Simulación de Cadenas de Markov-I

Vamos a simular una cadena de Markov en un espacio con dos estados cuya matriz de transición viene dada por

$$P = \begin{pmatrix} 0.175 & 0.825 \\ 0.526 & 0.474 \end{pmatrix}$$

Y cuya distribución inicial es $\mu_0 = (0.2, 0.8)$.

Lo primero que hacemos es definir la matriz de transición.

```
P = matrix(CC ,[[0.175 , 0.825],[0.526 ,0.474]])
P
```

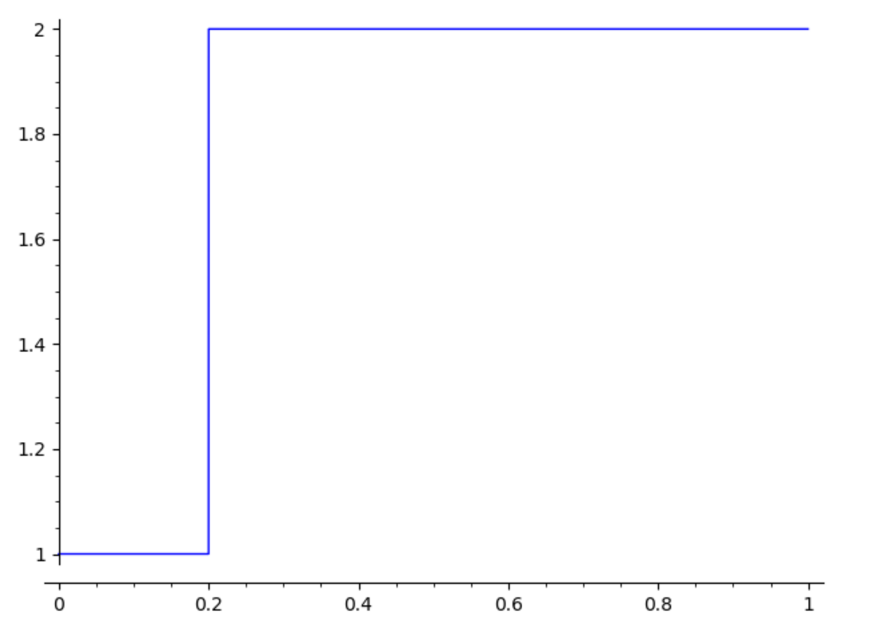
```
[0.175000000000000 0.825000000000000]
[0.526000000000000 0.474000000000000]
```

A continuación, definimos la función de inicialización

```
Phi=piecewise([[(0,0.2),1],[(0.2,1),2]]);

Phi

plot(Phi(x),(x,0,1))
```



Ya podemos definir la función de actualización:

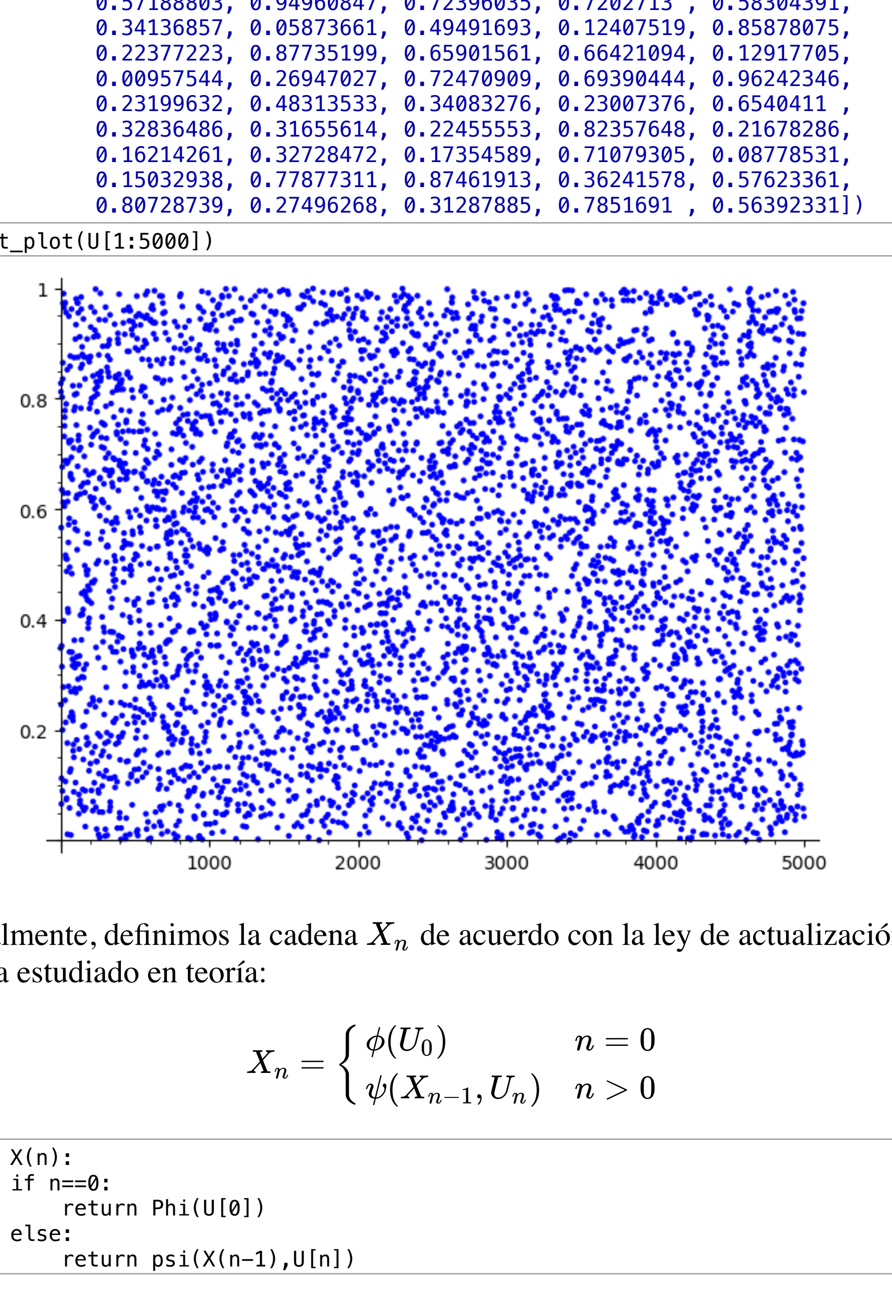
```
def psi(n,x):
    if n==1 and 0<=x<0.175:
        return 1
    if n==1 and 0.175<=x<=1:
        return 2
    if n==2 and 0<=x<0.526:
        return 1
    if n==2 and 0.526<=x<=1:
        return 2
```

Para poder generar la cadena es necesario generar antes una secuencia de valores aleatorios uniformemente distribuidos, que son los U_n que usamos para la actualización de los estados. Consideramos que basta generar 50000 de dichos valores y guardarlos en una única sucesión. (Aunque en realidad podríamos usar muchos menos valores... todo depende del experimento que vayamos a realizar)

```
import numpy as np
U=np.random.uniform(0,1,50000)
U[0:100]
```

```
array([0.40083746, 0.34835457, 0.82791346, 0.81265363, 0.24592586,
       0.56637186, 0.06589405, 0.11238956, 0.35229126, 0.63509451,
       0.31457454, 0.80275195, 0.67648502, 0.08938754, 0.19994731,
       0.8358177 , 0.8651584 , 0.97309107, 0.61223356, 0.27768768,
       0.39760514, 0.99053884, 0.62899497, 0.96493988, 0.97074842,
       0.73180532, 0.60526306, 0.75091456, 0.7666011 , 0.68757348,
       0.77862637, 0.25376806, 0.97411035, 0.10658911, 0.51654766,
       0.21372431, 0.51079361, 0.9564458 , 0.79047488, 0.07380427,
       0.68786802, 0.07634282, 0.48728061, 0.68317815, 0.50951518,
       0.85263426, 0.17623457, 0.01099148, 0.61374985, 0.41928304,
       0.62900189, 0.68700131, 0.41304873, 0.53957767, 0.27905101,
       0.57188803, 0.94960847, 0.72396035, 0.7202713 , 0.58304391,
       0.34136857, 0.05873661, 0.49491693, 0.12407519, 0.85878075,
       0.22377223, 0.87735199, 0.65901561, 0.66421094, 0.12917705,
       0.00957544, 0.26947027, 0.72470909, 0.69390444, 0.96242346,
       0.23199632, 0.48313533, 0.34083276, 0.23007376, 0.6540411 ,
       0.32836486, 0.31655614, 0.22455553, 0.82357648, 0.21678286,
       0.16214261, 0.32728472, 0.17354589, 0.71079305, 0.08778531,
       0.15032938, 0.77877311, 0.87461913, 0.36241578, 0.57623361,
       0.80728739, 0.27496268, 0.31287885, 0.7851691 , 0.56392331])
```

```
list_plot(U[1:5000])
```



Finalmente, definimos la cadena X_n de acuerdo con la ley de actualización que se ha estudiado en teoría:

$$X_n = \begin{cases} \phi(U_0) & n = 0 \\ \psi(X_{n-1}, U_n) & n > 0 \end{cases}$$

```
def X(n):
    if n==0:
        return Phi(U[0])
    else:
        return psi(X(n-1),U[n])
```

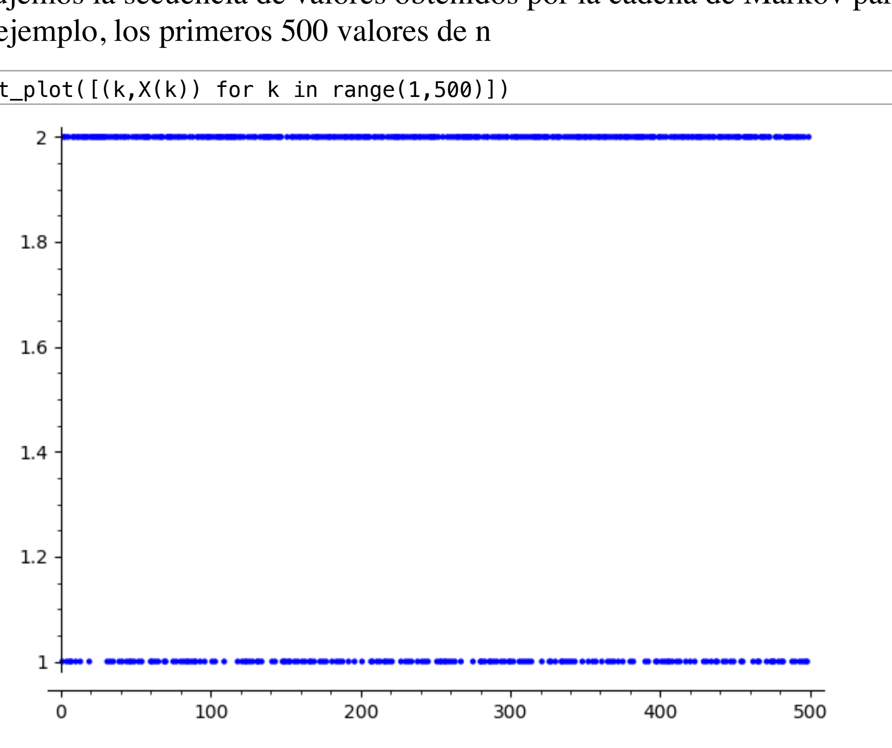
Los primeros valores que obtenemos son:

```
[X(0),X(1),X(2),X(3),X(50)]
```

```
[2, 1, 2, 2, 2]
```

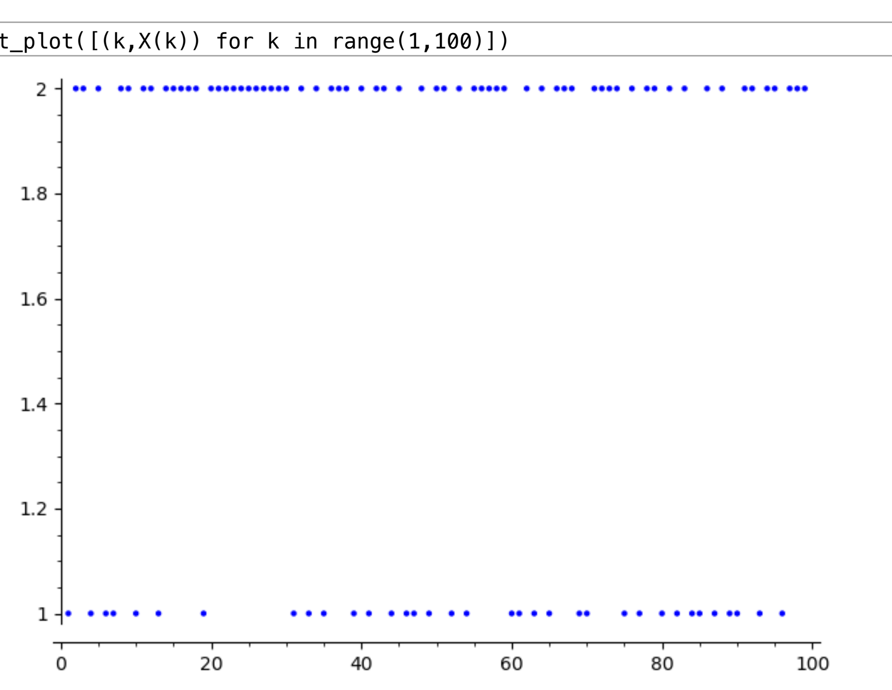
Dibujemos la secuencia de valores obtenidos por la cadena de Markov para, por ejemplo, los primeros 500 valores de n

```
list_plot([(k,X(k)) for k in range(1,500)])
```



Como salen muy apelotonados (debido a que el gráfico es pequeño), quizás es mejor pintar menos entradas de la sucesión:

```
list_plot([(k,X(k)) for k in range(1,100)])
```

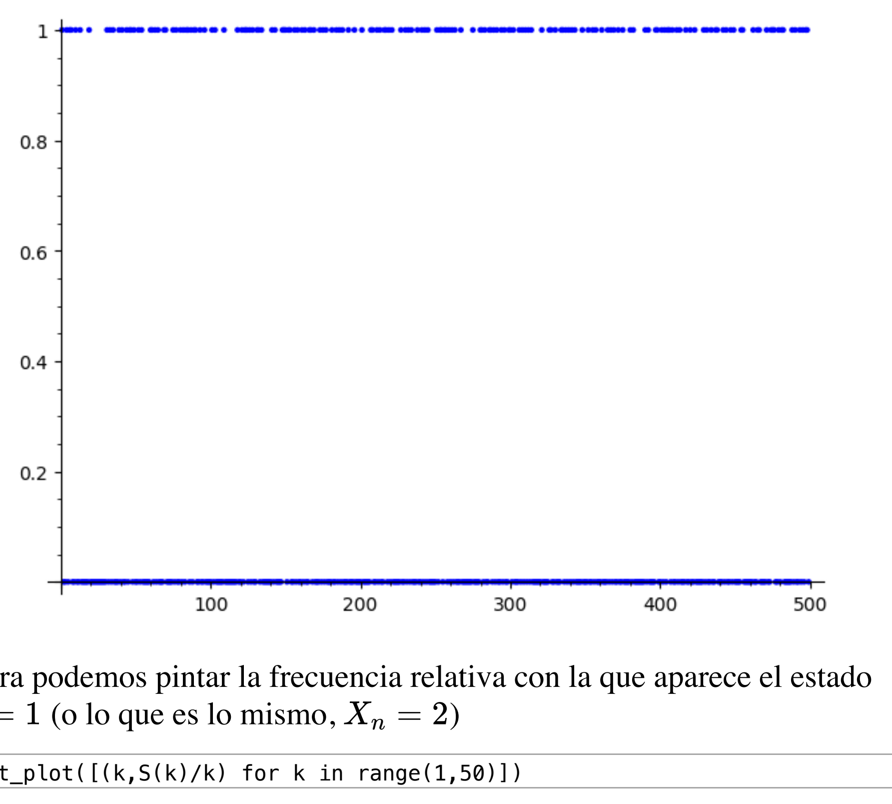


Con el ánimo de contabilizar la frecuencia con la que la cadena de Markov se encuentra en cada estado (son dos),hacemos las siguientes transformaciones:

```
def Y(n):
    if X(n)==1:
        return 1
    else:
        return 0
```

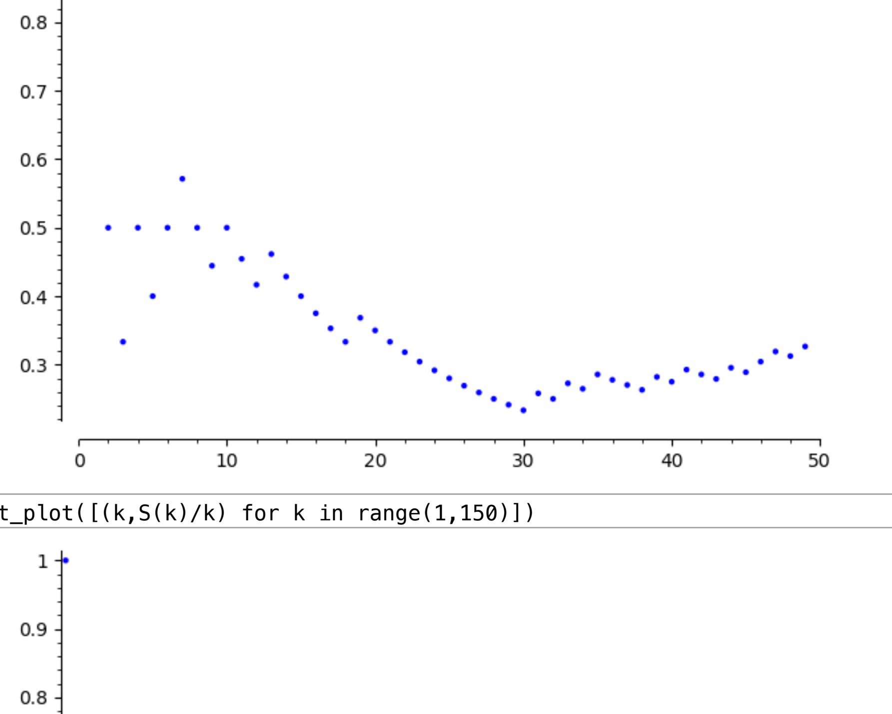
```
def S(n):
    if n==0:
        return Y(0)
    else:
        return Y(n)+S(n-1)
```

```
list_plot([(k,Y(k)) for k in range(1,500)])
```

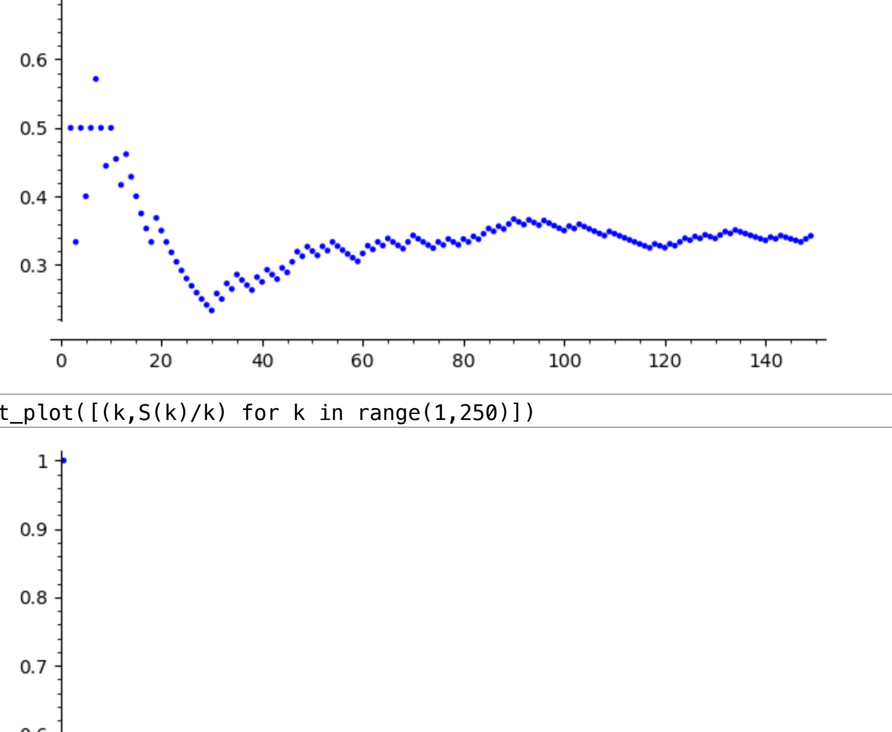


Ahora podemos pintar la frecuencia relativa con la que aparece el estado $Y_n = 1$ (o lo que es lo mismo, $X_n = 2$)

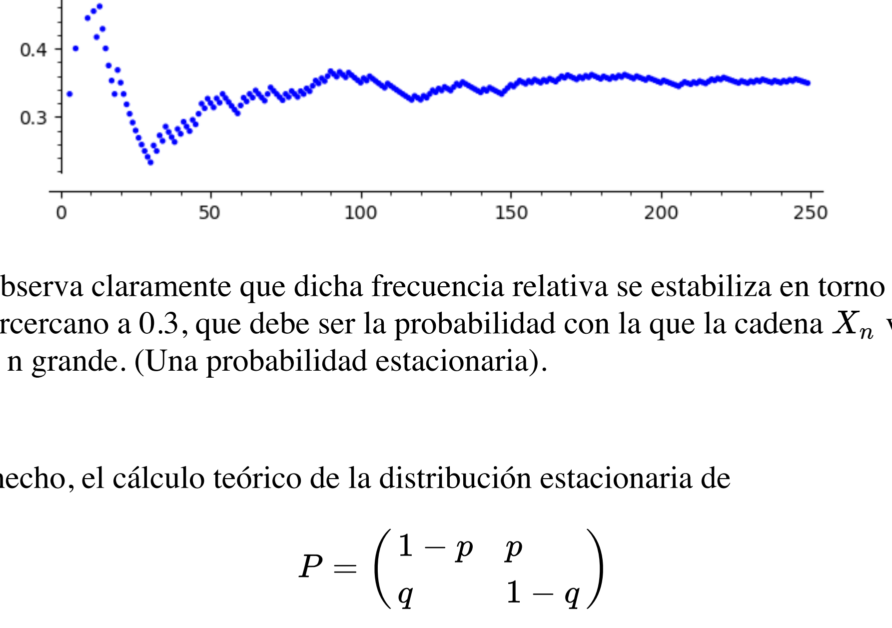
```
list_plot([(k,S(k)/k) for k in range(1,50)])
```



```
list_plot([(k,S(k)/k) for k in range(1,150)])
```



```
list_plot([(k,S(k)/k) for k in range(1,250)])
```



Se observa claramente que dicha frecuencia relativa se estabiliza en torno a un valorcercano a 0.3, que debe ser la probabilidad con la que la cadena X_n vale 2 para n grande. (Una probabilidad estacionaria).

De hecho, el cálculo teórico de la distribución estacionaria de

$$P = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$$

es $\mu_0 = (\frac{q}{p+q}, \frac{p}{p+q})$, que en nuestro caso vale:

```
[0.825/(0.526+0.825),0.526/(0.526+0.825)]
```

```
[0.610658771280533, 0.389341228719467]
```

Es decir, $X_n = 2$ con probabilidad 0.389...para n grande (que es aproximadamente lo que hemos obtenido)