# SL project clusterng

Steven

20240216

# Step 1: Data Cleaning and Preparation

```
library(readr)
library(dplyr)
```

```
##
## 载入程辑包：'dplyr'
```

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

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

```
# Load the dataset

data <- read_csv(file.choose())  # select the file interactively
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```
## Rows: 34857 Columns: 21
```

```
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr  (8): Suburb, Address, Type, Method, SellerG, Date, CouncilArea, Regionname
## dbl (13): Rooms, Price, Distance, Postcode, Bedroom2, Bathroom, Car, Landsiz...
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

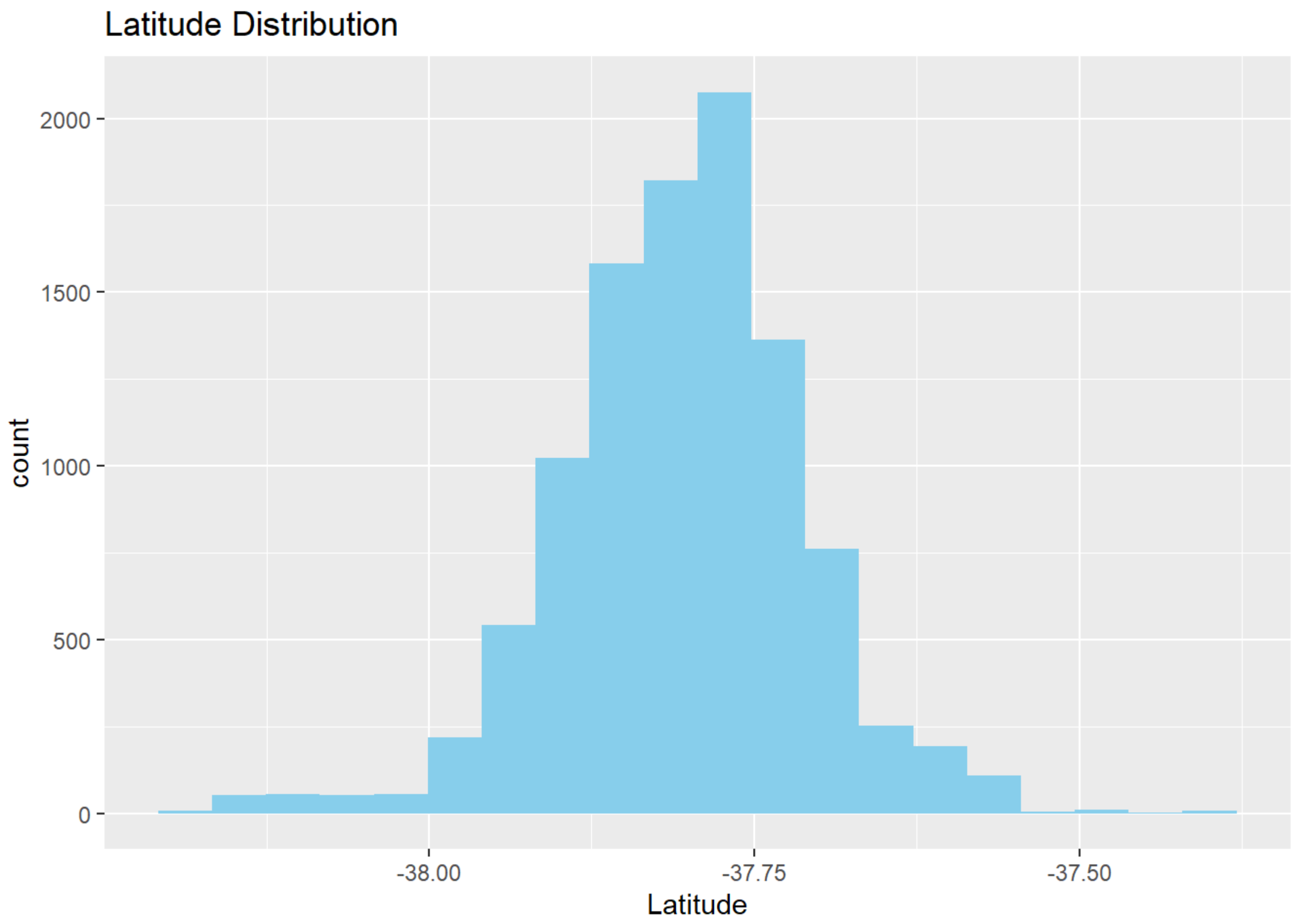
```
# Correcting the typos in column names
names(data)[names(data) == "Lattitude"] <- "Latitude"
names(data)[names(data) == "Longtitude"] <- "Longitude"
```

# Step 2: clean data

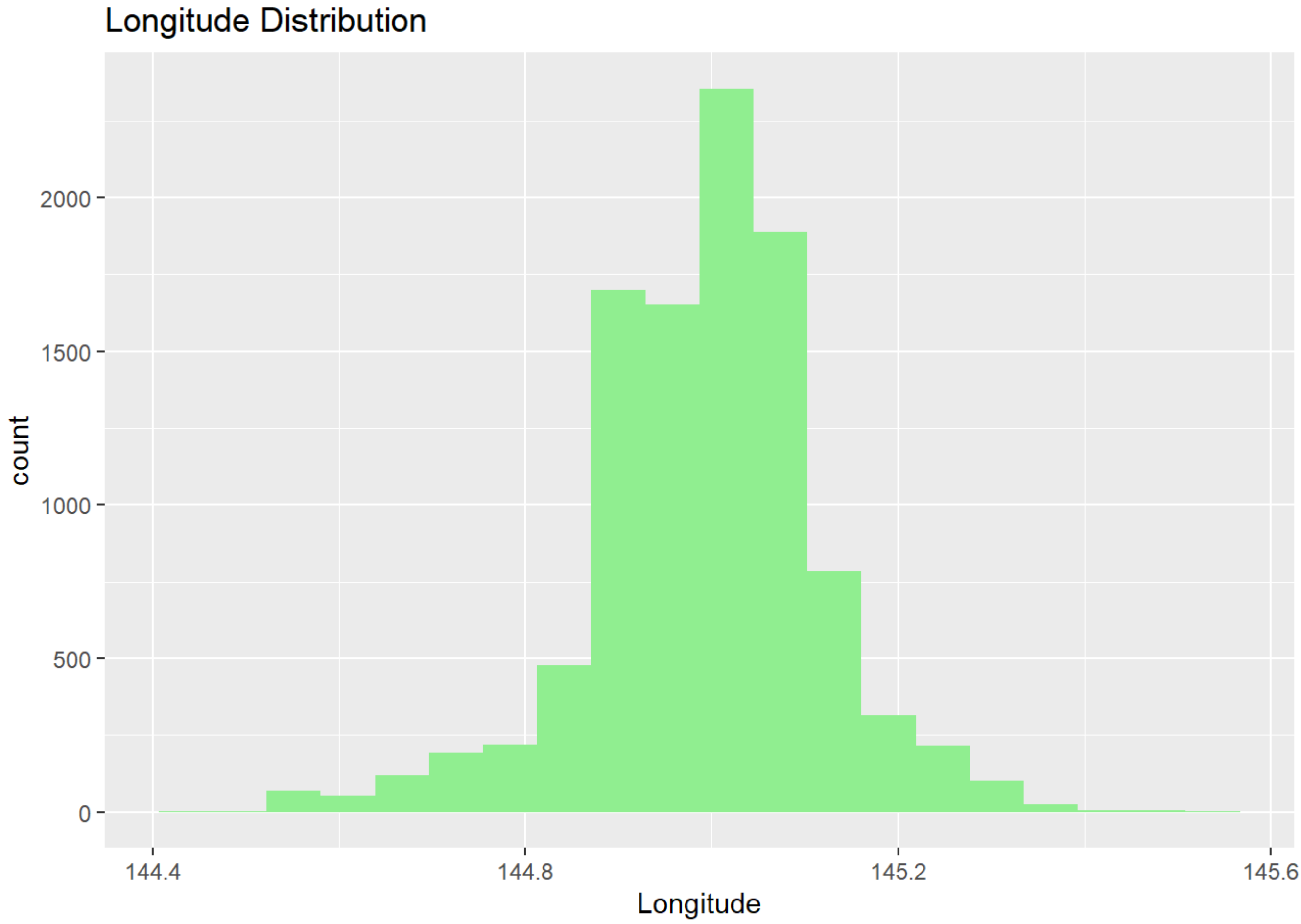
```
# Remove rows with NA values in specified columns and drop unnecessary columns
data <- data %>%
  filter(!is.na(Latitude), !is.na(Longitude), !is.na(YearBuilt), !is.na(Price), !is.na(Buildi
ngArea)) %>%
  select(Latitude, Longitude, YearBuilt, Price, BuildingArea)

# Calculate the property age and price per building area
data$PropertyAge <- as.numeric(format(Sys.Date(), "%Y")) - data$YearBuilt
data$PricePerArea <- data$Price / data$BuildingArea

# Drop rows with NA or infinite values after calculations
data <- na.omit(data)
data <- data[!is.infinite(data$PricePerArea),]
```

# Step 3: 3. Show Data Distributions:

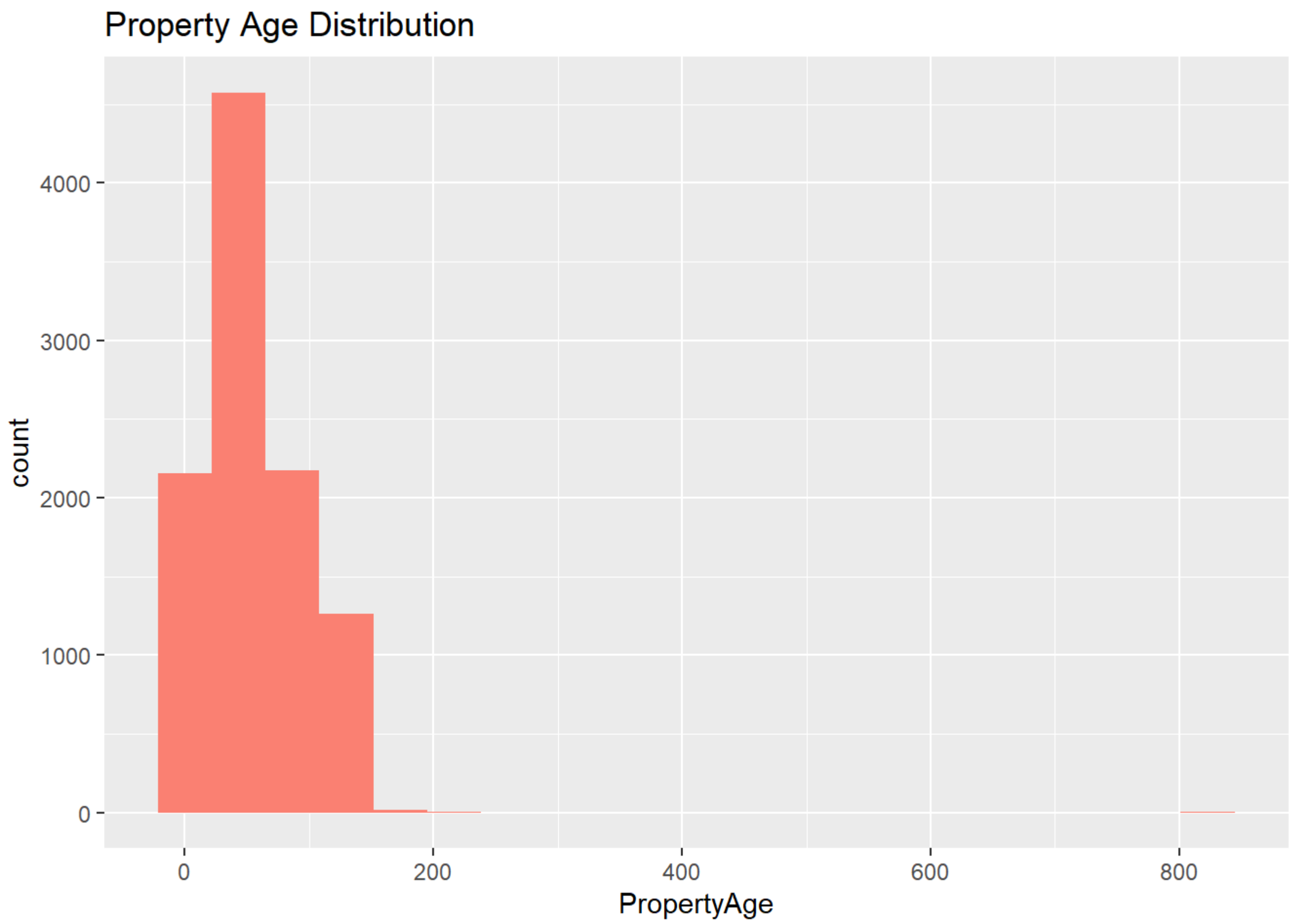
```
library(ggplot2)

# Plot distributions
ggplot(data, aes(x=Latitude)) + geom_histogram(bins=20, fill="skyblue") + ggtitle("Latitude D
istribution")
```



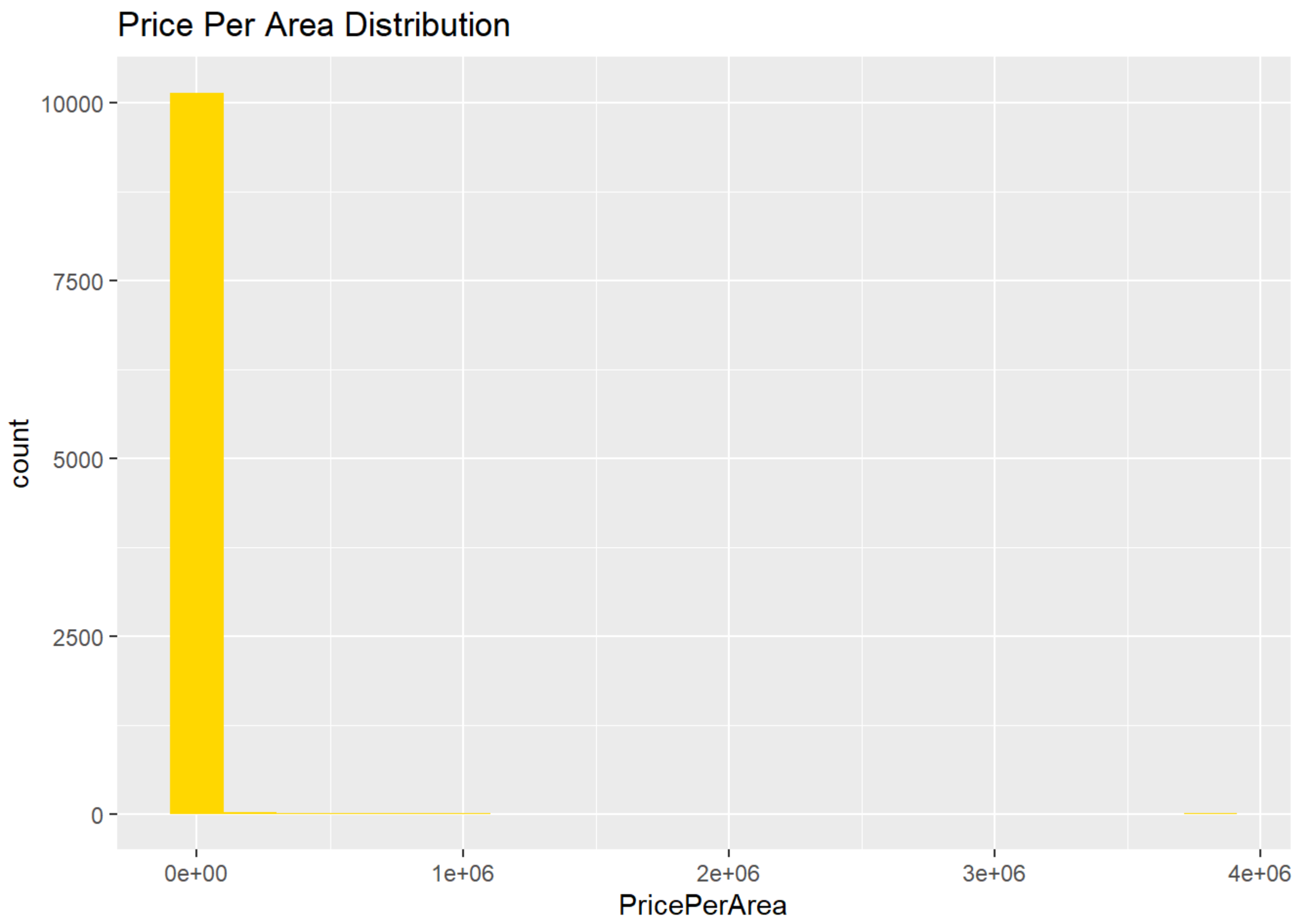

```
ggplot(data, aes(x=Longitude)) + geom_histogram(bins=20, fill="lightgreen") + ggtitle("Longit
ude Distribution")
```




```
ggplot(data, aes(x=PropertyAge)) + geom_histogram(bins=20, fill="salmon") + ggtitle("Property
Age Distribution")
```




```
ggplot(data, aes(x=PricePerArea)) + geom_histogram(bins=20, fill="gold") + ggtitle("Price Per
Area Distribution")
```




# Step 4: Implement Outlier Elimination Strategy

```
# Calculate IQR for each column and filter out outliers
for (col in c("PricePerArea")) {
  Q1 <- quantile(data[[col]], 0.25)
  Q3 <- quantile(data[[col]], 0.75)
  IQR <- Q3 - Q1
  data <- data[data[[col]] >= (Q1 - 1.5 * IQR) & data[[col]] <= (Q3 + 1.5 * IQR), ]
}
#for (col in c("Latitude", "Longitude", "PropertyAge", "PricePerArea")) {
```

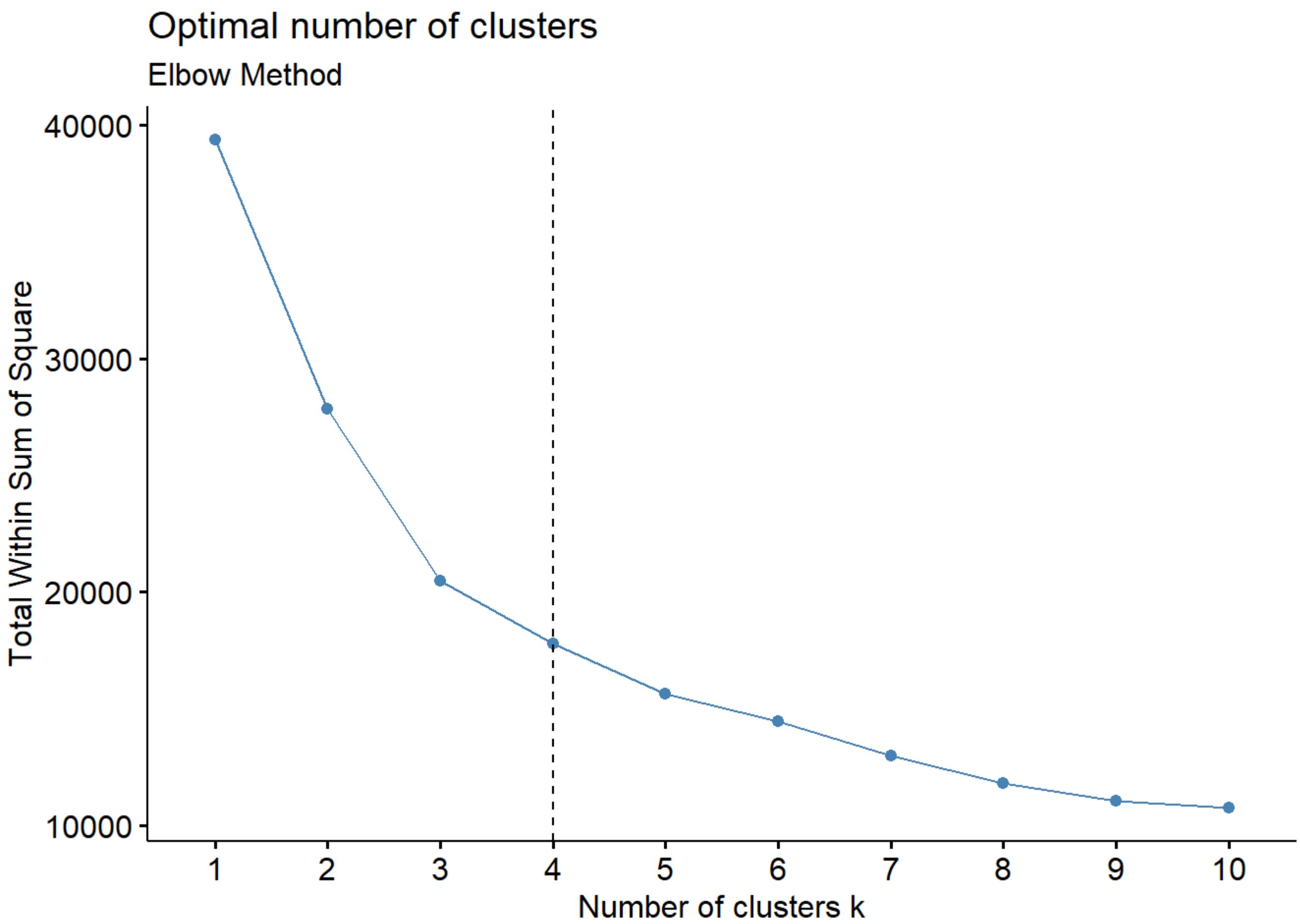
# Step 5: Further Steps (Clustering, Visualization, Comparison)

```
#install.packages("factoextra")
library(factoextra)
```

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

```
set.seed(123) # For reproducibility

# Scale the data
data_scaled <- scale(data[, c("Latitude", "Longitude", "PropertyAge", "PricePerArea")])

# Determine the optimal number of clusters
fviz_nbclust(data_scaled, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2) +
  labs(subtitle = "Elbow Method")
```

```
## Warning: 10迭代仍没有聚合
```



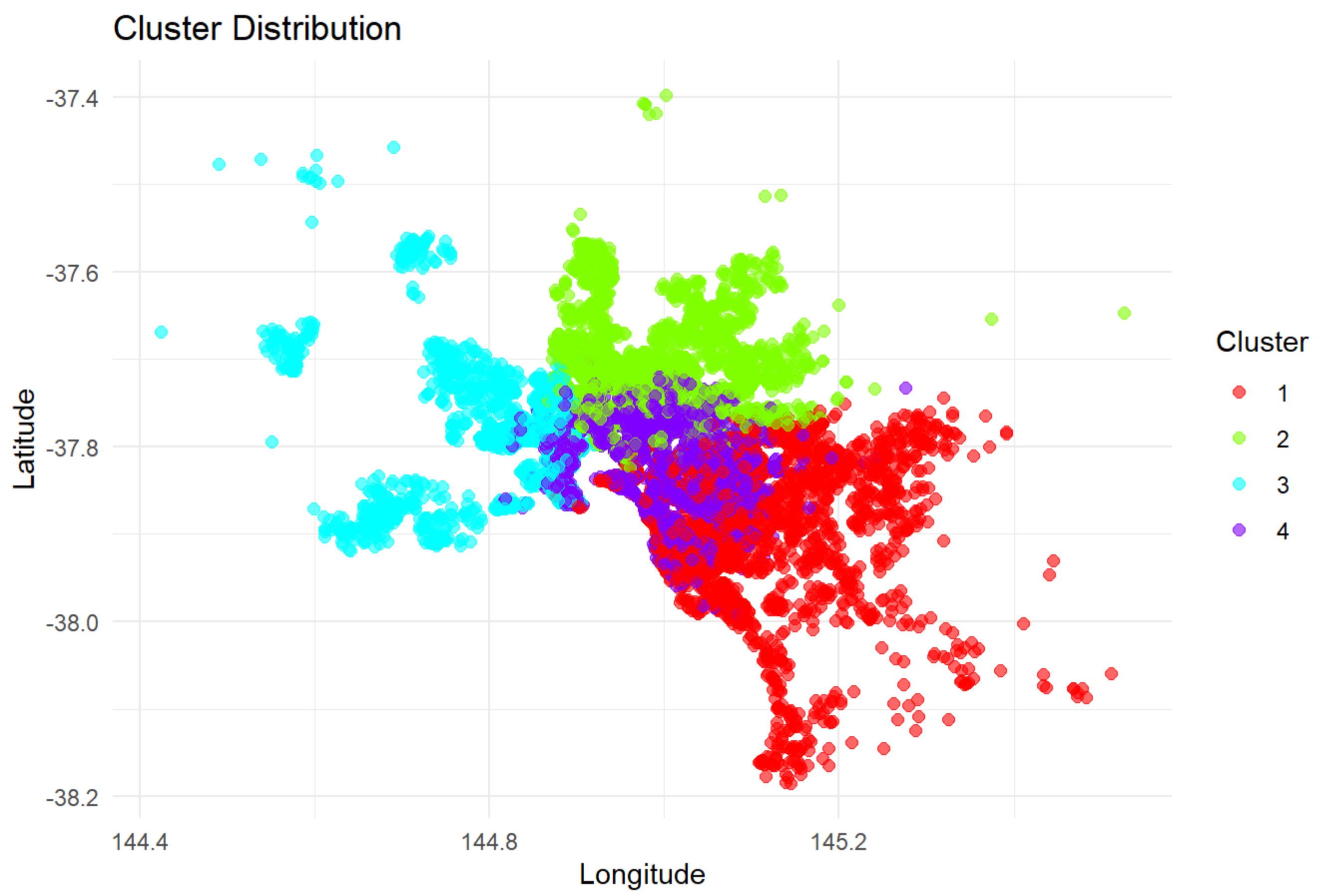

```
# Perform k-means clustering
set.seed(123) # Ensure reproducibility
k <- 4 # Assuming 4 is the chosen number of clusters
km_result <- kmeans(data_scaled, centers = k, nstart = 25)
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 492100)
```

```
# Add cluster assignment to the data
data$Cluster <- as.factor(km_result$cluster)
```

# Step 6: show result in geo map

```
library(ggplot2)

# Assuming 'data' is dataframe and it contains 'Latitude', 'Longitude', and 'Cluster' columns

ggplot(data, aes(x = Longitude, y = Latitude, color = Cluster)) +
  geom_point(alpha = 0.6, size = 2) +
  scale_color_manual(values = rainbow(length(unique(data$Cluster)))) +
  theme_minimal() +
  labs(title = "Cluster Distribution", x = "Longitude", y = "Latitude", color = "Cluster") +
  coord_fixed(ratio = 1) # This helps in keeping the aspect ratio consistent for geographical
data
```




# distance from center strategy

# Step 7: Calculate the Central Point

```
central_latitude <- mean(data$Latitude, na.rm = TRUE)
central_longitude <- mean(data$Longitude, na.rm = TRUE)
```

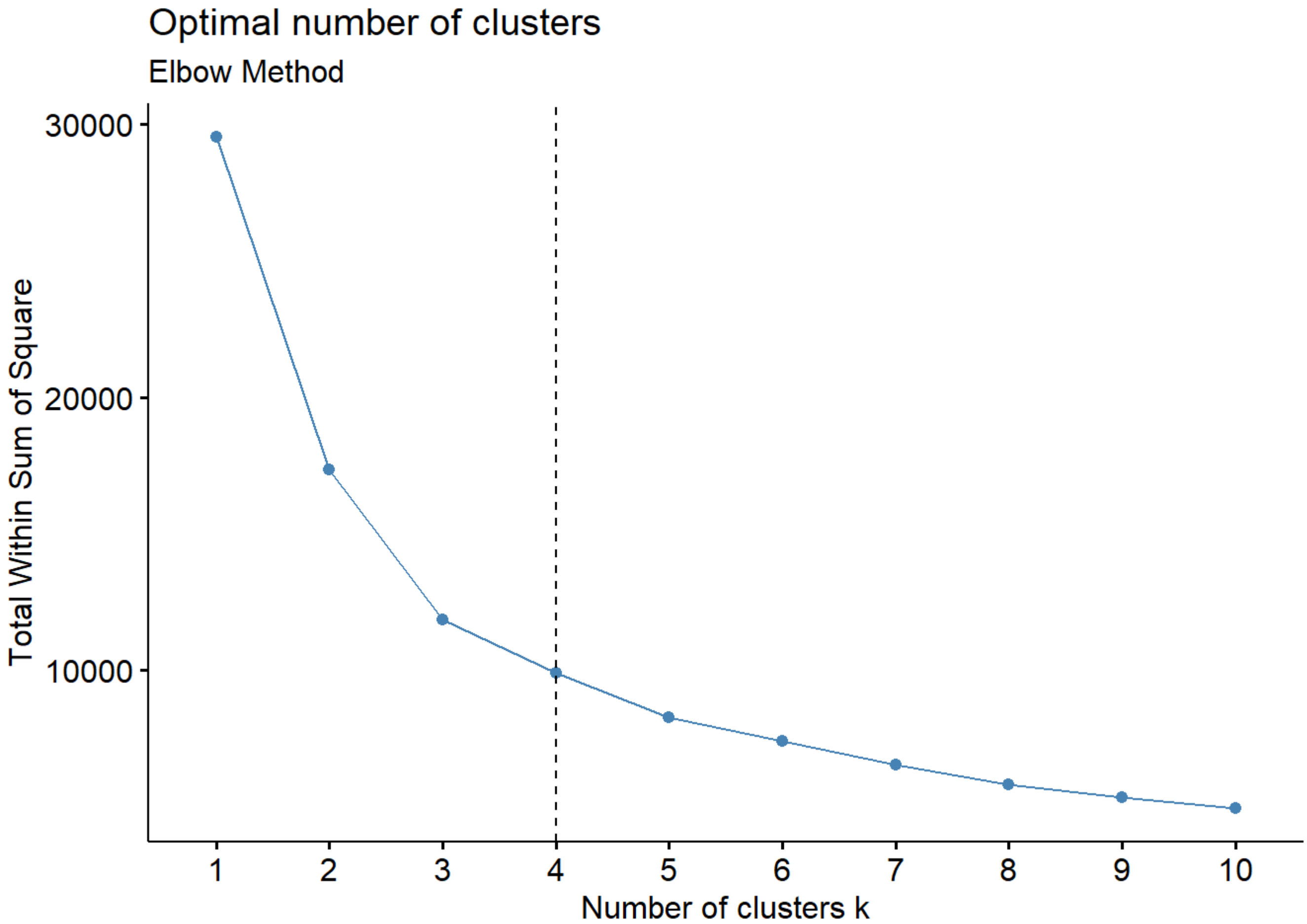
# Step 8: Calculate the Distance from the Center for

# Each Point

```
# Define the deg2rad function
deg2rad <- function(deg) {
  return(deg * (pi / 180))
}

# Haversine formula to calculate distances
haversine_distance <- function(lat1, long1, lat2, long2) {
  R <- 6371 # Earth radius in kilometers
  delta_lat <- deg2rad(lat2 - lat1)
  delta_long <- deg2rad(long2 - long1)
  a <- sin(delta_lat / 2)^2 + cos(deg2rad(lat1)) * cos(deg2rad(lat2)) * sin(delta_long / 2)^2
  c <- 2 * atan2(sqrt(a), sqrt(1 - a))
  d <- R * c
  return(d) # Distance in kilometers
}

# Apply the distance calculation for each row in the dataframe
data$DistanceFromCenter <- mapply(haversine_distance,
                                  lat1 = data$Latitude,
                                  long1 = data$Longitude,
                                  lat2 = central_latitude,
                                  long2 = central_longitude)
```

# Step 9: Cluster Using the New Feature

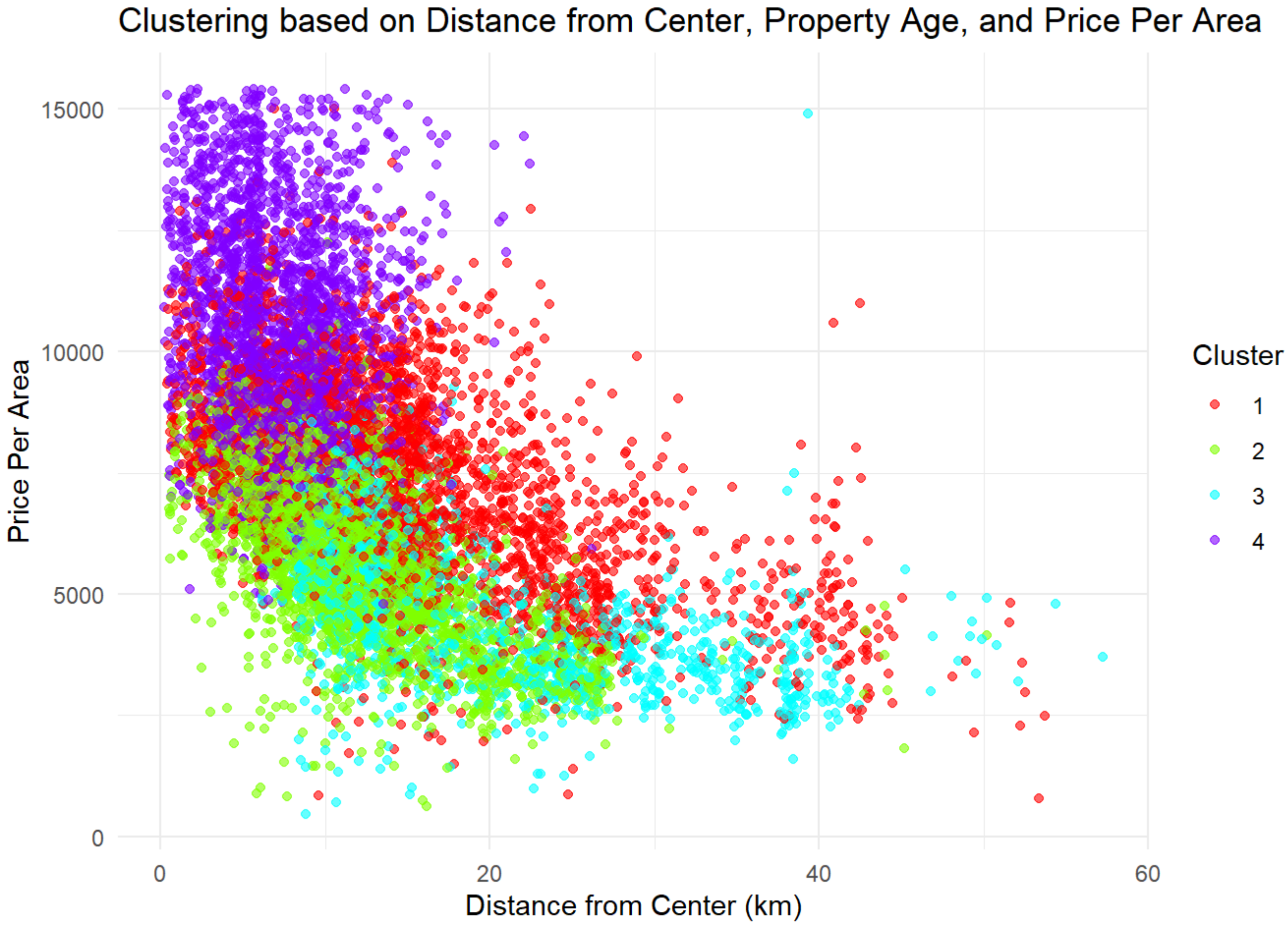
```
#install.packages("factoextra")
library(factoextra)

# Normalize the data before clustering
data_normalized <- scale(data[, c("DistanceFromCenter", "PropertyAge", "PricePerArea")])

# Determine the optimal number of clusters using the elbow method
set.seed(123) # Ensure reproducibility
fviz_nbclust(data_normalized, kmeans, method = "wss") +
  geom_vline(xintercept = 4, linetype = 2) +
  labs(subtitle = "Elbow Method")
```




```
# After visual inspection of the elbow plot, choose the optimal number of clusters
# For example, if the elbow plot suggests that 4 is a good choice:
num_clusters_optimal <- 4 # Adjust this based on the elbow method's outcome

# Perform k-means clustering with the optimal number of clusters
set.seed(123) # Ensure reproducibility again for the actual clustering
km_result <- kmeans(data_normalized, centers = num_clusters_optimal, nstart = 25)

# Add cluster assignment to the data
data$Cluster_km <- as.factor(km_result$cluster)
```

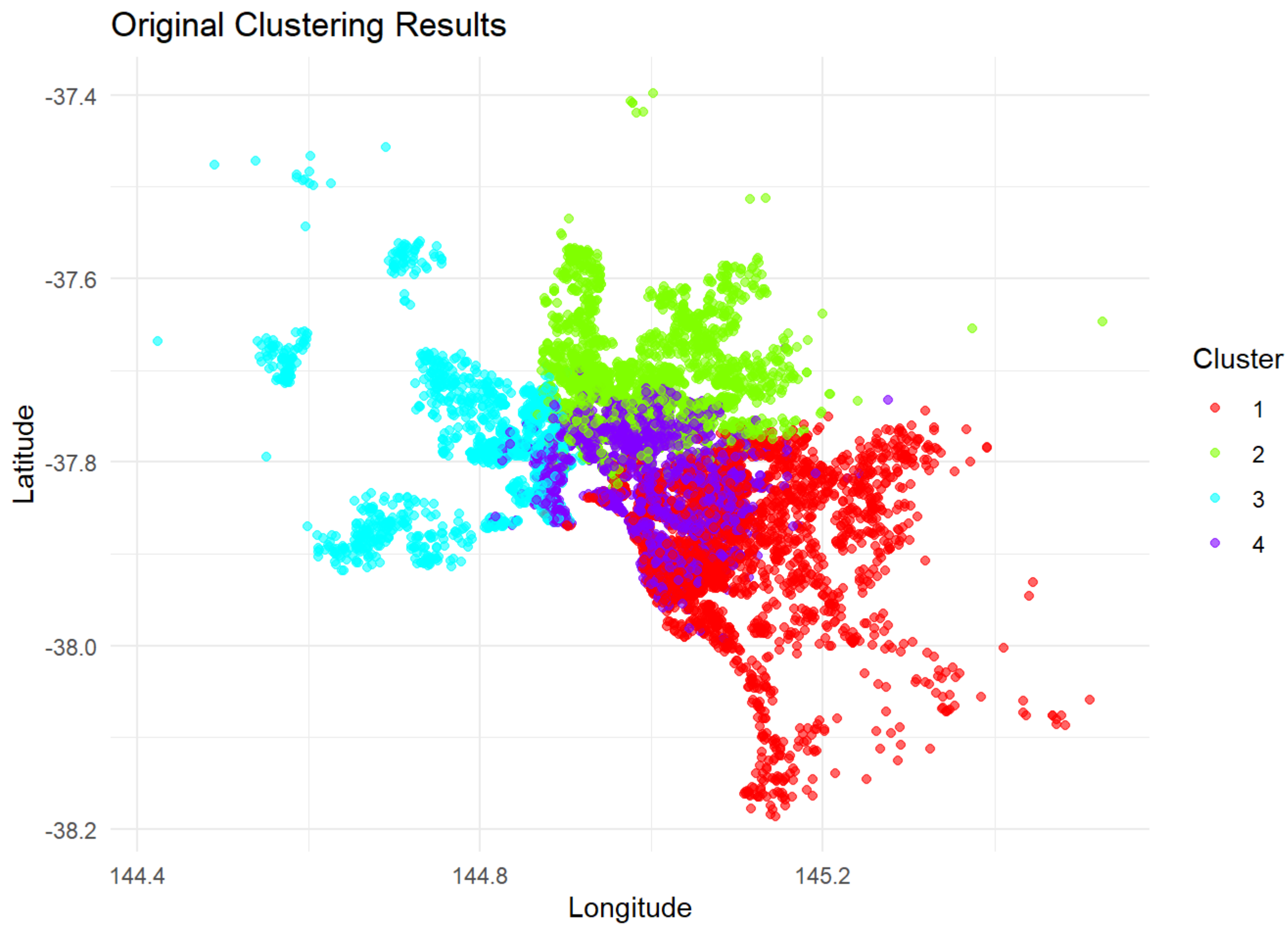
# Step 10: Visualization and Analysis

```
ggplot(data, aes(x = DistanceFromCenter, y = PricePerArea, color = Cluster)) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = rainbow(num_clusters_optimal)) +
  labs(title = "Clustering based on Distance from Center, Property Age, and Price Per Area",
       x = "Distance from Center (km)", y = "Price Per Area") +
  theme_minimal()
```

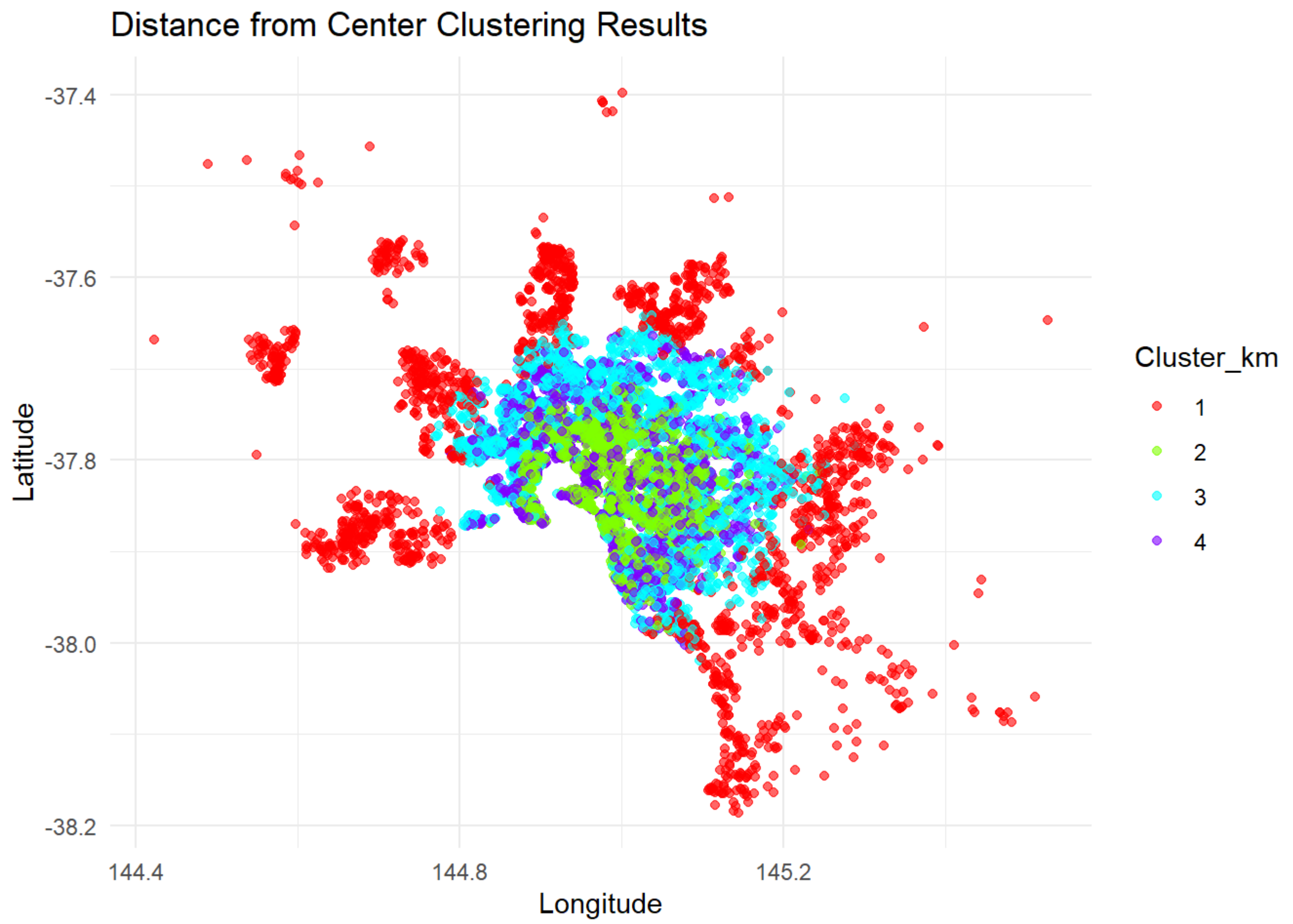



# Step 12: Visualizing Both Clustering Results

```
library(ggplot2)

# Visualization for original clustering based on latitude and longitude
ggplot(data, aes(x = Longitude, y = Latitude, color = as.factor(Cluster))) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = rainbow(length(unique(data$Cluster)))) +
  labs(title = "Original Clustering Based on Latitude and Longitude",
       x = "Longitude", y = "Latitude", color = "Cluster") +
  theme_minimal() +
  ggtitle("Original Clustering Results")
```




```
# Visualization for clustering based on distance from the center
ggplot(data, aes(x = Longitude, y = Latitude, color = as.factor(Cluster_km))) +
  geom_point(alpha = 0.6) +
  scale_color_manual(values = rainbow(length(unique(data$Cluster_km)))) +
  labs(title = "Clustering Based on Distance from Center",
       x = "Longitude", y = "Latitude", color = "Cluster_km") +
  theme_minimal() +
  ggtitle("Distance from Center Clustering Results")
```




# Step 13: Calculate Metrics for Both Clustering Approaches

```
# Assuming data is already prepared and normalized as needed

# Original spatial feature clustering
set.seed(123) # for reproducibility
kmeans_result_orig <- kmeans(data[, c("Longitude", "Latitude")], centers = 4, nstart = 25)

# Assuming you've decided on an appropriate number of centers (e.g., 4) after analysis such a
s the elbow method

# Clustering based on distance from the center
set.seed(123)
kmeans_result_km <- kmeans(data[, "DistanceFromCenter", drop = FALSE], centers = 4, nstart =
25)

# Update the data frame with cluster labels
data$Cluster <- kmeans_result_orig$cluster
data$Cluster_km <- kmeans_result_km$cluster

# Convert cluster labels to numeric if they're not already
data$Cluster_numeric <- as.numeric(data$Cluster)
data$Cluster_km_numeric <- as.numeric(data$Cluster_km)
```

```
library(cluster) # for silhouette calculations

# Calculate silhouette scores
silhouette_orig <- silhouette(data$Cluster_numeric, dist(data[, c("Longitude", "Latitude")]))
avg_silhouette_orig <- mean(silhouette_orig[, "sil_width"])

# For the distance-based clustering, assuming appropriate preparation
silhouette_km <- silhouette(data$Cluster_km_numeric, dist(data[, "DistanceFromCenter", drop =
FALSE]))
avg_silhouette_km <- mean(silhouette_km[, "sil_width"])

# Print the metrics for comparison
cat("Average Silhouette Score (Original):", avg_silhouette_orig, "\n")
```

```
## Average Silhouette Score (Original): 0.3427781
```

```
cat("Average Silhouette Score (Distance-based):", avg_silhouette_km, "\n")
```

```
## Average Silhouette Score (Distance-based): 0.5552604
```

```
# WSS values are already part of the kmeans result object
cat("WSS (Original):", kmeans_result_orig$tot.withinss, "\n")
```

```
## WSS (Original): 83.12818
```

```
cat("WSS (Distance-based):", kmeans_result_km$tot.withinss, "\n")
```

```
## WSS (Distance-based): 67887.45
```

# Step 14: additional visuals

```
# Check if 'plotly' package is installed; install it if not
if (!require(plotly)) {
  install.packages("plotly")
  library(plotly)
}
```

```
## 载入需要的程辑包：plotly
```

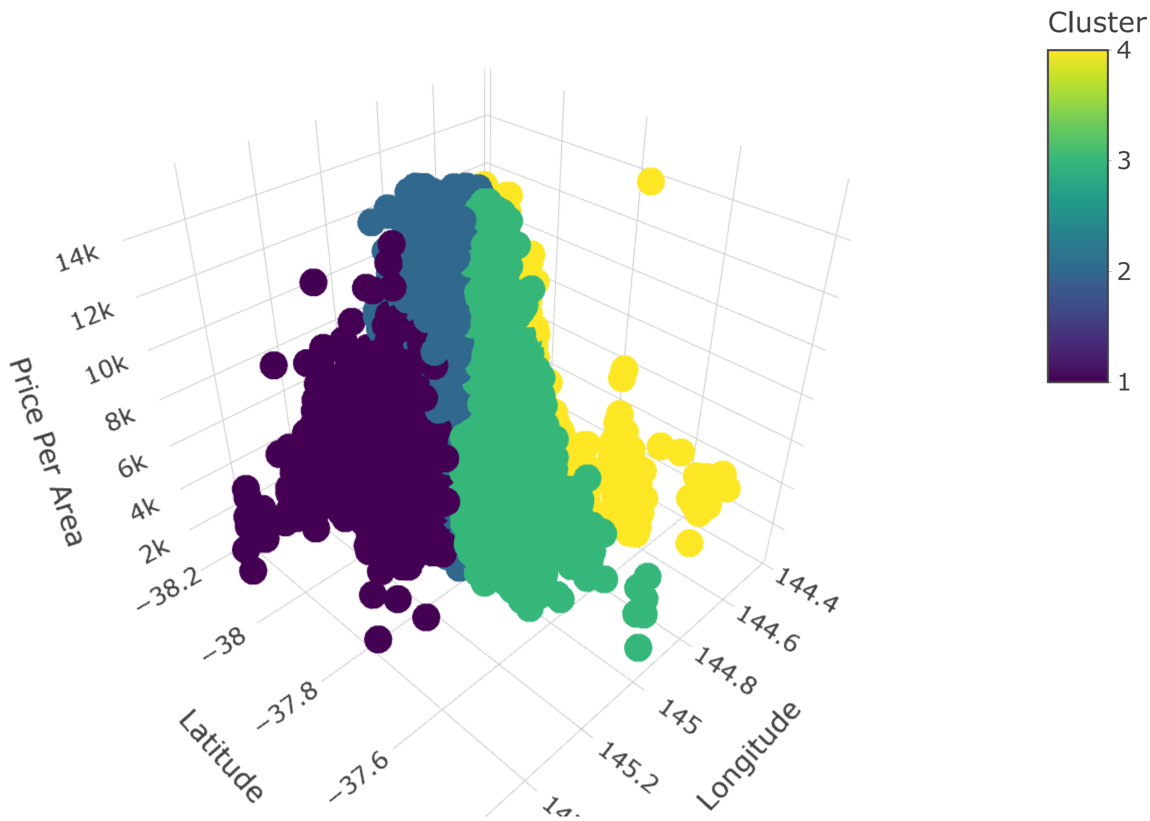
```
##
## 载入程辑包：'plotly'
```

```
## The following object is masked from 'package:ggplot2':
##
##     last_plot
```

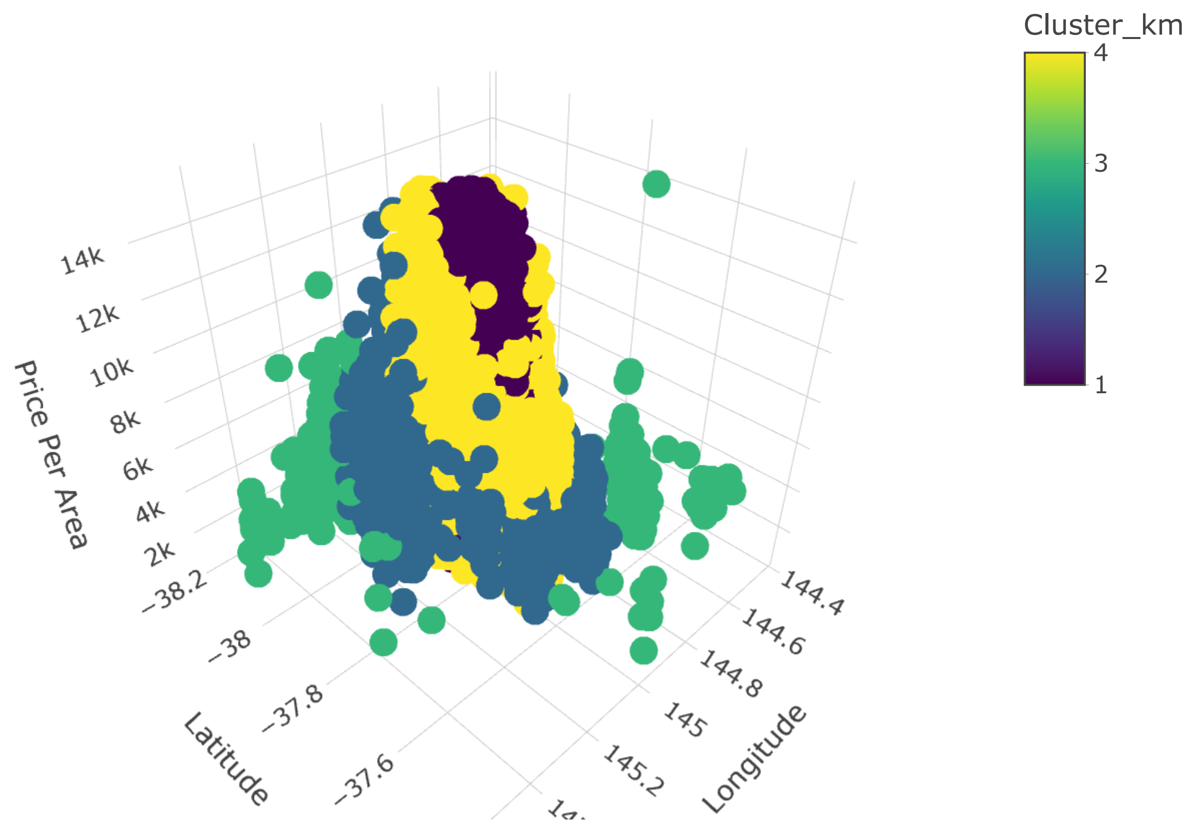
```
## The following object is masked from 'package:stats':
##
##     filter
```

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

```
fig <- plot_ly(data, x = ~Longitude, y = ~Latitude, z = ~PricePerArea, color = ~Cluster, type
= 'scatter3d', mode = 'markers')
fig <- fig %>% layout(scene = list(xaxis = list(title = 'Longitude'),
                                   yaxis = list(title = 'Latitude'),
                                   zaxis = list(title = 'Price Per Area')))
fig
```

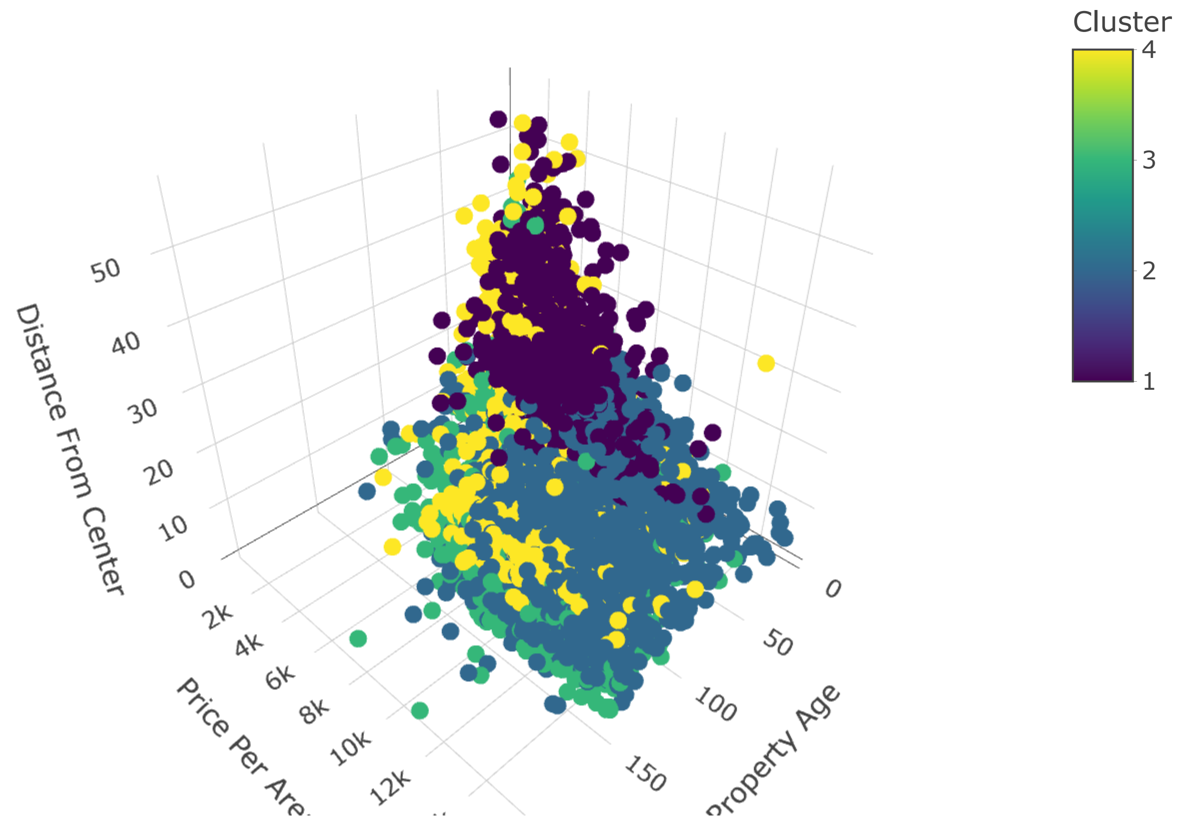


```
# Check if 'plotly' package is installed; install it if not
if (!require(plotly)) {
  install.packages("plotly")
  library(plotly)
}

fig <- plot_ly(data, x = ~Longitude, y = ~Latitude, z = ~PricePerArea, color = ~Cluster_km, t
ype = 'scatter3d', mode = 'markers')
fig <- fig %>% layout(scene = list(xaxis = list(title = 'Longitude'),
                                   yaxis = list(title = 'Latitude'),
                                   zaxis = list(title = 'Price Per Area')))
fig
```

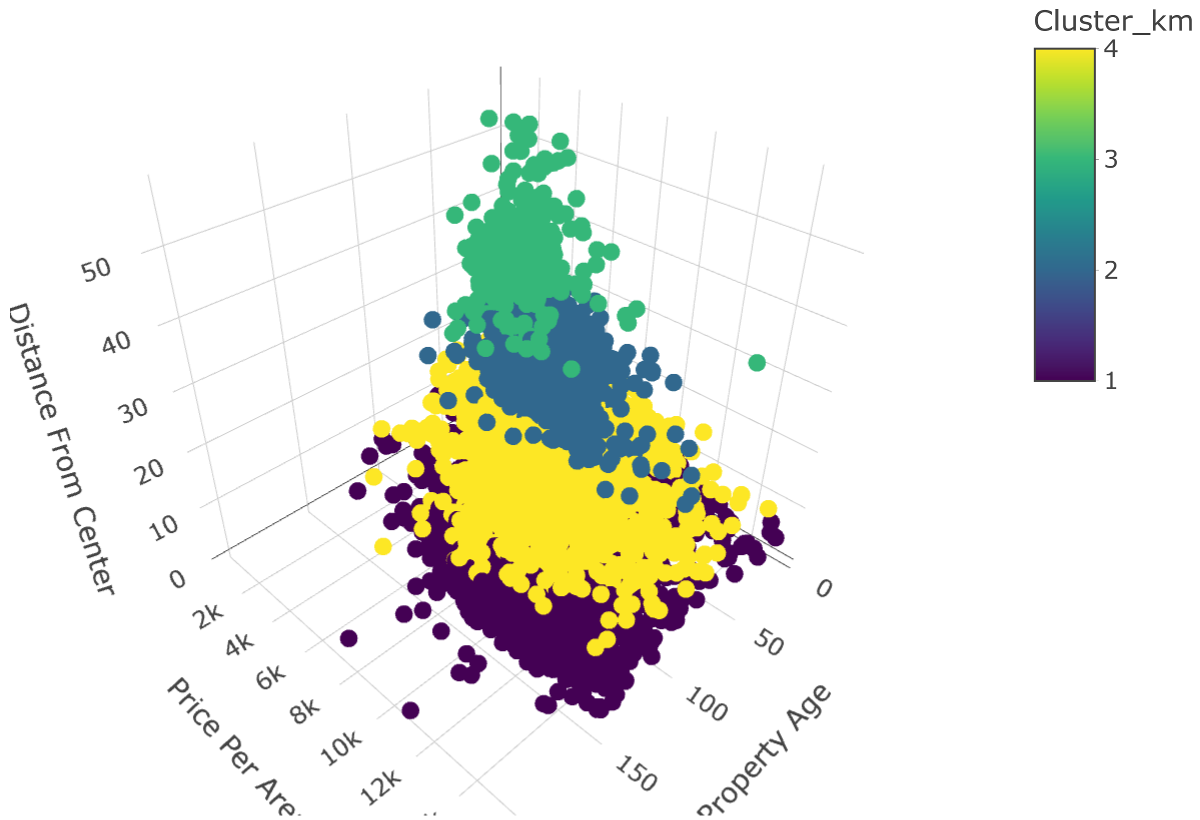



```
# Remove the row with the largest Property Age
data <- data[data$PropertyAge < max(data$PropertyAge), ]
```

```
# Check if 'plotly' package is installed; install it if not
if (!require(plotly)) {
  install.packages("plotly")
  library(plotly)
}

fig <- plot_ly(data, x = ~PropertyAge, y = ~PricePerArea, z = ~DistanceFromCenter, color = ~C
luster, type = 'scatter3d', mode = 'markers',
               marker = list(size = 5)) # Adjust marker size as needed
fig <- fig %>% layout(scene = list(xaxis = list(title = 'Property Age'),
                                   yaxis = list(title = 'Price Per Area'),
                                   zaxis = list(title = 'Distance From Center')))
fig
```



```r
# Check if 'plotly' package is installed; install it if not
if (!require(plotly)) {
  install.packages("plotly")
  library(plotly)
}


fig <- plot_ly(data, x = ~PropertyAge, y = ~PricePerArea, z = ~DistanceFromCenter, color = ~C
luster_km, type = 'scatter3d', mode = 'markers',
               marker = list(size = 5)) # Adjust marker size as needed
fig <- fig %>% layout(scene = list(xaxis = list(title = 'Property Age'),
                                   yaxis = list(title = 'Price Per Area'),
                                   zaxis = list(title = 'Distance From Center')))
fig
```




# Step 15: Final comment

# Step 16: Business implication