Razonamiento bajo incertidumbre

Tarea 6. Aplicación del Gibb Sampling con distibuciones gaussianas multivariables en imagenes

Ángel García Báez

2024-11-27

Índice

1 Instrucciones:

Dada una imagen digital que va a ser denotada por $\vec{x}$ que contenga "piel", un rostro, o piel de un cuerpo:

1- A partir de muestras de color de piel, determinar el aprendizaje encontrado de una distribución Gaussiana Multivariada en 3D. Reportar el vector de medias y la matriz de varianzas y covarianzas.

2.- Aplicar el muestreo de Gibbs para al menos 3 puntos iniciales y graficar 30 muestreos de la trayectoria. Al Pixel ponerle el color de las coordenadas RGB de su evaluación.

3.- Aplicar el muestreo de Gibbs en 100 puntos iniciales y para cada uno, usar 30 muestras de su trayectoria. Se gráfica el punto FINAL de cada trayectoria.

4.- La selección de los puntos iniciales a priori de la distribución quedan a la creatividad de cada quien.

2 Definiciones necesarias antes de comenzar

2.1 Vector de medias:

El vector de medias para una matriz sera definido como un vector fila de tamaño $1 \times P$ donde P es la cantidad de columnas que tenga la matriz de la que se quiere obtener.

$$\bar{x} = [\bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_p]$$

2.2 Matriz de Varianzas y Covarianzas

La forma de calcular la matriz de varianzas y covarianzas de la matriz de datos, puede resumirse en la siguiente expresión:

$$\Sigma = \frac{1}{n-1}(X^T X - n\bar{x}^T \bar{x})$$

Donde:

Σ = Matriz de varianzas y covarianzas
X = Matriz de datos
X^T = Matriz de datos transpuesta
n = Filas o casos de la matriz
$\bar{x}$ = Vector fila de las medias
$\bar{x}^T$ = Vector fila de las medias transpuesto

2.3 Distribución de probabilidad normal multivariante

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{P/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\bar{x})^T \Sigma^{-1}(x-\bar{x})\right)$$

Donde:

P = Cantidad de variables.
X = Vector de datos de tamaño $1 \times P$.
$\bar{x}$ = Vector de medias de tamapo $1 \times P$.
Σ = Matriz de varianzas y covarianzas.
Σ^{-1} = Inversa de la matriz de varianzas y covarianzas.
$|\Sigma|$ = Determinante de la matriz de varianzas y covarianzas.

3 Introducción

Para el presente reporte se pidió seguir los lineamientos expresados al inicio, antes de poder llevarlos a cabo, es necesario abordar las matemáticas que hay detrás para poder hacer esto posible.

3.1 Explicación del Gibbs Sampler (Muestreo de Gibbs)

El algoritmo de Gibbs Sampler es una tecnica de muestreo utilizada en estadística y machine learning para generar muestras de una distribución de probabilidad conjunta cuando por alguna razon, el muestrearla directamente no es posible dada su complejidad o alguna limitante. Dicha tecnica se basa principalmemte en métodos bayesianos y en simulaciones montecarlo para operar sobre modelos que resultan complejos.

La tecnica del muestreador de Gibbs es un caso especial de la tecnica conocida como métropolis Hasting. Dicho algoritmo es un caso especial de los métodos de Montecarlo con Cadenas de Markov (MCMC) cuyo objetivo es generar muestras de una distribución conjunta $p(x_1, x_2, \ldots . x_n)$ dividiendo el problema en muestreos de las distribuciones condicionales que resultan ser más manejables de la forma $p(x_i, x_{-i})$ donde x_{-i} denota el conjunto de variables excepto x_i.

4 Funcionamiento del algoritmo

Para la correccta implementación el algoritmo de Gibbs, es necesario seguir los pasos descritos a continuación.

1.- Inicialización del algoritmo: Se elige un valor arbitrario o punto de partida desde donde se quiera iniciar el funcionamiento del algoritmo. Suponiendo que sea $p(x)$ la distribución del problema que queremos muestrear y $x = (x_1, x_2, \ldots . x_n)$ el vector de variables aleatorias, se denota al punto inicial como $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \ldots . x_n^{(0)})$

2.- Iteraciones: Una vez inicializado el algoritmo, se actualizan progresivamente los valores dados inicialmente mediante un muestreo de las distribuciones condicionales $p(x_i, x_{-i})$ de forma ordenada. Supongase que se quiere calcular la t-ésima iteración con $x^{(t-1)} = (x_1^{(t-1)}, x_2^{(t-1)}, \ldots . x_n^{(t-1)})$

Entonces, se procede a hacer el muestreo de las condicionales de forma ordenada:

Muestrear $x_1^t \sim p(x_1 | x_2^{t-1}, \ldots , x_n^{t-1})$

Una vez actualizado dicho valor, se procede a muestrear x_2^t tal que:

$$x_2^t \sim p(x_2 | x_1^t, x_3^{t-1}, \ldots , x_n^{t-1})$$

Dicho proceso se repite hasta actualizar todas las variables x_i.

3.- Convergencia: Despues de un periodo suficiente de iteraciones (llamado burn-in period), las muestras generadas se distribuyen aproximadamente segun la distribución conjunta $p(x_1, x_2, \ldots , x_n)$.

5 Imagenes a utilizar y forma de abordar el problema

A continuación se muestra una selección de 3 imágenes donde aparecen 3 personas de distintas características mostrando partes de su piel como el rostro y los brazos.

Imagen 1	**Imagen 2**	**Imagen 3**
		
768x1152 = 884,736 px	266x320 = 85,120 px	768x1024 = 786,432 px

Table 1: Imagenes para el desarrollo del ejercicio

Las 3 imágenes tienen características completamente distintas, tanto en composición de colores como en resolución (tamaño), el objetivo es extraer la información de partes de la piel de las imagenes para "entrenar" una distribución normal multivariable para cada imagen con la finalidad de poner en practica el funcionamiento del gibb sampler.

Para ello, se describe a continuación el proceso.

Paso 1: Cargar una imagen en lenguaje R y descomponerla en 3 vectores (1 por cada canal de color R,G y B) que posteriormente serán unidos en un solo dataframe, al cual ademas se le hará el etiquetado de las coordenadas de cada pixel (fila del dataframe) para su posterior reconstrucción y manipulación.

Paso 2: Se extrajo una sección de la imagen (el rostro) la cual contiene pixeles de piel, de los cuales se seleccionaron 10 pixeles al azar de dicho subconjunto de datos con la finalidad de obtener el vector de medias y la matriz de varianzas y covarianzas de cada imagen.

Paso 3: Crear una distribución normal multivariante para los pixeles de la piel con el vector de medias y matriz de covarianzas calculados a partir de la muestra de 10 pixeles de piel.

Paso 4: Se define la distribución condicional de los puntos de la normal multivariable. Para ello se encontro que la media condicional para estimar la $i-$ésima variable aleatoria de la distribución, resulta que la distribución condicional para una variable de la distribución multivariada, se puede convertir en una distribución normal univariada, cuyos parametros de media y varianza se pueden expresar a continuación:

$$\mu_i^{condicional} = \mu_i + \Sigma_{ij}\Sigma_{jj}^{-i}(x_{-i}, \mu_{-i})$$

$$\sigma_i^{2,condicional} = \Sigma_{ii} - \Sigma_{ij}\Sigma_{jj}^{-i}\Sigma_{ij}^{T}$$

Paso 5: Se define un punto de inicio para el muestreador de Gibbs que va a ir calculando el valor de la variable x_i acorde una normal univariada que tiene los parámetros anteriormente descritos, así sucesivamente hasta estimar un valor para cada variable mediante la distribución condicional.

Paso 6: Se repite lo descrito en el punto 5 tomando como nuevo punto de inicio el punto de salida anterior, esto se repite k veces.

6 Resultados individuales

6.1 Imagen 1: Denisse Guerrero de Belanova

Para la primera foto que tiene unas dimensiones de 768 X 1152 (884,736 pixeles) se muestra a continuación la región de la piel de la cara extraída junto con la distribución de los pixeles seleccionados para ser piel.

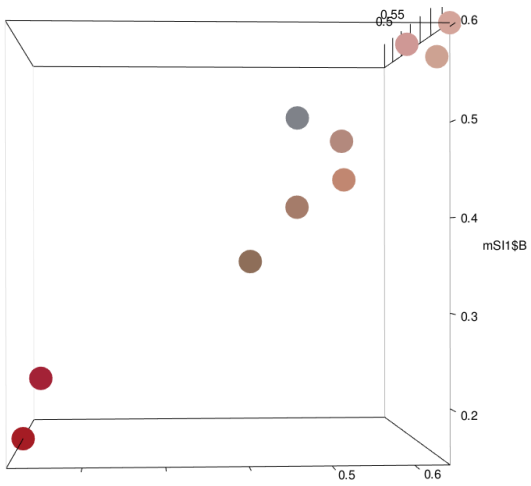

Table 2: Área de la piel.

Complementariamente, se reporta el vector de medias y la matriz de varianzas y covarianzas para la primer imagen:

$$\bar{X} = [0.6894118, 0.4611765, 0.4349020]$$

$$\Sigma = \begin{bmatrix} 0.012582511 & 0.010305780 & 0.008393011 \\ 0.010305780 & 0.036321760 & 0.029229100 \\ 0.008393011 & 0.029229100 & 0.025249947 \end{bmatrix}$$

Una vez que se tiene la media y la matriz de varianzas y covarianzas de la distribución de los pixeles de piel en la imagen, se procede a aplicar el algoritmo de gibb sampling con 30 iteraciones aplicadas a 3 puntos distintos.

Se propone que los puntos sean: $x_1 = [0, 0, 0], x2 = [1, 1, 1], x1 = [0.5, 0.5, 0.5]$ para probar como se comporta el algoritmo cuando se le pone en situaciones donde esta a los extremos de la distribución.

A continuación se muestran las trayectorias recorridas por los 3 puntos:

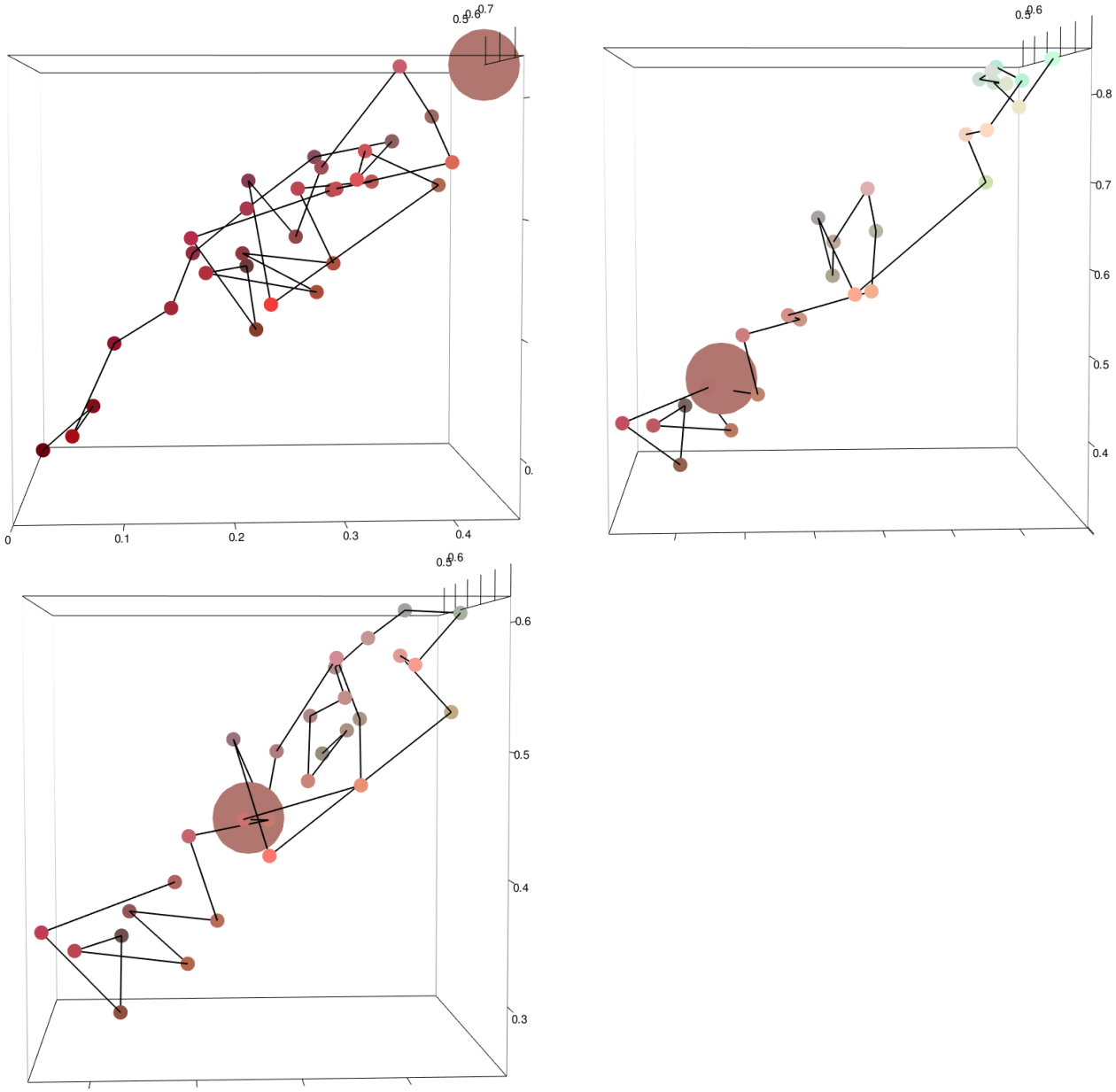

Table 3: Resultados de las trayectorias de los 3 puntos.

Posteriormente, se obtuvieron 100 valores aleatorios del rostro de la imagen para poder aplicarles el algoritmo y verificar a donde llegan al cabo de 30 iteraciones por cada punto.

Los pixeles seleccionados y el resultado de los 100 puntos se muestran en la siguiente tabla:

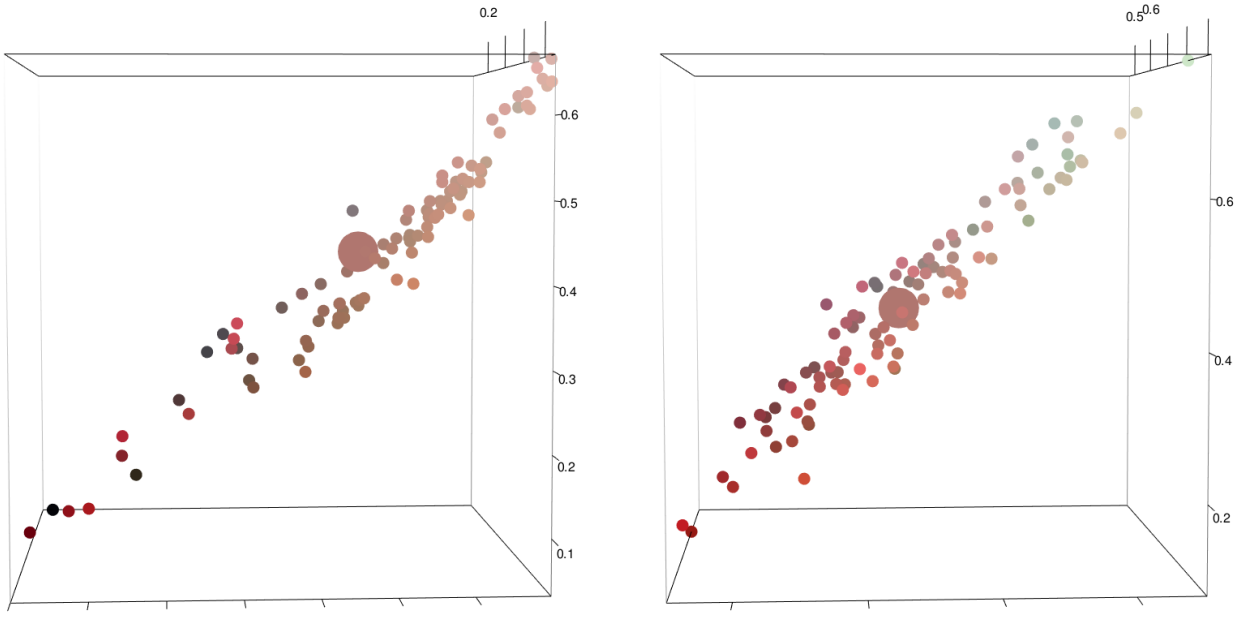

Table 4: Resultados de los 30 remuestreos a los 100 puntos.

Adicionalmente y para este unico caso, se decidio verificar el comportamiento de la media y la varianza del componente rojo de la imagen, para observar cual es su comportamiento como parámetro a lo largo del tiempo y el como va convergiendo a medida que se realizan los remuestreos y se va recalculando en cada iteración. Para ello se utilizo el punto $[1, 1, 1]$ y se realizaron 2000 muestreos de Gibbs para observar su evolución a traves de cada iteración.

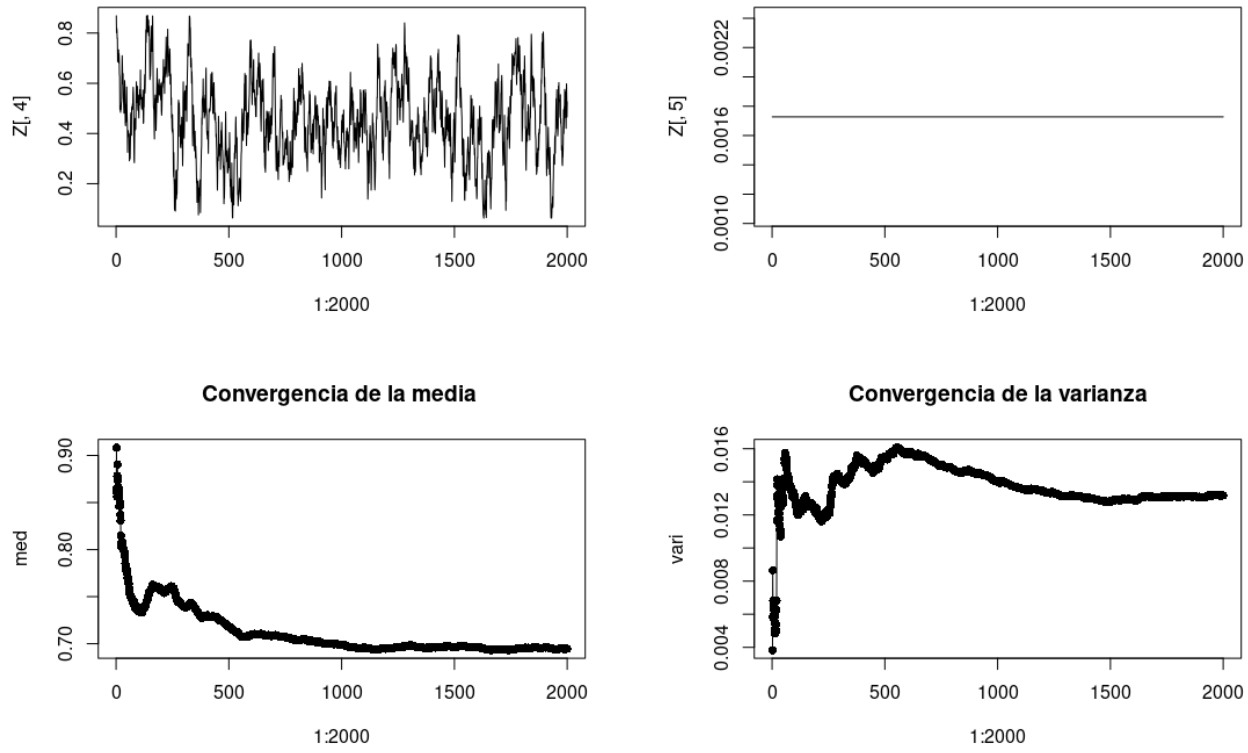

Table 5: Resultados de evaluar los parametros atraves de las iteraciones.

6.2 Imagen 2: Selena Quintanilla

Para la segunda foto que tiene unas dimensiones de 320 X 266 (85,120 pixeles) se muestra a continuación la región de la piel de la cara extraída junto con la distribución de los pixeles seleccionados para ser piel.

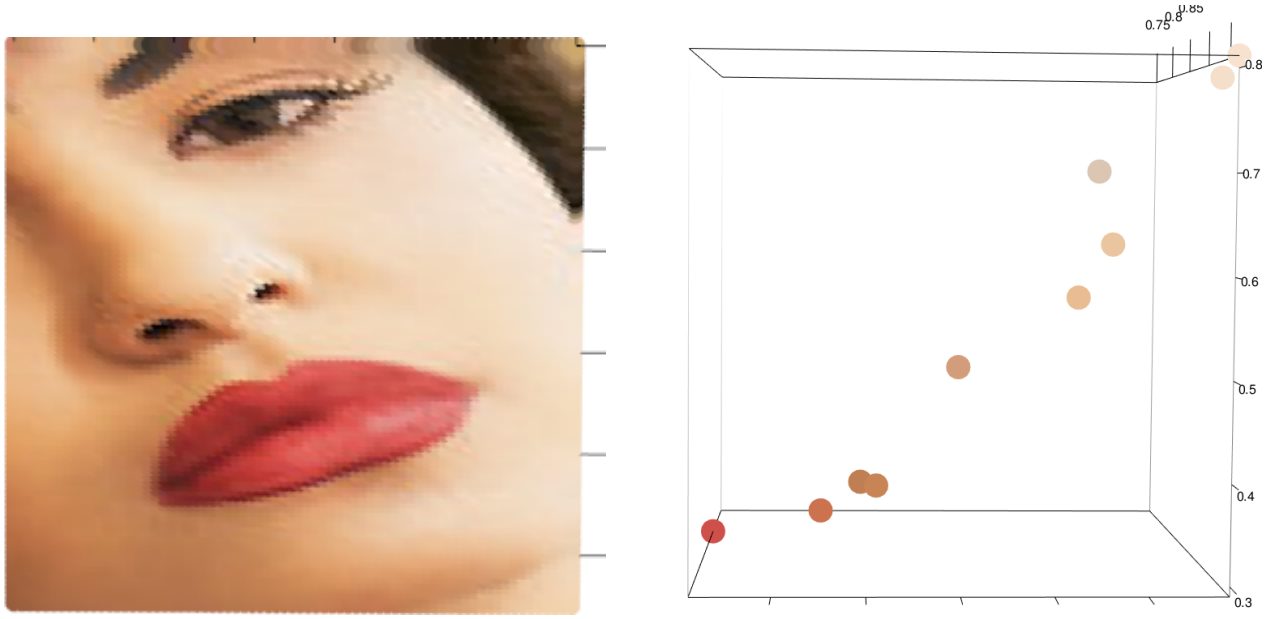

Table 6: Área de la piel.

Complementariamente, se reporta el vector de medias y la matriz de varianzas y covarianzas para la primer imagen:

$$\bar{X} = [0.8592157, 0.6439216, 0.5247059]$$

$$\Sigma = \begin{bmatrix} 0.005876201 & 0.01287163 & 0.01421710 \\ 0.012871631 & 0.03783775 & 0.03850895 \\ 0.014217096 & 0.03850895 & 0.04194216 \end{bmatrix}$$

Una vez que se tiene la media y la matriz de varianzas y covarianzas de la distribución de los pixeles de piel en la imagen, se procede a aplicar el algoritmo de gibb sampling con 30 iteraciones aplicadas a 3 puntos distintos.

Se propone que los puntos sean: $x_1 = [0, 0, 0], x2 = [1, 1, 1], x1 = [0.5, 0.5, 0.5]$ para probar como se comporta el algoritmo cuando se le pone en situaciones donde esta a los extremos de la distribución.

A continuación se muestran las trayectorias recorridas por los 3 puntos:

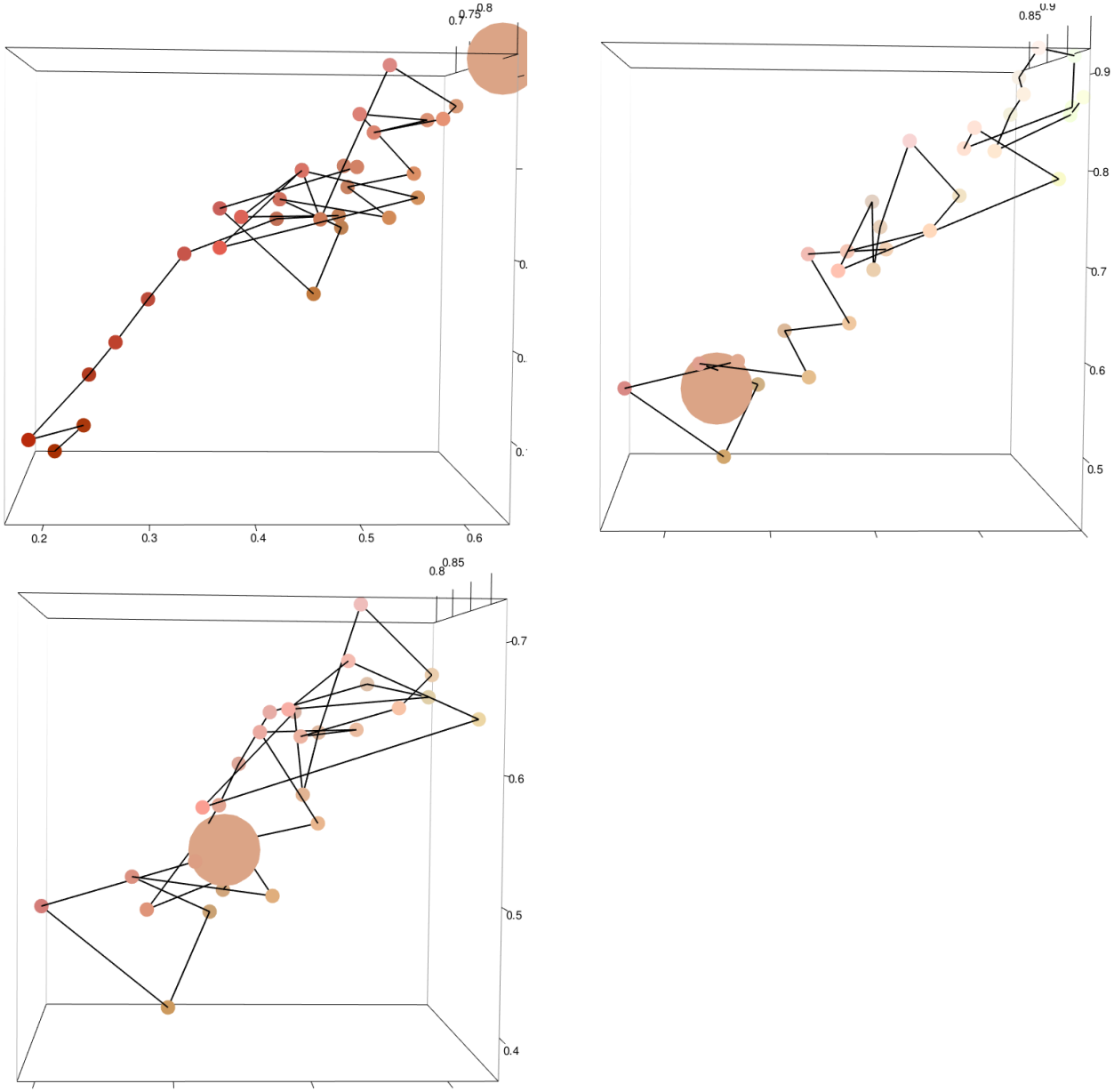

Table 7: Resultados de las trayectorias de los 3 puntos.

Posteriormente, se obtuvieron 100 valores aleatorios del rostro de la imagen para poder aplicarles el algoritmo y verificar a donde llegan al cabo de 30 iteraciones por cada punto.

Los pixeles seleccionados y el resultado de los 100 puntos se muestran en la siguiente tabla:

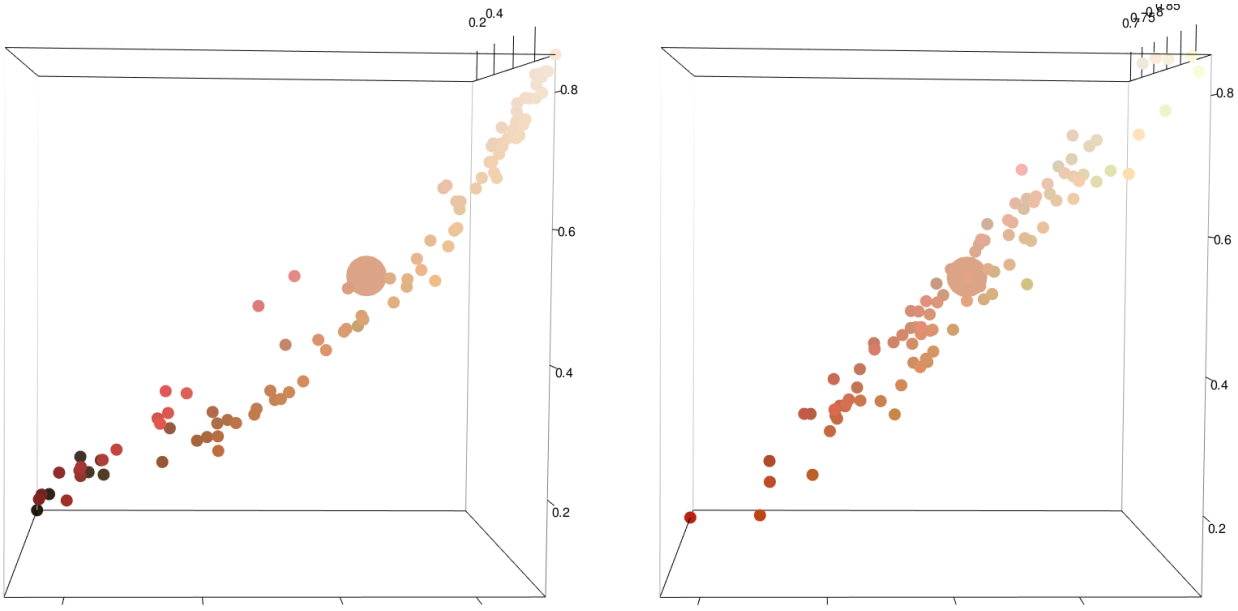

Table 8: Resultados de los 30 remuestreos a los 100 puntos.

6.3 Imagen 3: Ana Sofía

Para la tercera foto que tiene unas dimensiones de 1024 X 768 (786,432 pixeles) se muestra a continuación la región de la piel de la cara extraída junto con la distribución de los pixeles seleccionados para ser piel.

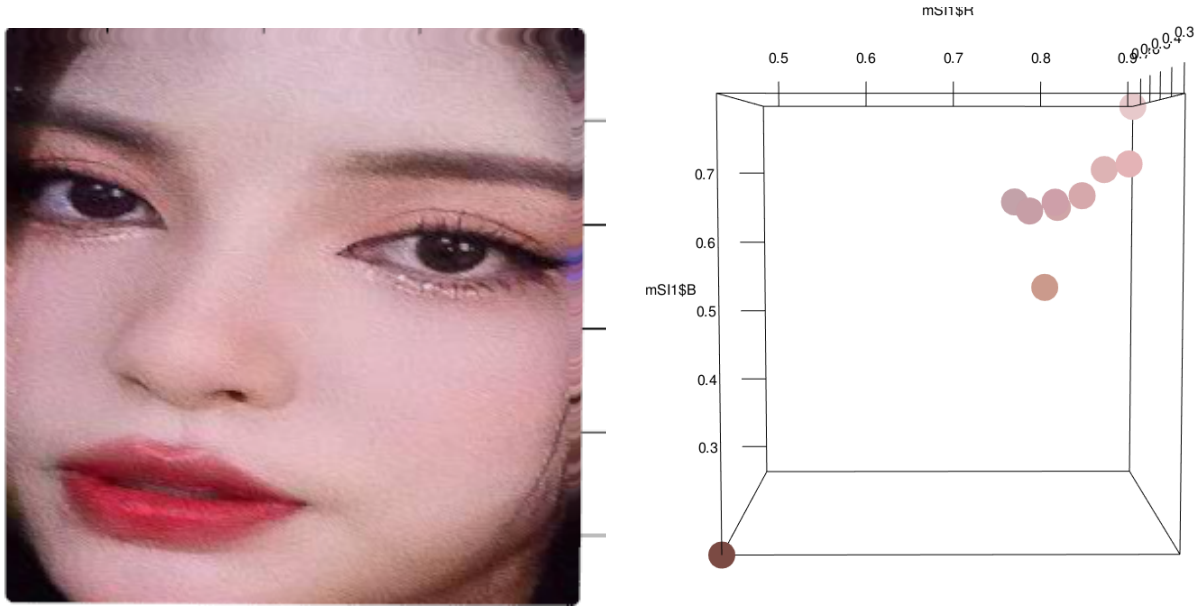

Table 9: Área de la piel.

Complementariamente, se reporta el vector de medias y la matriz de varianzas y covarianzas para la primer imagen:

$$\bar{X} = [0.7949020, 0.6270588, 0.6329412]$$

$$\Sigma = \begin{bmatrix} 0.01428186 & 0.01527908 & 0.01646922 \\ 0.01527908 & 0.01697279 & 0.01845512 \\ 0.01646922 & 0.01845512 & 0.02072775 \end{bmatrix}$$

Una vez que se tiene la media y la matriz de varianzas y covarianzas de la distribución de los pixeles de piel en la imagen, se procede a aplicar el algoritmo de gibb sampling con 30 iteraciones aplicadas a 3 puntos distintos.

Se propone que los puntos sean: $x_1 = [0, 0, 0], x2 = [1, 1, 1], x1 = [0.5, 0.5, 0.5]$ para probar como se comporta el algoritmo cuando se le pone en situaciones donde esta a los extremos de la distribución.

A continuación se muestran las trayectorias recorridas por los 3 puntos:

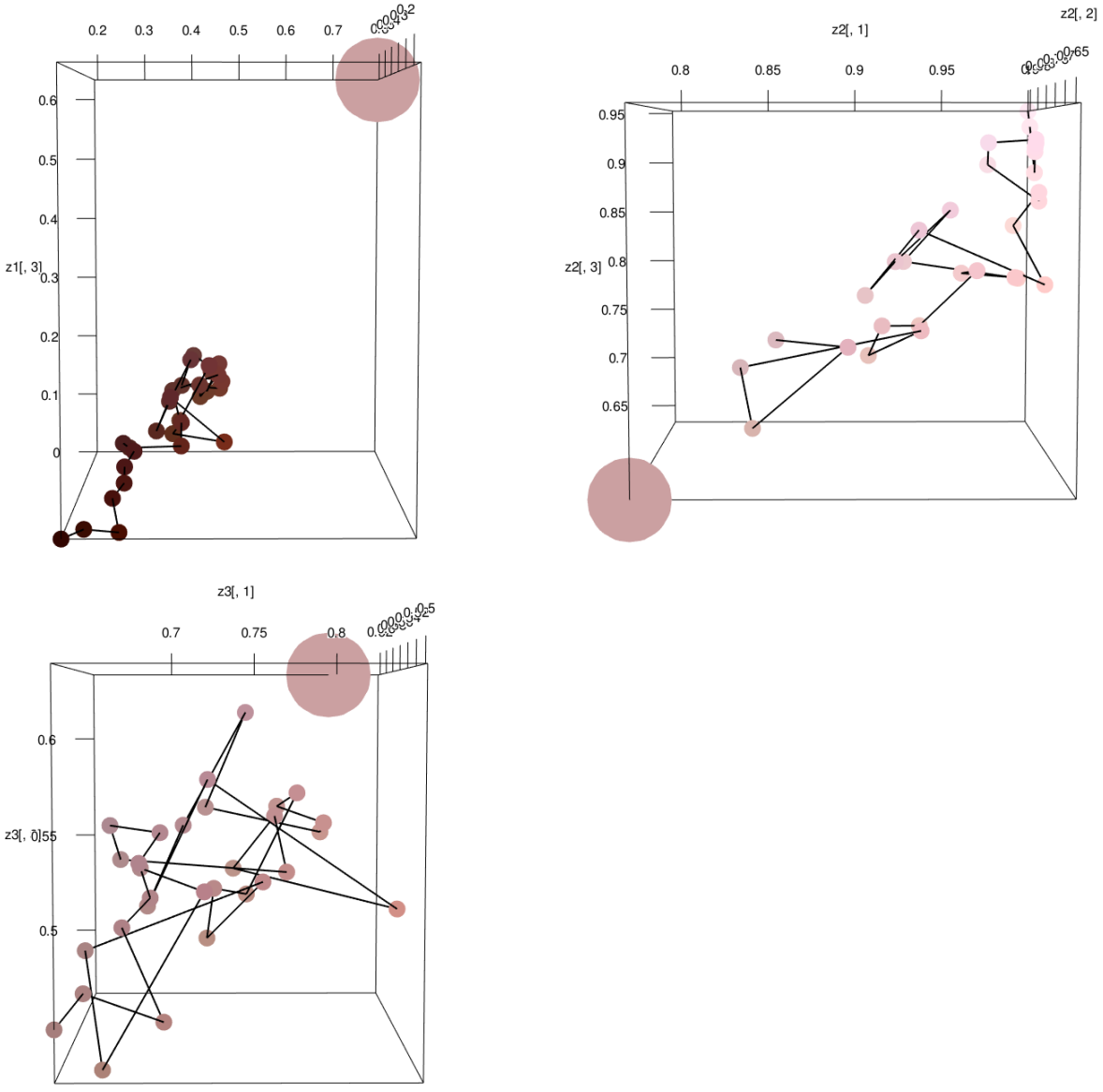

Table 10: Resultados de las trayectorias de los 3 puntos.

Posteriormente, se obtuvieron 100 valores aleatorios del rostro de la imagen para poder aplicarles el algoritmo y verificar a donde llegan al cabo de 30 iteraciones por cada punto.

Los pixeles seleccionados y el resultado de los 100 puntos se muestran en la siguiente tabla:

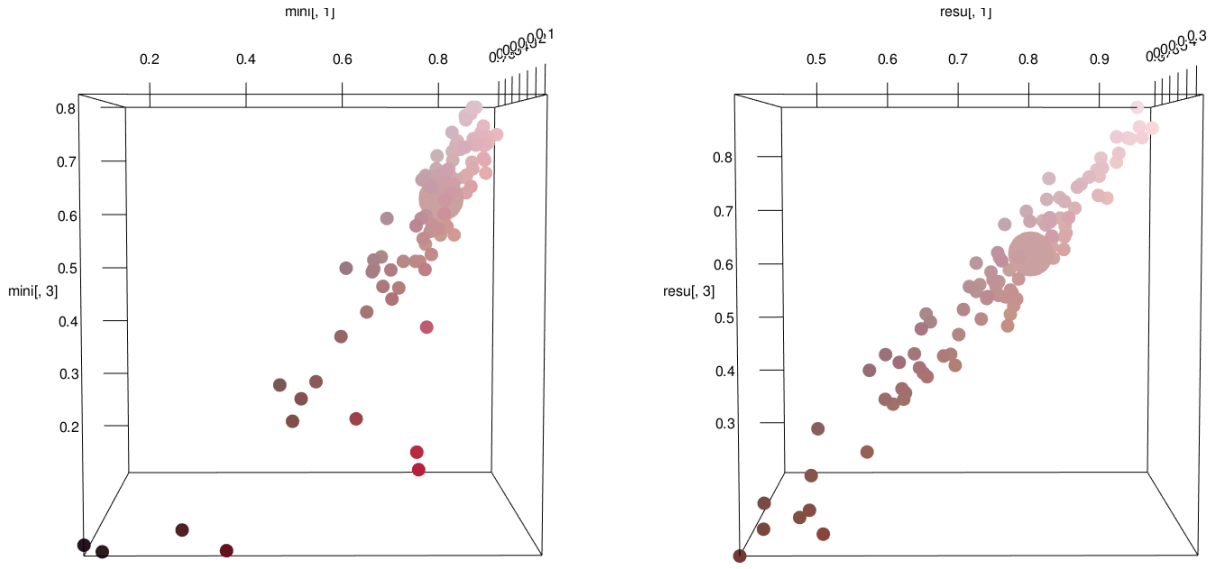

Table 11: Resultados de los 30 remuestreos a los 100 puntos.

7 Bibliografía

Prince, S. J. (2012). Computer vision: models, learning, and inference. Cambridge University Press.

8 Anexos

8.1 Codigo del algoritmo de Gibbs Sampling

```
# Algoritmo de Gibb sampler solo para el caso de la normal multivariada

condicional = function(i,x,media,var1){
  # Calcular las medias condicionales #
  # Supongamos que iniciamos en i = 1
  # Dimensión de la función
  D = length(media)
  # Conjunto de indices
  oindices = setdiff(1:D,i)
  # Partir la matriz de varianzas y covarianzas #
  sigma_ii = var1[i,i] #Varianza marginal de la variable i-esima
  sigma_ij = var1[i,oindices] # Covarianza de xi con las demás variables
  sigma_jj = var1[oindices,oindices] # Covarianza de las demás variables
  # Partir el vector de medias
  mui = media[i] # La media i-esima
  muj = media[oindices] # La media de las otras variables
  # Calcular la media condicional:
  mediac = mui + sigma_ij %*% solve(sigma_jj) %*% t(t((x[oindices]-muj)))
  # Calcular la varianza condicional
  varc = sigma_ii - sigma_ij %*% solve(sigma_jj)%*%t(t(sigma_ij))
  # Resultados
  resultados = list(media = mediac,var = varc)
  return(resultados)
}

# Implementación del Gibbs Sampler
gibbs <- function(inicial, iteraciones, media, var1){
  # Dimensión del vector
  D = length(media)
  # Generar la matriz de mezclas
  muestras <- matrix(0, ncol = (D+2), nrow = iteraciones)
  # Punto de inicio
  x = inicial
  # Para el numéro de iteraciones solicitado
  for(t in 1:iteraciones){
    # Para cada una de las dimensiones del problema
    for(i in 1:D){
      # Parametros de la distribución condicional
      parametros <- condicional(i, x, media, var1)
      # Nuevo valor de la normal dada la información condicional
      x[i] <- (rnorm(1, mean = as.numeric(parametros$media),
                        sd = sqrt(as.numeric(parametros$var))))
      # Verificar el tema de los valores que se salen por debajo
      if(x[i]<0){
        # Redondealo a su limite inmediato
        x[i] = 0
      }
      # Verificar los valores que se salen por arriba
      if(x[i]>1){
        # Redondealo a su limite inmediato
```

```
        x[i] = 1
        }
    }
    # Resultado de la distribución condicional para el t-esimo momento
    muestras[t, ] =  c(x,parametros$media,parametros$var)
  }
  # Devolver la matriz de resultante
  return(muestras)
}
```

8.2 Codigo del desarrollo del ejercicio en R.

```
rm(list=ls())
# Cargar las librerias
# Cargar la imagen en R
## Cargar librerias necesarias ##
library(png) # Libreria para leer PNG's
library(flextable)# LIbreria para hacer tablas bonitas
library(scatterplot3d) #gráficar en 3D
library(rsvg) # Manipular archivos SVG
library(jpeg) # Comprimir el SVG a JPEG
library(rgl) # OpenGL pero para los gráficos en 3D
library(corrplot) # Para gráficar las correlaciones
library(NbClust) # Correr el Kmedias
#library(factoextra) # Obtener exploraciones para el kmedias
library(JuliaCall) # Para pasarme objetos a Julia
### Función que extrae los pixeles
### Los separa en 3 canales de colores
### Calcula las posiciones de los pixeles para reconstruir la imagen
### Calcula el color de cada pixel a Hexadecimasl
### Comprime todo en un DataFrame
pixeles = function(imagen){
  # Extraer los datos
  datos = cbind.data.frame(R = as.vector(imagen[ , ,1]), # Extraer inf de R
                           G = as.vector(imagen[ , ,2]), # Extraer inf de G
                           B = as.vector(imagen[ , ,3])) # Extraer inf de B
  # Extraer las coordenadas de cada pixel para re-mapearlos
  #Filas
  nf = dim(imagen)[1]
  #Columnas
  nc = dim(imagen)[2]
  # Hacer el ordenamiento para remapearlos
  X = rep(nf:1,nc)
  Y = rep(1:nc,each = nf)
  datos$X = X
  datos$Y = Y
  # Añadir el vector de colores al conjunto de datos
  datos$color = rgb(datos$R,datos$G,datos$B)
  # Regresar el objeto transformado
  return(datos)
}

#### Imagen 1: Denisse Guerrero ####
### Cargando la imagen desde la ruta local
ruta1 = "/home/angel/Escritorio/MIAPrimerSemestre/Razonamiento bajo incertidumbre/Tarea 6. GIBB SAMPLER/
# Ruta local de la imagen
source("GIBB SAMPLER Normal.R")
imagen1 = readJPEG(ruta1) # Descomponer la imagen en sus canales RGB
datos1 = pixeles(imagen1)
# Selección de la Carita
subSI1 = datos1[(datos1$X>=610 &datos1$X<=815) &
                  (datos1$Y>=300 &datos1$Y<=490), ]
plot3d(x = subSI1$Y,
       y = 1,
```

```
       z = subSI1$X,
       col = rgb(subSI1$R,subSI1$G,subSI1$B),size = 10)# Fondo
# Puntos de la piel
set.seed(64)
N = 10
mSI1 = subSI1[sample(1:nrow(subSI1),
                     size= N),]
# Gráficar los valores de los puntos para entrenar
plot3d(x = mSI1$R,
       y = mSI1$G,
       z = mSI1$B,
       col = rgb(mSI1$R,mSI1$G,mSI1$B),size = 30)# Fondo
rgl.snapshot("RESULTADOS/R1.2.png", fmt = "png")

### Piel 1
piel1 = mSI1[,1:3]
# Obtener el vector de medias
medias1 = apply(piel1,2,mean);medias1
# Obtener la matriz de varianzas
var1 = var(piel1);var1




### Tomar 3 puntos y muestrear 30 puntos de su trayectoria ####
P = 30
# Definir 3 puntos , el primero sera el origen, el segundo el punto c(1,1,1)
x1 = c(0,0,0)
x2 = c(1,1,1)
x3 = c(0.5,0.5,0.5)
### G1 = Gráficar los 30 remuestreos ####
set.seed(19)
z1 = gibbs(x1,P,medias1,var1)
plot3d(x = z1[,1],
       y = z1[,2],
       z = z1[,3],
      col = rgb(z1[,1],z1[,2],z1[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z1[,1],  # Coordenadas x
  y = z1[,2],  # Coordenadas y
  z = z1[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional
```

```r
rgl.snapshot("RESULTADOS/R1.3.png", fmt = "png")


### G2 = Gráficar los 30 remuestreos ####
set.seed(19)
z2 = gibbs(x2,P,medias1,var1)
plot3d(x = z2[,1],
       y = z2[,2],
       z = z2[,3],
       col = rgb(z2[,1],z2[,2],z2[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z2[,1],  # Coordenadas x
  y = z2[,2],  # Coordenadas y
  z = z2[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional

rgl.snapshot("RESULTADOS/R1.4.png", fmt = "png")


### G3 = Gráficar los 30 remuestreos ####
set.seed(19)
z3 = gibbs(x3,P,medias1,var1)
plot3d(x = z3[,1],
       y = z3[,2],
       z = z3[,3],
       col = rgb(z3[,1],z3[,2],z3[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z3[,1],  # Coordenadas x
  y = z3[,2],  # Coordenadas y
  z = z3[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional
```

```r
rgl.snapshot("RESULTADOS/R1.5.png", fmt = "png")

#### Selección de los 100 puntos ####
##### Tomar 100 puntos aleatorios de toda la imagen ####
set.seed(19)
#mini = datos1[sample(1:nrow(datos1),100),1:3]
mini = as.matrix(subSI1[sample(1:nrow(subSI1),100),1:3])
# Aplicarle el gibbsampler y guardar solo el final
resu = matrix(0,nrow = 100, ncol = 5)
i = 1
for(i in 1:100){
  # Aplicar el gibbsampler
  gs = gibbs(mini[i,],30,medias1,var1)
  # Guardar el ultimo valor
  resu[i,] = gs[30,]
}

# Gráficar bonito los puntos
plot3d(x = mini[,1],
       y = mini[,2],
       z = mini[,3],
       col = rgb(mini[,1],mini[,2],mini[,3]),size = 15)
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 50)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R1.6.png", fmt = "png")


### Gráficar los 30 remuestreos ####
plot3d(x = resu[,1],
       y = resu[,2],
       z = resu[,3],
       col = rgb(resu[,1],resu[,2],resu[,3]),size = 15)
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 50)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R1.7.png", fmt = "png")


# Ver rapido la distribución de la media en el tiempo
# Mini benchmark
Z = gibbs(x2,2000,medias1,var1)
# Comportamiento de la media condicional
plot(1:2000,Z[,4],type = "l")
# Comportamiento de la media condicional
plot(1:2000,Z[,5],type = "l")
# Comportamiento de la media calculada con las observaciones de la cadena
```

```r
# Ver como hacer la media
med = c()
vari = c()
for(i in 1:nrow(Z)){
  # Actualizar med
  med[i] = mean(Z[1:i,1])
  vari[i] = var(Z[1:i,1])
}
# Mostrar el comportamiento de la media
plot(1:2000,med,type = "o",pch = 16,
     main = "Convergencia de la media ")
# Mostrar el comportamiento de la varianza
plot(1:2000,vari,type = "o",pch = 16,
     main = "Convergencia de la varianza")


par(mfrow = c(2,2))



### Ver que show con la distribución 2d


#### Imagen 2: Selena QUintanilla ####
### Cargando la imagen desde la ruta local
ruta1 = "/home/angel/Escritorio/MIAPrimerSemestre/Razonamiento bajo incertidumbre/Tarea 6. GIBB SAMPLER/
imagen1 = readJPEG(ruta1) # Descomponer la imagen en sus canales RGB
datos1 = pixeles(imagen1)

# Selección de la Carita
subSI1 = datos1[(datos1$X>=130 &datos1$X<=240) &
                  (datos1$Y>=150 &datos1$Y<=220), ]
plot3d(x = subSI1$Y,
       y = 1,
       z = subSI1$X,
       col = rgb(subSI1$R,subSI1$G,subSI1$B),size = 10)# Fondo
# Puntos de la piel
set.seed(64)
N = 10
mSI1 = subSI1[sample(1:nrow(subSI1),
                     size= N),]
# Gráficar los valores de los puntos para entrenar
plot3d(x = mSI1$R,
       y = mSI1$G,
       z = mSI1$B,
       col = rgb(mSI1$R,mSI1$G,mSI1$B),size = 30)# Fondo
rgl.snapshot("RESULTADOS/R2.2.png", fmt = "png")

### Piel 1
piel1 = mSI1[,1:3]
# Obtener el vector de medias
medias1 = apply(piel1,2,mean);medias1
# Obtener la matriz de varianzas
```

```r
var1 = var(piel1);var1


### Tomar 3 puntos y muestrear 30 puntos de su trayectoria ####
P = 30
# Definir 3 puntos , el primero sera el origen, el segundo el punto c(1,1,1)
x1 = c(0,0,0)
x2 = c(1,1,1)
x3 = c(0.5,0.5,0.5)
### G1 = Gráficar los 30 remuestreos ####
set.seed(19)
z1 = gibbs(x1,P,medias1,var1)
plot3d(x = z1[,1],
       y = z1[,2],
       z = z1[,3],
       col = rgb(z1[,1],z1[,2],z1[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z1[,1],  # Coordenadas x
  y = z1[,2],  # Coordenadas y
  z = z1[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R2.3.png", fmt = "png")


### G2 = Gráficar los 30 remuestreos ####
set.seed(19)
z2 = gibbs(x2,P,medias1,var1)
plot3d(x = z2[,1],
       y = z2[,2],
       z = z2[,3],
       col = rgb(z2[,1],z2[,2],z2[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z2[,1],  # Coordenadas x
  y = z2[,2],  # Coordenadas y
  z = z2[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
```

```r
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional

rgl.snapshot("RESULTADOS/R2.4.png", fmt = "png")


### G3 = Gráficar los 30 remuestreos ####
set.seed(19)
z3 = gibbs(x3,P,medias1,var1)
plot3d(x = z3[,1],
       y = z3[,2],
       z = z3[,3],
       col = rgb(z3[,1],z3[,2],z3[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z3[,1],  # Coordenadas x
  y = z3[,2],  # Coordenadas y
  z = z3[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R2.5.png", fmt = "png")

#### Selección de los 100 puntos ####
##### Tomar 100 puntos aleatorios de toda la imagen ####
set.seed(19)
#mini = datos1[sample(1:nrow(datos1),100),1:3]
mini = as.matrix(subSI1[sample(1:nrow(subSI1),100),1:3])
# Aplicarle el gibbsampler y guardar solo el final
resu = matrix(0,nrow = 100, ncol = 5)
i = 1
for(i in 1:100){
  # Aplicar el gibbsampler
  gs = gibbs(mini[i,],30,medias1,var1)
  # Guardar el ultimo valor
  resu[i,] = gs[30,]
}

# Gráficar bonito los puntos
plot3d(x = mini[,1],
       y = mini[,2],
       z = mini[,3],
```

```
       col = rgb(mini[,1],mini[,2],mini[,3]),size = 15)
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 50)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R2.6.png", fmt = "png")


### Gráficar los 30 remuestreos ####
plot3d(x = resu[,1],
       y = resu[,2],
       z = resu[,3],
       col = rgb(resu[,1],resu[,2],resu[,3]),size = 15)
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 50)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R2.7.png", fmt = "png")

#### Imagen 2: Selena QUintanilla ####
### Cargando la imagen desde la ruta local
ruta1 = "/home/angel/Escritorio/MIAPrimerSemestre/Razonamiento bajo incertidumbre/Tarea 6. GIBB SAMPLER/
imagen1 = readJPEG(ruta1) # Descomponer la imagen en sus canales RGB
datos1 = pixeles(imagen1)

# Selección de la Carita
subSI1 = datos1[(datos1$X>=520 &datos1$X<=790) &
                  (datos1$Y>=270 &datos1$Y<=450), ]
plot3d(x = subSI1$Y,
       y = 1,
       z = subSI1$X,
       col = rgb(subSI1$R,subSI1$G,subSI1$B),size = 10)# Fondo
# Puntos de la piel
set.seed(64)
N = 10
mSI1 = subSI1[sample(1:nrow(subSI1),
                     size= N),]
# Gráficar los valores de los puntos para entrenar
plot3d(x = mSI1$R,
       y = mSI1$G,
       z = mSI1$B,
       col = rgb(mSI1$R,mSI1$G,mSI1$B),size = 30)# Fondo
rgl.snapshot("RESULTADOS/R3.2.png", fmt = "png")

### Piel 1
piel1 = mSI1[,1:3]
# Obtener el vector de medias
medias1 = apply(piel1,2,mean);medias1
# Obtener la matriz de varianzas
```

```r
var1 = var(piel1);var1

### Tomar 3 puntos y muestrear 30 puntos de su trayectoria ####
P = 30
# Definir 3 puntos , el primero sera el origen, el segundo el punto c(1,1,1)
x1 = c(0,0,0)
x2 = c(1,1,1)
x3 = c(0.5,0.5,0.5)
### G1 = Gráficar los 30 remuestreos ####
set.seed(19)
z1 = gibbs(x1,P,medias1,var1)
plot3d(x = z1[,1],
       y = z1[,2],
       z = z1[,3],
       col = rgb(z1[,1],z1[,2],z1[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z1[,1],  # Coordenadas x
  y = z1[,2],  # Coordenadas y
  z = z1[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R3.3.png", fmt = "png")


### G2 = Gráficar los 30 remuestreos ####
set.seed(19)
z2 = gibbs(x2,P,medias1,var1)
plot3d(x = z2[,1],
       y = z2[,2],
       z = z2[,3],
       col = rgb(z2[,1],z2[,2],z2[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z2[,1],  # Coordenadas x
  y = z2[,2],  # Coordenadas y
  z = z2[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
```

```
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional

rgl.snapshot("RESULTADOS/R3.4.png", fmt = "png")


### G3 = Gráficar los 30 remuestreos ####
set.seed(19)
z3 = gibbs(x3,P,medias1,var1)
plot3d(x = z3[,1],
       y = z3[,2],
       z = z3[,3],
       col = rgb(z3[,1],z3[,2],z3[,3]),
       size = 20)
# Añadir líneas que conecten los puntos en el orden en que aparecen
lines3d(
  x = z3[,1],  # Coordenadas x
  y = z3[,2],  # Coordenadas y
  z = z3[,3],  # Coordenadas z
  col = "black",  # Color de las líneas
  lwd = 2       # Grosor de las líneas
)
# Añadir la media
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 100)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R3.5.png", fmt = "png")

#### Selección de los 100 puntos ####
##### Tomar 100 puntos aleatorios de toda la imagen ####
set.seed(19)
#mini = datos1[sample(1:nrow(datos1),100),1:3]
mini = as.matrix(subSI1[sample(1:nrow(subSI1),100),1:3])
# Aplicarle el gibbsampler y guardar solo el final
resu = matrix(0,nrow = 100, ncol = 5)
i = 1
for(i in 1:100){
  # Aplicar el gibbsampler
  gs = gibbs(mini[i,],30,medias1,var1)
  # Guardar el ultimo valor
  resu[i,] = gs[30,]
}

# Gráficar bonito los puntos
plot3d(x = mini[,1],
       y = mini[,2],
       z = mini[,3],
       col = rgb(mini[,1],mini[,2],mini[,3]),size = 15)
```

```r
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 50)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R3.6.png", fmt = "png")


### Gráficar los 30 remuestreos ####
plot3d(x = resu[,1],
       y = resu[,2],
       z = resu[,3],
       col = rgb(resu[,1],resu[,2],resu[,3]),size = 15)
# Usar points3d para agregarlo al gráfico
points3d(x = medias1[1],
         y = medias1[2],
         z = medias1[3],
         col = rgb(medias1[1],medias1[2],medias1[3]), # Color del punto adicional
         size = 50)  # Tamaño del punto adicional
rgl.snapshot("RESULTADOS/R3.7.png", fmt = "png")
```