Programación probabilística y problemas inversos en ecuaciones diferenciales ordinarias

Simulación de sistemas
Escuela de Gobierno y Transformación Pública, ITESM

6 de julio de 2022

Escuela de Gobierno y Transformación Pública
Tecnológico de Monterrey

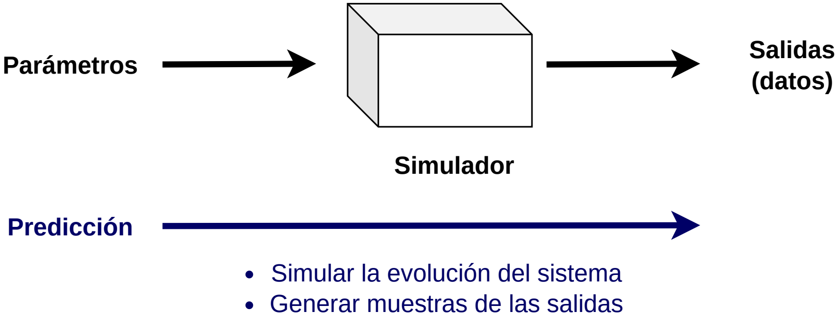


Figura: Adaptado de Atılım Güneş Baydin, *Probabilistic Programming for Inverse Problems in the Physical Sciences*. Recurso : https://www.youtube.com/watch?v=E3_Ey0z068o

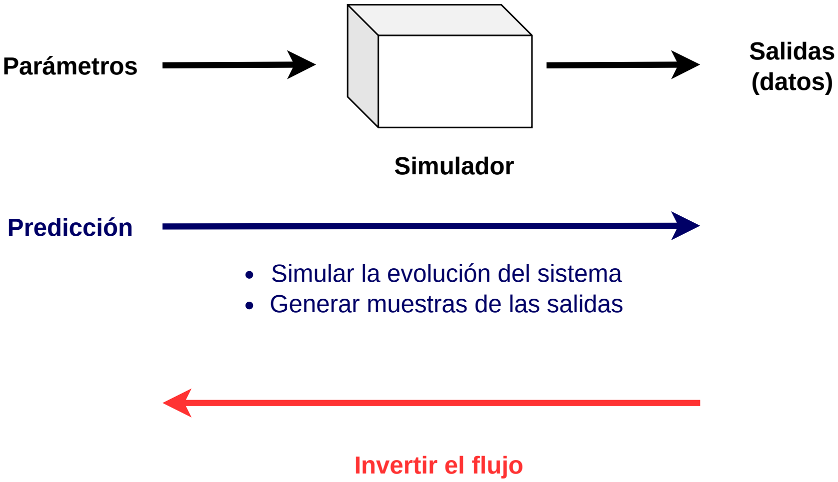
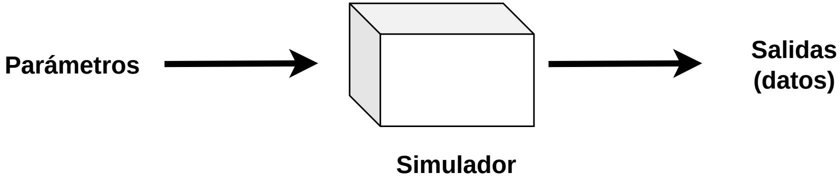


Figura: Adaptado de Atılım Güneş Baydin, *Probabilistic Programming for Inverse Problems in the Physical Sciences*. Recurso : https://www.youtube.com/watch?v=E3_Ey0z068o



Predicción

- Simular la evolución del sistema
- Generar muestras de las salidas

Inferencia

- **Encontrar los parámetros que producen (explican) los datos observados.**
- **Problema inverso.**

Figura: Adaptado de Atılım Güneş Baydin, *Probabilistic Programming for Inverse Problems in the Physical Sciences*. Recurso : https://www.youtube.com/watch?v=E3_Ey0z068o

Reconstrucción de redes

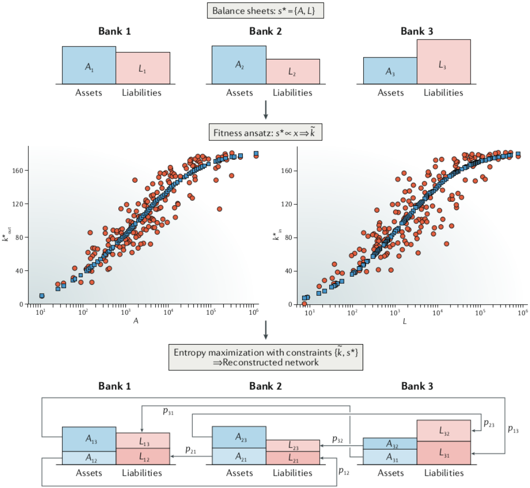


Figura: Tomado de Cimini, G., Squartini, T., Garlaschelli, D. & Gabrielli, A Systemic risk analysis on reconstructed economic and financial networks. Sci. Rep. 5, 15758 (2015).

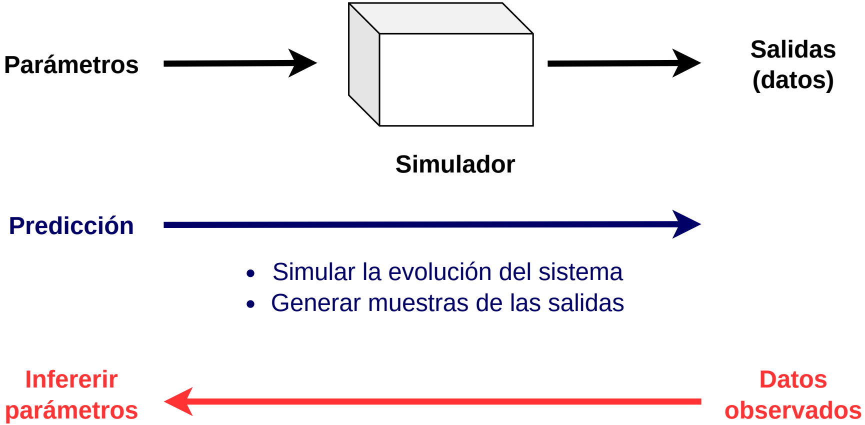


Figura: Adaptado de Atılım Güneş Baydin, *Probabilistic Programming for Inverse Problems in the Physical Sciences*. Recurso : https://www.youtube.com/watch?v=E3_Ey0z068o

Programación probabilística

- Es un enfoque de aprendizaje de máquinas que permite escribir programas que representan modelos probabilísticos.
- La programación probabilística soporta declaraciones probabilísticas como declaración de variables aleatorias.
- Permite la inferencia (bayesiana) de parámetros condicionados en datos observados.

```
@model gdemo(x) = begin
  s ~ InverseGamma(2,3)
  m ~ Normal(0,sqrt(s))

  for i=1:lengh(x)
    x[i] ~ Normal(m, sqrt(s))
  end

end
```

Frameworks para programación probabilística

(a) Pyro

(b) Stan

Turing.jl

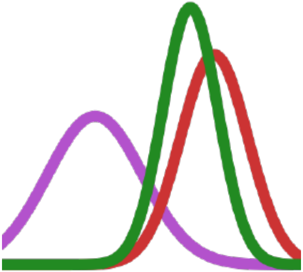


Figura: https://turing.ml/stable/

Julia

¿Por qué otro lenguaje de alto nivel?

Características típicas

- Tipado dinámico.
- Sintaxis de alto nivel.
- Open source.
- Built-in package manager.

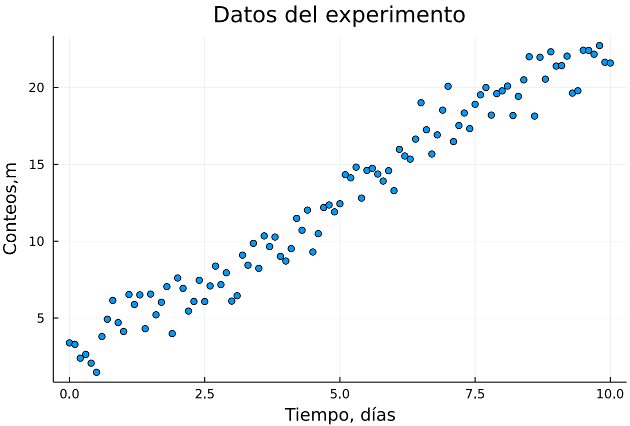
Características novedosas

- Buen perfomance: tiempo de ejecución rápido (Celeste.jl : Cataloging the Visible Universe Through Bayesian Inference at Petascale in Julia).
- Compilación JIT (Just in Time) a nivel de función.
- En un porcentaje alto, Julia está escrito en Julia.
- Metaprogramación (macros).

Figura: https://www.youtube.com/watch?v=KVnj4xuktpw&t=10s

Ecuaciones diferenciales ordinarias[1]

Realizamos experimentos donde se inoculan placas de agar con bacterias en el tiempo 0.



[1]Tomado de las notas de Ben Lampert sobre inferencia bayesiana https://ben-lambert.com/bayesian-short-course/

- Queremos modelar el crecimiento de una población de bacterias.
- Se asume el siguiente modelo de crecimiento poblacional,

$$\frac{dN}{dt} = \alpha N(1 - \beta N)$$

donde $\alpha > 0$ es la tasa de crecimiento debida a la división celular y $\beta > 0$ mide la reducción en la trasa de crecimiento debido a la aglomeración.

¿Cómo inferimos los parámetros de este modelo?

Asumamos un error de medición alrededor del valor verdadero:

$$N^*(t) \sim \text{normal}(N(t), \sigma)$$

donde:

- $N^*(t)$ es el conteo de bacterias en el tiempo t.
- $N(t)$ es la solución de la ODE en el tiempo t (número verdadero de bacterias en la placa).
- $\sigma > 0$ mide la magnitud del error de medición alrededor del valor verdadero.

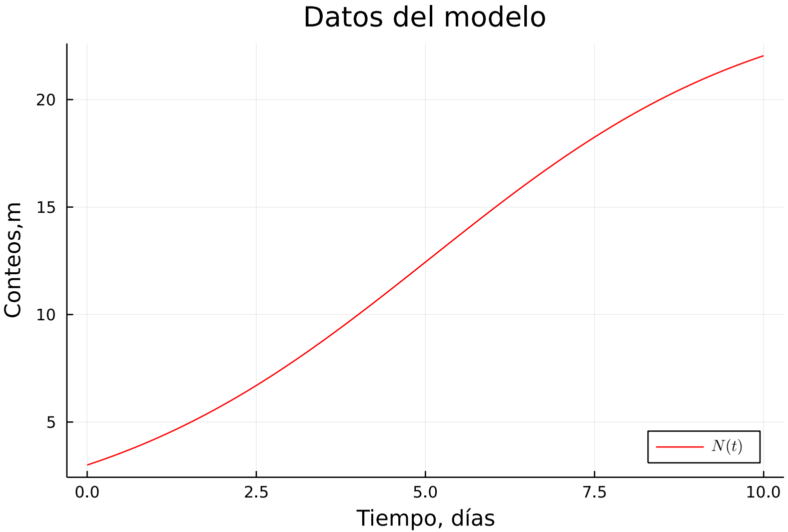
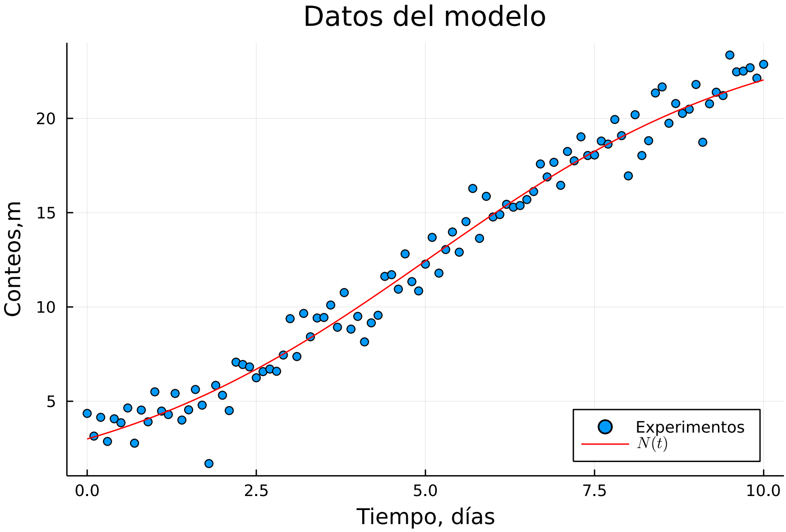


Figura: Número verdadero de bacterias, $N(t)$

Superposición de distribución de muestreo que representa el error de medición y de los datos experimentales.



¿Cómo calculamos $N(t)$?

$$\frac{dN}{dt} = \alpha N(1 - \beta N)$$

Cualquier solución de $N(t)$ -exacta o numérica- depende de los parámetros del modelo de ODE. En este caso,

$$N(t) = f(t, \alpha, \beta)$$

¿Cómo integramos los datos observados para estimar estos parámetros?

Cualquier solución de $N(t)$ -exacta o numérica- depende de los parámetros del modelo de ODE. En este caso,

$$N(t) = f(t, \alpha, \beta)$$

¿Cómo integramos los datos observados para estimar estos parámetros?

Inferencia Bayesiana

Cualquier solución de $N(t)$ -exacta o numérica- depende de los parámetros del modelo de ODE. En este caso,

$$N(t) = f(t, \alpha, \beta)$$

¿Cómo integramos los datos observados para estimar estos parámetros?

Inferencia Bayesiana

La única receta de la inferencia bayesiana...

consiste en encontrar la distribución condicional de todas aquellas cantidades de interés cuyo valor desconocemos dado el valor conocido de las variables observadas.

Inferencia bayesiana

- En un contexto de inferencia bayesiana, donde se tiene un modelo muestral $Y \sim p(y|\theta)$ y una distribución inicial $p(\theta)$, la distribución final se calcula de la siguiente forma:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{\int p(\theta')p(y|\theta')d\theta'}$$

- Aún descartando el denominador (el cual se puede interpretar como un factor de escalamiento que hace que la función integre a 1), el cálculo analítico de $p(\theta|y)$, en la mayoria de las ocasiones, es complicado.

- Hay distintas técnicas para calcular $p(\theta|y)$ (aproximación de Laplace, métodos de cuadratura, etc), pero por la capacidad de cómputo disponible las técnicas de simulación de Monte Carlo vía cadenas de Markov (MCMC) han sido mayormente utilizadas.
- La idea de MCMC es **construir una cadena de Markov que sea fácil de simular (a través de un proceso de muestreo) y cuya distribución de equilibrio corresponda a la distribución final de interés**.

- En el contexto de nuestro problema,
  - $p(\theta)$ representa la distribución inicial de los parámetros del modelo de ODE,
  - $p(y|\theta)$ (el modelo muestral o distribución de verosimilitud) representa el modelo de ODE.
- Podemos utilizar las técnicas de MCMC para construir la distribución $p(\theta|y)$, de los parámetros del modelo de ODE.

Método de Monte Carlo

Idea: Método para calcular el área bajo una curva. Es una solución estadística al problema de integración.

Supongamos que existe $M > 0$ tal que $0 \leq f(\theta) \leq M$ para todo $\theta \in [a, b]$ y que queremos calcular la integral

$$I = \int_a^b f(\theta) d\theta \tag{1}$$

el valor de la integral es el área bajo la curva $\phi = f(\theta)$ para $\theta \in [a, b]$. Dicha gráfica queda inscrita en el rectángulo $R = [a, b] \times [0, M]$.

Método de Monte Carlo

Sea

$$p(\theta, \phi) = \frac{1}{M(b-a)} I_R(\theta, \phi) \tag{2}$$

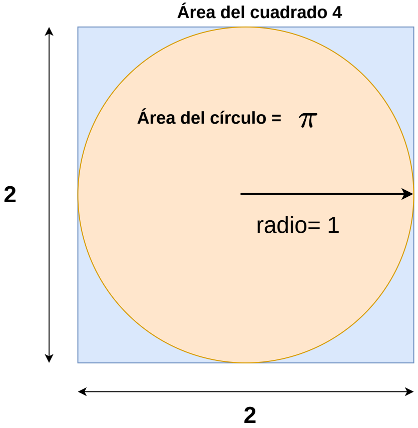
Entonces $p(\theta, \phi)$ corresponde a la función de densidad de una distribución uniforme sobre el rectángulo R. La integral I puede entonces estimarse simulando una muestra $(\theta_1, \phi_1), \ldots, (\theta_N, \phi_N)$ de $p(\theta, \phi)$ y contando cuántos de estos valores caen bajo la curva $\phi = f(\theta)$.

El estimador $\hat{I}$ obtenido es un estimador insesgado de I.

La varianza del estimador es

$$Var(\hat{I}) = \frac{I}{N}\{M(b-a) - I\}$$

Cálculo de π por el método de Monte Carlo (Wilkinson and Allen, 1998)



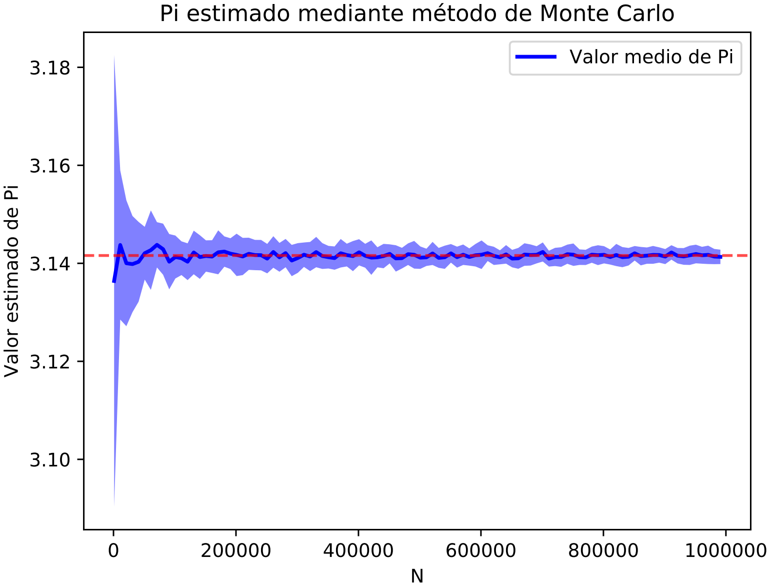
$$\frac{\text{Área del círculo}}{\text{Área del cuadrado}} = \frac{\pi(1)^2}{4} = \frac{\pi}{4}$$

que puede ser descrita por la integral

$$\int_0^1 \sqrt{1 - x^2} dx = \frac{\pi}{4}$$

Para calcular π se generan parejas aleatorias de números (x_r, y_r), distribuidos uniformemente entre 0 y 1, y contamos cuántos de estos puntos caen dentro del círculo, esto es, si se cumple la igualdad

$$y_r^2 + x_r^2 \leq 1$$


Pi estimado mediante método de Monte Carlo
Valor medio de Pi
3.18
3.16
3.14
3.12
3.10
Valor estimado de Pi
0
200000
400000
600000
800000
1000000
N

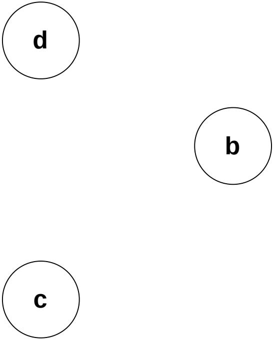
Cadenas de Markov

Definition

Cadena de Markov. Proceso estocástico discreto con estados finitos que cumple la propiedad de Markov.

$$P(X_{t+1} = s | X_1 = s_1, \ldots, X_t = s_t) = P(X_{t+1} = s | X_t = s_t)$$

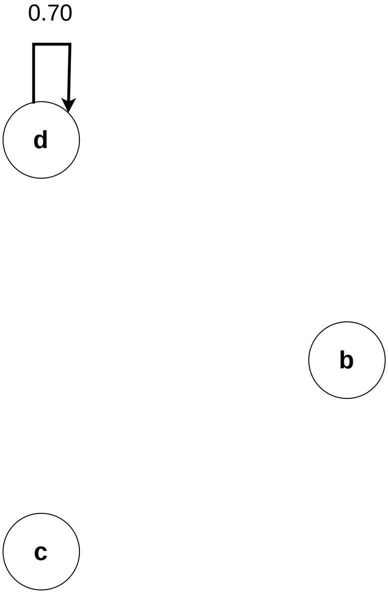
Cadenas de Markov



Estados:

d: Dormitorio
b: Bar
c: Comedor

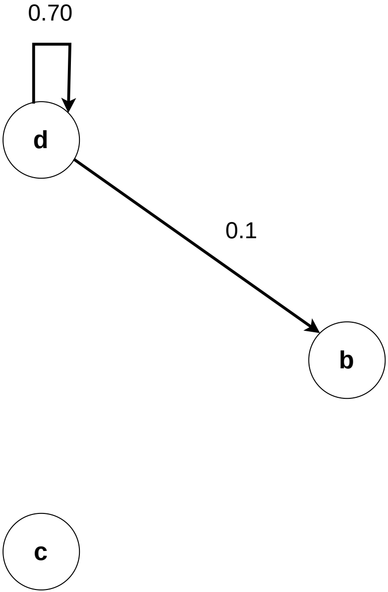
Ejemplo tomado de



La arista representa la posibilidad que un pasajero que está en el dormitorio en el tiempo t continúe en el dormitorio en el tiempo $t+1$.

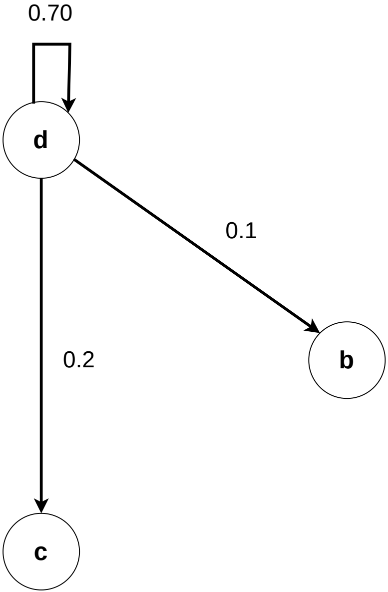
Se etiqueta con la probabilidad de que esto suceda (**Probabilidad de transición**)

$$P(X_{t+1} = d \mid X_t = d) = 0{,}7$$



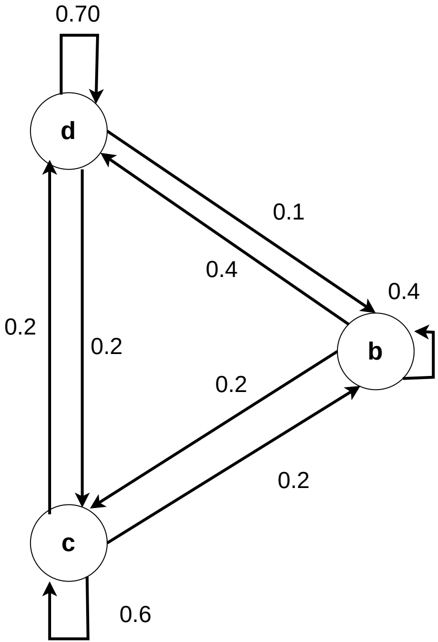
Arista para la transición del dormitorio al bar

$$P(X_{t+1} = b \mid X_t = d) = 0{,}1$$

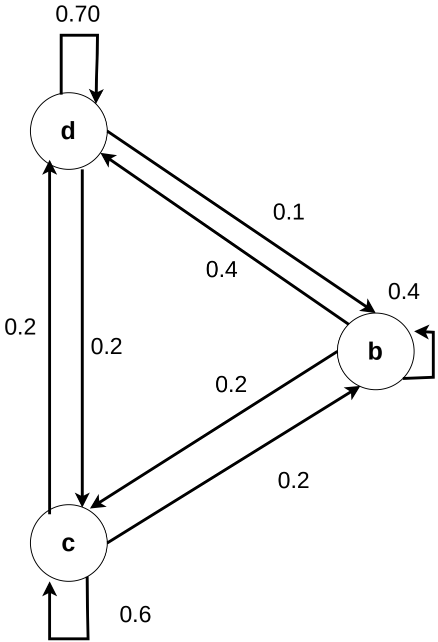


Arista para la transición del dormitorio al comedor

$$P(X_{t+1} = c \mid X_t = d) = 0{,}2$$



Agregamos todas las transiciones posibles y sus probabilidades.

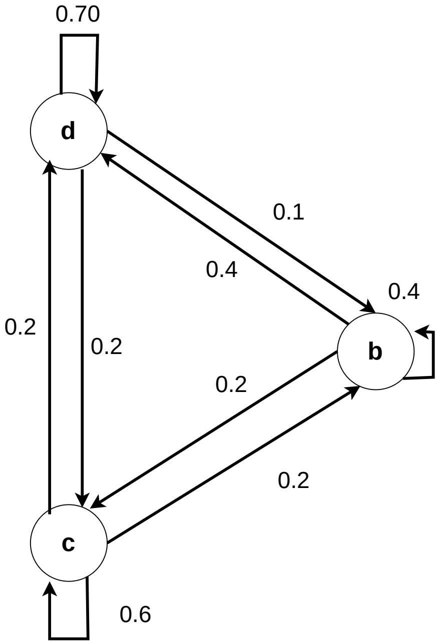


Las probabilidades de transición pueden representarse de manera matricial:

$$T = \begin{bmatrix} 0{,}7 & 0{,}4 & 0{,}2 \\ 0{,}1 & 0{,}4 & 0{,}2 \\ 0{,}2 & 0{,}2 & 0{,}6 \end{bmatrix} \tag{3}$$

con entradas

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



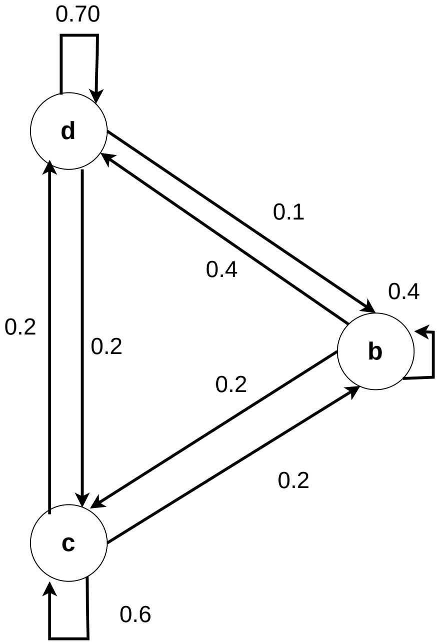
Matriz de probabilidad de estado:

$$T = \begin{bmatrix} 0{,}7 & 0{,}4 & 0{,}2 \\ 0{,}1 & 0{,}4 & 0{,}2 \\ 0{,}2 & 0{,}2 & 0{,}6 \end{bmatrix}$$

La dinámica para la distribución de probabilidad puede expresarse:

$$\begin{bmatrix} P(X_{t+1} = d) \\ P(X_{t+1} = b) \\ P(X_{t+1} = c) \end{bmatrix} = T \begin{bmatrix} P(X_t = d) \\ P(X_t = b) \\ P(X_t = c) \end{bmatrix} \tag{4}$$

$$P(X_{t+1}) = TP(X_t) \tag{5}$$



Si sabemos que el pasajero se encuentra en el dormitorio al tiempo $t = 0$:

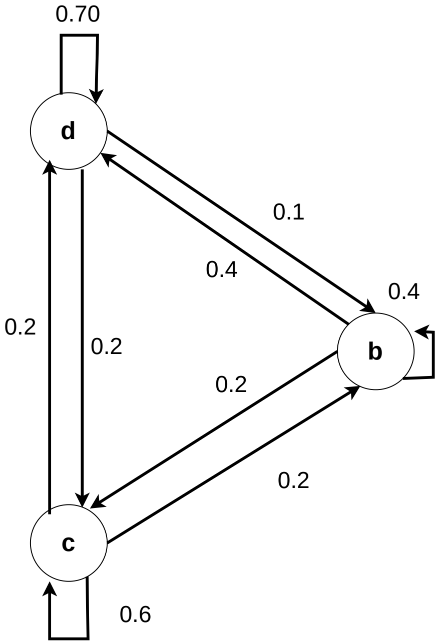
$$P(X_0) = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = P_0$$

Al tiempo siguiente la distribución es:

$$P(X_1) = TP_0 = P_1$$

$$P(X_1) = \begin{bmatrix} 0{,}7 & 0{,}4 & 0{,}2 \\ 0{,}1 & 0{,}4 & 0{,}2 \\ 0{,}2 & 0{,}2 & 0{,}6 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$P(X_1) = \begin{bmatrix} 0{,}7 \\ 0{,}1 \\ 0{,}2 \end{bmatrix}$$



Para tiempos futuros

$$P(X_t) = TP(X_{t-1}) \equiv P_t$$

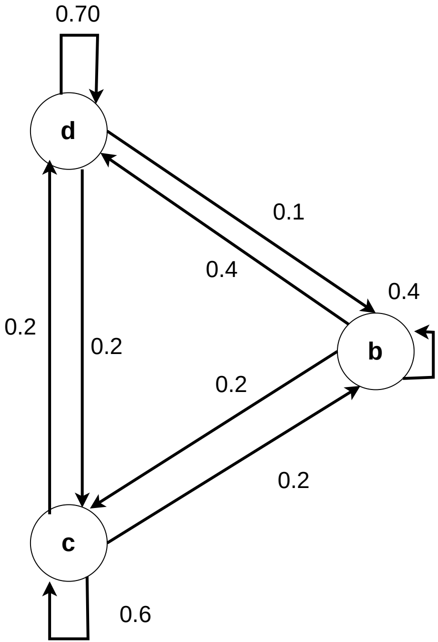
$$P1 = TP_0$$

$$P2 = TP_1$$

$$\begin{bmatrix} 0,7 & 0,4 & 0,2 \\ 0,1 & 0,4 & 0,2 \\ 0,2 & 0,2 & 0,6 \end{bmatrix} \begin{bmatrix} 0,7 \\ 0,2 \\ 0,1 \end{bmatrix}$$

$$= \begin{bmatrix} 0,57 \\ 0,15 \\ 0,28 \end{bmatrix}$$

$$TTP_0 = T^2 P_0$$



Para tiempos futuros

$$P_t = T^t P_0$$

Algoritmo de Metropolis (Theodoridis, 2020)

- La distribución propuesta cambia con el tiempo siguiendo la evolución de una cadena de Markov.
- La cadena se construye de manera que su matriz de transición tenga la distribución deseada $p(x)$ la cual es invariante.
- La distribución propuesta depende del valor del estado previo,x_{n-1}, esto es, $q(\cdot|x_{n-1})$.
- Es decir, generar una nueva muestra (un nuevo estado) depende del valor del estado previo.

Algorithm 1: Algoritmo Metropolis (Theodoridis, 2020)

Sea la distribución deseada $p(\cdot)$
Escoge una distribución de propuesta $q(\cdot|\cdot)$
Escoge el valor del estado inicial x_0

for $n = 1, 2, \ldots, N$ **do**
    Toma un valor $x \sim q(\cdot|x_{n-1})$
    Calcula el valor de aceptación
    /* Si la probabilidad de $p(x)$ es más grande que la de $p(x_{n-1})$, entonces se acepta la nueva muestra. En caso contrario, esta es aceptada-rechazada en base a su valor relativo */

$$\alpha(x|x_{n-1}) = \text{mín}\left\{1, \frac{p(x)}{p(x_{n-1})}\right\}$$

    Escoge $u \sim U(0, 1)$

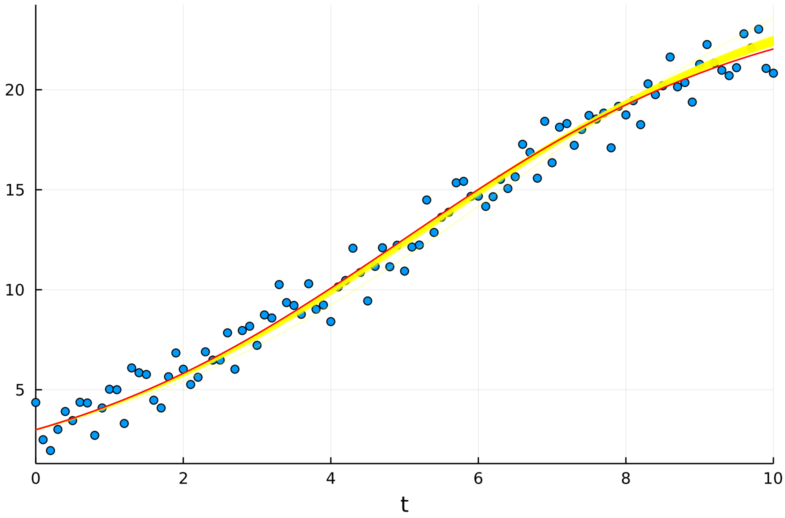
    if $u \leq \alpha(x|x_{n-1})$ **then**
        $x_n = x$
    else
        $x_n = x_{n-1}$
    end if
end for

Turing

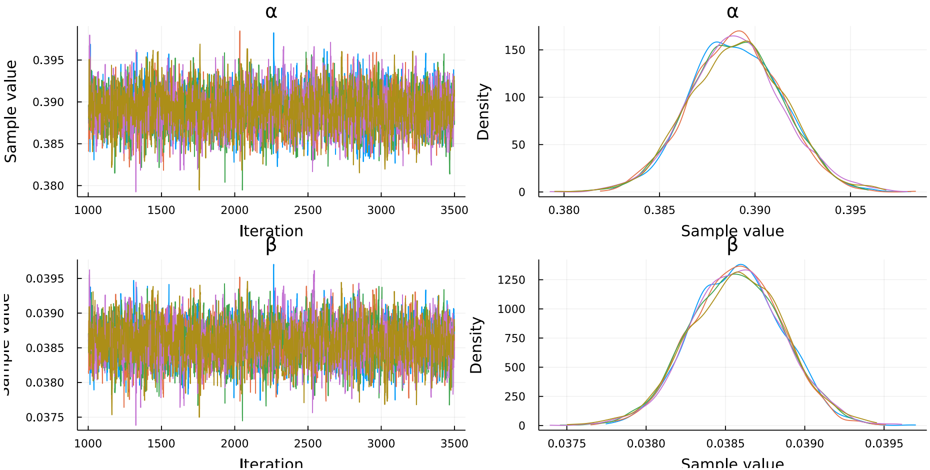
Turing es un lenguaje de programación probabilística de propósito general para una inferencia bayesiana robusta y eficiente. Las características actuales incluyen:

- Programación probabilística de propósito general con una interfaz de modelado intuitiva;
- Muestreo de Monte Carlo hamiltoniano (HMC) robusto y eficiente para distribuciones posteriores diferenciables;
- Muestreo de partículas MCMC para distribuciones posteriores complejas que involucran variables discretas y flujo de control estocástico; e
- Inferencia composicional a través del muestreo de Gibbs que combina partículas MCMC, HMC y paseo aleatorio MH (RWMH).

¡Vamos a programar!



La distribución posterior (línea amarilla) reproduce con bastante precisión la solución *verdadera* del ODE.


α
Sample value
0.395
0.390
0.385
0.380
1000
1500
2000
2500
3000
3500
Iteration
α
Density
150
100
50
0
0.380
0.385
0.390
0.395
Sample value
β
0.0395
0.0390
0.0385
0.0380
0.0375
1000
1500
2000
2500
3000
3500
Iteration
β
Density
1250
1000
750
500
250
0
0.0375
0.0380
0.0385
0.0390
0.0395

References I

Theodoridis, S. (2020). *Machine Learning. A Bayesian and Optimization Perspective*. Academic Press, second edition edition.

Wilkinson, B. and Allen, M. (1998). *Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers*. Prentice-Hall, Inc., USA.