Actividad:

Elaborar un ensayo sobre el algoritmo de Gibbs

Desarrollo:

Entender la regulación de genes a nivel de sistema es uno de los problemas más interesantes y retadores en biología, pero uno en el cual los principios, aún están por descubrir. Los algoritmos probabilísticos como EM[1] y muestreo de Gibbs desempeñan un papel esencial en la búsqueda de motivos, es por esta razón que programas como MEME y CONSENSUS los incorporan[2].

El muestreo de Gibbs, conocido también como el método "heatbath", es el más sencillo de los algoritmos de MCMC[3], el cual puede ser aplicado en un amplio rango de situaciones, especialmente cuando las probabilidades condicionales $P(x_i \mid x_j : j \neq i)$ pueden ser fácilmente calculadas cuando las variables X_i toman valores de un conjunto pequeño.

En el algoritmo de Gibbs se muestrea de manera iterativa cada variable, condicionada al valor más reciente de todas las demás variables y lo que se obtiene al aplicarla es una aproximación a la distribución de probabilidad conjunta a la cual corresponde al problema. Es así como lo podemos utilizar para dar solución al problema de no tener la distribución de probabilidad conjunta entre las variables definida, mediante la generación de números aleatorios de dichas funciones de densidad multivariadas.

Es un caso especial del algoritmo de Metropolis[4]. Para entender mejor el concepto, podemos recurrir al caso bivariado, en el cual tenemos como entradas las dos distribuciones condicionales y un valor de x_0 y elegimos la cantidad de veces que vamos a iterar el algoritmo (n).

A continuación se presenta código en R para la ejecución de dicho algoritmo para el caso bivariado para distribuciones condicionales gamma y normal:

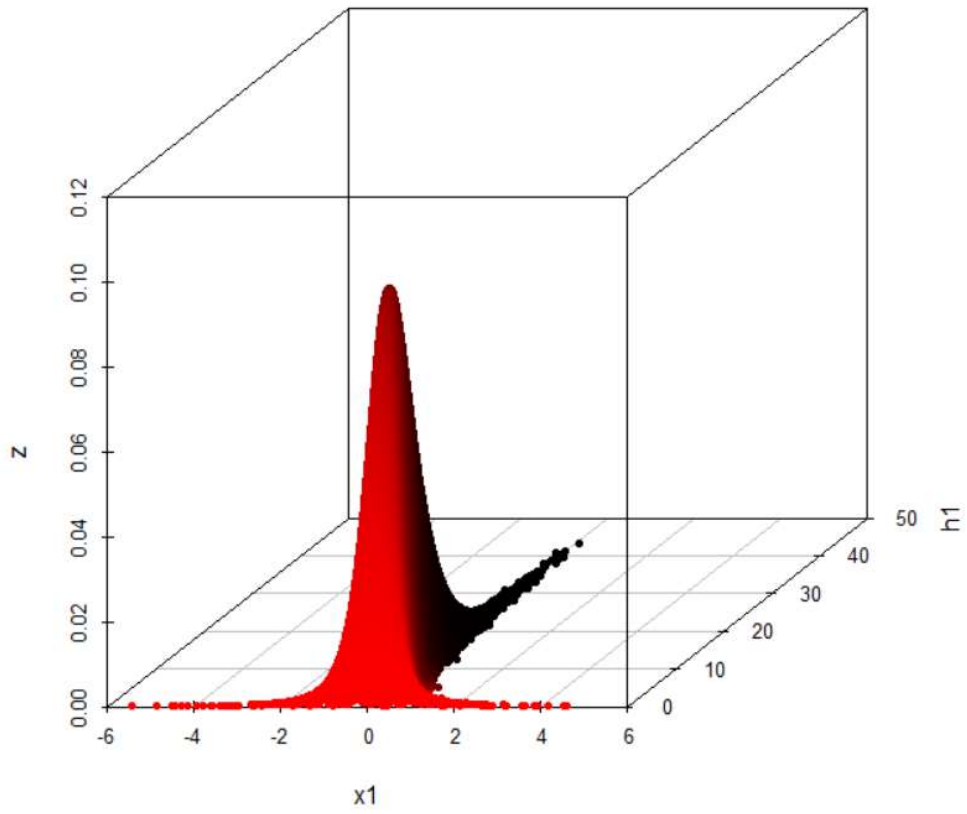
Código[5]:

```
Gibbs1 = function(n, ini , a, b, mu, l){
```

[1] Expectation-maximization
[2] Baldi, Pierre & Brunak, Søren. (2001). Bioinformatics: The Machine Learning Approach.
[3] Modelo de Montecarlo basado en Cadenas de Markov
[4] https://www.youtube.com/watch?v=ER3DDBFzH2g
[5] Adaptado de : https://www.youtube.com/watch?v=vdd29CyTMp0

```
  x = numeric()
  h = numeric()
  x[1] = ini
  for (i in 1:n){
    h[i] = rgamma(1, shape = a, scale = b)
    x[i+1] = rnorm(1, mu, 1/sqrt(l*h[i]))
  }
  list(aleatorio_h = h, aleatorio_x = x)
}
a = 2
b = 3
mu = 0
l = 4
n = 200000

resultado = Gibbs1( n, 0.45 , a, b, mu, l)
windows()
par(mfrow = c(2,1))
hist(resultado$aleatorio_x)

mean(resultado$aleatorio_h)
mean(resultado$aleatorio_x)

library(coda)
library(scatterplot3d)

MC1 = as.mcmc(resultado$aleatorio_h)
plot(MC1)
heidel.diag(MC1, pvalue=0.05)

MC2 = as.mcmc(resultado$aleatorio_x)
plot(MC2)
heidel.diag(MC2, pvalue=0.05)

f = function(x,h,a,b,l,mu){
  1 / (b^a*gamma(a))*sqrt(1/(2*pi))*h^(a-1/2)*exp(-h/b-(l*h*(x-mu)^2)/2)
}
```

```
x1 = resultado$aleatorio_x[-1]
h1 = resultado$aleatorio_h
z = numeric()
for (i in 1:n){
  z[i] = f(x1[i], h1[i], a, b, l, mu)
}
windows()
scatterplot3d(x1, h1, z, highlight.3d = T, pch = 20)
```



En la gráfica anterior podemos observar en el eje de las z la variable z, la cual corresponde a nuestro “problema”: distribución de probabilidad conjunta desconocida, conformada por las funciones de densidad multivariadas h (gamma) y x (Normal).

Así habremos construido la distribución de probabilidad conjunta desconocida.