DEPARTMENT OF
DATA SCIENCE

DS311
Exploratory Data Analysis

Multivariate Methods and Data Mining in R

Overview

This learning portfolio covers key multivariate analysis techniques in R: **Hierarchical Cluster Analysis**, **K-Means Clustering** We will explain each concept with real-world relevance, demonstrate step-by-step R code (using base R and the tidyverse), and provide a hands-on lab with a case study. By the end, you will understand how to group similar observations (clustering), reduce data dimensionality (PCA).

Prerequisites

- **Basic R skills:** You should be comfortable with R syntax, data frames, and plotting. Familiarity with RStudio and R Markdown is helpful.
- **Packages:** We will use functions from base R and the **tidyverse** (for data manipulation and ggplot2 visualization). Ensure you have installed and loaded the `tidyverse` package. We also use `stats` (built-in for clustering, PCA) and `MASS` (for LDA). For the lab case study, we'll use the `palmerpenguins` dataset – install this package (`install.packages("palmerpenguins")`) or use the provided data URL.

Throughout this portfolio, we'll intermix **conceptual explanations** with **illustrative R code**. Key results or outputs are discussed (with occasional check-in questions to reinforce understanding). All code is provided in an RMarkdown-style format so you can copy, paste, and run it in R or RStudio. Let's get started!

1. Hierarchical Cluster Analysis

Concept: Hierarchical clustering is an unsupervised learning method that seeks to build a hierarchy of clusters. In **agglomerative hierarchical clustering** (the most common approach), we start with each observation as its own cluster and iteratively merge the two closest clusters until one cluster remains. This produces a tree-like diagram called a *dendrogram*, which shows how clusters are merged at each step. Unlike K-means, hierarchical clustering does *not* require specifying the number of clusters in advance. You can decide how many clusters to use by "cutting" the dendrogram at a chosen height (distance) level. Hierarchical clustering is often used in gene expression analysis, customer segmentation, and anywhere you want to see a hierarchy or taxonomy of how observations group together.

Distance and linkage: Hierarchical methods require a definition of distance (or dissimilarity) between observations. By default, Euclidean distance is used in R (you can also use Manhattan, correlation distance, etc.). Additionally, we must choose a *linkage criterion* that determines the distance between clusters. Common linkage methods include: **complete linkage** (distance between farthest members of clusters, tends to create compact clusters), **single linkage** (distance between closest members, can result in long "chains"), **average linkage** (uses average pairwise distance), and **Ward's method** (minimizes the increase in total within-cluster variance when merging clusters). Ward's method is popular for its tendency to create balanced-sized clusters.

Data preprocessing: It's important to preprocess data before clustering. Typically, **standardize** each variable (mean 0, SD 1) so that all variables contribute equally. Otherwise, features with larger scales could dominate the distance calculation. Also ensure there are no missing values (remove or impute them).

Example – Clustering USArrests dataset: We will demonstrate hierarchical clustering using R's built-in `USArrests` dataset, which contains the rates of violent crimes (Murder, Assault, Rape) and UrbanPop (percent urban population) for each U.S. state in 1973. Our goal is to discover if states form groups based on these crime rates.

Hierarchical Clustering in R (USArrests Example)

```r
# Hierarchical clustering on USArrests data

# 1. Load and inspect the data
data("USArrests")
str(USArrests)
```

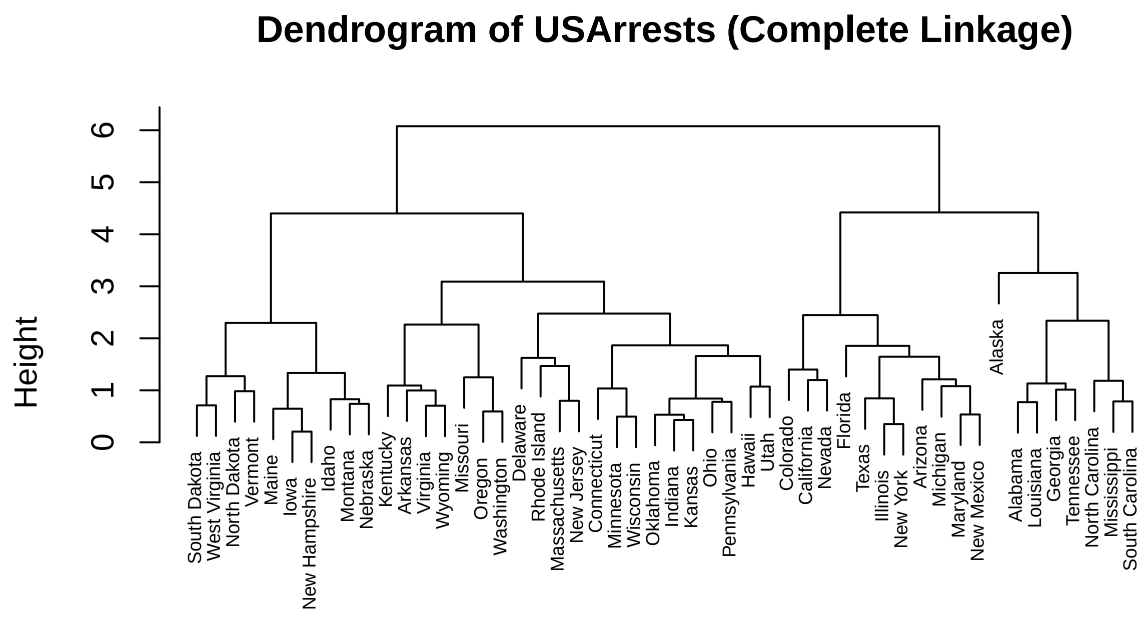
```
## 'data.frame':    50 obs. of  4 variables:
##  $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
##  $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
##  $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
##  $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
```

```r
# The data has 50 observations (states) and 4 variables.
# 2. Handle missing values (if any) and standardize variables
anyNA(USArrests)
```

```
## [1] FALSE
```

```r
# Standardize the variables to mean=0, sd=1:
USArrests_scaled <- scale(USArrests)
# scale() returns a matrix; convert to data frame for convenience
USArrests_scaled <- as.data.frame(USArrests_scaled)

# 3. Compute the distance matrix (Euclidean distance by default)
dist_matrix <- dist(USArrests_scaled, method = "euclidean")
```

```r
# 4. Perform hierarchical clustering using Complete linkage
hc_complete <- hclust(dist_matrix, method = "complete")

# 5. Plot the dendrogram
plot(hc_complete, main="Dendrogram of USArrests (Complete Linkage)",
     xlab="", sub="", cex=0.6)
```



```r
data("USArrests")
```

Step 1: What is Hierarchical Clustering?

Imagine this: You walk into a bookstore with 50 mystery books (in our case: **50 US states**). You want to group them based on similarity — but you're not told how many groups there should be.

Instead of guessing upfront, you **start by putting each book in its own pile**. Then, you look for the **two most similar piles** and merge them. You keep repeating this process: find the two closest groups and merge. Eventually, you end up with **one big pile**.

This whole process is what we call **agglomerative hierarchical clustering** — and it builds a **tree** (called a *dendrogram*) showing **which states got grouped together, when, and how strongly related they are**.

Step 2: How Are "Similar" States Found?

To know how similar states are, we measure **distance** between them based on their **crime statistics**:

- Murder rate
- Assault rate
- Urban population %
- Rape rate

But hold on! These are on different scales (e.g., Murder is in numbers like 13.2, UrbanPop is percent like 58). We **standardize** them first so that:

- Each feature is on the same footing
- All values become mean = 0, standard deviation = 1

This makes sure that no single feature dominates the similarity measure.

Step 3: What Does "Complete Linkage" Mean?

Let's say we want to merge two groups of states. "Complete linkage" asks:

> What's the **worst-case distance** between any state in Group A and any state in Group B?

So, we only merge groups **when even their most different members aren't too far apart**. This makes for **tight, compact clusters**. (If we had used "single linkage", we'd allow long, stretched-out chains.)

Step 4: What Does the Dendrogram Show?

Take a look at the plot you generated.

- **X-axis**: US States (one leaf per state)
- **Y-axis**: "Height" — this is the distance (or dissimilarity) at which clusters are merged

So if **two states are joined near the bottom**, it means they are very similar. If they merge only way up high, they're quite different.

Step 5: How to Read the Dendrogram

Let's go over what we see in the image:

- **Bottom left**, states like **South Dakota**, **West Virginia**, **North Dakota**, and **Vermont** are **clustered very early** — meaning their crime patterns are very similar (low crime states).
- **Far right**, states like **Alabama**, **Louisiana**, **Mississippi**, and **South Carolina** merge late — meaning they are more distinct (probably higher crime).
- There's a **big merge around height = 5.5** where two major super-clusters finally merge. This implies a significant division in the dataset — likely **low-crime vs high-crime states**.

Step 6: How Many Clusters Do We Have?

This is **your decision**! Hierarchical clustering lets *you* decide how many clusters by "cutting" the dendrogram at a certain height.

Let's actually cut into 4 clusters and examine the composition:

```r
library(tidyverse)
```

```
## Warning: package 'stringr' was built under R version 4.4.3

## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```r
# 6. Cut the dendrogram into 4 clusters
cluster_assignments <- cutree(hc_complete, k = 4)
table(cluster_assignments)
```

```
## cluster_assignments
##  1  2  3  4
##  8 11 21 10
```

```r
# 7. Investigate which states are in each cluster (printing first few from each)
split(rownames(USArrests), cluster_assignments) %>%
  lapply(head, 5)
```

```
## $`1`
## [1] "Alabama"     "Alaska"      "Georgia"     "Louisiana"   "Mississippi"
##
## $`2`
## [1] "Arizona"    "California" "Colorado"   "Florida"    "Illinois"
##
## $`3`
## [1] "Arkansas"    "Connecticut" "Delaware"    "Hawaii"      "Indiana"
##
## $`4`
## [1] "Idaho"    "Iowa"     "Maine"    "Montana"  "Nebraska"
```

(*Note:* We used a `%>%` pipeline from **tidyverse** for convenience in printing cluster members.) We won't list all states here, but in general we see that certain clusters correspond to states with higher crime rates versus lower crime rates.

From the output:

```
cluster_assignments
 1  2  3  4
 8 11 21 10
```

- **Cluster 1** = 8 states
- **Cluster 2** = 11 states

- **Cluster 3** = 21 states
- **Cluster 4** = 10 states

Now let's understand the characteristics of each cluster by **looking at the states in them**, and then linking them back to what we know about their crime statistics.

```
[1] "Alabama" "Alaska" "Georgia" "Louisiana" "Mississippi"
```

Cluster 1: **Interpretation:** These are **predominantly Southern states**, plus **Alaska**, and are known historically (especially in 1970s data like USArrests) for:

- **High murder rates**
- **High assault and rape rates**
- Generally **higher violent crime rates**

Conclusion: This cluster groups together the **most violent or high-crime states**. They stand out from the rest, which is why they were merged late in the dendrogram (i.e., at higher "height").

```
[1] "Arizona" "California" "Colorado" "Florida" "Illinois"
```

Cluster 2: **Interpretation:** These are **large, urbanized, and relatively high-crime states** — but maybe **not as extreme as Cluster 1**.

- Still high in urban population
- Likely moderate to high in assault or rape
- Slightly more "mixed" crime profile

Conclusion: This is a **second-tier high-crime cluster**, perhaps more urban and diverse in crime type but not as severe as Cluster 1.

```
[1] "Arkansas" "Connecticut" "Delaware" "Hawaii" "Indiana" ...
```

Cluster 3: (This one has **21 states** total – the largest cluster.)

Interpretation: This is the **"middle-of-the-road"** cluster. States here likely:

- Have **moderate crime rates** across all categories
- Do not particularly stand out for being extremely high or low
- Could be more urbanized but less violent

Conclusion: This cluster is the **"average" or typical states** — not too violent, not too safe.

```
[1] "Idaho" "Iowa" "Maine" "Montana" "Nebraska" ...
```

Cluster 4: **Interpretation:** These states are:

- **More rural**
- **Low in population density**
- **Low in all types of violent crime**

Conclusion: This cluster represents the **lowest-crime or safest states** in the dataset. They were likely merged early in the dendrogram because they are **very similar and very safe**.

Final Takeaway:

Cluster	General Description	Example States
1	**Very high crime**, especially violent	Alabama, Alaska, Georgia, Louisiana
2	**Urban + moderate-high crime**	California, Florida, Illinois
3	**Mid-level crime, average states**	Connecticut, Indiana, Missouri
4	**Low crime, rural states**	Iowa, Maine, Montana

Bonus Insight:

Step-by-Step Code to Map Clusters on a US Map

We'll use:

- `USArrests` (original data)
- `cluster_assignments` from your hierarchical clustering
- `maps`, `map_data`, `ggplot2`, and `dplyr` for visualization

```r
# Install if not yet available
#install.packages(c("ggplot2", "maps", "dplyr"))

# Load libraries
library(ggplot2)
library(maps)
```

Required Packages

```
## Warning: package 'maps' was built under R version 4.4.3
```

```
##
## Attaching package: 'maps'
## The following object is masked from 'package:purrr':
##
##     map
```

```r
library(dplyr)
```

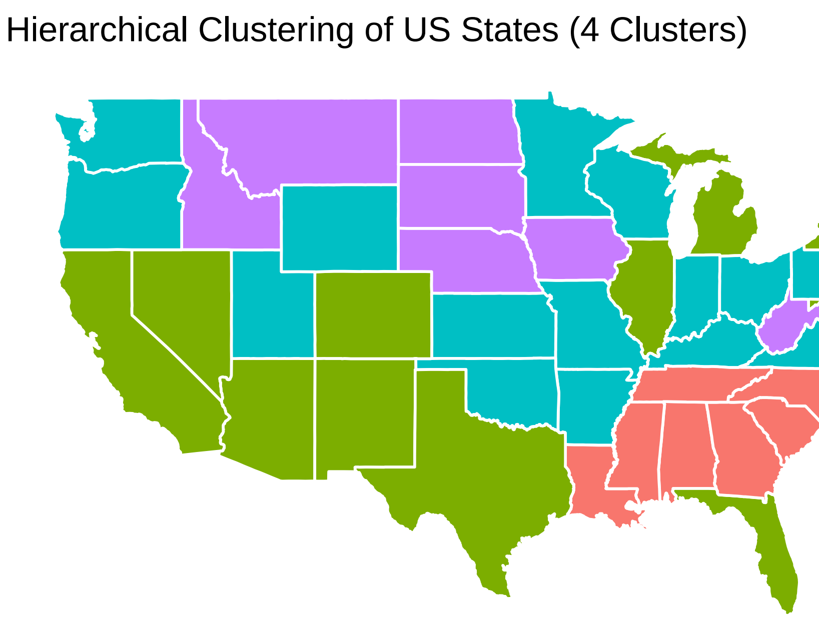
```r
# Get US map data
states_map <- map_data("state")

# Make sure state names are in lowercase to match map data
state_names <- tolower(rownames(USArrests))

# Create a data frame with state and cluster assignment
cluster_df <- data.frame(
  state = state_names,
  cluster = as.factor(cluster_assignments)
)
```

Step 1: Prepare US Map Data

```r
# Join cluster info to map data
map_with_clusters <- left_join(states_map, cluster_df, by = c("region" = "state"))
```

Step 2: Merge Cluster Info with Map Data

```r
# Plot US map with states colored by cluster
ggplot(map_with_clusters, aes(x = long, y = lat, group = group, fill = cluster)) +
  geom_polygon(color = "white") +
  coord_fixed(1.3) +
  labs(title = "Hierarchical Clustering of US States (4 Clusters)",
       fill = "Cluster") +
  theme_minimal() +
  theme(axis.text = element_blank(),
        axis.title = element_blank(),
        panel.grid = element_blank())
```



Step 3: Plot the US Map Colored by Cluster

What This Map Shows:

- Each **state is colored by its assigned cluster** (from the dendrogram you cut into 4).
- The **clusters are based on standardized crime statistics** — not geography.
- You'll visually see how certain **regions share similar crime profiles**, even if far apart (e.g., Alaska may be grouped with Southern states).

Check-in: *How do we decide the number of clusters from a dendrogram?* One common approach is to look for a large jump in the vertical distances between merges. This indicates that dissimilar clusters are being forced together at that height. Cutting just before a big jump yields a sensible cluster count. In our dendrogram, we chose 4 clusters by visually inspecting the merge distances. There is no single "correct" answer – domain knowledge and cluster validation methods (like silhouette scores) can also inform the choice.

Hierarchical clustering is a powerful exploratory tool. Its advantages are the intuitive dendrogram visualization and not requiring a pre-specified k. However, it can be computationally expensive on large datasets (because computing and storing all pairwise distances is $O(n^2)$). It's often used on smaller datasets or as a precursor to other analyses.

2. K-Means Clustering

Concept: K-Means clustering is a popular partitional clustering algorithm that divides the data into k clusters, where k is chosen by the analyst. The goal of K-means is to create clusters such that **within-cluster variation** is minimized – in other words, points in the same cluster are as similar as possible. Each cluster is defined by its **centroid** (the mean of all points in the cluster). At a high level, the K-means algorithm works as follows:

1. Choose the number of clusters k.
2. Initialize k cluster centroids (often random points in the data space).
3. Assign each observation to the nearest centroid (using Euclidean distance by default).
4. Update the centroid of each cluster as the mean of the points currently assigned to it.
5. Repeat steps 3-4 until cluster assignments stop changing (the algorithm has converged).

This procedure is essentially an iterative refinement: each iteration reduces the total within-cluster sum-of-squares (the objective function). K-means will converge to a solution, but it may be a *local* optimum. To improve results, it's recommended to run K-means with multiple random starts (`nstart` option in R) and take the best solution (lowest within-cluster variance).

Choosing k: The proper number of clusters is not known a priori. A common heuristic is the **elbow method** – run K-means for a range of k values and plot the total within-cluster sum of squares. The plot typically decreases as k increases (more clusters => less within-cluster variance). Look for an "elbow" point where the rate of decrease sharply slows; that k might be a good trade-off between simplicity and fidelity. Other methods include average silhouette width or the gap statistic, but the elbow method is a good starting point.

Just like hierarchical clustering, it's important to standardize features for K-means (if they are on different scales). Also, K-means assumes convex (roughly spherical) clusters of roughly similar sizes; it may struggle with very elongated or unevenly sized clusters.

Example – K-means on USArrests: To compare with hierarchical clustering, let's apply K-means to the same USArrests data. We will attempt $k = 4$ clusters (to see if we get a similar grouping as the hierarchical example) and use multiple starts for a robust result.

```r
# K-means clustering on USArrests data

set.seed(42)  # for reproducibility
# Use the scaled data from before (USArrests_scaled)
k4 <- kmeans(USArrests_scaled, centers = 4, nstart = 25)
# nstart=25 will try 25 different random initial cluster sets and pick the best.

# Examine the results:
print(k4)
```

```
## K-means clustering with 4 clusters of sizes 13, 16, 8, 13
##
## Cluster means:
##       Murder    Assault   UrbanPop        Rape
## 1 -0.9615407 -1.1066010 -0.9301069 -0.96676331
## 2 -0.4894375 -0.3826001  0.5758298 -0.26165379
## 3  1.4118898  0.8743346 -0.8145211  0.01927104
## 4  0.6950701  1.0394414  0.7226370  1.27693964
##
## Clustering vector:
##        Alabama         Alaska        Arizona       Arkansas     California
##              3              4              4              3              4
##       Colorado    Connecticut       Delaware        Florida        Georgia
##              4              2              2              4              3
##         Hawaii          Idaho       Illinois        Indiana           Iowa
##              2              1              4              2              1
##         Kansas       Kentucky      Louisiana          Maine       Maryland
##              2              1              3              1              4
##  Massachusetts       Michigan      Minnesota    Mississippi       Missouri
##               2              4              1              3              4
##         Montana       Nebraska         Nevada  New Hampshire     New Jersey
##               1              1              4              1              2
##      New Mexico       New York North Carolina   North Dakota           Ohio
##               4              4              3              1              2
##        Oklahoma         Oregon   Pennsylvania   Rhode Island South Carolina
##               2              2              2              2              3
##    South Dakota      Tennessee          Texas           Utah        Vermont
##               1              3              4              2              1
##        Virginia     Washington  West Virginia      Wisconsin        Wyoming
##               2              2              1              1              2
##
## Within cluster sum of squares by cluster:
## [1] 11.952463 16.212213  8.316061 19.922437
##  (between_SS / total_SS =  71.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

You've just run **K-Means clustering with `k = 4`** on the `USArrests` dataset using the standardized crime statistics. Let's now **understand what the K-Means results tell us**

Understanding the Output

Kmeans()' Summary

You ran:

```r
k4 <- kmeans(USArrests_scaled, centers = 4, nstart = 25)
```

This means:

- You **asked for 4 clusters** (`centers = 4`)
- You told R to try 25 different initial placements of clusters and pick the best (`nstart = 25`)
- R returns the best clustering based on minimizing **within-cluster variation**

Cluster Sizes

```
clusters of sizes 13, 16, 8, 13
```

- Cluster 1: 13 states
- Cluster 2: 16 states
- Cluster 3: 8 states
- Cluster 4: 13 states

This is fairly balanced — no single cluster dominates.

Cluster Means (Centroids)

Each row below represents a cluster's "average" profile based on the **standardized** (mean = 0, SD = 1) features:

Cluster	Murder	Assault	UrbanPop	Rape	Interpretation
1	-0.96	-1.11	-0.93	-0.97	Very low crime, low urban
2	-0.49	-0.38	+0.58	-0.26	Low–moderate crime, urban
3	+1.41	+0.87	-0.81	+0.02	High murder, assault; less urban
4	+0.70	+1.04	+0.72	+1.28	High crime all around, highly urban

These values are the **mean standardized values** for each variable in that cluster.

> Reminder: You **scaled** your data using `scale()` so that each variable has:
>
> - **Mean = 0**
> - **Standard Deviation = 1**

So, a value of:

- `0` means "average" (exactly at the dataset mean),
- Positive values (e.g., `+1`) mean **higher than average**,
- Negative values (e.g., `-1`) mean **lower than average**.

This allows you to compare across variables **on equal footing**.

Let's go through **Cluster 1**:

Variable	Value	Interpretation
Murder	-0.96	Almost 1 SD below average → **Low**
Assault	-1.11	Over 1 SD below average → **Very low**
UrbanPop	-0.93	Low percentage of urban population
Rape	-0.97	Very low rape rate

Plain-English Summary:

> States in Cluster 1 have **lower-than-average** values across all four variables, especially Assault and Rape. Urban population is also below average. So, these states are likely:
>
> - More rural,
> - Less violent,
> - Less densely populated or urbanized.

Hence, we describe Cluster 1 as:

> **"Very low crime, low urban"**

Let's contrast with Cluster 4:

Variable	Value	Interpretation
Murder	+0.70	High murder rate
Assault	+1.04	Very high assault rate
UrbanPop	+0.72	High urban population
Rape	+1.28	Very high rape rate

Plain-English Summary:

States in this cluster are:

- Highly urbanized,
- Exhibit high rates of **all forms of violent crime**.

So we describe Cluster 4 as:

"High crime all around, highly urban"

Rule of Thumb for Interpreting Scaled Values

Value Range	Meaning
~ -2.0 or below	Much lower than average
~ -1.0	Clearly below average
~ 0.0	About average
~ +1.0	Clearly above average
~ +2.0 or above	Much higher than average

Final Mental Model:

Think of the centroid values as **Z-scores** for each feature within a group:

- Negative = Below average for the population
- Positive = Above average
- The farther from 0, the more "extreme" the trait is

So when you see:

```
Cluster 1:
Murder    = -0.96
Assault   = -1.11
UrbanPop  = -0.93
Rape      = -0.97
```

You can safely summarize:

"This is a cluster of states with **low urbanization and very low crime rates** across the board."

Let’s Explore the States in Each Cluster

We can summarize the state-to-cluster assignments like this:

```
cluster_states <- data.frame(State = rownames(USArrests),
                             Cluster = k4$cluster)
split(cluster_states$State, cluster_states$Cluster)
```

```
## $`1`
##  [1] "Idaho"         "Iowa"          "Kentucky"      "Maine"
##  [5] "Minnesota"     "Montana"       "Nebraska"      "New Hampshire"
##  [9] "North Dakota"  "South Dakota"  "Vermont"       "West Virginia"
## [13] "Wisconsin"
##
## $`2`
##  [1] "Connecticut"   "Delaware"      "Hawaii"        "Indiana"
##  [5] "Kansas"        "Massachusetts" "New Jersey"    "Ohio"
##  [9] "Oklahoma"      "Oregon"        "Pennsylvania"  "Rhode Island"
## [13] "Utah"          "Virginia"      "Washington"    "Wyoming"
##
## $`3`
## [1] "Alabama"        "Arkansas"       "Georgia"        "Louisiana"
## [5] "Mississippi"    "North Carolina" "South Carolina" "Tennessee"
##
## $`4`
##  [1] "Alaska"     "Arizona"    "California" "Colorado"   "Florida"
##  [6] "Illinois"   "Maryland"   "Michigan"   "Missouri"   "Nevada"
## [11] "New Mexico" "New York"   "Texas"
```

BONUS: Compare With Hierarchical Clustering

If you saved your **hierarchical clustering vector**, you can do this comparison:

```
# Compare the two clustering methods
hc_clusters <- cutree(hc_complete, k = 4)

table(Hierarchical = hc_clusters, KMeans = k4$cluster)
```

```
##             KMeans
## Hierarchical  1  2  3  4
##            1  0  0  7  1
##            2  0  0  0 11
##            3  3 16  1  1
##            4 10  0  0  0
```

This **confusion matrix** will show how similarly the two methods grouped the states — perfect for comparing alignment and consistency.

This table shows how many states from each hierarchical cluster were placed into each K-means cluster:

```
            KMeans
Hierarchical  1  2  3  4
```

```
1  0  0  7  1
2  0  0  0 11
3  3 16  1  1
4 10  0  0  0
```

How to Read This

- Each **row** = a cluster from **Hierarchical Clustering**
- Each **column** = a cluster from **K-Means Clustering**
- Each **cell** = number of states assigned to that row–column combo

Hierarchical Cluster 1 → Mostly KMeans Cluster 3

```
Hierarchical 1: 0  0  7  1
```

- **7 states** went to **KMeans Cluster 3**
- These are states with **high murder and assault**, but **lower urban population**

Interpretation:

> **High-crime rural states** (Both methods agree these states have violence, but KMeans labeled them as cluster 3)

Hierarchical Cluster 2 → 100% KMeans Cluster 4

```
Hierarchical 2: 0  0  0 11
```

- All 11 states from this hierarchical cluster fell into **KMeans Cluster 4**
- These are **high-crime, highly urbanized states**

Interpretation:

> **High-crime urban states** Perfect alignment — both methods agree on these "big-city, violent" states.

Hierarchical Cluster 3 → Mostly KMeans Cluster 2 (but mixed)

```
Hierarchical 3: 3 16  1  1
```

- Majority (16) went to KMeans Cluster 2
- A few scattered across other clusters

Interpretation:

> **Moderate or mixed states** This is the "average" category — both methods interpret this group with some flexibility.

Hierarchical Cluster 4 → 100% KMeans Cluster 1

```
Hierarchical 4: 10  0  0  0
```

- All 10 states fall into **KMeans Cluster 1**
- These are **low-crime, rural or low-urbanization** states

Interpretation:

> **Low-crime rural states** Perfect agreement again — both methods identify the "safest" states.

Final Summary

Cluster Type	What We See	Updated Interpretation
Hier 1 → KMeans 3	Strong overlap	High-crime rural states
Hier 2 → KMeans 4	Perfect match	High-crime urban states
Hier 3 → KMeans 2	Mixed with some spread	Moderate/mixed crime states
Hier 4 → KMeans 1	Perfect match	Low-crime, low-urbanization states

Below is the **fully updated R code** for visualizing the results of **Hierarchical Clustering vs. K-Means Clustering** on a **side-by-side US map**, with:

```r
# --- Install and load required libraries (if not yet installed) ---
#install.packages(c("ggplot2", "dplyr", "maps", "gridExtra"), dependencies = TRUE)
library(ggplot2)
library(dplyr)
library(maps)
library(gridExtra)
```

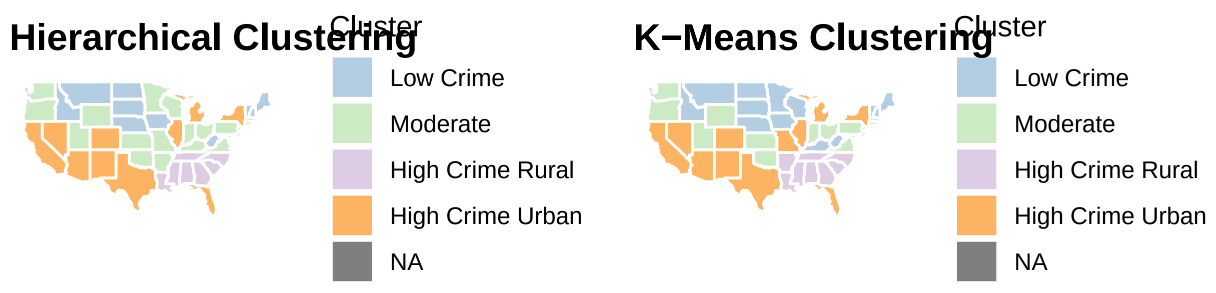
```
## Warning: package 'gridExtra' was built under R version 4.4.3
```

```
##
## Attaching package: 'gridExtra'
```

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

```r
# --- Load and prepare the USArrests data ---
data("USArrests")
USArrests_scaled <- scale(USArrests)

# --- Hierarchical clustering ---
dist_matrix <- dist(USArrests_scaled)
hc_complete <- hclust(dist_matrix, method = "complete")
hc_clusters <- cutree(hc_complete, k = 4)

# --- K-means clustering ---
```

```r
set.seed(42)
k4 <- kmeans(USArrests_scaled, centers = 4, nstart = 25)
kmeans_clusters <- k4$cluster

# --- Create a data frame of clusters and match to state names ---
state_names <- tolower(rownames(USArrests))

cluster_df <- data.frame(
  state = state_names,
  Hierarchical = as.factor(hc_clusters),
  KMeans = as.factor(kmeans_clusters)
)

# --- Define consistent cluster labels across both methods ---
# You may customize labels based on prior interpretation
cluster_df <- cluster_df %>%
  mutate(
    Hierarchical_named = recode(Hierarchical,
      `1` = "High Crime Rural",
      `2` = "High Crime Urban",
      `3` = "Moderate",
      `4` = "Low Crime"
    ),
    KMeans_named = recode(KMeans,
      `1` = "Low Crime",
      `2` = "Moderate",
      `3` = "High Crime Rural",
      `4` = "High Crime Urban"
    )
  )

# --- Force consistent factor levels and assign colors ---
cluster_levels <- c("Low Crime", "Moderate", "High Crime Rural", "High Crime Urban")
cluster_colors <- c(
  "Low Crime" = "#B3CDE3",           # Light blue
  "Moderate" = "#CCEBC5",            # Light green
  "High Crime Rural" = "#DECBE4",    # Light purple
  "High Crime Urban" = "#FDB462"     # Orange
)

cluster_df <- cluster_df %>%
  mutate(
    Hierarchical_named = factor(Hierarchical_named, levels = cluster_levels),
    KMeans_named = factor(KMeans_named, levels = cluster_levels)
  )

# --- Merge with US map data ---
states_map <- map_data("state")
map_data_named <- left_join(states_map, cluster_df, by = c("region" = "state"))

# --- Plot: Hierarchical Clustering Map ---
p1 <- ggplot(map_data_named, aes(long, lat, group = group, fill = Hierarchical_named)) +
  geom_polygon(color = "white") +
  coord_fixed(1.3) +
  scale_fill_manual(values = cluster_colors, drop = FALSE) +
  labs(title = "Hierarchical Clustering", fill = "Cluster") +
  theme_minimal() +
  theme(axis.text = element_blank(), axis.title = element_blank(), panel.grid = element_blank(),
        plot.title = element_text(size = 14, face = "bold"))

# --- Plot: K-Means Clustering Map ---
p2 <- ggplot(map_data_named, aes(long, lat, group = group, fill = KMeans_named)) +
  geom_polygon(color = "white") +
  coord_fixed(1.3) +
  scale_fill_manual(values = cluster_colors, drop = FALSE) +
  labs(title = "K-Means Clustering", fill = "Cluster") +
  theme_minimal() +
  theme(axis.text = element_blank(), axis.title = element_blank(), panel.grid = element_blank(),
        plot.title = element_text(size = 14, face = "bold"))

# --- Display side-by-side plots ---
grid.arrange(p1, p2, ncol = 2)
```



Check-in: *Why did we set* `nstart = 25` *in the K-means run?* K-means can converge to a local minimum depending on initial random centroids. By trying 25 random starts and picking the best result, we reduce the chance of getting a bad clustering due to unfortunate initialization. It's a recommended practice to use a large `nstart` (e.g. 20–50) for more reliable results.

We chose $k = 4$ here to mirror the hierarchical example. In practice, we would also examine the total within-cluster sum of squares for k=1,2,. . . , perhaps up to 10, and look for an elbow. (For brevity, we skip plotting the elbow chart, but you can do so with `tot.withinss` from the kmeans result.)

K-means is very efficient for large datasets, but remember it assumes clusters are convex and similar size. It also doesn't handle outliers well (an extreme outlier can skew a centroid). Always pair clustering with domain knowledge: *Do the clusters make sense?* In our crime example, one cluster contained nearly all high-crime states, and another cluster contained low-crime, largely Northeastern states – which matches known regional crime patterns in the 1970s.