Análisis de dominios semánticos o culturales con R

Etnografía y Análisis Cualitativo de Datos. Grado en Antropología Social y Cultural

Arturo Alvarez-Roldan
aalvarez@ugr.es
Departamento de Antropología Social, Universidad de Granada

14 / June / 2025

DOI: 10.5281/zenodo.15667110

Índice

Resumen: Este artículo presenta una metodología para analizar dominios culturales utilizando R y datos obtenidos mediante técnicas como el listado libre y la clasificación de tarjetas. Se explican paso a paso los procesos para construir matrices de proximidad, aplicar análisis como MDS, conglomerados jerárquicos y grafos, así como medir similitudes entre sujetos. También se utiliza la biblioteca FreeSortR para representar estructuras semánticas compartidas en un grupo. El enfoque combina herramientas cualitativas y cuantitativas para revelar cómo una comunidad organiza y percibe distintos elementos culturales.

Abstract: This paper outlines a methodology for analyzing cultural domains using R and data gathered through techniques like free listing and card sorting. It details steps to build proximity matrices, apply analy-

ses such as MDS, hierarchical clustering, and graph visualization, and assess inter-subject similarity. The FreeSortR package is also used to represent shared semantic structures. The approach combines qualitative and quantitative tools to uncover how a community organizes and perceives various cultural elements.

1 Introducción

Un **dominio semántico o cultural** es un conjunto organizado de elementos (conceptos, palabras, objetos, experiencias sensoriales, etc.) que pertenecen a una misma esfera temática y que se perciben como relacionados dentro de un sistema lingüístico o cultural determinado. Este conjunto refleja cómo una comunidad clasifica, conceptualiza y entiende un ámbito específico de la experiencia o del conocimiento (Borgatti 1994).

Los elementos de un dominio suelen situarse en un mismo nivel de contraste y comparten una función o categoría general. Por ejemplo, en el dominio de las "formas geométricas", términos como redondo, cuadrado o triangular constituyen distintas realizaciones del concepto abstracto de "forma", y su significado se construye en parte por las relaciones mutuas que establecen entre sí. De este modo, los dominios permiten acceder a la **estructura semántica** que subyace a las **categorizaciones culturales**, así como a los **modelos mentales compartidos** (o no) por una comunidad (Weller y Romney 1988).

En resumen, un dominio cultural:

- Es un conjunto coherente y temáticamente delimitado de elementos.
- Se define cultural y lingüísticamente.
- Su estudio permite explorar cómo las personas organizan e interpretan un ámbito específico de la experiencia.

1.1 Recogida de datos

Para identificar los elementos que componen un dominio cultural, se utiliza frecuentemente la técnica del **listado libre** (*free listing*), que consiste en pedir a los informantes que mencionen todos los elementos que conocen relacionados con un tema determinado. Esta técnica permite recopilar una lista de términos o estímulos que se consideran culturalmente relevantes. Generalmente, una muestra de 30 informantes resulta suficiente para identificar los principales elementos de un dominio. Un criterio habitual es considerar que un elemento forma parte del dominio cuando es mencionado por al menos el 10% de los participantes (Bernard, Wutich, y Ryan 2016).

Es importante señalar que los elementos de un dominio no tienen por qué ser únicamente términos lingüísticos. En muchos estudios, los dominios están formados por estímulos no verbales como imágenes, sonidos, olores o sabores, lo que permite investigar también dominios sensoriales o perceptivos.

Una vez definidos los elementos del dominio, es posible estudiar su estructura semántica mediante la técnica de **clasificación de tarjetas** (*pile sorting*). Esta técnica consiste en entregar a cada participante una serie de tarjetas —físicas o digitales— que representan los distintos elementos del dominio, y pedirle que los agrupe libremente según las semejanzas que perciba entre ellos. El número de grupos es libre, pero se establece que cada elemento debe pertenecer a un único grupo, y que deben realizarse al menos dos agrupaciones. Esta tarea permite obtener datos sistemáticos sobre la percepción de similitudes y diferencias entre los elementos del dominio por parte de los informantes (Bernard, Wutich, y Ryan 2016).

1.2 Objetivo y enfoque metodológico

El objetivo de este artículo es presentar una estrategia para el análisis de dominios culturales que combina herramientas cualitativas y cuantitativas, dentro de un enfoque metodológico mixto. En particular, se describe cómo analizar datos obtenidos mediante *pile sorting* utilizando **R**, un entorno de programación y análisis estadístico de código abierto, ampliamente empleado en ciencias sociales y del comportamiento.

El procedimiento analítico propuesto se compone de **tres fases principales**, que se desarrollan en detalle en las secciones siguientes del artículo:

1. **Preparación del proyecto y de los datos**: se construye un *data frame* a partir de las clasificaciones individuales de los elementos del dominio cultural, organizadas por sujeto.
2. **Análisis de la estructura del dominio**: se elabora una **matriz de proximidad** que refleja la frecuencia con la que los elementos fueron agrupados conjuntamente. A partir de esta matriz, se aplican diversas técnicas de análisis —como escalado multidimensional, análisis jerárquico de conglomerados y visualización en redes— para descubrir la **organización semántica** del dominio (DeJordy et al. 2007).
3. **Análisis de la similitud entre sujetos**: se calcula una **matriz de similitud** basada en la correlación entre las matrices individuales de clasificación, lo que permite estudiar la coherencia interna del dominio y explorar la **variabilidad intracultural** entre los participantes.

Este enfoque permite no solo visualizar y modelar la estructura conceptual de un dominio cultural, sino también analizar en qué medida esa estructura es compartida por los miembros de una comunidad.

2 Preparación del proyecto y de los datos

Para realizar el análisis vamos a utilizar como ejemplo el conjunto de datos AromaSort del paquete FreeSortR. Este conjunto de datos consiste en las clasificaciones en grupos de 16 aromas realizadas por 30 sujetos.

Abrimos un nuevo proyecto en R al que llamaremos `"Analisis de dominios culturales"`. Dentro del proyecto creamos cuatro carpetas para organizar sus elementos:

- `codigo`
- `datos`
- `tablas`
- `graficos`

2.1 Instalar y cargar las bibliotecas

Instalamos y cargamos las bibliotecas que vamos a usar.

```
library(reshape2)
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(MASS)
```

```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```

```
library(writexl)
library(FreeSortR)
```

```
## Loading required package: smacof

## Loading required package: plotrix

## Loading required package: colorspace

## Loading required package: e1071

##
## Attaching package: 'smacof'

## The following object is masked from 'package:base':
##
##     transform

## Loading required package: vegan

## Loading required package: permute

##
## Attaching package: 'permute'

## The following object is masked from 'package:igraph':
##
##     permute

## Loading required package: lattice

## This is vegan 2.6-4

##
## Attaching package: 'vegan'

## The following object is masked from 'package:igraph':
##
##     diversity

## Loading required package: ellipse

##
## Attaching package: 'ellipse'

## The following object is masked from 'package:graphics':
##
##     pairs
```

2.2 Crear el *data frame*

Construimos el *data frame* de `AromaSort` a partir de los vectores que continen los nombres de los aromas y los que contienen el resultado de las clasificaciones llevadas a cabo por los sujetos.

```
aromas <- c("limón", "pomelo", "piña", "pera", "miel", "mantequilla",
            "pan tostado", "avellana tostada", "fresa", "frambuesa",
            "cereza", "grosella", "pimiento verde", "ahumado", "pimienta",
            "regaliz")

S1 <- c(5, 5, 4, 5, 4, 1, 1, 5, 5, 3, 3, 3, 4, 2, 2, 2)
S2 <- c(1, 3, 1, 1, 2, 4, 2, 2, 1, 3, 3, 4, 3, 2, 2, 4)
S3 <- c(4, 5, 5, 5, 3, 5, 3, 3, 5, 4, 5, 4, 1, 2, 2, 3)
S4 <- c(1, 1, 1, 1, 2, 2, 5, 2, 1, 4, 3, 3, 4, 4, 1, 4)
S5 <- c(1, 2, 5, 1, 3, 3, 5, 5, 1, 1, 5, 4, 4, 3, 2, 2)
S6 <- c(1, 1, 1, 3, 6, 5, 5, 5, 2, 2, 6, 5, 3, 4, 3, 4)
S7 <- c(1, 2, 2, 2, 3, 3, 3, 3, 2, 2, 1, 2, 3, 3, 1, 3)
S8 <- c(2, 2, 1, 2, 5, 5, 3, 3, 1, 1, 1, 5, 4, 4, 2, 3)
S9 <- c(3, 3, 1, 1, 2, 4, 2, 4, 1, 4, 1, 4, 2, 7, 6, 5)
S10 <- c(1, 1, 1, 2, 3, 4, 3, 3, 2, 4, 2, 3, 5, 3, 5, 4)
S11 <- c(1, 1, 5, 1, 7, 4, 9, 6, 1, 1, 2, 7, 8, 3, 5, 7)
S12 <- c(1, 1, 1, 1, 2, 5, 2, 2, 1, 1, 1, 1, 2, 3, 4, 3)
S13 <- c(1, 1, 1, 1, 1, 2, 3, 3, 1, 1, 3, 2, 1, 5, 4, 5)
S14 <- c(4, 4, 1, 1, 4, 4, 2, 2, 1, 1, 3, 4, 3, 2, 3, 3)
S15 <- c(2, 2, 1, 1, 5, 9, 3, 3, 6, 1, 7, 1, 4, 4, 8, 5)
S16 <- c(1, 1, 1, 1, 5, 5, 2, 2, 1, 1, 8, 1, 7, 3, 6, 4)
S17 <- c(2, 1, 1, 2, 3, 6, 3, 3, 2, 2, 2, 4, 3, 5, 4, 3)
S18 <- c(2, 2, 3, 2, 1, 4, 1, 5, 3, 2, 4, 3, 1, 1, 4, 1)
S19 <- c(2, 1, 1, 1, 5, 3, 4, 5, 1, 2, 6, 6, 3, 5, 6, 4)
S20 <- c(3, 2, 2, 3, 4, 5, 1, 1, 3, 3, 7, 5, 2, 9, 8, 6)
S21 <- c(3, 3, 2, 5, 4, 4, 4, 4, 6, 5, 7, 4, 2, 1, 3, 3)
S22 <- c(4, 6, 4, 4, 3, 2, 2, 2, 6, 2, 4, 1, 2, 5, 3, 5)
S23 <- c(3, 3, 1, 2, 1, 1, 4, 3, 1, 2, 3, 2, 5, 6, 3, 2)
S24 <- c(1, 1, 3, 3, 2, 6, 6, 4, 3, 3, 2, 1, 4, 2, 5, 2)
S25 <- c(2, 2, 7, 6, 2, 4, 4, 4, 1, 1, 8, 4, 7, 3, 2, 5)
S26 <- c(6, 6, 6, 1, 7, 5, 4, 5, 1, 1, 2, 7, 4, 3, 8, 9)
S27 <- c(1, 1, 3, 1, 5, 3, 2, 2, 3, 3, 3, 1, 1, 2, 4, 4)
S28 <- c(2, 2, 3, 3, 4, 5, 1, 7, 3, 3, 3, 3, 6, 1, 4, 4)
S29 <- c(8, 2, 4, 4, 12, 9, 1, 6, 3, 4, 7, 9, 11, 10, 11, 5)
S30 <- c(1, 1, 1, 1, 1, 5, 5, 5, 1, 4, 1, 3, 3, 5, 2, 2)

df <- data.frame(aromas, S1, S2, S3, S4, S5, S6, S7, S8, S9, S10,
                 S11, S12, S13, S14, S15, S16, S17, S18, S19, S20,
                 S21, S22, S23, S24, S25, S26, S27, S28, S29, S30)
```

Guardamos el data frame.

```
save(df, file = "datos/Aromas.RData")
```

3 Análisis de la estructura del dominio

Dos estímulos (olores) se consideran "próximos" si muchos sujetos los pusieron juntos en el mismo grupo. La idea es generar una matriz simétrica donde cada celda indica qué tan frecuentemente dos estímulos fueron agrupados juntos por los participantes.

3.1 Crear la matriz de proximidades

Lo primero que hacemos es crear una matriz binaria de co-clasificación de estímulos para un solo sujeto.

```
crear_matriz_proximidad <- function(df, sujeto) {
  aromas <- as.character(df$aromas)
  matriz <- matrix(0, nrow = length(aromas), ncol = length(aromas),
                   dimnames = list(aromas, aromas))
  clasificacion <- df[[sujeto]]
  for (grupo in unique(clasificacion)) {
    indices <- which(clasificacion == grupo)
    if (length(indices) > 1) {
      for (i in 1:(length(indices) - 1)) {
        for (j in (i + 1):length(indices)) {
          a1 <- aromas[indices[i]]
          a2 <- aromas[indices[j]]
          matriz[a1, a2] <- 1
          matriz[a2, a1] <- 1
        }
      }
    }
  }
  return(matriz)
}
```

La entrada son:

- `df`, un data frame, donde:
  - La primera columna `(df$aromas)` contiene los nombres de los estímulos (olores).
  - Las siguientes columnas son las clasificaciones de los sujetos.
- `sujeto`: el nombre de la columna que representa a un sujeto.

El proceso sigue los siguientes pasos:

1. Inicializa una `matriz` cuadrada, con filas y columnas iguales al número de estímulos (olores) y nombres de filas/columnas iguales a los aromas.
2. Extrae las agrupaciones del sujeto (`clasificacion <- df[[sujeto]]`).
3. Para cada grupo identificado por ese sujeto:

- Encuentra qué aromas están en ese grupo (`indices`).
- Marca con `1` en la matriz todas las combinaciones de pares dentro del grupo (es decir, si limón y pomelo están juntos, entonces matriz[“limón”, “pomelo”] = 1).

La salida es una matriz binaria donde:

- `1` indica que dos aromas fueron agrupados juntos.
- `0` indica que no lo fueron.

A continuación creamos una función para sumar todas las matrices de proximidad individuales y obtener una matriz promedio, que refleja qué tan frecuentemente cada par de estímulos fue agrupado por los sujetos.

```
crear_matriz_proximidad_total <- function(df) {
  sujetos <- colnames(df)[-1]
  n_sujetos <- length(sujetos)
  aromas <- as.character(df$aromas)
  matriz_suma <- matrix(0, nrow = length(aromas), ncol = length(aromas),
                        dimnames = list(aromas, aromas))
  for (sujeto in sujetos) {
    matriz_sujeto <- crear_matriz_proximidad(df, sujeto)
```

```
    matriz_suma <- matriz_suma + matriz_sujeto
  }
  diag(matriz_suma) <- n_sujetos
  return(matriz_suma / n_sujetos)
}
matriz_proximidades <- crear_matriz_proximidad_total(df)
```

La entrada es:

-`df`: el mismo data frame con aromas y clasificaciones de los sujetos.

El proceso sigue los siguientes pasos:

1. Extrae todos los nombres de sujetos (`colnames(df)[-1]`).
2. Inicializa una matriz de suma (`matriz_suma`).
3. Para cada sujeto:

- Crea su matriz de proximidad individual con crear_matriz_proximidad().
- La suma a la matriz acumulativa.

4. La diagonal se fija al número total de sujetos (`diag(matriz_suma) <- n_sujetos`) para indicar que cada estímulo coincide consigo mismo siempre.
5. Se divide la matriz entre el número de sujetos, generando una matriz de proporciones (entre 0 y 1).

La salida es una matriz de proximidades promedio, donde:

- Cada celda [`i,j`] indica el porcentaje de sujetos que agruparon juntos el par de aromas i y j.

El resultado puede verse en la Tabla 1.

```
  knitr::kable(matriz_proximidades [, 1:8],
               format = "latex",
               booktabs = TRUE,
               caption = "Matriz de proximidades entre olores (truncada)") %>%
    kableExtra::kable_styling(latex_options = c("striped", "hold_position"),
                              font_size = 8)
```

Tabla 1: Matriz de proximidades entre olores (truncada)

	limón	pomelo	piña	pera	miel	mantequilla	pan tostado	avellana tostada
limón	1.0000000	0.7000000	0.3333333	0.5000000	0.1333333	0.0333333	0.0000000	0.0666667
pomelo	0.7000000	1.0000000	0.4333333	0.4333333	0.1333333	0.0666667	0.0000000	0.0666667
piña	0.3333333	0.4333333	1.0000000	0.5333333	0.1333333	0.1000000	0.0333333	0.0333333
pera	0.5000000	0.4333333	0.5333333	1.0000000	0.0666667	0.0333333	0.0000000	0.0333333
miel	0.1333333	0.1333333	0.1333333	0.0666667	1.0000000	0.2666667	0.3000000	0.3000000
mantequilla	0.0333333	0.0666667	0.1000000	0.0333333	0.2666667	1.0000000	0.2666667	0.3000000
pan tostado	0.0000000	0.0000000	0.0333333	0.0000000	0.3000000	0.2666667	1.0000000	0.6333333
avellana tostada	0.0666667	0.0666667	0.0333333	0.0333333	0.3000000	0.3000000	0.6333333	1.0000000
fresa	0.3666667	0.3666667	0.5666667	0.6666667	0.1000000	0.1000000	0.0000000	0.0333333
frambuesa	0.3333333	0.2333333	0.3666667	0.5666667	0.0333333	0.1333333	0.0333333	0.0666667
cereza	0.2000000	0.1666667	0.3000000	0.2666667	0.1000000	0.1000000	0.0666667	0.1000000
grosella	0.2000000	0.2000000	0.2000000	0.2333333	0.2000000	0.3333333	0.1333333	0.1666667
pimiento verde	0.0666667	0.1333333	0.1666667	0.1000000	0.2333333	0.1000000	0.2333333	0.1666667
ahumado	0.0000000	0.0000000	0.0000000	0.0000000	0.2333333	0.1000000	0.2666667	0.2333333
pimienta	0.2000000	0.2000000	0.0666667	0.1000000	0.1333333	0.0333333	0.0333333	0.0666667
regaliz	0.0333333	0.0666667	0.0000000	0.0333333	0.2666667	0.1000000	0.2000000	0.1333333

Convertimos la matriz de proximidades en *data frame* y la guardamos en la carpeta ‘tablas“.

```r
matriz_proximidades <- as.data.frame(matriz_proximidades)
save(matriz_proximidades, file = "tablas/matriz_proximidades.RDta")
write_xlsx(matriz_proximidades, "tablas/matriz_proximidades.xlsx")
```

3.2 Mapa de calor

Nos aseguramos de que matriz_proximidades es una matriz y no un *data frame*.

```r
matriz_proximidades <- as.matrix(matriz_proximidades)
```

Convertimos la matriz a formato largo.

```r
matriz_larga <- melt(matriz_proximidades)
colnames(matriz_larga) <- c("Elemento1", "Elemento2", "Proximidad")
```

Nos aseguramos del orden de los factores.

```r
matriz_larga$Elemento1 <- factor(matriz_larga$Elemento1,
                                 levels = rownames(matriz_proximidades))
matriz_larga$Elemento2 <- factor(matriz_larga$Elemento2,
                                 levels = colnames(matriz_proximidades))
```

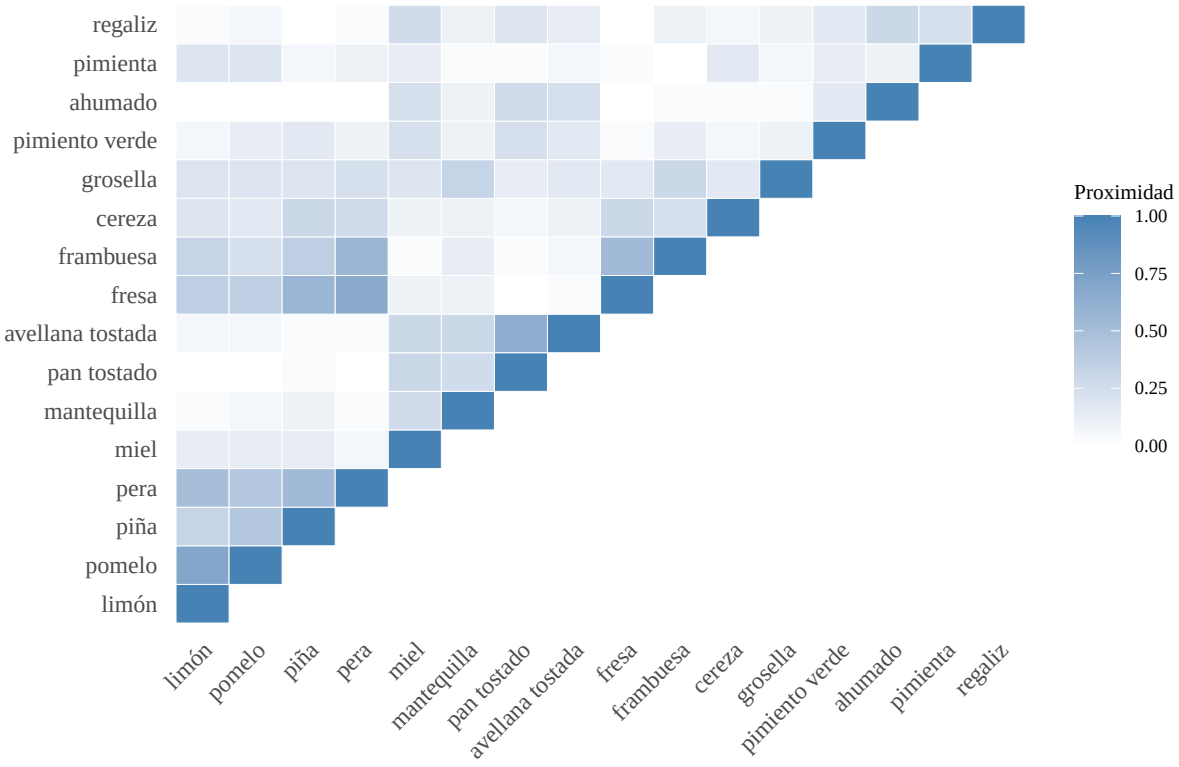
Filtramos solo la mitad superior de la matriz incluyendo la diagonal.

```r
matriz_larga_filtrada <- matriz_larga %>%
  filter(as.integer(Elemento1) <= as.integer(Elemento2))
```

Dibujamos el mapa de calor.

```r
ggplot(matriz_larga_filtrada, aes(x = Elemento1, y = Elemento2, fill = Proximidad)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "steelblue") +
  labs(title = "Figura 1: Similitud entre olores (Mapa de calor)",
       x = "", y = "") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, family = "Times"),
    axis.text.y = element_text(family = "Times"),
    plot.title = element_text(size = 10, family = "Times"),
    legend.title = element_text(size = 8, family = "Times"),
    legend.text = element_text(size = 7, family = "Times"),
    panel.grid = element_blank()
  )
```

Figura 1: Similitud entre olores (Mapa de calor)



3.3 Escalado multidimensional (MDS)

Para realizar el MDS transformamos la matriz de proximidades en matriz de distancias.

```
matriz_distancias <- as.dist(1 - matriz_proximidades)
```

Obtenemos las coordenadas y el ajuste del MDS. En este caso el stress es 12,7%, lo que indica que el escalado multidimensional es una buena representación de los datos en dos dimensiones.

```
fit <- isoMDS(matriz_distancias, k = 2)
```

```
## initial  value 16.164491
## final  value 12.710518
## converged
```

```
x <- fit$points[, 1]
y <- fit$points[, 2]
fit
```

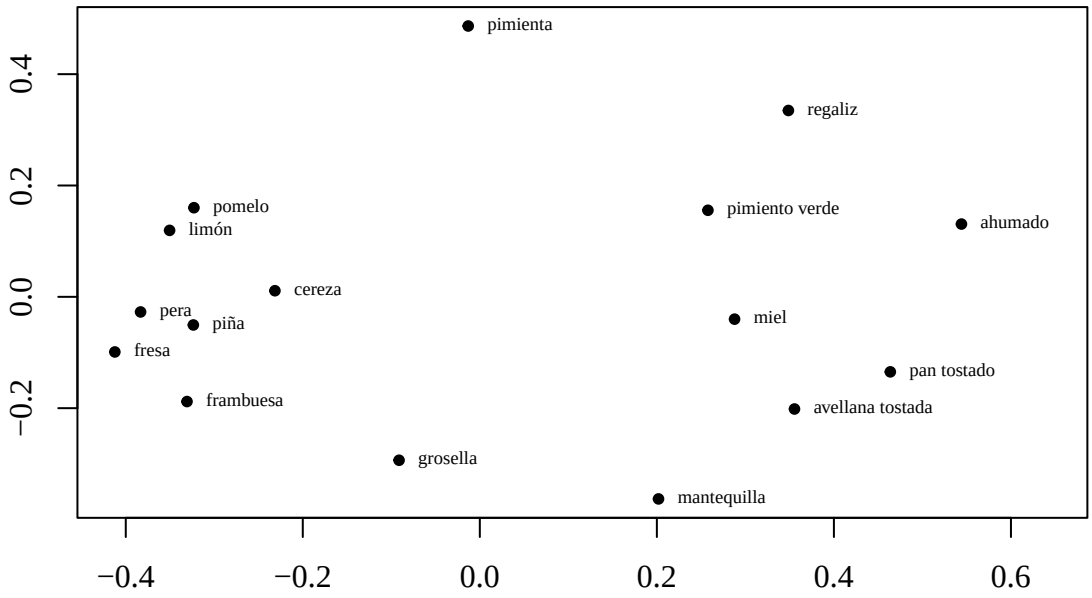
```
## $points
##                         [,1]        [,2]
## limón            -0.35044785  0.11935918
## pomelo           -0.32294712  0.16007144
## piña             -0.32362062 -0.05053717
## pera             -0.38324979 -0.02718538
## miel              0.28776941 -0.04008888
## mantequilla       0.20186392 -0.36303129
## pan tostado       0.46374891 -0.13477541
## avellana tostada  0.35547645 -0.20150218
## fresa            -0.41231637 -0.09905651
```

```
## frambuesa        -0.33079704 -0.18815925
## cereza           -0.23147562  0.01096731
## grosella         -0.09122303 -0.29357602
## pimiento verde    0.25766705  0.15549414
## ahumado           0.54407492  0.13073925
## pimienta         -0.01312800  0.48650913
## regaliz           0.34860477  0.33477165
##
## $stress
## [1] 12.71052
```

Dibujamos el diagrama MDS.

```
plot(x, y, pch = 20,
     xlim = range(x)+ c(0.0, 0.1),
     xlab = "", ylab = "",
     main = expression(
       "Figura 2: Similitud entre olores (MDS)"),
     cex.main = 0.8,
     family = "Times"
     )
text(x, y, pos = 4, labels = aromas, cex = 0.6, family = "Times") # Añadir los nombres
```

Figura 2: Similitud entre olores (MDS)



3.4 Análisis de conglomerados jerárquico

Para realizar el análisis de conglomerados vamos a utilizar la biblioteca MASS. Realizamos el análisis con el algoritmo ward.D.

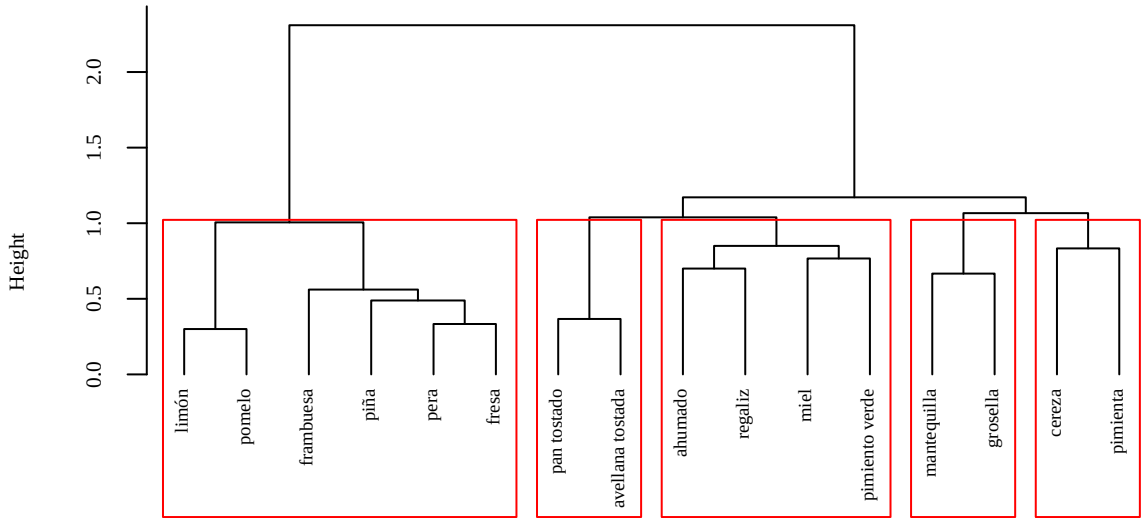
```
hclust_ward <- hclust(matriz_distancias, method = 'ward.D')
```

Dibujamos el dendrograma para visualizar el resultado del análisis.

```
plot(hclust_ward,
     hang = -1,
```

```
     cex = 0.6,
     main = expression("Figura 3: Similitud entre olores (Dedrograma)"),
     cex.main = 0.8,
     family = "Times",
     xlab = "",
     sub = "",
     cex.lab = 0.7,
     cex.axis = 0.7
     )
grupos <- cutree(hclust_ward, k = 5) # k = número de grupos elegido
rect.hclust(hclust_ward, k = 5, border = "red") # separar los grupos con rectángulos
```

Figura 3: Similitud entre olores (Dedrograma)



3.5 Grafo con igraph

Para crear un gráfico de red tenemos que usar la matriz de proximidades. Así que nos aseguramos de que es una matriz y no un data frame.

```
matriz_proximidades <- as.matrix(matriz_proximidades)
```

Establecemos la diagonal a 0 para evitar autoconexiones en el grafo.

```
diag(matriz_proximidades) <- 0
```

Creamos el grafo desde la matriz de proximidades.

```
g1 <- graph_from_adjacency_matrix(matriz_proximidades, mode = "undirected", weighted = TRUE)
```

Eliminamos las aristas con pesos bajos.

```
g1 <- delete.edges(g1, which(E(g1)$weight < 0.16))
```

Calculamos las medidas de red que vamos a utilizar en el grafo.

```
a <- E(g1)$weight
d <- degree(g1) # Grado: número de conexiones por nodo
```

Asignamos nombres a los nodos.

```r
if (is.null(V(g1)$name)) {
  V(g1)$name <- colnames(matriz_proximidades)
}
```

Asignamos colores a los nodos según los grupos del dendrograma.

```r
colores_grupo <- rainbow(length(unique(grupos)))
V(g1)$color <- colores_grupo[grupos]
```

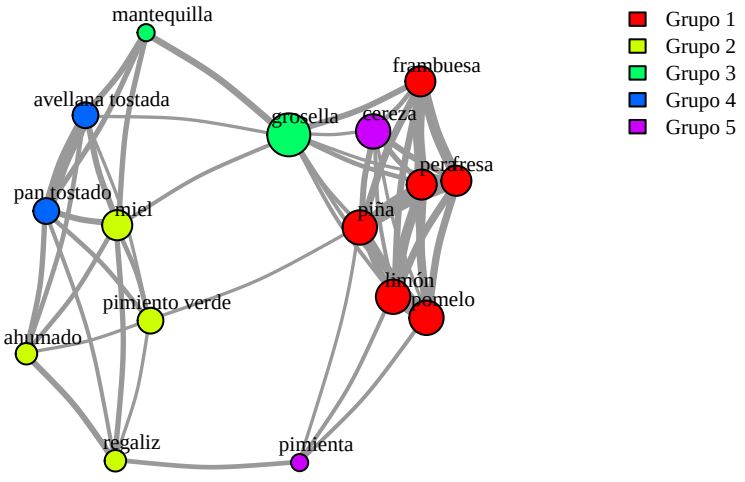
Finalmente, dibujamos el grafo.

```r
par(family = "Times")
set.seed(123)  # Layout reproducible
plot(g1,
     layout = layout_with_fr(g1),  # Fruchterman-Reingold
     vertex.size = d * 2,          # tamaño proporcional al grado
     vertex.label.cex = 0.7,
     vertex.label.color = "black",
     vertex.label.dist = 1.5,
     edge.width = a * 10,
     edge.color = "gray60",
     edge.curved = 0.1
     )

title(main = "Figura 4: Relaciones entre olores",
  cex.main = 0.8,
  family = "Times",
  font.main = 1
  )

legend("topright", legend = paste("Grupo", sort(unique(grupos))),
       fill = colores_grupo[sort(unique(grupos))], bty = "n", cex = 0.7)
```

Figura 4: Relaciones entre olores



4 Análisis de la similitud entre sujetos

Vamos a comparar cómo de similares son dos sujetos a la hora de agrupar los olores, si los clasifican en categorías similares. Para ello se:

- Convierte cada agrupación individual en una matriz de coclasificación
- Compara esas matrices entre sí mediante correlación de Pearson.

4.1 Crear la matriz de similitudes

Creamos una matriz de co-clasificación para un sujeto: indica para cada par de estímulos si fueron colocados en el mismo grupo.

La entrada es un vector con los grupos asignados por un sujeto a cada estímulo. Por ejemplo, el vector para `S1` es `c(5, 5, 4, 5, 4, 1, 1, 5, 5, 3, 3, 3, 4, 2, 2, 2)`.

La salida de la función es una matriz cuadrada donde: `1` indica que el par de estímulos están en el mismo grupo, y `0` indica que están en grupos diferentes.

Los elementos de la diagonal se asignan como NA porque no se comparan los estímulos consigo mismos.

```
generate_coclass_matrix <- function(grouping_vector) {
  n <- length(grouping_vector)
  matrix <- outer(grouping_vector, grouping_vector, FUN = function(x, y) as.integer(x == y))
  diag(matrix) <- NA
  return(matrix)
}
```

Comparamos dos matrices de co-clasificación (de dos sujetos diferentes) extrayendo solo los elementos del triángulo superior (para evitar duplicados) y calculamos la correlación de Pearson entre ellos.

`vec1` y `vec2`: son vectores con los pares de comparaciones entre estímulos (sin repetir ni contar la diagonal).

Se calcula la correlación entre estos vectores.

```
compare_coclass_matrices <- function(m1, m2) {
  vec1 <- m1[upper.tri(m1)]
  vec2 <- m2[upper.tri(m2)]
  return(cor(vec1, vec2, use = "complete.obs", method = "pearson"))
}
```

Construimos la matriz completa de similitud entre todos los sujetos.

Asumimos que el data frame tiene una estructura donde cada columna representa un sujeto y cada fila es un estímulo. Seguimos los siguientes pasos:

1. Para cada sujeto, se genera su matriz de co-clasificación.
2. Se compara cada par de sujetos usando compare_coclass_matrices.
3. Se llena la matriz de similitud con esos valores.
4. Se asigna 1 en la diagonal (porque un sujeto es siempre idéntico a sí mismo).

Resultado final es una matriz simétrica donde la entrada `(i, j)` representa la similitud (correlación) entre las clasificaciones del sujeto `i` y el sujeto `j`.

```
calculate_matriz_similitud <- function(df) {
  subject_names <- names(df)[2:ncol(df)]
  coclass_matrices <- lapply(subject_names, function(name) generate_coclass_matrix(df[[name]]))
  names(coclass_matrices) <- subject_names
  matriz_similitud <- matrix(0, length(subject_names), length(subject_names),
                             dimnames = list(subject_names, subject_names))
  for (i in 1:(length(subject_names) - 1)) {
```

```
    for (j in (i + 1):length(subject_names)) {
      sim <- compare_coclass_matrices(coclass_matrices[[i]], coclass_matrices[[j]])
      matriz_similitud[i, j] <- sim
      matriz_similitud[j, i] <- sim
    }
  }
  diag(matriz_similitud) <- 1
  return(matriz_similitud)
}

matriz_similitud <- calculate_matriz_similitud(df)
```

El resultado puede verse en la Tabla 2.

```
  knitr::kable(matriz_similitud [, 1:8],
               format = "latex",
               booktabs = TRUE,
               caption = "Matriz de similitud entre sujetos (truncada)") %>%
    kableExtra::kable_styling(latex_options = c("striped", "hold_position"),
                              font_size = 8)
```

Tabla 2: Matriz de similitud entre sujetos (truncada)

	S1	S2	S3	S4	S5	S6	S7	S8
S1	1.0000000	0.0458831	0.0458831	0.2110625	0.0510443	0.0338062	0.0238705	0.0510443
S2	0.0458831	1.0000000	0.1410526	0.1410526	0.0585519	-0.0077557	0.1259541	0.0585519
S3	0.0458831	0.1410526	1.0000000	0.0905263	0.0023421	-0.0698010	0.2135744	0.1709714
S4	0.2110625	0.1410526	0.0905263	1.0000000	0.0585519	0.1783803	0.2135744	0.2833910
S5	0.0510443	0.0585519	0.0023421	0.0585519	1.0000000	-0.0258842	0.0402090	0.1870766
S6	0.0338062	-0.0077557	-0.0698010	0.1783803	-0.0258842	1.0000000	0.0605228	0.1811894
S7	0.0238705	0.1259541	0.2135744	0.2135744	0.0402090	0.0605228	1.0000000	0.1864234
S8	0.0510443	0.0585519	0.1709714	0.2833910	0.1870766	0.1811894	0.1864234	1.0000000
S9	0.0877058	0.1006055	0.2816953	0.1006055	-0.0358151	0.1482499	0.1465507	0.1656447
S10	-0.0800000	0.2661223	0.1560027	0.1009429	-0.0102089	0.3042555	0.1193525	0.0510443
S11	0.2553999	0.1331654	0.1331654	0.2610041	0.2690304	0.0196229	0.1357862	0.1979211
S12	0.2541752	0.2125559	0.3931344	0.2577005	0.1234642	0.0900847	0.4550466	0.2239096
S13	0.2253524	0.1895644	0.0516994	0.2355194	0.1929941	0.2186735	0.0906507	0.1418699
S14	-0.0642364	0.1410526	0.0400000	0.0400000	0.1147617	0.1163350	0.2135744	0.2271812
S15	0.0269680	-0.0061869	0.1422983	0.0680557	0.0344141	0.0683763	0.3701502	0.1996017
S16	0.1798949	0.1672550	0.2715179	0.3236493	0.2527635	0.1360348	0.4305481	0.2527635
S17	0.0770514	0.1281568	0.3402785	0.0751264	0.2074677	0.0162801	0.3609539	0.2074677
S18	0.0770514	0.0751264	0.1281568	0.1281568	0.0304810	-0.0488402	0.4529166	0.1484722
S19	0.1014185	0.1783803	0.3645164	0.3024710	0.0431403	-0.0666667	0.3833108	0.0431403
S20	0.1678993	0.2637052	0.1214821	0.1214821	0.4159894	0.2292243	0.0878641	0.1786579
S21	0.0626224	0.0143666	-0.0430997	0.1292991	0.2013808	0.3351955	0.1071294	0.2653112
S22	0.0510443	0.1147617	0.0585519	0.1709714	-0.0005211	0.1811894	0.1864234	-0.0005211
S23	0.0770514	-0.0309344	0.0220960	0.0751264	0.0304810	-0.1139606	0.0390842	0.0894766
S24	0.0748118	0.0269704	0.0858150	0.0858150	0.0856484	0.2077475	0.3303719	0.0856484
S25	0.0464363	0.0053266	-0.1225121	0.0692460	0.0557023	0.4905734	0.0249403	0.2690304
S26	0.0424476	0.0097382	0.0097382	0.2434539	0.1365028	0.3707063	0.2077155	0.1365028
S27	0.0559017	0.1025978	0.2051957	-0.0512989	0.1255525	0.0629941	0.0533761	0.3538296
S28	0.1600000	0.1009429	0.2110625	0.0458831	0.1122975	-0.0338062	0.2625755	0.2348039
S29	-0.0932505	0.0984084	-0.0042786	-0.0042786	0.0237995	0.1733843	0.1224264	0.1380373
S30	0.2176383	0.1897327	0.2376651	0.2376651	0.1817465	0.1986530	0.1485808	0.1284224

Convertimos la matriz de similitud en *data frame* y la guardamos en la carpeta ‘tablas“.

```r
matriz_similitud <- as.data.frame(matriz_similitud)
save(matriz_similitud, file = "tablas/matriz_similitud.RDta")
write_xlsx(matriz_similitud, "tablas/matriz_similitud.xlsx")
```

4.2 Mapa de calor de la similitud entre sujetos

Nos aseguramos de que matriz_similitud es una matriz y no un *data frame*.

```r
matriz_similitud <- as.matrix(matriz_similitud)
```

Convertimos la matriz a formato largo

```r
matriz_larga <- melt(matriz_similitud)
colnames(matriz_larga) <- c("Elemento1", "Elemento2", "Proximidad")
```

Aseguramos el orden de los factores.

```r
matriz_larga$Elemento1 <- factor(matriz_larga$Elemento1, levels = rownames(matriz_similitud))
matriz_larga$Elemento2 <- factor(matriz_larga$Elemento2, levels = colnames(matriz_similitud))
```

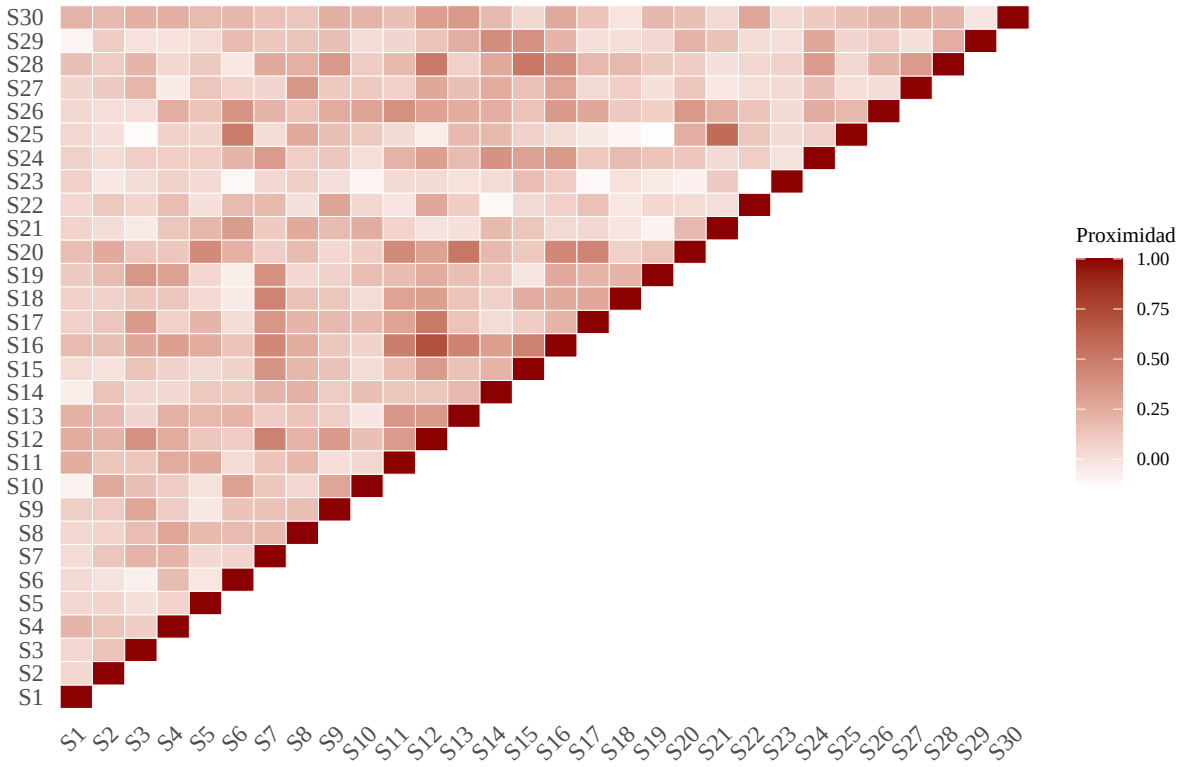
Filtramos solo la mitad superior de la matriz (incluyendo la diagonal).

```r
matriz_larga_filtrada <- matriz_larga %>%
  filter(as.integer(Elemento1) <= as.integer(Elemento2))
```

Dibujamos el gráfico:

```r
ggplot(matriz_larga_filtrada, aes(x = Elemento1, y = Elemento2, fill = Proximidad)) +
  geom_tile(color = "white") +
  scale_fill_gradient(low = "white", high = "darkred") +
  labs(title = "Figura 5: Similitud entre sujetos (Mapa de calor)",
       x = "", y = "") +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1, family = "Times"),
    axis.text.y = element_text(family = "Times"),
    plot.title = element_text(size = 10, family = "Times"),
    legend.title = element_text(size = 8, family = "Times"),
    legend.text = element_text(size = 7, family = "Times"),
    panel.grid = element_blank()
  )
```

Figura 5: Similitud entre sujetos (Mapa de calor)



4.3 Escalado multidimensional (MDS)

Transformamos la matriz de similitud en matriz de distancias

```
matriz_distancias <- as.dist(1-matriz_similitud)
```

Obtenemos las coordenadas y ajuste del MDS. En este caso el stress es 27,3%, por lo que debe interpretarse con cautela el gráfico. Para tener una idea más precisa de los grupos de sujetos, debemos hacer un análisis de conglomerados.

```
fit <- isoMDS(matriz_distancias, k = 2) # k es el número de dimensiones
```

```
## initial  value 36.825596
## iter   5 value 28.468820
## iter  10 value 27.500416
## final  value 27.315883
## converged
```

```
x <- fit$points[, 1]
y <- fit$points[, 2]
fit # ver coordenadas y el porcentaje de stress
```

```
## $points
##              [,1]         [,2]
## S1   0.623213382  0.635190927
## S2   0.317799019  0.638448179
## S3   0.751593952 -0.018338199
## S4   0.166666390  0.594367135
## S5  -0.667536073 -0.347540838
## S6  -0.737834602  0.379024859
```

```
## S7   0.462616859 -0.156420516
## S8  -0.258573119 -0.306362465
## S9  -0.226677856  0.487065644
## S10 -0.496080129  0.572849512
## S11  0.355239086  0.045526530
## S12  0.227429059  0.026982954
## S13 -0.055185042  0.126724058
## S14 -0.353222393 -0.464102004
## S15  0.175200697 -0.544081955
## S16  0.134851168 -0.143682473
## S17  0.508437657  0.218379428
## S18  0.708123361 -0.339624887
## S19  0.778794724  0.216828151
## S20 -0.215582535  0.028521722
## S21 -0.848010424  0.007658585
## S22 -0.136313672  0.931105756
## S23  0.001605272 -1.183408637
## S24  0.296234527 -0.409812663
## S25 -0.928156829  0.188844355
## S26 -0.349341730  0.144636382
## S27  0.038209692 -0.704235159
## S28  0.179113283 -0.426806293
## S29 -0.439781853 -0.621834026
## S30 -0.012831868  0.424095938
##
## $stress
## [1] 27.31588
```

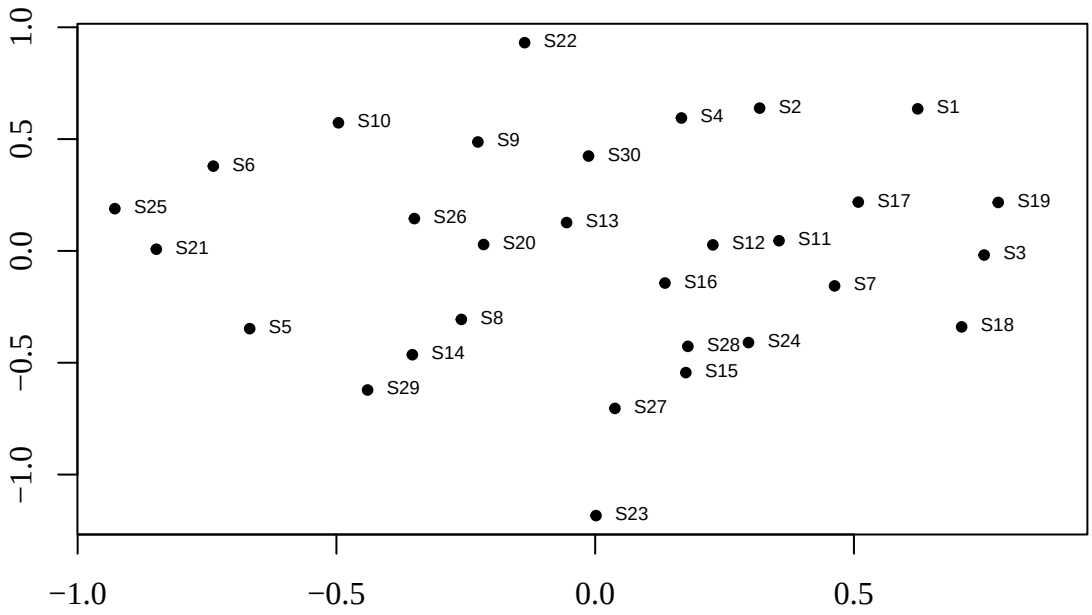
Dibujamos el diagrama MDS.

```
plot(x, y, pch = 20, xlim = range(x)+ c(0.0, 0.1),
     main = "Figura 6: Similitud entre sujetos (MDS)",
     xlab = "", ylab = "",
     font.main = 1,
     cex.main = 0.8,
     family ="Times"
     )

sujetos <- c("S1", "S2", "S3", "S4", "S5", "S6", "S7", "S8", "S9", "S10",
             "S11", "S12", "S13", "S14", "S15", "S16", "S17", "S18", "S19",
             "S20", "S21", "S22", "S23", "S24", "S25", "S26", "S27", "S28",
             "S29", "S30")

text(x, y, pos = 4, labels = sujetos, cex = 0.6) # Añadir los nombres
```

Figura 6: Similitud entre sujetos (MDS)



4.4 Análisis de conglomerados jerárquico

Realizamos el análisis de conglomerados con el algoritmo ward.D.

```
hclust_ward <- hclust(matriz_distancias, method = 'ward.D')
```

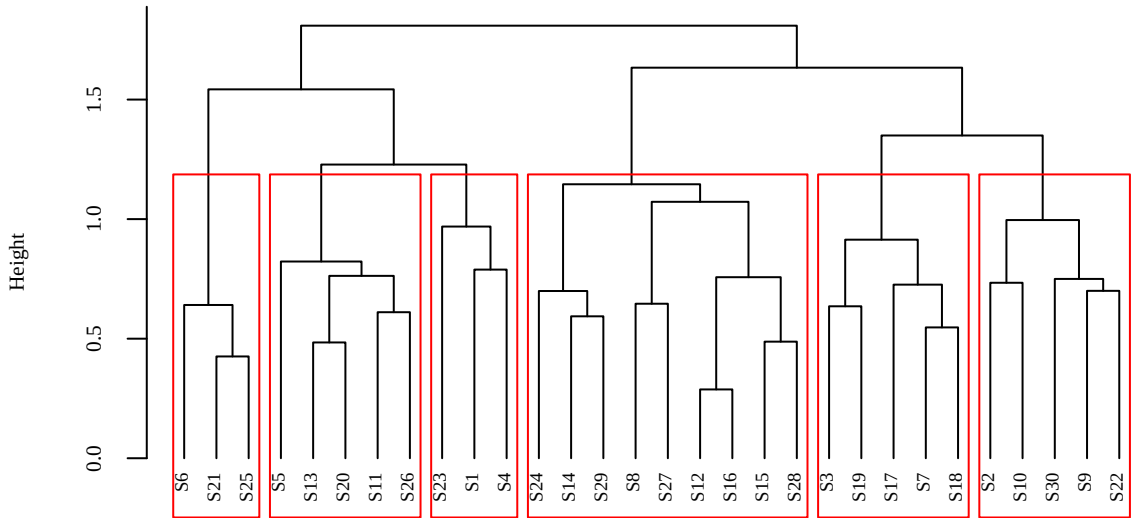
Dibujamos el dendrograma para ver el resultado del análisis de conglomerados.

```
plot(hclust_ward,
     hang = -1,
     cex = 0.6,
     main = expression("Figura 7: Similitud entre sujetos (Dendrograma)"),
     cex.main = 0.8,
     family = "Times",
     xlab = "",
     sub = "",
     cex.lab = 0.7,
     cex.axis = 0.7
     )
grupos <- cutree(hclust_ward, k = 6) # k = número de clusters
print(grupos) # Imprimir los grupos
```

```
##  S1  S2  S3  S4  S5  S6  S7  S8  S9 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S20
##   1   2   3   1   4   5   3   6   2   2   4   6   4   6   6   6   3   3   3   4
## S21 S22 S23 S24 S25 S26 S27 S28 S29 S30
##   5   2   1   6   5   4   6   6   6   2
```

```
rect.hclust(hclust_ward, k=6, border="red") #Separar los clusters con rectángulos.
```

Figura 7: Similitud entre sujetos (Dendrograma)



5 Análisis de clasificaciones con la biblioteca FreeSortR

También podemos analizar los datos de las clasificaciones de estímulos por parte de una muestra de sujetos utilizando la biblioteca FreeSortR, creada por Philippe Courcoux (https://github.com/cran/FreeSortR) (Courcoux, Faye, y Qannari 2014).

5.1 Datos

El data frame que se utiliza como ejemplo en esta biblioteca contiene 16 observaciones (aromas) 31 variables (sujetos).

La lista de aromas incluye: Lemon, Grapefruit, Pineapple, Pear, Honey, Butter, Grilledbread, Grilledhazelnut, Strawberry, Raspberry, Cherry, Blackcurrant, Greenpepper, Smoked, Pepper, Licorice.

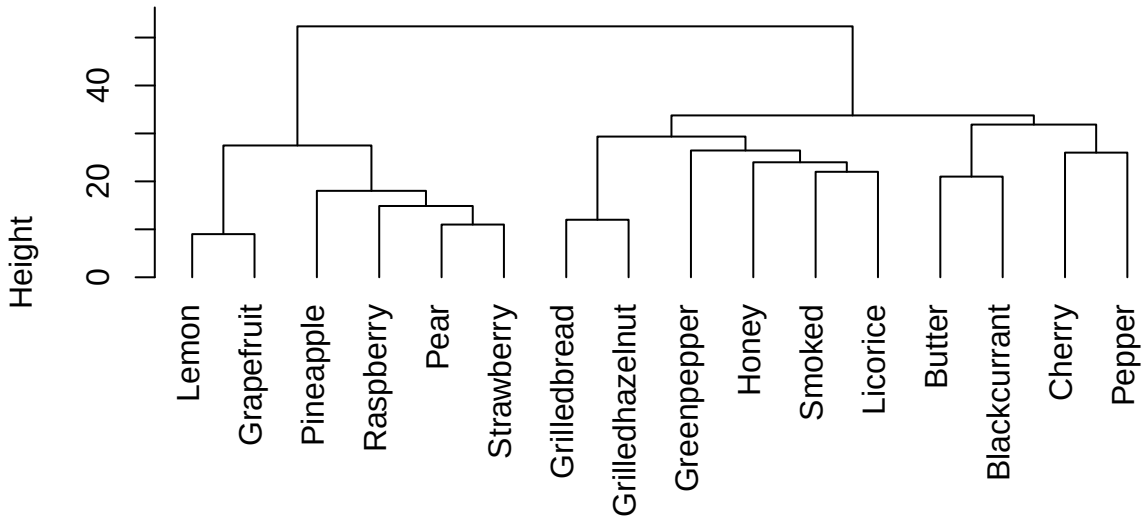
```
data(AromaSort)
```

5.2 Funciones utilizadas en el análisis

La función utilizada para manejar las clasificaciones de los estímulos es SortingPartition(). La función para computar la partición de consenso es ConsensusPartition(). Se puede realizar un escalado multidimensional de las clasificaciones con la función MdsSort().

Veámos en detalle cómo se aplica cada una de ellas.

```
Aroma<-SortingPartition(AromaSort)
ConsensusPartition(Aroma, ngroups = 0, type = "cutree", plotDendrogram = TRUE)
```




```
## $Consensus
##           Lemon      Grapefruit       Pineapple            Pear           Honey
##               1               1               1               1               2
##          Butter    Grilledbread Grilledhazelnut      Strawberry       Raspberry
##               3               4               4               1               1
##          Cherry    Blackcurrant     Greenpepper          Smoked          Pepper
##               5               3               2               2               5
##        Licorice
##               2
##
## $Crit
## [1] 0.2819987
```

El criterio para un consenso óptimo es la media del índice Rand ajustado entre el consenso y las particiones proporcionadas por los sujetos.

Si ngroups = 0, el consenso se calcula entre 2 y nstimuli-1, y se obtiene el mejor consenso.

Para type="cutree", el paso de inicialización se basa en cortar el árbol generado mediante la agrupación de los estímulos. Para type="fusion", el paso de inicialización se basa en el algoritmo de fusión. En este caso, los resultados son más precisos, pero el algoritmo puede requerir mucho tiempo. Para type="medoid", el consenso es la partición más cercana a todas las particiones proporcionadas por los sujetos.

Si queremos ver la matriz de coocurrencias entre los estímulos (el número de veces que dos estímulos han sido clasificados en el mismo grupo, podemos utilizar el comando `Coocurrences()`.

```
matriz_coocurrencias <- Cooccurrences(Aroma)
```

El resultado puede verse en la Tabla 3.

```
knitr::kable(matriz_similitud [, 1:9],
             format = "latex",
             booktabs = TRUE,
             caption = "Matriz de coocurrencias entre olores (truncada)") %>%
  kableExtra::kable_styling(latex_options = c("striped", "hold_position"),
                            font_size = 8)
```

Tabla 3: Matriz de coocurrencias entre olores (truncada)

	S1	S2	S3	S4	S5	S6	S7	S8	S9
S1	1.0000000	0.0458831	0.0458831	0.2110625	0.0510443	0.0338062	0.0238705	0.0510443	0.0877058
S2	0.0458831	1.0000000	0.1410526	0.1410526	0.0585519	-0.0077557	0.1259541	0.0585519	0.1006055
S3	0.0458831	0.1410526	1.0000000	0.0905263	0.0023421	-0.0698010	0.2135744	0.1709714	0.2816953
S4	0.2110625	0.1410526	0.0905263	1.0000000	0.0585519	0.1783803	0.2135744	0.2833910	0.1006055
S5	0.0510443	0.0585519	0.0023421	0.0585519	1.0000000	-0.0258842	0.0402090	0.1870766	-0.0358151
S6	0.0338062	-0.0077557	-0.0698010	0.1783803	-0.0258842	1.0000000	0.0605228	0.1811894	0.1482499
S7	0.0238705	0.1259541	0.2135744	0.2135744	0.0402090	0.0605228	1.0000000	0.1864234	0.1465507
S8	0.0510443	0.0585519	0.1709714	0.2833910	0.1870766	0.1811894	0.1864234	1.0000000	0.1656447
S9	0.0877058	0.1006055	0.2816953	0.1006055	-0.0358151	0.1482499	0.1465507	0.1656447	1.0000000
S10	-0.0800000	0.2661223	0.1560027	0.1009429	-0.0102089	0.3042555	0.1193525	0.0510443	0.2850439
S11	0.2553999	0.1331654	0.1331654	0.2610041	0.2690304	0.0196229	0.1357862	0.1979211	0.0101818
S12	0.2541752	0.2125559	0.3931344	0.2577005	0.1234642	0.0900847	0.4550466	0.2239096	0.3415808
S13	0.2253524	0.1895644	0.0516994	0.2355194	0.1929941	0.2186735	0.0906507	0.1418699	0.0878432
S14	-0.0642364	0.1410526	0.0400000	0.0400000	0.1147617	0.1163350	0.2135744	0.2271812	0.1006055
S15	0.0269680	-0.0061869	0.1422983	0.0680557	0.0344141	0.0683763	0.3701502	0.1996017	0.1478281
S16	0.1798949	0.1672550	0.2715179	0.3236493	0.2527635	0.1360348	0.4305481	0.2527635	0.1204097
S17	0.0770514	0.1281568	0.3402785	0.0751264	0.2074677	0.0162801	0.3609539	0.2074677	0.1942884
S18	0.0770514	0.0751264	0.1281568	0.1281568	0.0304810	-0.0488402	0.4529166	0.1484722	0.1309335
S19	0.1014185	0.1783803	0.3645164	0.3024710	0.0431403	-0.0666667	0.3833108	0.0431403	0.0741249
S20	0.1678993	0.2637052	0.1214821	0.1214821	0.4159894	0.2292243	0.0878641	0.1786579	0.0453100
S21	0.0626224	0.0143666	-0.0430997	0.1292991	0.2013808	0.3351955	0.1071294	0.2653112	0.1785014
S22	0.0510443	0.1147617	0.0585519	0.1709714	-0.0005211	0.1811894	0.1864234	-0.0005211	0.2999512
S23	0.0770514	-0.0309344	0.0220960	0.0751264	0.0304810	-0.1139606	0.0390842	0.0894766	0.0042237
S24	0.0748118	0.0269704	0.0858150	0.0858150	0.0856484	0.2077475	0.3303719	0.0856484	0.1218551
S25	0.0464363	0.0053266	-0.1225121	0.0692460	0.0557023	0.4905734	0.0249403	0.2690304	0.1629095
S26	0.0424476	0.0097382	0.0097382	0.2434539	0.1365028	0.3707063	0.2077155	0.1365028	0.2606033
S27	0.0559017	0.1025978	0.2051957	-0.0512989	0.1255525	0.0629941	0.0533761	0.3538296	0.1103153
S28	0.1600000	0.1009429	0.2110625	0.0458831	0.1122975	-0.0338062	0.2625755	0.2348039	0.3508232
S29	-0.0932505	0.0984084	-0.0042786	-0.0042786	0.0237995	0.1733843	0.1224264	0.1380373	0.1635722
S30	0.2176383	0.1897327	0.2376651	0.2376651	0.1817465	0.1986530	0.1485808	0.1284224	0.2366929

Para computar el MDS utilizamos el comando `MdsSort()`. Para ver las coordenadas empleamos el comando `getConfig()` y para conocer el stress `getStress()`. El comando `MdsDimChoice()` nos proporciona una tabla con el stress en distintas dimensiones.

```
Mds <- MdsSort(Aroma, ndim = 2, metric = FALSE) # 2 dimensiones, MDS no métrico
configuracion <- getConfig(Mds)
configuracion
```

```
##                       Dim 1       Dim 2
## Lemon           -0.57612704 -0.24859857
## Grapefruit      -0.50956414 -0.23787504
## Pineapple       -0.53833780  0.04480215
## Pear            -0.69819916 -0.04659969
## Honey            0.46300812  0.01216799
## Butter           0.36394948  0.57126745
```

```
## Grilledbread     0.80526588  0.27808923
## Grilledhazelnut  0.60430204  0.34572447
## Strawberry      -0.78082383  0.08311184
## Raspberry       -0.58970127  0.18494023
## Cherry          -0.45125212  0.36280746
## Blackcurrant    -0.08056357  0.39600430
## Greenpepper      0.36027178 -0.30273508
## Smoked           0.97026740 -0.15369263
## Pepper          -0.02359376 -0.83259852
## Licorice         0.68109800 -0.45681559
```

```
getStress(Mds)
```

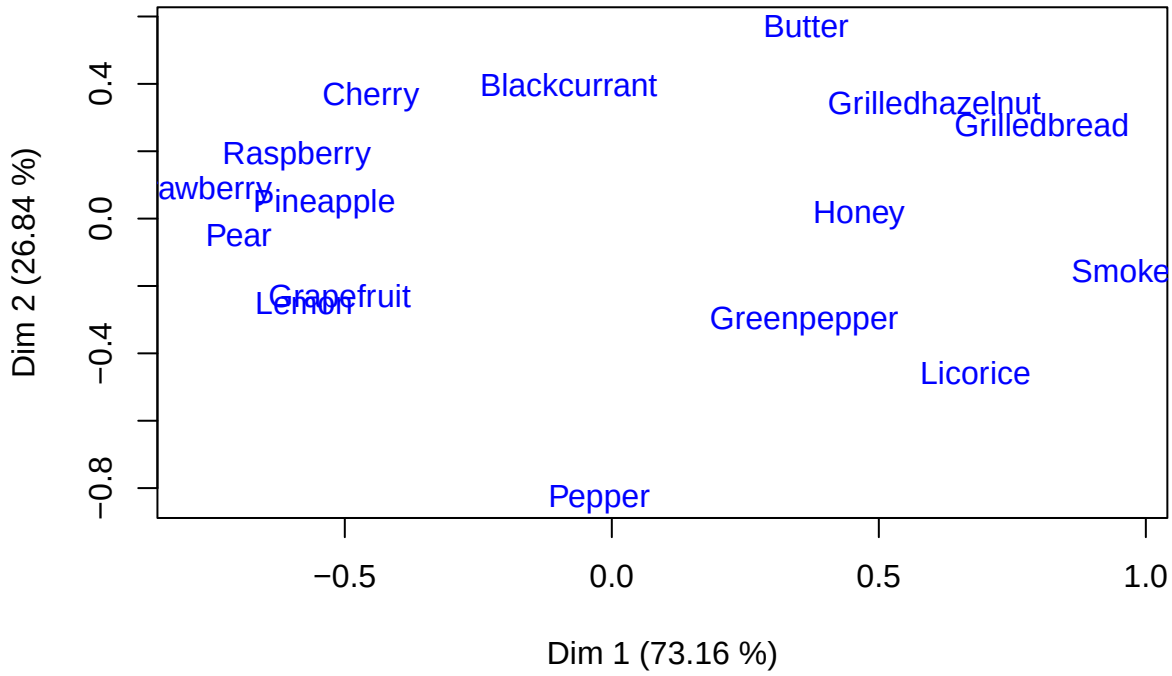
```
## [1] 0.1003636
```

```
MdsDimChoice(Aroma, dimen = c(2, 4), metric = FALSE)
```

```
##      Dimension     Stress
## [1,]         2 0.10036358
## [2,]         3 0.05995434
## [3,]         4 0.02883446
```

Finalmente dibujamos el diagrama.

```
plotMds(Mds, dim = c(1,2), col = "blue")
```



Para obtener un gráfico mejor del MDS podemos obtener la matriz de disimilitudes y usar la biblioteca `MASS`.

```
matriz_disimilitud <- DissTot(Aroma)
```

El resultado puede verse en la Tabla 4.

```
knitr::kable(matriz_disimilitud [, 1:9],
             format = "latex",
             booktabs = TRUE,
             caption = "Matriz de disimilitud entre olores (truncada)") %>%
```

```
    kableExtra::kable_styling(latex_options = c("striped", "hold_position"),
                              font_size = 8)
```

Tabla 4: Matriz de disimilitud entre olores (truncada)

	Lemon	Grapefruit	Pineapple	Pear	Honey	Butter	Grilledbread	Grilledhazelnut	Strawberry
Lemon	0	9	21	15	26	30	31	29	20
Grapefruit	9	0	18	17	26	29	31	29	20
Pineapple	21	18	0	15	27	28	30	30	14
Pear	15	17	15	0	28	30	31	30	11
Honey	26	26	27	28	0	23	22	22	28
Butter	30	29	28	30	23	0	22	22	28
Grilledbread	31	31	30	31	22	22	0	12	31
Grilledhazelnut	29	29	30	30	22	22	12	0	30
Strawberry	20	20	14	11	28	28	31	30	0
Raspberry	21	24	20	14	30	27	30	29	14
Cherry	25	26	22	23	28	27	28	28	22
Blackcurrant	25	25	24	24	25	21	27	26	26
Greenpepper	29	27	25	28	24	28	24	26	30
Smoked	31	31	31	31	24	28	23	24	31
Pepper	24	24	29	27	26	30	30	29	30
Licorice	30	29	31	30	23	27	24	27	31

Obtenemos las coordenadas y el ajuste del MDS. En este caso el stress es 13,7%, lo que indica que el escalado multidimensional es una buena representación de los datos en dos dimensiones.

```
fit <- isoMDS(matriz_disimilitud, k = 2)
```

```
## initial  value 18.315823
## iter   5 value 14.080981
## iter  10 value 13.714695
## iter  10 value 13.707415
## iter  10 value 13.705176
## final  value 13.705176
## converged
```

```
x <- fit$points[, 1]
y <- fit$points[, 2]
fit
```

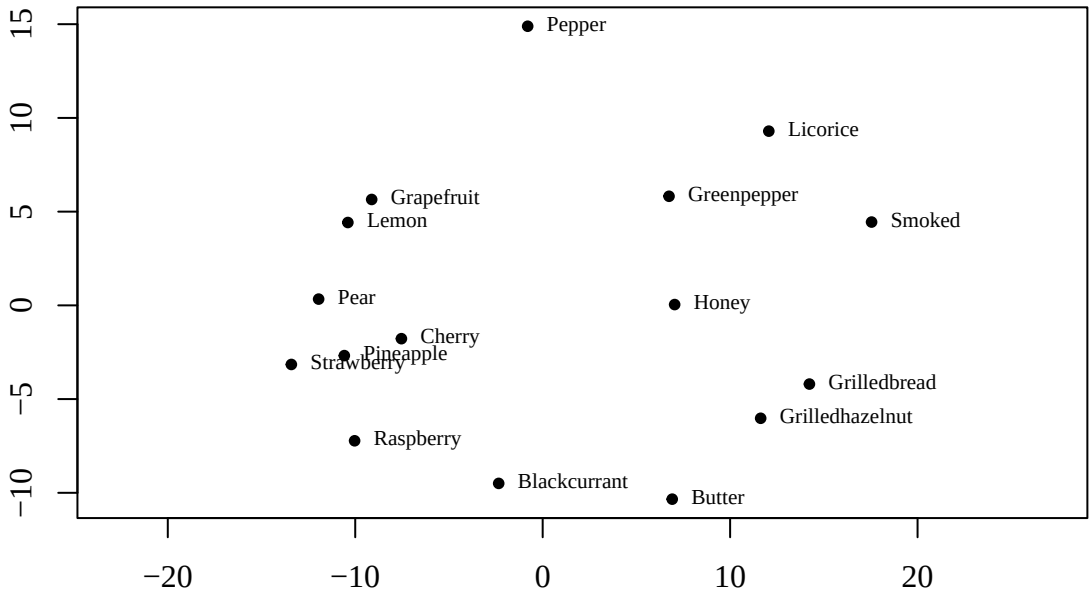
```
## $points
##                         [,1]        [,2]
## Lemon           -10.3924109   4.4198446
## Grapefruit       -9.1217401   5.6496105
## Pineapple       -10.5809659  -2.6808106
## Pear            -11.9519454   0.3331956
## Honey             7.0412771   0.0424424
## Butter            6.9121857 -10.3384211
## Grilledbread     14.2306061  -4.1982280
## Grilledhazelnut  11.6301255  -6.0233265
## Strawberry      -13.4065492  -3.1523478
## Raspberry       -10.0328608  -7.2244069
## Cherry           -7.5351268  -1.7780172
## Blackcurrant     -2.3453405  -9.4974136
## Greenpepper       6.7404347   5.8194380
## Smoked           17.5467033   4.4445678
```

```
## Pepper           -0.7991666  14.8900337
## Licorice         12.0647737   9.2938392
##
## $stress
## [1] 13.70518
```

Graficamos el MDS.

```
plot(x, y, pch = 20,
     xlim = range(x)+ c(0.0, 0.1),
     xlab = "", ylab = "",
     main = "Figura 8: Similitud entre olores (MDS)",
     font.main = 1,
     cex.main = 0.8,
     family = "Times",
     asp = 1)
text(x, y,
     pos = 4,
     labels = row.names(matriz_disimilitud),
     cex = 0.7, family = "Times") # Añadir los nombres
```

Figura 8: Similitud entre olores (MDS)



También podemos obtener un dendrograma más claro del análisis de conglomerados con la biblioteca `stats`, aplicando el algoritmo `ward.D2`, igual que hace `FreeSortR`.

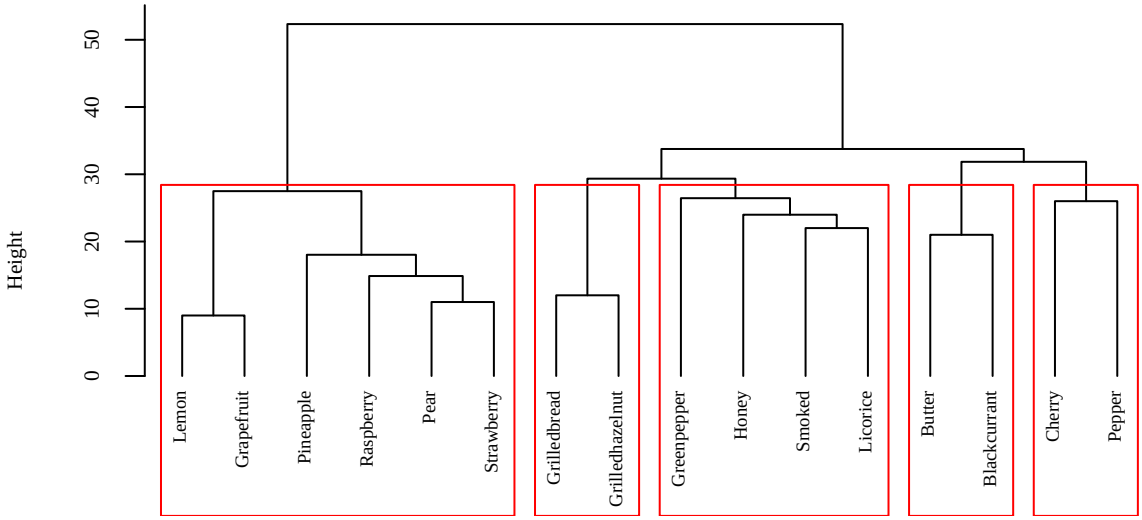
```
hclust_wardD2 <- stats::hclust(as.dist(matriz_disimilitud), method = 'ward.D2')
```

Dibujamos el dendrograma para visualizar el resultado del análisis.

```
plot(hclust_wardD2,
     hang = -1, cex = 0.6,
     main = expression("Figura 9: Similitud entre olores (Dedrograma)"),
     cex.main = 0.8,
     family = "Times",
     xlab = "",
     sub = "",
```

```
     cex.lab = 0.7,
     cex.axis = 0.7
     )
grupos <- cutree(hclust_wardD2, k = 5) # k = número de grupos elegido
rect.hclust(hclust_wardD2, k = 5, border = "red") # separar los grupos con rectángulos
```

Figura 9: Similitud entre olores (Dedrograma)



Referencias bibliográficas

Bernard, H. Russell, Amber Wutich, y Gery W. Ryan. 2016. *Analyzing qualitative data: Systematic approaches*. Sage Publications.

Borgatti, Stephen P. 1994. «Cultural domain analysis». *Journal of Quantitative Anthropology* 4 (4): 261-78.

Courcoux, Phillipe., Pauline Faye, y El Mostafa Qannari. 2014. «Determination of the consensus partition and cluster analysis of subjects in a free sorting task experiment». *Food quality and preference* 32: 107-12.

DeJordy, Rich, Stephen P. Borgatti, Chris Roussin, y Daniel S. Halgin. 2007. «Visualizing proximity data». *Field Methods* 19 (3): 239-63.

Weller, Susan C., y A. Kimball Romney. 1988. *Systematic data collection*. Sage Publications.