Razonamiento bajo incertidumbre: tarea 1.

Angel García Báez

2024-08-21

Índice

Instrucciones

Dada una imagen, descomponerla en sus valores RGB:

- Calcular el vector de medias.
- Gráficar en 3D con sus respectivos colores de RGB para cada pixel.
- Calcula la matriz de varianzas y covarianzas.
- Calcular la matriz de correlaciones.
- Dar una interpretación a los resultados obtenidos.

Cargar una imagen simple y descomponerla.

A manera de realizar una primera incursión en la descomposición de una imagen en sus valores de RGB, se va a tratar de descomponer la siguiente imagen de colores muy marcados y simples para buscar la manera de decirle a R que obtenga los valores de la siguiente imagen:

Dicha imagen esta compuesta por 5 colores: Azul, amarillo, rojo, naranja y negro, por lo que se espera que al descomponerla en sus canales RGB se obtengan 5 casos únicos.

A continuación se realiza el proceso de cargar la imagen y descomponerla:

```
# Cargar las librerias necesarias
library(png)
library(reshape2)

# Leer la imagen
ruta = "im1.png"
imagen = readPNG(ruta)
str(imagen) # EL archivo cargado es un CUBO de información
```

```
##  num [1:328, 1:798, 1:4] 0 0 0 0 0 0 0 0 0 0 ...
```

```
# Del cubo de información se extrae cada capa de color
rojo = imagen[ , ,1] # Extraer inf de  R
verde = imagen[ , ,2] # Extraer inf de G
azul = imagen[ , ,3] # Extraer inf de B

# Convertir a vectores los canales de color
vrojo = as.vector(rojo) # Vector de rojo
vverde = as.vector(verde) # Vector de verde
vazul = as.vector(azul) # Vector de azul

# Compactar los vectores en un dataframe
datos = cbind.data.frame(R = vrojo,
                         G = vverde,
                         B = vazul)
```

A continuación se muestran los primeros 6 casos de la matriz formada por los valores de RGB de cada pixel de la imagen:

```r
# EL dataframe esta compuesto por 261744 filas (pixeles) y 3 columnas (R,G y B)
library(flextable)# LIbreria para hacer tablas bonitas
autofit(theme_box(flextable(head(datos))))
```

R	G	B
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0
0	0	0

Ahora, se calcula el vector de medias de la matriz conformada por los colores de los pixeles de la imagen:

```r
# Calcular la media de cada variable y hacerla vector
media = data.frame(t(apply(datos,2,mean)))
autofit(theme_box(flextable(media)))
```

R	G	B
0.802177	0.6180255	0.3007519

$$\vec{\mu} = [0.802177, 0.6180255, 0.3007519]$$

Ahora, se añade ese vector a los datos y se procede a realizar la gráfica de dispersión en 3 dimensiones:

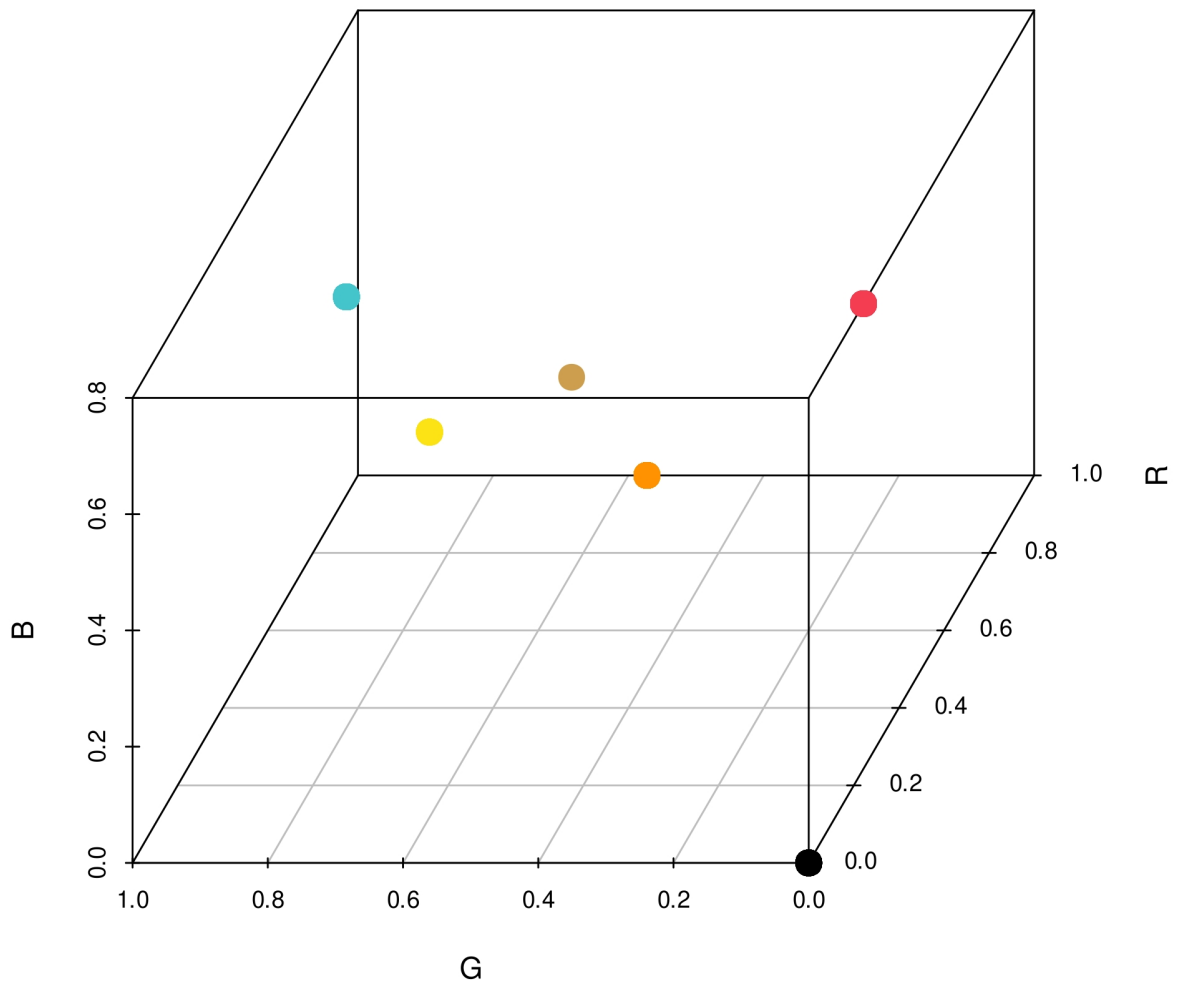
```r
# Añadir el vector de medias a los datos
datos1 = rbind.data.frame(datos,media)
# Libreria para gráficar en 3D los puntos
library(scatterplot3d) #gráficar en 3D
library(rsvg) # Manipular archivos SVG
library(jpeg) # Comprimir el SVG a JPEG

# Realizar el gráfico en 3D

## Crear el gráfico en SVG para no perder calidad
svg("r1.svg")
# Ejecutar el gráfico
scatterplot3d(datos1[,1:3], # Datos de R,G y B como X,Y y Z
              color = rgb(datos1$R,datos1$G,datos1$B),# Colores acorde a su RGB
              pch = 16, # Forma rellenita de los puntos
              angle = 600, #Rotación del gráfico
              cex.symbols = 2)
# Exportarlo
dev.off()
```

```
## pdf
##   2
```

```r
### Comprimir el gráfico a JPEG
sv = rsvg("r1.svg",
          height = 2000,
          width = 2000)
### Exportar en jpeg
writeJPEG(sv, "r1.jpg", quality = 10)
```



Obtención de la 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:

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

```
### Calcu.ar la matriz de varianzas y covarianzas a mano
X = as.matrix(datos) # Matriz de datos X
m = t(apply(datos,2,mean)) # Vector de medias
n = nrow(X)
XTX = t(X)%*%X
S = (XTX- n*(t(m)%*%m))/(n-1)
# Pasar a una tabla bonita:
autofit(theme_box(flextable(as.data.frame(S))))
```

R	G	B
0.09732109	-0.024705326	-0.090221562
-0.02470533	0.061503514	0.005918216
-0.09022156	0.005918216	0.095997109

Obtención de la matriz de correlaciones

Para obtener la matriz de correlaciones a partir de la matriz de vairanzas y covarianzas, se puede usar la siguiente expresión:

$$R = D^{-1/2}\Sigma D^{-1/2}$$

Donde:

$$D = diag(\Sigma)$$

```
# Hacer la matriz diagonal
D = matrix(0,3,3)
diag(D) = diag(S)
# Usarla en la expresión
R = sqrt(solve(D)) %*% S %*% sqrt(solve(D))
colnames(R) = c("R","G","B")
# Pasar a una tabla bonita:
autofit(theme_box(flextable(as.data.frame(R))))
```

R	G	B
1.0000000	-0.31932811	-0.93342147
-0.3193281	1.00000000	0.07702146
-0.9334215	0.07702146	1.00000000

Imagen 1

Despues de hacer la prueba de que se puede realizar la descomposición en lenguaje R, se procede a trabajar con una imagen más complicada, dicha imagen es la siguiente:

Lo primero que hay que hacer es decirle a R que lea la imagen y obtenga la información de cada uno de sus pixeles:

```
# Leer la imagen en jpg
ruta = "im2.jpg"
imagen = readJPEG(ruta)
str(imagen) # EL archivo cargado es un CUBO de información
```

```
##  num [1:1125, 1:2000, 1:3] 0.0863 0.0667 0.0667 0.0549 0.0667 ...
```

Después de cargar la imagen y que R identificara las capas de colores en los pixeles, dichas capas o canales se extraen y se convierten en vectores:

```
# Del cubo de información se extrae cada capa de color
rojo = imagen[ , ,1] # Extraer inf de  R
verde = imagen[ , ,2] # Extraer inf de G
azul = imagen[ , ,3] # Extraer inf de B

# Convertir a vectores los canales de color

# Compactar los vectores en un dataframe
datos = cbind.data.frame(R = as.vector(rojo),
                         G = as.vector(verde),
                         B = as.vector(azul))
```

Posteriormente se muestra como esta conformado el arreglo de los datos:

```
# EL dataframe esta compuesto por 261744 filas (pixeles) y 3 columnas (R,G y B)
autofit(theme_box(flextable(head(datos)))) #2,250,000 pixeles/casos
```

R	G	B
0.08627451	0.07058824	0.1372549
0.06666667	0.05490196	0.1215686
0.06666667	0.06274510	0.1176471
0.05490196	0.05490196	0.1019608
0.06666667	0.05882353	0.1098039
0.07843137	0.07058824	0.1254902

Ahora, se calcula el vector de medias de la matriz conformada por los colores de los pixeles de la imagen:

```
# Calcular la media de cada variable y hacerla vector
media = data.frame(t(apply(datos,2,mean)))
autofit(theme_box(flextable(media)))
```

R	G	B
0.5489734	0.3176347	0.4056583

$$\vec{\mu} = [0.5489734, 0.3176347, 0.4056583]$$

Ahora, se añade ese vector a los datos y se procede a realizar la gráfica de dispersión en 3 dimensiones:

```
# Añadir el vector de medias a los datos
datos1 = rbind.data.frame(datos,media)

# Realizar el gráfico en 3D

## Crear el gráfico en SVG para no perder calidad
svg("r2.svg")
# Ejecutar el gráfico
scatterplot3d(datos1[,1:3], # Datos de R,G y B como X,Y y Z
              color = rgb(datos1$R,datos1$G,datos1$B),# Colores acorde a su RGB
              pch = 16, # Forma rellenita de los puntos
              angle = 600, #Rotación del gráfico
              cex.symbols = 2)
# Exportarlo
dev.off()
### Comprimir el gráfico a JPEG
sv = rsvg("r2.svg",
          height = 2000,
          width = 2000)
### Exportar en jpeg
writeJPEG(sv, "r2.jpg", quality = 10)
```