Clustering Analysis and Association Case Study - House Prices

Grace Bianchi

2025-12-11

Load Libraries and Read Data

```
#Loading libraries
library(factoextra)
```

```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

```
library(FactoMineR)
library(dendextend)
```

```
##
## ---------------------
## Welcome to dendextend version 1.19.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

```
library(hopkins)
library(cluster)
library(arules)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```r
library(arulesViz)
```

```r
#Reading the data
house_price_data <- read.csv("houseprice.csv")
house_price_data <- as.matrix(house_price_data)
head(house_price_data)
```

```
##      Living.Area Bathrooms Bedrooms Lot.Size Age Fireplace   Price
## [1,]       1.982       1.0        3     2.00 133         0 14.2212
## [2,]       1.676       1.5        3     0.38  14         1 13.4865
## [3,]       1.694       2.0        3     0.96  15         1 11.8007
## [4,]       1.800       1.0        2     0.48  49         1 13.8297
## [5,]       2.088       1.0        3     1.84  29         1 12.9470
## [6,]       1.456       2.0        3     0.98  10         0 20.6512
```

In this dataset there are 1047 records for 1047 houses. The variables are: Living.Area, Bathrooms, Bedrooms, Lot.Size, Age, Fireplace, Price.

Cluster Analysis

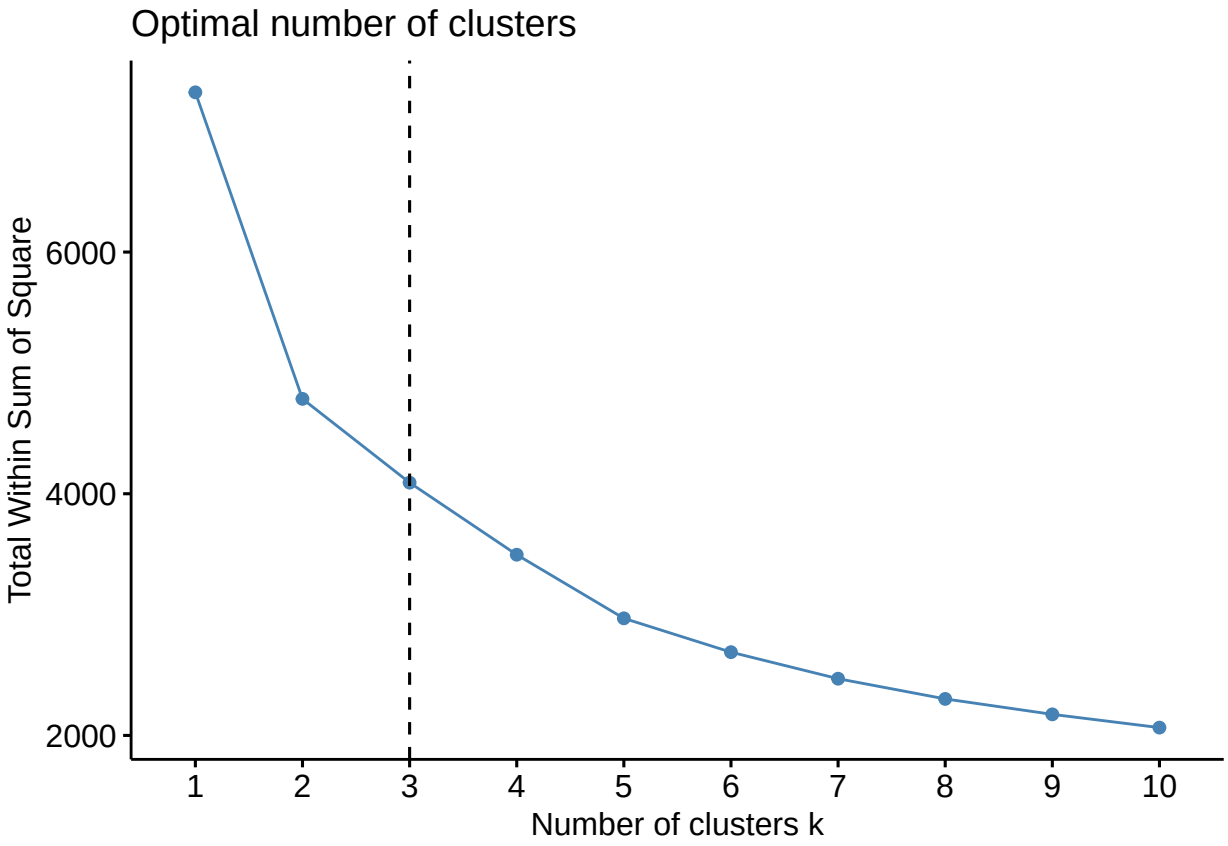
Standardize Data

```r
#Standardize data
house_price_data.st <- scale(house_price_data)
head(house_price_data.st)
```

```
##      Living.Area  Bathrooms   Bedrooms   Lot.Size         Age  Fireplace
## [1,]  0.27234275 -1.4310235 -0.2444218  1.8388729  3.00676706 -1.2067950
## [2,] -0.20469333 -0.6518859 -0.2444218 -0.2437137 -0.40288725  0.8278497
## [3,] -0.17663238  0.1272517 -0.2444218  0.5019037 -0.37423469  0.8278497
## [4,] -0.01138459 -1.4310235 -1.5772842 -0.1151590  0.59995225  0.8278497
## [5,]  0.43759054 -1.4310235 -0.2444218  1.6331853  0.02690111  0.8278497
## [6,] -0.54766044  0.1272517 -0.2444218  0.5276146 -0.51749747 -1.2067950
##           Price
## [1,] -0.3200240
## [2,] -0.4286246
## [3,] -0.6778133
## [4,] -0.3778941
## [5,] -0.5083715
## [6,]  0.6304345
```

K-Means Algorithm

```r
#k-means algorithm
fviz_nbclust(house_price_data.st, kmeans, method = "wss", nstart=20) + geom_vline(xintercept = 3, linety
```



The total within-cluster sum of squares drops sharply from k=1 to k=2 and again to k=3. After k=3, the curve begins to flatten and each additional cluster yields only a modest reduction in WSS. This "elbow" around k=3 suggests that three clusters captures most of the structure in the data. One could also argue for k=2 or k=4 as good solutions, but k=3 was chosen as the ideal number of clusters.

```
house_price_data.km <- kmeans(house_price_data.st, 3, nstart =20)
house_price_data.km
```

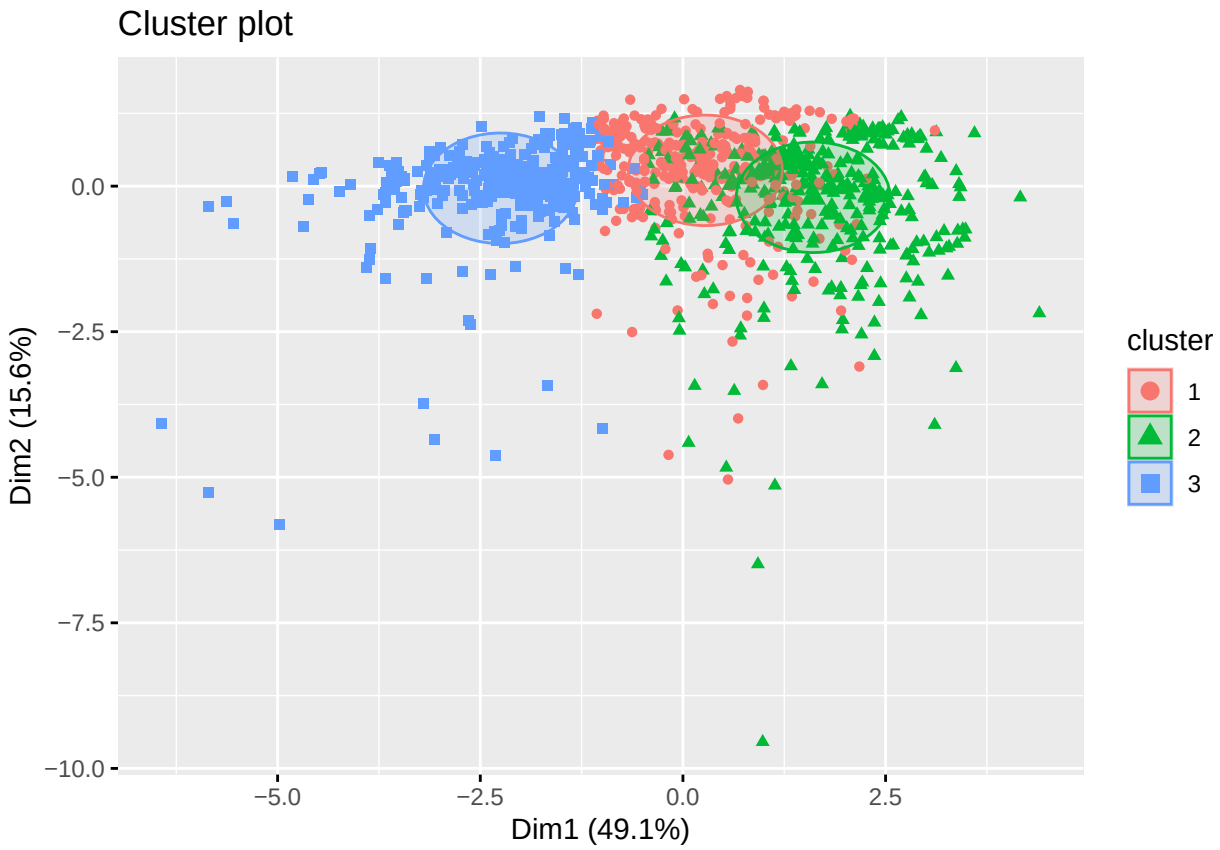
```
## K-means clustering with 3 clusters of sizes 325, 398, 324
##
## Cluster means:
##   Living.Area  Bathrooms   Bedrooms    Lot.Size         Age  Fireplace
## 1  -0.3175991 -0.2539264 -0.3797586 -0.06223402  0.05705242  0.8278497
## 2  -0.6840012 -0.6362249 -0.4252624 -0.09713551  0.35633353 -1.2067950
## 3   1.1588030  1.0362456  0.9033209  0.18174687 -0.49494685  0.6520162
##        Price
## 1 -0.2677053
## 2 -0.6614726
## 3  1.0810812
##
## Clustering vector:
##    [1] 2 1 1 1 1 2 2 2 2 2 3 1 2 2 1 1 2 1 3 3 3 1 2 1 1 1 3 2 3 1 1 2 1 2 1 1 1
##   [38] 1 2 3 2 3 2 2 1 3 1 2 1 3 1 2 2 1 1 2 2 2 2 1 2 2 1 2 2 1 2 2 1 2 2 1 3 2
##   [75] 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 1 2 1 1 2 2 1 2 1 1 1 2 1
##  [112] 2 2 1 3 2 1 3 2 1 3 3 2 3 1 1 1 1 2 1 2 2 3 2 3 1 3 2 2 2 2 2 3 2 1 2 3 3
##  [149] 2 2 2 1 2 2 1 2 3 2 2 2 1 3 3 2 2 2 3 1 1 1 1 2 3 1 2 2 2 2 1 2 2 1 1 3 1
##  [186] 3 1 1 1 2 3 1 3 3 1 1 2 3 1 1 3 1 1 1 1 1 3 3 3 3 1 3 3 3 3 3 3 3 1 2 3 3
##  [223] 3 3 3 3 2 3 1 1 2 1 3 3 3 3 1 3 3 3 3 3 3 3 1 2 2 1 1 1 3 3 1 3 3 2 3 3 3
```

```
##  [260] 2 2 3 1 3 3 3 2 3 1 2 2 1 2 2 2 2 2 2 1 1 2 1 2 2 1 2 2 1 3 1 1 1 1 2 2 1
##  [297] 2 2 2 2 3 3 1 1 1 1 2 3 2 2 2 2 1 3 2 2 1 2 1 1 2 1 1 1 1 2 3 1 3 3 1 2 3
##  [334] 3 2 1 2 2 2 1 1 1 1 2 1 1 3 1 1 3 3 1 1 1 3 2 1 2 2 3 1 1 2 3 1 2 2 3 1 2
##  [371] 3 2 2 3 2 1 1 1 3 2 2 3 2 2 2 3 3 2 2 2 3 3 3 2 2 3 3 2 2 2 2 3 3 2 2 3 2
##  [408] 3 3 1 1 2 3 1 2 1 3 1 3 1 1 2 2 3 2 1 3 1 2 3 1 2 2 1 2 2 2 2 2 1 1 2 2 2 2
##  [445] 2 3 1 1 1 1 1 2 3 2 1 3 2 2 2 1 1 1 3 2 2 2 3 2 2 1 1 1 2 3 3 2 1 3 3 1 3
##  [482] 3 3 3 3 3 2 3 2 1 3 3 3 3 3 1 3 3 3 1 1 3 1 2 2 2 2 2 3 3 3 3 2 2 1 1 2 1
##  [519] 1 2 2 1 2 2 2 1 2 1 1 2 1 3 2 1 2 3 2 2 3 2 2 1 3 1 3 2 1 1 2 2 2 2 1 1 2
##  [556] 3 2 3 2 3 1 2 2 1 3 3 2 1 2 1 3 3 1 1 2 3 3 1 3 3 2 1 3 3 2 2 3 2 3 1 3 3
##  [593] 2 3 1 3 2 1 1 3 3 3 3 3 3 3 1 3 1 1 1 3 3 1 3 1 3 3 3 3 3 3 3 1 1 3 2 1 1
##  [630] 3 2 3 3 1 1 2 3 1 2 3 1 2 3 2 3 1 3 2 2 1 1 1 2 2 1 3 3 3 1 3 1 3 3 3 1 2
##  [667] 2 1 2 3 1 1 1 2 2 2 2 2 2 2 1 1 1 2 2 1 2 1 1 2 1 1 1 3 1 2 2 3 2 1 1 1 2
##  [704] 3 2 3 2 3 2 2 2 2 2 2 1 1 2 1 2 2 1 3 3 1 3 2 1 3 3 2 1 1 2 2 1 2 2 2 2 1
##  [741] 2 2 1 3 3 2 2 3 3 2 2 1 2 2 2 2 1 2 2 2 2 2 2 2 2 1 3 1 2 3 2 2 3 3 2 2 1
##  [778] 2 2 3 1 1 1 3 1 3 1 2 3 2 3 2 3 1 2 3 1 1 2 3 3 2 3 3 3 1 2 2 3 1 3 2 1 3
##  [815] 1 2 3 1 3 3 1 2 2 3 3 3 3 2 3 2 2 3 3 3 2 2 3 1 1 3 2 2 1 2 2 2 2 1 2 1 2
##  [852] 1 1 3 1 2 2 3 2 3 1 2 3 3 3 1 3 1 3 3 3 1 3 2 3 2 3 3 2 1 3 1 1 2 2 1 3 2
##  [889] 2 2 2 2 3 1 2 2 2 2 2 2 3 2 3 2 3 2 1 3 1 3 3 1 2 1 3 3 3 3 3 3 2 1 1 3 1 1
##  [926] 3 2 2 1 1 3 3 3 1 2 1 3 1 2 1 2 3 3 1 3 2 1 1 3 2 2 3 3 3 1 1 3 3 3 1 3 3
##  [963] 2 2 2 1 2 2 2 1 1 3 1 2 2 1 3 1 2 2 2 1 1 1 1 2 3 2 3 3 3 1 3 3 2 3 3 2 3
## [1000] 1 2 3 2 2 2 3 3 3 2 3 3 1 3 3 3 3 2 1 3 2 2 1 1 1 2 2 2 3 2 2 1 3 1 3 2 1
## [1037] 1 2 1 2 1 3 1 3 1 3 3
##
## Within cluster sum of squares by cluster:
## [1] 1091.338 1788.945 1211.563
##  (between_SS / total_SS =  44.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
aggregate(house_price_data, by=list(cluster=house_price_data.km$cluster), mean)
```

```
##   cluster Living.Area Bathrooms Bedrooms  Lot.Size      Age Fireplace    Price
## 1       1    1.603575  1.755385 2.898462 0.5211692 30.05231 1.0000000 14.57514
## 2       2    1.368543  1.510050 2.864322 0.4940201 40.49749 0.0000000 11.91125
## 3       3    2.550630  2.583333 3.861111 0.7109568 10.78704 0.9135802 23.69990
```

```
fviz_cluster(house_price_data.km, data = house_price_data.st, repel = T, ellipse.type = "euclid", geom=
```



Cluster Interpretations:

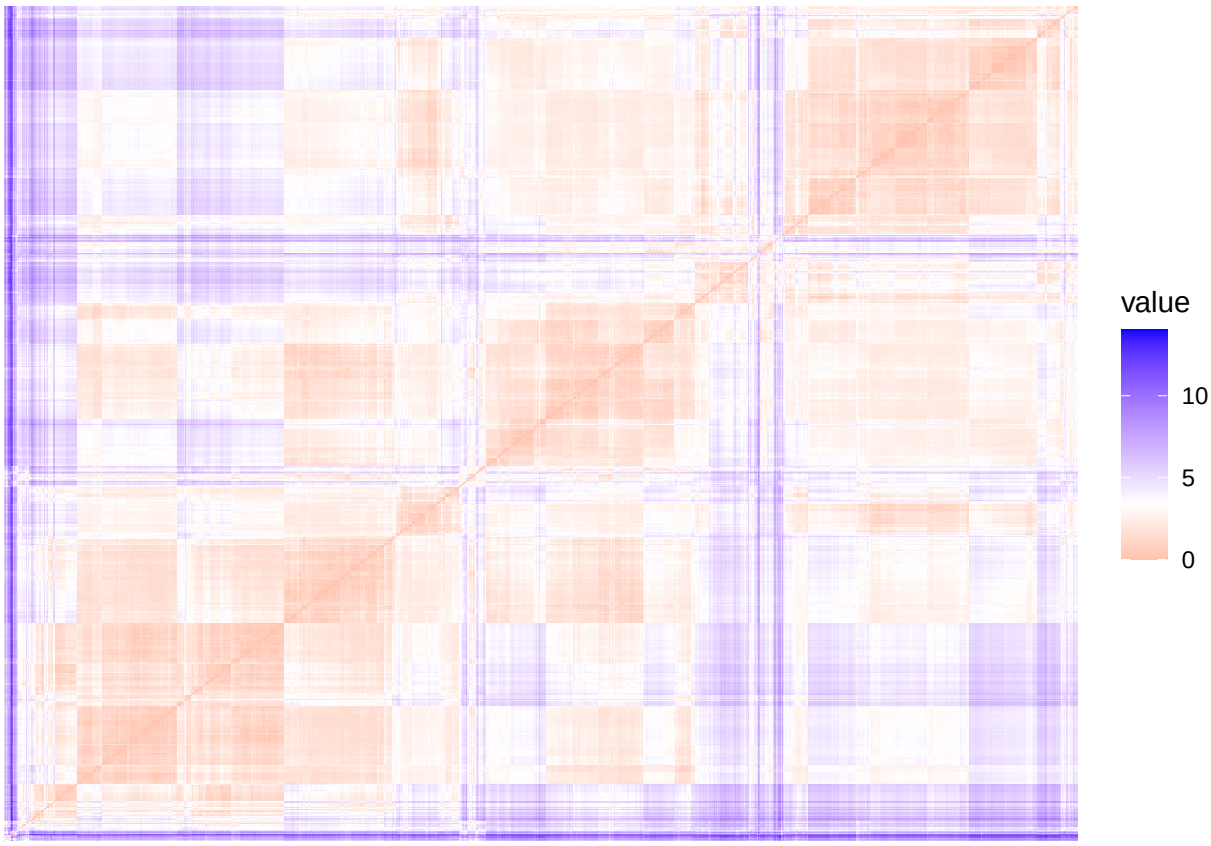
Cluster 1: Houses in Cluster 1 feature medium sized living rooms, an avergae number of bedrooms and bathrooms, as well as a moderate price. They are mid-range homes with balanced features and prices

Cluster 2: Houses in Cluster 2 are smaller and older. They feature the least amount of bathrooms, fewer fireplaces, and a lower price. They are budget/aging homes that have low-market value or are entry-level homes.

Cluster 3: Houses in Cluster 3 are the largest, featuring more bedrooms, bathrooms, and fireplaces. Additionally, they are newer and have the highest price. They are high-value and modern homes.

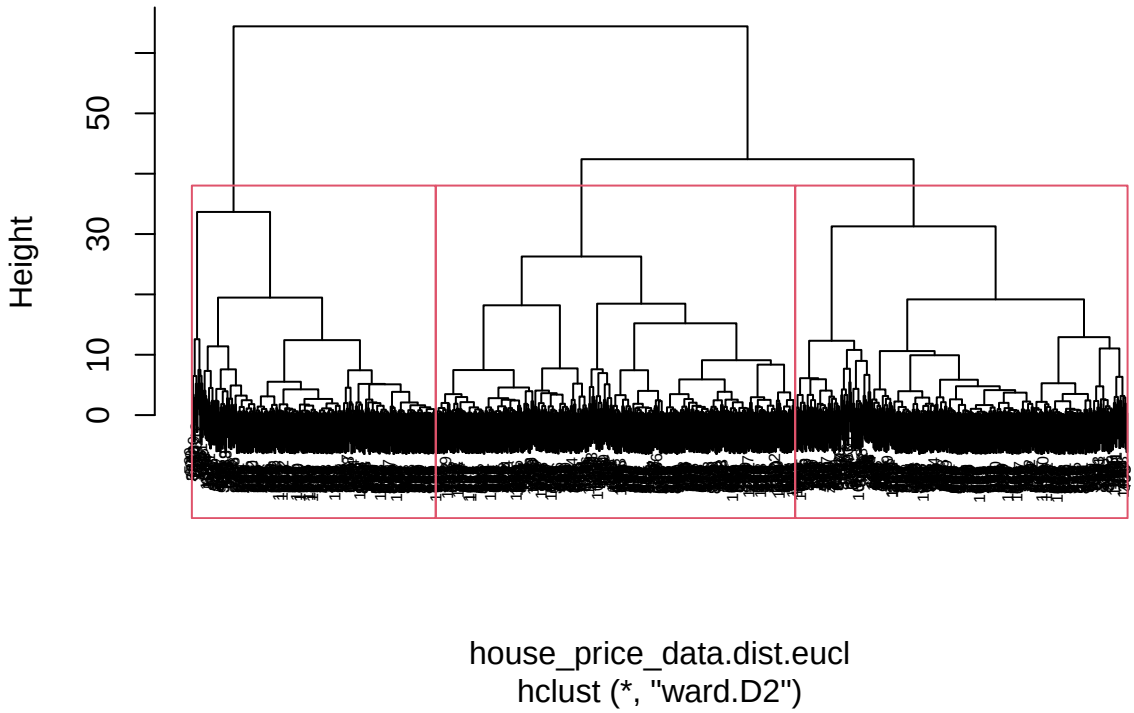
Hierarchical Methods

```
#distance matrix
house_price_data.dist.eucl <- dist(house_price_data.st, method = "euclidean")
fviz_dist(house_price_data.dist.eucl, show_labels = FALSE)
```




```
house_price_data.hc = hclust(d = house_price_data.dist.eucl, method = "ward.D2")

plot(house_price_data.hc,cex=0.5)
rect.hclust(house_price_data.hc,3)
```



Cluster Tree Verification

```
# ward.D2 linkage
res.coph = cophenetic(house_price_data.hc)
cor(house_price_data.dist.eucl,res.coph)
```

```
## [1] 0.5230662
```

```
# average linkage
hc_avg = hclust(d = house_price_data.dist.eucl, method = "average")
cor(house_price_data.dist.eucl,cophenetic(hc_avg))
```

```
## [1] 0.7604494
```

```
# complete linkage
hc_comp = hclust(d = house_price_data.dist.eucl, method = "complete")
cor(house_price_data.dist.eucl,cophenetic(hc_comp))
```

```
## [1] 0.634443
```

```
# single linkage
hc_sing = hclust(d = house_price_data.dist.eucl, method = "single")
cor(house_price_data.dist.eucl,cophenetic(hc_sing))
```

```
## [1] 0.6814562
```

Among the hierarchial linkage methods, average linkage achieved the highest cophenetic correlation (0.760), indicating the closest preservation of the original Euclidean distances. Complete linkage (0.634) and single linkage (0.681) performed moderately, while Ward.D2 showed a lower value (0.523).

However, despite the lower cophenetic correlation, Ward.D2 was retained as the primary hierarchical method.

Ward.D2 is well suited for standardized numerical data, produces compact and well-separated clusters, and closely matches the structure obtained from k-means (which also minimizes within-cluster variance). This method provided the most interpretable and stable partition of the data, consistent with common practice in clustering analysis.

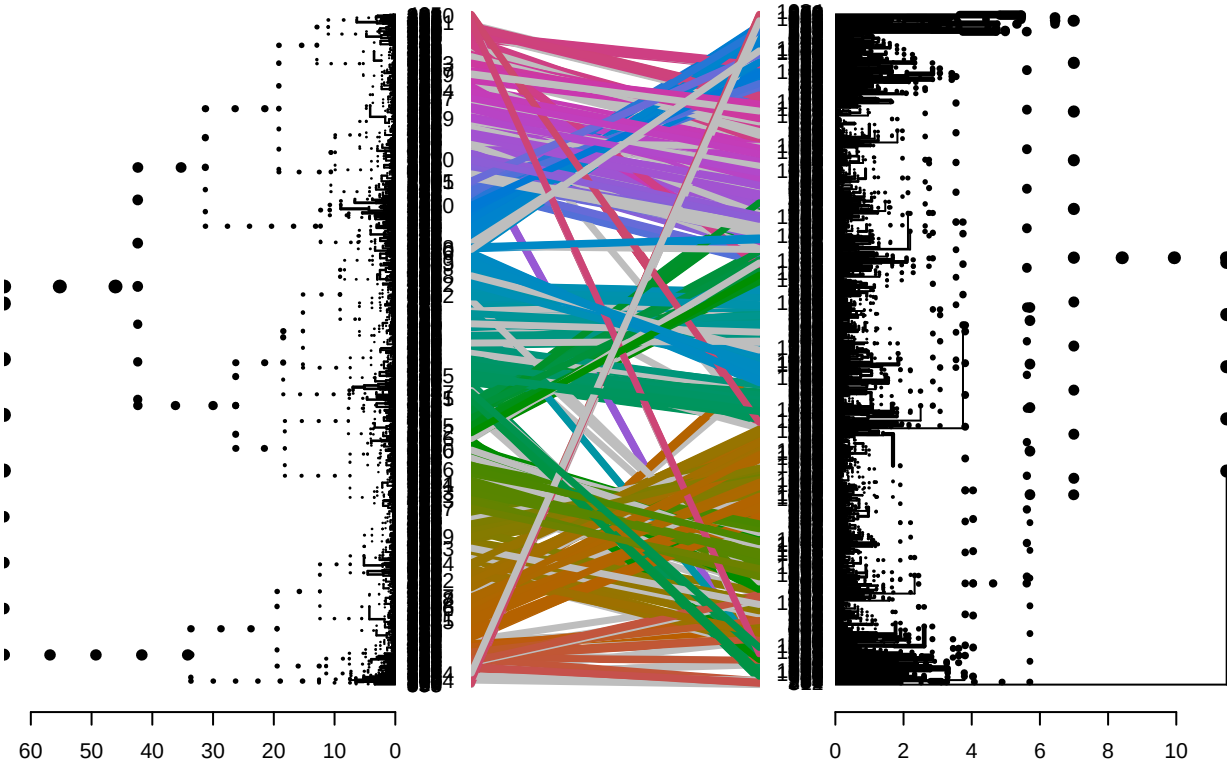
Compare Dendrogams

```
house_price_data.hc1 = hclust(d = house_price_data.dist.eucl, method = "ward.D2")
house_price_data.hc2 = hclust(d = house_price_data.dist.eucl, method = "average")

dend1 = as.dendrogram(house_price_data.hc1)
dend2 = as.dendrogram(house_price_data.hc2)

tanglegram(dend1,dend2)
```

```
## Loading required namespace: colorspace
```




```
entanglement_value = entanglement(dend1, dend2)
entanglement_value
```

```
## [1] 0.2098015
```

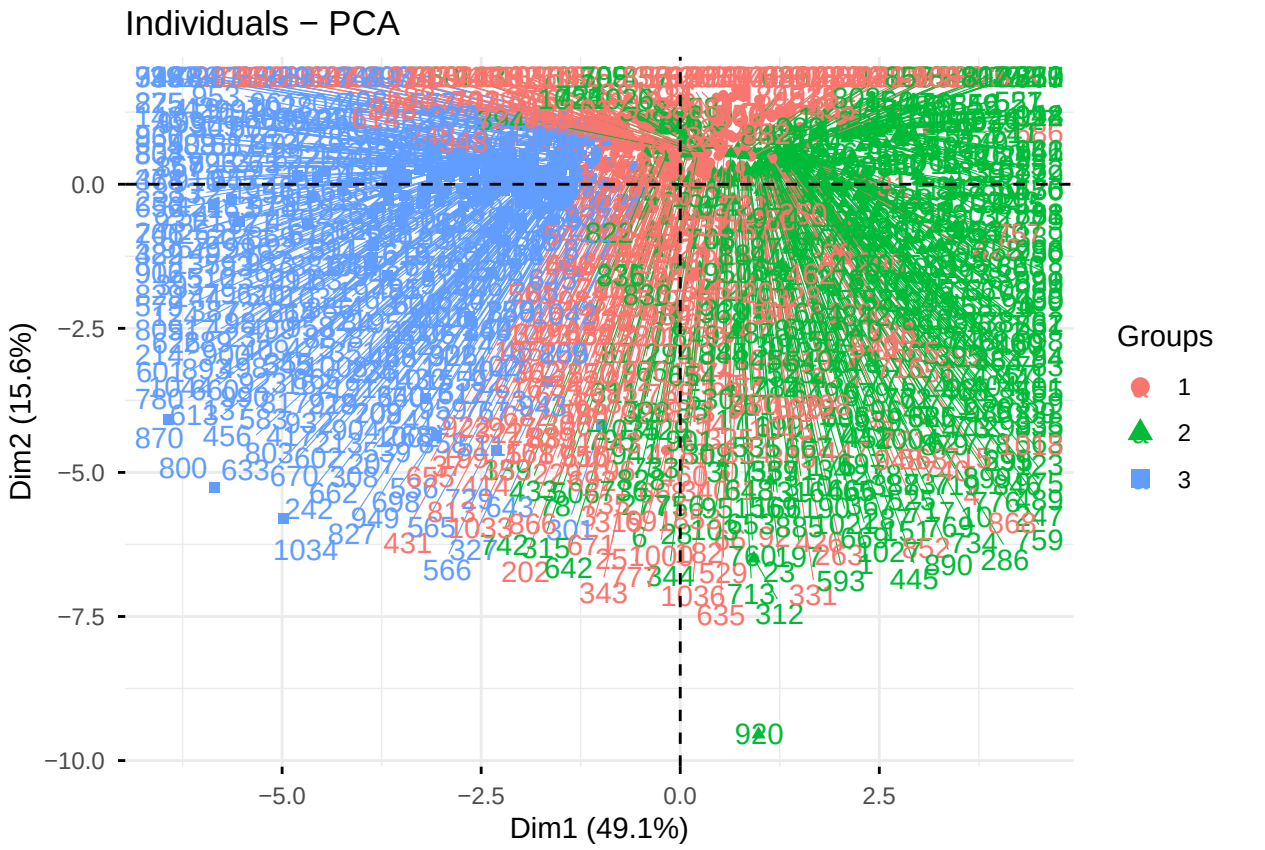
```
dend_list <- dendlist(dend1, dend2, as.dendrogram(hc_comp), as.dendrogram(hc_sing))
cor.dendlist(dend_list, method="cophenetic")
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 1.0000000 0.4525460 0.5887601 0.4676483
## [2,] 0.4525460 1.0000000 0.6454262 0.6441742
## [3,] 0.5887601 0.6454262 1.0000000 0.4615983
## [4,] 0.4676483 0.6441742 0.4615983 1.0000000
```

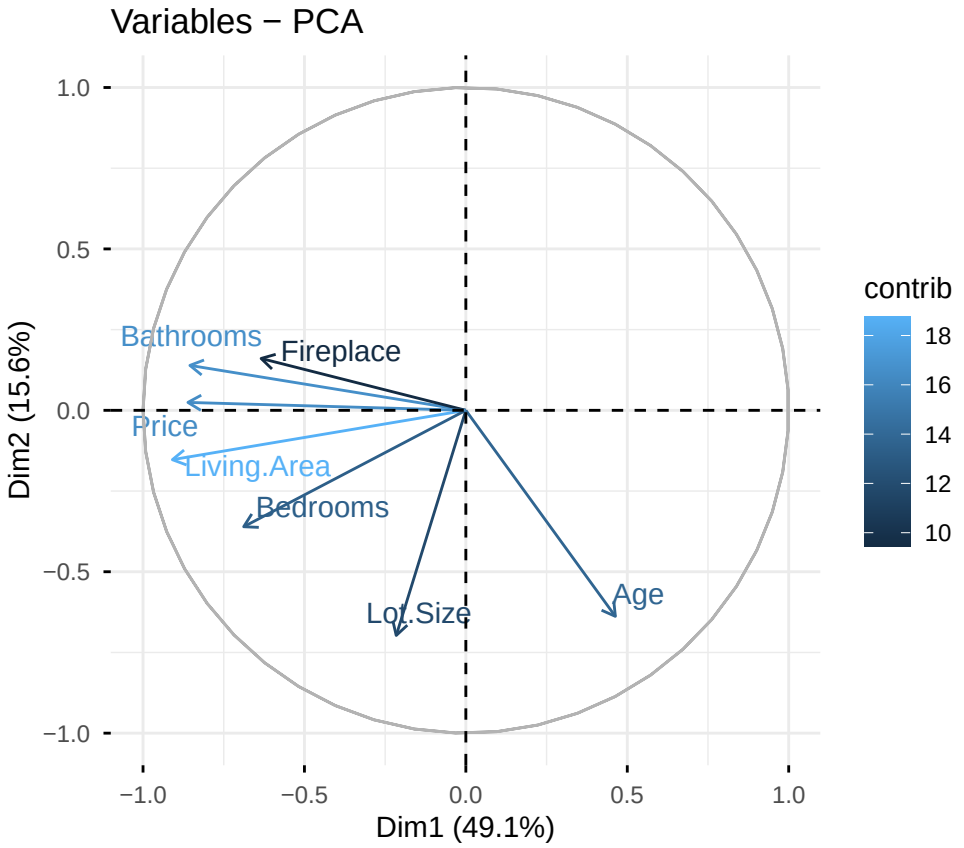
The tanglegram comparing Ward.D2 and average linkage shows an entanglement of 0.21, a relatively low value indicating that the two dendrograms share a broadly similar structure with limited crossing between matched branches. The cophenetic correlation matrix reinforces this: complete and average linkage agree most strongly (~0.65), while Ward.D2 shows moderate similarity with complete linkage (~0.59) and weaker similarity with average linkage (~0.45), and single linkage is the least consistent with the others due to chaining. Although Ward.D2 is not the method that best preserves pairwise distances, it produces more compact and interpreatble clusters and remains reasonably aligned with alternative linkage methods. Together, the low entanglement and moderate correlations suggest that the underlying cluster structure is stable, and they support using Ward.D2 as the preferred hierarchial approach.

PCA Visualization of Variables

```
fviz_pca_ind(prcomp(house_price_data.st),
             habillage = house_price_data.km$cluster,
             repel = TRUE)
```




```
fviz_pca_var(
  prcomp(house_price_data.st),
  col.var = "contrib",
  repel = TRUE
)
```



The PCA results provide a clear structural interpretation of the housing dataset and help explain the seperation of the three clusters. Dimension 1 (49.1% of total variance) represents a strong size-value axis: Living Area, Lot Size, Price, Bedrooms, and Bathrooms all load negatively on this component, meaning houses located toward the right of the plot are larger, more amenity-rich, and more expensive. Dimension 2 (15.6% of total variance) contrasts Age with features such as Fireplaces and Bathrooms, separating older homes (low on Dim2) from newer or updated homes (higher on Dim2). When projecting the k-means clusters onto the individual PCA space, the three groups align naturally with these gradients: Cluster 1 (blue) falls between the two extremes, indicating mid-sized, moderately priced homes; Cluster 2 (red) occupies the left side of Dim1, corresponding to smaller, older, and less expensive homes; and Cluster 3 (green) appears on the right side of Dim1 and slightly higher on Dim2, representing large, never, and high-priced homes;. Overall, the PCA plot confirm that the clusters identified by k-means correspond to meaningful and interpretable structural difference in house characteristics.

Hopkins Statistics

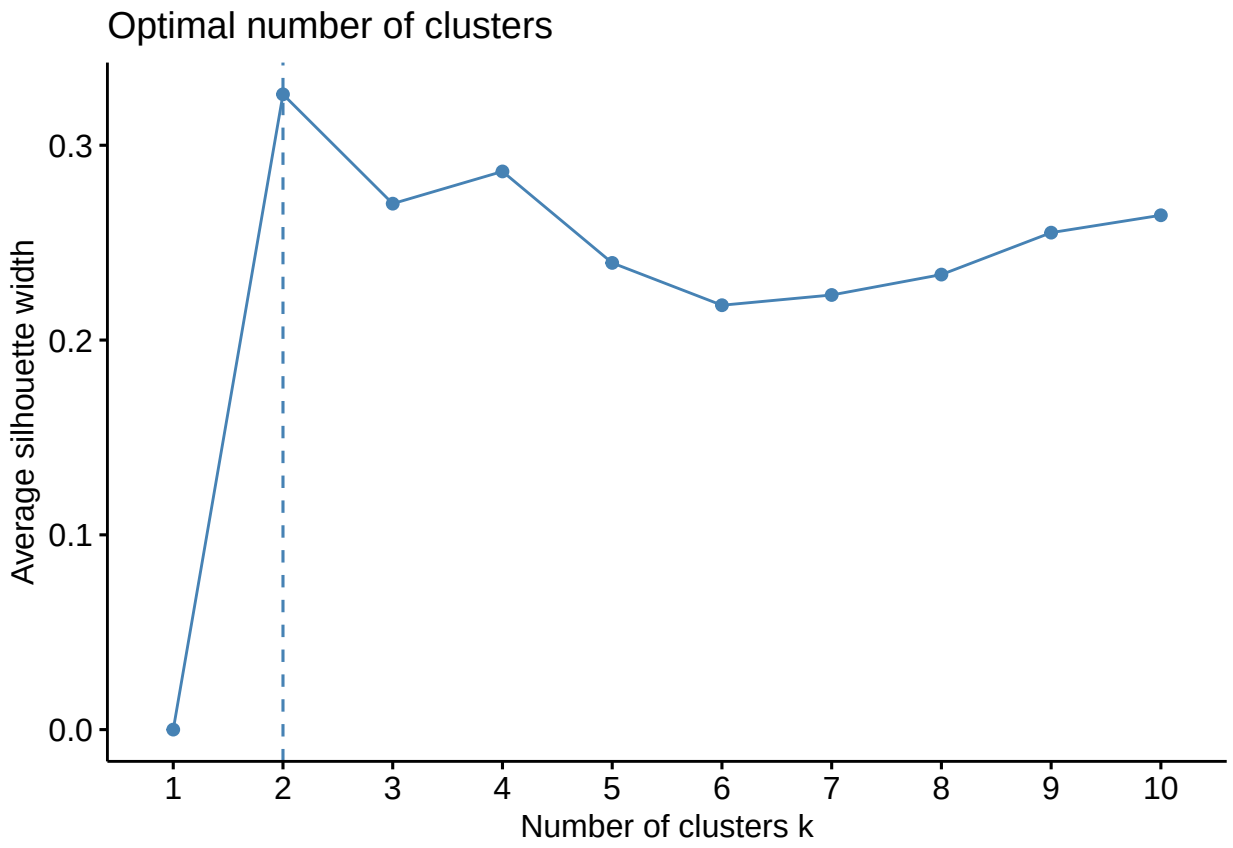
```
hopkins_stat = hopkins(house_price_data.st, m=100)
hopkins_stat
```

```
## [1] 0.9992359
```

The Hopkins statistic calculated using the hopkins library gives a value of 0.999 which is incredibly close to 1, indicating clustered data.

Alternative Algorithms

```
# Partitioning Around Medoids (PAM)
fviz_nbclust(house_price_data.st, pam, method = "silhouette",nstart = 200)
```

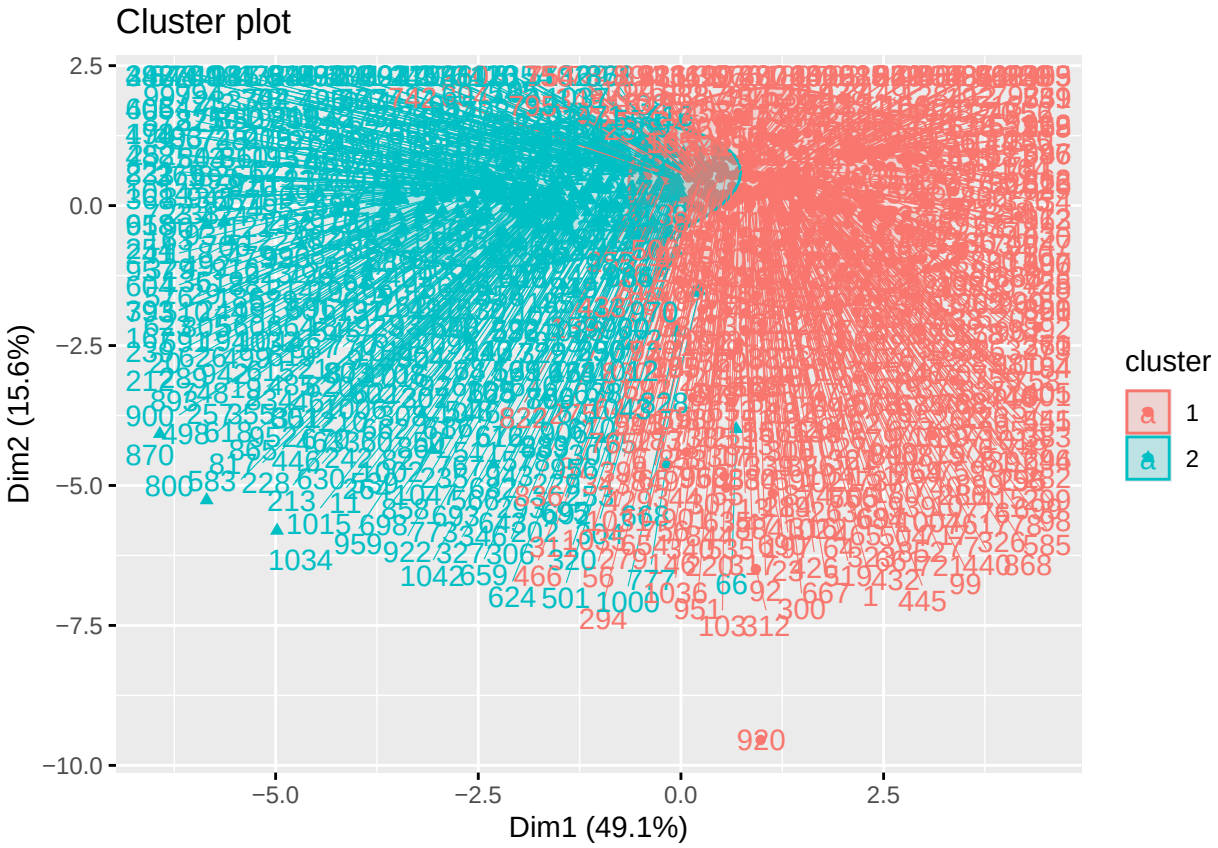



```
pam.res = pam(house_price_data.st,2)
head(pam.res)
```

```
## $medoids
##      Living.Area  Bathrooms   Bedrooms   Lot.Size         Age  Fireplace
## [1,]  -0.7129082 -0.6518859 -0.2444218 -0.1408700 -0.05905656 -1.2067950
## [2,]   0.7462609  0.9063893  1.0884407 -0.1023035 -0.48884492  0.8278497
##           Price
## [1,] -0.7570132
## [2,]  0.7521907
##
## $id.med
## [1]  61 990
##
## $clustering
##    [1] 1 1 1 1 1 1 1 1 1 2 2 1 1 2 2 1 2 2 2 2 1 1 1 1 1 2 1 2 1 1 1 1 1 1 1 2
##   [38] 2 1 2 1 2 1 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 2 1 1 2 1 1 1 2 1
##   [75] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 2
##  [112] 1 1 2 2 1 2 2 1 2 2 2 1 2 2 2 1 1 1 1 1 1 2 1 2 2 2 1 1 1 1 2 1 2 1 2 2
##  [149] 1 1 1 1 1 1 1 1 2 1 1 1 2 2 2 1 1 1 2 2 2 2 2 1 2 1 1 1 1 1 2 1 1 1 1 2 2
##  [186] 2 1 2 1 1 2 1 2 2 1 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2
##  [223] 2 2 2 2 1 2 2 2 1 2 2 2 2 2 1 2 2 2 2 2 2 2 2 1 1 2 1 2 2 2 2 2 1 1 2 2 2
##  [260] 1 1 2 1 2 2 2 1 2 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 1
##  [297] 1 1 1 1 2 2 1 2 1 2 1 2 1 1 1 1 1 2 1 1 1 1 1 2 1 2 1 1 1 1 2 1 2 2 1 1 2
##  [334] 2 1 1 1 1 1 1 1 2 1 1 1 2 2 1 2 2 2 1 2 2 2 1 1 1 1 2 2 2 1 2 1 1 1 2 2 1
##  [371] 2 1 1 2 1 2 2 2 2 1 1 2 1 1 1 2 2 1 1 1 2 2 2 1 1 2 2 1 1 1 1 2 2 1 1 2 1
##  [408] 2 2 2 1 1 2 2 1 2 2 1 2 1 2 1 1 2 1 1 2 1 1 2 2 1 1 2 1 1 1 1 1 1 1 1 1 1
```

```
##  [445] 1 2 1 1 1 1 1 1 2 1 1 2 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 2 1 2 2 1 1 2 2 1 2
##  [482] 2 2 2 2 2 2 1 2 1 1 2 2 2 2 2 2 2 2 2 1 2 2 1 1 1 1 1 1 2 2 2 2 1 1 1 1 1 2
##  [519] 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 2 1 1 2 1 1 2 2 2 2 1 1 2 1 1 1 1 2 2 1
##  [556] 2 1 2 1 2 2 1 1 2 2 2 1 1 1 2 2 2 1 1 1 2 2 2 2 2 1 1 2 2 1 1 2 1 2 2 2 2
##  [593] 1 2 1 2 1 1 1 2 2 2 2 2 2 2 2 2 1 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1
##  [630] 2 1 2 2 2 1 1 2 2 1 2 2 1 2 1 2 2 2 1 1 1 1 1 1 1 2 2 2 2 2 2 1 2 2 2 1 1
##  [667] 1 2 1 2 1 2 2 1 1 1 1 1 1 1 2 2 2 1 1 1 1 2 2 1 1 2 2 2 2 1 1 2 1 2 2 2 1
##  [704] 2 1 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 1 2 2 2 1 1 2 1 1 2 1 1 1 1 1
##  [741] 1 1 1 2 2 1 1 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 1 1 2 2 1 1 2
##  [778] 1 1 2 1 1 1 2 2 2 1 1 2 1 2 1 2 1 1 2 2 1 1 2 2 1 2 2 2 1 1 1 2 1 2 1 2 2
##  [815] 1 1 2 1 2 2 1 1 1 2 2 2 2 2 2 1 1 2 2 2 1 1 2 1 2 2 1 1 1 1 1 1 1 2 1 1 1
##  [852] 1 1 2 1 1 1 2 1 2 2 1 2 2 2 2 2 1 2 2 2 1 2 1 2 1 2 2 1 1 2 2 2 1 1 1 2 1
##  [889] 1 1 1 1 2 1 1 1 1 1 1 2 1 2 1 2 1 2 2 1 2 2 2 1 2 2 2 2 2 2 2 1 1 2 2 2 1
##  [926] 2 1 1 1 1 2 2 2 2 1 1 2 1 1 1 1 2 2 2 2 1 2 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2
##  [963] 1 1 1 1 1 1 1 1 2 2 2 1 1 1 1 2 1 1 1 1 1 2 1 1 1 2 1 2 2 2 2 2 2 1 2 2 1 2
## [1000] 2 1 2 1 1 1 2 2 2 1 2 2 2 2 2 2 2 1 1 2 1 1 1 2 2 1 1 1 2 1 1 1 2 2 2 1 1
## [1037] 2 1 1 1 2 2 2 2 1 2 2
##
## $objective
##    build     swap
## 2.075590 1.939948
##
## $isolation
##  1  2
## no no
## Levels: no L L*
##
## $clusinfo
##      size  max_diss  av_diss diameter separation
## [1,]  578 11.123704 2.071183 14.01809  0.2731546
## [2,]  469  9.648005 1.778213 11.67259  0.2731546
```

```
fviz_cluster(pam.res,
             ellipse.type = "t",
             repel = TRUE)
```



The silhouette analysis for PAM indicates that k=2 provides the best overall partition, with the highest average silhouette width. This suggests that the dataset has a strong underlying two-group structure, mainly separating larger, higher-value homes from smaller, less expensive homes, which aligns with the dominant gradient seen in the PCA results. The medoids from the two clusters reflect this contrast: one medoid represents smaller homes with smaller living area, fewer bathrooms, and lower prices, while the other corresponds to larger homes with higher values. Compared with k-means, which identified three more granular clusters, PAM produces a broader, more conservative segmentation, grouping the mid-market cluster with either the higher-end or lower-end cluster depending on proximity in the data space. The cluster visualization confirms this two-block pattern, supporting the overall conclusion that the primary structure in the data is driven by a size-value dimension, with k-means offering a more detailed refinement of this underlying split.

Association Rules

Read the Data

```
house_price_data <- read.csv("houseprice.csv")
```

Discretize Numeric Variables Using Quantiles

```
#Living Area
house_price_data$LivingAreaLevel = cut(house_price_data$Living.Area,
                        breaks = quantile(house_price_data$Living.Area, probs = c(0, 0.33, 0.66, 1), na.
                        labels = c("Small", "Average", "Large"),
                        include.lowest = TRUE)
```

```
#Bathrooms
house_price_data$BathroomsLevel = cut(house_price_data$Bathrooms,
                        breaks = c(0,1,2,3),
                        labels = c("One", "Two", "Three+"),
                        include.lowest = TRUE)

#Bedrooms
house_price_data$BedroomsLevel = cut(house_price_data$Bedrooms,
                        breaks = c(0,1,2,3),
                        labels = c("One", "Two", "Three+"),
                        include.lowest = TRUE)

#Lot Size
house_price_data$LotSizeLevel = cut(house_price_data$Lot.Size,
                        breaks = quantile(house_price_data$Lot.Size, probs = c(0, 0.33, 0.66, 1), na.rm =
                        labels = c("Small", "Average", "Large"),
                        include.lowest = TRUE)

#Age
house_price_data$AgeLevel = cut(house_price_data$Age,
                        breaks = quantile(house_price_data$Age, probs = c(0, 0.33, 0.66, 1), na.rm = TRU
                        labels = c("Recent", "Mid", "Old"),
                        include.lowest = TRUE)

#Fireplace
house_price_data$FireplaceLevel = cut(house_price_data$Fireplace,
                        breaks = c(-1, 0, 1, 2),
                        labels = c("None", "One", "Two+"),
                        include.lowest = TRUE)

#Price
house_price_data$PriceLevel = cut(house_price_data$Price,
                        breaks = quantile(house_price_data$Price, probs = c(0, 0.33, 0.66, 1), na.rm = T
                        labels = c("Less Expensive", "Average", "Expensive"),
                        include.lowest = TRUE)
```

Keep only final variables and convert all variables to factors

```
vars = c("LivingAreaLevel", "BathroomsLevel", "BedroomsLevel", "LotSizeLevel", "AgeLevel", "FireplaceLe

data_final = house_price_data[,vars]

data_final[] = lapply(data_final, as.factor)

head(data_final)
```

```
##   LivingAreaLevel BathroomsLevel BedroomsLevel LotSizeLevel AgeLevel
## 1           Large            One        Three+        Large      Old
## 2         Average            Two        Three+      Average      Mid
## 3         Average            Two        Three+        Large      Mid
## 4         Average            One           Two      Average      Old
## 5           Large            One        Three+        Large      Old
## 6           Small            Two        Three+        Large   Recent
```

```
##   FireplaceLevel     PriceLevel
## 1           None        Average
## 2            One        Average
## 3            One Less Expensive
## 4            One        Average
## 5            One        Average
## 6           None      Expensive
```

Transform to Transactions and Generate Rules

```
#Convert to transaction format
trans = as(data_final, "transactions")
summary(trans)
```

```
## transactions as itemMatrix in sparse format with
##  1047 rows (elements/itemsets/transactions) and
##  21 columns (items) and a density of 0.3166871
##
## most frequent items:
##    FireplaceLevel=One  BedroomsLevel=Three+    BathroomsLevel=Two
##                   621                   522                   433
##   FireplaceLevel=None BathroomsLevel=Three+               (Other)
##                   426                   396                  4565
##
## element (itemset/transaction) length distribution:
## sizes
##   5   6   7
##  20 326 701
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    5.00    6.00    7.00    6.65    7.00    7.00
##
## includes extended item information - examples:
##                    labels       variables  levels
## 1   LivingAreaLevel=Small LivingAreaLevel   Small
## 2 LivingAreaLevel=Average LivingAreaLevel Average
## 3   LivingAreaLevel=Large LivingAreaLevel   Large
##
## includes extended transaction information - examples:
##   transactionID
## 1             1
## 2             2
## 3             3
```

```
#Apply apriori algorithm
rules = apriori(trans, parameter = list(supp = 0.01, conf = 0.3))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##      10  rules TRUE
##
```

```
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 10
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[20 item(s), 1047 transaction(s)] done [0.00s].
## sorting and recoding items ... [19 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 done [0.00s].
## writing ... [6749 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
summary(rules)
```

```
## set of 6749 rules
##
## rule length distribution (lhs + rhs):sizes
##    1    2    3    4    5    6    7
##   17  177 1056 2555 2169  702   73
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   4.000   4.000   4.345   5.000   7.000
##
## summary of quality measures:
##     support          confidence        coverage            lift
##  Min.   :0.01051   Min.   :0.3000   Min.   :0.01051   Min.   :0.5058
##  1st Qu.:0.01528   1st Qu.:0.4286   1st Qu.:0.02674   1st Qu.:1.1748
##  Median :0.02388   Median :0.5893   Median :0.04298   Median :1.5188
##  Mean   :0.03783   Mean   :0.5948   Mean   :0.07005   Mean   :1.6530
##  3rd Qu.:0.04298   3rd Qu.:0.7391   3rd Qu.:0.08023   3rd Qu.:2.0288
##  Max.   :0.59312   Max.   :1.0000   Max.   :1.00000   Max.   :5.5989
##      count
##  Min.   : 11.00
##  1st Qu.: 16.00
##  Median : 25.00
##  Mean   : 39.61
##  3rd Qu.: 45.00
##  Max.   :621.00
##
## mining info:
##   data ntransactions support confidence
##  trans          1047    0.01        0.3
##                                                                call
##  apriori(data = trans, parameter = list(supp = 0.01, conf = 0.3))
```

The discretized dataset produces 1,047 transactions and 21 categorical items, with most homes contributing 6–7 attributes. This density is appropriate for association rule mining and ensures that meaningful co-occurrence patterns can emerge. Apriori generated 6,749 rules under the 1% support and 30% confidence thresholds, with a median lift of about 1.52, indicating that many rules reflect non-random relationships between structural features of homes.

```
#Sort by lift and inspect top rules
top_rules = sort(rules, by = "lift", decreasing = T)
inspect(head(top_rules, 10))
```

```
##      lhs                                rhs                     support confidence   coverage     lift co
## [1]  {BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid}                  => {BedroomsLevel=Two} 0.01528176  0.9411765 0.01623687 5.598930
## [2]  {BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid,
##       PriceLevel=Less Expensive}     => {BedroomsLevel=Two} 0.01528176  0.9411765 0.01623687 5.598930
## [3]  {LivingAreaLevel=Small,
##       BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid}                  => {BedroomsLevel=Two} 0.01528176  0.9411765 0.01623687 5.598930
## [4]  {LivingAreaLevel=Small,
##       BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid,
##       PriceLevel=Less Expensive}     => {BedroomsLevel=Two} 0.01528176  0.9411765 0.01623687 5.598930
## [5]  {BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid,
##       FireplaceLevel=None}           => {BedroomsLevel=Two} 0.01050621  0.9166667 0.01146132 5.453125
## [6]  {BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid,
##       FireplaceLevel=None,
##       PriceLevel=Less Expensive}     => {BedroomsLevel=Two} 0.01050621  0.9166667 0.01146132 5.453125
## [7]  {LivingAreaLevel=Small,
##       BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid,
##       FireplaceLevel=None}           => {BedroomsLevel=Two} 0.01050621  0.9166667 0.01146132 5.453125
## [8]  {LivingAreaLevel=Small,
##       BathroomsLevel=One,
##       LotSizeLevel=Small,
##       AgeLevel=Mid,
##       FireplaceLevel=None,
##       PriceLevel=Less Expensive}     => {BedroomsLevel=Two} 0.01050621  0.9166667 0.01146132 5.453125
## [9]  {LivingAreaLevel=Small,
##       AgeLevel=Mid,
##       FireplaceLevel=One,
##       PriceLevel=Less Expensive}     => {BedroomsLevel=Two} 0.02005731  0.8750000 0.02292264 5.205256
## [10] {LivingAreaLevel=Small,
##       LotSizeLevel=Small,
##       AgeLevel=Mid,
##       FireplaceLevel=One,
##       PriceLevel=Less Expensive}     => {BedroomsLevel=Two} 0.01814709  0.8636364 0.02101242 5.137655
```

The highest-lift rules predominantly predict two-bedroom homes, revealing a strong structural pattern in the dataset. Homes that are mid-aged, have one bathroom, and sit on small lots are associated with a 94% likelihood of having exactly two bedrooms (lift $\sim$ 5.6). This indicates that smaller, older homes follow a very consistent layout pattern, making the number of bedrooms one of the most predictable attributes in the dataset.

```r
#Price rules
price_rules = subset(rules, rhs%in% "PriceLevel=Expensive")
inspect(head(sort(price_rules, by = "lift"), 5))
```

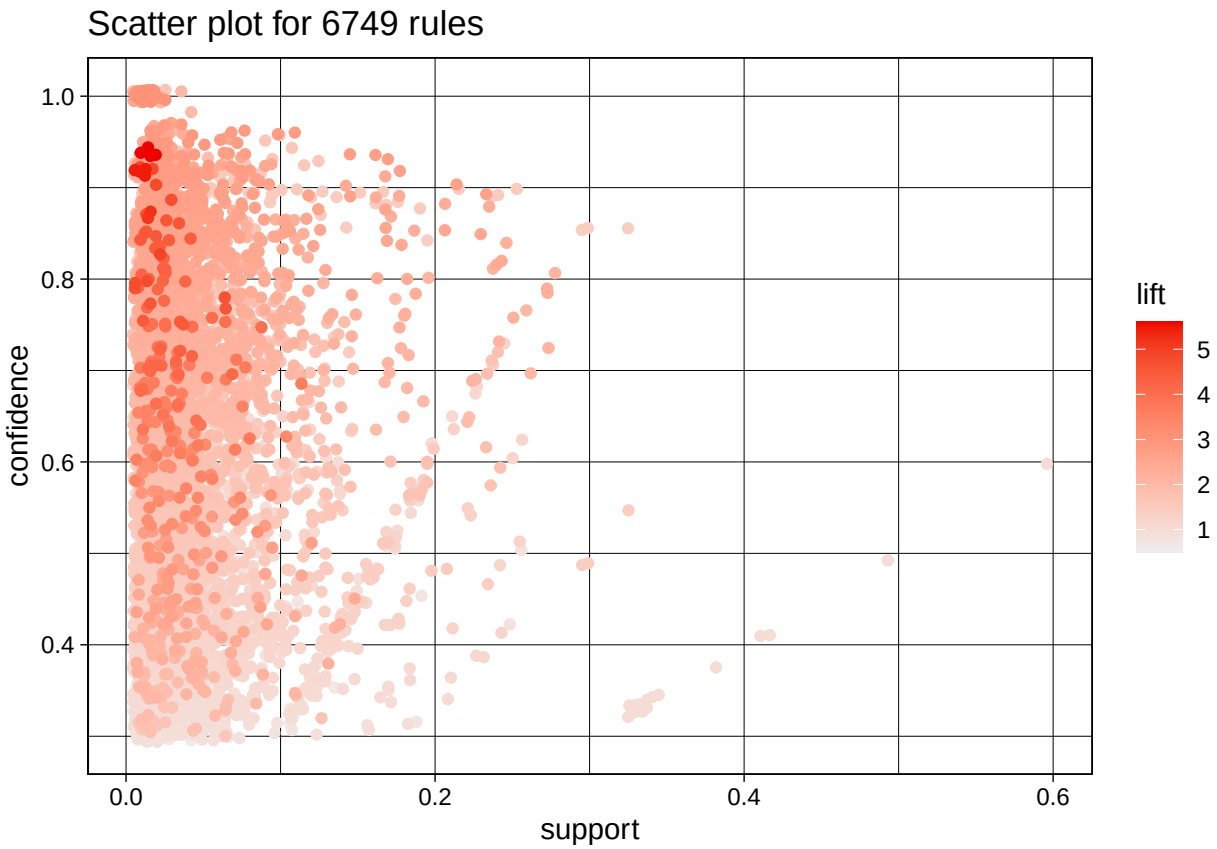
```
##     lhs                          rhs                        support confidence   coverage     lift cou
## [1] {LivingAreaLevel=Large,
##      BathroomsLevel=Three+,
##      BedroomsLevel=Three+,
##      FireplaceLevel=None}    => {PriceLevel=Expensive} 0.01146132          1 0.01146132 2.941011
## [2] {LivingAreaLevel=Large,
##      BathroomsLevel=Three+,
##      BedroomsLevel=Three+,
##      LotSizeLevel=Large,
##      AgeLevel=Mid}           => {PriceLevel=Expensive} 0.01337154          1 0.01337154 2.941011
## [3] {LivingAreaLevel=Large,
##      BedroomsLevel=Three+,
##      LotSizeLevel=Large,
##      AgeLevel=Mid,
##      FireplaceLevel=One}     => {PriceLevel=Expensive} 0.01050621          1 0.01050621 2.941011
## [4] {LivingAreaLevel=Large,
##      BathroomsLevel=Three+,
##      BedroomsLevel=Three+,
##      LotSizeLevel=Average,
##      AgeLevel=Recent}        => {PriceLevel=Expensive} 0.01528176          1 0.01528176 2.941011
## [5] {LivingAreaLevel=Large,
##      BathroomsLevel=Three+,
##      BedroomsLevel=Three+,
##      LotSizeLevel=Average,
##      AgeLevel=Recent,
##      FireplaceLevel=One}     => {PriceLevel=Expensive} 0.01337154          1 0.01337154 2.941011
```

Rules predicting expensive homes consistently involve large living areas, three or more bathrooms, multiple bedrooms, and large lots. The highest-confidence rules (confidence = 1.00, lift ~ 2.94) reveal that houses with this combination of size and amenities are always in the expensive tier. This reinforces the idea that house price is strongly driven by total size and number of features.

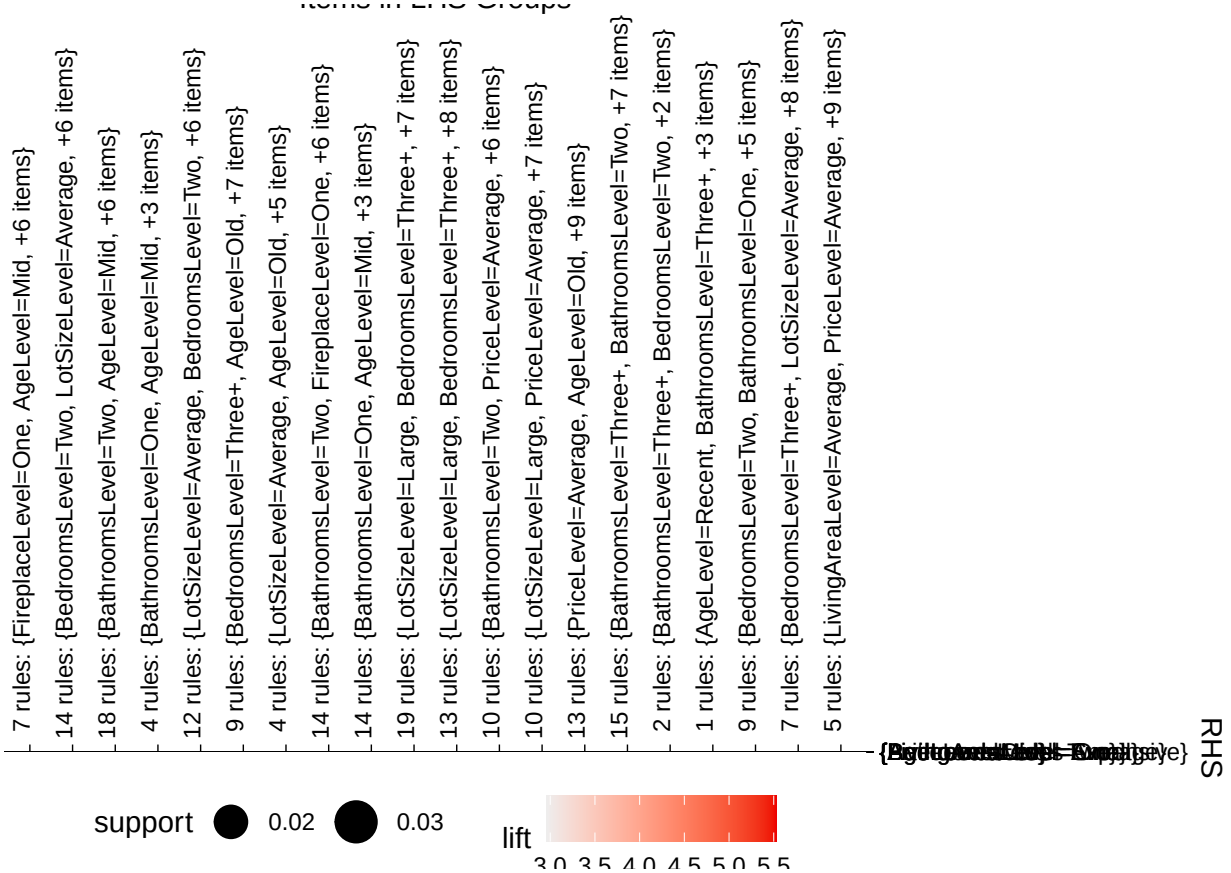
Visualizations

```r
#Scatterplot of rules
plot(rules, method = "scatterplot", measure = c("support", "confidence"), shading = "lift")
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```




```
#Grouped matrix of top 200 rules
top200 <- head(sort(rules, by = "lift"), 200)
plot(top200, method = "grouped")
```



Interpretation: The rule scatterplot shows that while most rules have low support, many exhibit high confidence and lift, indicating reliable patterns even among less frequent combinations. The grouped matrix highlights two major structural archetypes: rules involving small, older homes with fewer bathrooms cluster around "less expensive" outcomes, whereas rules involving large, amenity-rich homes cluster around the "expensive" outcome. These patterns align directly with the three clusters identified earlier.

Conclusion and Limitations

This project used multiple clustering techniques and association rule mining to uncover structure within the housing dataset and identify the features most strongly associated with home pricing. Across k-means, hierarchical clustering, and PAM, a consistent segmentation emerged: properties naturally group into clusters representing smaller and older homes, mid-sized mid-aged homes, and larger, newer, more expensive homes. PCA reinforced this structure by showing that the first principal component—driven primarily by lot size, age, and total living area—cleanly separates smaller/older properties from larger/newer ones. Association rules added further insight by highlighting stable co-occurrence patterns: combinations such as small living area, small lot size, and one bathroom frequently predict homes with only two bedrooms, while large lots, three or more bathrooms, and multiple bedrooms almost always imply an expensive price category. Together, these methods provide a coherent and interpretable view of the underlying patterns shaping variation in the housing market.

Despite these strengths, several limitations affect the generalizability of the findings. Clustering outcomes depend on the chosen distance metric, the decision to standardize the data, and the selected number of clusters; alternative preprocessing choices could shift the cluster boundaries, particularly within the mid-market region where homes are more heterogeneous. K-means also imposes spherical cluster assumptions that may not perfectly reflect real housing variability. On the association-rule side, continuous variables were discretized into categorical bins, meaning rule patterns depend partly on the quantile thresholds used. A minimum

support of 1% ensures reliability but may exclude rarer, potentially meaningful relationships. Future analyses could incorporate additional variables such as neighborhood characteristics or renovation quality, experiment with model-based or density-based clustering, or apply supervised learning methods to directly predict price tiers. These extensions would deepen the interpretability of the clusters and strengthen the predictive power of the association insights.