Representación gráfica de datos tridimensionales

Vídeo 2.5. Representación gráfica de datos tridimensionales

Para representar gráficamente datos tridimensionales en R es necesario instalar unos paquetes extras. En esta sección vamos a aprender a representarlos usando **dos paquetes**: `scatterplot3d` y `rgl`. Y como ejemplo, **vamos a representar en 3Ddiagramas de dispersión compuestos por 3 variables**. La ventaja del segundo respecto al primero es que es interactivo, es decir, puedes desplazar con el ratón el cubo generado para buscar la mejor visualización de los datos, mientras que en el primero tienes que indicarle los grados desde donde ver tus datos. Independientemente, ambos son una buena herramienta para ver en 3D la distribución de nuestros datos.

Datos de ejemplo

En un estudio sobre la caracterización del sílex se han tomado algunas variables (IRA, IRC y Peso) que ayudan a identificar su procedencia y contextualizar movimientos de aprovisionamiento de esta materia prima durante el Pleistoceno y el Holoceno. Los datos os los podéis descargar del Aula Virtual (*litica_silex.xlsx*). Importamos el archivo excel siguiendo la siguiente ruta en R Commander:

`Data ▶ Import data ▶ from Excel file...`

Diagramas de dispersión en 3D con `scatterplot3d`

`scatterplot3d` es un paquete relativamente sencillo de utilizar. Pero antes de ponernos manos a la obra con él tenemos que instalarlo y cargarlo en R Commander. Para ello ejecutamos los dos códigos siguientes:

```
install.packages("scatterplot3d")
library(scatterplot3d)
```

Con los datos cargados, vamos a ejecutar la siguiente función (Figura 1):

```
scatterplot3d(litica$IRA,litica$IRC,litica$Peso)
```

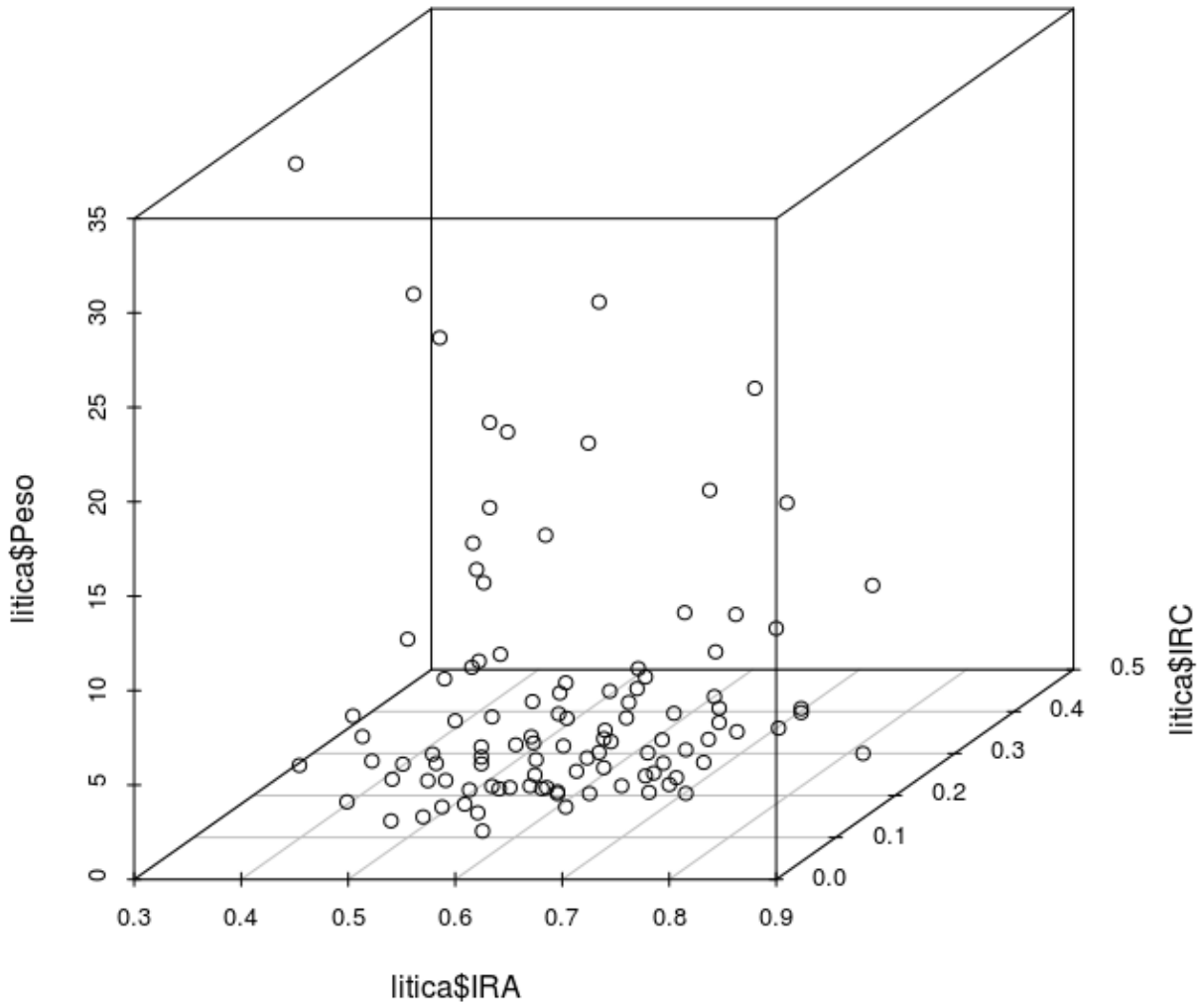

Figura 1: Representación básica de un diagrama de dispersión en 3D usando scatterplot3d.

En esta función hay que definir primero cuáles son nuestras variables X, Y y Z, respectivamente. Por lo tanto, hemos añadido cuáles son las variables X (`litica$IRA`), Y (`litica$IRC`) y Z (`litica$Peso`). Recordamos que el símbolo de dólar ($) lo que hace es unir el nombre del conjunto de datos (litica) con el nombre de la columna de una variable concreta (IRA, IRC o Peso).

Sin embargo, podemos personalizar la figura anterior. Atención al siguiente código a ejecutar (Figura 2):

```
# Como tenemos 5 grupos en la variable Grupo_Laplace, definimos qué símbolo
va a representar cada uno de ellos.
shapes=c(14,15,16,17,18)
shapes=shapes[as.numeric(litica$Grupo_Laplace)]

# Igual que en el caso anterior, definimos los colores de cada grupo. Al
haber 6 grupos, damos 5 colores
colors=c("red", "blue", "green", "orange", "purple")
colors=colors[as.numeric(litica$Grupo_Laplace)]

# Aquí es la función como tal con más atributos
scatterplot3d(litica$IRA, litica$IRC, litica$Peso, pch=shapes, type="h",
color=colors, grid=TRUE, box=TRUE, angle=130, main="Diagrama de dispersión
en 3D", xlab="Variable X = IRA", ylab="Variable Y = IRC", zlab="Variable Z =
Peso")
```

Vamos a intentar comprender el código anterior. Como véis, lo he dividido en 3 partes. Aun así, os comento un elemento clave en programación para que os vayáis familiarizando con él. **Todo el código que está escrito después del símbolo almohadilla (#) es código que no se va a ejecutar**. Por eso veréis en muchos sitios (y en este curso también) que se utiliza para hacer comentarios sobre el código, principalmente para facilitar su comprensión y saber qué va a hacer. Volvamos a las 3 partes del código:

1. Parte 1: está escrita para personalizar los símbolos de cada grupo. Como tenemos 6 grupos en nuestros datos, obviamente tenemos que especificar 6 tipos de símbolos, uno por grupo. En este caso, se hace con números (14-19). En el Anexo 2.1 podréis ver la equivalencia de los símbolos son sus respectivos números (`pch`).
2. Parte 2. De modo análogo a la parte 1, aquí se especifican los colores de cada grupo. Igualmente, en el Anexo 2.1 podréis encontrar información sobre los colores para que lo personalicéis con vuestras preferencias.
3. Parte 3. Una vez especificados los puntos anteriores, vamos a ejecutar la función propiamente dicha de `scatterplot3d`. Pero vamos a modificar la función añadiendo más atributos:
    1. Como ya se ha explicado, hay que definir inicialmente las variables X, Y, Z (`litica$IRA, litica$IRC, litica$Peso`).
    2. `pch` y `color` hacen referencia a los objetos creados en la Parte 1 y Parte 2, respectivamente.
    3. `type` se refiere a si se quieren representar solo los puntos (`p`), líneas que unen los puntos (`l`) o la línea que une cada punto con el plano inferior del 3D (`h`).

4. `grid` y `box` pueden adquirir los valores TRUE / FALSE, y representan la malla cuadrada del plano inferior o el cubo 3D donde se ubican los puntos, respectivamente.
5. `main`, `xlab`, `ylab`, `zlab`, se refieren a los nombres del título de la figura, y de los ejes X, Y, Z respectivamente.
6. `angle` permite cambiar el ángulo de la figura 3D.

Sin embargo, y esto ya es una opinión personal, me resulta complicado poder interpretar los datos utilizando esta figura, ya que no te permite tener libertad de rotar el 3D para poder facilitar la comprensión de los datos. Pero el que sí te lo permite es el paquete que vamos a aprender a continuación (`rgl`).

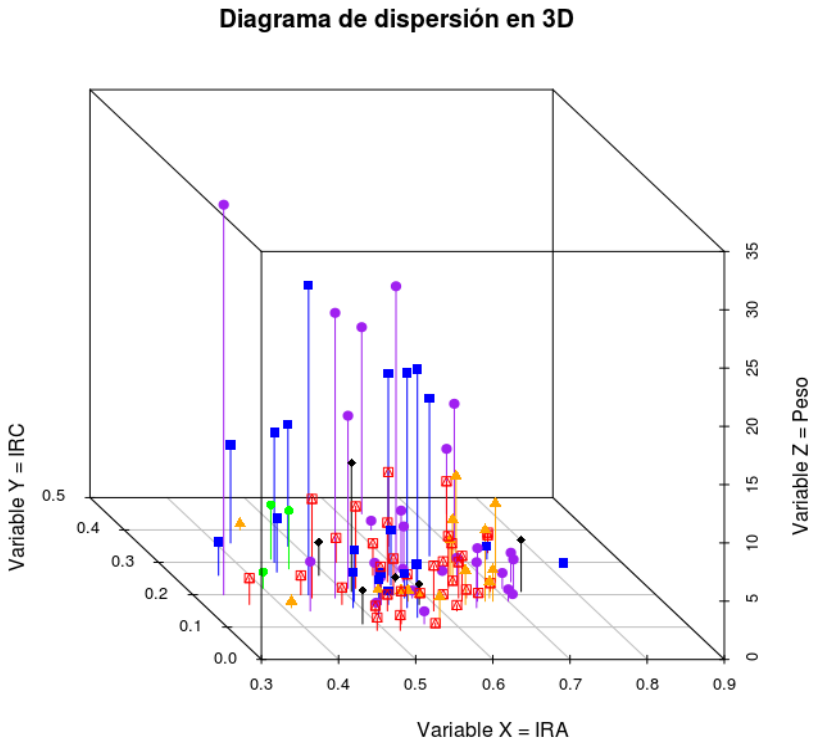

Figura 2: Diagrama de dispersión en 3D con scatterplot3d personalizado.

Diagramas de dispersión en 3D con el paquete `rgl`

El paquete `rgl` es extraordinariamente más completo (y complejo) que `scatterplot3d`. Además te permite generar figuras interactivas que puedes mover interaccionando con el ratón. De entre las cuasi-infinitas opciones que tiene a la hora de representar gráficamente he seleccionado un tipo de visualización que permite

representar las observaciones en 3D, diferenciando por grupos y marcando las elipses de cada uno de ellos, tal y como se muestra en la Figura 3.

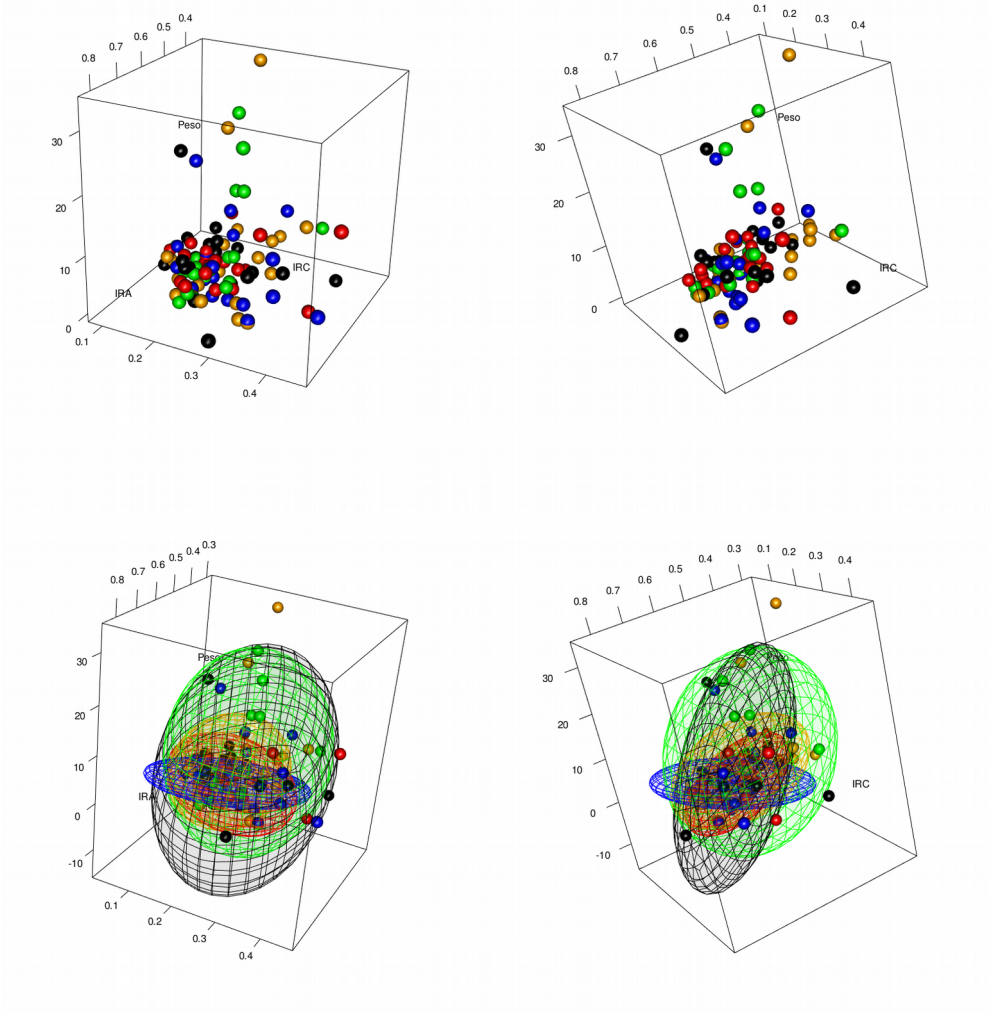

Figura 3: Diagramas de dispersión en 3D usando el paquete rgl. En la fila superior se ven dos vistas del cubo con las observaciones. En la fila inferior se muestran además las elipses al 90%.

Lamentablemente, `rgl` no dispone de una interfaz gráfica dentro de R Commander que facilite la interacción con el usuario, por lo que no queda más remedio que recurrir a código. Sin embargo, y para vuestra tranquilidad, para conseguir las imágenes de la figura anterior he preparado un código para que sea más fácil su interacción. Pero como siempre, tenemos que tener instalado y cargado el paquete `rgl`:

```
install.packages("rgl")
library(rgl)
```

Con los datos cargados (`litica`), copiamos y pegamos el siguiente código en R Commander:

```
# Indica qué variables representan la X, Y, Z, así como la categórica
x=litica$IRA
y=litica$IRC
z=litica$Peso
Grupo=litica$Grupo_Laplace

# Selecciona los colores para cada grupo siguiendo el formato c("red",
"orange",...)
# Tiene que haber tantos colores como grupos tengamos
Colores=c("red", "green", "blue", "orange", "black")

# Este funciona para representar el cubo con colores por grupo
plot3d(x, y, z, col.var=as.integer(Grupo),col=Colores, size=2, type="s",
xlab="IRA", ylab="IRC", zlab="Peso")

# Color de fondo
rgl.bg(color="white")

# Definir cuáles son los grupos
groups = Grupo
levs = levels(groups)
group.col = Colores

# Esta es la función para hacer las elipses de cada grupo
for (i in 1:length(levs)) {
    group = levs[i]
    selected = groups == group
    xx = x[selected]; yy = y[selected]; zz = z[selected]
    ellips = ellipse3d(cov(cbind(xx,yy,zz)), centre=c(mean(xx), mean(yy),
    mean(zz)), level = 0.90)
    shade3d(ellips, col = group.col[i], alpha = 0.1, lit = FALSE)
    wire3d(ellips, col = group.col[i],  lit = FALSE)
    # show group labels
    texts3d(mean(xx),mean(yy), mean(zz), text = group,
            col= group.col[i], cex = 1)
  }
```

Como podéis observar, el código lo he dividido en 6 partes. En las primeras dos partes se ha destacado lo que es esencial escribir para adaptar el código a nuestras necesidades (está destacado en **rojo**). En concreto son las referencias a las variables X, Y y Z, más la columna que contiene los grupos, así como los colores (tantos como grupos tengamos).

En el resto de secciones he destacado en negrita lo que considero puede ser importante para adaptar la apariencia de nuestra figura final:

- `size=2`. Representa el tamaño de cada una de las esferas de las observaciones.
- `type="s"`. Representa el modo de visualizar las observaciones.
  - s = esfera
  - p = puntos
  - n = no se representan
  - h = representa la algura del punto respecto al plano inferior
- `xlab, ylab, zlab`. En el Anexo 2.1 se explica.
- `color="white"`. Se refiere al color de fondo de la figura.

- `level=0.90`. Hace alusión a la probabilidad de las elipses.
- `cex=1`. Indica el tamaño del texto que identifica a cada elipse con su grupo.