ACTIVIDAD FINAL

Feranando López Fernández-Freire

2023-05-09

Carga de Datos

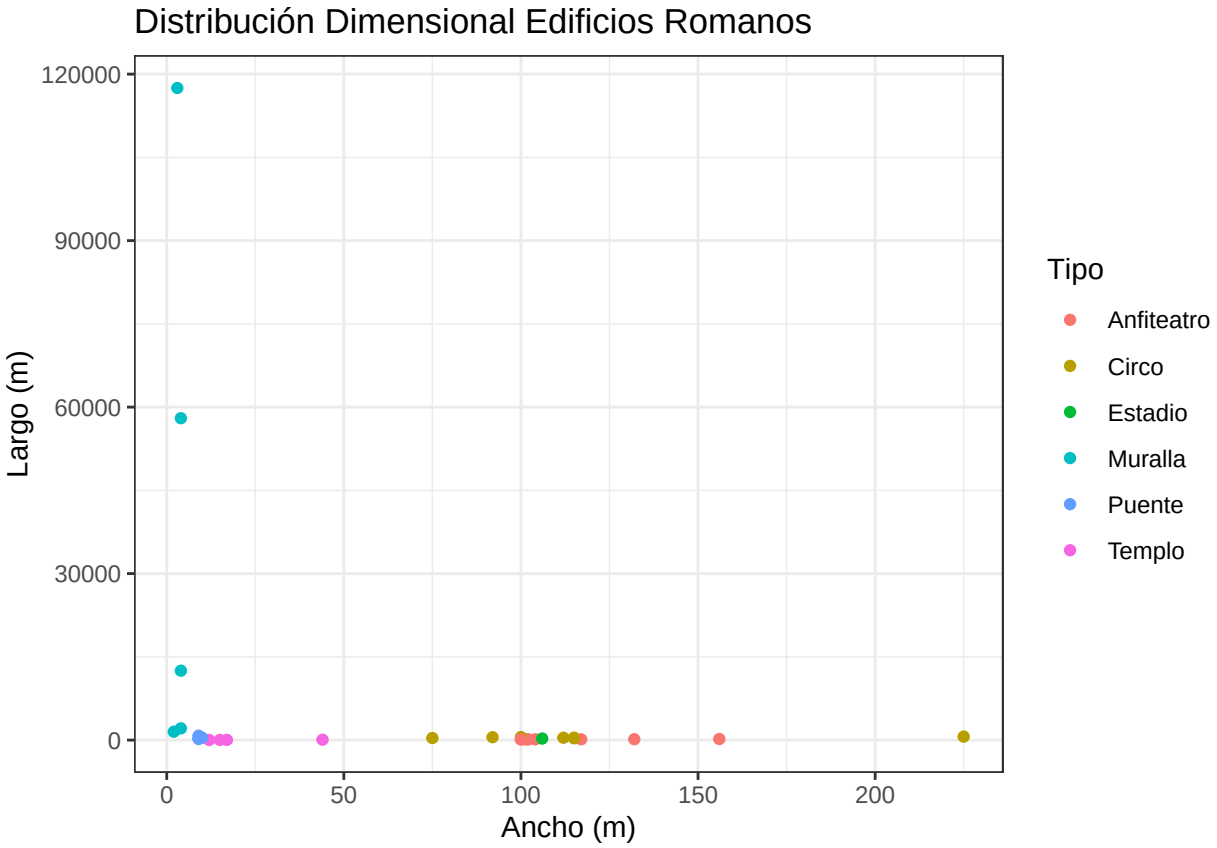
```
df.base <- read_excel('data_edificios_ludicos_romanos.xlsx') %>%
  select(-ID)

df.base <- df.base %>%
  column_to_rownames(var = "Edificio")

head(df.base)
```

```
##                             Tipo Largo Ancho
## Coliseo                Anfiteatro   190   156
## Circo Maximo                Circo   627   225
## Anfiteatro de Mérida   Anfiteatro   126   102
## Anfiteatro de Itálica  Anfiteatro   153   132
## Circo de Mérida             Circo   440   115
## Anfiteatro de Pompeya  Anfiteatro   135   104
##
## Coliseo                 Arquitectura del Coliseo | Cómo se construyó el Coliseo (romecolosseumtickets
## Circo Maximo             Circo Máximo de Roma - Información, historia y ubicación en Roma (disfrutaror
## Anfiteatro de Mérida                                    Anfiteatro de Mérida - Wikipedia, la enciclopedia
## Anfiteatro de Itálica Anfiteatro romano - Detalle - Conjunto arqueológico de Itálica (museosdeandalu
## Circo de Mérida                                      Circo romano de Mérida - Wikipedia, la enciclopedia
## Anfiteatro de Pompeya                          Anfiteatro romano de Pompeya - Wikipedia, la enciclopedia
```

Exploración y Preprocesado de Datos

En la siguiente imagen, se observa que algunos edificios tienen medidas muy grandes en comparación con el resto. Por ejemplo, las murallas llegan a tener casi 120,000 metros. Este fenómeno genera un problema de sesgo en los datos y causa que los algorítmos no sean capaces de ajustarse a las características de los datos.

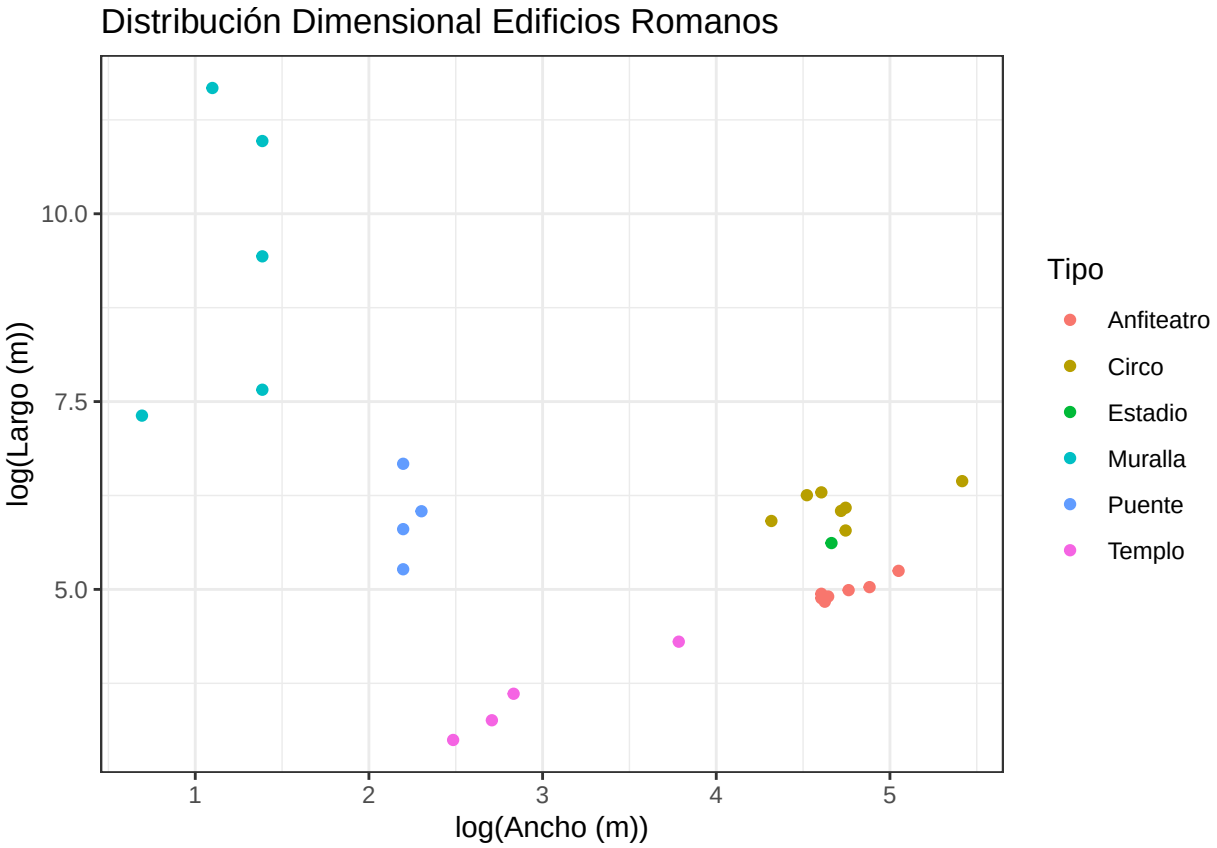
```
ggplot(data=df.base, mapping=aes(Ancho, Largo)) +
  geom_point(aes(color=Tipo)) +
  theme_bw() +
  labs(x='Ancho (m)', y='Largo (m)') +
  ggtitle('Distribución Dimensional Edificios Romanos')
```



Si se aplica una transformación logarítmica a las columnas, las observaciones distantes se contraerán, minimizando así el problema de los outliers.

En la siguiente gráfica, se aprecian mejor las diferencias entre los edificios tras haber realizado dicha transformación:

```
df <- df.base %>%
  mutate_if(is.numeric, log)

ggplot(data=df, mapping=aes(Ancho, Largo)) +
  geom_point(aes(color=Tipo)) +
  theme_bw() +
  labs(x='log(Ancho (m))', y='log(Largo (m))') +
  ggtitle('Distribución Dimensional Edificios Romanos')
```



Por otro lado, los algoritmos de cluster agrupan las observaciones en función de sus variaciones en los datos. Para que las variaciones del `Largo` y `Ancho` sean comparables, las variables deben ser normalizadas.

En otras palabras, las distribuciones de las variables pasan a tener la media en 0 y la desviación típica en 1.

$$X_{std} = \frac{X - \mu}{\sigma}$$

```
df.norm <- df %>%
  select_if(is.numeric) %>%
  mutate_all(function(x) (x - mean(x))/sd(x))

head(df.norm)
```

```
##                              Largo     Ancho
## Coliseo                -0.35322886 1.0219727
## Circo Maximo            0.26068457 1.2732121
## Anfiteatro de Mérida   -0.56443197 0.7305078
## Anfiteatro de Itálica  -0.46459719 0.9073756
## Circo de Mérida         0.07856987 0.8127985
## Anfiteatro de Pompeya  -0.52895592 0.7438284
```

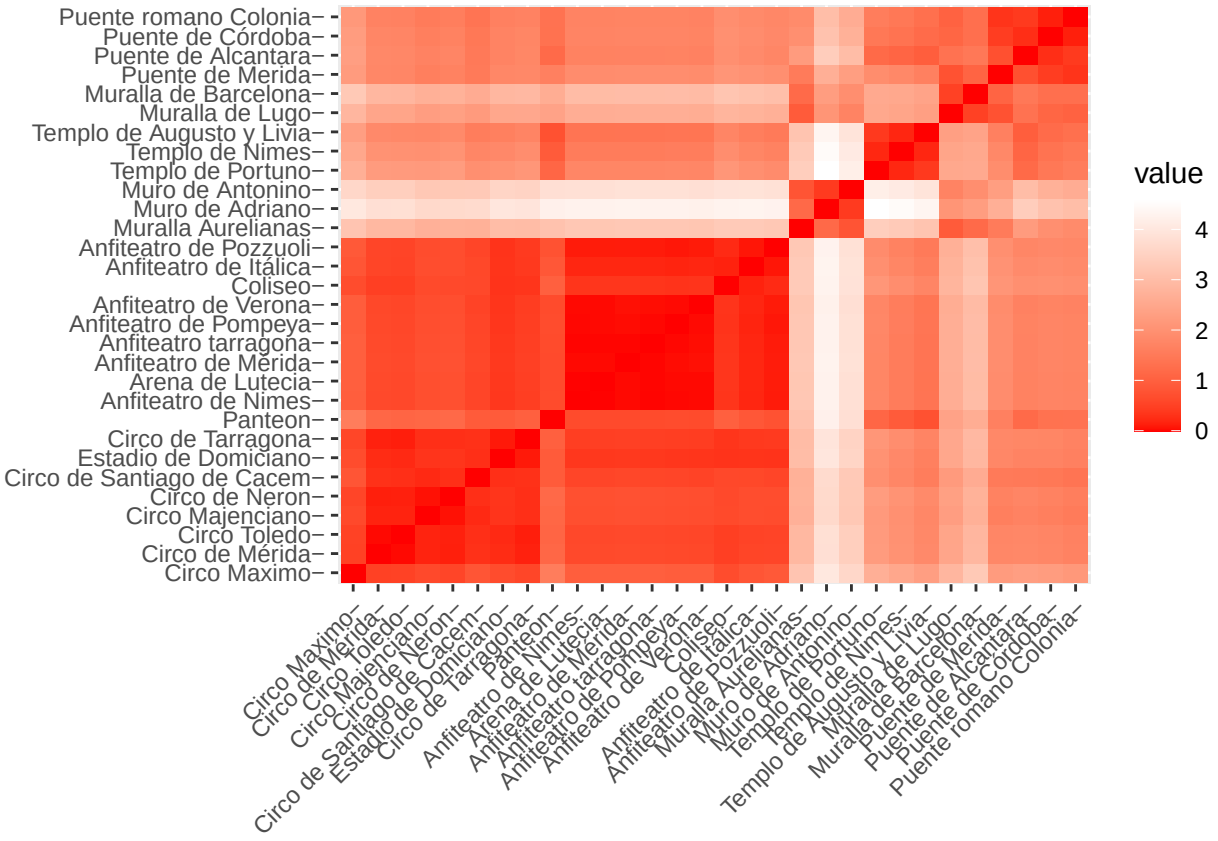
Una vez los datos están en condiciones óptimas de ser procesados por los algorítmos, es hora de pasar a los modelos de clusterización jerárquica, cuyos procesos involucran el cálculo de distancias.

Distancia Euclídea

Las distancias entre todas las combinaciones de edificios se muestran a continuación donde estos aparecen de manera ordenada en base al mayor número de homólogos que se encuentren cerca desde abajo hacia arriba en el eje vertical.

1. Los circos son muy similares a los anfiteatros (abajo)
2. Los puentes van por su lado (arriba), seguidos de los templos

```
distances <- dist(df.norm, method='euclidean')
fviz_dist(distances, gradient = list(low = "red", high = "white"))
```



Comparación Dendogramas

Los métodos de clusterización jerárquica que se usan en este estudio se basan el "Average Linkage" y el "Ward Linkage". Tras aplicar ambos métodos a los datos, el mejor de ellos será seleccionado para las conclusiones finales.

Visualización

```
model_average <- hclust(distances, method = 'average')
dd_average <- as.dendrogram(model_average)
```

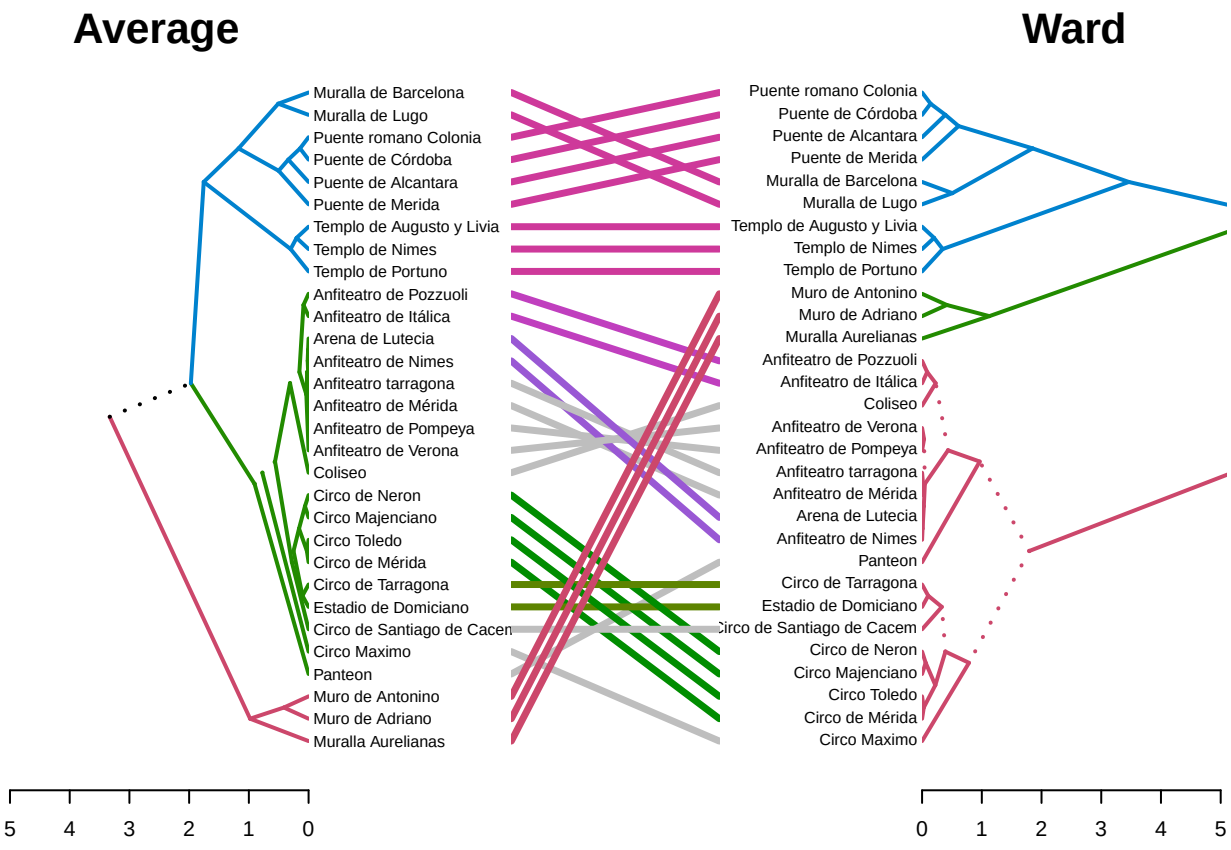
```
model_ward <- hclust(distances, method = 'ward.D2')
dd_ward <- as.dendrogram(model_ward)
```

```
library(dendextend)
dd_list <- dendlist(dd_average, dd_ward)
```

El método "Average" calcula la distancia promedio entre todos los pares de objetos de los dos clusters que se van a unir, mientras que el método "Ward" se centra en minimizar la intravarianza los clusters que se van a unir. Por ello, el "Average" es menos sensible a los outliers.

Sin embargo, estos ya han sido tratados, por lo que el linkage de "Ward" ofrece mejores resultados. Por ejemplo, las murallas se sitúan muy cerca entre sí, mientras que el método "Linkage" las separa al principio y final del cluster, respectivamente.

```
# using rank_branches can make the comparison even easier
tanglegram(dd_average, dd_ward, main_left = 'Average', main_right = 'Ward',
  lab.cex = 0.8, edge.lwd = 2,
  margin_inner = 8, type = "t", center = TRUE,
  dLeaf = -0.1, xlim = c(5.1, 0), columns_width = c(5, 2, 5),
  k_branches = 3
)
```



Correlación

¿Cómo de difernetes son las distintas metodologías?

- Las siguientes correlaciones servirán para medir las diferencias entre ambos métodos.
- Ambas presentan alrededor de un valor de 0.85, lo cual es muy alto e indica que ambos métodos son muy similares.

```
cor_cophenetic(dd_average, dd_ward)
```

Correlación Cofenética

```
## [1] 0.8411127
```

```
cor_bakers_gamma(dd_average, dd_ward)
```

Correlación Baker

```
## [1] 0.8515893
```

A continuación, el número óptimo de clusters será calculado a través de diferentes métodos para ver cómo se agrupan los edificios sin tener en cuenta el tipo.

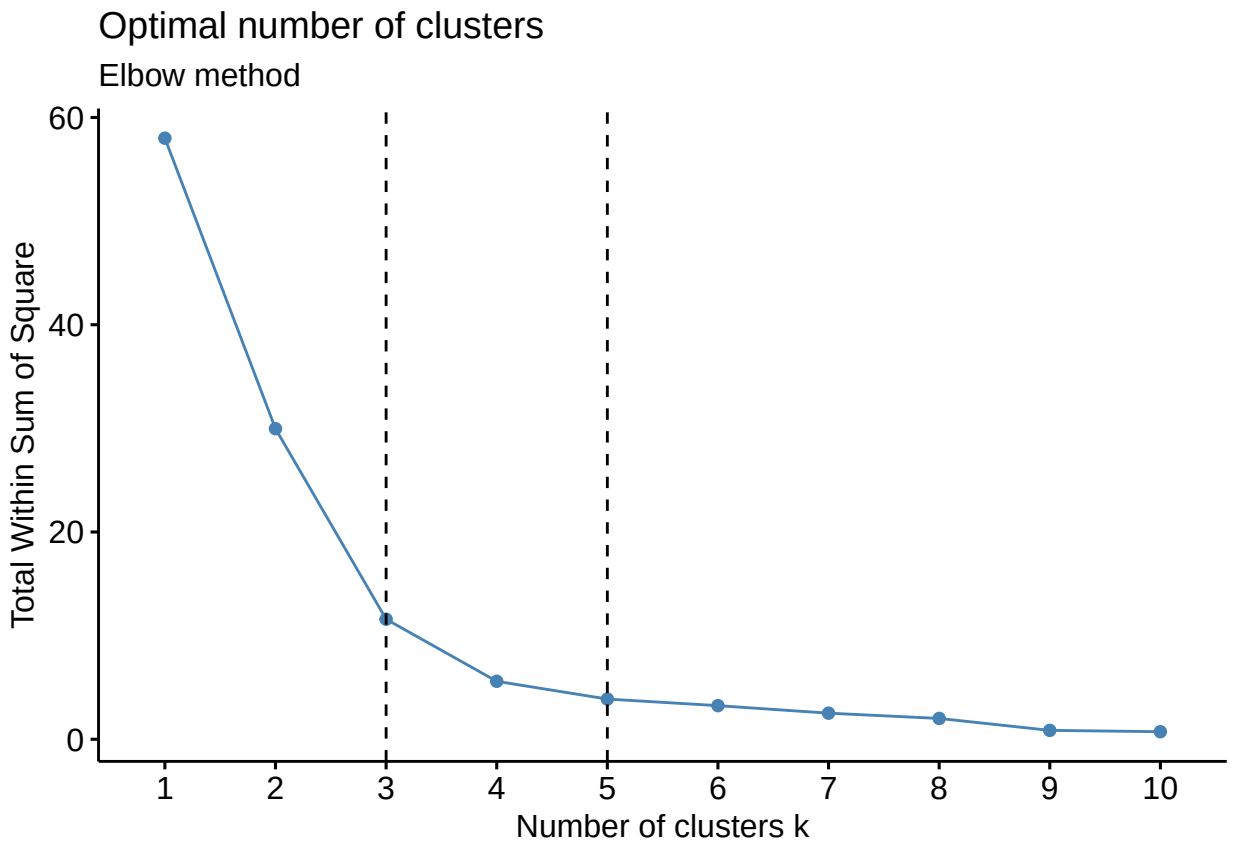
Clustering Jerárquico

Selección Número Óptimo de Clusters

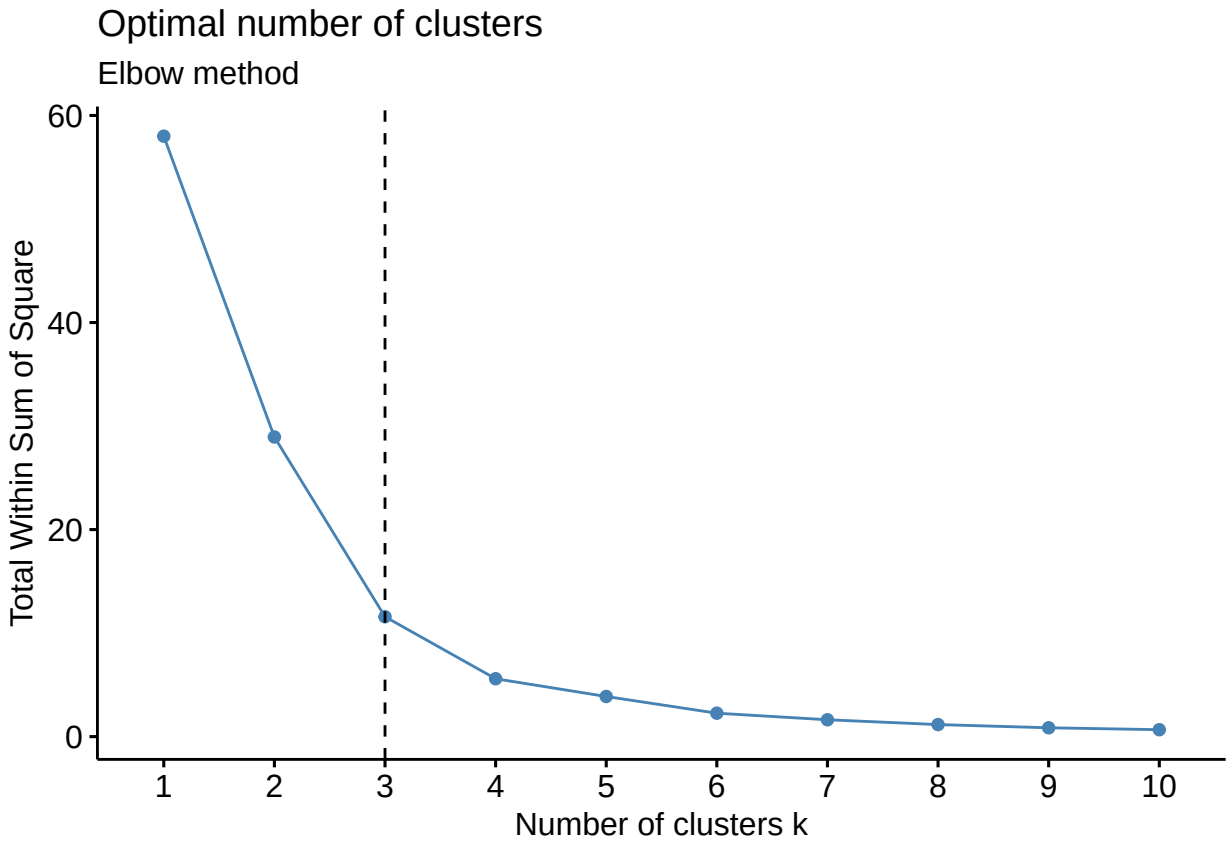
Average Linkage El método del codo sugiere los números de clusters más razonables. El "codo" se refiere al punto en el gráfico donde la disminución de la variabilidad total dentro de cada cluster comienza a disminuir a una tasa más lenta. En este punto, se considera que se ha encontrado el número óptimo de clusters para el conjunto de datos.

Tras ver las siguientes gráficas, podemos observar que el número óptimo de clusters para cada modelo gira en torno a 3, 4 y 5. En 3 se observa cómo la variación dentro de cada cluster no cambia tanto como en 4. Por tanto, `k=3`.

```
# Elbow method
fviz_nbclust(df.norm, hcut, hc_method = 'average', method = "wss") +
    geom_vline(xintercept = 3, linetype = 2) +
    geom_vline(xintercept = 5, linetype = 2) +
  labs(subtitle = "Elbow method")
```




```
# Elbow method
fviz_nbclust(df.norm, hcut, hc_method = 'ward.D2', method = "wss") +
    geom_vline(xintercept = 3, linetype = 2) +
  labs(subtitle = "Elbow method")
```



Ward Linkage

Como se ha concluido anteriormente, el mejor modelo usa el método "Ward" mientras que el número óptimo de clusters es 3:

Conclusiones Mejor Modelo

```
model_best <- model_ward
dd_best <- as.dendrogram(model_best)
k_best <- 3
clusters <- cutree(model_best, k = k_best) %>%
  as.factor()
```

Agregar las observaciones en torno al cluster

```
df %>%
  mutate(Cluster = clusters) -> df

df.base %>%
  mutate(Cluster = clusters) -> df.base
```

A continuación, se muestra una tabla agrupada con los valores más significativos de cada cluster.

1. Cluster 1: el edificio más común es el Anfiteatro, con un 50% de presencia en el cluster. Se caracteriza por tener un ancho mucho mayor al resto, mientras que el largo es el más corto.

2. Cluster 2: Puentes, representan el 44% de las observaciones del grupo, con valores medios comparados con el resto de grupos. En la gráfica, son los puntos que se sitúan en el centro.
3. Cluster 3: Las murallas representan un grupo para ellas solas debido a la enorme largura que tienen. Eso sí, con poco ancho.

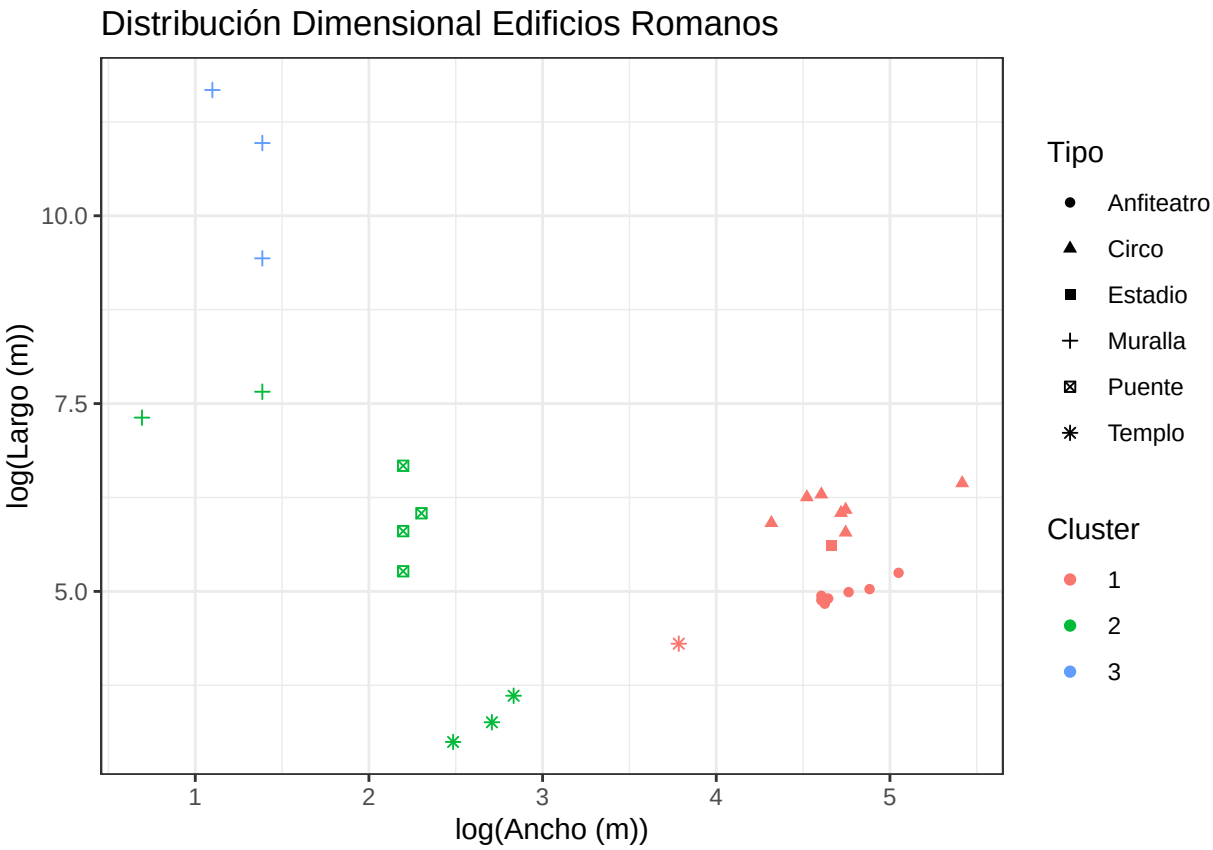
```
df.base %>%
  group_by(Cluster) %>%
  summarise(
    Tipo_Moda = mfv(Tipo)[1],
    Tipo_Porcentaje_Grupo = mean(Tipo_Moda == Tipo),
    Ancho_Media = mean(Ancho),
    Largo_Media = mean(Largo)
  )
```

```
## # A tibble: 3 x 5
##   Cluster Tipo_Moda  Tipo_Porcentaje_Grupo Ancho_Media Largo_Media
##   <fct>   <chr>                      <dbl>       <dbl>       <dbl>
## 1 1       Anfiteatro                 0.5        111           271
## 2 2       Puente                     0.444        9.67        604.
## 3 3       Muralla                    1            3.67      62667.
```

Visualizar Punto en el Espacio

En la siguiente gráfica se observa cómo los clusters agrupan bien los datos porque sus objetos están muy cerca entre sí. Sin embargo, las 2 murallas que presentan menor largo, no han podido ser clasificadas en el grupo de las murallas, sino con los estadios mayormente (cluster 2 verde).

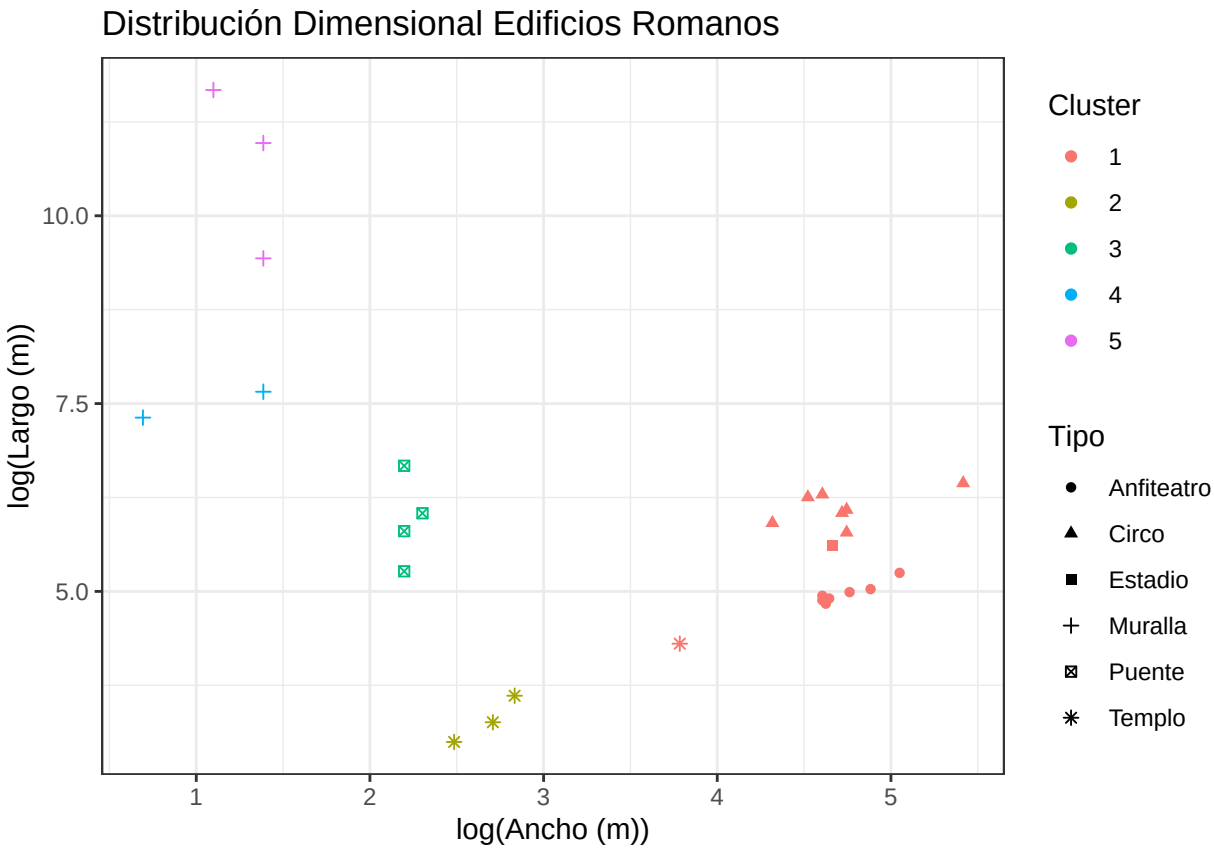
```
ggplot(data=df, mapping=aes(Ancho, Largo)) +
  geom_point(aes(shape=Tipo, color=Cluster)) +
  theme_bw() +
  labs(x='log(Ancho (m))', y='log(Largo (m))') +
  ggtitle('Distribución Dimensional Edificios Romanos')
```



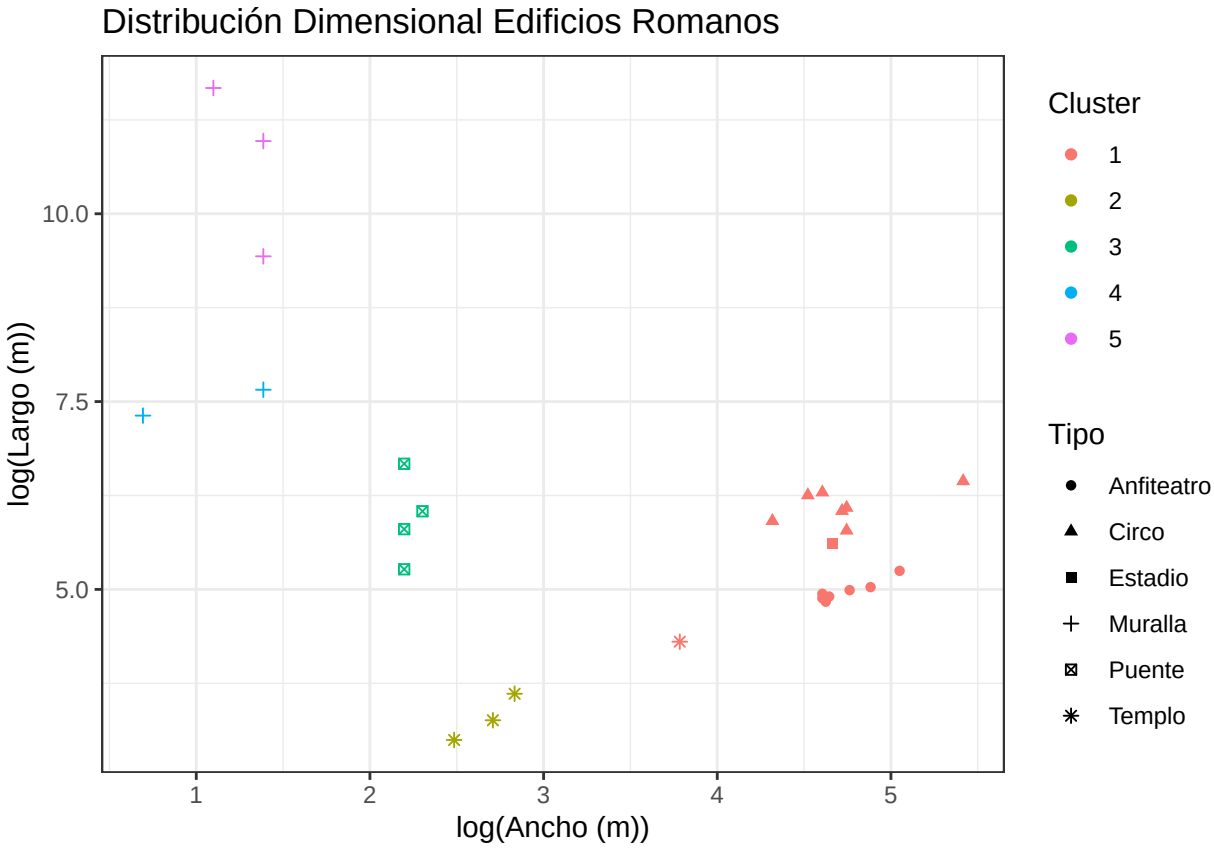
Dada la la distribución de los clusters, quizás sea conviniente agrupar por 5 clusters para separar en diferentes clusters todas las murallas, todos los puentes y todos los templos.

Al seleccionar 5, el modelo ha podido separar los estadios en un grupo único. Sin embargo, resto sigue mezclado. Por ello, lo mejor es dejar el número de clusters en 3 y el mejor Likage "Ward".

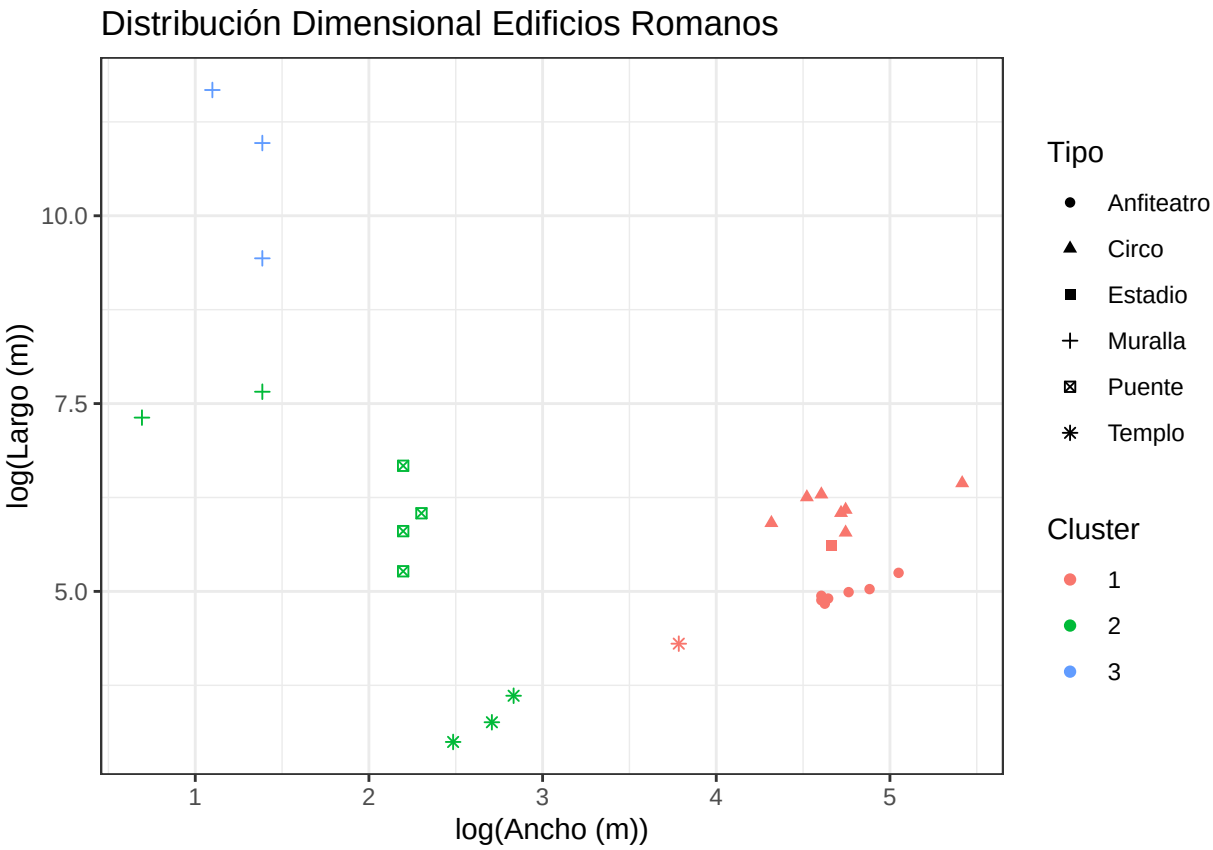
```
model_best <- model_ward
dd_best <- as.dendrogram(model_best)
k_best <- 5
clusters <- cutree(model_best, k = k_best) %>%
  as.factor()

df %>%
  mutate(Cluster = clusters) -> df

df.base %>%
  mutate(Cluster = clusters) -> df.base

ggplot(data=df, mapping=aes(Ancho, Largo)) +
  geom_point(aes(shape=Tipo, color=Cluster)) +
  theme_bw() +
  labs(x='log(Ancho (m))', y='log(Largo (m))') +
  ggtitle('Distribución Dimensional Edificios Romanos')
```




```
model_best <- model_average
dd_best <- as.dendrogram(model_best)
k_best <- 5
clusters <- cutree(model_best, k = k_best) %>%
  as.factor()

df %>%
  mutate(Cluster = clusters) -> df

df.base %>%
  mutate(Cluster = clusters) -> df.base

ggplot(data=df, mapping=aes(Ancho, Largo)) +
  geom_point(aes(shape=Tipo, color=Cluster)) +
  theme_bw() +
  labs(x='log(Ancho (m))', y='log(Largo (m))') +
  ggtitle('Distribución Dimensional Edificios Romanos')
```




```
model_best <- model_ward
dd_best <- as.dendrogram(model_best)
k_best <- 3
clusters <- cutree(model_best, k = k_best) %>%
  as.factor()

df %>%
  mutate(Cluster = clusters) -> df

df.base %>%
  mutate(Cluster = clusters) -> df.base

ggplot(data=df, mapping=aes(Ancho, Largo)) +
  geom_point(aes(shape=Tipo, color=Cluster)) +
  theme_bw() +
  labs(x='log(Ancho (m))', y='log(Largo (m))') +
  ggtitle('Distribución Dimensional Edificios Romanos')
```



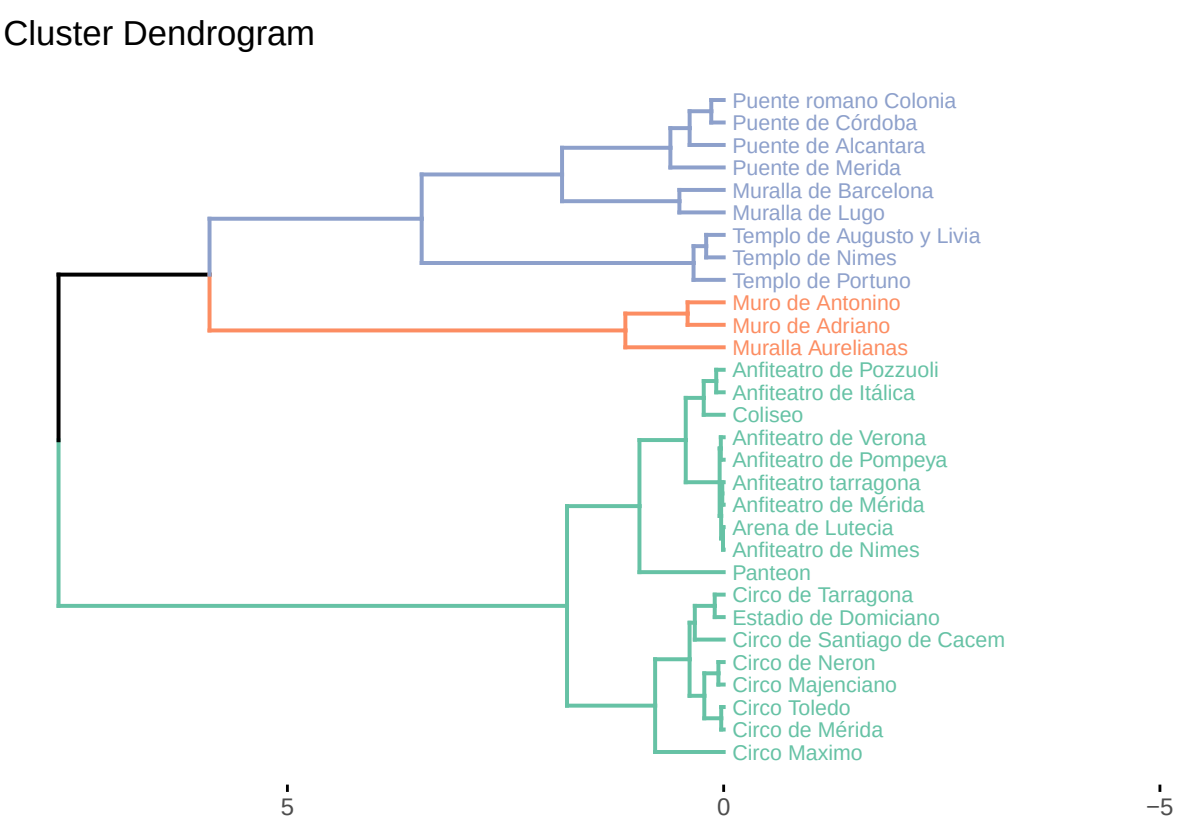
Visualizar Dendogramas

A continuación se muestran varias formas de visualizar los dendrogramas, que son compuestos aglormerando las observaciones conforme son más similares entre sí.

Cabe resaltar que, al distinguir 3 clusters, los puentes y templos se agrupan primero y después los muros. Aunque debido a magnitudes mínimas de las murallas de Barcelona y Lugo, estás se sitúan más cerca de los puentes que del cluster de murallas-muros.

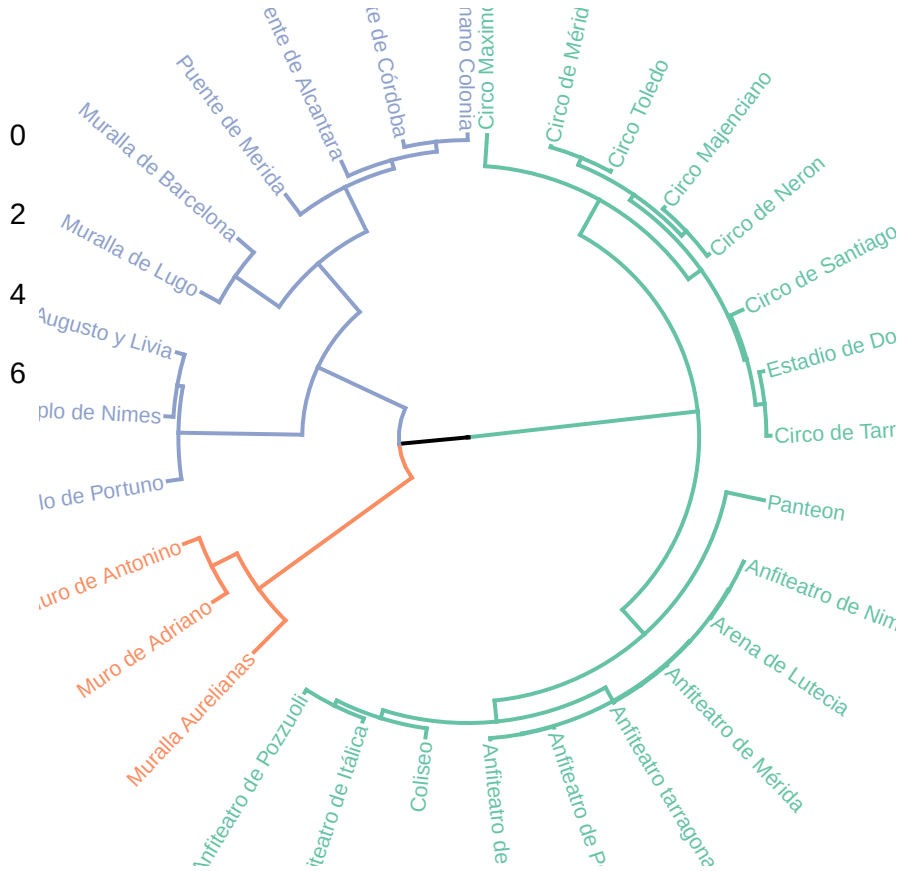
En última instancia, se agrupan los circos, anfiteatros, panteón, estadio y coliseo, que son edificios propios del núcleo de la urbe.

```
fviz_dend(dd_best,
          k = k_best, horiz = TRUE,
          cex = 0.55, labels_track_height = 5,
          k_colors = "Set2",
          color_labels_by_k = TRUE)
```



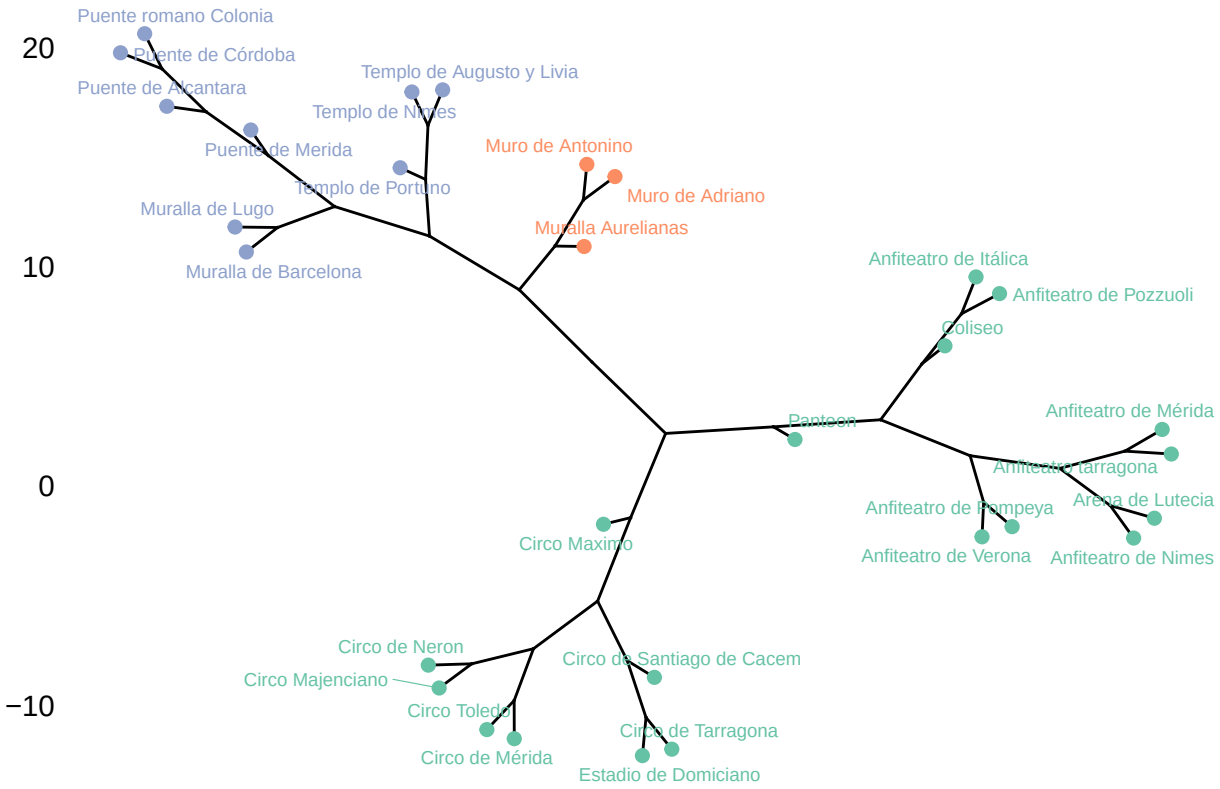
Line Color

```
fviz_dend(dd_best,
          k = k_best, horiz = TRUE, repel = TRUE,
          cex = 0.55, labels_track_height = 5,
          k_colors = "Set2", type = 'circular',
          color_labels_by_k = TRUE)
```



Circular

```
fviz_dend(dd_best,
          k = k_best, horiz = TRUE, repel = TRUE,
          cex = 0.55, labels_track_height = 5,
          k_colors = "Set2", type = 'phylogenic',
          color_labels_by_k = TRUE)
```



Phylogenic

Referencias

1. Análisis jerárquico de cluster: https://rquer.netlify.app/clustering/
2. Transformación logarítmica: https://rpubs.com/marvinlemos/log-transformation
3. Número óptimo de clusters: http://www.sthda.com/english/articles/29-cluster-validation-essentials/96-determiningthe-optimal-number-of-clusters-3-must-know-methods/