mu y}\ dy - \int_0^\infty \mu e^{-(\mu+\lambda)y}\ dy \\
&= \int_0^\infty f_Y(y)\ dy - \frac{\mu}{\mu+\lambda} \int_0^\infty (\mu+\lambda) e^{-(\mu+\lambda)y}\ dy \\
&= 1 - \frac{\mu}{\mu+\lambda} \int_0^\infty f_Z(y)\ dy \qquad \{Z \sim \mathcal{E}(\mu+\lambda)\} \\
&= 1 - \frac{\mu}{\mu+\lambda} \\
&= \frac{\mu+\lambda}{\mu+\lambda} - \frac{\mu}{\mu+\lambda} \\
&= \frac{\lambda}{\mu+\lambda}
\end{aligned}
$$

(3) Sean X, Y independientes y exponenciales con el mismo parámetro λ. Computar la densidad condicional de X dado que $X + Y = t$.

Solución. Por def. de densidad condicional, tomando $T := X + Y$,

$$f_{X|T}(x \mid t) = \frac{f_{X,T}(x,t)}{f_T(t)}$$

Calculemos estas distribuciones.

$$\begin{aligned}
f_{X,T}(x,t) &= P(X = x \cap X + Y = t) \\
&= P(X = x \cap Y = t - x) \\
&= P(X = x) \cdot P(Y = t - x) && \{X, Y \text{ son independientes}\} \\
&= f_X(x) \cdot f_Y(t - x) \\
&= \lambda e^{-\lambda x} \mathcal{I}_{(0,\infty)}(x) \cdot \lambda e^{-\lambda(t-x)}\, \mathcal{I}_{(0,\infty)}(t - x) \\
&= \lambda^2 e^{-\lambda t}\, \mathcal{I}_{(0,\infty)}(t - x)\mathcal{I}_{(0,\infty)}(x) \\
&= \lambda^2 e^{-\lambda t}\, \mathcal{I}_{(x,\infty)}(t)\mathcal{I}_{(0,\infty)}(x)
\end{aligned}$$

El último paso se justifica porque $\mathcal{I}_{(0,\infty)}(t - x)\mathcal{I}_{(0,\infty)}(x) = \mathcal{I}_{(0,\infty)}(x)\mathcal{I}_{(x,\infty)}(t)$. En efecto, $t - x$ pertenece a $(0, \infty)$ si y solo si $t > x$. Ahora bien,

$$\begin{aligned}
f_T(t) &= f_{X+Y}(t) \\
&= \int_{\infty}^{t} f_X(x) f_Y(t - x)\, dx && \{X, Y \text{ independientes}\} \\
&= \int_{-\infty}^{t} \mathcal{I}_{(0,\infty)} \cdot \lambda e^{-\lambda x} \cdot \mathcal{I}_{(0,\infty)}(t - x) \lambda e^{-\lambda(t-x)}\, dx \\
&= \int_{0}^{t} \lambda^2 e^{-\lambda t} \cdot \mathcal{I}_{(x,\infty)}(t)\, dx \\
&= \lambda^2 e^{-\lambda t} \int_{0}^{t} \mathcal{I}_{(x,\infty)}(t)\, dx \\
&= \lambda^2 e^{-\lambda t} \int_{0}^{t} 1\, dx \qquad (\star) \\
&= t\lambda^2 e^{-\lambda t}
\end{aligned}$$

El paso a $(\star)$ se justifica como sigue. Si la región de integración es $(0, t)$ y la variable de integración es x, siempre se cumple que $x < t$, y por lo tanto $\mathcal{I}_{(x,\infty)}(t) = 1$.

$$\therefore \quad f_{X|T}(x \mid t) = \frac{f_{X,T}(x,t)}{f_T(t)} = \frac{\lambda^2 e^{-\lambda t}\mathcal{I}_{(0,\infty)(x)}\mathcal{I}_{(x,\infty)}(t)}{t\lambda^2 e^{-\lambda t}} = \begin{cases} \frac{1}{t} & 0 \le x \le t \\ 0 & c.c. \end{cases}$$

(4) Sean $N_1(t), N_2(t)$ procesos con $t \geq 0$ y tasas $\lambda = 1, \mu = 2$ respectivamente. Sea $N(t) = N_1(t) + N_2(t)$.

(a) Calcular la probabilidad de que $N(1) = 2, N(2) = 5$.

(b) Calcular la probabilidad de que $N_1(1) = 1$ dado $N(1) = 2$.

(c) Calcular la probabilidad de que el segundo arribo en N_1 ocurra antes que el tercer arribo en $N_2(t)$.

(a) Es muy fácil porque $N(t)$ es un proceso de Poisson con parámetro $\lambda+\mu = 3$, con lo cual $P(N(t) = k) = e^{-3}\frac{3^k}{k!}$ para cualquier $k \geq 0$ entero.

(b) Claramente, si $N(1) = 2$, la probabilidad de que $N_1(1) = 1$ es la probabilidad de que $N_2(1) = 1$.

$$\begin{aligned} P(N_1(1) = 1 \mid N(1) = 2) &= \frac{P(N_1(1) = 1 \cap N(1) = 2)}{P(N(1) = 2)} \\ &= \frac{P(N_1(1) = 1 \cap N_1(1) + N_2(1) = 2)}{P(N(1) = 2)} \\ &= \frac{P(N_1(1) = 1 \cap N_2(1) = 1)}{P(N(1) = 2)} \\ &= \frac{P(N_1(1) = 1) \cdot P(N_2(1) = 1)}{P(N(1) = 2)} \qquad \{\text{Independencia de } N_1, N_2\} \end{aligned}$$

Ahora solo resta ver que

$$e^{-\lambda\, l}\frac{(\lambda l)^k}{k!} \leq e^{-\mu\, l}\frac{(\mu l)^k}{k!} \iff e^{-\lambda l}\lambda^k \leq e^{-\lambda\mu}\mu^k \iff \lambda \ln \lambda \geq \mu \ln \mu$$

$$\begin{aligned} P(N_1(1) = 1) &= e^{-1}\frac{1^1}{1!} = e^{-1} \approx 0.368 \\ P(N_2(1) = 1) &= e^{-2}\frac{2^1}{1!} = 2e^{-2} \approx 0.27 \\ P(N(1) = 2) &= e^{-3}\frac{3^2}{2!} = \frac{9}{2}e^{-3} \approx 0.224 \end{aligned}$$

$$\therefore P(N_1(1) = 1 \mid N(1) = 2) \approx \frac{0.368 \cdot 0.27}{0.224} = 0.443$$

Otra forma de calcularlo es usando el teorema de Bayes:

$$
\begin{aligned}
P(N_1(1) = 1 \mid N(1) = 2) &= \frac{P(N(1) = 1 \mid N_1(1) = 1)P(N_1(1) = 1)}{P(N(1) = 1)} \\
&= \frac{P(N_1(1) + N_2(1) = 1 \mid N_1(1) = 1)P(N_1(1) = 1)}{P(N(1) = 1)} \\
&= \frac{P(N_2(1) = 1) \cdot P(N_1(1) = 1)}{P(N(1) = 2)}
\end{aligned}
$$

que es lo mismo que tuvimos antes.

(c) Se nos pide la probabilidad de que el segundo arribo en $N_1(t)$ ocurra antes que el tercer arribo en $N_2(t)$. A primera vista no parece, pero este es simplemente un problema de conteo.

Primero, observemos que el evento "el segundo arribo en $N_1(t)$ ocurre antes que el tercero de $N_2(t)$" es equivalente al evento "de 4 eventos en $N(t)$, al menos dos provienen de $N_1(t)$". Sea ω este evento.

La probabilidad de que un evento dado de $N(t)$ provenga de $N_1(t)$ es $\frac{\lambda}{\lambda+\mu}$. La probabilidad de que un evento dado de $N(t)$ provenga de $N_2(t)$ es $\frac{\mu}{\lambda+\mu}$.

$$
\therefore \; P(\omega) = \binom{4}{2}\frac{\lambda^2\mu^2}{(\lambda+\mu)^4} + \binom{4}{3}\frac{\lambda^3\mu}{(\lambda+\mu)^4} + \binom{4}{4}\frac{\lambda^4}{(\lambda+\mu)^4}
$$

Sustituyendo λ, μ por sus valores y calculando los binomiales, que son fáciles, sale el resultado.

(5) Sean X, Y independientes con distribuciones de Poisson parametrizadas bajo λ, μ respectivamente. Probar que $Z = X + Y \sim \mathcal{P}(\lambda + \mu)$.

Como son independientes, la distribución de su suma es dada por la "convolución" discreta de sus distribuciones. Si con $\mathbb{N}$ denotamos los naturales y el cero, entonces

$$\begin{aligned}
f_Z(z) &= \sum_{k\in\mathbb{N}} p_X(k)p_Y(z-k) \\
&= \sum_{k\in\mathbb{N}} \left(e^{-\lambda}\frac{\lambda^k}{k!}\right)\left(e^{-\mu}\frac{\mu^{z-k}}{(z-k)!}\right) \\
&= e^{-\lambda}e^{-\mu}\sum_{k\in\mathbb{N}} \frac{\lambda^k\mu^{z-k}}{k!(z-k)!} \\
&= \frac{\eta}{z!}\sum_{k\in\mathbb{N}} \lambda^k\mu^{z-k}\frac{z!}{k!(z-k)!} \qquad \{\eta := e^{-\lambda}e^{-\mu}\} \\
&= \frac{\eta}{z!}\sum_{k\in\mathbb{N}} \lambda^k\mu^{z-k}\binom{z}{k}
\end{aligned}$$

Ahora bien, como $Y \sim \mathcal{P}(\mu)$, $Im(Y) = \mathbb{N}$, y por lo tanto $p_Y(u) = 0$ si $u < 0$. En otras palabras, dado un k arbitrario, $p_Y(z-k) \neq 0 \iff z - k \geq 0 \iff k \leq z$.

$$\therefore \frac{\eta}{z!}\sum_{k\in\mathbb{N}} \lambda^k\mu^{z-k}\binom{z}{k} = \frac{\eta}{z}\sum_{k=0}^{z} \lambda^k\mu^{z-k}\binom{z}{k} = \frac{\eta}{z!}(\lambda+\mu)^z = e^{-(\lambda+\mu)}\frac{(\lambda+\mu)^z}{z!}$$

por el teorema del binomio de Newton.
$\therefore Z \sim \mathcal{P}(\lambda + \mu)$. ■

2 P3

(2) (Todas las variables en este problema son uniformes continuas e independientes en $(0, 1)$). En un juego, se simula la variable aleatoria U. Si $U < \frac{1}{2}$ se suman dos números aleatorios. Si $U \geq \frac{1}{2}$, se suman tres. El resultado X de cualquiera de estas sumas es el puntaje, y se tiene éxito si $X \geq 1$.

(a) Dar la probabilidad de ganar y (b) escribir una simulación para estimar la probabilidad de ganar.

(a) Por ley de probabilidad total,

$$\begin{aligned}
&P(X_1 + X_2 \geq 1 \mid U < \frac{1}{2})P(U < \frac{1}{2}) + P(X_1 + X_2 + X_3 \geq 1 \mid U \geq \frac{1}{2})P(U \geq \frac{1}{2}) \\
=&P(X_1 + X_2 \geq 1)P(U < \frac{1}{2}) + P(X_1 + X_2 + X_3 \geq 1)P(U \geq \frac{1}{2}) \\
=&P(X_1 + X_2 \geq 1) \cdot \frac{1}{2} + P(X_1 + X_2 + X_3 \geq 1) \cdot \frac{1}{2}
\end{aligned}$$

Estudiemos $P(X_1 + X_2 = z)$. Veamos que si $z \in [0, 1]$, entonces

$$P(X_1 + X_2 = z) = \int_0^z f_{X_1}(x) f_{X_2}(z - x)\, dx = \int_0^z 1\, dx = z$$

Si $z \in (1, 2]$, entonces hacemos variar X_1 entre $[z - 1, 1]$ y requerimos $X_2 = z - X_1$:

$$P(X_1 + X_2 = z) = \int_{z-1}^1 f_{X_1}(x) f_{X_2}(z - x)\, dx = 2 - z$$

$$\therefore\ f_{X_1+X_2}(x) = \begin{cases} x & 0 \leq x \leq 1 \\ 2 - x & 1 < x \leq 2 \\ 0 & c.c. \end{cases}$$

Por lo tanto,

$$P(X_1 + X_2 \geq 1) = \int_1^2 f_{X_1+X_2}(x)\, dx = \int_1^2 (2 - x)\, dx = 1/2$$

Resta ver que

$$
\begin{aligned}
P(X_1 + X_2 + X_3 \geq 1) &= P(X_1 + X_2 \geq 1 - X_3) \\
&= \int_0^1 P(X_1 + X_2 \geq 1 - z) P(X_3 = z)\, dz \quad \{\text{Por prob. total}\} \\
&= \int_0^1 \left(1 - \frac{(1-z)^2}{2}\right) \cdot 1 \, dz \qquad \{\text{Por } \star\} \\
&= 5/6
\end{aligned}
$$

dado que para $z \in [0, 1]$

$$
(\star) \qquad P(X_1 + X_2 \geq 1 - z) = 1 - P(X_1 + X_1 \leq 1 - z) = 1 - \int_0^{1-z} x \, dx = 1 - \frac{(1-z)^2}{2}
$$

$$
\begin{aligned}
\therefore \quad & P(X_1 + X_2 \geq 1) \cdot \frac{1}{2} + P(X_1 + X_2 + X_3 \geq 1) \cdot \frac{1}{2} \\
=& \frac{1}{2} \cdot \frac{1}{2} + \frac{5}{6} \cdot \frac{1}{2} \\
=& \frac{1}{4} + \frac{5}{12} \\
=& \frac{2}{3}
\end{aligned}
$$

(b) En Python,

```
[python]
from random import uniform

def sim_2(n):

  wins = 0

  # S = summation of uniform rvs
  def S(k):
    return sum( [uniform(0, 1) for _ in range(k)] )

  for _ in range(n):

    u = uniform(0, 1)
```

```
  X = S(2) if u < 0.5 else S(3)
  wins = wins + 1 if X >= 1 else wins

return wins/n
```

(3) Las máquinas tragamonedas usualmente generan un premio cuando hay un acierto. Supongamos que se genera el acierto con el siguiente esquema: se genera un número aleatorio, y

- si es menor a un tercio, se suman dos nuevos números aleatorios
- si es mayor o igual a un tercio, se suman tres números aleatorios .

Si el resultado de la suma es menor o igual a 2, se genera un acierto.

a) ¿Cuál es la probabilidad de acertar? (*b*) Simular.

Sea $Z_n = \sum_{i=1}^{n} U_i$ con $U_i \sim \mathcal{U}(0,1)$ para $1 \leq i \leq n$. Por ley de probabilidad total, se llega con facilidad a

$$P(\text{Ganar}) = P(Z_2 \leq 2) \cdot \frac{1}{3} + P(Z_3 \leq 2) \cdot \frac{2}{3}$$

Claramente, $P(Z_2 \leq 2) = 1$, con lo cual solo queda computar $P(Z_3 \leq 2) = P(Z_2 \leq 2 - U_3)$. La densidad de Z_2 fue calculada en el punto **(2)**. Para calcular $P(Z_3 \leq 2)$, podemos hacer variar a U_3 en todo el rango $[0,1]$ y a Z_2 en todo el rango $[0, 2 - U_3]$:

$$\begin{aligned}
P(Z_3 \leq Z_2) &= P(Z_2 \leq 2 - U_3) \\
&= \int_0^1 P(U_3 = z) \int_0^{2-z} f_{Z_2}(x)\, dx\, dz \\
&= \int_0^1 \left[\int_0^1 f_{Z_2}(x)\, dx + \int_1^{2-z} f_{Z_2}(x)\, dx \right] dz \\
&= \int_0^1 \left[\int_0^1 x\, dx + \int_1^{2-x} (2-x)\, dx \right] dz \\
&= \int_0^1 \left[\frac{1}{2} - \frac{z^2}{2} + \frac{1}{2} \right] dz \\
&= \int_0^1 1 - \frac{z^2}{2}\, dz \\
&= 5/6
\end{aligned}$$

Una forma alternativa de calcular esta probabilidad es hacer variar a Z_2 en todo el rango $[0,2]$ y luego hacer variar a U_3 en todo el rango $[0, 2 - Z_2]$:

$$
\begin{aligned}
P(Z_3 \leq 2) &= P(Z_2 \leq 2 - U_3) \\
&= \int_0^2 f_{Z_2}(x) \int_0^{2-x} f_{U_3}(y)\, dy\, dx \\
&= \int_0^1 f_{Z_2}(x) \int_0^1 f_{U_3}(y)\, dy\, dx + \int_1^2 f_{Z_2}(x) \int_0^{2-x} f_{U_3}(y)\, dy\, dx \quad \{\text{Por } \star\} \\
&= \int_0^1 x \int_0^1 1\, dy\, dx + \int_1^2 (2-x) \int_0^{2-x} 1\, dy\, dx \\
&= \int_0^1 x\, dx + \int_1^2 (2-x)^2\, dx \\
&= \frac{1}{2} + \frac{1}{3} \\
&= \frac{5}{6}
\end{aligned}
$$

($\star$) Si Z_2 varía en $[0, 1]$, $Z \leq 2 - U_3$ para cualquier valor de U_3, con lo cual hacemos a U_3 variar en todo el rango $[0, 1]$. Pero si Z_2 varía en $[1, 2]$, entonces U_3 solo puede tomar valores en $[0, 2 - Z_2]$.

$$
\begin{aligned}
\therefore \qquad P(\text{Ganar}) &= P(Z_2 \leq 2) \cdot \frac{1}{3} + P(Z_3 \leq 2) \cdot \frac{2}{3} \\
&= \frac{1}{3} + \frac{5}{6}\frac{2}{3} \\
&= 0.\overline{8}
\end{aligned}
$$

(b)En Python, una implementación posible es:

```
from random import uniform

def sim_3(n):
  wins = 0

  # S = summation of uniform r.vs
  def S(k):
    return sum( [uniform(0, 1) for _ in range(k)] )
```

```
for _ in range(n):
  u = uniform(0, 1)
  X = S(2) if u < 1/3 else S(3)
  wins = wins + 1 if X <= 2 else wins

return wins/n
```

(**4**) Un supermercado posee 3 cajas. Por una cuestión de ubicación, el 40% de los clientes eligen la caja 1 para pagar, el 32% la caja 2, y el 28% la caja 3.

El tiempo que espera una persona para ser atendida en cada caja distribuye exponencial con medias de 3, 4 y 5 minutos respectivamente.

(a) ¿Cuál es la probabilidad de que un cliente espere menos de 4 minutos para ser atendido?

(b) Si el cliente tuvo que esperar más de 4 minutos. ¿Cuál es la probabilidad de que el cliente haya elegido cada una de las cajas?

(c) Simule el problema y estime las probabilidades anteriores con 1000 iteraciones.

Recordemos que la media de la exponencial es $1/\lambda$, es decir que $\lambda_1 = 1/3, \lambda_2 = 1/4, \lambda_3 = 1/5$.

El evento "un cliente espera menos de 4 minutos en ser atendido" se divide en un sub-eventos correspondiente por cada caja. Más formalmente, si T es el tiempo de espera de un cliente arbitrario, y C_i el evento de que el cliente elige la i-ésima caja, entonces por ley de probabilidad total:

$$P(T \leq 4) = \sum_{i=1}^{3} P(C_i)P(T \leq 4 \mid C_i) = \sum_{i=1}^{3} P(C_i)P(X_i \leq 4)$$

donde X_i es una exponencial con parámetro λ_i. En general,

$$\int_0^4 \lambda_i e^{-\lambda_i x}\, dx = 1 - e^{-4\lambda_i}$$

de lo cual se sigue:

$$P(X_i \leq 4) = \begin{cases} 0.736 & i = 1 \\ 0.632 & i = 2 \\ 0.55 & i = 3 \end{cases}$$

$$\therefore P(T \leq 4) = 0.40 \cdot 0.736 + 0.32 \cdot 0.632 + 0.28 \cdot 0.55 = 0.65064$$

(b) Sale fácil con teorema de Bayes y los resultados de (a):

$$P(C_i \mid T \leq 4) = \frac{P(T \leq 4 \mid C_i)P(C_i)}{P(T \leq 4)}$$

(c) En Python,

```
from numpy.random import exponential
from random import uniform

def sim_4(n):

  wins = 0
  rates = [1/3, 1/4, 1/5]

  def choose_rate(x):
    if x <= 0.4:
      return 0
    return 1 if u <= 0.4 + 0.32 else 2

  for _ in range(n):

    u = uniform(0, 1)
    rate = rates[choose_rate(u)]

    e = exponential(1/rate)

    wins = wins + 1 if e <= 4 else wins

  return wins/n
```

(5) Calcule con el método de Monte Carlo y escriba simulaciones para cada una de las siguientes integrales.

(a) $\int_0^1 (1-x^2)^{3/2}\,dx$

Sea $f_U(x)$ la densidad de $U \sim \mathcal{U}(0,1)$. Entonces

$$\theta := \int_0^1 (1-x^2)^{3/2}\,dx = \int_0^1 (1-x^2)^{3/2} f_U(x)\,dx = \mathbb{E}\left[g(U)\right]$$

con $g(x) = (1-x^2)^{3/2}$. Por lo tanto, $\theta \approx \frac{1}{n}\sum_{i=1}^n g(u_i)$ con $u_1, \ldots, u_n$ realizaciones de $U_1, \ldots, U_n \sim \mathcal{U}(0,1)$. Es decir,

$$\theta \approx \frac{1}{n}\sum^n (1-u_i^2)^{3/2}$$

En Python,

```
from random import uniform
def sim_5a(n):
  uniforms = [uniform(0, 1) for _ in range(n)]
  return 1/n * sum( [ (1 - u**2)**1.5  for u in uniforms ] )
```

$(b)\ \int_2^3 \frac{x}{x^2-1}\,dx$

Si hacemos $u = x - 2$, tal que $du = dx$ y $x = 2 + u$, tenemos

$$\theta := \int_2^3 \frac{x}{x^2-1}\,dx = \int_0^1 \frac{2+u}{(2+u)^2-1}\,du = \int_0^1 \frac{2+u}{u^2-4u+3}$$

Entonces $\theta = \int_0^1 g(x) f_U(x)\,dx = \mathbb{E}\left[g(U)\right]$ con $U \sim \mathcal{U}(0,1), g(x) = \frac{2+x}{(2+x)^2-1}$.

$$\therefore\ \theta \approx \frac{1}{n}\sum_{i=1}^{n} \frac{2+u_i}{u_i^2 - 4u_i + 3}$$

donde $u_1, \ldots, u_n$ son realizaciones de $U_1, \ldots, U_n \sim \mathcal{U}(0,1)$ independientes para $n \geq 1$. En Python,

```
def sim_5b(n):
  uniforms = [uniform(0, 1) for _ in range(n)]
  return 1/n * sum( [ (2 + u)/((2+u)**2 - 1) for u in uniforms ] )
```

Con $n = 1000$ obtuve $\theta \approx 0.487$, y el valor real de la integral es $\theta = 0.49$.

(c) $\int_0^\infty x(1+x^2)^{-2}\,dx$

Sea $u := \frac{1}{x+1}$, de manera que

$$du = -\frac{1}{(x+1)^2}\,dx = -u^2\,dx, \qquad x = \frac{1-u}{u} = \frac{1}{u} - 1$$

Entonces

$$\theta := \int_0^\infty g(x)\,dx = -\int_0^1 \frac{g(\frac{1}{u}-1)}{u^2}\,du = -\int_0^1 \frac{(\frac{1}{u}-1)(1+(\frac{1}{u}-1)^2)^{-2}}{u^2}\,du$$

Luego

$$\theta = -\int_0^1 \psi(u) f_U(u)\,du = -\mathbb{E}\left[\psi(U)\right]$$

con $\psi(u) = g(\frac{1}{u}-1)/u^2$ y $U \sim \mathcal{U}(0,1)$. En consecuencia,

$$\theta \approx \frac{1}{n}\sum_{i=1}^{n} \psi(u_i)$$

donde $u_1, \ldots, u_n$ son realizaciones de $U_1, \ldots, U_n \sim \mathcal{U}(0,1)$ independientes (no confundir con u, la variable de integración) y $n \geq 1$. En Python,

```
def sim_5c(n):
  uniforms = [uniform(0, 1) for _ in range(n)]

  def h(u):
    return (1/u - 1)*(1 + (1/u - 1)**2)**(-2) * (1/u**2)

  return 1/n * sum( [h(u) for u in uniforms] )
```

Con $n = 1000$, la simulación dio $\varphi \approx 0.504$, y el valor real de la integral es $\theta = \frac{1}{2}$.

$(d)\ \int_{-\infty}^{\infty} e^{-x^2}\,dx$

Let $g(x) := e^{-x^2}$. Clearly, $g(-x) = e^{-(-x^2)} = g(x)$, which means g is symmetric.

$\therefore\ \int_{-\infty}^{\infty} e^{-x^2}\,dx = 2\int_0^{\infty} e^{-x^2}\,dx$. If we let

$$\psi = \frac{1}{x+1}, \qquad d\psi = -\psi^2, \qquad x = \frac{1}{\psi} - 1$$

Entonces

$$\theta := 2\int_0^{\infty} e^{-x^2} = 2\int_0^1 \frac{\exp\left(-\left(\left(\frac{1}{\psi}-1\right)^2\right)\right)}{\psi^2} f_U(\psi)\,d\psi$$

donde f_U es la densidad de $U \sim \mathcal{U}(0,1)$. $\therefore\ \theta = \mathbb{E}\left[h(U)\right]$ con $h(x) = \exp\left(-\left(\left(\frac{1}{x}-1\right)^2\right)\right)/x^2$.

$$\therefore \quad \theta \approx \frac{1}{n}\sum_{i=1}^{n} \frac{\exp\left(-\left(\left(\frac{1}{u_i}-1\right)^2\right)\right)}{u_i^2}, \qquad n \geq 1$$

con $u_1, \ldots, u_n$ realizaciones de $U_1, \ldots, U_n \sim \mathcal{U}(0,1)$. En Python,

```
from math import exp
from random import uniform

def sim_5d(n):
    def h(u):
        return exp(-(( 1/u - 1 )**2)) / (u**2)

    uniforms = [uniform(0, 1) for _ in range(n)]
    return 2*sum([ h(u) for u in uniforms ]) / n
```

La simulación con $n = 1000$ dio $\theta \approx 1.73$ que es casi exactamente $\sqrt{\pi}$, el verdadero valor.

Observación. El ejercicio se puede resolver aplicando directamente la sustitución sin notar que g es simétrica, aunque se dan pasos previos algo más engorrosos:

$$\begin{aligned}
\int_{-\infty}^{\infty} g(x)\,dx &= \lim_{t\to-\infty}\int_{t}^{0} g(x)\,dx + \lim_{t\to\infty}\int_{0}^{\infty} g(x)\,dx \\
&= -\int_{\lim_{t\to-\infty}\psi(t)}^{0} \frac{g(\frac{1}{\psi}-1)}{\psi^2}\,d\psi - \int_{0}^{\lim_{t\to\infty}\psi(t)} \frac{g(\frac{1}{\psi}-1)}{\psi^2}\,d\psi \\
&= \int_{0}^{1} \frac{g(\frac{1}{\psi}-1)}{\psi^2}\,d\psi + \int_{0}^{1} \frac{g(\frac{1}{\psi}-1)}{\psi^2}\,d\psi \\
&= 2\int_{0}^{1} \frac{g(\frac{1}{\psi}-1)}{\psi^2}\,d\psi
\end{aligned}$$

A partir de acá todo es igual.

(e) $\theta := \int_0^1 \int_0^1 \exp((x+y)^2)\, dx\, dy$

Let $X, Y \sim \mathcal{U}(0,1)$ independent with densities f_Y, f_U. Clearly, their joint density function is

$$f_{X,Y}(x,y) = f_X(x) f_Y(y) = \mathcal{I}_{(0,1)\times(0,1)}$$

Then

$$\theta = \int_0^1 \int_0^1 g(x,y) f_{X,Y}(x,y)\, dx\, dy = \mathbb{E}\left[g(X,Y)\right]$$

$$\therefore \quad \theta \approx \frac{1}{n} \sum_{i=1}^{n} g(x_i, y_i), \qquad n \geq 1$$

where $(x_1, y_1), \ldots, (x_n, y_n)$ are realizations of $(X_1, Y_1), \ldots, (X_n, Y_n)$, with $X_i, Y_i \sim \mathcal{U}(0,1)$ for $1 \leq i \leq n$.

In Python,

```
from math import exp
from random import uniform

def sim_5e(n):
    def g(x, y):
        return exp((x+y)**2)

    X = [uniform(0, 1) for _ in range(n)]
    Y = [uniform(0, 1) for _ in range(n)]

    return sum([ g(x, y) for x, y in zip(X, Y) ]) / n
```

With $n = 100000$, the estimation was 4.902, and the true value of the integral according to Wolfram Alpha is 4.899.

(7) Definamos

$$N = \min_n \left\{ n : \sum_{i=1}^{n} U_i > 1 \right\}, \qquad U_i \sim \mathcal{U}(0,1)$$

(a) Estimar $\mathbb{E}[N]$ generating n values for N.
(b) Calculate $E[N]$.

Notemos que $Im(N) = \{2, 3, \ldots\}$. Calculemos $p_N(n)$ la función de probabilidad de masa de N. Para ello, observemos que

$$P(N = n) = P\left(\sum_{i=1}^{n-1} U_i \leq 1 \cap U_n > 1 - \sum_{i=1}^{n-1} U_i \right)$$

Sea $X(n) = \sum_{i=1}^{n} U_i$ y, para simplificar la notación, usemos $\varphi(n, x) := P(X(n) \leq x)$ para $x \geq 0$. Es decir, $\varphi(n, x)$ es la función de probabilidad acumulada de $X(n)$. Entonces, la expresión de arriba nos queda

$$P(N = n) = P(X(n-1) < 1 \cap U_n > 1 - X(n-1))$$

De acuerdo con la ley de probabilidad total,

$$\begin{aligned}
\varphi(n, x) &= P(X(n) \leq x) \\
&= P(U_n \leq x - X(n-1)) \\
&= \int_{\mathbb{R}} P(U_n \leq x - z) P\left(X(n-1) = z\right)\ dz \\
&= \int_0^x (x - z) \frac{d}{dz} \varphi(n-1, z)\ dz
\end{aligned}$$

donde estamos utilizando el hecho de que la función de densidad es la derivada de la función de probabilidad acumulada. Esta expresión recursiva de $\varphi(n, x)$ nos permite avanzar: si restringimos $x \in [0, 1]$,

$$\begin{aligned}
\varphi(2, x) &= \int_0^x (x - z) \frac{d}{dz} \varphi(1, z)\ dz \\
&= \int_0^x (x - z)\ dz \\
&= \frac{x^2}{2}
\end{aligned}$$

Siguiendo probando casos, obtenemos

$$
\begin{aligned}
f(x,3) &= \int_0^x (x-z)z \; dz = \frac{x^3}{6} \\
f(x,4) &= \int_0^x (x-z)\frac{z^2}{2} \; dz = \frac{x^4}{24} \\
&\vdots \\
f(x,n) &= (x-t)\frac{d}{dz}\varphi(x,n-1)(x := z) \; dz = \frac{x^n}{n!}
\end{aligned}
$$

Es decir, obtenemos una expresión general para la probabilidad acumulada de $X(n)$, de lo cual se deduce que la densidad de $X(n)$ es

$$
\psi(n,x) = \frac{d}{dx}\frac{x^n}{n!} = \frac{nx^{n-1}}{n!} = \frac{x^{n-1}}{(n-1)!} = \varphi(n-1,x), \qquad 0 \leq x \leq 1
$$

Por lo tanto,

$$
\begin{aligned}
P(N=n) &= P(X(n-1) < 1 \cap U_n > 1 - X(n-1)) \\
&= \int_0^1 \psi(n-1,x) P(U_n > 1-x) \; dx \\
&= \int_0^1 \frac{x^{n-2}}{(n-2)!}(1 - P(U_n \leq 1-x)) \; dx \\
&= \frac{1}{(n-2)!}\int_0^1 x^{n-1} \; dx \\
&= \frac{1}{(n-2)!}\frac{1^n}{n} \\
&= \frac{1}{n(n-2)!}
\end{aligned}
$$

Por lo tanto,

$$
\mathbb{E}\left[N\right] = \sum_{i=2}^{\infty} \frac{1}{(n-2)!} = \sum_{i=0}^{\infty} \frac{1}{n!} = e
$$

La simulación está de acuerdo, porque en Python

```
def sim_7(iterations):
```

```
counts = []
n, s = 0, 0

for _ in range(iterations):
  while s <= 1:
    n += 1
    s += uniform(0, 1)
  counts.append(n)
  n, s = 0, 0

return sum(counts)/len(counts)
```

nos devuelve siempre aproximadamente e.

(8) Sea

$$N = \max_n \left(\prod_{i=1}^{n} U_i \geq e^{-3} \right)$$

con $\prod_{i=1}^{0} U_i = 1$. Estimar mediante simulaciones $E[N]$ y $P(N = n)$ para $0 \leq n \leq 6$.

Estimar con simulaciones es aburrido y fácil. Más vale pensemos en el problema y tratemos de dar p_N, la función de probabilidad de masa de N, y con ella $\mathbb{E}[N]$ de manera analítica.

Claramente,

$$\begin{aligned} P(N = n) &= P\left(\prod_{i=1}^{n-1} U_i \geq e^{-3} \cap U_n \cdot \prod_{i=1}^{n-1} U_i < e^{-3} \right) \\ &= P\left(\prod_{i=1}^{n-1} U_i \geq e^{-3} \cap U_n < \frac{e^{-3}}{\prod_{i=1}^{n-1} U_i} \right) \end{aligned}$$

Para simplificar la notación, hagamos $X(n) := \prod_{i=1}^{n} U_i$ y $\varphi(n, x) = P(X(n) \leq x)$ para $x \in [0, 1]$. Es decir, φ es la CDF de $X(n)$. Asumimos que $X(0) = 1$, con lo cual $\varphi(0, x) = 1$. Es fácil ver que $\varphi(1, x) = x$. Veamos además que

$$\begin{aligned} \varphi(2, x) &= P(U_1 U_2 \leq x) \\ &= P(U_1 \leq \frac{x}{U_2}) \\ &= \int_0^1 P(U_1 \leq \frac{x}{z}) P(U_2 = z)\, dz \\ &= \int_0^1 P\left(U_1 \leq \frac{x}{z} \right) dz \end{aligned}$$

Como $x/z \leq 1$ para $z \in [x, 1]$ y $x/z > 1$ para $z \in [0, x)$,

$$\begin{aligned}\int_0^1 P\left(U_1 \le \frac{x}{z}\right)\, dz &= \int_0^x P\left(U_1 \le \frac{x}{z}\right)\, dz + \int_x^1 P\left(U_1 \le \frac{x}{z}\right)\, dz \\ &= \int_0^x 1\, dz + \int_x^1 \frac{x}{z}\, dz \\ &= x + x\Big[\ln z\Big]_x^1 \\ &= x - x\ln x\end{aligned}$$

Una forma alternativa de dar con la probabilidad buscada es la siguiente:

$$\begin{aligned}\varphi(2,x) &= P(U_1U_2 \le x) \\ &= \int_0^x \int_0^1 dy\, dz + \int_x^1 \int_0^{x/z} dy\, dz \\ &= \int_0^x dz + \int_x^1 \frac{x}{z}\, dz \\ &= x + x\int_x^1 \frac{1}{z}\, dz \\ &= x + x\Big[\ln z\Big]_x^1 \\ &= x + x(\ln 1 - \ln x) \\ &= x - x\ln x\end{aligned}$$

Esta forma es más simple en este caso particular, pero la que usamos antes es más útil para el caso general.

Generalizando, llegamos a

$$
\begin{aligned}
\varphi(n,x) &= P\left(\prod_{i=1}^{n-1} U_i \leq \frac{x}{U_n}\right) \\
&= \int_0^1 \varphi\left(n-1, \frac{x}{z}\right) P(U_n = z)\, dz \\
&= \int_0^1 \varphi\left(n-1, \frac{x}{z}\right)\, dz \\
&= \int_0^x \varphi\left(n-1, \frac{x}{z}\right)\, dz + \int_x^1 \varphi\left(n-1, \frac{x}{z}\right) \\
&= \int_0^x 1\, dz + \int_x^1 \varphi\left(n-1, \frac{x}{z}\right) \\
&= x + \int_x^1 \varphi\left(n-1, \frac{x}{z}\right)
\end{aligned}
$$

Entonces, usando el hecho de que conocemos $\varphi(2,x)$, tenemos

$$
\begin{aligned}
\varphi(3,x) &= x + \int_x^1 \varphi(2, \frac{x}{z})\, dz \\
&= x + \int_x^1 \frac{x}{z} - \frac{x}{z}\ln\frac{x}{z} \\
&= x - x\ln x + \frac{x}{2}\ln^2 x
\end{aligned}
$$

$$
\begin{aligned}
\varphi(4,x) &= x + \int_x^1 \varphi(3, \frac{x}{z})\, dz \\
&= x + \int_x^1 \frac{x}{z} - \frac{x}{z}\ln\left(\frac{x}{z}\right) + \frac{x}{2z}\ln^2\left(\frac{x}{z}\right) \\
&= x - x\ln x + \frac{x\ln^2 x}{2} - \frac{x\ln^3 x}{6}
\end{aligned}
$$

Vistos estos resultados, notamos (aunque no demostramos formalmente) que:

$$
\varphi(n,x) = \sum_{k=0}^{n-1} (-1)^k \frac{x\ln^k x}{k!}
$$

Por lo tanto, la función de probabilidad de masa $\psi(n,x)$ del producto de n uniformes aleatorias es $\frac{d}{dx}\varphi(n,x)$, que es

$$
\begin{aligned}
&\sum_{k=0}^{n-1}(-1)^k \frac{\ln^k x + k\ln^{k-1} x}{k!} \\
=&1 - (\ln x + 1) + \left(\ln x + \frac{\ln^2 x}{2}\right) - \left(\frac{\ln^3 x}{6} + \frac{1}{2}\ln^2 x\right) + \dots \\
=&(-1)^{n-1}\frac{\ln^{n-1} x}{(n-1)!} && \{\text{Suma telescópica}\}
\end{aligned}
$$

Luego

$$
\begin{aligned}
P(N=n) &= P\left(\prod_{i=1}^{n-1} U_i \geq e^{-3} \cap U_n \cdot \prod_{i=1}^{n-1} U_i < e^{-3}\right) \\
&= P\left(\prod_{i=1}^{n-1} U_i \geq e^{-3} \cap U_n < \frac{e^{-3}}{\prod_{i=1}^{n-1} U_i < e^{-3}}\right) \\
&= \int_{e^{-3}}^{1} P(U_n < \frac{e^{-3}}{z}) P\left(\prod_{i=1}^{n-1} U_i = z\right)\, dz \\
&= \int_{e^{-3}}^{1} \left(\frac{e^{-3}}{z}\right) \psi(n-1, z)\, dz \\
&= \int_{e^{-3}}^{1} \left(\frac{e^{-3}}{z}\right) (-1)^{n-2} \frac{\ln^{n-2} z}{(n-2)!}\, dz \\
&= \frac{(-1)^{n-2} e^{-3}}{(n-2)!} \int_{e^{-3}}^{1} \frac{\ln^{n-2} z}{z}\, dz \\
&= \frac{(-1)^{n-2} e^{-3}}{(n-2)!} \left(-\frac{(-3)^{n-1}}{n-1}\right) \\
&= \frac{3^{n-1} e^{-3}}{(n-1)!} \\
&= \frac{3^{n-1}}{e^3 (n-1)!}
\end{aligned}
$$

Por lo tanto,

$$
\begin{aligned}
\mathbb{E}[N] &= \sum_{k=1}^{\infty} \frac{k3^{k-1}}{e^3(k-1)!} \\
&= \frac{1}{e^3} \sum_{k=0}^{\infty} 3^k \frac{k+1}{k!} \\
&= \frac{1}{e^3} \left(\sum_{k=0}^{\infty} k\frac{3^k}{k!} + \sum_{k=0}^{\infty} \frac{3^k}{k!} \right) \\
&= \frac{1}{e^3} \left(\frac{1}{e^{-3}} \sum_{k=0}^{\infty} k \cdot e^{-3} \frac{3^k}{k!} + \sum_{k=0}^{\infty} \frac{3^k}{k!} \right)
\end{aligned}
$$

Ahora somos pillos y vemos que la primera sumatoria es el valor esperado de una Poisson con parámetro $\lambda = 3$, y por lo tanto es 3. la otra sumatoria es e^3, dado que $e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$. Como $1/e^{-3} = e^3$,

$$
\therefore \ \mathbb{E}[N] = \frac{1}{e^3}(3e^3 + e^3) = \frac{1}{e^3} \cdot 4e^3 = 4
$$

Este valor coincide con la estimación de la siguiente simulación:

```
def sim_9(iterations):

  counts = []
  n, s = 0, 1

  for _ in range(iterations):
    while s >= exp(-3):
      n += 1
      s *= uniform(0, 1)
    counts.append(n)
    n, s = 0, 1

  return sum(counts)/len(counts)
```

(9) Un juego consiste en dos pasos. En el primer paso se tira un dado convencional. Si sale 1 o 6 tira un nuevo dado y se le otorga al jugador como puntaje el doble del resultado obtenido en esta nueva tirada; pero si sale 2, 3, 4 o 5 en la primer tirada, el jugador debería tirar dos nuevos dados, y recibiría como puntaje la suma de los dados. Si el puntaje del jugador excede los 6 puntos entonces gana.

(a) Calcular la probabilidad de que un jugador gane.

(b) Estimar mediante simulación.

Sea $X = 1$ si el primer dado es 1 o 6, $X = 0$ si el primer dado es 2, 3, 4 o 5. Claramente X es una Bernoulli con $p = 1/3$. Entonces, de acuerdo con la ley de probabilidad total,

$$P(\text{Ganar}) = P(2U > 6 \mid X = 1)P(X = 0) + P(U_1 + U_2 > 6 \mid X = 0)P(X = 0)$$

con U, U_2, U_3 uniformes discretas en $\{1, \ldots, 6\}$. Claramente, $U_1 + U_2 > 6$ y $2U$ son independientes de X. Por lo tanto

$$P(\text{Ganar}) = P(U > 3)1/3 + P(U_1 + U_2 > 6)2/3$$

Ahora bien, $P(U_1 + U_2 > 6) = 1 - P(U_1 \leq 6 - U_2)$, y

$$\begin{aligned} P(U_1 \leq 6 - U_2) &= \sum_{k=1}^{6} P(U_1 \leq 6 - k)P(U_2 = k) \\ &= \sum_{k=1}^{6} \frac{6-k}{6}\frac{1}{6} \\ &= \frac{1}{6}\sum_{k=1}^{6} 1 - \frac{k}{6} \\ &= 0.416 \end{aligned}$$

Por lo tanto, $P(U_1 + U_2 > 6) = 0.583$. Es fácil ver que $P(U > 3) = \frac{1}{2}$. Luego

$$P(\text{Ganar}) = \frac{1}{2} \cdot \frac{1}{3} + 0.583 \cdot \frac{2}{3} = 0.555$$

(b) En Python,

```
from random import randint
def sim_8(iterations):

  wins = 0

  for _ in range(iterations):
    first_dice = randint(1, 6)
    X = first_dice == 0 or first_dice == 6

    if X:
      result = 2*randint(1, 6)
    else:
      result = randint(1, 6) + randint(1, 6)

    wins = wins + 1 if result > 6 else wins

  return wins/iterations
```

3 P4

(1) . Se baraja un conjunto de $n = 100$ cartas (numeradas consecutivamente del 1 al 100) y se extrae del mazo una carta por vez. Consideramos que ocurre un "éxito" si la i-ésima carta extraída es aquella cuyo número es i ($i = 1, ..., n$).

a) Calcule la probabilidad de que (i) las primeras r cartas sean coincidencias y dé su valor para $r = 10$. (ii) Haya exactamente r coincidencias y estén en las primeras r cartas. Dé su valor para $r = 10$.

b) Pruebe que $E(X) = Var(X) = 1$ donde X es el número de coincidencias obtenidas en una baraja de n cartas.

c) Escriba un programa de simulación para estimar la esperanza y la varianza del número total de éxitos, y de los eventos del inciso (a) con $r = 10$, y compare los resultados obtenidos con 100, 1000, 10000 y 100000 iteraciones. Use el archivo "Problemas de coincidencias" para guiarse.

(a) (i) Sea $Y : \Omega \mapsto \mathbb{N}_0$ tal que $Y(\omega) = k$ si y solo si en $\omega \in \Omega$ las primeras k extracciones son coincidencias. Entonces $P(Y = 0) = 99/100$, y para $k \geq 1$:

$$P(Y = k) = \frac{1}{100} \cdot \frac{1}{99} \cdot \ldots \cdot \frac{1}{100 - (k-1)} = \frac{(100-k)!}{100!}$$

(ii) Sea $Z : \Omega \mapsto \mathbb{N}_0$ tal que $Z(\omega) = k$ si y solo si, en ω, las primeras k cartas extraídas son coincidencias y ninguna otra carta extraída es coincidencia. Sea $r := 100 - k$ la cantidad de cartas restantes en el mazo después de las primeras k extracciones.

Las r cartas restantes no tendrán éxitos si y solo si la permutación de ellas no contiene puntos fijos. La cantidad de permutacaiones sin puntos fijos en un conjunto de r elementos se denomina el *subfactorial* de r y se escribe $!r$. Su valor es

$$!r = r! \sum_{k=0}^{r} \frac{(-1)^k}{k!}$$

Luego, la probabilidad de que una permutación arbitraria de r elementos carezca de puntos fijos es

$$\frac{!r}{r!} = \frac{r! \sum_{k=0}^{r} \frac{(-1)^k}{k!}}{r!} = \sum_{k=0}^{r} \frac{(-1)^k}{k!}$$

Notar, como simple observación, que cuando $r \to \infty$ tenemos

$$\sum_{k=0}^{r} \frac{(-1)^k}{k!} \to \frac{1}{e}$$

Es decir, la probabilidad de que una permutación arbitraria carezca de puntos fijos se aproxima a $\frac{1}{e}$ a medida que la cantidad de elementos permutados crece. Pero volviendo a nuestro problema, tenemos entonces:

$$P(Z = k) = P(Y = k) \cdot \sum_{i=0}^{r} \frac{(-1)^i}{i!} = \frac{(100-k)!}{100!} \sum_{i=0}^{100-k} \frac{(-1)^i}{i!}$$

(b) Sea X el número de coincidencias obtenidas en n cartas, o bien el número de puntos fijos en una permutación aleatoria de $\{1, \ldots, n\}$.

Como la permutación es aleatoria, la probabilidad de que el i-écimo elemento tome el valor k es $\frac{1}{n}$. Sea $E_k = 1$ si el k-écimo elemento es una coincidencia, 0 de otro modo. Entonces que $E_k \sim \text{Bernoulli}(1/n)$. Es claro que $X = \sum_{i=1}^{n} E_i$. Luego

$$\mathbb{E}[X] = \sum_{i=1}^{n} \mathbb{E}[E_i] = n \cdot \frac{1}{n} = 1$$

Además,

$$\begin{aligned}
\mathbb{E}\left[X^2\right] &= \mathbb{E}\left[\left(\sum_{i=1}^{n} E_i\right)^2\right] \\
&= \mathbb{E}\left[\sum_{k=1}^{n} \sum_{\substack{i=1 \\ i\neq k}}^{n} E_k E_i + \sum_{i=1}^{n} E_i^2\right] \qquad \{\text{Ver } \star \text{ al final si hay dudas}\} \\
&= \sum_{k=1}^{n} \sum_{\substack{i=1 \\ i\neq k}}^{n} \mathbb{E}[E_k E_i] + \sum_{i=1}^{n} \mathbb{E}\left[E_i^2\right]
\end{aligned}$$

La razón por la cual dejamos los cuadrados a un lado y los otros productos del otro es que $E_i \cdot E_j$ siguen una distribución cuando $i \neq j$ y otra cuando $i = j$. Si $i = j$ obviamente $P(E_i E_j = 1) = 1/n$ y son Bernoulli con $p = 1/n$. Si $i \neq j$,

$$\begin{aligned}
&= P(E_j = 1 \cap E_i = 1) \\
&= P(E_j = 1 \mid E_i = 1) P(E_i = 1) \\
&= \frac{1}{n(n-1)}
\end{aligned}$$

$$
\begin{aligned}
\therefore\ \mathbb{E}\left[X^2\right] &= \sum_{k=1}^{n} \sum_{\substack{i=1 \\ i \neq k}}^{n} \mathbb{E}\left[E_k E_i\right] + \sum_{i=1}^{n} \mathbb{E}\left[E_i^2\right] \\
&= \frac{n(n-1)}{n(n-1)} + 1 \\
&= 2
\end{aligned}
$$

$$
\therefore\ \mathbb{V}\left[X\right] = \mathbb{E}\left[X^2\right] - \mathbb{E}^2\left[X\right] = 2 - 1 = 1
$$

(⋆) La expansión de la sumatoria es sencilla:

$$
\begin{aligned}
&(E_1 + \ldots + E_n)(E_1 + \ldots + E_n) \\
=&(E_1^2 + E_1E_2 + \ldots + E_1E_n) \\
+&(E_2E_1 + E_2^2 + \ldots + E_2E_n) \\
&\vdots \\
+&(E_nE_1 + E_nE_2 + \ldots + E_n^2) \\
=&(E_1E_2 + E_1E_3 + \ldots + E_1E_n) \\
+&(E_2E_1 + E_2E^3 + \ldots + E_2E_n) \\
&\vdots \\
+&(E_nE_1 + E_nE_2 + E_nE_{n-1}) \\
+&(E_1^2 + E_2^2 + \ldots + E_n^2) \\
=&\sum_{k=1}^{n} \sum_{i=1, i \neq k}^{n} E_kE_i + \sum_{i=1}^{n} E_i^2
\end{aligned}
$$

(c) En Python,

```
from random import random
from statistics import mean, variance

def random_permutation(l):
  N = len(l)
  for j in range(N-1, 0, -1):
    i = int( (j+1)*random() )
    l[j], l[i] = l[i], l[j]

  return l


def count_fixed_points(permutation):
  count = 0
  for i in range(len(permutation)):
    if permutation[i] == i:
      count +=1

  return count

def sim1(n_iterations):

  L = [i for i in range(100)]
  results = []

  for _ in range(n_iterations):
    permutation = random_permutation(L)
    fixed_points = count_fixed_points(permutation)
    results.append(fixed_points)

  return mean(results), variance(results)

# Example
x, y = sim1(1000)
print(f"E(X) = {x}, V(X) = {y}")
```

For $n = 1000$ this printed $E(X) = 0.988, V(X) = 1.0288848848848848$.

(2) Se desea construir una aproximación de

$$\sum_{k=1}^{N} \exp\left(\frac{k}{N}\right), \qquad N = 10\,000$$

(a) Escriba un algoritmo para estimar la cantidad deseada.
(b) Obtenga la aproximación sorteando 100 números aleatorios.
(c) Escriba un algoritmo para calcular la suma de los primeros 100 términos, y compare el valor exacto con las dos aproximaciones, y el tiempo de cálculo

(a) Sea $f(x) = \exp(x/N)$ y θ el valor que deseamos calcular. Dado que

$$\theta = N \cdot \left(\frac{1}{N}\sum_{k=1}^{N} f(k, N)\right) \approx N \cdot \mathbb{E}\left[f(U, N)\right]$$

con $U \sim \mathcal{U}\{1, \ldots, N\}$, podemos generar n variables uniformes $U_1, \ldots, U_n$ y calcular el promedio de $f(U_1), \ldots, f(U_n)$ para dar con una estimación.

El algoritmo entonces queda

```
def sim2(n):

  N = 10000
  def f(x):
    return exp(x/N)

  return N * mean([ f( randint(1, N) ) for _ in range(n)])

# Para comparar
def sim2_true():
  S = 0
  N = 10000
  for k in range(N):
    S += exp(k/N)
  return S
```

(3) Se lanzan simultáneamente un par de dados legales y se anota el resultado de la suma de ambos. El proceso se repite hasta que todos los resultados posibles: 2,3,...,12 hayan aparecido al menos una vez. Estudiar mediante una simulación la variable N, el número de lanzamientos necesarios para cumplir el proceso. Cada lanzamiento implica arrojar el par de dados.

(a) Describa la estructura lógica del algoritmo que permite simular en computadora el número de lanzamientos necesarios para cumplir el proceso.

(b) Mediante una implementación en computadora, (i) estime el valor medio y la desviación estándar del número de lanzamientos, repitiendo el algoritmo: 100, 1000, 10000 y 100000 veces. (ii) Estime la probabilidad de que N sea por lo menos 15 y la probabilidad de que N sea a lo sumo 9, repitiendo el algoritmo: 100, 1000, 10000 y 100000.

Estudiar sólo mediante simulaciones la variable N es aburrido porque programarlas es fácil. Estudiemos el problema matemáticamente antes de simular.

Sea $S = X_1 + X_2$ con X_1, X_2 independientes y uniformes discretas en $\{1, \ldots, 6\}$. La distribución de S (ver $\star$ al final de este ejercicio) es

$$p_S(x) = \begin{cases} \frac{x-1}{36} & 2 \leq x \leq 6 \\ \frac{13-x}{36} & 7 \leq x \leq 12 \\ 0 & c.c. \end{cases}$$

Sea $G(n) = \{S_i\}_{i=1}^{n}$ una secuencia aleatoria de n variables, donde cada $S_i = X_1 + X_2$ es independiente de los demás.

Como la imagen de S tiene 11 elementos (números del 2 al 12), nos interesa estudiar la probabilidad de que, para $n \geq 11$, se de el evento

$$\mathcal{A}(n) := \forall k : 2 \leq k \leq 12 : k \in G(n)$$

Si $n = 11$, hay una única combinación de números válida (la que contiene todos distintos) que puede darse de $11!$ maneras diferentes (no es lo mismo $\{2, 3, \ldots, 12\}$ que $\{12, 11, \ldots, 2\}$, por ejemplo). En particular, si $k_1, \ldots, k_{11}$ es el orden en que se dan los números de una secuencia arbitraria que satisface $\mathcal{A}(n)$, la probabilidad de la secuencia es

$$P(S_1 = k_1) \cdot \ldots \cdot P(S_{11} = k_{11})$$

Pero $P(S_i = k_i) = P(S = k_i)$ (es decir, la posición en que el número aparece no afecta la probabilidad). Por ende, el producto arriba es

$$P(S = k_1) \cdot \ldots \cdot P(S = k_{11}) = \prod_{i=2}^{12} P(S = i)$$

Como debemos tener en cuenta que hay $11!$ secuencias $k_1, \ldots, k_{11}$ que satisfacen la condición,

$$\therefore \; P(\mathcal{A}(11)) = 11! \prod_{i=2}^{12} P(S = i)$$

Para $n > 11$, el problema es inabordable a mano. Nos damos por satisfechos con el estudio del experimento que hicimoso hasta acá, porque consideramos que da una intuición correcta sobre el problema en cuestión.

($\star$) Es fácil ver que, si $2 \leq x \leq 6$,

$$p_S(x) = \sum_{k=1}^{x-1} p_{X_1}(k) p_{X_1}(x-k) = \frac{x-1}{36}$$

Si $7 \leq x \leq 12$, podemos escribir x como $6+k$ para $1 \leq k \leq 6$, y

$$P(S=x) = P(S=6+k) = \sum_{j=k}^{6} p_{X_1}(j) p_{X_1}(x-j) = \frac{6-k+1}{36} = \frac{13-x}{36}$$

donde en el último paso usamos que $k = x-6$.

$$\therefore \; p_S(x) = \begin{cases} \frac{x-1}{36} & 2 \leq x \leq 6 \\ \frac{13-x}{36} & 7 \leq x \leq 12 \\ 0 c.c. & \end{cases}$$

(4) Implemente cuatro métodos para generar una variable X que toma los valores del 1 al 10, con probabilidades

$$\begin{array}{llll} p_1 = 0.11 & p_2 = 0.14 & p_3 = 0.09 & p_4 = 0.08 \\ p_5 = 0.12 & p_6 = 0.10 & p_7 = 0.09 & p_8 = 0.07 \\ & p_9 = 0.11 & p_{10} = 0.09 & \end{array}$$

usando

(a) Método de rechazo con una uniforme discreta, buscando la cota c más baja posible.

(b) Método de rechazo con una uniforme discreta, usando c = 3.

(c) Transformada inversa.

(d) Método de la urna: utilizar un arreglo A de tamaño 100 donde cada valor i está en exactamente $p_i \cdot 100$ posiciones. El método debe devolver A[k] con probabilidad 0,01. ¿Por qué funciona?

Compare la eficiencia de los tres algoritmos realizando 10000 simulaciones.

(a) Sea $Y \sim \mathcal{U}\{1, \ldots, 10\}$ uniforme discreta. Para encontrar la c mínima, resolvemos

$$\frac{0.12}{1.10} \leq c \iff 1.2 \leq c$$

con lo cual fijamos $c = 1.2$. Un algoritmo para generar n valores de X:

```
def sim4_rejection_method(n_generations):

  p = [0.11, 0.14, 0.09, 0.08, 0.12, 0.10, 0.09, 0.07, 0.11, 0.09]

  generations = []

  for _ in range(n_generations):
    while True:
      u = random()
      y = randint(1, 10)
      if u < p[y - 1]/( 1.2 * 0.1 ):
        generations.append(y)
        break

  return generations
```

(b) No es muy distinto de (a), lo salteamos.

(c) Ordenemos las probabilidades de mayor a menor:

$$p_2 = 0.14 \qquad p_5 = 0.12 \qquad p_1 = p_9 = 0.11 \qquad p_6 = 0.10$$
$$p_3 = p_7 = p_{10} = 0.09 \qquad p_4 = 0.08 \qquad p_8 = 0.07$$

Sean $\{x_i\}_{i\in\mathbb{N}} = Im(X)$. Recordemos que en el método de la transformada inversa, generamos una uniforme $U \sim \mathcal{U}(0,1)$. Si $U \in [F(x_i), F(x_{i+1}))$ (es decir, si U supera la probabilidad acumulada de x_i pero no la de x_{i+1}), producimos la variable x_{i+1}.

Es fácil ver que F_X es dada por

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 0.11 & 0.25 & 0.34 & 0.42 & 0.54 & 0.64 & 0.73 & 0.8 & 0.91 & 1 \end{bmatrix}$$

donde la primera fila es $Im(X)$ y la segunda fila la probabilidad acumulada de cada valor. Usando el ordenamiento de las variables que dimos al principio, obtenemos:

```
def sim4_inverse_transform(n_generations):

  p = [0.11, 0.14, 0.09, 0.08, 0.12, 0.10, 0.09, 0.07, 0.11, 0.09]
  ordering = [1, 4, 0, 5, 2, 6, 9, 4, 7]                              # := o
  ordered_probabilities = [p[i] for i in ordering]                    # := op
  u = random()
  for i in range(len(p)):
    if u < sum(ordered_probabilities[0:i+1]):
      return ordering[i] + 1
```

Este algoritmo guardan en $\overrightarrow{p}$ las probabilidades, en $\overrightarrow{o}$ el orden decreciente de mayor a menor, y en $\overrightarrow{op}$ las probabilidades ordenadas de mayor a menor. En la iteración i, revisa si $U \sim \mathcal{U}(0,1)$ es menor a

$$\overrightarrow{op}[0] + \ldots + \overrightarrow{op}[i-1] = \sum_{0=i}^{i-1} \overrightarrow{op}[i]$$

y si lo es, devuelve $\overrightarrow{o}[i] + 1$, donde sumar uno se hace para pasar de $\{0, \ldots, 9\}$ (útil en Python) a $\{1, \ldots, 10\}$.

(d) El método funciona porque elige de manera uniforme un índice $i \in \{0, \ldots, 99\}$ y devuelve $A[i]$. Pero

$$P(A[i] = k) = \frac{\#\text{ veces que aparece } k \text{ en } A}{\#\text{elementos de } A} = \frac{p_k \cdot 100}{100} = p_k$$

```
def sim4_urna():

  p = [0.11, 0.14, 0.09, 0.08, 0.12, 0.10, 0.09, 0.07, 0.11, 0.09]
  times_it_occurs = [int(x * 100) for x in p]
  A = [ [i+1] * times_it_occurs[i] for i in range(len(p))] # list of lists
  A = sum(A, []) # Python trick, simple but inefficient: joins the list of list
  return choice(A) # import choice from random
```

(5) Genere una variable binomial $\mathcal{B}(n, p)$ con (a) método de la transformada inversa y (b) simulación de n Bernoullis con probabilidad de éxito p.

(a) Lo primero es encontrar una fórmula recursiva para para la probabilidad de una binomial. Así, cada iteración podrá calcularse a partir de la anterior. Pero si probamos $P(X \leq k)$ para $k = 0, 1, 2, \ldots$, un patrón se hace evidente y es fácil demostrar por inducción que

$$P(X = k) = \begin{cases} (1-p)^n & k = 0 \\ \frac{n-(k-1)}{k} \frac{p}{1-p} F_B(k-1) & k > 0 \end{cases}$$

Por lo tanto, podemos hacer:

```
def sim5_binomial(p, n):

  cdf = (1-p)**n           # Probabilidad acumulada en k = 0
  u = random()
  k = 0
  constant = p/(1-p)       # Calculemos esto una sola vez
  while True:
    if u < cdf:
      return k
    k += 1
    # El producto a la derecha es la probabilidad P(X = k),
    # lo sumamos a cdf y tenemos la acumulada P(X <= k).
    cdf += cdf * ( ( n-k+1 )/k ) * constant

  return k
```

(b) Con Bernoullis es mucho más fácil:

```
def sim5_bernoullis(p, n):

  count = 0
  for _ in range(n):
    if random() <= p:
      count +=1

  return count
```

(8) (a) Simule usando transformada inversa y aceptación y rechazo la variable X con distribución:

$$P(X = i) = \frac{\frac{\lambda^i}{i!}e^{-\lambda}}{\sum_{j=0}^{k}\frac{\lambda^j}{j!}e^{-\lambda}}, \qquad i = 0, \ldots, k$$

(b) Hágalo en el caso en que $\lambda = 0.7, k = 10$.

(a) Notemos que

$$P(X = i) = \frac{f_Y(i)}{F_Y(k)}$$

donde $Y \sim \mathcal{P}(\lambda)$. El denominador es simplemente una constante C, y por lo tanto puede guardarse en una variable `den`. Para el numerador, podemos usar simplemente la recurrencia de la distribución de Poisson. Es decir,

$$P(X = 0) = \frac{e^{-\lambda}}{C}, P(X = 1) = \frac{e^{-\lambda}(\frac{\lambda}{1})}{C}, P(X = 2) = \frac{e^{-\lambda}\frac{\lambda}{1}\frac{\lambda}{2}}{C}, \ldots$$

Entonces, usando el método de la transformada inversa,

```
def sim8(k, lamda):
  den = sum( [ (lamda**j / factorial(j)) * exp(-lamda) for j in range(k+1) ] )
  i = 0
  p = exp(-lamda)/den
  F = p
  U = random()

  while True:
    if U < F:
      return i
    i += 1
    p = p * (lamda/i)
    F += p
```

Usando el método de aceptación y rechazo, tomamos $Y \sim \mathcal{U}\{0, \ldots, k\}$. Claramente, $f_Y(x_j) = 1/(k+1)$ para todo j, con lo cual buscamos el máximo $f_X(x_j)$. Pero f_X tiene un denominador constante y un numerador que es la densidad de Poisson, cuyo máximo es $\lfloor\lambda\rfloor$. Por lo tanto, tomamos

$$c := \frac{\lceil\lambda\rceil}{\frac{1}{k+1}} = (k+1)\lceil\lambda\rceil$$

Por lo tanto,

```
def sim8_aceptacionyrechazo(k, lamda):
  den = sum( [ (lamda**j / factorial(j)) * exp(-lamda) for j in range(k+1) ] )
  c = ceil(lamda) * ( k+1 )
  while True:
    y = randint(0, k)
    num = exp(-lamda) * ((lamda**y)/factorial(y)) # Numerator of density of X=y
    px = num/den # P(X = y)
    u = random()
    if u < px/(c /(k+1)):
      return y
```

(b) El valor exacto es

$$\begin{aligned} P(X > 2) &= 1 - P(X \leq 2) \\ &= 1 - \left(\frac{e^{-\lambda}}{C} + \frac{e^{-\lambda}\lambda}{C} + \frac{e^{-\lambda}\frac{\lambda}{1}\frac{\lambda}{2}}{C} \right) \\ &= 1 - \frac{1}{C}(e^{-\lambda} + \lambda e^{-\lambda} + \frac{\lambda^2}{2}e^{-\lambda}) \\ &= 1 - \frac{0.965}{0.999} \\ &= 0.034 \end{aligned}$$

Simular `sim8(10, 0.7)` $n = 1000$ veces y contar las veces que da un valor mayor a 2 me dio 0.032. Hacerlo con el método de aceptación y rechazo me dio 0.036.

(9) Implemente dos métodos para generar una variable geométrica.

(a) Usando transformada inversa aplicando la fórmula recursiva para $P(X = i)$.

(b) Simulando ensayos con probabilidad de éxito p hasta obtener un éxito.

La variable geométrica cuenta la cantidad de ensayos hasta tener un éxito. Si X es geométrica con probabilidad p, entonces $P(X = 1) = p, P(X = 2) = (1 - p)p, P(X = 3) = (1 - p)^2 p, \ldots$ En general, para $k \geq 2$,

$$P(X = k) = (1 - p) \cdot P(X = k - 1)$$

De aquí se sigue fácilmente cómo aplicar el método de aceptación y rechazo:

```
def sim9_geometrica_transformada_inversa(p):
  k = 1
  prob = p
  F = prob
  u = random()

  while True:
    if u < F:
      return k
    k += 1
    prob *= (1 - p)
    F += prob

def sim9_geometrica_alobestia(p):

  k = 1
  while True:
    u = random()
    if u <= p:
      return k
    k += 1
```

4 P5

(1) De métodos para generar variables con distribución:

$$(a) \qquad f(x) = \begin{cases} \frac{x-2}{2} & 2 \leq x \leq 3 \\ \frac{2-x/3}{2} & 3 \leq x \leq 6 \\ 0 & c.c. \end{cases}$$

$$(b) \qquad f(x) = \begin{cases} \frac{6(x-3)}{35} & 0 \leq x \leq 1 \\ \frac{6x^2}{35} & 1 \leq x \leq 2 \\ 0 & c.c. \end{cases}$$

$$(c) \qquad f(x) = \begin{cases} \frac{e^{4x}}{4} & -\infty < x \leq 0 \\ \frac{1}{4} & 0 \leq x \leq \frac{15}{4} \\ 0 & c.c. \end{cases}$$

(a) Determinemos la función de densidad acumulada de cada una.

$$(x \in [2,3]) \qquad F(X) = \int_2^x \frac{t-2}{2}\, dt = \frac{(x-2)^2}{4}$$

$$(x \in [2,3]) \qquad F(X) = \int_3^x \frac{2-\frac{x}{3}}{2}\, dt = \frac{(x-2)^2}{4}$$

donde estamos fijando la constante de integración C en cero. Observemos que

$$x - \frac{x^2}{12} + \frac{x^2}{4} - x = \frac{3x^2}{12} - \frac{x^2}{12} = \frac{x^2}{6}$$

Entonces

$$F(x) = \begin{cases} 0 & x < 2 \\ \frac{x^2}{4} - x & 2 \leq x \leq 3 \\ \frac{x^2}{6} & 3 \leq x \leq 6 \\ 1 & c.c. \end{cases}$$

Observemos que $F(3) = \frac{9}{6}$

(2) (a) Genere

$$(i)\ f(x) = ax^{-(a+1)} \qquad 1 \leq x < \infty, a > 0$$
$$(ii)\ f(x) = \frac{x^{k-1}\exp(-x/\mu)}{(k-1)!\mu^k} \qquad 0 \leq x < \infty, \mu > 0$$
$$(iii)\ f(x) = \frac{\beta}{\gamma}(\frac{x}{\lambda})^{\beta-1}\exp\left(-\left(\frac{x}{\lambda}\right)^{\beta}\right), 0 \leq x, \lambda > 0, \beta > 0$$

(a) (i) Sea $f(x) = ax^{-(a+1)}$ con $1 \leq x < \infty, a > 0$. Entonces

$$F(x) = \int_1^x f(z)\, dz = 1 - x^{-a}$$

Sea $u \in (0,1)$. Luego

$$\begin{aligned} F(x) = u &\iff 1 - x^{-a} = u \\ &\iff x^{-a} = 1 - u \\ &\iff x^a = \frac{1}{1-u} \\ &\iff x = \frac{1}{\sqrt[a]{1-u}} \end{aligned}$$

Por lo tanto, el método de la transformada inversa nos da

```
def pareto(a):
  u = random()
  den = (1-u)**(1/a)
  return 1/den

# Las dist. de U y 1 - U son la misma, así que podríamos haber hecho:
def pareto(a):
  u = random()
  den = u**(1/a)
  return 1/den
```

(ii) Sea

$$f(x;k,\mu) = \frac{x^{k-1}\exp\left(-\frac{x}{\mu}\right)}{(k-1)!\mu^k}, \qquad 0 \leq x < \infty, \mu > 0$$

La distribución Erlang, cuya densidad es f, corresponde a la suma de k exponentiales independientes con media μ. Como la media de la exponencial es

$\frac{1}{\lambda}$, que tenga media μ significa que $\frac{1}{\lambda} = \mu \Rightarrow \lambda = \frac{1}{\mu}$. Es decir, debemos ser cuidadosos y asegurarnos de tomar $1/\mu$ como el parámetro de las exponenciales simuladas.

Veamos que si $u \in (0,1)$ y $Y \sim \mathcal{E}(\lambda)$

$$\begin{aligned} F_Y(x) = u &\iff 1 - e^{-\lambda x} = u \\ &\iff 1 - u = e^{-\lambda x} \\ &\iff \ln(1-u) = -\lambda x \\ &\iff -\frac{\ln(1-u)}{\lambda} = x \end{aligned}$$

Por lo tanto, una única exponencial se simula como sigue:

```
def exponential(lamda):
  U = random()
  num = log(1 - U)
  den = - lamda
  return num/den
```

Por lo tanto, la variable Erlang se simula como sigue:

```
def erlang(k, mu):
  return sum( [exponential(1/mu) for _ in range(k)] ) # 1 / mu como parametro
```

(*iii*) Sea

$$f(x) = \frac{\beta}{\gamma}\left(\frac{x}{\lambda}\right)^{\beta-1} \exp\left(-\left(\frac{x}{\lambda}\right)^{\beta}\right), 0 \leq x < \infty, \lambda > 0, \beta > 0$$

Veamos que

$$\begin{aligned} F(x) &= \int_0^x f(z)\, dz \\ &= \frac{\beta}{\gamma\lambda^{\beta-1}} \int_0^x z^{\beta-1} \exp\left(-\left(\frac{z}{\lambda}\right)^{\beta}\right) dz \\ &= \frac{\beta}{\gamma\lambda^{\beta-1}} \left(\lambda^{\beta-1}\right) \int_0^{-(\frac{x}{\lambda})^{\beta}} u^{\frac{\beta}{\beta-1}} \exp u \, \frac{\beta z^{\beta-1}}{\lambda^{\beta}} du \qquad \left\{u := -\left(\frac{z}{\lambda}\right)^{\beta}\right\} \\ &= \frac{\beta}{\gamma} \cdot \frac{\beta}{\lambda^{\beta}} \int_0^{(-\frac{x}{\lambda})^{\beta}} u^{\frac{\beta}{\beta-1}} \exp u \left(-\lambda^{\beta-1} u^{\frac{\beta}{\beta-1}}\right) du \\ &= -\frac{\beta^2\lambda^{\beta-1}}{\gamma\lambda^{\beta}} \int_0^{-(\frac{x}{\lambda})^{\beta}} u^{\frac{2\beta}{\beta-1}} \exp u \, du \end{aligned}$$

Acá nos quedamos...

(3) (a) Suponga que se pueden generar n variables aleatorias a partir de sus distribuciones de probabialidad $F_i, i = 1, \ldots, n$. Implemente un método para generar una aleatoria cuya distribución es

$$F(x) = \sum_{i=1}^{n} p_i F_i(x)$$

con p_i positivos y tales que $\sum p_i = 1$.

Sea $\{p_i\}$ una secuencia de n valores positivos cuya suma es 1.

Hay n variables $X_1, \ldots, X_n$ que sabemos generar.

1. Generar Y una variable tal que $P(Y = i) = p_i$.
2. Generá un valor de X_Y.
3. Devolvé ese valor.

```
def generation():
  funciones_generadoras = [gen1, gen2, ..., genn]
  Y = generar_Y_con_metedo_de_la_urna()
  funcion_generadora = funciones_generadoras[Y]
  return funcion_generadora()
```

(4) Desarrolle un método para generar la variable aleatoria con distribución

$$F(x) = \int_0^\infty x^y e^{-y}\, dy, \qquad 0 \le x \le 1$$

suponiendo que

$$P(X \le x \mid Y = y) = x^y, \qquad 0 \le x \le 1$$

Notemos que $f(y) = e^{-y}$ es la densidad de una exponencial con parámetro $\lambda = 1$. Por ende, según el supuesto dado, lo que tenemos es

$$\begin{aligned} F(x) &= \int_0^\infty (P(X \le x) \mid Y = y)\, e^{-y}\, dy \\ &= \int_0^\infty P(X \le x \mid Y = y) P(Y = y)\, dy \end{aligned}$$

donde $Y \sim \mathcal{E}(1)$. Es decir, la fórmula de $F(x)$ se deriva de aplicar la ley de probabilidad total a X condicionada sobre el soporte de Y.

Como, dado un valor y de Y, tenemos $F(x) = x^y$, obtenemos que si $u \in \mathcal{U}(0, 1)$:

$$F(x; y) = u \iff x^y = u \iff x = \sqrt[y]{u}$$

1. Generar un valor de y de Y. Sabemos hacerlo porque $Y \sim \mathcal{E}(1)$.
   (a) Notar que esto condiciona $P(X \le x) = x^y$.
2. Generar $U \sim \mathcal{U}(0, 1)$.
3. Devolver $\sqrt[y]{u}$.

Dado un valor de Y, la probabilidad de que X esté en $[0, 0.5]$ es 0.5^y, y la probabilidad de que esté en $(0.5, 1)$ es $1 - 0.5^y$.

(7) (a) Desarrolle dos métodos para generar X con densidad

$$f_X(x) = \begin{cases} \frac{1}{x} & 1 \leq x \leq e \\ 0 & c.c. \end{cases}$$

uno aplicando transformada inversa y el otro con aceptación y rechazo.
(c) Estime $P(X \leq 2)$ y compare con el algoritmo.

(a: Con transformada inversa) Veamos que

$$\begin{aligned} F(x) &= \int_1^x \frac{1}{t}\, dt, \qquad x \leq e \\ &= [\ln |t|]_1^x \\ &= \ln x - \ln 1 \\ &= \ln x \end{aligned}$$

Si $u \in (0, 1)$,

$$\begin{aligned} F(x) = u &\iff \ln x = u \\ &\iff x = e^u \end{aligned}$$

Por lo tanto, `return exp(random())` hace lo que queremos.
(b: Con aceptación y rechazo) Sea $U \sim \mathcal{U}(1, e)$. Queremos encontrar $c \in \mathbb{R}$ tal que

$$\frac{f_X(x)}{f_U(x)} \leq c \iff \frac{f_X(x)}{\frac{1}{e-1}} \leq c \iff (e-1) f_X(x) \leq c$$

Veamos que, para $x \in [1, e]$,

$$\frac{d}{dx} f = -\frac{1}{x^2}$$

Por lo tanto, f es monótona decreciente en dicho intervalo y tiene máximo dado por $x = 1$. Por lo tanto, elegimos

$$c := e - 1 \cdot f_X(1) = e - 1$$

El algoritmo queda así:

```
c = e - 1
while True:
  y = random(1, e)
  u = random()

  if u <= (1/y) / (c * 1/(e-1)):
    return y
```

Si notamos que $\frac{c}{e-1} = 1$ en realidad podríamos escribir solamente `1/y`, pero por didáctica prefiero dejarlo explícito.

(c) Sabemos que $P(X \leq 2) = F(2) = \ln 2$.

(8) Sean U, V dos uniformes independientes en $(0, 1)$. Probar que $X = U + V$ tiene densidad triangular y desarrolle tres algoritmos que simulen X.

Estudiemos $P(U + V = z)$. Veamos que si $z \in [0, 1]$, entonces

$$P(U + V = z) = \int_0^z f_U(x) f_V(z - x)\, dx = \int_0^z 1\, dx = z$$

Si $z \in (1, 2]$, entonces hacemos variar U entre $[z - 1, 1]$ y requerimos $V = z - U$:

$$P(U + V = z) = \int_{z-1}^1 f_U(x) f_V(z - x)\, dx = 2 - z$$

$$\therefore\ f_{U+V}(x) = \begin{cases} x & 0 \le x \le 1 \\ 2 - x & 1 < x \le 2 \\ 0 & c.c. \end{cases}$$

El primer algoritmo es una tontería: hacés `random() + random()`. Con transformada inversa, vemos lo siguiente:

$$F_X(x) = \begin{cases} 0 & x < 0 \\ \int_0^x t\, dt & 0 \le x \le 1 \\ \int_1^x 2 - t\, dt & 1 \le x \le 2 \\ 1 & c.c. \end{cases} = \begin{cases} 0 & x < 0 \\ \frac{x^2}{2} & 0 \le x \le 1 \\ -\frac{x}{2} + 2x - \frac{3}{2} & 1 \le x \le 2 \\ 1 & c.c. \end{cases}$$

Veamos que para $x \in [0, 1]$, $u \in (0, 1)$,

$$\frac{x^2}{2} = u \iff x = \sqrt{2u}$$

Para $x \in [1, 2]$,

$$-\frac{x}{2} + 2x - \frac{3}{2} = u \iff -\frac{x}{2} + 2x - \left(\frac{3}{2} - u\right)$$
$$\iff x = 2 \pm \sqrt{1 - 2u}$$

Claramente, $2 + \sqrt{1 - 2u} > 2$ y por lo tanto x no puede tomar dicho valor. Por otra parte, $2 - \sqrt{1 - 2u}$

PAUSA.

Con método de aceptación y rechazo, tomamos $U \sim (0, 2)$, de manera tal que $Im(X) \subset Im(U)$. Tenemos encontrar c tal que

$$\forall x \in (0, 2) : \frac{f_X(x)}{f_U(x)} \le c \equiv \forall x \in (0, 2) : 2 \cdot f_X(x) \le c$$

No es difícil ver que $\max_x \{f_X(x)\} = 1$. Por lo tanto, definimos

$$c := 2 \cdot 1 = 2$$

```
while True:
  U = random()
  Y = uniform(0, 2)
  c = 2

  def densidad_de_x(x):
    if x <= 1:
      return x
    if x <= 2:
      return 2 - x

  if U <= densidad_de_x(Y)/(c*1/2):
    return Y
```

(10) Sea X una variable con distribución de Cauchy con parámetro λ, i.e.

$$f_X(x) = \frac{1}{\lambda\pi\left(1 + \left(\frac{x}{\lambda}\right)^2\right)}, \qquad x \in \mathbb{R}$$

(a) Implemente método de razón entre uniformes par asimular X con $\lambda = 1$.

Describamos el método de razón entre uniformes. Adapto el teorema de Wikipedia al caso de una variable unidimensional (i.e. $X \in \mathbb{R}^1$).

Sea $r > 0$ un parámetro arbitrario, X una variable aleatoria con densidad f_X. Sea

$$A = \left\{(u, v) \in \mathbb{R}^2 : 0 \leq u \leq f_X\left(\frac{v}{u^r}\right)^{\frac{1}{2r}}\right\}$$

Sea (U, V) una muestra aleatoria uniforme de A. Entonces $\frac{V}{U^r}$ es una variable aleatoria con distribución idéntica a la de X.

Ahora observemos que, para $\lambda = 1$, la densidad de X es

$$f_X(x) = \frac{1}{\pi\left(1 + x^2\right)} \in (0, 1]$$

Por lo tanto, si fijamos $r = 1$,

$$\begin{aligned}
0 \leq u \leq f_X\left(\frac{v}{u^r}\right)^{\frac{1}{2}} &\iff 0 \leq u \leq \frac{1}{\sqrt{\pi\left(1 + \left(\frac{v}{u}\right)^2\right)}} \\
&\iff 0 \leq u^2 \leq \frac{1}{\pi + \pi\left(\frac{v}{u}\right)^2} \\
&\iff 0 \leq u^2\left(\pi + \pi\left(\frac{v}{u}\right)^2\right) \leq 1 \\
&\iff 0 \leq \pi + \pi\left(\frac{v}{u}\right)^2 \leq \frac{1}{u^2} \\
&\iff 0 \leq \pi\left(\frac{v}{u}\right)^2 \leq \frac{1}{u^2} - \pi \\
&\iff 0 \leq \frac{v^2}{u^2} \leq \frac{1}{\pi u^2} - 1 \\
&\iff 0 \leq v^2 \leq \frac{1}{\pi} - u^2 \\
&\iff 0 \leq v \leq \sqrt{\frac{1}{\pi} - u^2}
\end{aligned}$$

Como ya vimos que f_X es a lo sumo 1, podemos generar $U \sim \mathcal{U}(0, 1)$ para garantizar que tenemos $0 \leq U \leq f_X\left(\frac{v}{u}\right)^{\frac{1}{1+2r}}$ para todo v. Luego, tomamos

$$V \sim \mathcal{U}\left(0, \sqrt{\frac{1}{\pi} - U^2}\right)$$

para garantizar que V pertenece al intervalo requerido. Se sigue que (U, V) será una muestra uniforme de A. Por lo tanto, en pseudo-código, obtenemos el siguiente algoritmo:

```
U = random()
V = uniform(0, sqrt( 1/pi - U**2 )  )
return V/U
```

donde el método garantiza que `V/U` es aleatoria con la misma distribución que X.

5 P6

(2) Monte-carlo estimation:

$$\int_0^1 \frac{e^x}{\sqrt{2x}}\,dx, \qquad \int_{-\infty}^{\infty} x^2 \exp(-x^2)\,dx$$

Generate $n = 100, 101, 102, \ldots$ samples and stop only when the standard error of the estimator (its standard deviation) is < 0.01.

Sea f_U la densidad de $U \sim \mathcal{U}(0,1)$. Observemos que

$$\theta := \int_0^1 \psi(x)\,dx = \int_0^1 \psi(x) f_U(x)\,dx = \mathbb{E}\left[\psi(U)\right]$$

para toda $\psi(x)$ integrable en $[0,1]$. Por ende, by the law of large numbers,

$$\theta \simeq \frac{1}{N}\sum_1^N \psi(u_i)$$

for sufficiently large N and $u_1, \ldots, u_N$ being N samples of U. Notemos que acá el estimador es

$$\frac{1}{N}\sum_1^N \psi(u_i)$$

la media muestral de N muestras de $\psi(U)$. La desviación estándar de la media muestral es $\frac{\sigma}{\sqrt{N}}$, y por ende la desviación estándar muestral del estimador es $\frac{s}{\sqrt{N}}$ con s la desviación estándar muestral. Por ende el algoritmo no hace más que esto:

1. Seteá $n = 100$.
2. Generá $\psi(u_1), \psi(u_2), \ldots, \psi(u_{100})$ y calculá la media muestral $\overline{\psi}$ y la desviación estándar muestral s de dicha muestra de manera recursiva.
3. Si $\frac{s}{\sqrt{n}}$ es menor a 0.01 terminá y devolvé $\overline{\psi}$.
4. Generá $\psi(u)$ y actualizá la media y la desviación estándar tras añadir este elemento. Seteá $n = n + 1$. Volvé a (3).

Para la segunda integral, tenemos

$$\int_{-\infty}^{\infty} \psi(x)\,dx = 2\int_0^1 \frac{\psi\left(\frac{1}{u} - 1\right)}{u^2}\,du$$

con $u = \frac{1}{x+1}, x = \frac{1}{u} - 1, du = -u^2$. En nuestro caso particular,

$$\psi\left(\frac{1}{u} - 1\right) = \left(\frac{1}{u} - 1\right)^2 \exp\left(-\left(\frac{1}{u} - 1\right)^2\right)$$

y por ende

$$\theta = 2\int_0^1 \frac{\left(\frac{1}{u} - 1\right)^2 \exp\left(-\left(\frac{1}{u} - 1\right)^2\right)}{u^2}\,du$$

Demás está decir que lo más inteligente es hacer un único algoritmo que, dada una función ψ, una cantidad mínima de muestras $n_{\min}$, y una cota superior para el error de estmación ϵ, estima la integral $\int_0^1 \psi(x)\, dx$ recursivamente.

```
def mean_recursive(old_mean, new_sample, n):
    new_mu = old_mean + ( (new_sample - old_mean) / ( n+1 ) )
    return new_mu

def var_recursive(old_var, new_sample, old_mean, new_mean, n):
    return old_var + ((new_sample - old_mean) * (new_sample - new_mean) - old_var) / (n + 1)

def montecarlo_min_error(f, bound, min_sample_size):
  n, mean, sd = 0, 0, 0
  while n < min_sample_size or sd/sqrt(n) >= bound:
    e = f(random())
    new_mean = mean_recursive(mean, e, n)
    sd = sqrt( var_recursive(sd**2, e, mean, new_mean, n) )
    mean = new_mean
    n += 1

  return mean
```

Luego basta pasarle a ese algoritmo las expresiones que corresponden a las integrales en $[0, 1]$. Para este ejercicio particular, la solución es dada por:

```
def f1(x):
  return exp(x)/(sqrt(2*x))

def f2(u):
  return 2* ( ( (1/u - 1)**2 * exp( -(1/u - 1)**2 ) ) / (u**2) )

print(montecarlo_min_error(f1, 0.01, 100))
print(montecarlo_min_error(f2, 0.01, 100))
```

(3) (a) Obtenga mediante simulación el valor deteniendo cuando el semi-ancho del intervalo de confianza del 95% sea justo inferior a 0.001.

(b) Indique cuál es el número de simulaciones necesarias en la simulación realizadap ara lograr la condición pedida.

$$\textbf{(i)}\ \int_{\pi}^{2\pi} \frac{\sin x}{x}\, dx, \qquad \textbf{(ii)}\ \int_{0}^{\infty} \frac{3}{3+x^4}\, dx$$

En método de Monte Carlo, el estimador es una media muestral

$$\hat{\theta} = \frac{1}{N}\sum_{i=1}^{N} g(u_i)$$

Sabemos que cuando $N \to \infty$,

$$\frac{\hat{\theta} - \theta}{\frac{\sigma}{\sqrt{n}}} \simeq \mathcal{N}(0,1)$$

Por lo tanto,

$$P\left(-z_{\alpha/2} \leq \sqrt{n}\frac{\hat{\theta} - \theta}{\sigma} \leq z_{\alpha/2}\right) = 1 - \alpha$$

de lo cual se sigue

$$P\left(\hat{\theta} - z_{\alpha/2}\frac{\sigma}{\sqrt{n}} \leq \theta \leq \hat{\theta} + z_{\alpha/2}\frac{\sigma}{\sqrt{n}}\right) = 1 - \alpha$$

Lo que esto nos dice es que el intervalo de confianza para una estimación de Monte Carlo y un nivel de significancia α es dado por

$$\hat{\theta} \pm z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$$

Para 95% de confianza, debemos tomar $z_{0.05/2} = z_{0.025} = 1.96$. Más generalmente, la longitud del intervalo de confianza siempre es $2 \cdot z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$, es decir que depende de una sóla variable (n). Se sigue que el semi-ancho del intervalo no es más que $z_{\alpha/2}\frac{\sigma}{\sqrt{n}}$. Por lo tanto, sólo queremos simular hasta que

$$1.96\frac{S}{\sqrt{n}} < 0.001$$

Ahora simplifiquemos las integrales. Como

$$u = \frac{x-\pi}{\pi} = \frac{x}{\pi} - 1, \qquad du = \frac{dx}{\pi}, \qquad x = \pi(u+1)$$

tenemos

$$\int_{\pi}^{2\pi} \frac{\sin x}{x}\, dx = \pi \int_{0}^{1} \frac{\sin(\pi(u+1))}{\pi(u+1)}\, du \approx \frac{\pi}{N}\sum_{i=1}^{N} \psi(u_i)$$

con $\psi(u)$ la función siendo integrada. Respecto a **(i i)**, hagamos

$$u = \frac{1}{x+1}, \qquad du = -\frac{dx}{(x+1)^2}, \qquad x = \frac{1}{u} - 1$$

Por ende,

$$\int_0^\infty \frac{3}{3+x^4}dx = \int_0^1 \frac{3\left(\frac{1}{u}\right)^2}{3+(\frac{1}{u}-1)^4}\,du = \int_0^1 \frac{3}{u^2(3+(1/u-1)^4)}du$$

(b) Ya dijimos que queremos $1.96\frac{\sigma}{\sqrt{n}} < 0.001$, lo cual se cumple si y solo si $3841.6 \cdot \sigma^2 < n$

(**7**): Sean $X_1, ..., X_n$ variables aleatorias i.i.d., con media desconocida μ. Para constantes $a < b$, se quiere estimar $p = P(a < \Sigma_{i=1}^{n} X_i/n - \mu < b)$. Estimar p if $n = 10$ y los valores de X_i son

$$56,\ 101,\ 78,\ 67,\ 93,\ 87,\ 64,\ 72,\ 80,\ 69$$

Tomar $a = -5, b = 5$.

El promedio es 76.7. Con $a = 5, b = 5$:

$$\begin{aligned} p &= P(-5 - 76.7 < -\mu < 5 - 76.7) \\ &= P(76.7 - 5 < \mu < 76.7 + 5) \\ &= P(71.7 < \mu < 81.7) \end{aligned}$$

$X_1, \ldots, X_n$ i.i.d. con σ^2 desconcoida. Usando como estiamdor:

$$S^2 = \sum_{i=1}^{n} (X_i - \overline{X})^2/(n-1)$$

(a) Estimar via bootstrap con $n = 2, X_1 = 1, X_2 = 3$.
(b) Si $n = 15$ con
$5,\ 4,\ 9,\ 6,\ 21,\ 17,\ 11,\ 20,\ 7,\ 10,\ 21,\ 15,\ 13,\ 16,\ 8$

6 P7

(1) De acuerdo con la teoría genética de Mendel, cierta planta de guisantes debe producir flores blancas, rosas o rojas con probabilidad 1/4, 1/2 y 1/4, respectivamente. Para verificar experimentalmente la teoría, se estudió una muestra de 564 guisantes, donde se encontró que 141 produjeron flores blancas, 291 flores rosas y 132 flores rojas. Aproximar el p-valor de esta muestra: a) utilizando la prueba de Pearson con aproximación chi-cuadrada, b) realizando una simulación.

(a) Por ser el primer problema, iremos lento. Nuestra hipótesis nula es que la hipótesis de Mendel se sostiene, la hipótesis alternativa es que la distribución de los guisantes viola las leyes de la herencia.

Sea N_i la frecuencia observada de guisantes del color i. Sea $N = 564$ el tamaño de la muestra. Las proporciones esperadas son

$$np_i = \begin{cases} 564 \cdot \frac{1}{4} = 141 & i = \text{blancas} \\ 564 \cdot \frac{1}{2} = 282 & i = \text{rosas} \\ 564 \cdot \frac{1}{4} = 141 & i = \text{rojas} \end{cases}$$

El estadístico de prueba será en nuestra muestra de tamaño $N = 564$ será

$$\begin{aligned} T &= \sum_{i=1}^{3} \frac{(N_i - np_i)^2}{np_i} = \frac{(141-141)^2}{141} + \frac{(291-282)^2}{282} + \frac{(132-141)^2}{141} \\ &= \frac{81}{282} + \frac{81}{141} \\ &= 0.861 \end{aligned}$$

Tenemos 2 grados de libertad y por lo tanto sabemos que $T \sim \chi^2_2$. El p-valor es la probabilidad de obtener un valor del estadístico tan extremo como el observado o mayor:

$$P(\chi^2_2 \geq 0.861) = 1 - P(\chi^2_2 \leq 0.861) = 1 - 0.34982 = 0.65018$$

Bajo ningún nivel de significancia mentalmente sano rechazaremos la hipótesis nula.

(b) Usando bootstrap, simulamos N veces el experimento y calculamos cuántas veces el estadístico experimental es tan o más extremo que el estadístico que hemos observado:

```
def sim1(n_simulations):

  probs = [1/4, 1/2, 1/4]
  N = 564

  # X: Valores observados
  X = np.array([141, 291, 132])
  # E_X: Valores esperados
  E_X = np.array([141, 282, 141])
  # T: Estadístico
  T = np.sum( (X - E_X)**2 / E_X)
```

```
K = 0

for i in range(n_simulations):

  # Generamos una sample eligiendo aleatoriamente entre [0, 1, 2]
  sample = np.random.choice([0,1,2], size=N, p=probs)
  # Cuántas veces aparece cada valor posible
  frequencies = np.array( [np.sum(sample == i) for i in range(3)] )
  chi_sim = np.sum((frequencies - E_X)**2 / E_X)
  if chi_sim >= T:
      K += 1

p = K/n_simulations
return p
```

El resultado con 10000 simulaciones fue 0.654.

(a) Para verificar que cierto dado no estaba trucado, se registraron 1000 lanzamientos, resultando que el número de veces que el dado arrojó el valor i ($i = 1, 2, 3, 4, 5, 6$) fue, respectivamente, 158, 172, 164, 181, 160, 165. Aproximar el $p$valor de la prueba: "el dado es honesto" a) utilizando la prueba de Pearson con aproximación chi-cuadrada, b) realizando una simulación.

Tomemos como hipótesis nula que el dado es honesto—i.e. que la distribución de las posibles caras es uniforme con probabilidad $\frac{1}{6}$. Bajo la hipótesis nula, en 1000 lanzamientos se espera que cada cara salga $1000 \cdot 1/6 = 166.666$ veces. Por ende, el estadístico de prueba de Pearson con aproximación χ^2_5 es

$$\begin{aligned} T = \frac{1}{166.666} \sum_{i=1}^{6} & (158 - 166.666)^2 \\ & +(172 - 166.666)^2 \\ & +(164 - 166.666)^2 \\ & +(181 - 166.666)^2 \\ & +(160 - 166.666)^2 \\ & +(165 - 166.666)^2 \end{aligned}$$

lo cual da ≈ 2.18. Usando Python, computamos p-value $= P(\chi^2_5 \geq 2.18) = 0.8237$. El dado es honesto.

(b) La simulación via bootstrapping me dio 0.8177:

```
def sim2(n_simulations):

  probs = [1/6 for _ in range(6)]
  T = 2.18
  k = 0
  N = 1000
  E_X = 166.666

  for _ in range(n_simulations):

    sample = np.random.choice([0,1,2,3,4,5], size=N, p=probs)
    frequencies = np.array([np.sum(sample == i) for i in range(6)])
    sample_t = np.sum( ( frequencies - E_X )**2 / E_X )

    if sample_t >= T:
      k += 1


  return k/n_simulations
```

(3) Calcular una aproximación del $p$valor de la hipótesis: "Los siguientes 10 números son aleatorios":

$$0.12,\ 0.18,\ 0.06,\ 0.33,\ 0.72,\ 0.83,\ 0.36,\ 0.27,\ 0.77,\ 0.74$$

Queremos testear si los datos provienen de una uniforme $\mathcal{U}(0,1)$. Tenemos dos maneras, y vamos a hacer ambas para practicar.

(a) Discretizamos $[0,1]$ en $[0,0.1), [0.1,0.2), \ldots, [0.9,1]$ donde, asumiendo que los datos provienen de $\mathcal{U}(0,1)$, tenemos $P(u_i) \in I_j = 1/10$. Claramente, esperamos que de 10 datos, 1 caiga en cada intervalo. Es decir, el estadístico es

$$T = \sum_{j=1}^{10} \frac{(\#j - 1)^2}{1} = \sum_{j=1}^{10} (\#j - 1)^2 \sim \chi^2_9$$

donde $\#j$ es la cantidad de valores que cayeron en el intervalo j-ésimo. Claramente,

$$\begin{aligned} T &= (1-1)^2 + (2-1)^2 + (1-1)^2 + (2-1)^2 + (0-1)^2 + (0-1)^2 + (0-1)^2 + (3-1)^2 + (1-1)^2 + (0-1)^2 \\ &= 1 + 1 + 1 + 1 + 1 + 2^2 + 1 \\ &= 10 \end{aligned}$$

El p-valor de este estadístico es ≈ 0.35 y no se rechaza la hipótesis de que los datos son aleatorios y uniformes en $[0,1]$.

(b) Usamos el test de Kolmogorov-Smirnov para ver si los datos vienen de una uniforme $\mathcal{U}(0,1)$. Usamos F para denotar la FPA de dicha distribución. El primer paso es ordenar los valores:

$$0.06,\ 0.12,\ 0.18,\ 0.27,\ 0.33,\ 0.36,\ 0.72,\ 0.74,\ 0.77,\ 0.83$$

Usaremos y_j para denotar el j-écimo elemento del ordenamiento. El estadístico de prueba es

$$\begin{aligned} D &= \max_{1 \le j \le n} \left\{ \frac{j}{n} - F(y_j), F(y_j) - \frac{j-1}{n} \right\} \\ &= \max_{1 \le j \le n} \left\{ \frac{j}{10} - y_j, y_j - \frac{j-1}{10} \right\} \end{aligned}$$

Es una tontera calcular esto en Python y $D = 0.24$ para nuestro conjunto de datos:

```
def ks_statistic(F, sample):

  sample = sorted(sample)
  n = len(sample)

  D = 0
  for j in range(1, n+1):
    y = sample[j-1]
    d1 = j/n - F(y)
```

```
    d2 = F(y) - (j-1)/n
    d = max(d1, d2)
    D = max(D, d)

  return(D)


sample = [0.06, 0.12, 0.18, 0.27, 0.33, 0.36, 0.72, 0.74, 0.77, 0.83]
def uniform_cdf(x):
  if x > 1 or x < 0:
    raise ValueError
  return x

ks_statistic(uniform_cdf, sample)
```

Pero continuemos. Debemos calcular el p-valor de dicho estadístico vía simulaciones. Actualicemos el código para ello:

```
def ks_statistic(F, sample):

  sample = sorted(sample)
  n = len(sample)

  D = 0
  for j in range(1, n+1):
    y = sample[j-1]
    d1 = j/n - F(y)
    d2 = F(y) - (j-1)/n
    d = max(d1, d2)
    D = max(D, d)

  return D


def ks(F, sample, sim_sample_size=10, n_sims=1000):

  D = ks_statistic(F, sample)
  print(D)
  k = 0

  for _ in range(n_sims):
    sim_sample = np.random.uniform(0, 1, size=sim_sample_size)
    sim_D = ks_statistic(F, sim_sample)
    if sim_D >= D:
      k += 1

  return k/n_sims
```

```
sample = [0.06, 0.12, 0.18, 0.27, 0.33, 0.36, 0.72, 0.74, 0.77, 0.83]
def uniform_cdf(x):
  if x > 1 or x < 0:
    raise ValueError
  return x

p_value = ks(uniform_cdf, sample)
```

El p-valor simulado fue 0.541 con lo cual no se rechaza la hipótesis nula. (Nuestros resultados coinciden con los de la librería scipy).

(4) Calcular una aproximación del *p*valor de la hipótesis: "Los siguientes 13 valores provienen de una distribución exponencial con media 50":

86.0, 133.0, 75.0, 22.0, 11.0, 144.0, 78.0, 122.0, 8.0, 146.0, 33.0, 41.0, 99.0

Podemos generalizar nuestra función **ks** del ejercicio **(3)** para que sirva para cualquier distribución.

```
def ks(F, sample, sampler, sim_sample_size=None, n_sims=1000):
    """
    Compute the empirical p-value of the KS statistic by simulating samples from
    the null distribution.

    Parameters:
        F: Callable
            CDF of the null hypothesis distribution.
        sample: list or array
            The observed data.
        sampler: Callable
            A function that generates samples from the null distribution.
            Should take a single ‘size‘ argument.
        sim_sample_size: int, optional
            Size of the simulated samples. If None, use len(sample).
        n_sims: int
            Number of simulations to perform.

    Returns:
        float
            Empirical p-value.
    """
    if sim_sample_size is None:
        sim_sample_size = len(sample)

    D = ks_statistic(F, sample)
    print("D-statistic: ", D)

    k = 0
    for _ in range(n_sims):
        sim_sample = sampler(sim_sample_size)
        sim_D = ks_statistic(F, sim_sample)
        if sim_D >= D:
            k += 1

    return k / n_sims
```

Ahora ya tenemos nuestro código para calcular el test de Kolmogorov-Smirnov con cualquier distribución. Solo basta notar que la prob. acumulada de una exponencial con media 50 es

$$F_X(x) = 1 - e^{-x/50}$$

Por lo tanto, hacemos:

```
def exponential_cdf(x, mean):
  if x < 0:
    raise ValueError

  return 1 - exp(-x/mean)

def exponential_mean_50(x):
  return exponential_cdf(x, 50)

def exponential_sampler(size):
    return np.random.exponential(scale=50, size=size)


sample = [86.0, 133.0, 75.0, 22.0, 11.0, 144.0, 78.0, 122.0, 8.0, 146.0, 33.0,
          41.0, 99.0]
p_value = ks(exponential_mean_50, sample, exponential_sampler)
```

El resultado es un p-valor de 0.023 y un estadístico $D \approx 0.392$. Se rechaza la hipótesis nula ($\alpha = 0.05$).

(5) Calcular una aproximación del p-valor de la prueba de que los siguientes datos corresponden a una distribución binomial con parámetros $(n = 8,\ p)$, donde p no se conoce:

$$6,\ 7,\ 3,\ 4,\ 7,\ 3,\ 7,\ 2,\ 6,\ 3,\ 7,\ 8,\ 2,\ 1,\ 3,\ 5,\ 8,\ 7$$

Un estimador insesgado y consistente de p es

$$\frac{\#\text{éxitos}}{\#\text{experimentos}}$$

y se puede estimar a través de las distintas muestras si consideramos la suma de los experimentos binomiales. (Suma de binomiales sigue una binomial). Por ende, veamos que tenemos $6 + 7 + 3 + 4 + \ldots + 8 + 7 = 89$ éxitos de un total de $8 \cdot 18 = 144$ experimentos, lo cual da

$$p \approx \frac{89}{144} = 0.618$$

Por lo tanto, haremos la prueba de hipótesis usando el test de Pearson con aproximación χ^2 (pues los datos son discretos). Ahora haremos un código general que permite hacerlo:

```
def pearson_chi(pmf, sample, imX, simulate_p_value=False, n_simulations=1000):
    """
    Bondad de ajuste: Pearson chi-cuadrado.

        pmf: callable
            Función de prob. de masa bajo H
        sample: list or array
            Datos observados (frecuencia absoluta).
        imX: list
            Soporte de la distribución (su imagen).
        simulate_p_value: bool
            Simular o usar la CDF de ²?
        n_simulations: int
            Si se simula el p-valor, cuántas simulaciones?

    Returns:
        T: float
           El estadístico T
        p: float
            El p-valor
    """
    N = len(sample)
    counts = Counter(sample)
    observedFreq = np.array([counts.get(x, 0) for x in imX])
    probs = np.array([pmf(x) for x in imX])
    expectedFreq = probs * N

    # Usamos el ‘with‘ para cuidarnos de la división por cero
    with np.errstate(divide='ignore', invalid='ignore'):
        T = np.sum((observedFreq - expectedFreq)**2 / expectedFreq)

    if simulate_p_value:
        K = 0
        for _ in range(n_simulations):
            simulated_sample = np.random.choice(imX, size=N, p=probs)
            sim_counts = Counter(simulated_sample)
            sim_observedFreq = np.array([sim_counts.get(x, 0) for x in imX])
            chi_sim = np.sum((sim_observedFreq - expectedFreq)**2 / expectedFreq)
            if chi_sim >= T:
                K += 1
        p = K / n_simulations
        return T, p
    else:
        df = len(counts) - 1  # grados de libertad
        print("DF ", df)
        p = 1 - chi2.cdf(T, df)
        return T, p
```

Ahora solo hacemos:

```
p = 0.618
n = 8
sample = [6, 7, 3, 4, 7, 3, 7, 2, 6, 3, 7, 8, 2, 1, 3, 5, 8, 7]
t, p_value = pearson_chi(
  lambda x: binom.pmf(x, n, p), # de scipy.stats
  sample,
  [0, 1, 2, 3, 4, 5, 6, 7, 8],
  simulate_p_value=True,
  n_simulations=10000
                           )

print(t)
print(p_value)
```

El resultado es: T: 31.49620198292496, p-valor: 0.0137

(6) Un escribano debe validar un juego en cierto programa de televisión. El mismo consiste en hacer girar una rueda y obtener un premio según el sector de la rueda que coincida con una aguja. Hay 10 premios posibles, y las áreas de la rueda para los distintos premios, numerados del 1 al 10, son respectivamente:

$$31\%,\ 22\%,\ 12\%,\ 10\%,\ 8\%,\ 6\%,\ 4\%,\ 4\%,\ 2\%,\ 1\%$$

Los premios con número alto (e.j. un auto 0Km) son mejores que los premios con número bajo (e.j. 2x1 para entradas en el cine). El escribano hace girar la rueda hasta que se cansa, y anota cuántas veces sale cada sector. Los resultados, para los premios del 1 al 10, respectivamente, son:

$$188,\ 138,\ 87,\ 65,\ 48,\ 32,\ 30,\ 34,\ 13,\ 2$$

a) Construya una tabla con los datos disponibles

b) Diseñe una prueba de hipótesis para determinar si la rueda es justa

c) Defina el p-valor a partir de la hipótesis nula

d) Calcule el p-valor bajo la hipótesis de que la rueda es justa, usando la aproximación chi cuadrado

e) Calcule el p-valor bajo la hipótesis de que la rueda es justa, usando una simulación.

(a) Hagamos una tabla con las probabilidades de cada valor, su frecuencia esperada, su frecuencia observada. Veamos que

$$188 + 138 + 87 + 65 + 48 + 32 + 30 + 34 + 13 + 2 = 637$$

Es decir, hubieron $N = 637$ muestras.

X	$P(X)$	**Frec. esperada** $(N \cdot P(X))$	**Frec. observada**
1	0.31	197.4	188
2	0.22	140.14	138
3	0.12	76.44	87
4	0.1	63.7	65
5	0.08	50.96	48
6	0.06	38.22	32
7	0.04	25.48	30
8	0.04	25.48	34
9	0.02	12.74	13
10	0.01	6.37	2

(b, c, d) La hipótesis nula es que la rueda es que la rueda es justa y tiene la distribución dada en el ejercicio. Solo hay que computar:

$$T = \sum_{i=1}^{10} \frac{(i_{\text{Frec obs.}} - i_{\text{Frec. esperada}})^2}{i_{\text{Frec. esperada}}} \simeq \chi^2_9$$

El p-valor bajo la hipótesis nula es $P(\chi^2_9 \geq T)$. El código es una tontera, sigue la misma estructura de siempre:

```
def sim6(n_sim=10000):
  obs = np.array([188, 138, 87, 65, 48, 32, 30, 34, 13, 2])
  esp = np.array([197.4, 140.14, 76.44, 63.7, 50.96, 38.22, 25.48, 25.48, 12.74,
                  6.37])
  N = 637

  T = np.sum ( (obs - esp)**2 / esp )
  df = 9
  p_value = 1 - chi2.cdf(T, df)

  probs = [0.31, 0.22, 0.12, 0.1, 0.08, 0.06, 0.04, 0.04, 0.02, 0.01]
  k = 0

  for _ in range(n_sim):
    sample = np.random.choice([0,1,2,3,4,5, 6, 7, 8, 9], size=N, p=probs)
    frequencies = np.array([np.sum(sample == i) for i in range(10)])
    sample_t = np.sum( ( frequencies - esp )**2 / esp )

    if sample_t >= T:
      k += 1

  sim_p_value = k/n_sim

  print(f"T = {T}; p = {p_value}; sim_p = {sim_p_value}")
```

El resultado: `T = 9.803840445269017; p = 0.36659773905255266; sim_p = 0.3689`. No rechazamos la hipótesis de que la rueda es justa.

7 P8

(1) La Figura 1 muestra el diagrama de transición de una cadena de Markov. Dar la matriz de transición, y determinar:
a) $P(X_4 = 2|X_3 = 1) y P(X_3 = 1|X_2 = 1)$.
b) $P(X_4 = 2|X_1 = 1) y P(X_4 = 2|X_0 = 1)$.
c) Si se sabe que $P(X_0 = 0) = 1/3$, dar $P(X_0 = 0, X_1 = 1)$ y $P(X_0 = 0, X_1 = 1, _{X\,2} = 2)$

La matriz de transición es

$$Q = \begin{pmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & 0 & 0.5 \\ 0.5 & 0 & 0.5 \end{pmatrix}$$

(a) $P(X_4 = 2 \mid X_3 = 1) = p_{21} = 0$. $P(X_3 = 1 \mid X_2 = 1) = p_{11} = 0$.

(b) Consideremos $P(X_4 = 2 \mid X_1 = 1)$. Se quiere la posibilidad de llegar desde el estado 2 al estado 1 en 3 transiciones. Recordemos que

$$AB_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$$

En nuestro caso, para Q^2:

$$p_{00}^2 = 0.25^2 + 0.5^2 + 0.25 * 0.5 = 0.4375$$
$$p_{10}^2 = 0.5 * 0.25 + 0 * 0.5^2 = 0.375$$
$$p_{20}^2 = 0.5 * 0.25 + 0.5^2 = 0.375$$

$$p_{01} = 0.5 * 0.25 = 0.125$$
$$p_{11} = 0.5^2 = 0.25$$
$$p_{21} = 0.5^2 = 0.25$$

$$p_{02} = 0.25^2 + 0.5^2 + 0.5 * 0.25 = 0.4375$$
$$p_{12} = 0.5 * 0.25 + 0.5^2 = 0.375$$
$$p_{22} = 0.5 * 0.25 + 0.5^2 = 0.375$$

$$Q^2 = \begin{pmatrix} 0.4375 & 0.125 & 0.4375 \\ 0.375 & 0.25 & 0.375 \\ 0.375 & 0.25 & 0.375 \end{pmatrix}$$

Como no quiero ser infeliz, a partir de ahora computo las matrices con Python. Según numpy,

$$Q^3 = \begin{pmatrix} 0.390625 & 0.21875 & 0.390625 \\ 0.40625 & 0.1875 & 0.40625 \\ 0.40625 & 0.1875 & 0.40625 \end{pmatrix}$$

Con esta matriz, ya sabemos que pasar del estado 2 al 1 en 3 transiciones es $p_{21}^{(3)} = 0.1875$.

(c) Digamos que $P(X_0 = 0) = 1/3$. Entonces

$$P(X_0 = 0, X_1 = 1) = \frac{1}{3} \cdot p_{01} = 1/3 \cdot 0.5 \approx 0.1666$$

$$P(X_0 = 0, X_1 = 1, X_2 = 2) = 1/3 \cdot 0.5 \cdot 0.5 \approx 0.0833$$

Considere la cadena de Markov con

$$Q = \begin{bmatrix} 0.5 & 0.5 & 0.5 \\ 0.5 & 1/3 & 2/3 \\ 0.5 & 0.5 & 2/3 \end{bmatrix}$$

(a) Calcule la probabilidad de alcanzar el estado j dado que $X_0 = 0$, $j = 0, 1, 2$.

(a) Calculemos el vector de probabilidades de alcance $h^{\{0\}}$. Sabemos que dicho vector es definido por la menor solución no negativa de:

$$h_i^{\{0\}} = \begin{cases} 1 & i = 0 \\ \sum_{j \in S} p_{ij} h_j^{\{0\}} & i \neq 0 \end{cases}$$

Sabemos entonces que $h_0^{\{0\}} = 1$. Veamos que

$$\begin{cases} h_1^0 &= p_{10}h_0^0 + p_{11}h_1^0 + p_{12}h_2^0 \\ h_2^0 &= p_{20}h_0^0 + p_{21}h_1^0 + p_{22}h_2^0 \end{cases}$$
$$\rightarrow \begin{cases} h_1^0 &= \frac{1}{2}(1) + \frac{1}{3}h_1^0 + \frac{2}{3}h_2^0 \\ h_2^0 &= \frac{1}{2}(1) + \frac{1}{2}h_1^0 + \frac{2}{3}h_2^0 \end{cases}$$
$$\rightarrow \begin{cases} h_1^0 &= \frac{1}{2} + \frac{1}{3}h_1^0 + \frac{2}{3}h_2^0 \\ h_2^0 &= \frac{1}{2} + \frac{1}{2}h_1^0 + \frac{2}{3}h_2^0 \end{cases}$$

Ahora bien, la ecuación (1) nos da

$$\frac{2}{3}h_1^0 = \frac{1}{2} + \frac{2}{3}h_2^0$$
$$\iff h_1^0 = \frac{3}{4} + h_2^0$$

Sustituyendo en (2),

$$
\begin{aligned}
h_2^0 &= \frac{1}{2} + \frac{1}{2}\left(\frac{3}{4} + h_2^0\right) + \frac{2}{3}h_2^0 \\
\iff h_2^0 &= \frac{1}{2} + \frac{3}{8} + \left(\frac{1}{2} + \frac{2}{3}\right)h_2^0 \\
\iff h_2^0 &= \frac{7}{8} + \frac{5}{6}h_2^0 \\
\iff \frac{1}{6}h_2^0 &= \frac{7}{8} \\
\iff h_2^0 &= \frac{42}{8} = \frac{21}{4}
\end{aligned}
$$