Cluster Analysis Submission

2022-11-12

Junseok Yang

With 'house_2' dataset, which is comprised of observations with no missing value at all, we are going to do unsupervised learning, specifically diverse clustering methods, to group observations sharing similar features and try to figure out any interesting trend that is going behind the scene.

Data Preprocessing again

Although we have done some data preprocessing steps, we still need to do more in order to have more suitable data for our analysis.

Read the data

```
house = read.csv("house_2.csv")
#head(house)
```

Unique values

```
# livingRoom, drawingRoom, bathRoom, constructionTime
# They should be either 'dbl' or 'int', not 'chr'
unique(house$livingRoom)
```

```
## [1] 2 3 1 4 5 6 0 7
```

```
unique(house$drawingRoom)
```

```
## [1] 1 2 0 4 3 5
```

```
unique(house$kitchen)
```

```
## [1] 1 0 2 3
```

```
unique(house$bathRoom)
```

```
## [1] 1 2 3 0 4 5 6
```

```r
unique(house$buildingType)
```

```
## [1] 1 4 3 2
```

```r
unique(house$constructionTime)
```

```
##  [1] "2005" "2004" "2008" "1960" "1997" "2009" "1991" "2001" "1990" "2011"
## [11] "2000" "1998" "2010" "1996" "1993" "2006" "2002" "Î´Öª" "2012" "1989"
## [21] "2003" "2007" "1994" "1984" "1992" "2014" "1985" "1999" "1979" "1981"
## [31] "1976" "1982" "1975" "1983" "1986" "1995" "1965" "2013" "1988" "1987"
## [41] "2015" "1955" "1980" "1978" "1958" "1970" "1956" "1977" "1964" "1963"
## [51] "1967" "2016" "1974" "1973" "1959" "1954" "1962" "1966" "1957" "1972"
## [61] "1971" "1953" "1968" "1961" "1950" "1952" "1969"
```

```r
unique(house$renovationCondition)
```

```
## [1] 3 4 1 2
```

```r
unique(house$buildingStructure)
```

```
## [1] 6 2 4 5 1 3
```

```r
unique(house$elevator)
```

```
## [1] 1 0
```

```r
unique(house$fiveYearsProperty)
```

```
## [1] 0 1
```

```r
unique(house$subway)
```

```
## [1] 1 0
```

```r
unique(house$district)
```

```
##  [1]  7  6  1 13 10  2  8  4  5  3  9 12 11
```

Check the number of observations with an unknown value

```r
# Number of obsevations with "Î´Öª"
nrow(house[house$constructionTime == "Î´Öª", ])
```

```
## [1] 7110
```

```
# Percentage of the observations
perc = (nrow(house[house$constructionTime == "Î´Öª", ]) / nrow(house)) * 100
perc
```

```
## [1] 4.461148
```

The variable 'constructionTime' has an unknown value of "Î´Öª", and the number of observations with that value is 7110 or 4.5% of the whole dataset. We can simply remove these observations considering that the percentage is pretty small.

Remove rows with "Î´Öª"

```
df = house[house$constructionTime != "Î´Öª", ]
#head(house, 3)
nrow(house) - nrow(df)
```

```
## [1] 7110
```

After deleting some observations, we are going to convert variables to appropriate data types and also add a new variable called 'distance', which calculates the distance between an observation's location and the epicenter of Beijing, which turns out to be 'Jingshan park'.

'Distance'

```
# Distance from Jingshan Park.
# Output in meters.
#library(geosphere)

# For loop to iterate
#for (i in 1:nrow(df)) {
#park_lng = 116.3966463362935
#park_lat = 39.92600108466268
  #df$distance[i] = distVincentyEllipsoid(c(park_lng, park_lat), c(df$Lng[i], df$Lat[i]))
#}

# Save the csv file
#write.csv(df, "C:\\Users\\jsyang354\\Desktop\\STAT 432 - Basic Stat Learning\\Final Project\\housing_c
```

Read the edited data

```
df = read.csv("house_cluster.csv")
#head(df)
```

Convert data type

```r
# Convert into numeric/factor types
df$livingRoom = as.numeric(df$livingRoom)
df$drawingRoom = as.numeric(df$drawingRoom)
df$kitchen = as.numeric(df$kitchen)
df$bathRoom = as.numeric(df$bathRoom)
df$buildingType = as.factor(df$buildingType)
df$constructionTime = as.numeric(df$constructionTime)
df$renovationCondition = as.factor(df$renovationCondition)
df$buildingStructure = as.factor(df$buildingStructure)
df$elevator = as.factor(df$elevator)
df$fiveYearsProperty = as.factor(df$fiveYearsProperty)
df$subway = as.factor(df$subway)
df$district = as.factor(df$district)

# Double check the converted data
#head(df, 3)
```

Clustering Analysis with K-Means

Unsupervised learning is a type of machine learning techniques with a purpose of identifying or grouping data. Since there is no pre-existing label, our goal is not to classify or predict, but rather try to understand how the data is formed and could be “clustered”. There are many different clustering algorithms we can use, and some famous methods are K-Means and hierarchical clustering.

We will skip explaining about the details of how these algorithms work for right now, but the basic concept for both of them would be calculating the distance between observations. This way, we can figure out how close or far away a data point is to another. Considering the data we are using is large and complex (fairly high dimensional data), it is inevitable of facing some critical issues such as expensive computational cost and limitation of our machines (We did try fitting the raw data to these algorithms naively, and it did not work with R session aborted).

Therefore, one alternative we are going to use is to reduce the size of the data we would like to use for clustering analysis. Based on the regression part in the beginning, we were able to figure out some important variables that are likely to be closely related and affecting the response variable, ‘price’, and we would like to test whether these variables would also reveal some association with ‘price’ in clustering analysis as well.

One way to reduce the size of the data is simple random sample, meaning that we are going to randomly sample small proportion of the data with the chosen variable. The chose variables in this case are:

- Top 10 variables considered to be important based on the regression analysis

1. DOM
2. livingRoom
3. drawingRoom
4. bathRoom
5. constructionTime
6. renovationCondition
7. fiveyearsProperty
8. communityAverage
9. floorNumber
10. distance

Another data preprocessing for KMeans

Due to some special features of hierarchical clustering such as more complex computation, it is not recommended to apply it to our data. As a result, Kmeans clustering would be the most appropriate algorithm in this case.

One thing we need to be aware of is that the basic concept or computation of Kmeans is the 'Euclidean' distance, and therefore it is essential that all variable should be numeric. Although the data is technically numeric, there are some variables like 'fiveyearsProperty' which represent whether an owner has the property for less than 5 years or not (0 - no, 1 - yes), and doing arithmetic calculation does not make sense. Thus, we need to find some other way that would allow us to convert and calculate the Euclidean distance.

Plus, the algorithm is affected by the scale of a variable, meaning that one variable with large scale could possibly dominate over other variables with relatively small scale. Unless if we want to spotlight some specific variables, it is recommended to scale all variables to have same size (Typically with a mean of 0 and standard deviation 1).

Methods to handle mixed data

There are several ways to manipulate mixed (numeric + categorical) data to be suitable for clustering analysis, and we are thinking of two different methods:

1. Gower's distance ('daisy' or StatMatch' package)
2. One-hot encoding

One way to handle this issue is 'Gower's distance', and the logic behind this method is to calculate the 'dissimilarity' between non-numeric variables, which allows us to convert non-numeric variables (categorical, factor…) into numeric variables. However, the main issue with converting these non-numeric variables is that we are losing some information and it cannot reflect somewhat 'nuanced' similarities between observations.

Some relevant links

- https://cran.r-project.org/web/packages/gower/vignettes/intro.pdf
- https://medium.com/analytics-vidhya/clustering-on-mixed-data-types-in-python-7c22b3898086

Another method we can try is using one-hot encoding, which converts a factor variable into several dummy variables (0 and 1), but considering that the column is going to be expanded based on the number of levels it has, the curse of dimensionality can be problematic. Plus, there is still a question about calculating the distance of dummy variables.

Some relevant links

- https://stackoverflow.com/questions/56171837/kmodes-vs-one-hot-encoding-kmeans-for-categorical-data
- https://medium.com/analytics-vidhya/clustering-on-mixed-data-types-in-python-7c22b3898086

These methods have both pros and cons, and based on the comparison, conversion with Gower's distance would be the most appropriate method.

Data Preprocessing 2 - Clustering

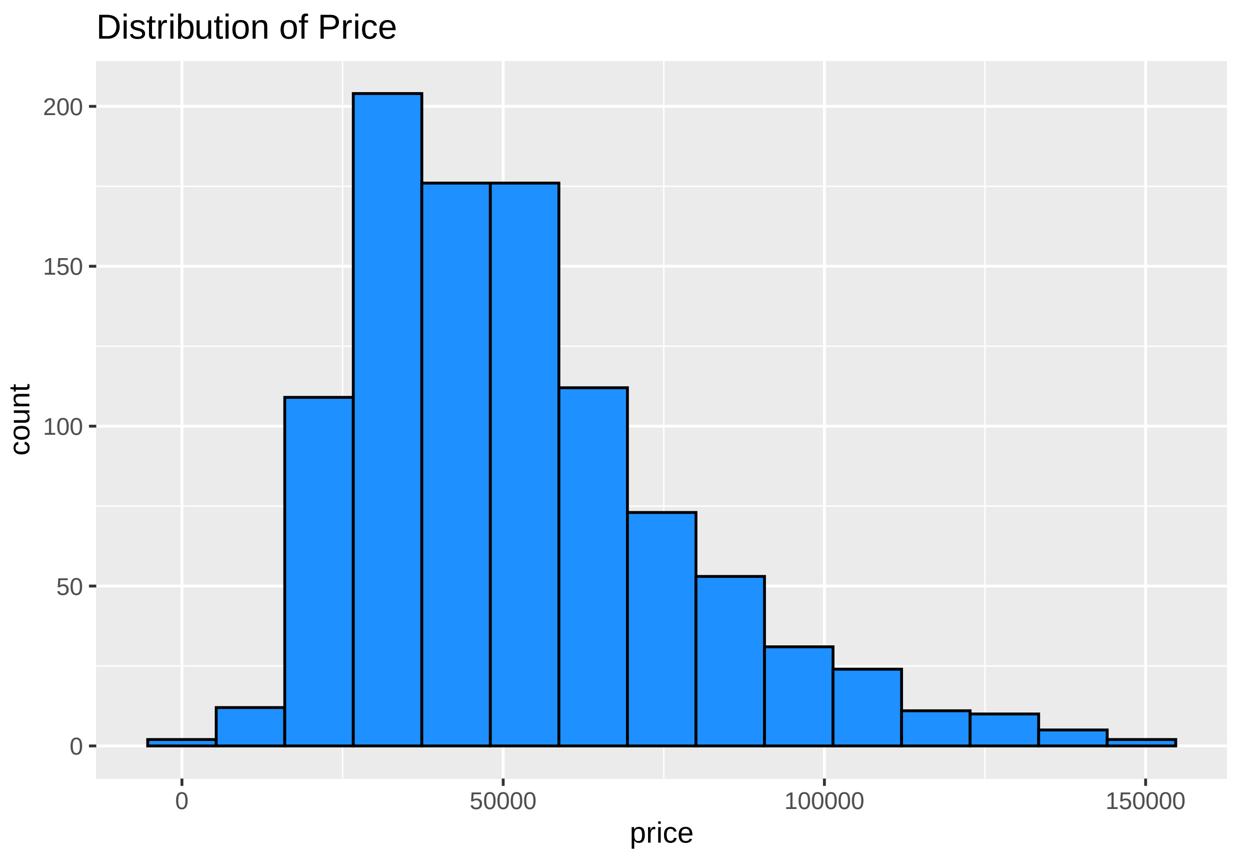
```
# Simple random sample of 1000 (tried 5000 and 10000, but the session aborted)
set.seed(432)
sample = df[sample(nrow(df), 1000),]
#head(sample)

# Subset the dataset to only contain top 10 variables + 'price' for later additional analysis
df.clst = subset(sample, select = c(price, DOM, livingRoom, drawingRoom, bathRoom, constructionTime, re
# head(df.clst)
```

Exploratory Data Analysis (EDA)

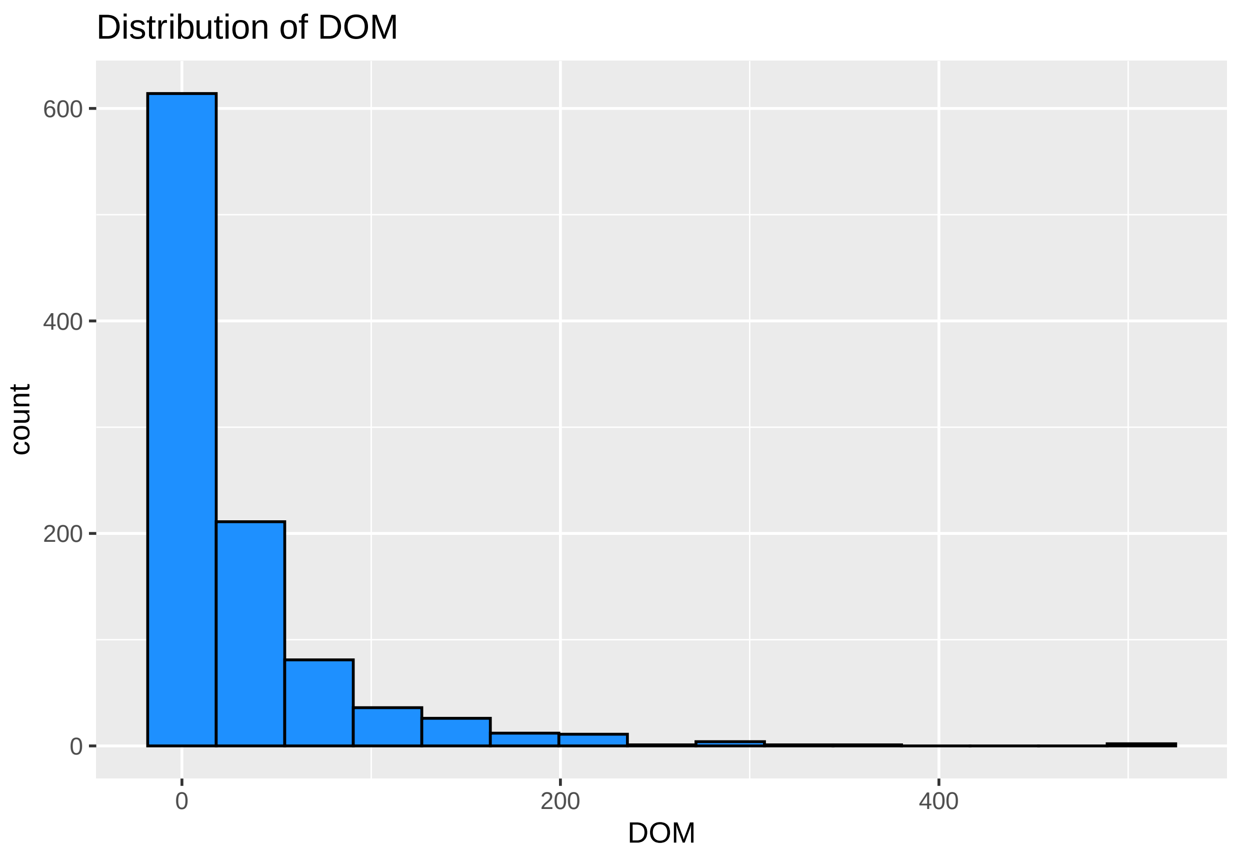
```
# Summary Statistics
summary(df.clst)
```

```
##      price             DOM           livingRoom    drawingRoom
##  Min.   :     2   Min.   :  1.00   Min.   :1.00   Min.   :0.00
##  1st Qu.: 33878   1st Qu.:  1.00   1st Qu.:1.00   1st Qu.:1.00
##  Median : 47730   Median :  8.00   Median :2.00   Median :1.00
##  Mean   : 52014   Mean   : 30.11   Mean   :2.02   Mean   :1.14
##  3rd Qu.: 65162   3rd Qu.: 38.00   3rd Qu.:2.00   3rd Qu.:1.00
##  Max.   :149353   Max.   :508.00   Max.   :7.00   Max.   :3.00
##     bathRoom     constructionTime renovationCondition fiveYearsProperty
##  Min.   :0.000   Min.   :1955     1:219               0:368
##  1st Qu.:1.000   1st Qu.:1993     2: 17               1:632
##  Median :1.000   Median :2001     3:313
##  Mean   :1.184   Mean   :1999     4:451
##  3rd Qu.:1.000   3rd Qu.:2005
##  Max.   :4.000   Max.   :2015
##  communityAverage  floorNumber       distance
##  Min.   : 24224   Min.   : 3.00   Min.   : 1599
##  1st Qu.: 46505   1st Qu.: 6.00   1st Qu.: 6904
##  Median : 58488   Median :11.00   Median :10596
##  Mean   : 63400   Mean   :12.97   Mean   :12444
##  3rd Qu.: 75388   3rd Qu.:18.00   3rd Qu.:17114
##  Max.   :183109   Max.   :32.00   Max.   :37472
```

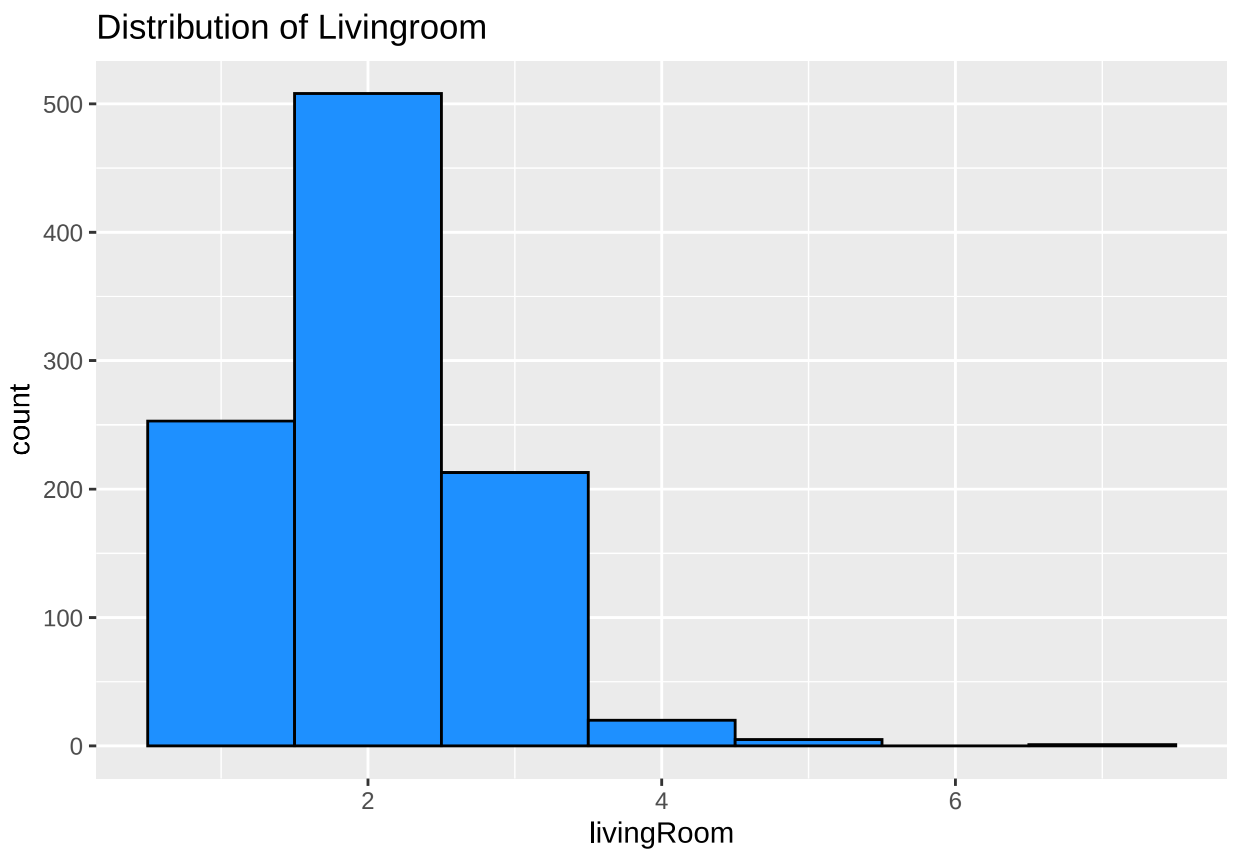
```
# Visualizations of Top 10 predictors
library(ggplot2)

ggplot(df.clst, aes(x=price)) +
  geom_histogram(bins = 15, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Price")
```



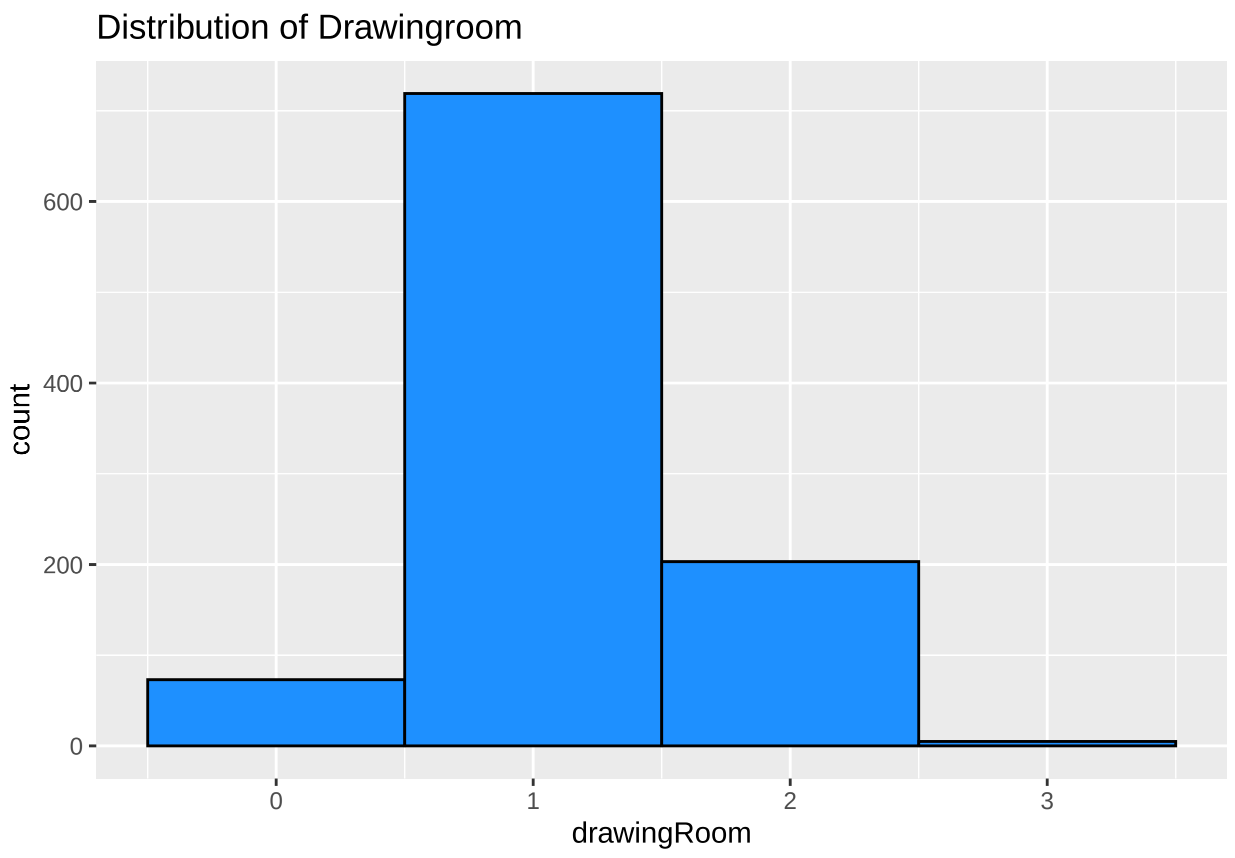
```
ggplot(df.clst, aes(x=DOM)) +
  geom_histogram(bins = 15, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of DOM")
```



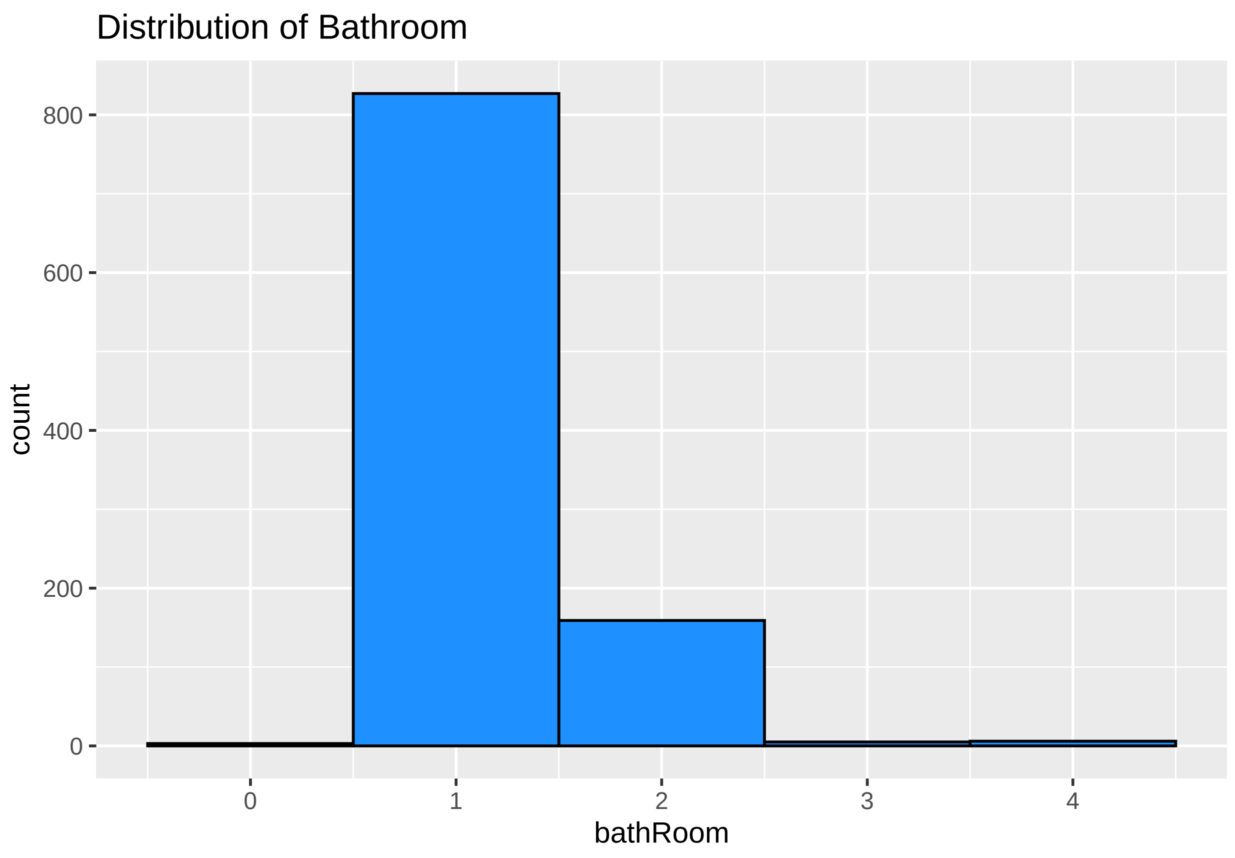
```
ggplot(df.clst, aes(x=livingRoom)) +
  geom_histogram(binwidth = 1, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Livingroom")
```



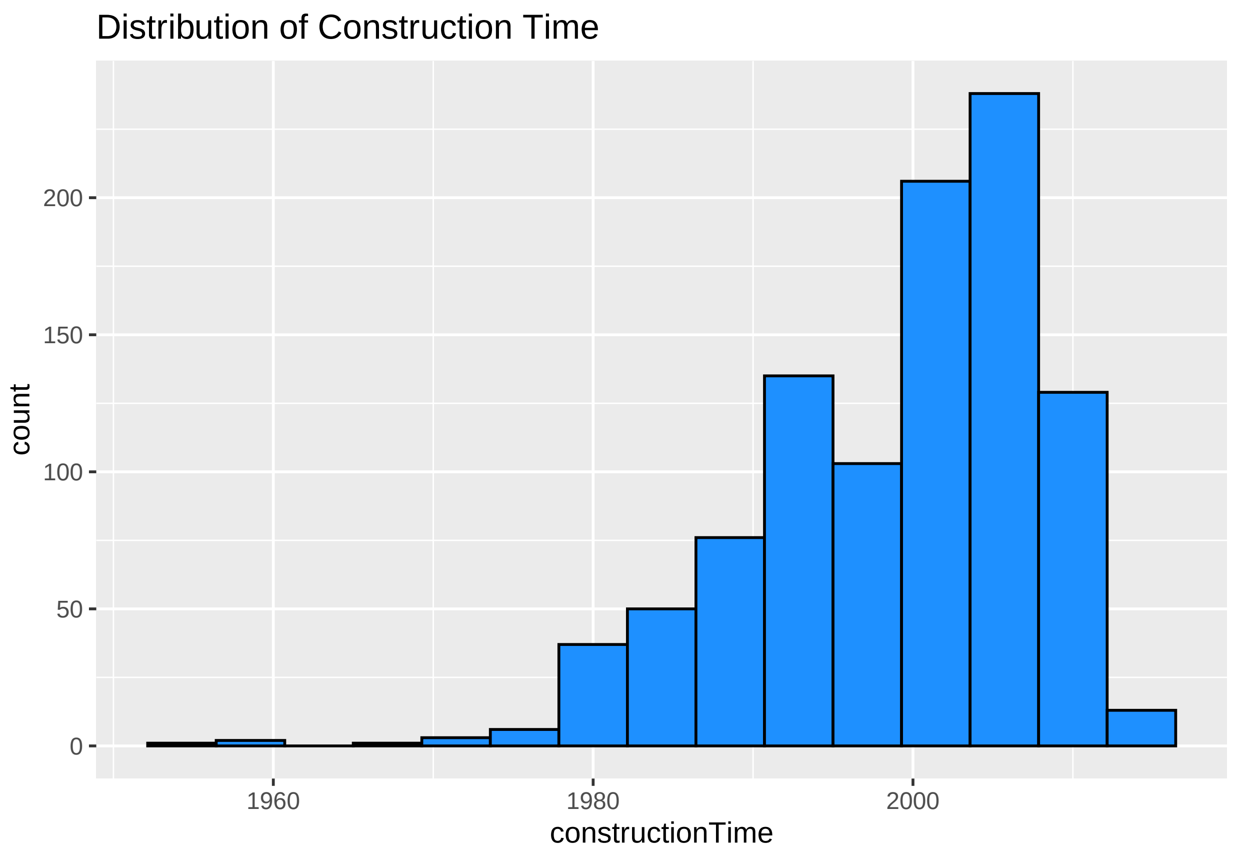
```
ggplot(df.clst, aes(x=drawingRoom)) +
  geom_histogram(binwidth = 1, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Drawingroom")
```



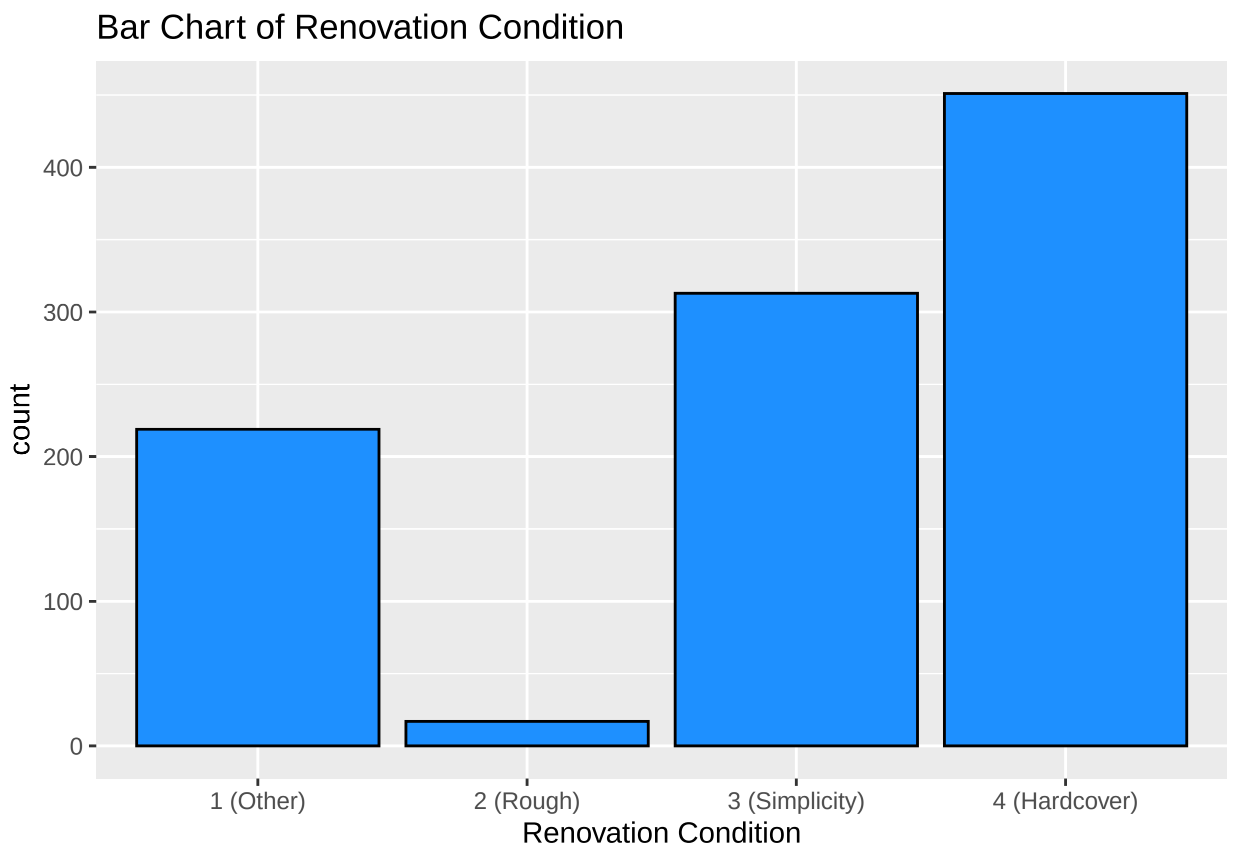
```
ggplot(df.clst, aes(x=bathRoom)) +
  geom_histogram(binwidth = 1, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Bathroom")
```



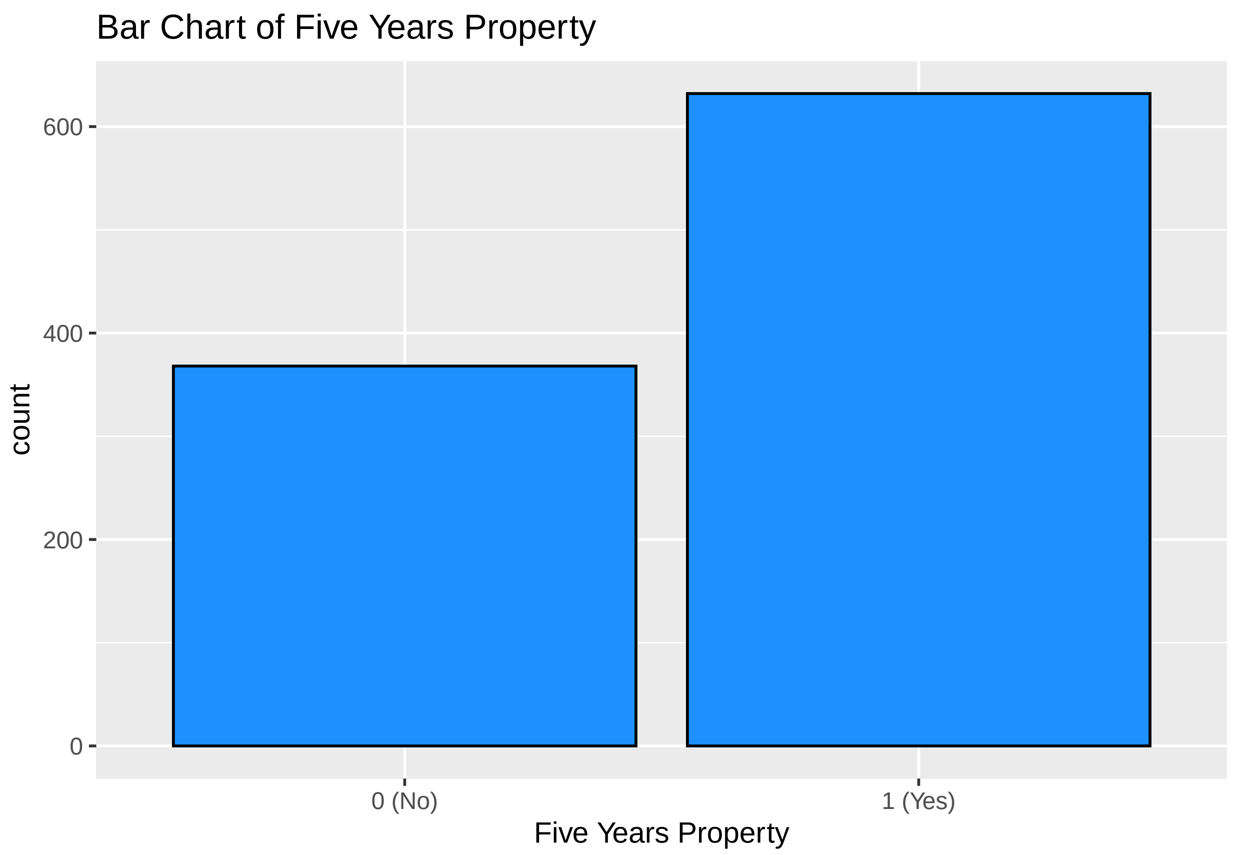
```
ggplot(df.clst, aes(x=constructionTime)) +
  geom_histogram(bins = 15, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Construction Time")
```



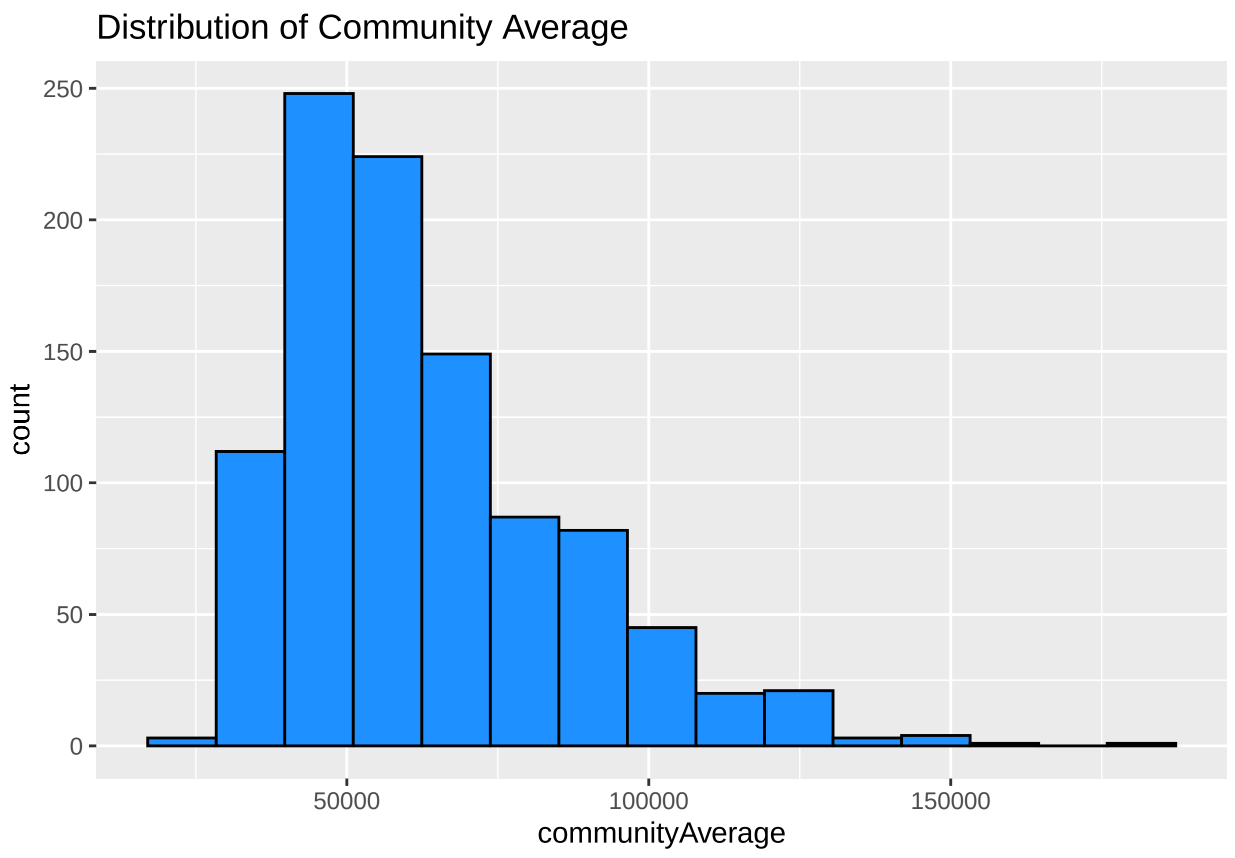
```
ggplot(df.clst, aes(x=renovationCondition)) +
  geom_bar(fill='dodgerblue', col = 'black') +
  scale_x_discrete(labels=c('1 (Other)', '2 (Rough)', '3 (Simplicity)', '4 (Hardcover)')) +
  labs(x="Renovation Condition", title="Bar Chart of Renovation Condition")
```



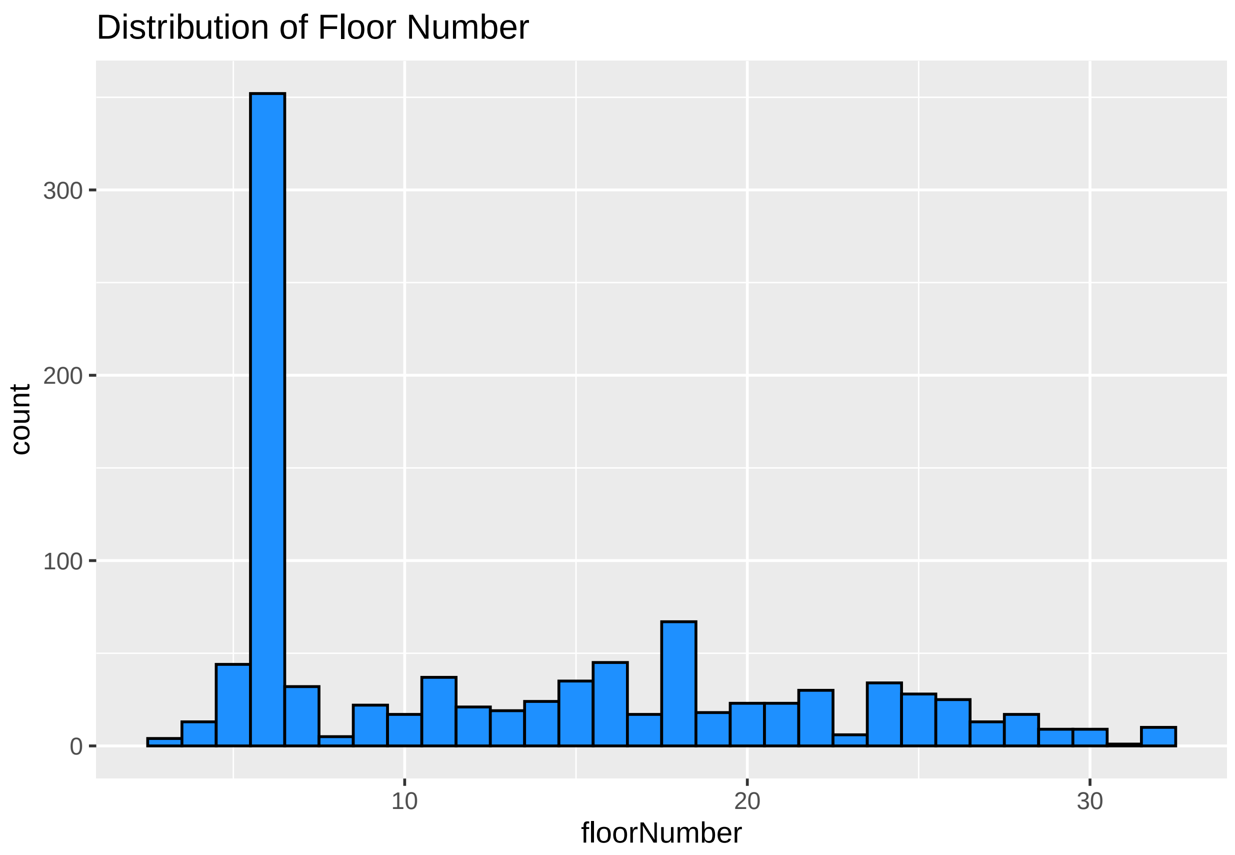
```
ggplot(df.clst, aes(x=fiveYearsProperty)) +
  scale_x_discrete(labels=c('0 (No)', '1 (Yes)')) +
  geom_bar(fill='dodgerblue', col = 'black') +
  labs(x="Five Years Property", title="Bar Chart of Five Years Property")
```



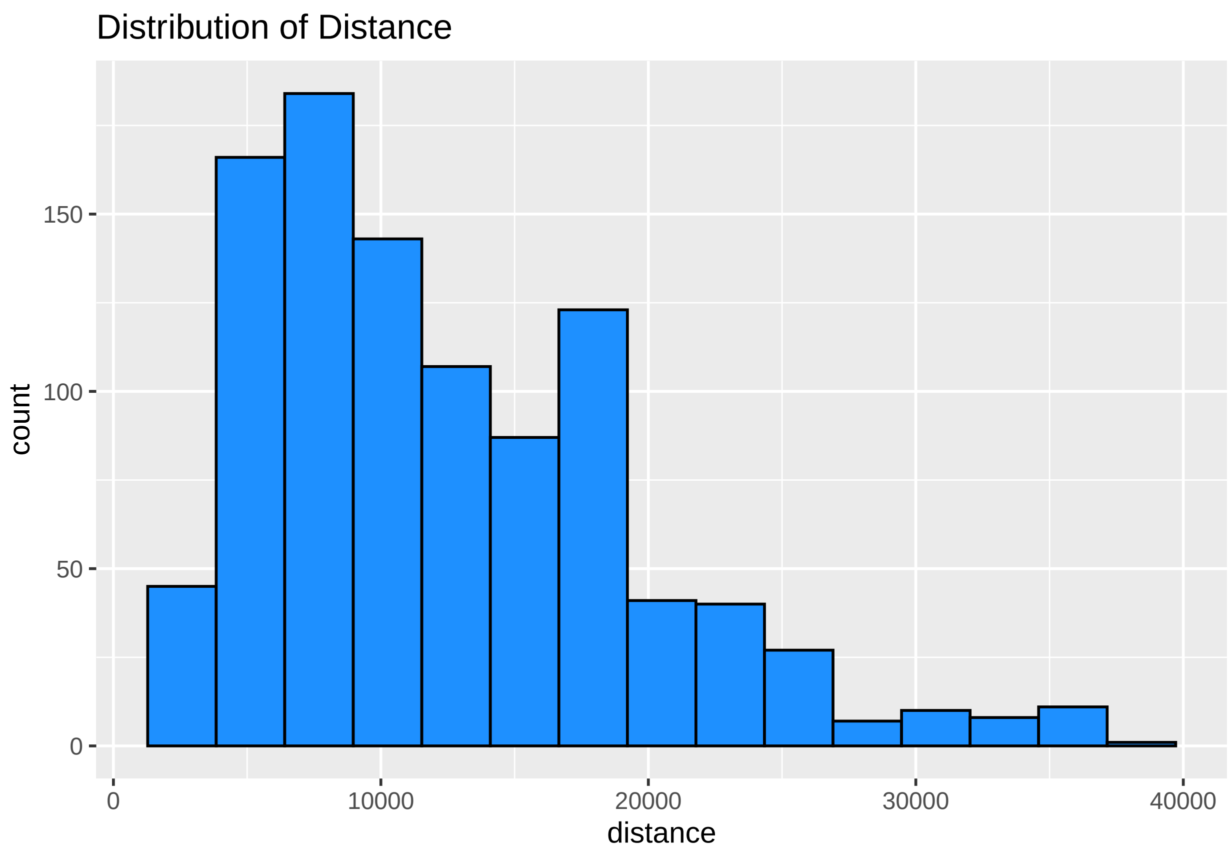
```r
ggplot(df.clst, aes(x=communityAverage)) +
  geom_histogram(bins = 15, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Community Average")
```



```
ggplot(df.clst, aes(x=floorNumber)) +
  geom_histogram(binwidth = 1, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Floor Number")
```



```
ggplot(df.clst, aes(x=distance)) +
  geom_histogram(bins = 15, fill = "dodgerblue", col = "black") +
  labs(title="Distribution of Distance")
```



Scale the data

```
# Remove 'price' variable for clustering steps
df.scale = df.clst[, -c(1)]

# Scale only the numeric variables
df.scale[, -c(6,7)] = scale(df.scale[, -c(6,7)])
#head(df.scale)


# Check mean and standard deviation of every column
colMeans(df.scale[, -c(6,7)])
```

```
##              DOM       livingRoom      drawingRoom         bathRoom
##     5.288651e-17    -2.498002e-18     1.643685e-16     1.080247e-16
## constructionTime communityAverage      floorNumber         distance
##    -4.733538e-15    -1.307773e-16    -1.776357e-18     2.326394e-17
```

```
sapply(df.scale[, -c(6,7)], sd)
```

```
##              DOM       livingRoom      drawingRoom         bathRoom
##                1                1                1                1
## constructionTime communityAverage      floorNumber         distance
##                1                1                1                1
```

Gower's distance

```
# 'StatMatch' package to convert the into gower's distance
library(StatMatch)
```

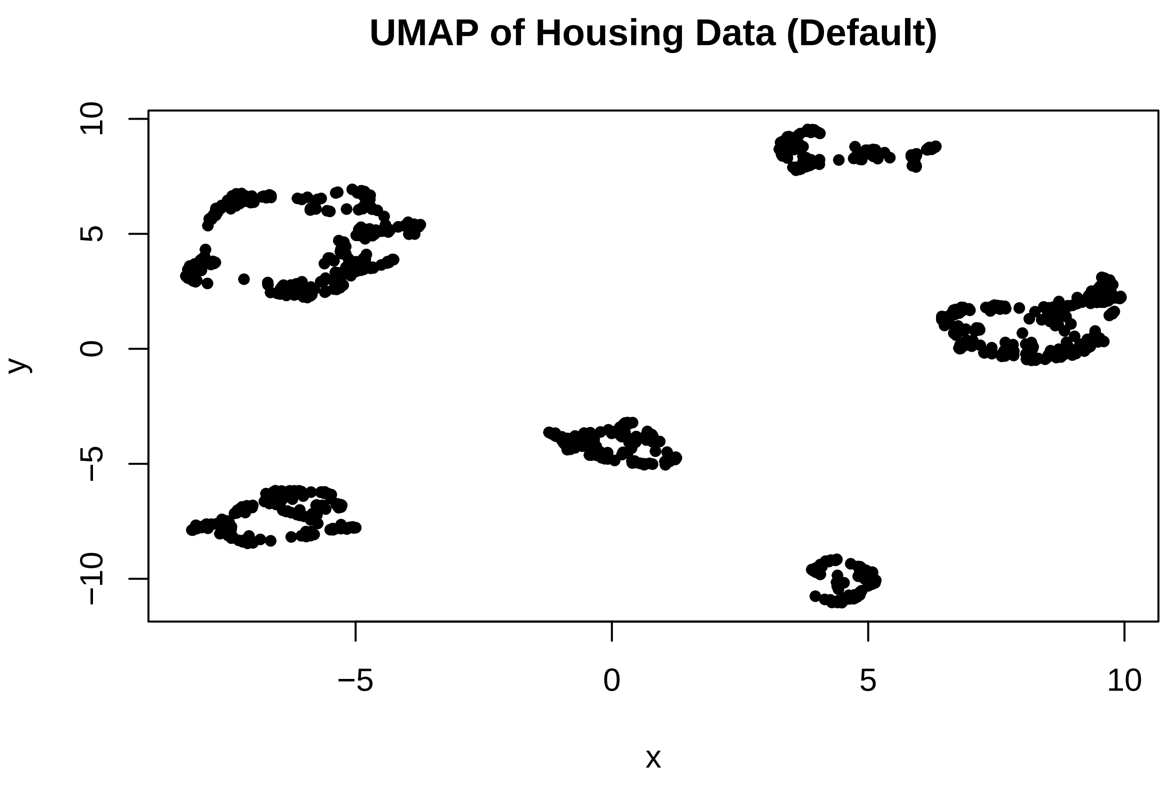
```
## Loading required package: proxy

##
## Attaching package: 'proxy'

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix

## Loading required package: survey

## Loading required package: grid

## Loading required package: Matrix

## Loading required package: survival

##
## Attaching package: 'survey'

## The following object is masked from 'package:graphics':
##
##     dotchart

## Loading required package: lpSolve
```

```
df.gower = gower.dist(df.scale)
```

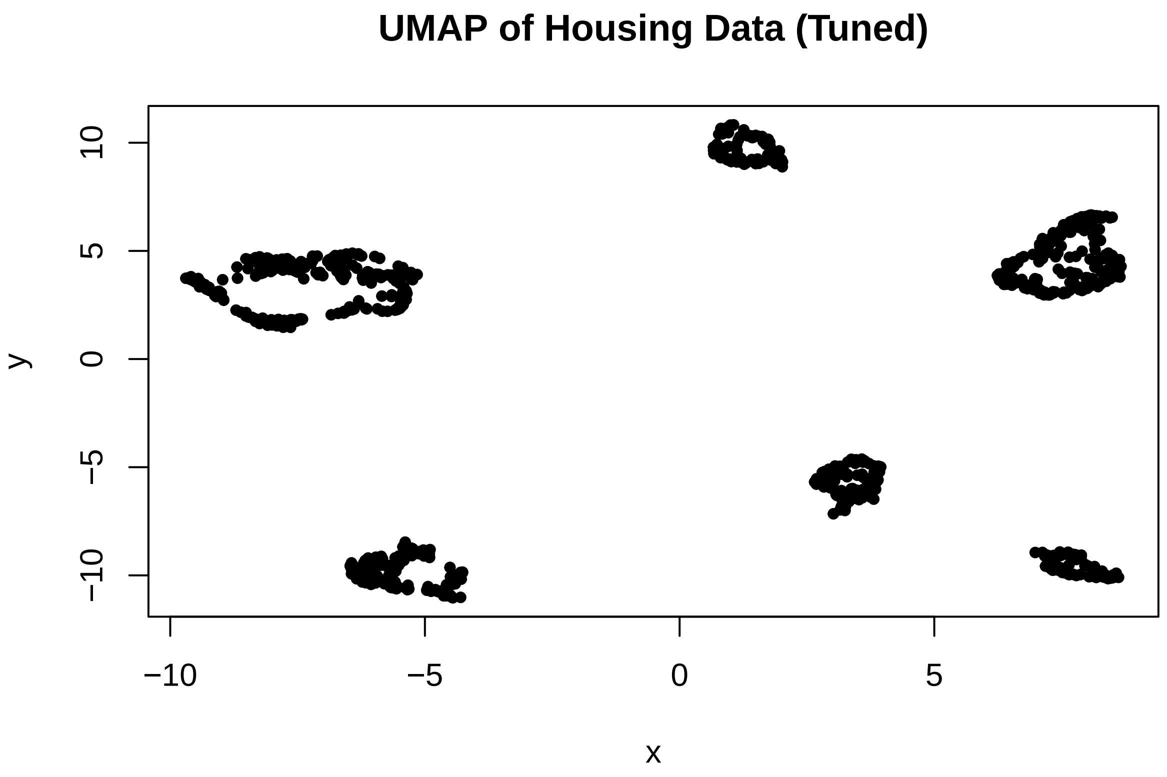
This can be done with 'daisy' package as well.

Clustering Visualization

```
# Map the data
library(umap)

# Default parameters
set.seed(432)
df.umap = umap(df.gower)
plot(df.umap$layout, xlab = "x", ylab = "y", pch = 20, main = "UMAP of Housing Data (Default)")
```



```r
# Personal tuning - increase the number of neighbor points
myumap.tuning = umap.defaults
myumap.tuning$n_neighbors = 30

set.seed(432)
df.myumap = umap(df.gower, myumap.tuning)
plot(df.myumap$layout, xlab = "x", ylab = "y", pch = 20, main = "UMAP of Housing Data (Tuned)")
```



There are several ways to determine the optimal number of clusters in the data, and one way is to display the data using 'UMAP'. The 'UMAP' algorithm allows to get a brief idea of how the data is formed. It is one of the popular dimensionality reduction algorithms which project high dimensional data into lower dimension. One thing to remind is that the x and y axis do not represent the actual distance between observations.

Based on the umap visualization with default parameters value, it seems that there would be 6 different clusters in the data. When tuning the number of neighbor points to 30, the visualization still displays 6 well separated groups.

We would want to check the optimal number of clusters with an 'elbow' method as well to compare whether the number of clusters matches with the umap above.

Relevant link

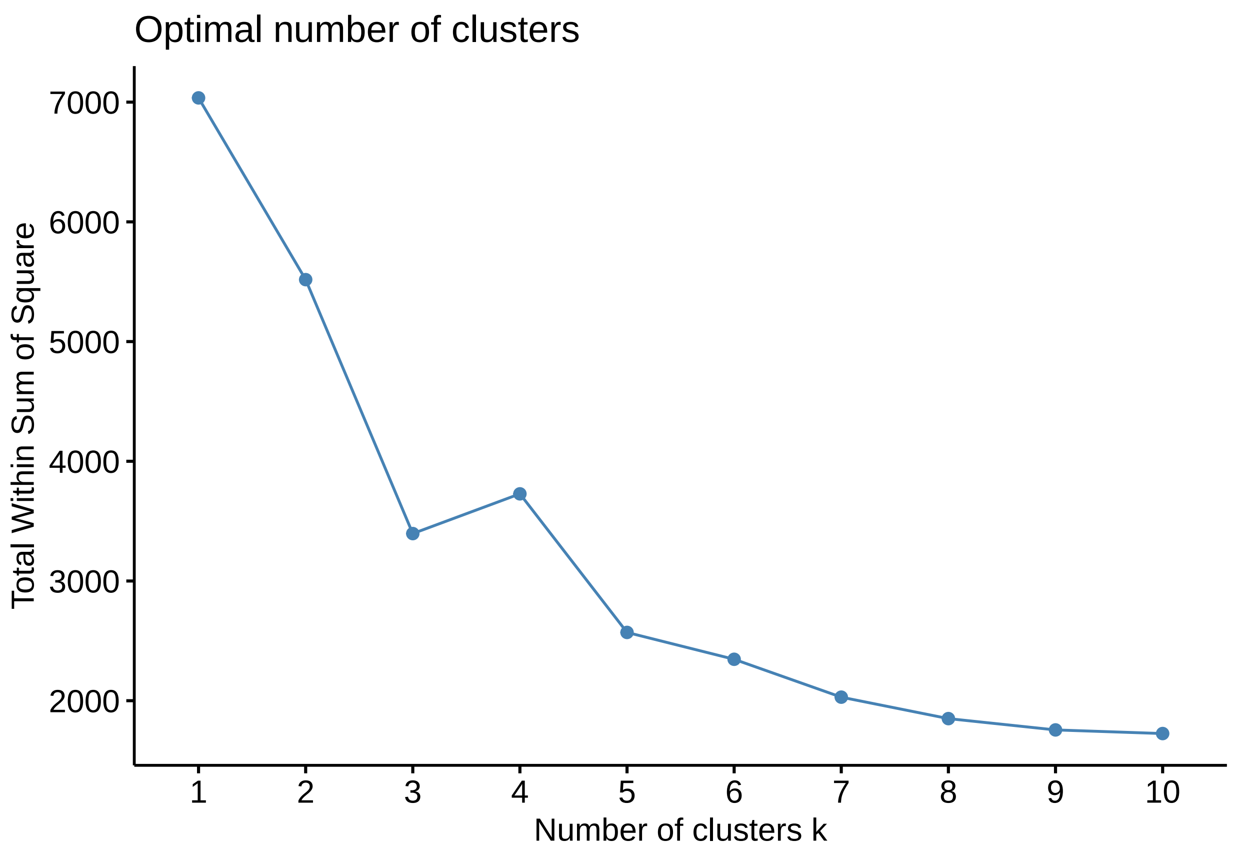
- https://www.statology.org/elbow-method-in-r/
- https://uc-r.github.io/kmeans_clustering

Elbow plot

```
library(cluster)
library(factoextra)
```

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

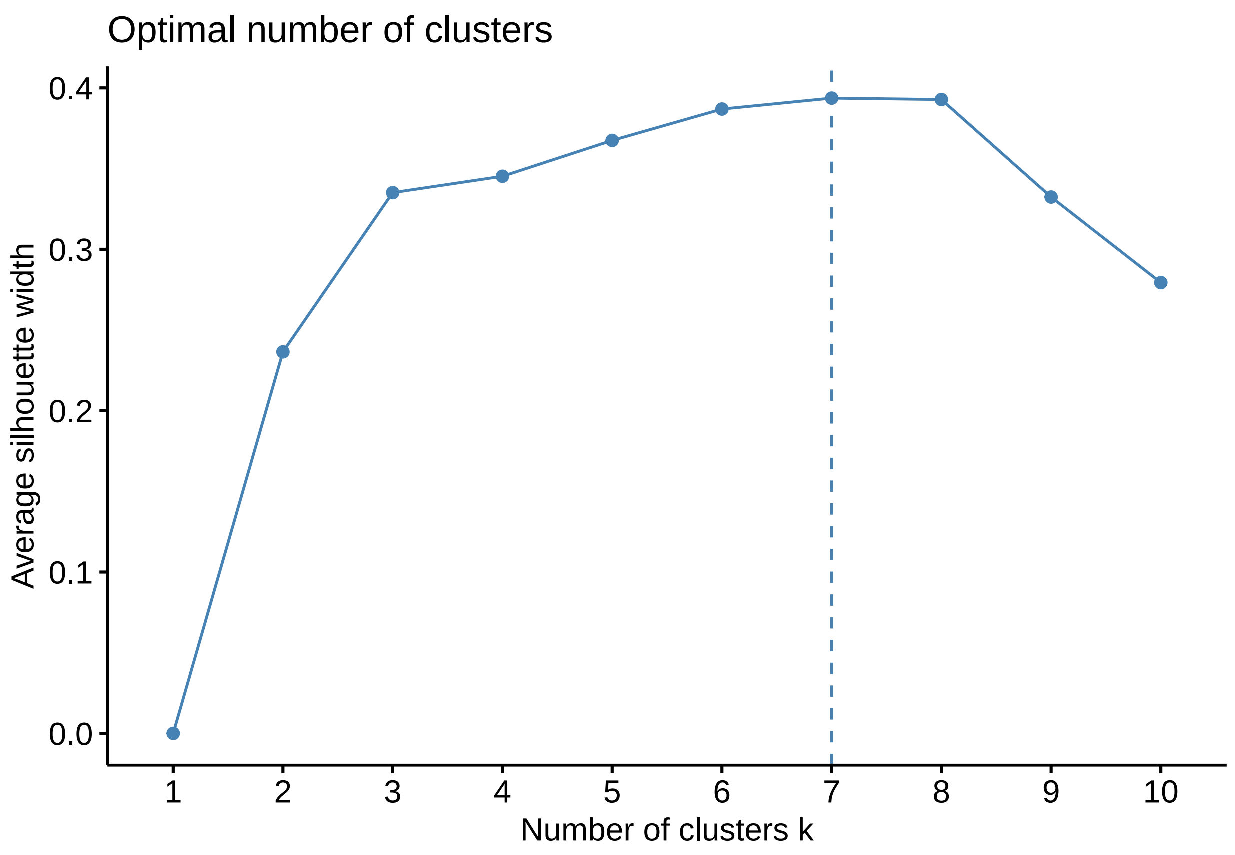
```
#create plot of number of clusters vs total within sum of squares
fviz_nbclust(df.gower, kmeans, method = "wss")
```



The main computation behind 'Elbow' method is to calculate the variation (total sum of square) within a cluster. Based on the elbow plot above, we can figure out that there is a drastic drop from k=1 to k=3 and somewhat spikes up at k=4. Starting from k=5, the total within sum of square value smoothly levels off. In general, the location of a bend (elbow) in the plot is considered as an indicator of the appropriate number of clusters. Based on the plot above, the number of clusters around k=5 seems to be the most optimal number of clusters, which is pretty close to the umap visualization.

Silhouette Score

```r
fviz_nbclust(df.gower, kmeans, method = "silhouette")
```



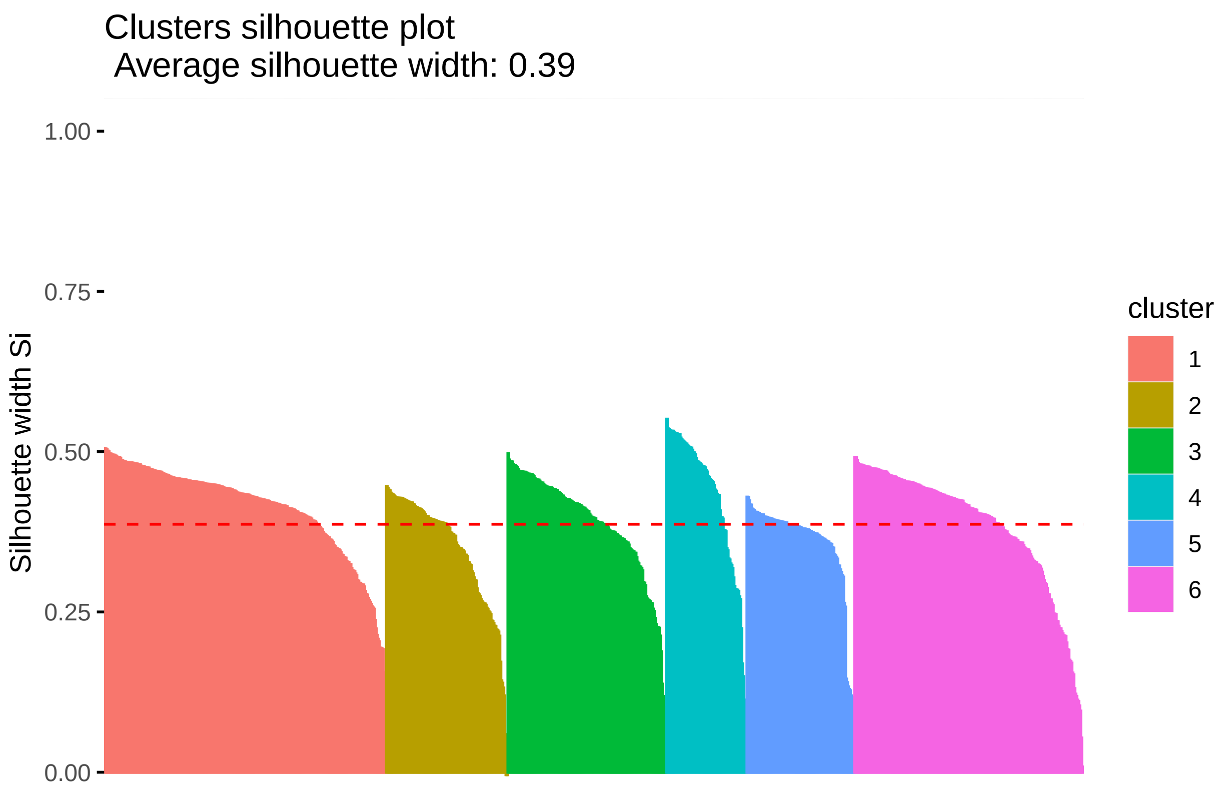
'Silhouette score' is another method to determine the optimal number of clusters. It measures how well the observations in data are cohesive within their cluster and separated from other clusters. Higher average silhouette score indicates good clustering formation whereas low silhouette score would be a signal of bad (widely spread out, vague separation) clustering. With the silhouette plot created above, it suggests the best number of clusters in this data to be k=7.

To sum up, all methods introduced above suggested similar number of clusters (k=5,6,7), and the optimal number based on our perspective would be k=6.

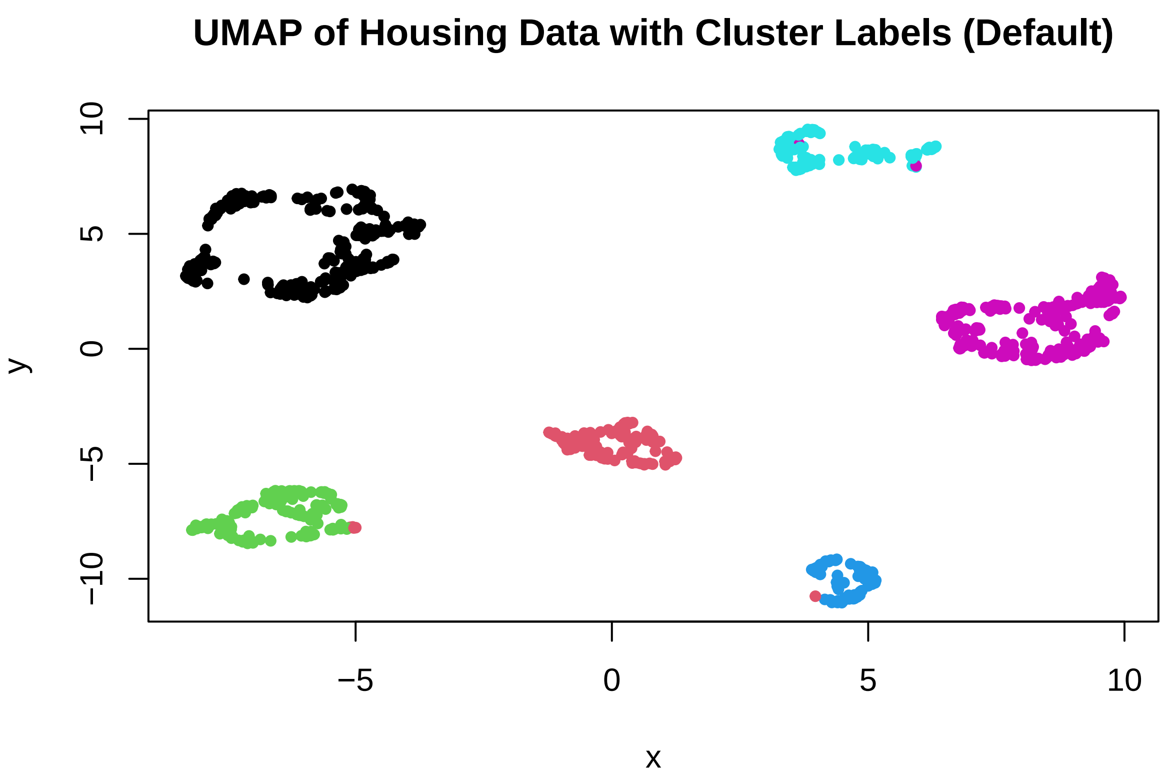
Color the clusters by KMeans labels

```
# Final decision - 6 clusters
set.seed(432)
final = kmeans(df.gower, 6, nstart = 25)
sil = silhouette(final$cluster, dist(df.gower))
fviz_silhouette(sil)
```

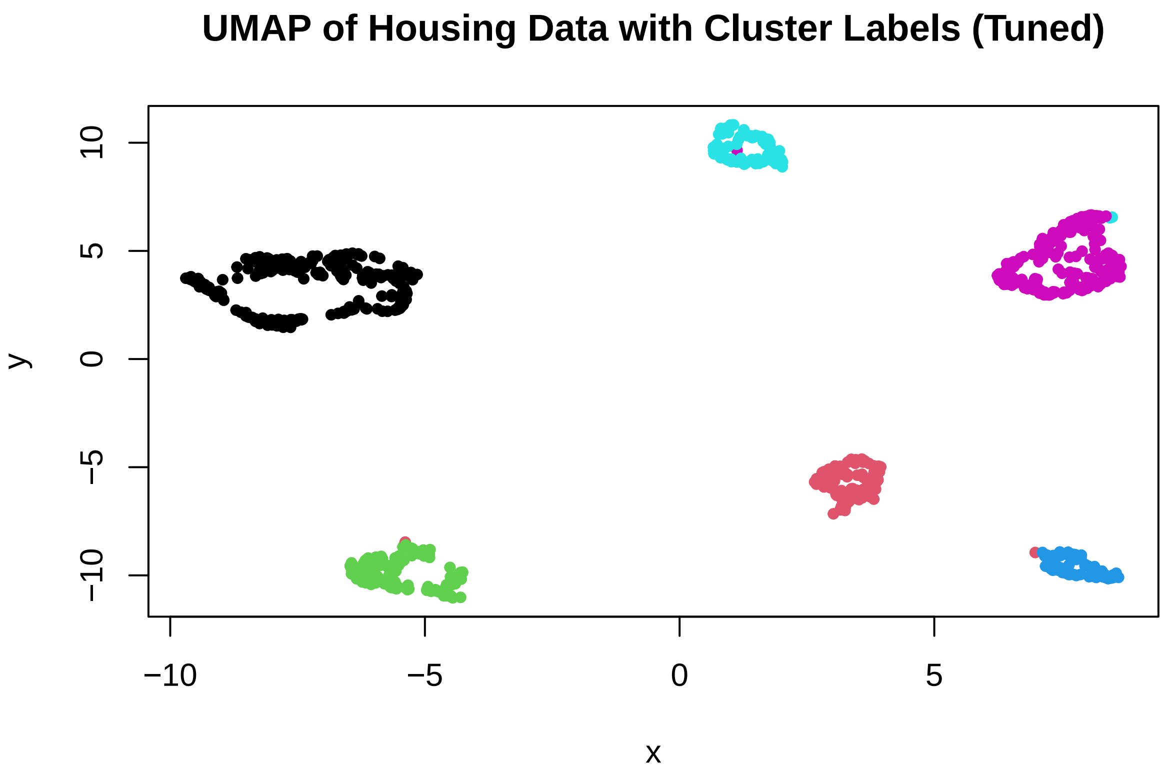
```
##   cluster size ave.sil.width
## 1       1  288          0.41
## 2       2  124          0.34
## 3       3  162          0.39
## 4       4   82          0.43
## 5       5  110          0.36
## 6       6  234          0.38
```



```
# Convert cluster labels into factor & create a new column to the original data
final$cluster = as.factor(final$cluster)
df.clst$clst_label = final$cluster

# Plot the UMAP again to check how the cluster labels are colored
set.seed(432)
df.umap = umap(df.gower)
plot(df.umap$layout, xlab = "x", ylab = "y", pch = 20,
     main = "UMAP of Housing Data with Cluster Labels (Default)", col = df.clst$clst_label)
```



```
#legend("bottomright", legend = c(1,2,3,4,5,6), col = final$cluster)

set.seed(432)
df.myumap = umap(df.gower, myumap.tuning)
plot(df.myumap$layout, xlab = "x", ylab = "y", pch = 20,
     main = "UMAP of Housing Data with Cluster Labels (Tuned)", col = df.clst$clst_label)
```



'Average Silhouette' plot shows how cohesive a cluster is, and the range of width (score) is between -1 and 1. The cluster is well cohesive if the width is positive and closer to 1 whereas low and negative score indicates that the cluster is not cohesive and there might be some observations in the cluster that are actually closer to other clusters. Based on the plot, the average silhouette score is 0.39, which is somewhat mild.

The last two plots are the same UMAP visualizations made in the beginning of the cluster analysis above with the colors based on the cluster labels. Except for few observations, all clusters are colored what we have expected.

Exploratory Data Analysis (EDA) again based on cluster labels

```
library(dplyr)
```

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

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

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

```
# Count the number of observations by cluster labels
df.clst %>% count(clst_label)
```

```
##   clst_label   n
## 1          1 288
## 2          2 124
## 3          3 162
## 4          4  82
## 5          5 110
## 6          6 234
```

```
# Summary Statistics by clusters
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.price = min(price), q1.price = quantile(price, 0.25),
            median.price = median(price), mean.price = mean(price),
            q3.price = quantile(price, 0.75), max.price = max(price))
```

```
## # A tibble: 6 x 7
##   clst_label min.price q1.price median.price mean.price q3.price max.price
##   <fct>          <int>    <dbl>        <dbl>      <dbl>    <dbl>     <int>
## 1 1              11483    34823        49207     52585.   65870.    141096
## 2 2               7888    27487        38396.    45014.   57512     142858
## 3 3              18241    38783        53358     57620.   73262     149353
## 4 4              19017    39066.       50764.    56849.   64743.    144452
## 5 5                  2    28448.       36934.    42982.   49387     139690
## 6 6              15901    35758        48128.    53690.   69268.    131368
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.DOM = min(DOM), q1.DOM = quantile(DOM, 0.25),
            median.DOM = median(DOM), mean.DOM = mean(DOM),
            q3.DOM = quantile(DOM, 0.75), max.DOM = max(DOM))
```

```
## # A tibble: 6 x 7
##   clst_label min.DOM q1.DOM median.DOM mean.DOM q3.DOM max.DOM
##   <fct>        <int>  <dbl>      <dbl>    <dbl>  <dbl>   <int>
## 1 1                1   1          10.5    31.3    41.2     495
## 2 2                1   1           1      17.8    10       300
## 3 3                1   3.25       20      36.7    45.8     316
## 4 4                1   5.25       17.5    40.0    57.2     201
## 5 5                1   1           1       8.72    1       140
## 6 6                1   1          12.5    37.2    46.8     508
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.liv = min(livingRoom), q1.liv = quantile(livingRoom, 0.25),
            median.liv = median(livingRoom), mean.liv = mean(livingRoom),
            q3.liv = quantile(livingRoom, 0.75), max.liv = max(livingRoom))
```

```
## # A tibble: 6 x 7
##   clst_label min.liv q1.liv median.liv mean.liv q3.liv max.liv
##   <fct>        <dbl>  <dbl>      <dbl>    <dbl>  <dbl>   <dbl>
## 1 1                1      2          2     2.03      2       4
## 2 2                1      1          2     2.02      2       7
## 3 3                    1    1              2     1.96      2       5
## 4 4                    1    1              2     1.89      2       4
## 5 5                    1    1.25           2     2.05      3       4
## 6 6                    1    2              2     2.08      2       5
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.draw = min(drawingRoom), q1.draw = quantile(drawingRoom, 0.25),
            median.draw = median(drawingRoom), mean.draw = mean(drawingRoom),
            q3.draw = quantile(drawingRoom, 0.75), max.draw = max(drawingRoom))
```

```
## # A tibble: 6 x 7
##   clst_label min.draw q1.draw median.draw mean.draw q3.draw max.draw
##   <fct>         <dbl>   <dbl>       <dbl>     <dbl>   <dbl>    <dbl>
## 1 1                 0       1           1      1.19       2        3
## 2 2                 0       1           1      1.21       2        3
## 3 3                 0       1           1      1.10       1        3
## 4 4                 0       1           1      1.06       1        3
## 5 5                 0       1           1      1.16       1        2
## 6 6                 0       1           1      1.08       1        2
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.bath = min(bathRoom), q1.bath = quantile(bathRoom, 0.25),
            median.bath = median(bathRoom), mean.bath = mean(bathRoom),
            q3.bath = quantile(bathRoom, 0.75), max.bath = max(bathRoom))
```

```
## # A tibble: 6 x 7
##   clst_label min.bath q1.bath median.bath mean.bath q3.bath max.bath
##   <fct>         <dbl>   <dbl>       <dbl>     <dbl>   <dbl>    <dbl>
## 1 1                 0       1           1      1.21       1        3
## 2 2                 1       1           1      1.22       1        4
## 3 3                 0       1           1      1.22       1        4
## 4 4                 1       1           1      1.09       1        3
## 5 5                 1       1           1      1.17       1        3
## 6 6                 0       1           1      1.15       1        4
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.const = min(constructionTime), q1.const = quantile(constructionTime, 0.25),
            median.const = median(constructionTime), mean.const = mean(constructionTime),
            q3.const = quantile(constructionTime, 0.75), max.const = max(constructionTime))
```

```
## # A tibble: 6 x 7
##   clst_label min.const q1.const median.const mean.const q3.const max.const
##   <fct>          <dbl>    <dbl>        <dbl>      <dbl>    <dbl>     <dbl>
## 1 1               1973     1996         2003      2000.     2006      2010
## 2 2               1973     1999.        2004      2003.     2009      2015
## 3 3               1955     1996         2004      2002.     2009      2015
## 4 4               1975     1989         1996.     1996.     2003      2013
## 5 5               1979     1992         2000      1998.     2004      2010
## 6 6               1958     1991         1997      1996.     2003      2010
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.comm = min(communityAverage), q1.comm = quantile(communityAverage, 0.25),
            median.comm = median(communityAverage), mean.comm = mean(communityAverage),
            q3.comm = quantile(communityAverage, 0.75), max.comm = max(communityAverage))
```

```
## # A tibble: 6 x 7
##   clst_label min.comm q1.comm median.comm mean.comm q3.comm max.comm
##   <fct>         <int>   <dbl>       <dbl>     <dbl>   <dbl>    <int>
## 1 1             30167  45447        58332    63337.  75091    183109
## 2 2             30795  47359        57305    62203.  70772    145581
## 3 3             24224  47968.       59020    62978.  75554    152118
## 4 4             28609  46403        56973    62825.  68904.   155145
## 5 5             31650  46930.       59355    63468.  71822.   138746
## 6 6             28039  46378.       59680    64572.  81430.   128939
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.floor = min(floorNumber), q1.floor = quantile(floorNumber, 0.25),
            median.floor = median(floorNumber), mean.floor = mean(floorNumber),
            q3.floor = quantile(floorNumber, 0.75), max.floor = max(floorNumber))
```

```
## # A tibble: 6 x 7
##   clst_label min.floor q1.floor median.floor mean.floor q3.floor max.floor
##   <fct>          <int>    <dbl>        <dbl>      <dbl>    <dbl>     <int>
## 1 1                  3     6              13       13.9     20          32
## 2 2                  3     6.75           15       14.8     20          30
## 3 3                  4     6              14       13.9     19.8        30
## 4 4                  3     6               6       10.8     15          32
## 5 5                  4     6               7       12.3     18          32
## 6 6                  3     6               6       11.3     16          32
```

```
df.clst %>%
  group_by(clst_label) %>%
  summarise(min.dist = min(distance), q1.dist = quantile(distance, 0.25),
            median.dist = median(distance), mean.dist = mean(distance),
            q3.dist = quantile(distance, 0.75), max.dist = max(distance))
```

```
## # A tibble: 6 x 7
##   clst_label min.dist q1.dist median.dist mean.dist q3.dist max.dist
##   <fct>         <dbl>   <dbl>       <dbl>     <dbl>   <dbl>    <dbl>
## 1 1             1803.   7966.      10762.    12331.  17409.   34715.
## 2 2             1599.   7619.      12007.    13435.  17423.   37472.
## 3 3             2940.   6579.      11639.    12936.  17045.   36961.
## 4 4             1810.   7205.      10679.    12705.  17007.   35468.
## 5 5             2474.   6597.       9814.    12176.  16904.   34789.
## 6 6             2317.   6468.       9389.    11754.  16465.   36491.
```

```
# Renovation Condition and Five Years Property
df.clst %>%
  group_by(clst_label, renovationCondition) %>%
  summarise(count = n(), .groups = 'drop')
```

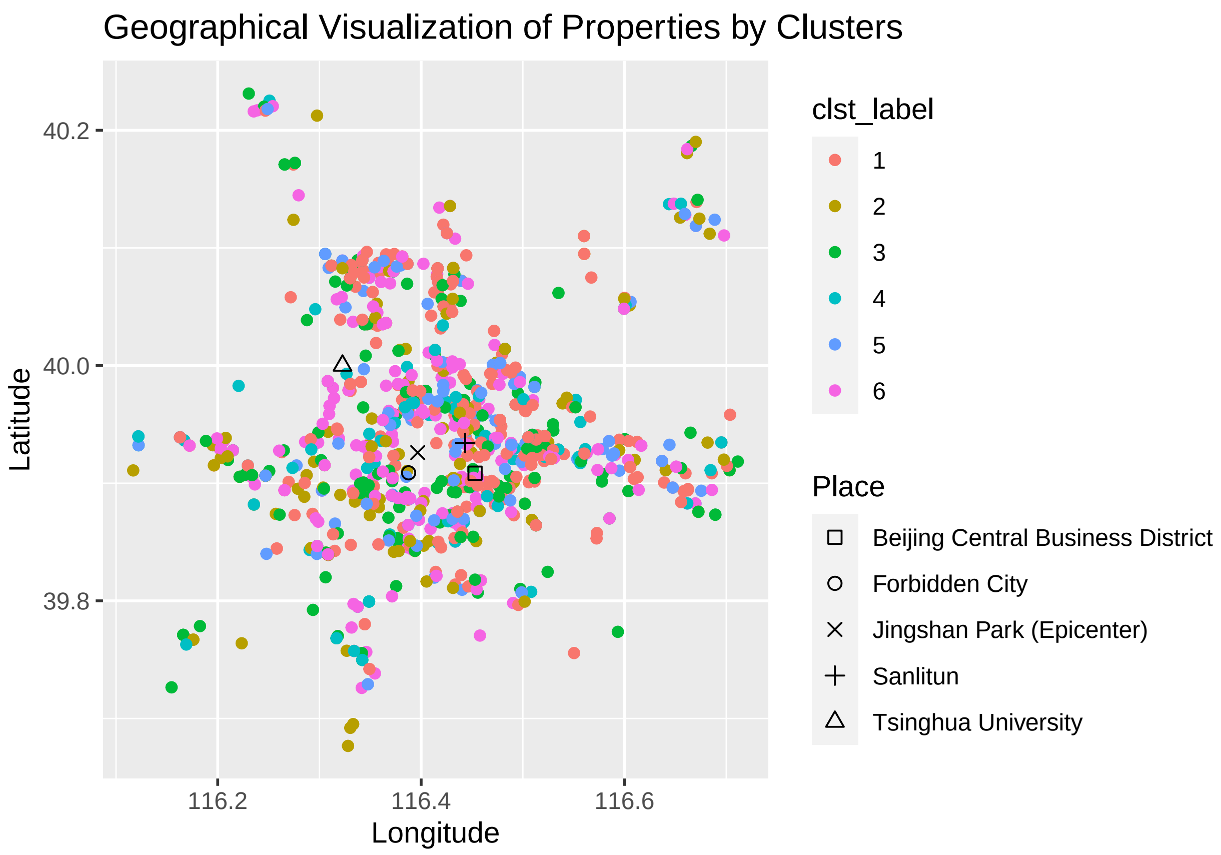
```
## # A tibble: 11 x 3
##    clst_label renovationCondition count
##    <fct>      <fct>               <int>
##  1 1          4                     288
##  2 2          1                     118
##  3 2          2                       5
##  4 2          4                       1
##  5 3          4                     162
##  6 4          2                       1
##  7 4          3                      81
##  8 5          1                     101
##  9 5          2                       9
## 10 6          2                       2
## 11 6          3                     232
```

```
df.clst %>%
  group_by(clst_label, fiveYearsProperty) %>%
  summarise(count = n(), .groups = 'drop')
```

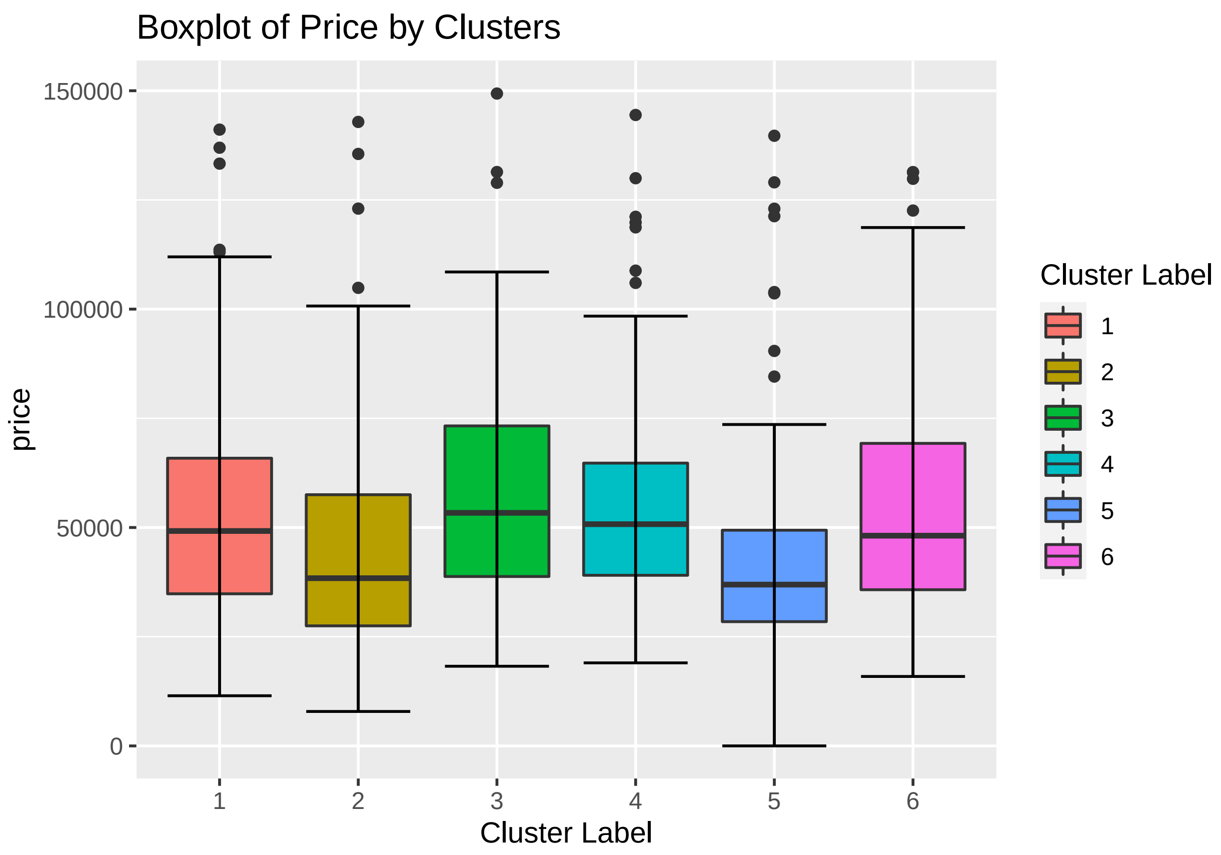
```
## # A tibble: 6 x 3
##   clst_label fiveYearsProperty count
##   <fct>      <fct>             <int>
## 1 1          1                   288
## 2 2          0                   124
## 3 3          0                   162
## 4 4          0                    82
## 5 5          1                   110
## 6 6          1                   234
```

Visualizations

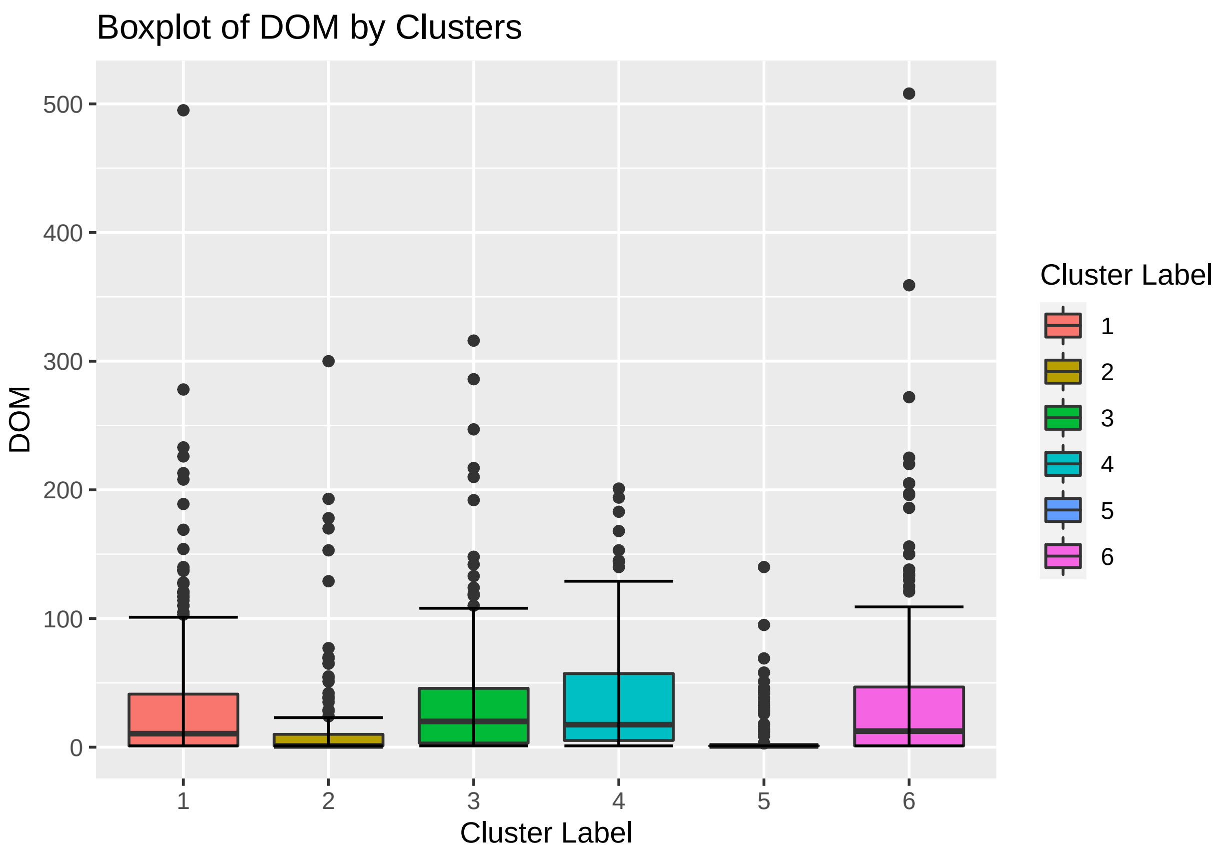
```
# Add additional columns to the original sample dataset
sample$clst_label = final$cluster

# Make a new dataframe for some landmarks
landmarks = data.frame(
  Place = c("Jingshan Park (Epicenter)", "Beijing Central Business District",
            "Forbidden City", "Sanlitun", "Tsinghua University"),
  Lng = c(116.3966463362935, 116.453, 116.387665116,
          116.443248227, 116.322665376),
  Lat = c(39.92600108466268, 39.9085, 39.908829698,
          39.93416293, 40.0))

# Geographical visualization
ggplot(data = sample, aes(x = Lng, y = Lat, col = clst_label)) +
  geom_point() +
  geom_point(data = landmarks, aes(x = Lng, y = Lat, shape = Place), col = "black", size = 2) +
  scale_shape_manual(values=c(0,1,4,3,2)) +
  labs(x = "Longitude", y = "Latitude",
       title = "Geographical Visualization of Properties by Clusters")
```


Geographical Visualization of Properties by Clusters
Latitude
40.2
40.0
39.8
116.2
116.4
116.6
Longitude
clst_label
1
2
3
4
5
6
Place
Beijing Central Business District
Forbidden City
Jingshan Park (Epicenter)
Sanlitun
Tsinghua University

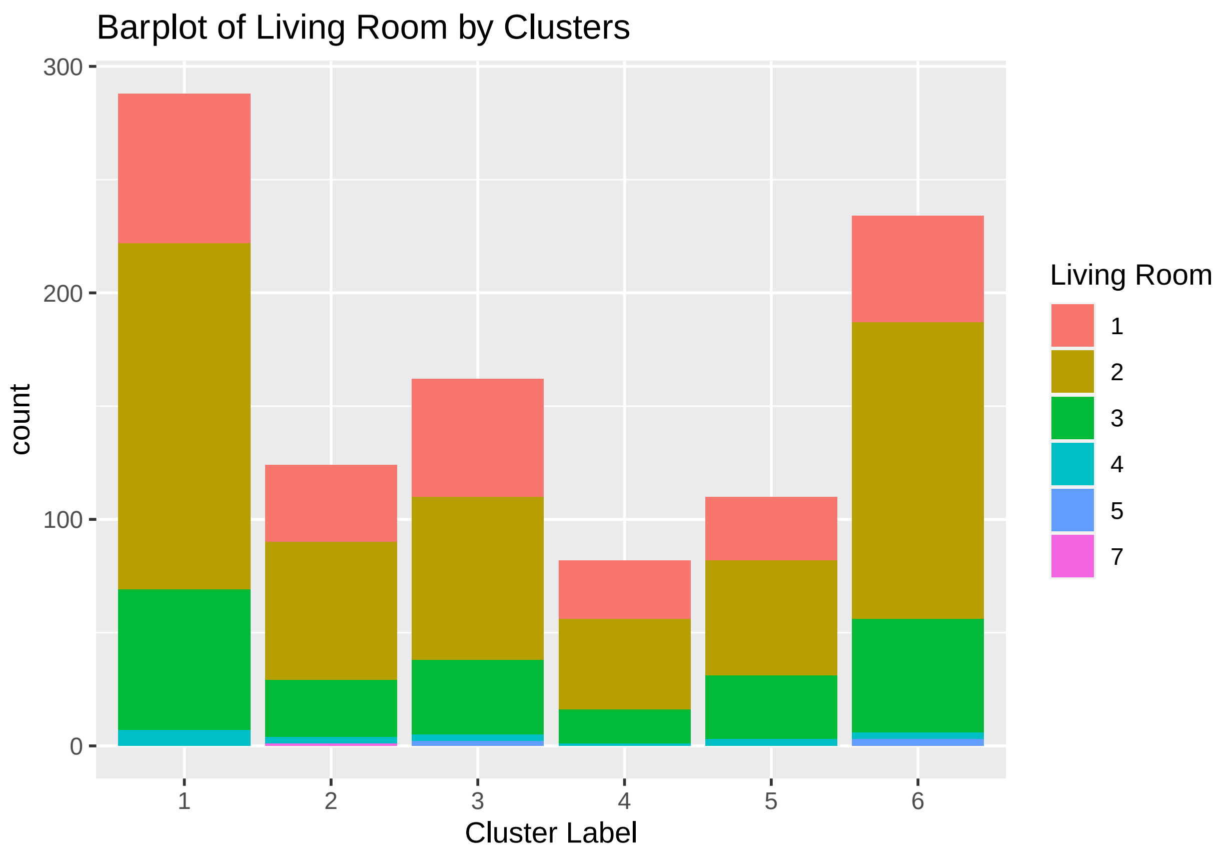
```
# Cluster Visualizations
ggplot(df.clst, aes(x = clst_label, y = price, fill = clst_label)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  labs(title = "Boxplot of Price by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Cluster Label")
```



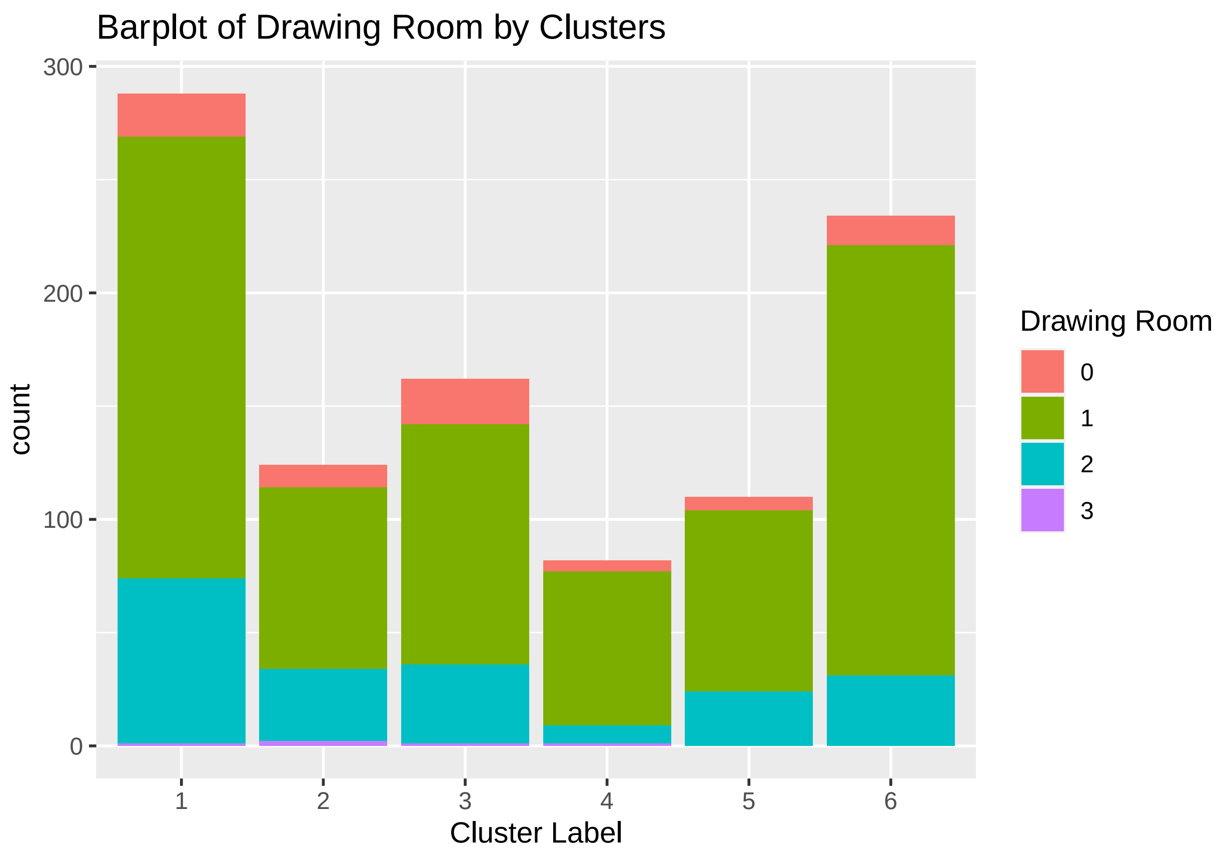
```
ggplot(df.clst, aes(x = clst_label, y = DOM, fill = clst_label)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  labs(title = "Boxplot of DOM by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Cluster Label")
```



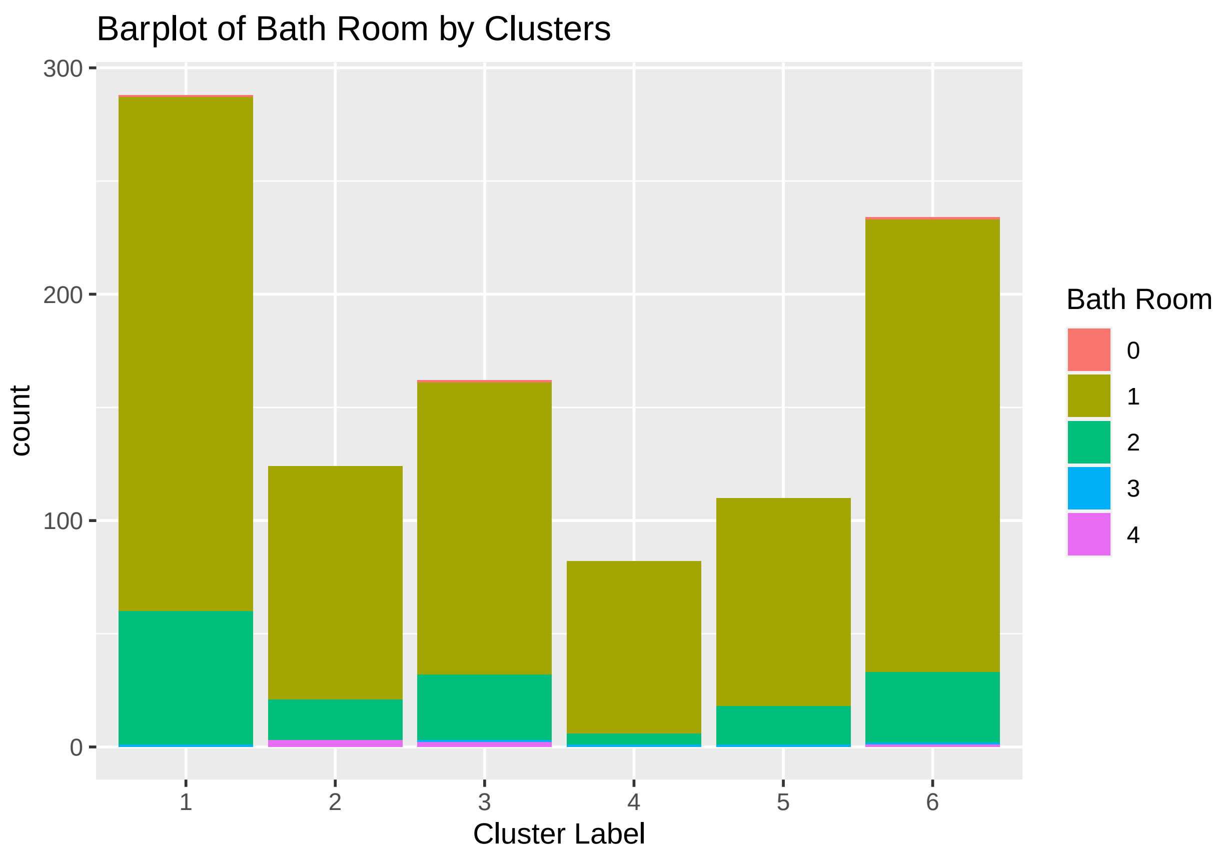
```
ggplot(df.clst, aes(x = clst_label, fill = as.factor(livingRoom))) +
  geom_bar() +
  labs(title = "Barplot of Living Room by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Living Room")
```



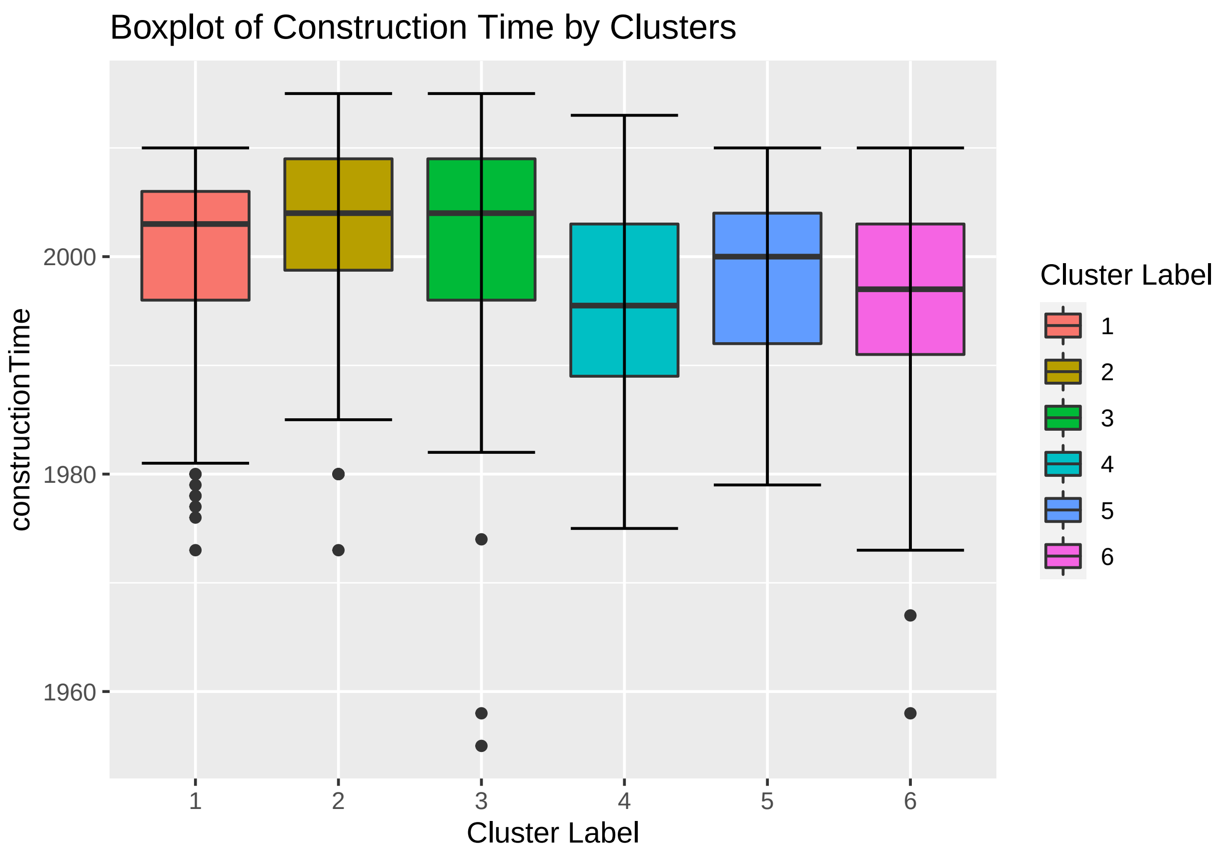
```r
ggplot(df.clst, aes(x = clst_label, fill = as.factor(drawingRoom))) +
  geom_bar() +
  labs(title = "Barplot of Drawing Room by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Drawing Room")
```



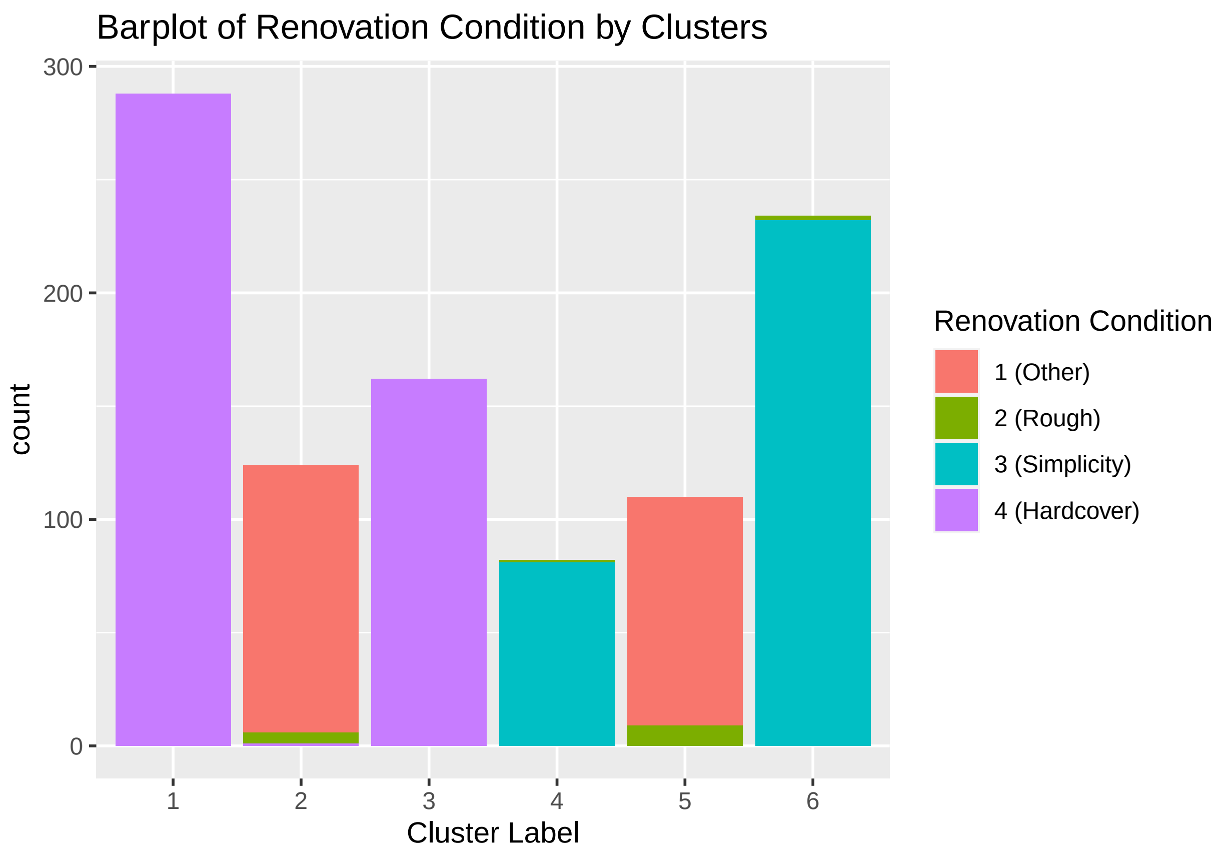
```r
ggplot(df.clst, aes(x = clst_label, fill = as.factor(bathRoom))) +
  geom_bar() +
  labs(title = "Barplot of Bath Room by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Bath Room")
```



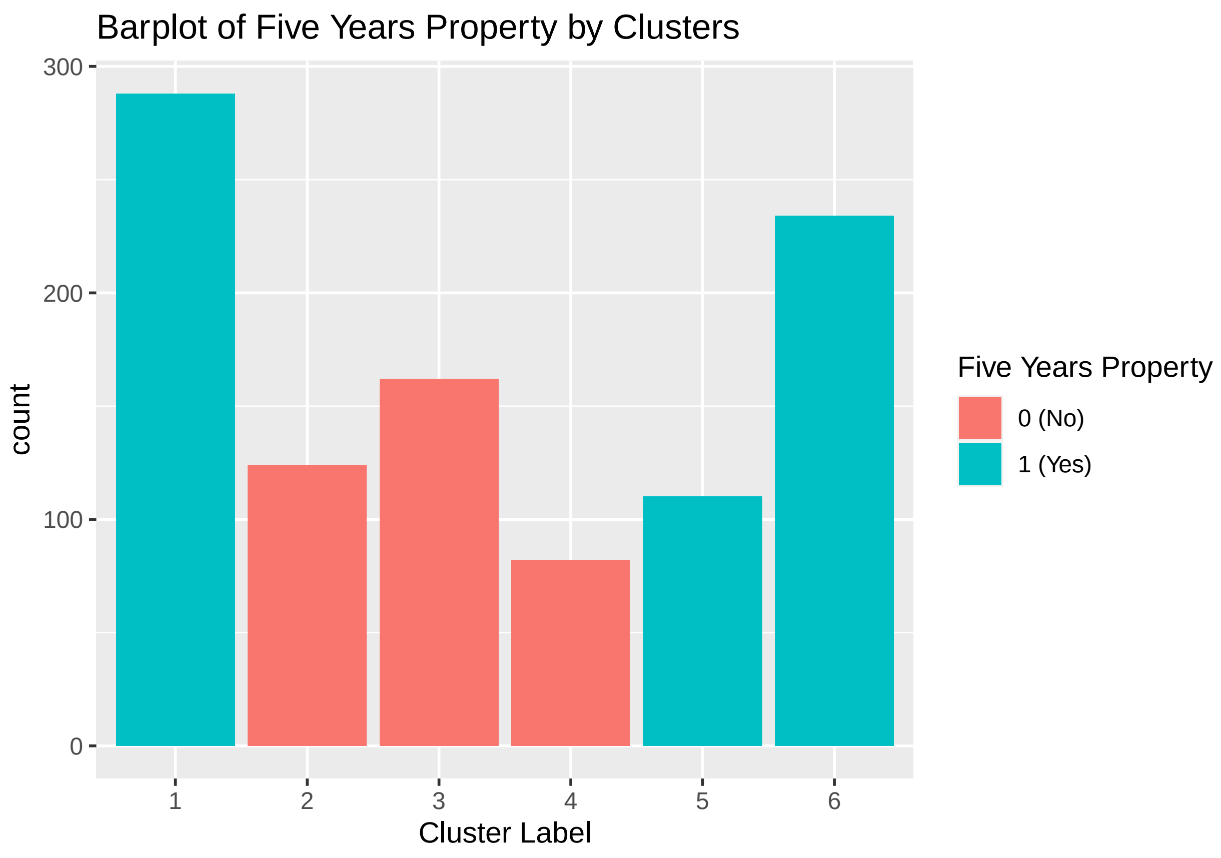
```r
ggplot(df.clst, aes(x = clst_label, y = constructionTime, fill = clst_label)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  labs(title = "Boxplot of Construction Time by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Cluster Label")
```



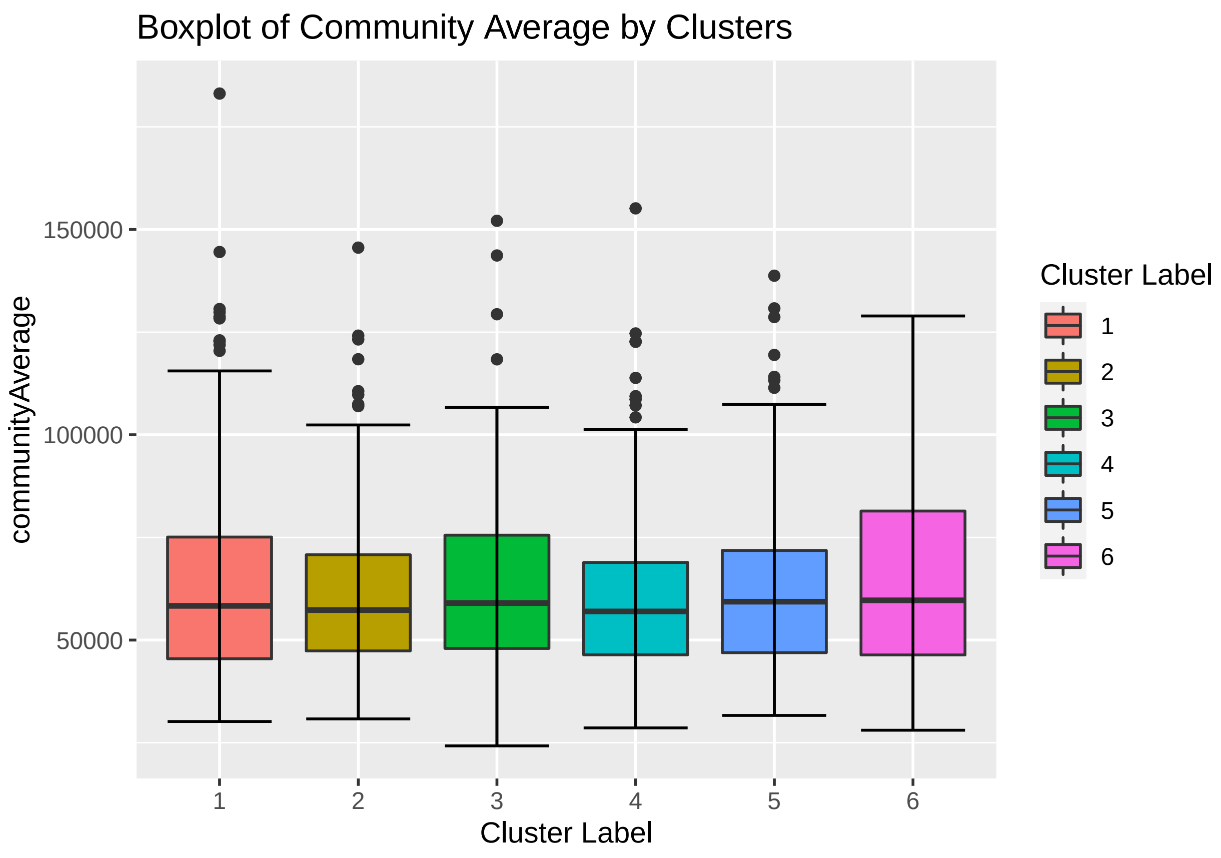
```
ggplot(df.clst, aes(x = clst_label, fill = renovationCondition)) +
  geom_bar() +
  scale_fill_discrete(name = "Renovation Condition",
  labels=c('1 (Other)', '2 (Rough)', '3 (Simplicity)', '4 (Hardcover)')) +
  labs(title = "Barplot of Renovation Condition by Clusters", x = "Cluster Label")
```



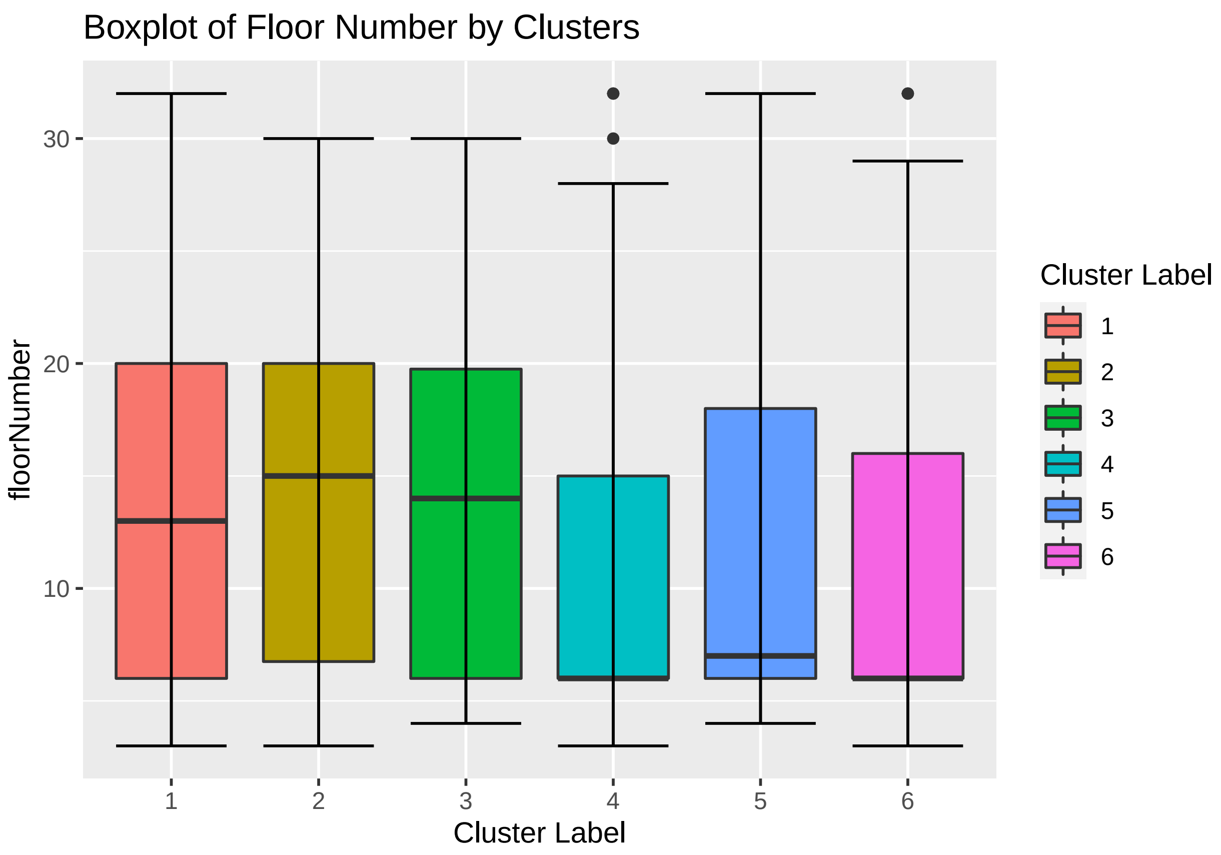
```r
ggplot(df.clst, aes(x = clst_label, fill = fiveYearsProperty)) +
  geom_bar() +
  scale_fill_discrete(name = "Five Years Property", labels=c('0 (No)', '1 (Yes)')) +
  labs(title = "Barplot of Five Years Property by Clusters", x = "Cluster Label")
```



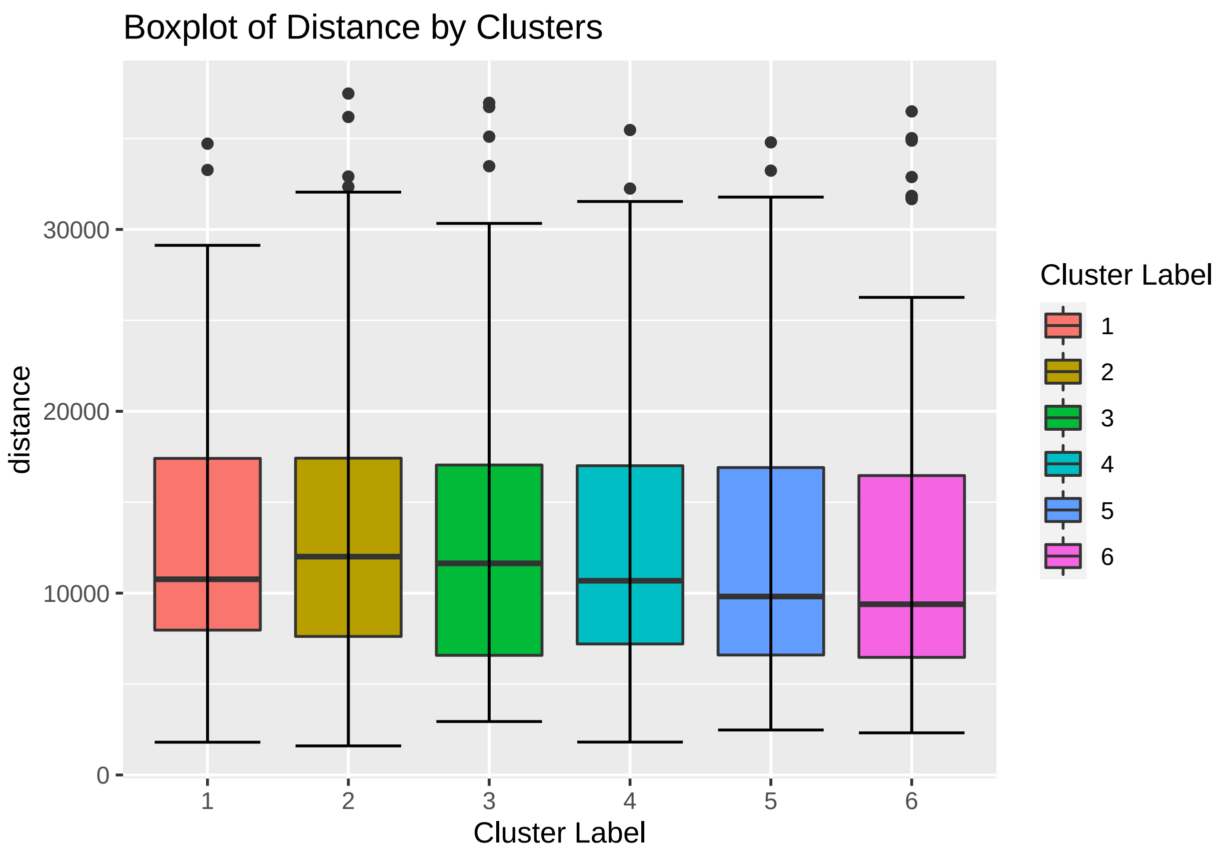
```
ggplot(df.clst, aes(x = clst_label, y = communityAverage, fill = clst_label)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  labs(title = "Boxplot of Community Average by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Cluster Label")
```



```
ggplot(df.clst, aes(x = clst_label, y = floorNumber, fill = clst_label)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  labs(title = "Boxplot of Floor Number by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Cluster Label")
```



```
ggplot(df.clst, aes(x = clst_label, y = distance, fill = clst_label)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  labs(title = "Boxplot of Distance by Clusters", x = "Cluster Label") +
  scale_fill_discrete(name = "Cluster Label")
```



Above is the summary statistics and visualization of every variable by cluster labels. When plotting observations colored with cluster labels based on geographical information, we do not see any clear patterns.

Most variables seem to show similar trends such as number of living/drawing/bathrooms and communityAverage.

For 'price', although we do not know whether the differences between clusters are significant or not (need to do some tests for significance such as ANOVA), cluster 3 seems to have the highest 'price' average while cluster 2 and 5 have the lowest.

There are somewhat notable differences of average 'constructionTime' by clusters as well. Cluster 3 seems to have the highest average constructionTime, meaning that most of the properties in the cluster are relatively new, while cluster 4 has the lowest average, indicating relatively old.

Some interesting points can be also detected from 'renovationCondition' and 'fiveYearsProperty'. For 'renovationCondition', cluster 1 and 3 have a renovation condition of 4 (hardcover) whereas cluster 4 and 6 are 3 (simplicity) and 2 and 5 are mainly 1 (other). For 'fiveYearsProperty', the owners of cluster 1, 5, and 6 had their property less than 5 years while the properties of other clusters (2, 3, and 4) are more than 5 years.

'floorNumber' has somewhat notable difference of the median between clusters. Cluster 1, 2, and 3 have relatively higher medians than the rest clusters (4, 5, 6). We may assume most properties in the former clusters as a large and high-rise apartment whereas the latter clusters to be a low-medium house (multi-complex) or walk-up.

We can do further analysis specifically on the clusters with the highest and lowest average price, which would be cluster 3 (highest) and 5 (lowest), to compare and figure out any interesting features that might be significant in terms of distinguishing these two clusters.

1. 'DOM' - This is 'Days on Market', which tells us how many days a property has been on the market. It is somewhat obvious that cheaper properties would be more likely to be sold out quickly than the

expensive ones, and our assumption matches with the plot and summary statistics values; Cluster 3 has both the median and mean 'DOM' (20.0, 36.697531) than cluster 5 (1.0, 8.718182).

2. 'living/drawing/bathroom' - They are the numbers of living/drawing/bathrooms, and both cluster 3 and 5 show fairly identical proportion trends; mostly 2 livingrooms, 1 drawing and bathroom.

3. 'constructionTime' - This indicates when the property has been constructed, and we can see that cluster 3 again has both the median and mean 'constructionTime' (2004.0, 2001.809) than cluster 5 (2000.0, 1997.718). This also somewhat make sense that the properties that were constructed relatively recently tend to be more expensive than the older ones.

4. 'renovationCondition' - This is the variable about the type of renovation conditions, and it is interesting that all properties in cluster 3 are 4 (hardcover) whereas almost all properties in cluster 5 are 1 (other). Since there is not enough information about the conditions, we could naively consider condition 4 (hardcover) to be the most preferred renovation condition for the expensive properties.

5. 'fiveYearsProperty' - This tells us whether the owner of a property has had the property for less than 5 years or not (1 for yes, 0 for no). All properties in cluster 3 were owned by their owner for more than 5 years while the properties in cluster 5 were owned less than 5 years. One hypothesis we could make is that the owners with more expensive properties may possibly contract with tenants and get high monthly rent from them whereas having relatively cheaper properties would not be that profitable and thus selling to others would be more beneficial than getting monthly rent payment from tenants.

6. 'communityAverage' - Unfortunately, there is no specific description about the variable in Kaggle (Average income of a commnunity? Average price of a community?). We can only find out that the properties in cluster 3 tend to have slighlty lower median and mean 'communityAverage' (59020, 62978.34) than cluster 5 (59355, 63468.37). Further analysis cannot be done due to lack of information, but if we assume this variable to be one of our guesses, then this opposite trend is interesting considering a general assumption that most people have, which is that the people living in a expensive property would have higher income than the people who are living in a cheaper property, is not actually true in this case.

7. 'floorNumber' - This is the number of floors (x-story) in a property, and the same analysis we made above can be brought here as well; Since the median 'floorNumber' for cluster 5 (14) is double the cluster 3 (7), cluster 5 could be thought as a large and high-rise apartment whereas cluster 3 would be relatively a lower building or even wallk-up.

8. 'distance' - This is the variable that was not in the original dataset, but instead there were 'longitude' and 'latitude'. Using these two original variables, we have calculated the distance between the epicenter of Beijing (Jingshan park) and a property. When observing both the median and mean, it is interesting that cluster 3 has higher values (11638.617, 12935.55) than cluster 5 (9814.364, 12175.88), meaning that the cheaper properties (cluster 5) are actually closer to the epicenter than the expensive properties (cluster 3). Similar to 'communityAverage', one of common thoughts that most people including us have is that the properties near the epicenter of a city would be more expensive than the properties that are relatively far away from the epicenter, and this is not the case for this data.

To summarize, most variables revealed somewhat trivial trends that match with our prior assumptions. However, there were some variables such as 'distance' that showed opposite tendencies.

Additional Analysis - Under / Normal / Over priced

We would like to do an additional analysis with 'price' variable. Within each cluster, we would like to figure out the trend of 'price', meaning which property has a price greater or less than the middle 50% (above Q3 and below Q1). This way, we can not only see the general distribution of 'price', but also able to get an idea of what kind of features these properties with so called 'not a trend' price have. With the analysis, it may be possible to learn more about the property market trend that goes behind the scene.

Make another column, 'price_trend'

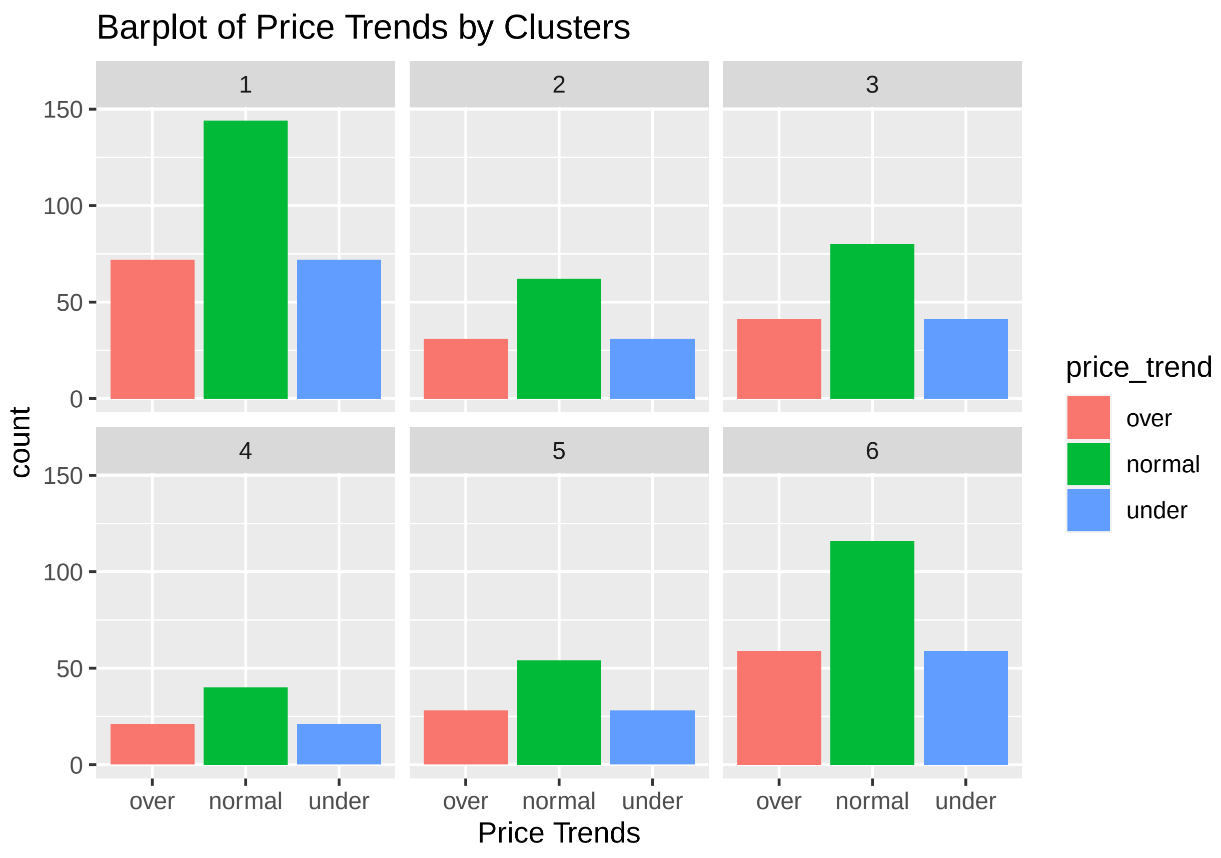
```r
df.clst$price_trend = ""

for (i in 1:nrow(df.clst)) {
  if (df.clst[i,]$clst_label == 1) {
    # Check whether the price...
    # 1) lower than the cluster's Q1 (25%)
    # 2) higher than the cluster's Q3 (75%)
    # 3) in between middle 50% (Interquatile (IQR) = Q3 - Q1)
    if (df.clst[i,]$price < quantile(df.clst[df.clst$clst_label == 1, ]$price, 0.25)) {
      df.clst[i,]$price_trend = 'under'
    } else if (df.clst[i,]$price > quantile(df.clst[df.clst$clst_label == 1, ]$price, 0.75)) {
      df.clst[i,]$price_trend = 'over'
    } else {
      df.clst[i,]$price_trend = 'normal'
    }
  } else if (df.clst[i,]$clst_label == 2) {
    if (df.clst[i,]$price < quantile(df.clst[df.clst$clst_label == 2, ]$price, 0.25)) {
      df.clst[i,]$price_trend = 'under'
    } else if (df.clst[i,]$price > quantile(df.clst[df.clst$clst_label == 2, ]$price, 0.75)) {
      df.clst[i,]$price_trend = 'over'
    } else {
      df.clst[i,]$price_trend = 'normal'
    }
  } else if (df.clst[i,]$clst_label == 3) {
    if (df.clst[i,]$price < quantile(df.clst[df.clst$clst_label == 3, ]$price, 0.25)) {
      df.clst[i,]$price_trend = 'under'
    } else if (df.clst[i,]$price > quantile(df.clst[df.clst$clst_label == 3, ]$price, 0.75)) {
      df.clst[i,]$price_trend = 'over'
    } else {
      df.clst[i,]$price_trend = 'normal'
    }
  } else if (df.clst[i,]$clst_label == 4) {
    if (df.clst[i,]$price < quantile(df.clst[df.clst$clst_label == 4, ]$price, 0.25)) {
      df.clst[i,]$price_trend = 'under'
    } else if (df.clst[i,]$price > quantile(df.clst[df.clst$clst_label == 4, ]$price, 0.75)) {
      df.clst[i,]$price_trend = 'over'
    } else {
      df.clst[i,]$price_trend = 'normal'
    }
  } else if (df.clst[i,]$clst_label == 5) {
    if (df.clst[i,]$price < quantile(df.clst[df.clst$clst_label == 5, ]$price, 0.25)) {
      df.clst[i,]$price_trend = 'under'
    } else if (df.clst[i,]$price > quantile(df.clst[df.clst$clst_label == 5, ]$price, 0.75)) {
      df.clst[i,]$price_trend = 'over'
    } else {
      df.clst[i,]$price_trend = 'normal'
    }
  } else {
    if (df.clst[i,]$price < quantile(df.clst[df.clst$clst_label == 6, ]$price, 0.25)) {
      df.clst[i,]$price_trend = 'under'
    } else if (df.clst[i,]$price > quantile(df.clst[df.clst$clst_label == 6, ]$price, 0.75)) {
      df.clst[i,]$price_trend = 'over'
    } else {
      df.clst[i,]$price_trend = 'normal'
    }
  }
}

# Customize the order of the variable
df.clst$price_trend = factor(df.clst$price_trend, levels = c("over", "normal", "under"))

#head(df.clst, 3)
```

Visualizations

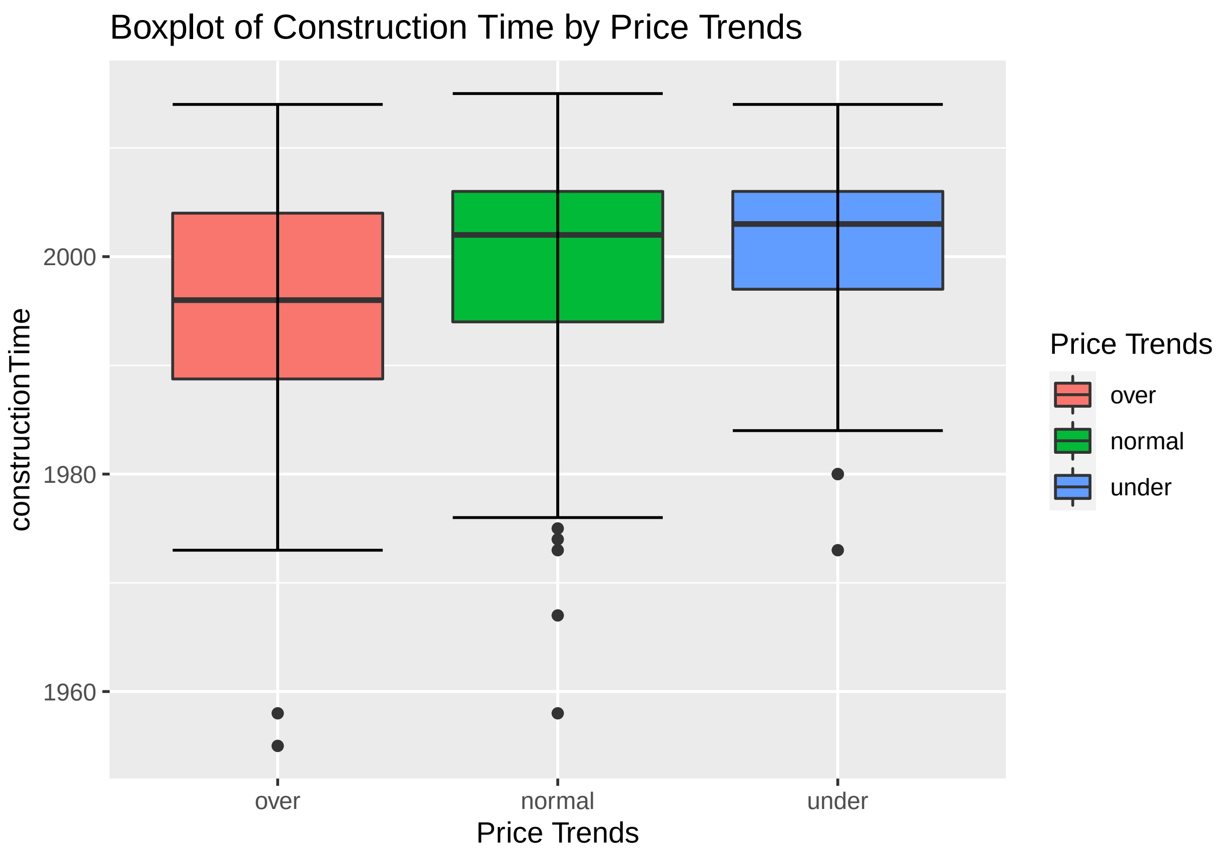
```
df.clst %>%
  group_by(clst_label, price_trend) %>%
  summarise(count = n(), .groups = 'drop')
```

```
## # A tibble: 18 x 3
##    clst_label price_trend count
##    <fct>      <fct>       <int>
##  1 1          over           72
##  2 1          normal        144
##  3 1          under          72
##  4 2          over           31
##  5 2          normal         62
##  6 2          under          31
##  7 3          over           41
##  8 3          normal         80
##  9 3          under          41
## 10 4          over           21
## 11 4          normal         40
## 12 4          under          21
## 13 5          over           28
## 14 5          normal         54
## 15 5          under          28
## 16 6          over           59
## 17 6          normal        116
## 18 6          under          59
```

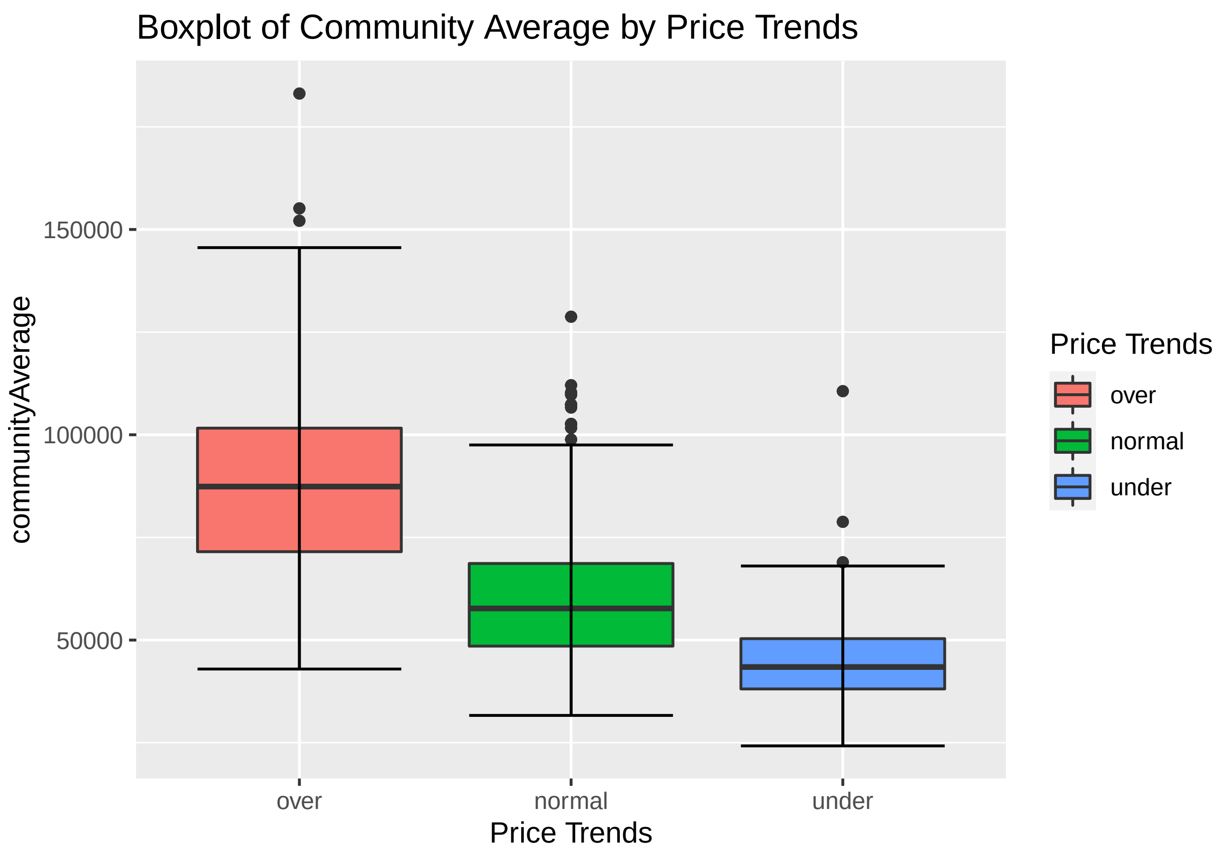
```
ggplot(df.clst, aes(x = price_trend, fill = price_trend)) +
  geom_bar() +
  facet_wrap(~clst_label) +
  labs(title = "Barplot of Price Trends by Clusters", x = "Price Trends")
```



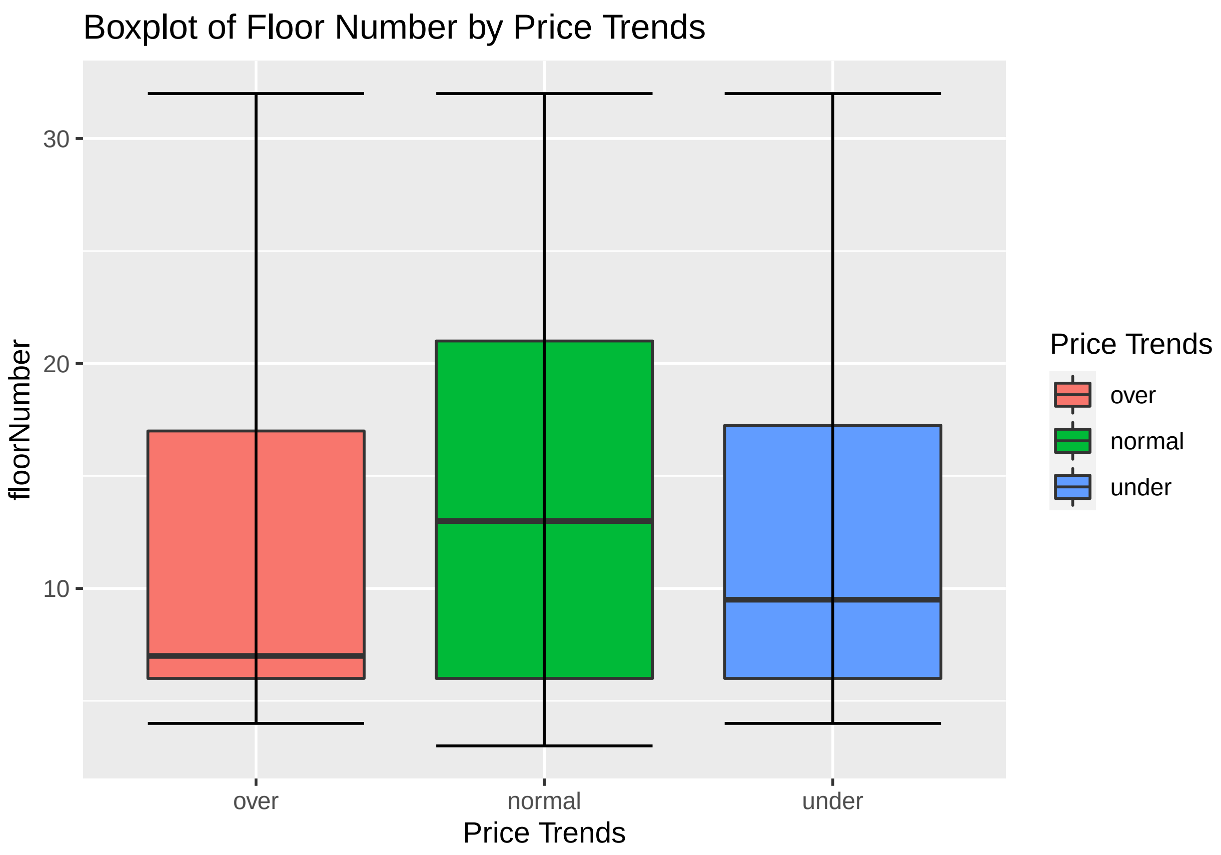
```r
ggplot(df.clst, aes(x = price_trend, y = constructionTime, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Construction Time by Price Trends", x = "Price Trends")
```



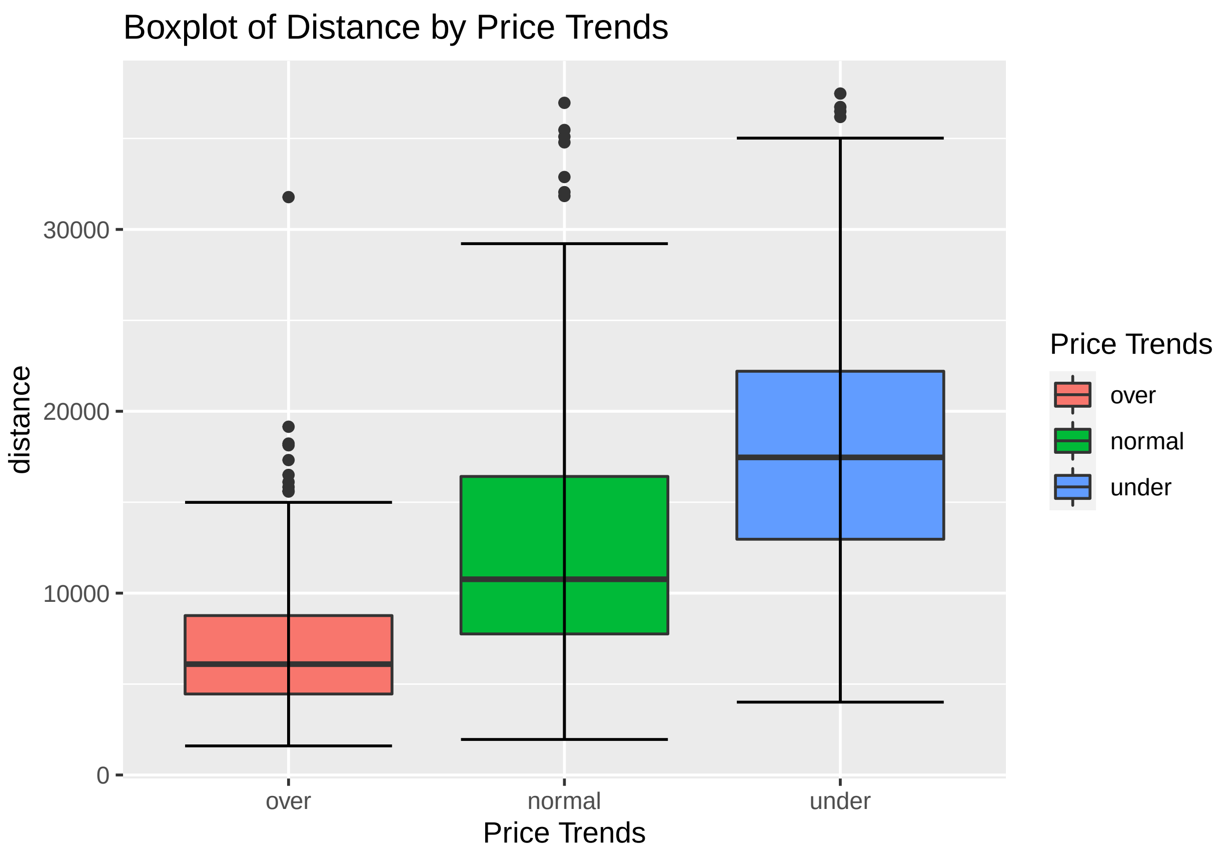
```
ggplot(df.clst, aes(x = price_trend, y = communityAverage, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Community Average by Price Trends", x = "Price Trends")
```



```
ggplot(df.clst, aes(x = price_trend, y = floorNumber, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Floor Number by Price Trends", x = "Price Trends")
```

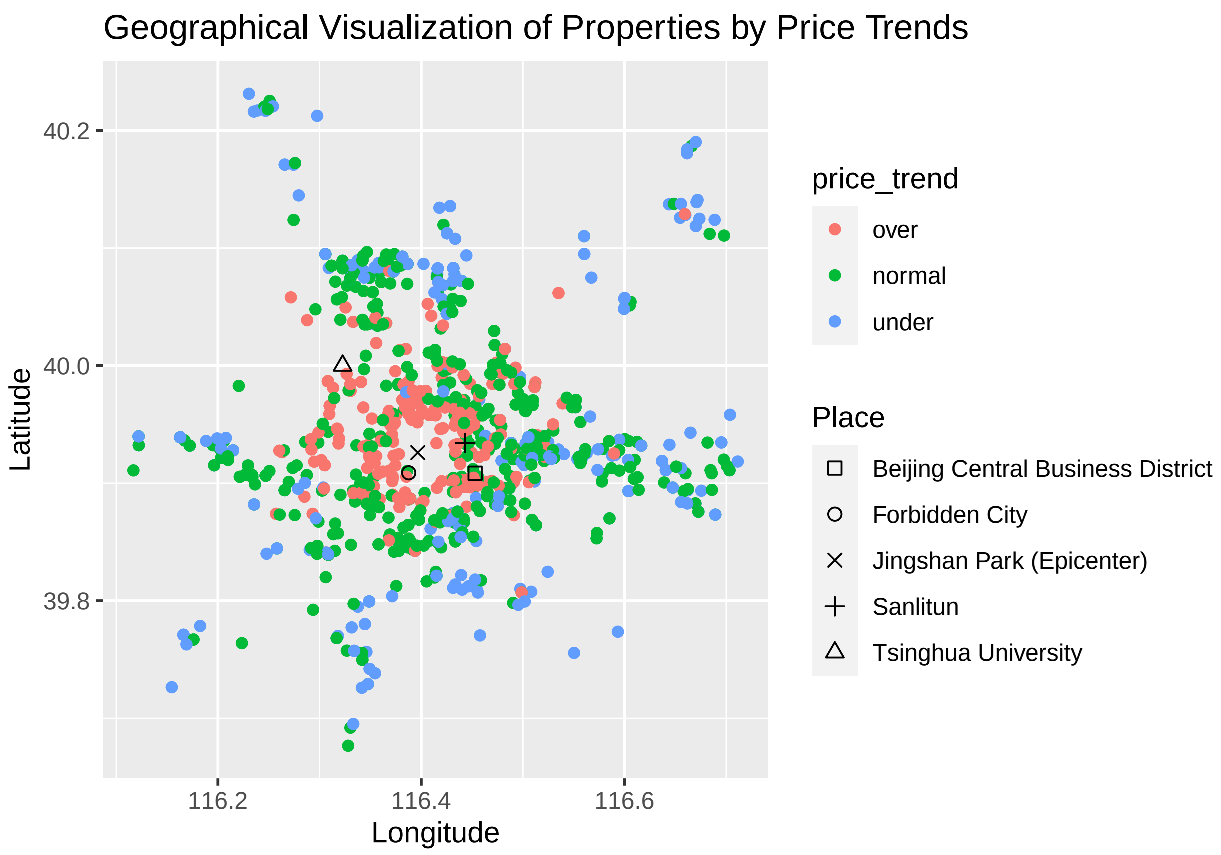


```
ggplot(df.clst, aes(x = price_trend, y = distance, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Distance by Price Trends", x = "Price Trends")
```

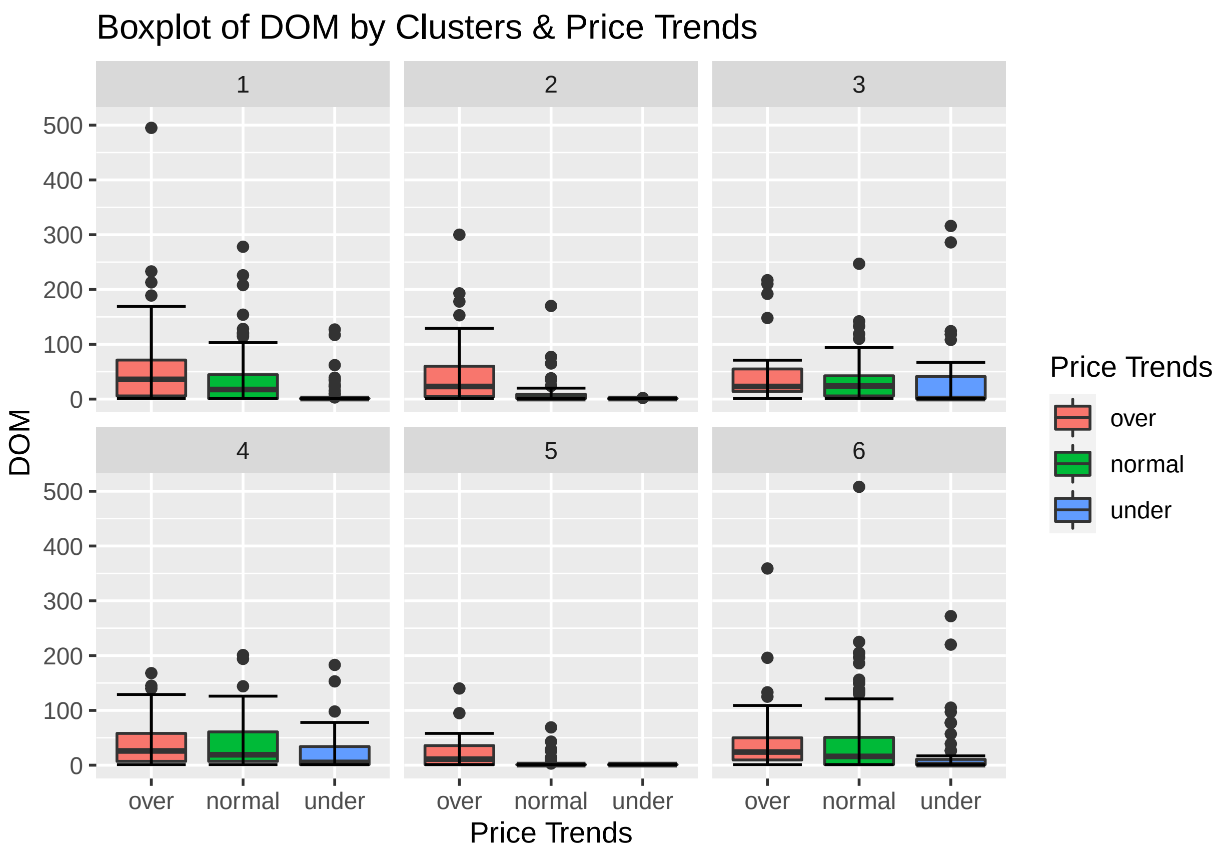


More visualizations by cluster labels & 'price_trend'

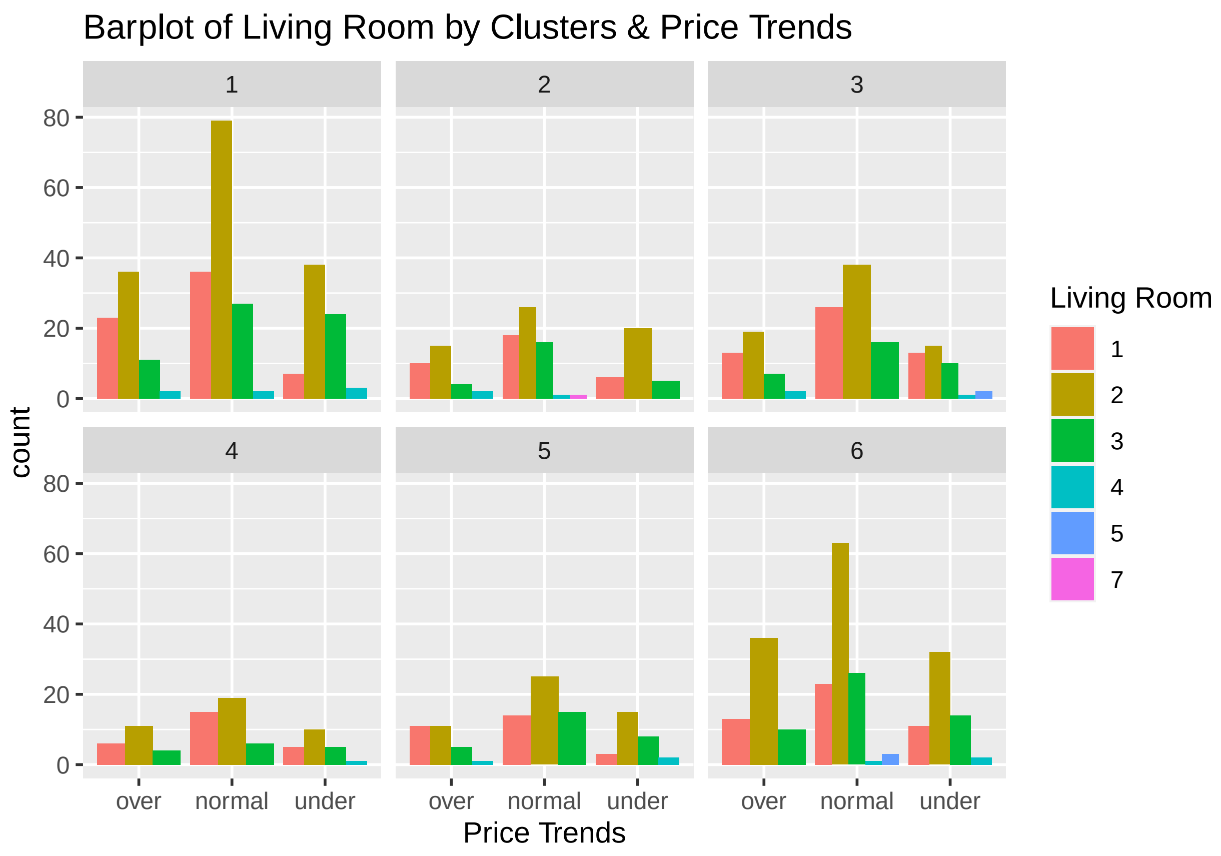
```
# Add additional columns to the original sample dataset
sample$price_trend = df.clst$price_trend

# Geographical Visualization
ggplot(data = sample, aes(x = Lng, y = Lat, col = price_trend)) +
  geom_point() +
  geom_point(data = landmarks, aes(x = Lng, y = Lat, shape = Place), col = "black", size = 2) +
  scale_shape_manual(values=c(0,1,4,3,2)) +
  labs(x = "Longitude", y = "Latitude",
       title = "Geographical Visualization of Properties by Price Trends")
```



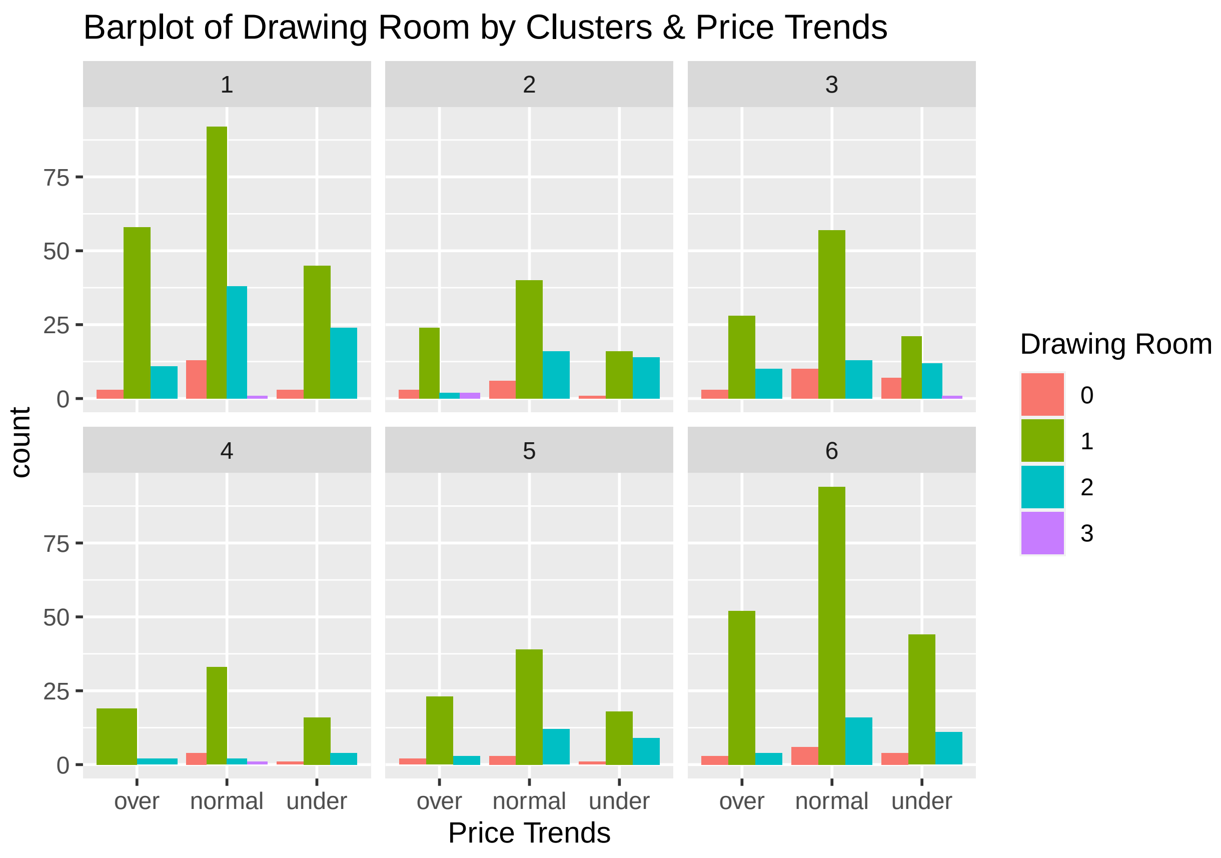
```r
# Price Trends Visualizations
ggplot(df.clst, aes(x = price_trend, y = DOM, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of DOM by Clusters & Price Trends", x = "Price Trends")
```



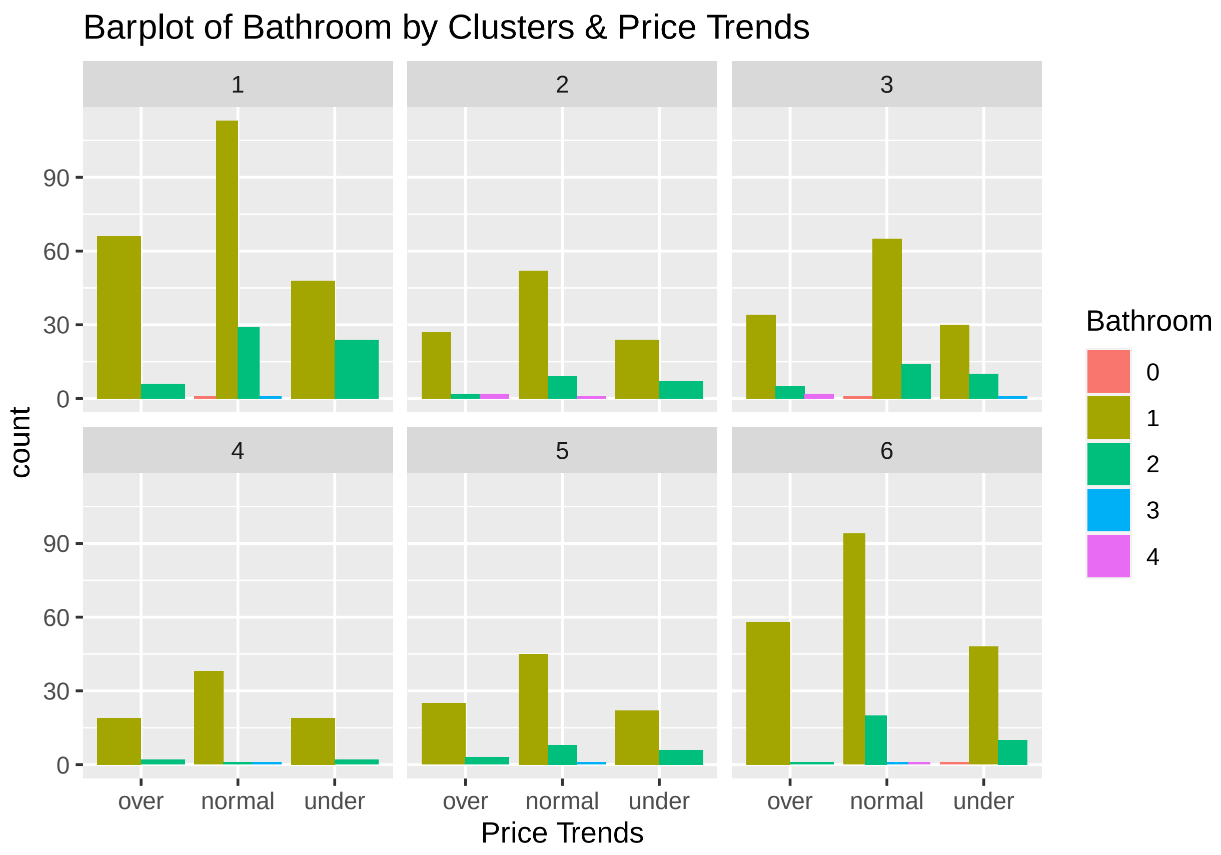
```r
ggplot(df.clst, aes(x = price_trend, fill = as.factor(livingRoom))) +
  geom_bar(position = "dodge") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Living Room") +
  labs(title = "Barplot of Living Room by Clusters & Price Trends", x = "Price Trends")
```



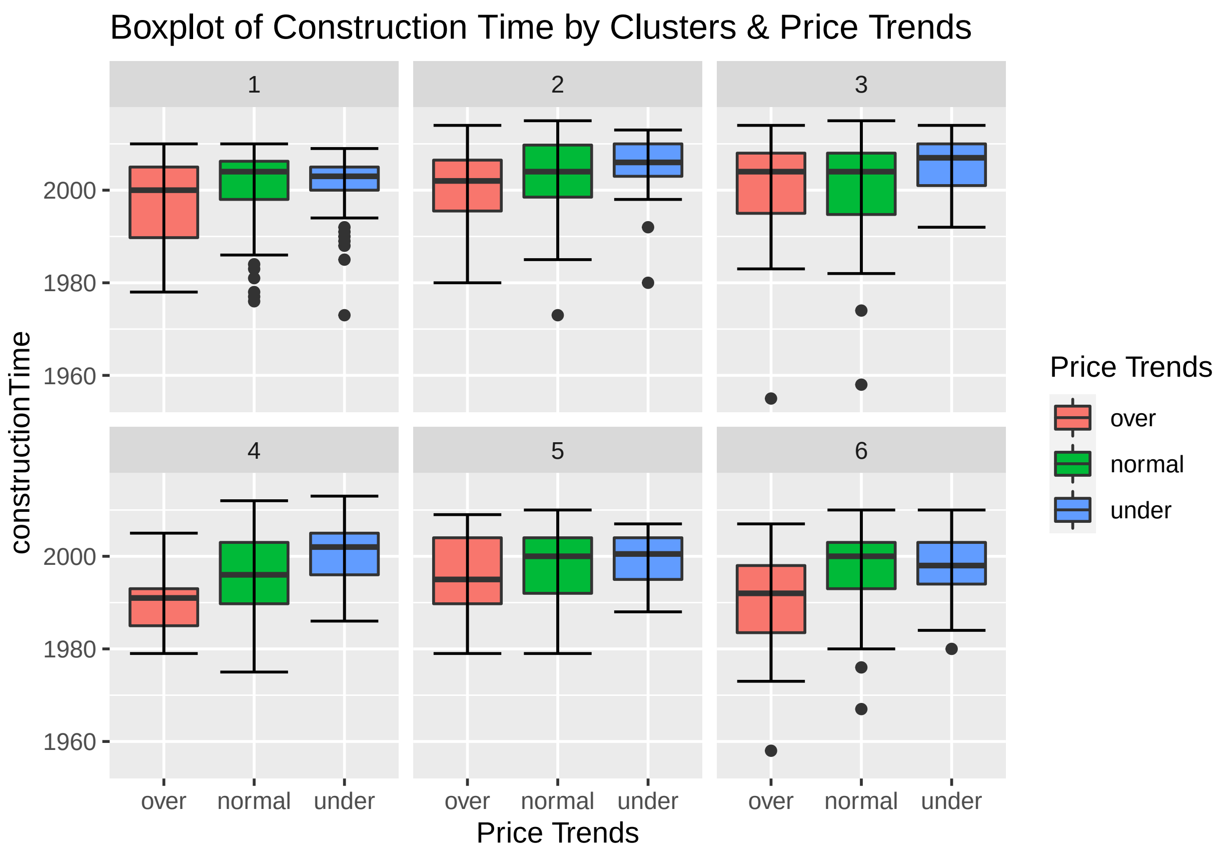
```r
ggplot(df.clst, aes(x = price_trend, fill = as.factor(drawingRoom))) +
  geom_bar(position = "dodge") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Drawing Room") +
  labs(title = "Barplot of Drawing Room by Clusters & Price Trends", x = "Price Trends")
```



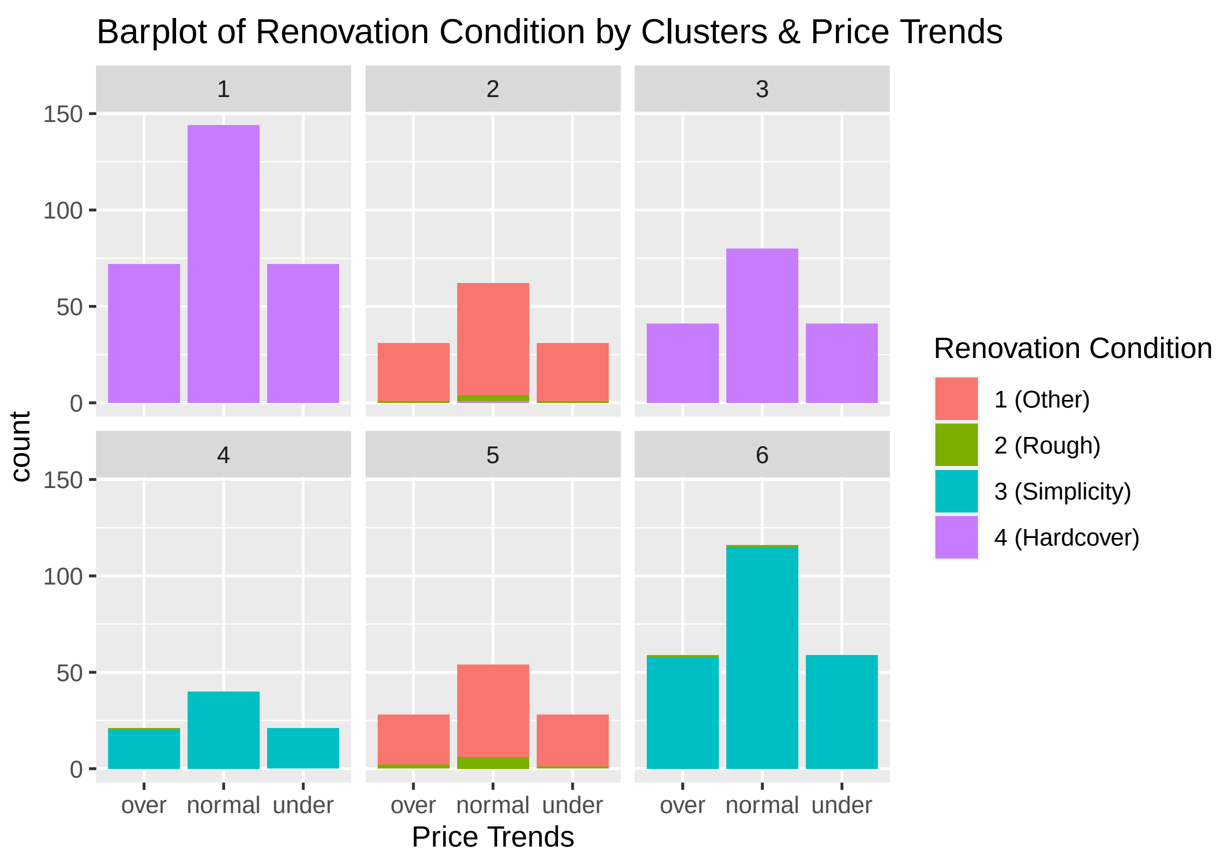
```
ggplot(df.clst, aes(x = price_trend, fill = as.factor(bathRoom))) +
  geom_bar(position = "dodge") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Bathroom") +
  labs(title = "Barplot of Bathroom by Clusters & Price Trends", x = "Price Trends")
```



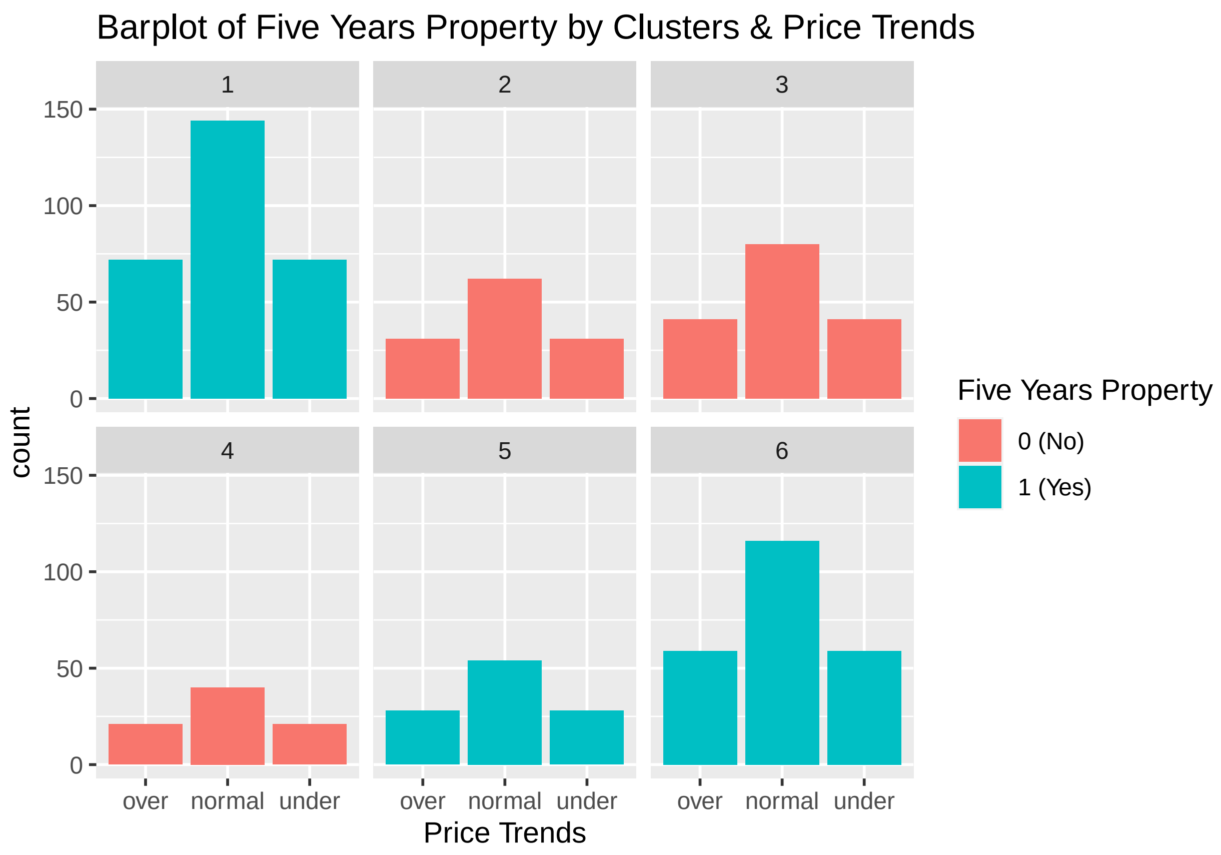
```
ggplot(df.clst, aes(x = price_trend, y = constructionTime, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Construction Time by Clusters & Price Trends", x = "Price Trends")
```



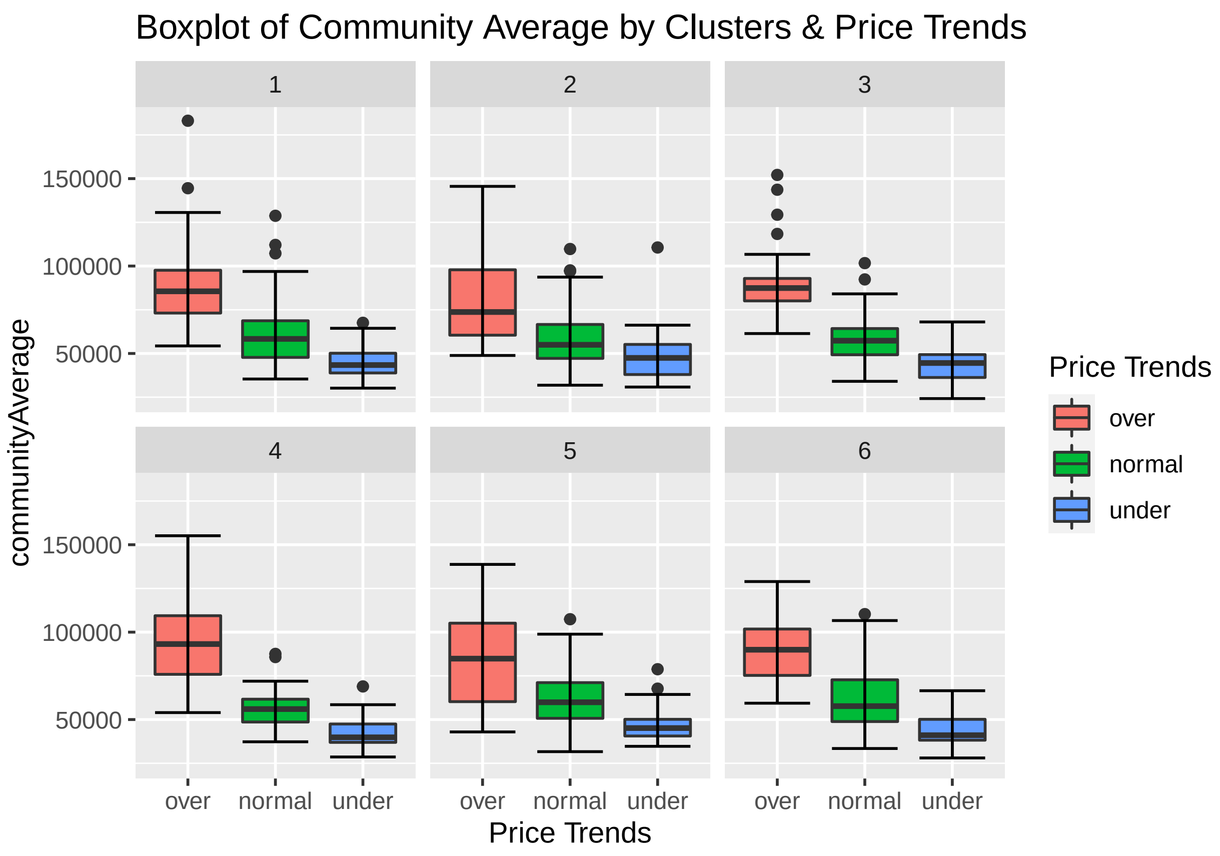
```
ggplot(df.clst, aes(x = price_trend, fill = renovationCondition)) +
  geom_bar() +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Renovation Condition",
                      labels=c('1 (Other)', '2 (Rough)', '3 (Simplicity)', '4 (Hardcover)')) +
  labs(title = "Barplot of Renovation Condition by Clusters & Price Trends", x = "Price Trends")
```



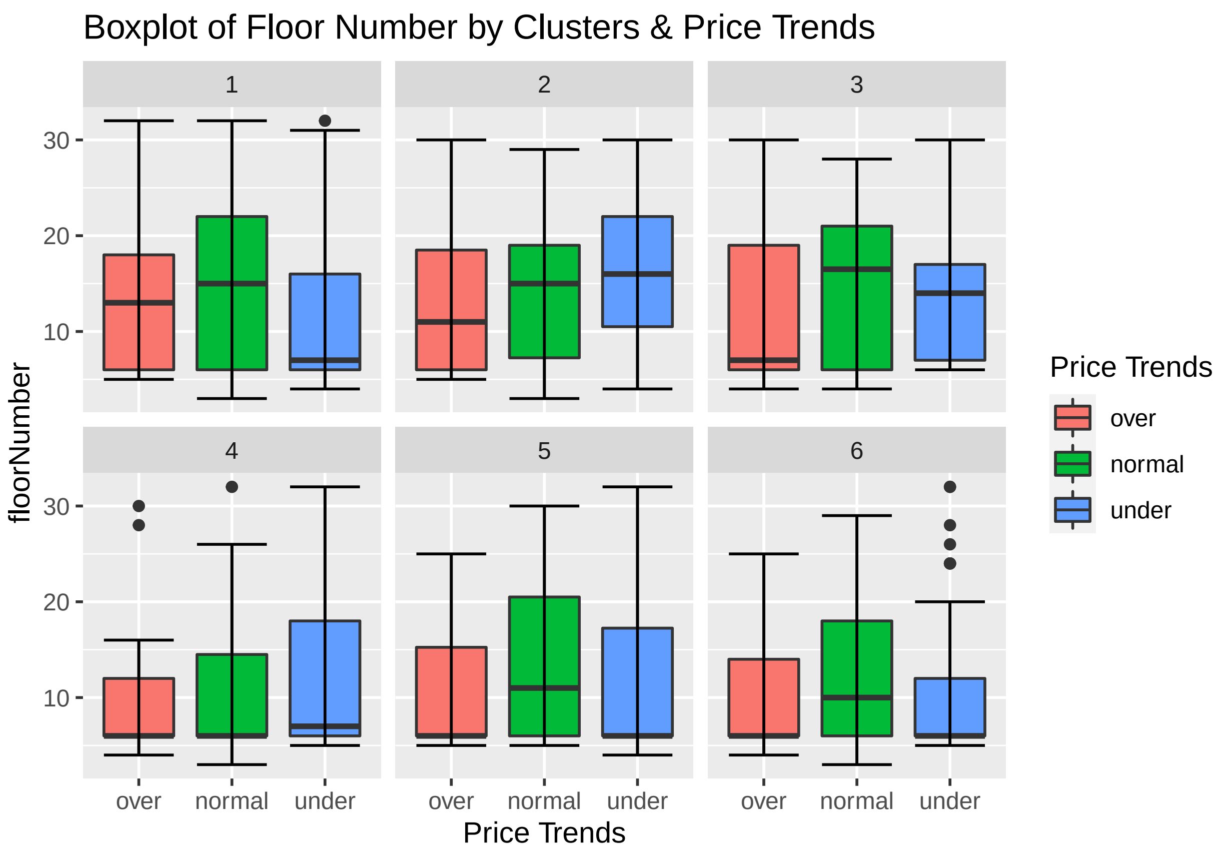
```
ggplot(df.clst, aes(x = price_trend, fill = fiveYearsProperty)) +
  geom_bar() +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Five Years Property", labels=c('0 (No)', '1 (Yes)')) +
  labs(title = "Barplot of Five Years Property by Clusters & Price Trends", x = "Price Trends")
```



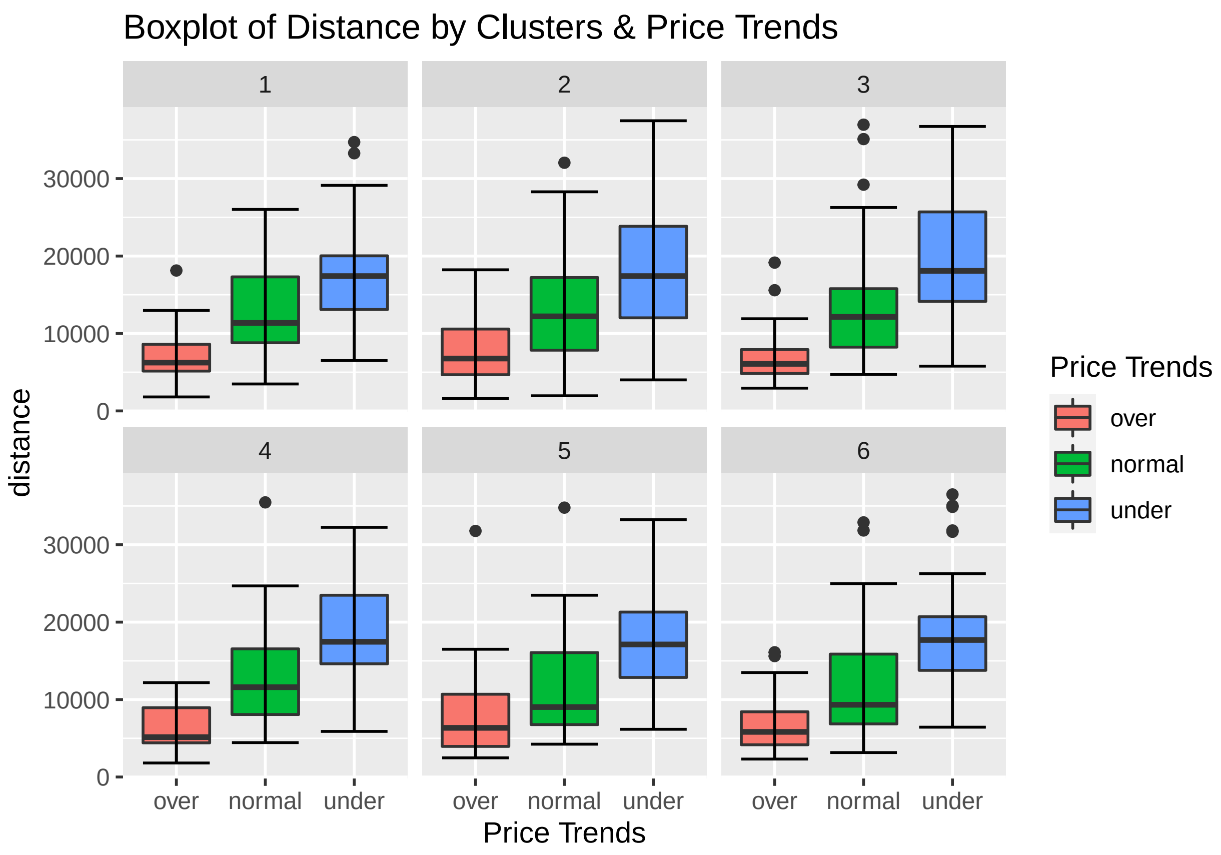
```
ggplot(df.clst, aes(x = price_trend, y = communityAverage, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Community Average by Clusters & Price Trends", x = "Price Trends")
```



```
ggplot(df.clst, aes(x = price_trend, y = floorNumber, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Floor Number by Clusters & Price Trends", x = "Price Trends")
```



```
ggplot(df.clst, aes(x = price_trend, y = distance, fill = price_trend)) +
  geom_boxplot() +
  stat_boxplot(geom = "errorbar") +
  facet_wrap(~clst_label) +
  scale_fill_discrete(name = "Price Trends") +
  labs(title = "Boxplot of Distance by Clusters & Price Trends", x = "Price Trends")
```



Based on the plots above, every cluster displays analogous trends for most of the variables. Few notable points we can find out are:

1) From 'constructionTime', 'over' priced properties tend to be older than the other two price trends ('normal' and 'under'), which is opposite to the analysis we have made above with cluster 3 and 5.
2) 'communityAverage' is indeed highest for 'over', followed by 'normal' and 'under' on average. If assuming the variable as an average income of a community, then this makes sense that the rich would more likely to afford expensive properties.
3) 'floorNumber' for the 'over' priced are relatively lower than the other two price trends.
4) The price trend of 'distance' is found to be identical across the clusters and matches with our prior assumption; 'under' priced properties are relatively far away from the epicenter whereas the 'over' priced ones are closer than the other two price trends in general. This was not the case in the previous analysis based on just cluster labels.

Some possible interpretation would be like below:

With deeper analysis with 'price_trends', we were able to find some interesting points. Although many people would think that properties that were built relatively recently (newer) would be more expensive, this is only true in the analysis with only cluster labels. When further divide by price trend, we can see that the result is the opposite. There may be many factors about this, and one possible reason from urban planning perspective would be that a city is usually developed from the inside (center) to outside (suburb), which can be tied with 'distance' as well. Considering that most crucial infrastructures such as public/social service are built around the epicenter area, it may be obvious that the value and need of the properties near these area would also increase regardless of their age. This trend is perfectly reflected in the last plot of 'distance', with all clusters having 'over' price properties to be the closest and 'under' price properties to be the farthest from the epicenter.

As a result, some common beliefs and trends related to housing/property market were detected from this analysis. However, one caveat we need to be aware of is that the data is from 2011 to 2017, which might not reflect the current trend in Beijing. Also, some other crucial factors that are relevant to the field such as policies should be considered as well in order to understand the overall picture better.

Limitations

We have faced some critical issues during the analyses, and below are the possible limitations:

1) Representativeness

- Due to expensive computational cost and limitation of our machines, we had no choice but to randomly sample only 1000 observations with 10 variables from the original dataset. Based on this, it is possible to question whether the sample is representative of the original dataset and therefore not sufficiently reflecting the truth and possibly coming up with somewhat distorted analyses.

2) Lack diversity

- The clustering algorithms we have learned in the class were 'Kmeans' and 'hierarchical' clustering. However, since hierarchical clustering is not recommended for large data, we were only able to work on the clustering analysis with 'Kmeans', which may lack in terms of comparing a result with other clustering algorithms.

Some notable landmarks & places beside the epicenter:

- GPS coordinates from https://latitude.to/articles-by-country/

1) Jingshan park 39.92600108466268 116.3966463362935
2) Beijing Central Business District 39.9085 116.453
3) Forbidden city 39.908829698 116.387665116
4) Sanlitun 39.93416293 116.443248227 (District with many bar streets & restaurants, Good area for nightlife)
5) Tsinghua University 40.0 116.322665376