



FACULTAD DE CIENCIAS  
**EXACTAS**  
UNIVERSIDAD NACIONAL DEL CENTRO  
DE LA PROVINCIA DE BUENOS AIRES

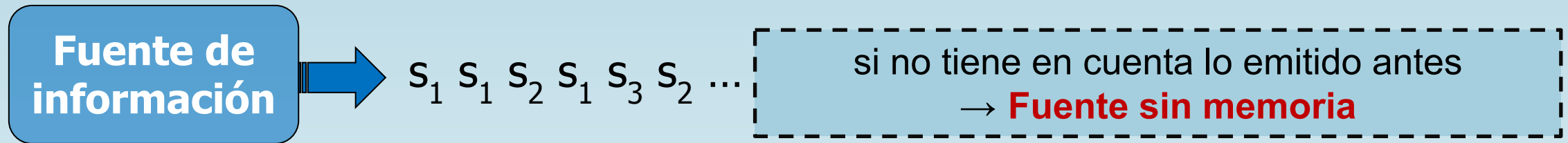
# TEORÍA DE LA INFORMACIÓN

## Fuentes de Información <sup>(2)</sup>



# Emisión de símbolos de la Fuente por muestreo computacional

¿Cómo simular la emisión aleatoria de un símbolo de la fuente?



Generar cada símbolo según su probabilidad de ocurrencia:

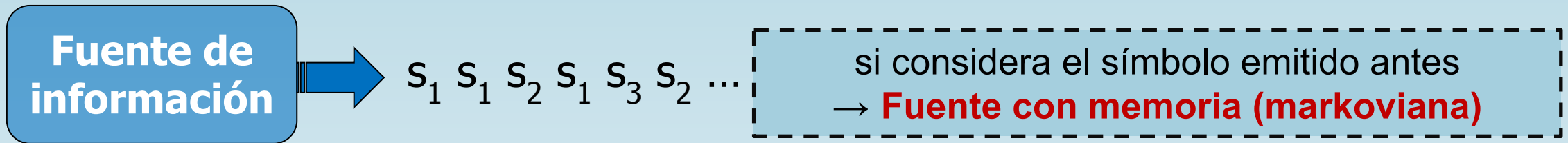
```
Emitir_Simbolo ()  
{  
  r=rand()  
  for (i= 0 to #simbolos)  
    if (r < Vacum[i])  
      return i  
}
```

**Ejemplo**

$$V = \begin{bmatrix} 0.3 \\ 0.5 \\ 0.2 \end{bmatrix} \quad V_{\text{acum}} = \begin{bmatrix} 0.3 \\ 0.8 \\ 1 \end{bmatrix}$$

# Emisión de símbolos de la Fuente por muestreo computacional

¿Cómo simular la emisión aleatoria de un símbolo de la fuente?



```
Primer_Simbolo ()
{
  r=rand ()
  for (i= 0 to #simbolos)
    if (r < V0acum [i])
      return i
}
```

condición inicial :

$$V_0 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$V_{0acum} = \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \\ \mathbf{1} \end{bmatrix}$$

```
Sig_dado_Ant (s_ant)
{
  r=rand ()
  for(i=0 to #simbolos)
    if ( r < Macum[i, s_ant] )
      return i
}
```

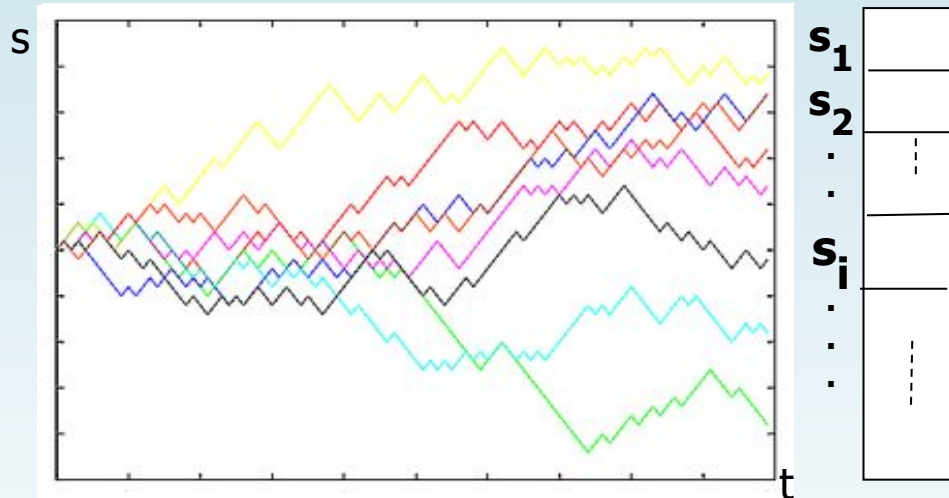
$$M = \begin{matrix} & \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \end{matrix} & \begin{bmatrix} 1/2 & 1/4 & 1/4 \\ 1/4 & 1/2 & 1/4 \\ 1/4 & 1/4 & 1/2 \end{bmatrix} \end{matrix}$$

$$M_{acum} = \begin{matrix} & \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \end{matrix} & \begin{bmatrix} \mathbf{1/2} & \mathbf{1/4} & \mathbf{1/4} \\ \mathbf{3/4} & \mathbf{3/4} & \mathbf{1/2} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} \end{bmatrix} \end{matrix}$$

# Vectores de Estado por muestreo computacional

En **fuentes markovianas** ¿cómo obtener  $\mathbf{V}_t$  por simulación computacional?

- Generar secuencias (mensajes) de  $t$  símbolos emitidos por la fuente según sus probabilidades
- Contar la cantidad de veces que cada símbolo  $s_i$  se emite en  $t$  (en un vector de tamaño *#símbolos*)
- ¿Cuántos mensajes deben generarse? → los suficientes para asegurar la convergencia



En cada elemento del vector:

$$P(S_t = s_i) \approx \frac{\#s_i \text{ en } t}{\#\text{mensajes}}$$

# Vectores de Estado por muestreo computacional

## Ejemplo

$$M = \begin{matrix} & \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \end{matrix} & \begin{bmatrix} 0 & 2/3 & 1/4 \\ 1/2 & 1/3 & 1/4 \\ 1/2 & 0 & 1/2 \end{bmatrix} \end{matrix}$$

Generación del 1º símbolo  
(según condición de inicio)

$$V_0 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Generación del próximo símbolo  
(a partir del símbolo anterior)

$$M_{acum} = \begin{bmatrix} 0 & 2/3 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1 & 1 & 1 \end{bmatrix}$$

¿Cómo obtener **V3** por simulación computacional?

#mensajes	$s(t_0)$	$s(t_1)$	$s(t_2)$	$s(t_3)$	# $s_i$ en $t_3$	$P(S_t = s_i)$
0						
1	1	0	1	1		
2	1	1	0	1		
3	1	0	2	2		
4	1	1	0	2		
5	1	0	2	0		
...	...	...	...	...	...	...

( salida de generación aleatoria )

0					[ 0 , 0 , 0 ]	[ 0 , 0 , 0 ]
1	1	0	1	1	[ 0 , 1 , 0 ]	[ 0 , 1/1 , 0 ]
2	1	1	0	1	[ 0 , 2 , 0 ]	[ 0 , 2/2 , 0 ]
3	1	0	2	2	[ 0 , 2 , 1 ]	[ 0 , 2/3 , 1/3 ]
4	1	1	0	2	[ 0 , 2 , 2 ]	[ 0 , 2/4 , 2/4 ]
5	1	0	2	0	[ 1 , 2 , 2 ]	[ 1/5 , 2/5 , 2/5 ]

→ convergencia temprana!? (controlar)

→ continuar iterando hasta convergencia de todas las probabilidades (vector de estado)



# Vectores de Estado por muestreo computacional

```
Calcular_Vector_Estado (int t )
{
    emisiones= [0, 0, 0] // cantidad de emisiones de cada  $s_i$ 
    Vt= [0, 0, 0] // Vector de estado actual
    Vt_ant= [-1, 0, 0] // Vector de estado anterior
    mensajes= 0 //cantidad de mensajes emitidos
    while not converge (Vt, Vt_ant) or (mensajes< T_MIN)
    {
        s= Primer_Simb ();
        for (i= 0 to t )
            s= Sig_dado_Ant (s)
            emisiones[s]++
            mensajes++
            Vt_ant  $\leftarrow$  Vt
            Vt  $\leftarrow$  emisiones/mensajes
        }
    }
    return Vt
}
```

```
converge (A[ ], B[ ] )
{
    for (i=0 to #simbolos)
        {
            if (abs(A[i] - B[i]) >  $\xi$  )
                return FALSE
        }
    return TRUE
}
```

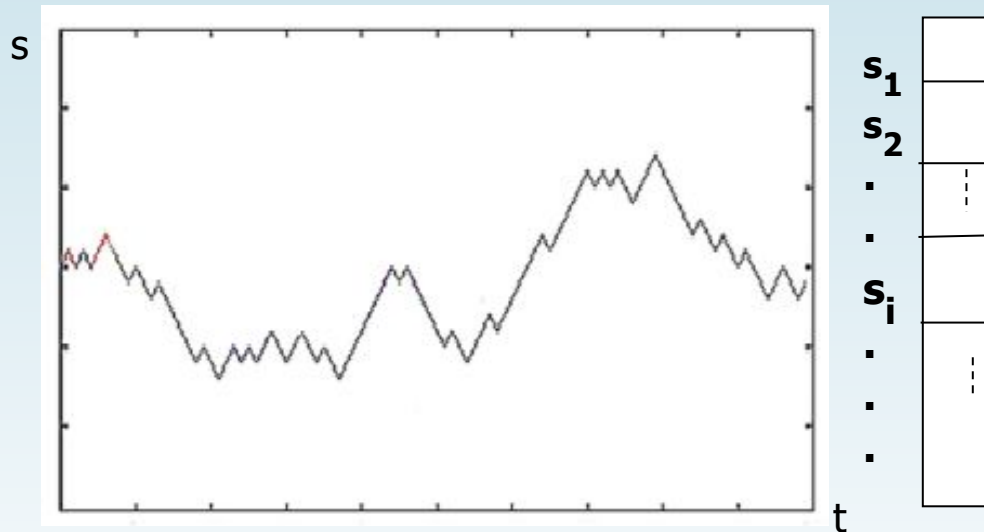
```
Primer_Simbolo ()
{
    r=rand ()
    for (i= 0 to #simbolos)
        if (r < V0acum [i])
            return i
    }
}
```

```
Sig_dado_Ant (s_ant)
{
    r=rand ()
    for(i=0 to #simbolos)
        if ( r < Macum[i, s_ant] )
            return i
    }
}
```

# Vector Estacionario por muestreo computacional

En **fuentes markovianas** ¿cómo obtener  $\mathbf{V}^*$  por simulación computacional?

- Generar una secuencia (mensaje) de símbolos emitidos por la fuente según sus probabilidades
- Contar el número de veces que se emite cada símbolo  $s_i$  en el mensaje
- ¿Cuántos símbolos del mensaje deben generarse? → suficientes para asegurar la convergencia



En cada elemento del vector:

$$P(S^* = s_i) \approx \frac{\#s_i}{\text{long. mensaje}}$$

# Vector Estacionario por muestreo computacional

## Ejemplo

$$M = \begin{matrix} & \begin{matrix} 0 & 1 & 2 \end{matrix} \\ \begin{matrix} 0 \\ 1 \\ 2 \end{matrix} & \begin{bmatrix} 0 & 2/3 & 1/4 \\ 1/2 & 1/3 & 1/4 \\ 1/2 & 0 & 1/2 \end{bmatrix} \end{matrix}$$

Generación del 1º símbolo  
(según condición de inicio)

$$V_{0acum} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Generación del próximo símbolo  
(según símbolo anterior)

$$M_{acum} = \begin{bmatrix} 0 & 2/3 & 1/4 \\ 1/2 & 1 & 1/2 \\ 1 & 1 & 1 \end{bmatrix}$$

¿Cómo obtener  $V^*$  por simulación computacional?

cant. simbolos	símbolos generados aleatoriamente	emisiones $\#s_i$	$V^*$ $P(S^*) = s_i$
0		[ 0 , 0 , 0 ]	[ 0 , 0 , 0 ]
1	<b>1</b>	[ 0 , 1 , 0 ]	[ 0 , 1/1 , 0 ]
2	<b>0</b>	[ 1 , 1 , 0 ]	[ 1/2, 1/2 , 0 ]
3	<b>1</b>	[ 1 , 2 , 0 ]	[ 1/3, 2/3 , 0 ]
4	<b>0</b>	[ 2 , 2 , 0 ]	[ 2/4, 2/4 , 0 ]
5	<b>2</b>	[ 2 , 2 , 1 ]	[ 2/5, 2/5, 1/5 ]
....	...	....	....

→ continuar iterando hasta convergencia de todas las probabilidades (vector estacionario)



# Vectores Estacionario por muestreo computacional

## *Calcular\_Vector\_Estacionario* ()

```
{
  emisiones= [0,..., 0] // cantidad de emisiones de cada si
  V*= [0,..., 0] // Vector estacionario actual
  V*_ant= [-1,..., 0] // Vector estacionario anterior
  cant_simb= 0 //cantidad de símbolos generados

  s=Primer_Simb ();
  while not converge (V*, V*_ant) or (cant_simb < S_MIN)
  {
    s= Sig_dado_Ant (s)
    ...
    ...
    V*_ant ← V*
    V* ← ...
  }
  return V*
}
```

## *Primer\_Simbolo* ()

```
{ r=rand ()
  for (i= 0 to #simbolos)
    if (r < V0acum [i])
      return i
}
```

## *Sig\_dado\_Ant* (s\_ant)

```
{ r=rand ()
  for(i=0 to #simbolos)
    if ( r < Macum[i, s_ant] )
      return i
}
```

## *converge* (A[ ], B[ ] )

```
{ for (i=0 to #simbolos)
  { if (abs(A[i] - B[i]) > ξ )
    return FALSE }
  return TRUE }
```



**Practicar ejercicios**

**Plantear pseudocódigos**

**Interpretar resultados**



**Consultar dudas!**

**Implementar algoritmos**

**Evaluar convergencia**

```
CalcularProbabilidadSumaDados(int suma)
{
    int exitos = 0;
    int tiradas = 0;
    probAnterior = 0;
    probAnterior = -1;

    while (!this.Converge(prob, probAnterior))
    {
        int dado1 = this.ArrojarDado();
        int dado2 = this.ArrojarDado();

        if (dado1 + dado2 == suma) {
            exitos++;
        }
        tiradas++;

        probAnterior = prob;
        prob = (float) (exitos) / tiradas;
    }
}
```

