

Cadenas de Markov

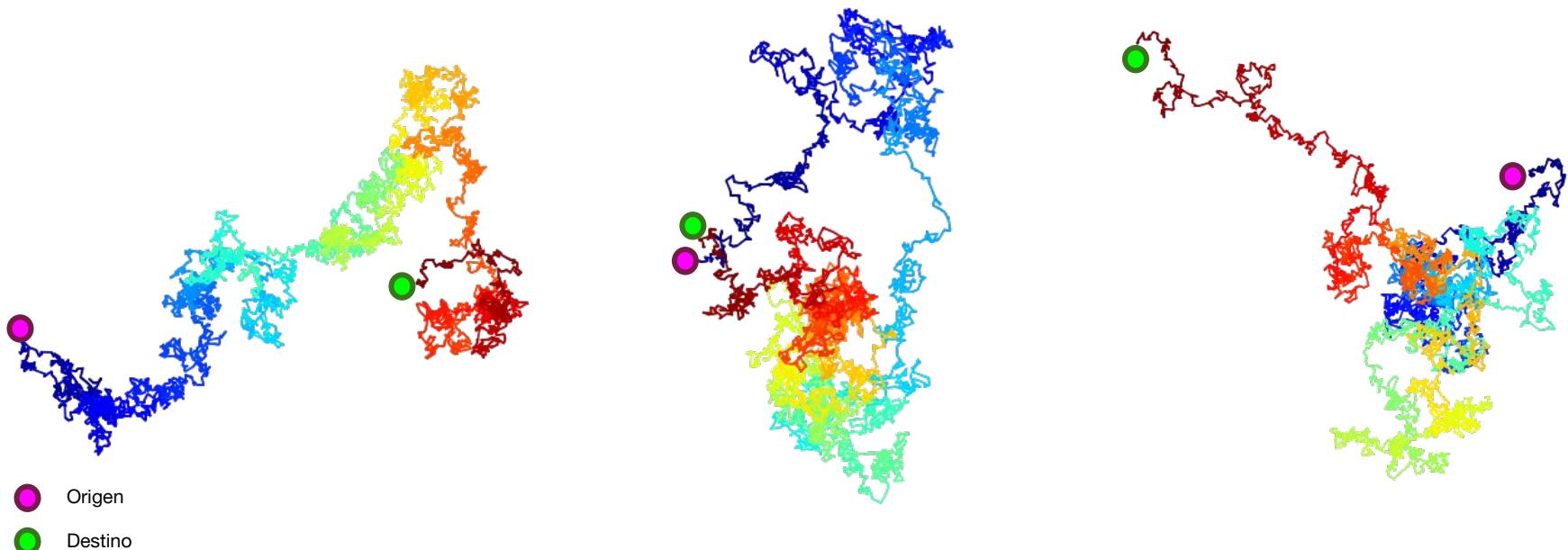
Clase 03

Investigación Operativa UTN FRBA
Curso: I4051
Docente: Martín Palazzo

Agenda clase 03

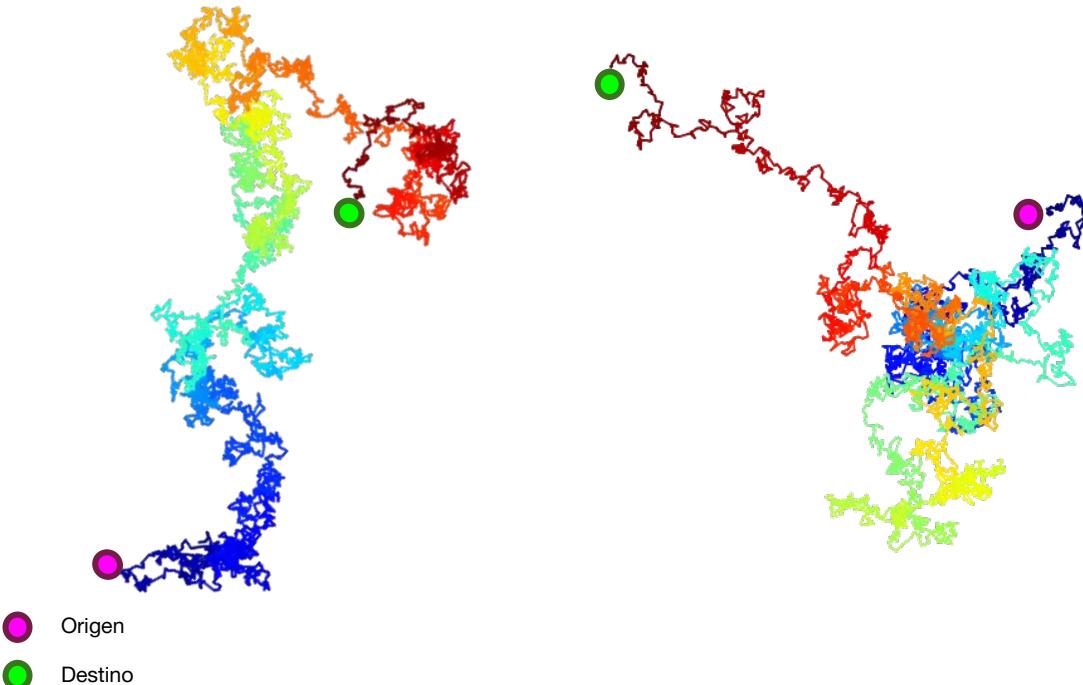
- Intro teórica a Cadenas de Markov
 - Probabilidad de transición
 - Chapman-Kolmogorov
 - Vector de probabilidad de estado
 - Estado estacionario
- Ejercicios Cadenas de Markov
- Implementación en Python de Cadenas de Markov

Procesos estocásticos



Movimiento browniano de 5000 pasos en tres experimentos independientes. El movimiento browniano modela una caminata aleatoria de una partícula en tiempo continuo, donde una partícula evoluciona en el espacio dando pasos aleatorios independientes en todas las direcciones. Podemos pensar a la posición de la partícula como el estado de nuestro sistema que será se modificará en cada instante de tiempo de manera aleatoria bajo cierta distribución de probabilidad [1].

Procesos estocásticos: movimiento browniano



Movimiento browniano de 5000 pasos en tres experimentos independientes. El movimiento browniano modela una caminata aleatoria de una partícula en tiempo continuo, donde una partícula evoluciona en el espacio dando pasos aleatorios independientes en todas las direcciones. Podemos pensar a la posición de la partícula como el estado de nuestro sistema que se modificará en cada instante de tiempo de manera aleatoria bajo cierta distribución de probabilidad [1]. [1] <https://ipython-books.github.io/133-simulating-a-brownian-motion/>

```
# Importamos librerías
import numpy as np
import matplotlib.pyplot as plt

# definimos la longitud del recorrido
n = 5000

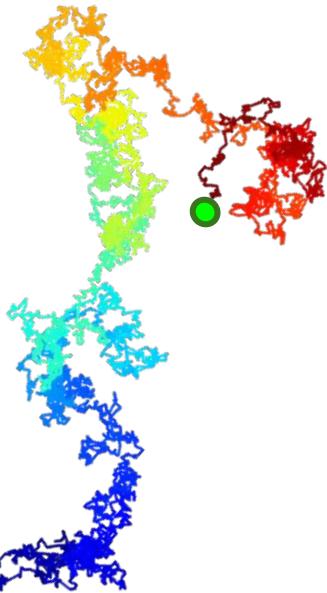
# se muestra aleatoriamente una función uniforme para dos dimensiones
# y se acumulan los resultados
x = np.cumsum(np.random.randn(n))
y = np.cumsum(np.random.randn(n))

# se agregan 10 puntos intermedios entre dos estados sucesivos
# y luego se interpolan las dimensiones x e y.
k = 10
x2 = np.interp(np.arange(n * k), np.arange(n) * k, x)
y2 = np.interp(np.arange(n * k), np.arange(n) * k, y)

# se agregan 10 puntos intermedios entre dos estados sucesivos
# y luego se interpolan las dimensiones x e y.
k = 10
x2 = np.interp(np.arange(n * k), np.arange(n) * k, x)
y2 = np.interp(np.arange(n * k), np.arange(n) * k, y)

# visualizamos el movimiento browniano con gradiente de colores.
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
ax.scatter(x2, y2, c=range(n * k), linewidths=0,
           marker='o', s=3, cmap=plt.cm.jet,)
ax.axis('equal')
ax.set_axis_off()
```

Breakout rooms: cada estudiante simula su propio movimiento browniano y comparte la visualización



Origen

Destino

```
● ● ●

# Importamos librerias
import numpy as np
import matplotlib.pyplot as plt

# definimos la longitud del recorrido
n = 5000

# se muestrea aleatoriamente una funcion uniforme para dos dimensiones
# y se acumulan los resultados
x = np.cumsum(np.random.randn(n))
y = np.cumsum(np.random.randn(n))

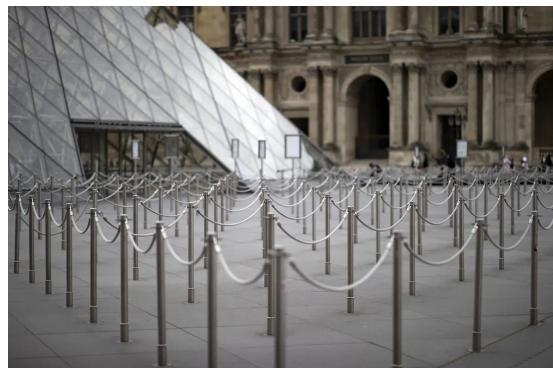
# se agregan 10 puntos intermedios entre dos estados sucesivos
# y luego se interpolan las dimensiones x e y.
k = 10
x2 = np.interp(np.arange(n * k), np.arange(n) * k, x)
y2 = np.interp(np.arange(n * k), np.arange(n) * k, y)

# se agregan 10 puntos intermedios entre dos estados sucesivos
# y luego se interpolan las dimensiones x e y.
k = 10
x2 = np.interp(np.arange(n * k), np.arange(n) * k, x)
y2 = np.interp(np.arange(n * k), np.arange(n) * k, y)

# visualizamos el movimiento browniano con gradiente de colores.
fig, ax = plt.subplots(1, 1, figsize=(8, 8))
ax.scatter(x2, y2, c=range(n * k), linewidths=0,
           marker='o', s=3, cmap=plt.cm.jet)
ax.axis('equal')
ax.set_axis_off()
```

Ejecutar el código y compartir la visualización obtenida. Comparar la visualización obtenida de cada estudiante.

Procesos estocásticos: filas de espera



$t = 0$



$t = 1$



$t = 2$

Una **Fila de Espera** puede considerarse un proceso estocástico ya que a medida que transcurre el tiempo el sistema presenta filas de distinta longitud. En este caso los estados del sistema están representados por la cantidad de clientes en la fila. A medida que transcurre el tiempo el sistema tomará distintos estados (distintas longitudes de la fila) determinado por cierta distribución de probabilidad.

Procesos estocásticos



Prob
esta
dos

$t = 0$

$$\begin{aligned} P(x = 0 \mid t = 0) \\ P(x = 1 \mid t = 0) \\ P(x = 2 \mid t = 0) \\ \dots \\ P(x = i \mid t = 0) \\ \dots \\ P(x = n \mid t = 0) \end{aligned}$$

$t = 1$

$$\begin{aligned} P(x = 0 \mid t = 1) \\ P(x = 1 \mid t = 1) \\ P(x = 2 \mid t = 1) \\ \dots \\ P(x = i \mid t = 1) \\ \dots \\ P(x = n \mid t = 1) \end{aligned}$$

$t = 2$

$$\begin{aligned} P(x = 0 \mid t = 2) \\ P(x = 1 \mid t = 2) \\ P(x = 2 \mid t = 2) \\ \dots \\ P(x = i \mid t = 2) \\ \dots \\ P(x = n \mid t = 2) \end{aligned}$$

El sistema se modela compuesto por **variables aleatorias** $x = f(t)$ que tomarán distintos valores a medida que evoluciona el parámetro (tiempo). Para cada estado posible del sistema, dado un instante de tiempo "t" existirá una probabilidad asociada $P(x \mid t)$.

Procesos estocásticos



Proba
estados

t = 0

$$\begin{aligned} P(x = 0 \mid t = 0) \\ P(x = 1 \mid t = 0) \\ P(x = 2 \mid t = 0) \\ \dots \\ P(x = i \mid t = 0) \\ \dots \\ P(x = n \mid t = 0) \end{aligned}$$

t = 1

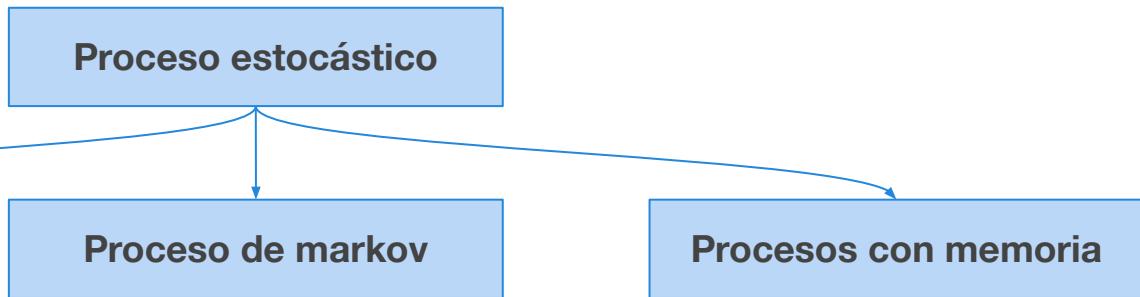
$$\begin{aligned} P(x = 0 \mid t = 1) \\ P(x = 1 \mid t = 1) \\ P(x = 2 \mid t = 1) \\ \dots \\ P(x = i \mid t = 1) \\ \dots \\ P(x = n \mid t = 1) \end{aligned}$$

t = 2

$$\begin{aligned} P(x = 0 \mid t = 2) \\ P(x = 1 \mid t = 2) \\ P(x = 2 \mid t = 2) \\ \dots \\ P(x = i \mid t = 2) \\ \dots \\ P(x = n \mid t = 2) \end{aligned}$$

Estado X = Cantidad de vehículos esperando en el semáforo.

Procesos estocásticos: categorización según memoria



Proceso aleatorio puro

El estado del sistema en cada tiempo t es independiente de cualquier estado pasado. La probabilidad de estado se determinará por una variable aleatoria.



Proceso de markov

El estado del sistema en cada tiempo t es determinado directamente por el estado del sistema en el tiempo $t-1$. Es un proceso con memoria = 1 .

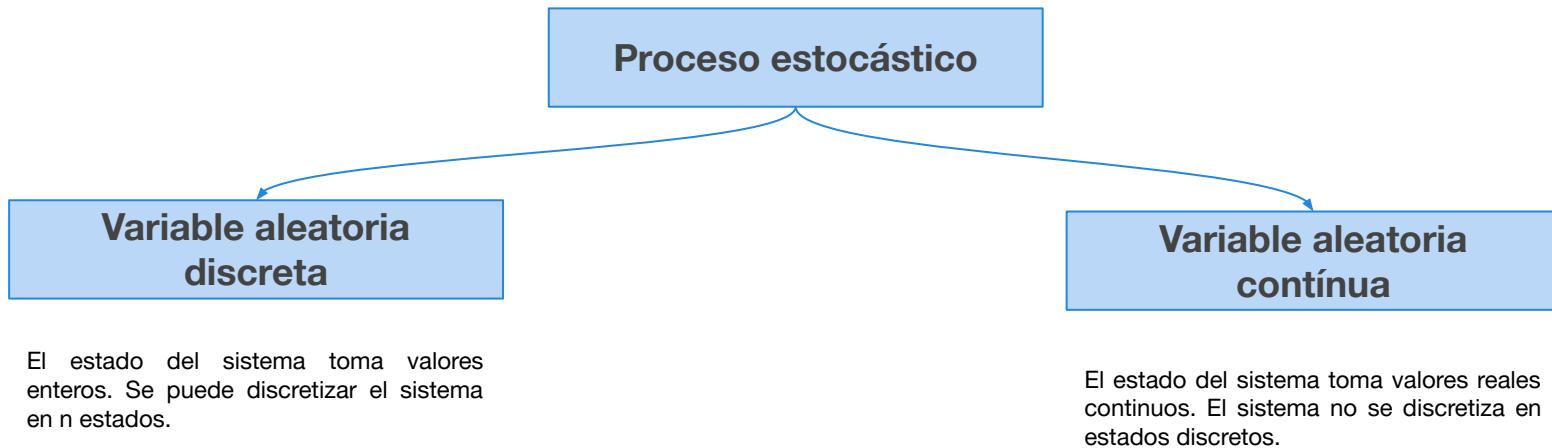


Procesos con memoria

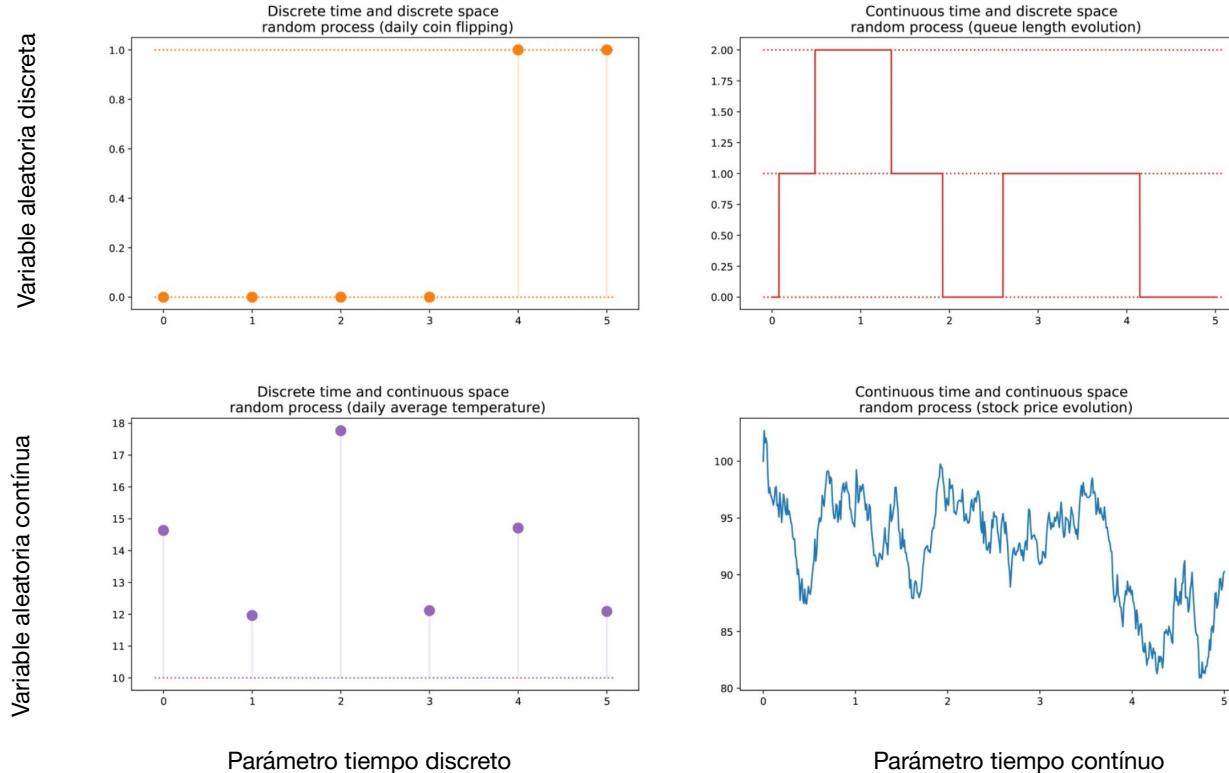
El estado del sistema en cada tiempo t es determinado por los estados que tomo el sistema en un período de tiempo $\Delta t = m$ con $m > 1$.



Procesos estocásticos: categorización según variable aleatoria



Procesos estocásticos: categorías por V.A. y parámetro



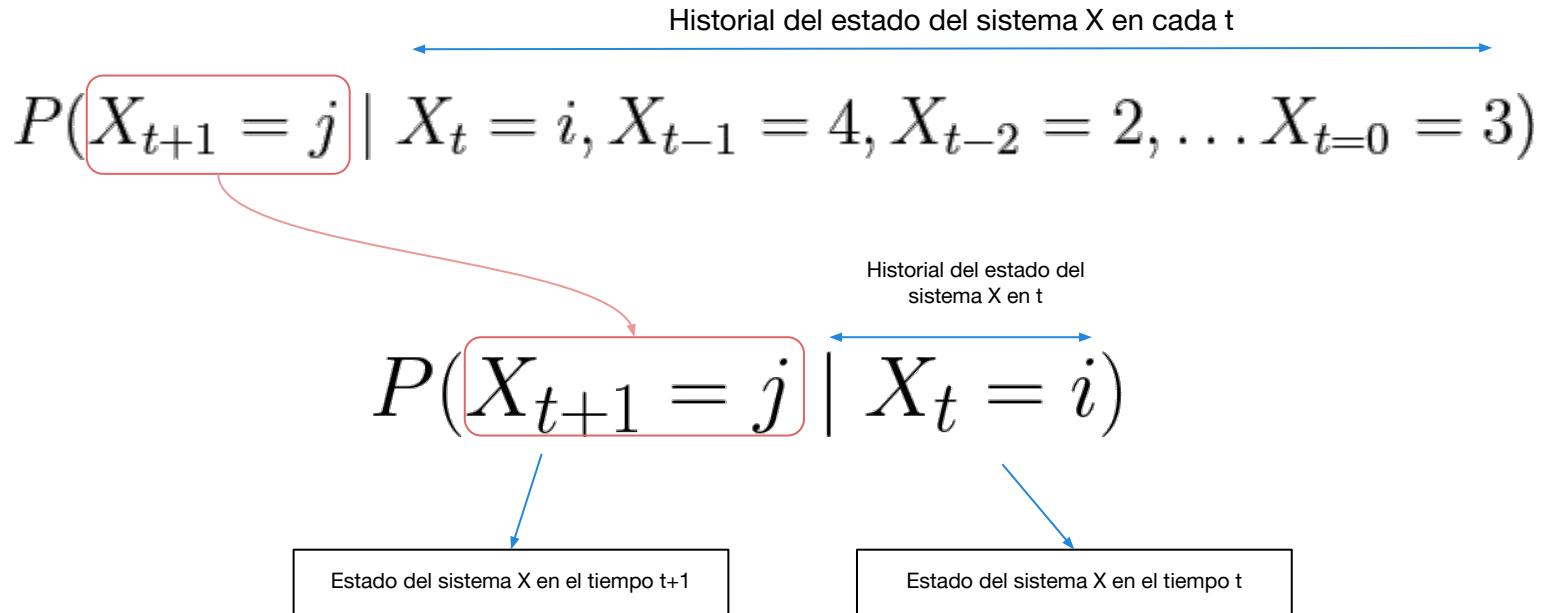
Cadenas de Markov

Cadenas de Markov

Las cadenas de Markov corresponden a una rama de los Procesos Estocásticos. Las mismas se caracterizan por definir sus estados actuales únicamente partiendo del estado inmediato anterior. **Es decir que las cadenas de Markov son procesos estocásticos con memoria t=-1.** Visto de otra manera, si conozco el estado X actual del sistema en el instante t, en caso de ser “markoviano” puedo estimar la probabilidad **condicional** de estado en el siguiente intervalo de tiempo t+1.

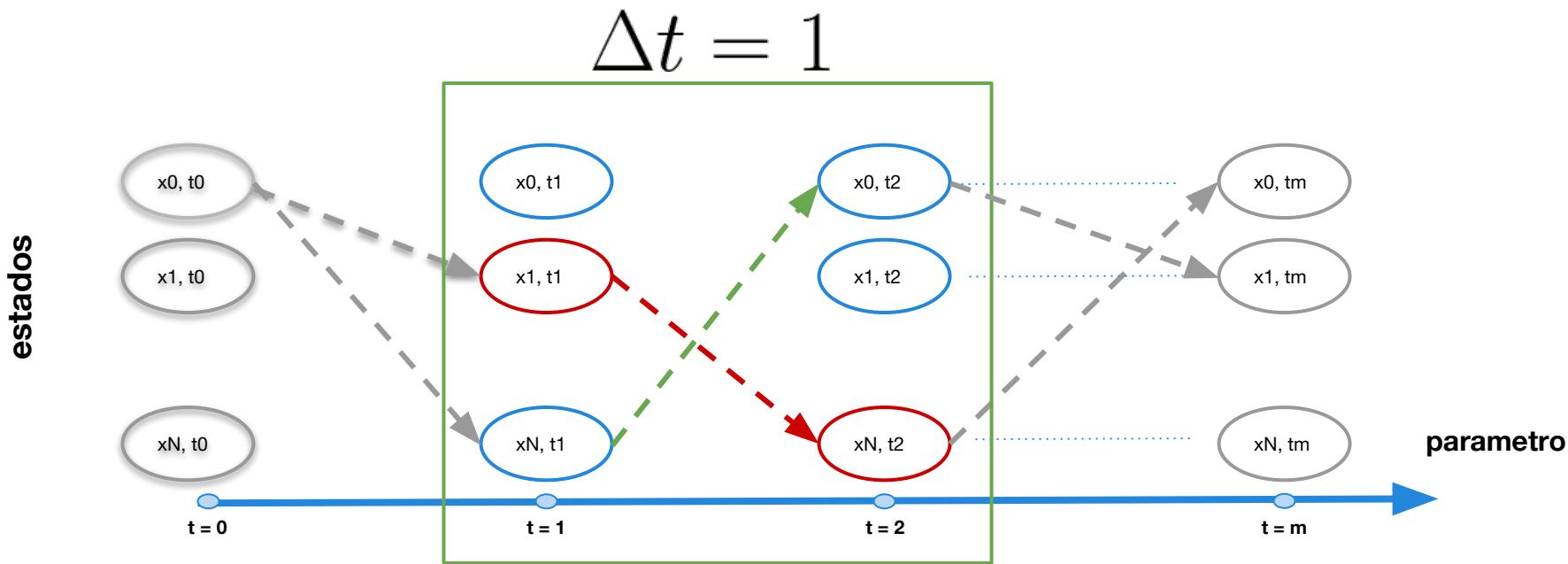
$$P(X_{t=i+1} \mid X_{t=i})$$

Cadenas de Markov



Las cadenas de Markov se caracterizan por definir sus estados actuales únicamente partiendo del estado inmediato anterior. **Es decir que las cadenas de Markov son procesos estocásticos con memoria t=-1**. Visto de otra manera, si conozco el estado X actual del sistema en el instante t, en caso de ser “markoviano” puedo conocer la probabilidad **condicional** de estado en el siguiente intervalo de tiempo t+1.

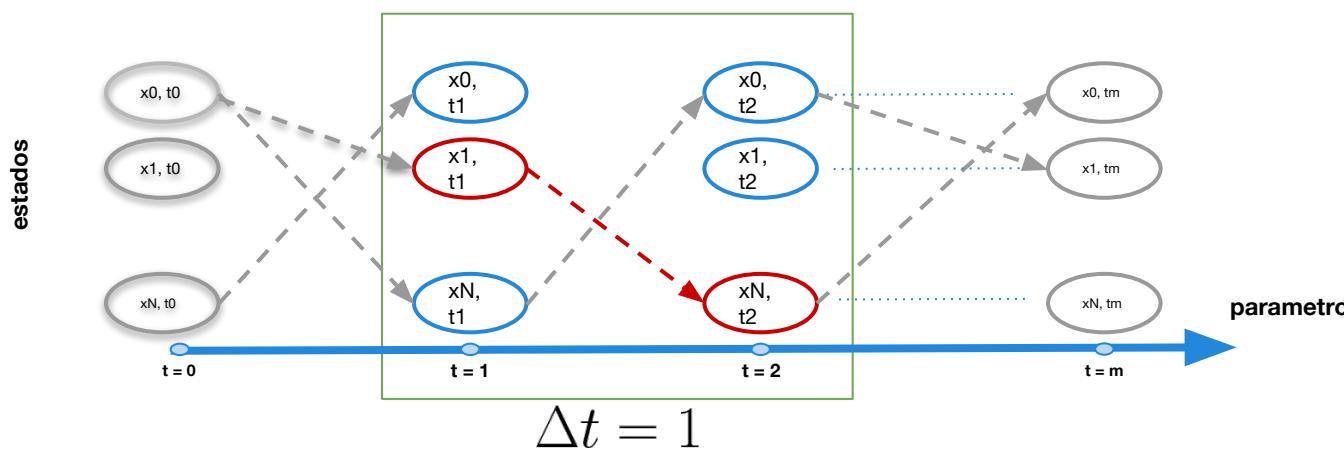
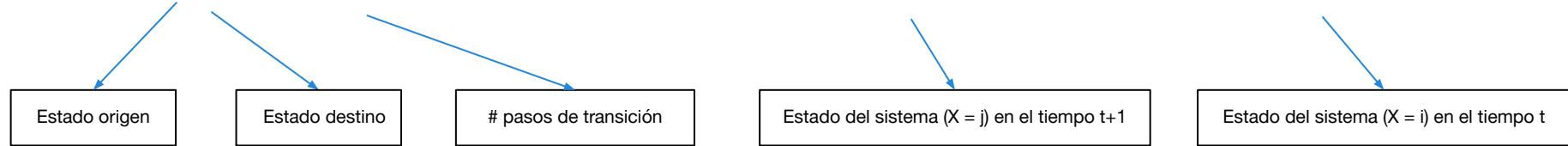
Cadenas de Markov



El concepto Markoviano nace de comprender la **transición de un solo paso**. La probabilidad de que en el tiempo t mi sistema este en cualquiera de los estados posibles depende únicamente del estado del sistema en el instante inmediato anterior $t-1$.

Probabilidad condicional de Transición de 1 paso

$$p_{ij}(\Delta t = 1) = P(X_{t+1} = j \mid X_t = i)$$

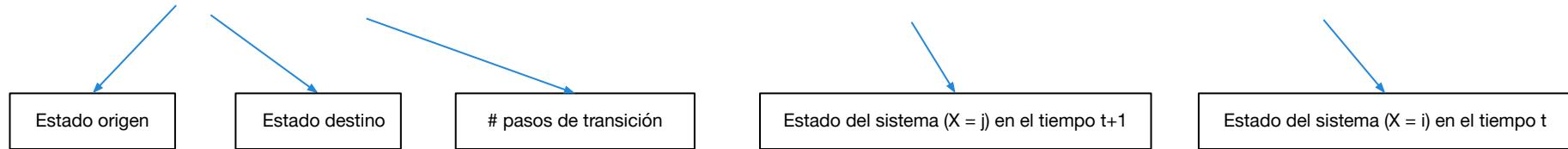


Dados N estados, la probabilidad de transición de un paso es aquella **probabilidad condicional** que indica la probabilidad de estar en el estado “ j ” si en el paso anterior el sistema se encontraba en el estado “ i ”.

Probabilidad condicional de Transición de m pasos

Dados N estados, la probabilidad de transición de un paso es aquella que indica la probabilidad de estar en el estado “j” si en el paso anterior el sistema se encontraba en el estado “i”.

$$p_{ij}(\Delta t = 1) = P(X_{t+1} = j \mid X_t = i)$$



Si conozco las probabilidades de transición de un paso de un sistema (entre todos los pares de estados) y supongo que las mismas se mantienen constantes en el tiempo, entonces puedo calcular las probabilidades de transición luego de M pasos.

$$p_{ij}(\Delta t = m) = P(X_{t+m} = j \mid X_t = i)$$

Matriz de Transición de 1 paso

Filas = Estado en tiempo t →

Columnas = Estado en tiempo t+1 ↓

"Probabilidad de que mi sistema estando el estado 1 en el tiempo t, se encuentre en el estado 3 en el tiempo t+1"

$$P(\Delta t = 1) = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \dots & P_{1n} \\ P_{21} & P_{22} & P_{32} & \dots & P_{2n} \\ P_{31} & P_{32} & P_{33} & \dots & P_{3n} \\ \dots & \dots & \dots & P_{ij} & \dots \\ P_{n1} & \dots & \dots & \dots & P_{nn} \end{bmatrix}$$

$0 \leq P_{ij} \leq 1$

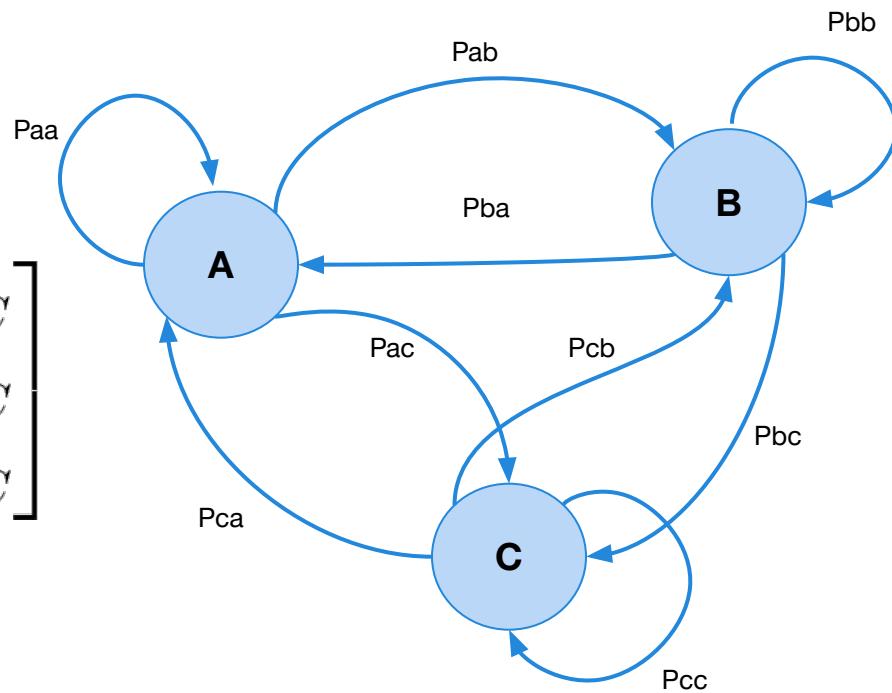
$\sum_{j=1}^n P_{ij} = 1$

Para poder modelar todas las probabilidades de transición de 1 paso en un sistema markoviano de N estados utilizamos la Matriz de Transición de 1 paso de (nxn). Por ley de probabilidad, la suma de todas las probas que salgan de un estado deben sumar 1 (suma de probas por fila). Los **renglones/filas** representan los nodos/estados de origen mientras que las **columnas** los nodos/estados destino.

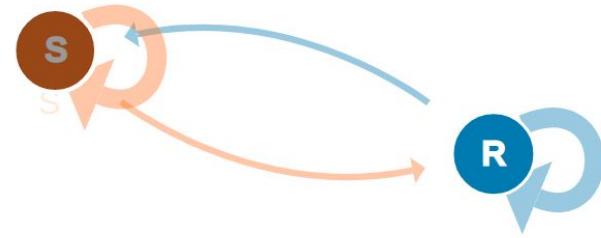
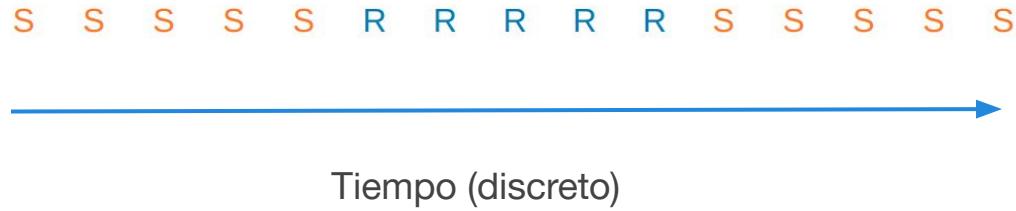
Grafo de Transición de 1 paso

Una matriz tiene un grafo equivalente

$$P(\Delta t = 1) = \begin{bmatrix} P_{AA} & P_{AB} & P_{AC} \\ P_{BA} & P_{BB} & P_{BC} \\ P_{CA} & P_{CB} & P_{CC} \end{bmatrix}$$



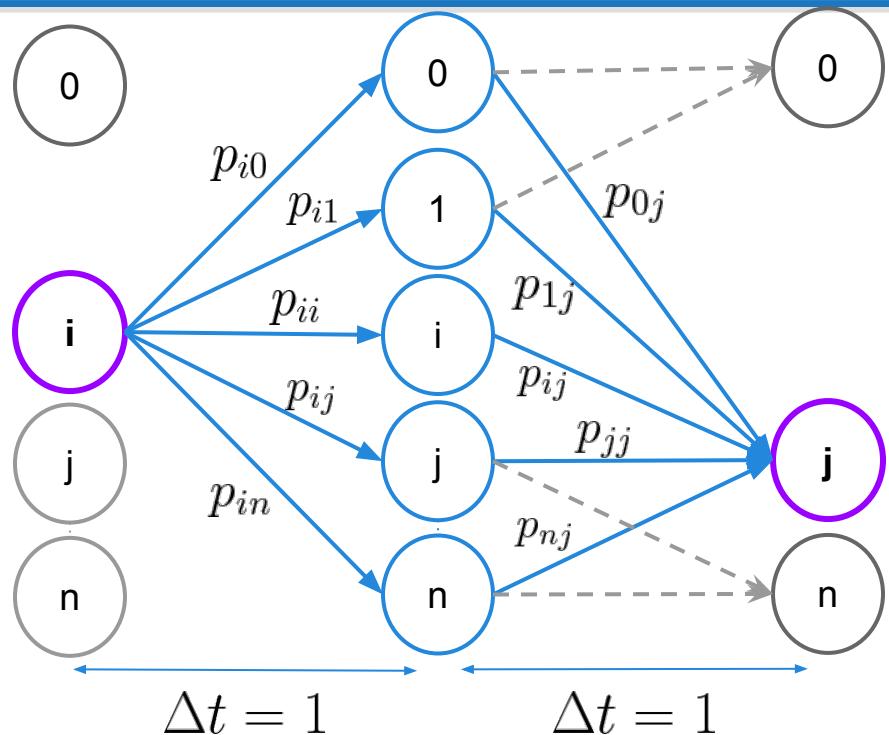
Simulación con cadenas de markov



En este ejemplo observamos un sistema de dos estados con el supuesto de que sigue un fenómeno de markov en tiempo discreto. A la derecha se encuentra la cadena de markov correspondiente con sus probabilidades de transición representadas por el grosor de los arcos. A la izquierda se puede observar el estado del sistema para cada instante de tiempo (fuente <https://setosa.io/ev/markov-chains/>)

Probabilidades de transición de m pasos

Probabilidad Transición de 2 pasos ($\Delta t = 2$)



$$P_{ij}^{(2)} = P(X_{t+2} = j | X_t = i)$$

$$P_{ij}^{(2)} = \sum_{k=0}^{k=n} (P_{ik} \times P_{kj})$$

*Ecuación de Chapman Kolmogorov

Podemos calcular la probabilidad de transición = 2 desde un estado origen “ i ” a un estado destino “ j ” mediante la union de todas las probabilidades de transición posibles entre los dos estados.

Ecuación de Chapman-Kolmogorov ($\Delta t = 2$)

La manera de expresar las probabilidades de transición de una cadena de Markov en “n” pasos es a través de la ecuación de Chapman-Kolmogorov. Estas indican que la probabilidad de pasar del estado “i” al “j” en n pasos (en este caso n=2) es primero sumando todas las probabilidades de pasar del “i” a cualquiera de los estados intermedios “k” en 1 paso y luego sumar todas las probabilidades de pasar del estado “k” al destino “j” en el siguiente paso.

$$P_{ij}^{(2)} = \sum_{k=0}^{k=n} (P_{ik} \times P_{kj})$$

Matriz de Probabilidad Transición de 2 pasos ($\Delta t = 2$)



```
import numpy as np

# matriz de transicion de 1 paso
P1 = np.array([p11, p12, p13],
              [p21, p22, p23],
              [p31, p32, p33])

# matriz de transicion de 2 pasos
P2 = np.dot(P1, P1)
```

Matriz de transición
de 2 pasos

Matriz de transición
de 1 pasos

Matriz de transición
de 1 pasos

$$P^{(2)} = P^{(1)} \times P^{(1)}$$

$$P^{(2)} = \begin{bmatrix} P_{11}^{(2)} & P_{12}^{(2)} & \dots & P_{1n}^{(2)} \\ P_{21}^{(2)} & P_{22}^{(2)} & \dots & P_{2n}^{(2)} \\ P_{31}^{(2)} & \dots & \dots & \dots \\ \dots & \dots & P_{ij}^{(2)} & \dots \\ P_{n1}^{(2)} & \dots & \dots & P_{nn}^{(2)} \end{bmatrix} = \begin{bmatrix} P_{11}^{(1)} & P_{12}^{(1)} & \dots & P_{1n}^{(1)} \\ P_{21}^{(1)} & P_{22}^{(1)} & \dots & P_{2n}^{(1)} \\ P_{31}^{(1)} & \dots & \dots & \dots \\ \dots & \dots & P_{ij}^{(1)} & \dots \\ P_{n1}^{(1)} & \dots & \dots & P_{nn}^{(1)} \end{bmatrix} \times \begin{bmatrix} P_{11}^{(1)} & P_{12}^{(1)} & \dots & P_{1n}^{(1)} \\ P_{21}^{(1)} & P_{22}^{(1)} & \dots & P_{2n}^{(1)} \\ P_{31}^{(1)} & \dots & \dots & \dots \\ \dots & \dots & P_{ij}^{(1)} & \dots \\ P_{n1}^{(1)} & \dots & \dots & P_{nn}^{(1)} \end{bmatrix}$$

El producto interno de las matrices de transición de 1 paso da como resultado la matriz de transición de 2 pasos.

Probabilidad de transición en M pasos

$$p_{ij}^{(m)} = \sum_{k=0}^n p_{ik} \cdot p_{kj}^{(m-1)}$$

Probabilidad de transición en M pasos: expresión analítica para los estados “i” y “j”

$$\begin{aligned} P^{(2)} &= P^{(1)} \cdot P^{(1)} \\ P^{(3)} &= P^{(1)} \cdot P^{(2)} \\ P^{(4)} &= P^{(1)} \cdot P^{(3)} \\ P^{(5)} &= P^{(1)} \cdot P^{(4)} \\ &\dots && \dots \\ P^{(m)} &= P^{(1)} \cdot P^{(m-1)} \end{aligned}$$

Probabilidad de transición en M pasos: Expresión matricial para todos los estados del sistema

Mediante el producto de probabilidades de transición de 1 paso y (m-1) pasos podemos obtener la probabilidad de transición de m pasos de cualquier estado origen “i” a cualquier estado destino “j” del sistema.



```
import numpy as np

# matriz de transición de 1 paso
P1 = np.array([[p11, p12, p13],
               [p21, p22, p23],
               [p31, p32, p33]])

# matriz de transición de 2 pasos
P2 = np.dot(P1, P1)

# matriz de transición de 3 pasos
P3 = np.dot(P1, P2)
```

Vector de probabilidad incondicional de estado

Probabilidad incondicional de estado

La probabilidad incondicional de estado es aquella que indica la chance que hay de que el sistema se encuentre en el estado “i” dado el instante “Ti”.

$$p_i(t) = p_{x=i}(t)$$

El conjunto de probabilidades de estado individuales de cada uno de los estados del sistema definen el **vector de probabilidades de estado p(t)**

$$p(t) = [p_1(t), p_2(t), \dots, p_i(t), \dots, p_n(t)]$$

¿Qué condiciones debe cumplir este vector?

Cadenas de Markov: Vector Probabilidad de Estado

Probabilidad de estado $x = i$ en tiempo t .

$$p_i(t) = p_{x=i}(t)$$

$$p(t_0) = [p_1(t_0), p_2(t_0), \dots, p_i(t_0), \dots, p_n(t_0)]$$

$$p(t_1) = [p_1(t_1), p_2(t_1), \dots, p_i(t_1), \dots, p_n(t_1)]$$

$$p(t_2) = [p_1(t_2), p_2(t_2), \dots, p_i(t_2), \dots, p_n(t_2)]$$

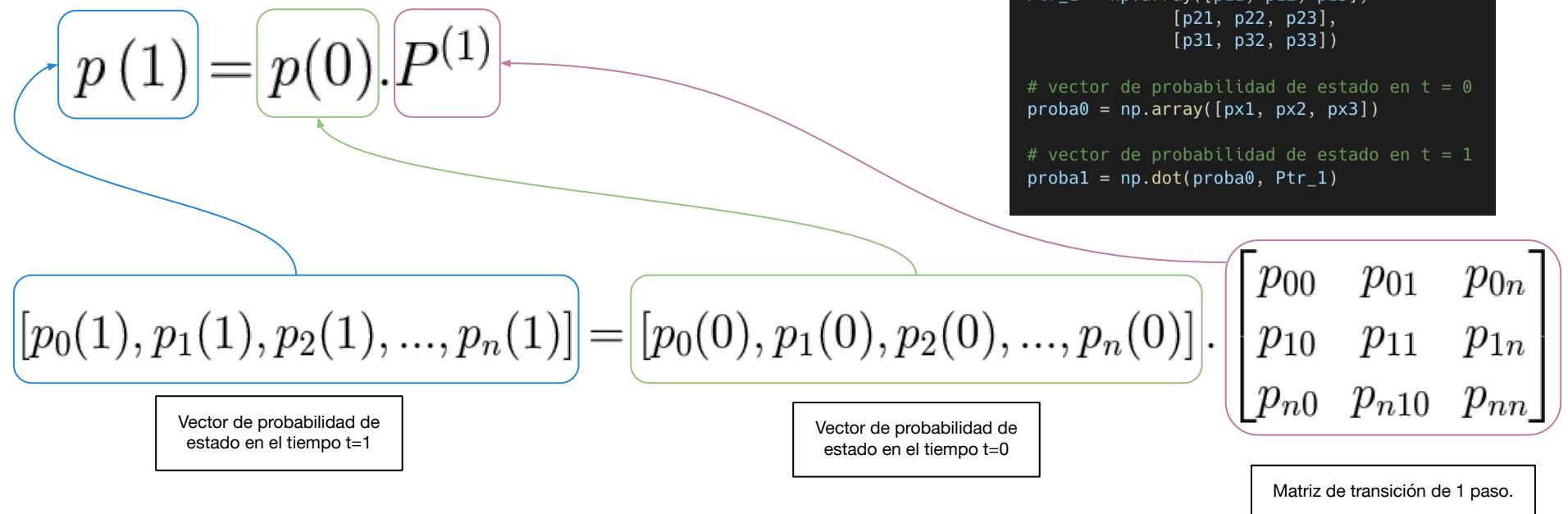
...

$$p(t_m) = [p_1(t_m), p_2(t_m), \dots, p_i(t_m), \dots, p_n(t_m)]$$

Cada “paso” o instante de tiempo tendrá un vector de probabilidad de estado asociado que indica la probabilidad de que el sistema se encuentre en alguno de los estados en cualquier tiempo t .

Ojo!! Este vector **no** es de transición ya que no hay un estado origen y un estado destino. Desde esta óptica no hay ningún “salto”. De todos modos el vector de probabilidad (incondicional) de estado estará definido de alguna manera por la matriz de transición de 1 paso.

Cadenas de Markov: Vector Probabilidad de Estado en t+1



Por ejemplo: Para calcular el vector de probabilidad de estado en el “t=1” teniendo el vector de probabilidad de estado en “t=0” (a.k.a. vector de estado inicial) y la matriz de transición de 1 paso.

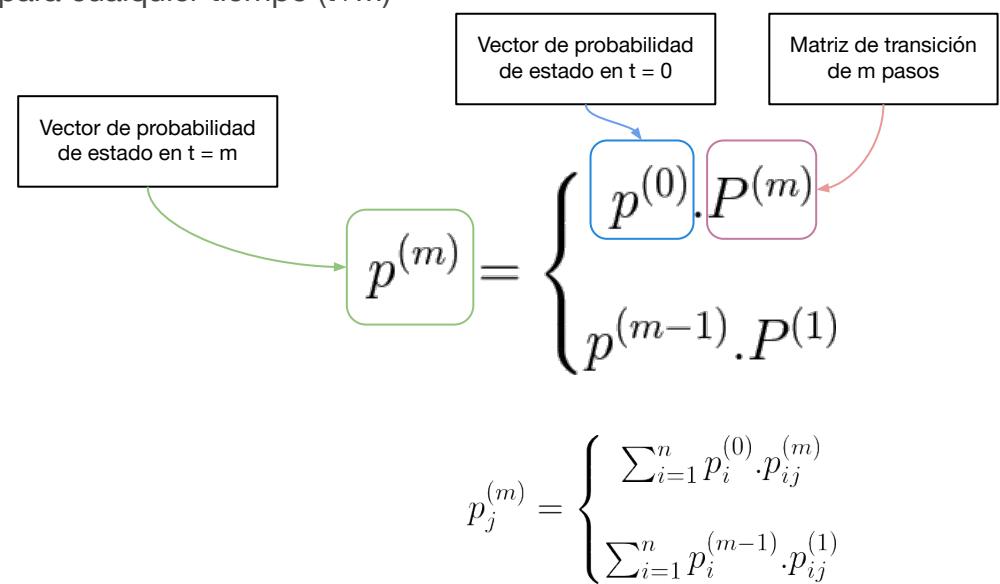
Cadenas de Markov: Vector Probabilidad de Estado en t = m

Entonces si conocemos

- Vector de proba de estado en un tiempo T (podemos llamarlo vector de estado inicial)
- Matriz de transición de M pasos

Podemos estimar la proba las probabilidades de estado para cualquier tiempo (t+M)

```
● ● ●  
import numpy as np  
  
# matriz de transicion de m pasos  
Ptr_m = np.array([[p11, p12, p13],  
                 [p21, p22, p23],  
                 [p31, p32, p33]])  
  
# vector de probabilidad de estado en t = 0  
proba0 = np.array([px1, px2, px3])  
  
# vector de probabilidad de estado en t = m  
probam = np.dot(proba0, Ptr_m)
```



Las ecuaciones reflejan que se puede arribar a un mismo vector de probabilidad de estado partiendo desde distintos vectores de probabilidad de estado inicial con la matriz de transición correspondiente.

Cadenas de Markov: Vector Probabilidad de Estado en t = m -> infinito

```
● ● ●  
import numpy as np  
  
# cantidad de pasos a dar  
m = 10000  
  
# matriz de transicion de 1 pasos  
Ptr_1 = np.array([[p11, p12, p13],  
                 [p21, p22, p23],  
                 [p31, p32, p33]])  
  
for i in range(0,m):  
    # en la primera iteración la matriz de M pasos es la de 1 paso  
    if i == 0:  
        Ptr_m = Ptr_1 # Ptr_m = matriz de transicion de M pasos  
  
    # en las siguientes iteraciones calculo y sobreescribo la matriz de M pasos  
    else:  
        Ptr_m = np.dot(Ptr_1,Ptr_m)
```

$$\lim_{m \rightarrow \infty} P^{(m)} = \begin{bmatrix} p_{11}^{(m)} & p_{12}^{(m)} & \dots & p_{1n}^{(m)} \\ p_{21}^{(m)} & p_{22}^{(m)} & \dots & p_{2n}^{(m)} \\ \dots & \dots & \dots & \dots \\ p_{n1}^{(m)} & p_{n2}^{(m)} & \dots & p_{nn}^{(m)} \end{bmatrix}$$

Podemos estudiar como se comportaría el sistema modelado con una cadena de Markov en el “**estado estacionario**” o “**régimen permanente**” ($m \rightarrow \infty$). Para ello debemos observar el comportamiento de la cadena luego de “ m ” pasos de tiempo con “ m ” tendiendo a infinito. Si observamos la matriz de transición de m pasos sucederá es que las probabilidades de transición serán igual no importa desde que estado origen se inicie la transición. Esto quiere decir que el vector de probabilidad de estado se repetirá en cada fila de la matriz de transición.

Cadenas de Markov en régimen permanente

La probabilidad de estado en “t=m” cuando t tiende a infinito será igual que la probabilidad de estado en “t = m-1”. De esta manera multiplicando el vector de probabilidad de estado por la matriz de transición de un paso dará por resultado el vector de probabilidad de estado en régimen permanente.

$$\lim_{m \rightarrow \infty} p^{(m)} = \lim_{m \rightarrow \infty} p^{(m-1)} \cdot P^{(1)}$$

$$\rightarrow p^{(m)} = p^{(m)} \cdot P^{(1)}$$

$$\sum_{i=1}^n p_i = 1$$

Toy example: cadenas de markov en clima

Cadenas de Markov: Aplicación en clima

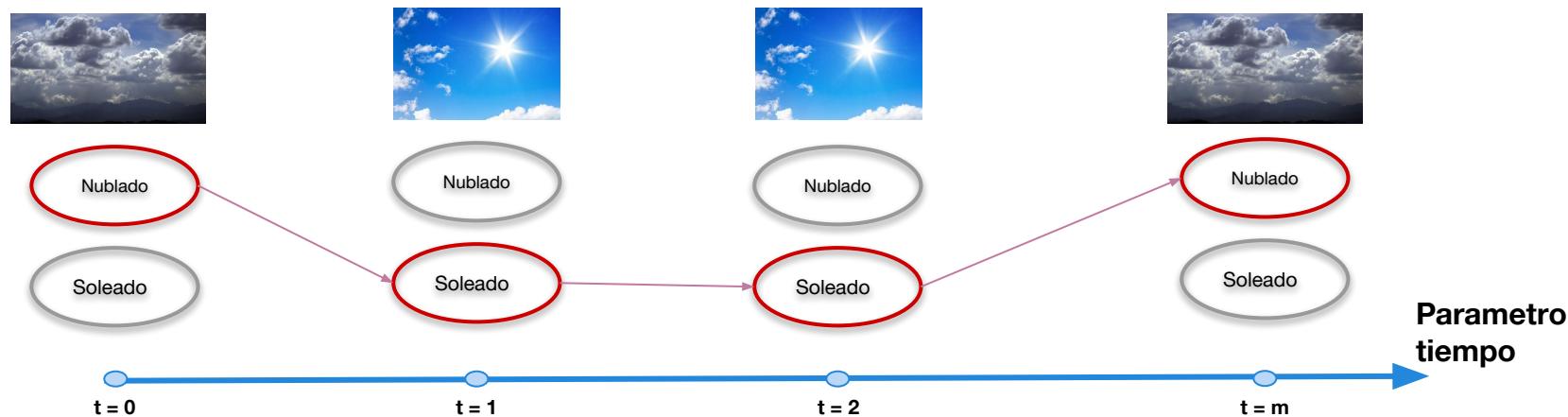
En un pueblo, al 90% de los días soleados le siguen días soleados, y al 80% de los días nublados le siguen días nublados. Con esta información modelar el clima del pueblo como una cadena de Markov.

Si el dia de hoy el clima es soleado, cual es la probabilidad que en 3 dias el clima sea soleado?

Cual es la probabilidad de tener dias soleados en el largo plazo?

Procesos estocásticos: toy example clima

estados



Supongamos que nuestro sistema de estudio es el clima caracterizado por tres posibles estados: Nublado, Soleado y Lluvioso. En cada instante de tiempo “t” el sistema se encontrará en alguno de los estados posible. A medida que transcurre el tiempo el sistema tomará distintos estados bajo cierta distribución de probabilidad.

Cadenas de Markov: construcción de la matriz de transición de 1 paso



	Soleado	Nublado
Soleado	0.9	
Nublado		0.8



Matriz original que obtenemos desde los datos.

	Soleado	Nublado
Soleado	0.9	0.1
Nublado	0.2	0.8

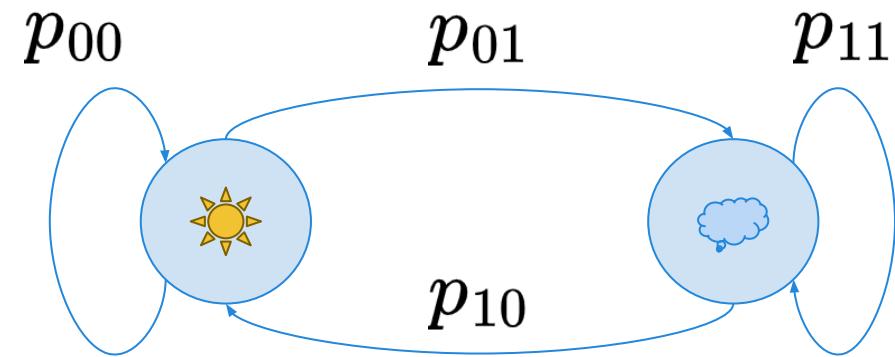
Matriz final que completamos con las propiedades de markov.

$$\sum_{j=1}^n p_{ij} = 1$$

$$\sum_{j=1}^n p_{ij} = 1$$

Matriz de transición: grafo asociado

	Soleado	Nublado
Soleado	0.9	0.1
Nublado	0.2	0.8



En python

Estado 0: Soleado



Estado 1: Nublado



Calculamos la matriz de transición en M pasos

```
● ● ●  
# Importamos Numpy para hacer cuentas con vectores, matrices  
import numpy as np  
# Importamos Matplotlib para visualizar  
import matplotlib.pyplot as plt  
# importamos seaborn  
import seaborn as sns  
  
# defino desde los datos la matriz de transición de 1 paso  
P1 = np.array([[0.9, 0.1],[0.2, 0.8]])  
# defino desde los datos el vector de estado inicial en t=1  
p1 = np.array([1,0])  
  
# determinar la cantidad total de estados  
estados = np.shape(P1)[0]  
  
# cantidad de pasos a simular  
m = 20  
  
# en esta matriz vamos a guardar el vector de estados para cada paso t  
vector_estado = np.zeros((m,estados))  
  
# for loop para cantidad de pasos  
for i in range(0,m):  
    # configuración del instante inicial  
    if i == 0:  
        # únicamente en el paso inicial la matriz de transición de m pasos Pm es P1  
        Pm = P1  
        # el vector de estado en el tiempo inicial es p1  
        vector_estado[i, :] = p1  
  
    else:    # para el resto de los pasos  
        Pm = np.dot(P1, Pm) # calculo la matriz de transición de M pasos  
        vector_estado[i,:] = np.dot(p0, Pm) # calculo el vector de estado en el tiempo M
```

Matriz de transición de
2 pasos

$$P^{(2)} = P^{(1)}.P^{(1)}$$
$$P^{(3)} = P^{(1)}.P^{(2)}$$
$$P^{(4)} = P^{(1)}.P^{(3)}$$
$$P^{(5)} = P^{(1)}.P^{(4)}$$

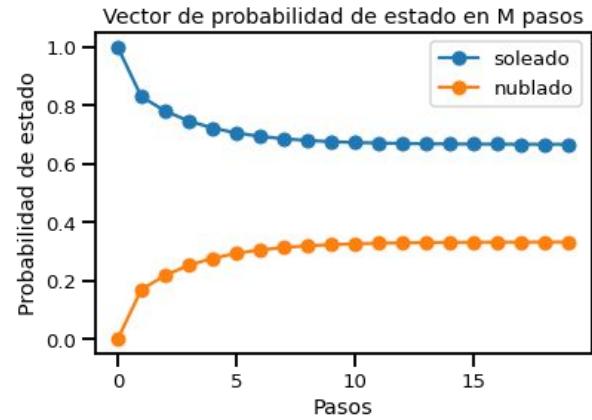
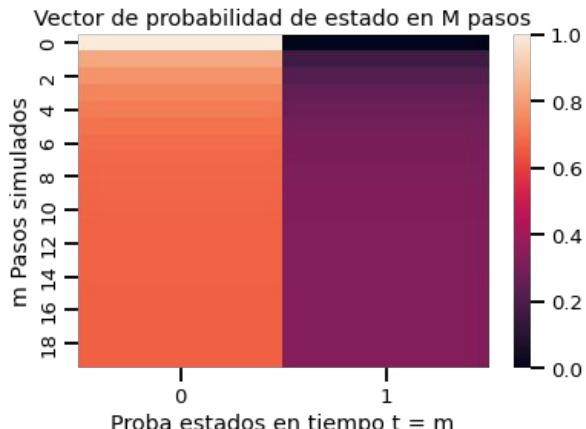
...

Matriz de transición de
m pasos

$$P^{(m)} = P^{(1)}.P^{(m-1)}$$

Cálculo del vector de probabilidad de estado de la cadena de markov en M pasos

```
● ● ●  
print(vector_estado)  
  
array([[1.        , 0.        ],  
       [0.83      , 0.17     ],  
       [0.781     , 0.219    ],  
       [0.7467    , 0.2533   ],  
       [0.72269   , 0.27731  ],  
       [0.705883  , 0.294117 ],  
       [0.6941181 , 0.3058819],  
       [0.68588267, 0.31411733],  
       [0.68011787, 0.31988213],  
       [0.67608251, 0.32391749],  
       [0.67325776, 0.32674224],  
       [0.67128043, 0.32871957],  
       [0.6698963 , 0.3301037 ],  
       [0.66892741, 0.33107259],  
       [0.66824919, 0.33175681],  
       [0.66777443, 0.33222557],  
       [0.6674421 , 0.3325579 ],  
       [0.66720947, 0.33279853],  
       [0.66704663, 0.33295337],  
       [0.66693264, 0.33306736]])  
  
## visualizar el vector de estado en M pasos  
sns.heatmap(vector_estado)  
plt.title('Vector de probabilidad de estado en M pasos')  
plt.xlabel('Proba estados en tiempo t = m')  
plt.ylabel('m Pasos simulados')  
plt.show()  
  
## visualizar el vector de estado en M pasos  
sns.set_context(context = 'talk', font_scale = 0.8)  
plt.title('Vector de probabilidad de estado en M pasos')  
plt.plot(vector_estado[:,0], marker = 'o', label = 'soleado')  
plt.plot(vector_estado[:,1], marker = 'o', label = 'nublado')  
plt.xlabel('Pasos')  
plt.ylabel('Probabilidad de estado')  
plt.legend()
```

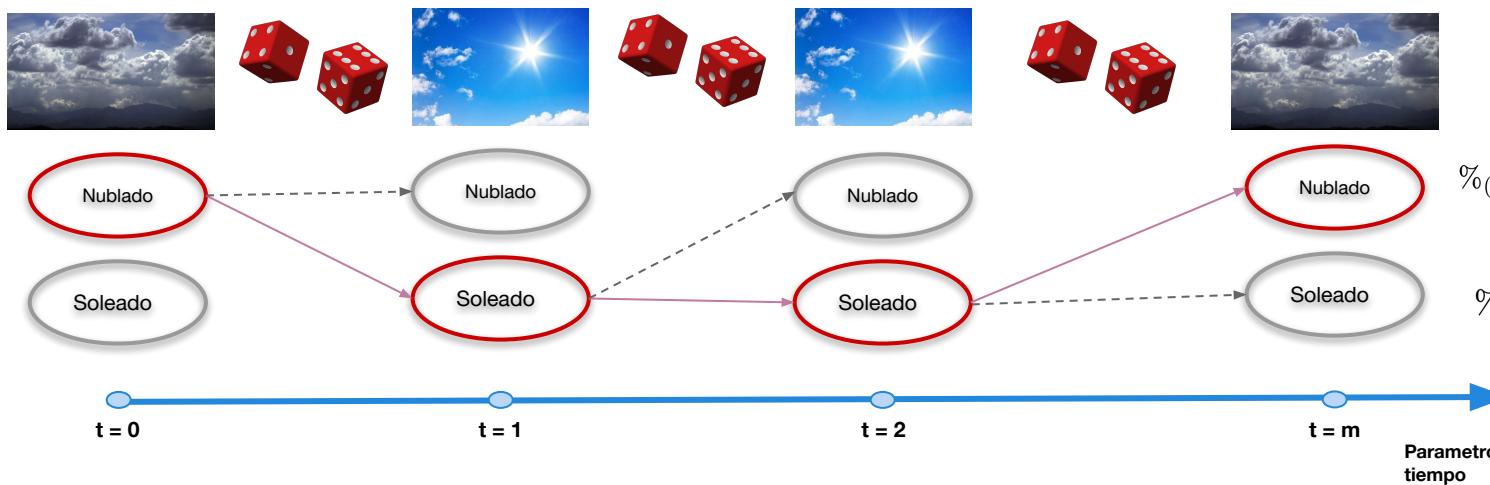


La proba de encontrar un dia soleado se estabiliza en 0.67

La proba de encontrar un dia nublado se estabiliza en 0.33

Luego de M=20 pasos la la probabilidad de estado en el paso M converge a un valor estable. Esto quiere decir que si “estoy parado en el presente y miro a largo plazo existe una proba de 0.67 de que el dia este soleado y un 0.33 de que este nublado”.

Cadenas de Markov: simulación de 1 experimento



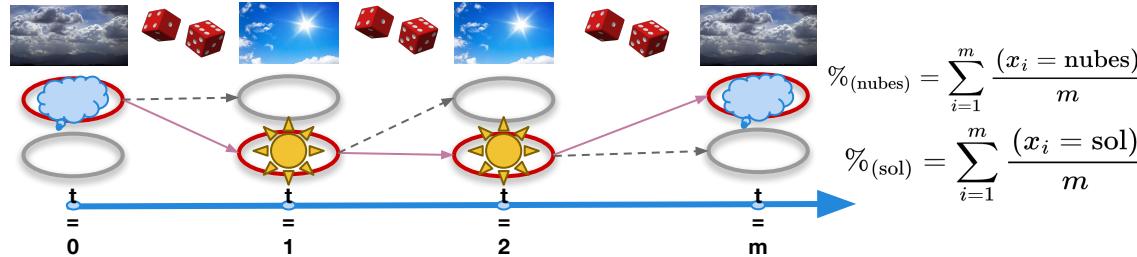
$$\%_{(\text{nubes})} = \sum_{i=1}^m \frac{(x_i = \text{nubes})}{m}$$

$$\%_{(\text{sol})} = \sum_{i=1}^m \frac{(x_i = \text{sol})}{m}$$

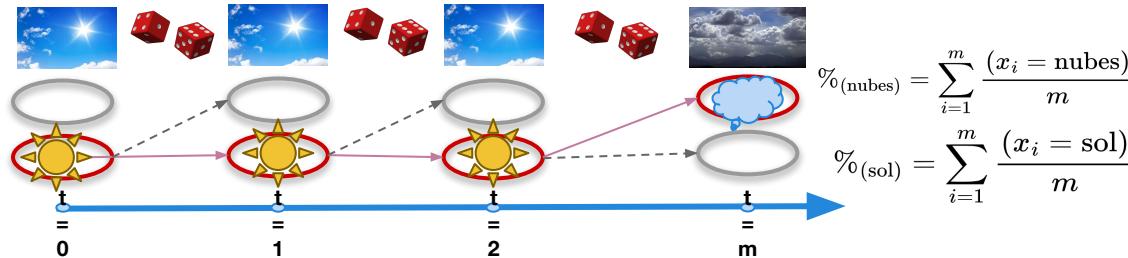
Para simular el sistema, partiremos de un estado inicial por ejemplo nublado. Luego en cada paso dependiendo del estado donde se encuentre el sistema se realizará un muestreo aleatorio para determinar utilizando las probabilidades de transición de 1 paso a que estado saltará el sistema. Este proceso se repite m veces para simular el sistema en m pasos. Al finalizar se computará el % de veces que el sistema estuvo en estado Soleado y en estado Nublado.

Simulamos el sistema multiples veces

Experimento 1
Estado inicial = 1

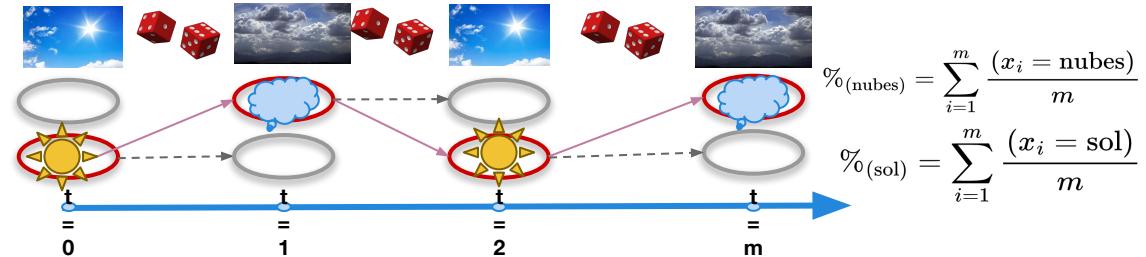


Experimento 2
Estado inicial = 0



En python
Estado 0: Soleado ☀
Estado 1: Nublado ☁

Experimento 3
Estado inicial = 0





```
from random import choices
# cantidad de experimentos
exp = 1000

# cantidad de pasos a simular
m = 500

# array con los estados posibles en el sistema
estados_posibles = [0, 1] # 0 = soleado , 1 = nublado

# creo matriz de estados. Cara renglon es un vector booleano marcando el estado actual del sistema.
estado_actual = np.zeros((m, estados, exp))

# loop para cada experimento
for e in range(0,exp):

    # loop para simular m en m pasos el sistema.
    for i in range(0,m):

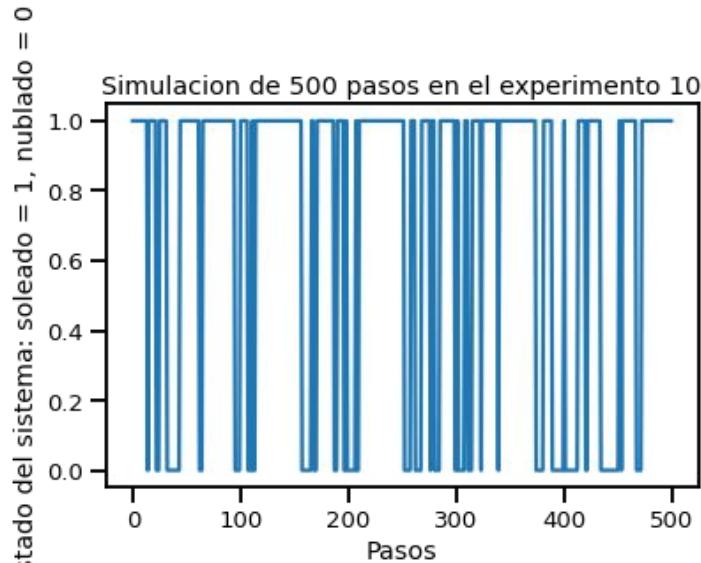
        # configuracion del instante inicial
        if i == 0:
            # unicamente en el paso inicial se determina el estado de inicio del sistema
            estado_actual[i, :, e] = np.array([1,0])

        else:
            # conociendo el estado previo del sistema "estado_actual[i-1]"
            # calculo la distribucion de proba de transicion hacia cada estado
            proba_transicion_m = np.dot(estado_actual[i-1,:,:e],P1)

            # muestreamos la distribucion de proba de transicion
            # basandonos en el estado del sistema en t=t-1
            estado_actual_m = choices(estados_posibles, proba_transicion_m)

            # el estado actual lo guardo en la fila "i", "columna estado actual" y capa "e"
            estado_actual[i, estado_actual_m ,e] = 1
```

Observamos uno de los experimentos.



% de veces que sale cada estado en cada experimento

