USStates

Dambar

4/12/2023

Loading data

```
df <- read_csv("E:/kmeans.csv")
```

```
## Rows: 51 Columns: 17
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): state
## dbl (16): emp, output_worker, price_def, MinWage, Poverty_pc, poverty_18, Me...
##
## 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.
```

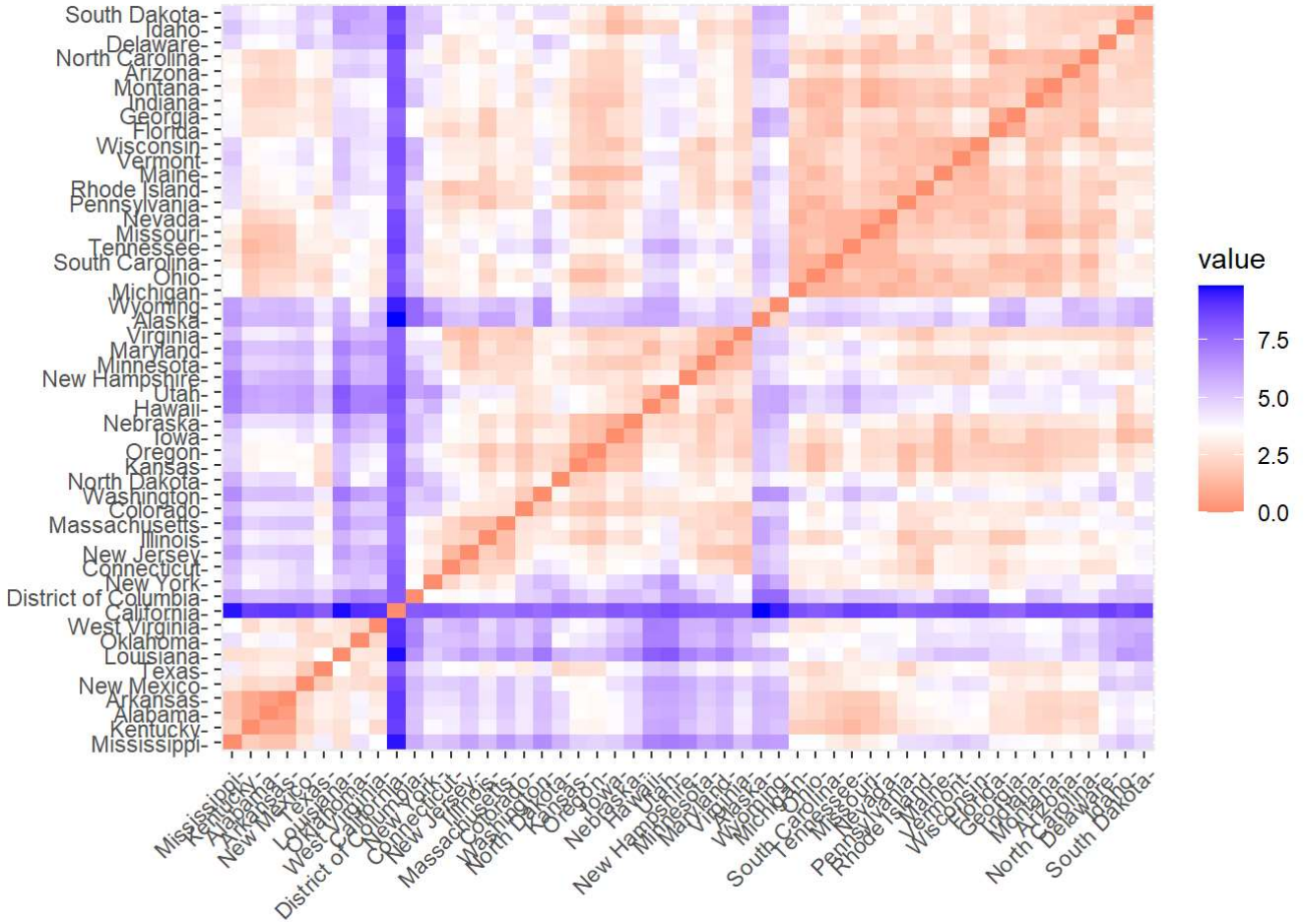
```
View(df)
```

```
t(t(names(df)))
```

```
##       [,1]
##  [1,] "state"
##  [2,] "emp"
##  [3,] "output_worker"
##  [4,] "price_def"
##  [5,] "MinWage"
##  [6,] "Poverty_pc"
##  [7,] "poverty_18"
##  [8,] "Med_hh_Income"
##  [9,] "GDP_per_capita"
## [10,] "pop"
## [11,] "gini"
## [12,] "happiness"
## [13,] "unemp_rate"
## [14,] "growth"
## [15,] "permanent res"
## [16,] "nonimm"
## [17,] "naturalization"
```

I am selecting the desired columns and storing the data in variable df1

```
df1 <- df[,c(1:4,6,8,11,12,14)]
df2<- as.data.frame(df1)
row.names(df2)<-df2[,1]
df3 <- df2[,-1]
df4 <-scale(df3)
distance <- get_dist(df4)
fviz_dist(distance)
```

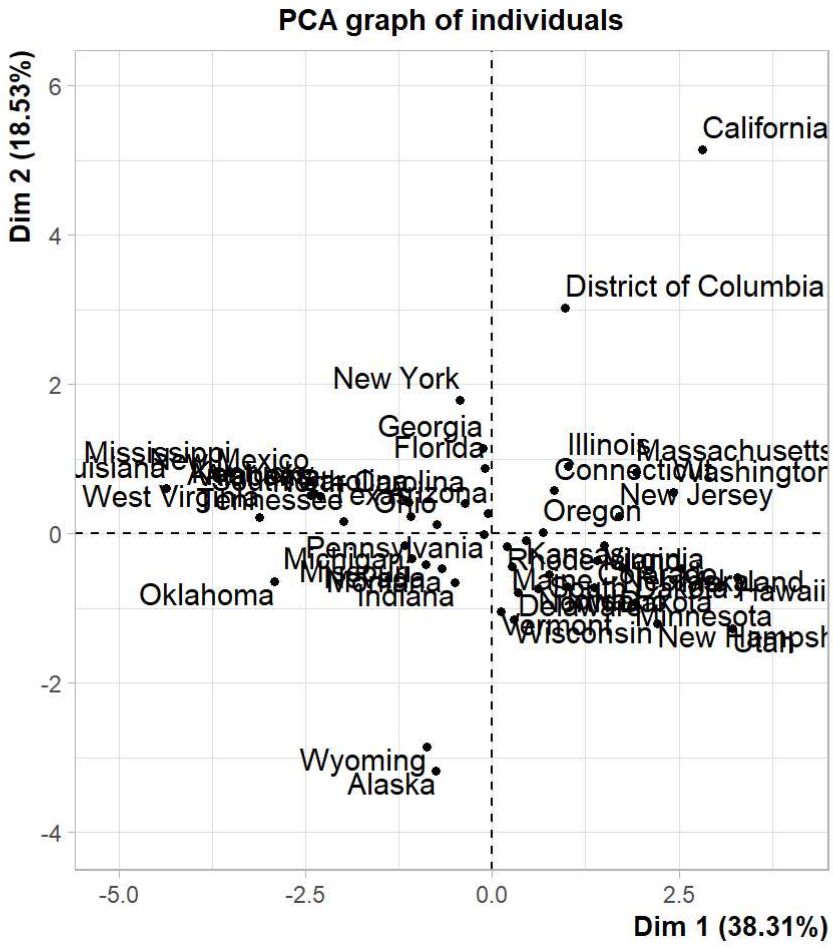


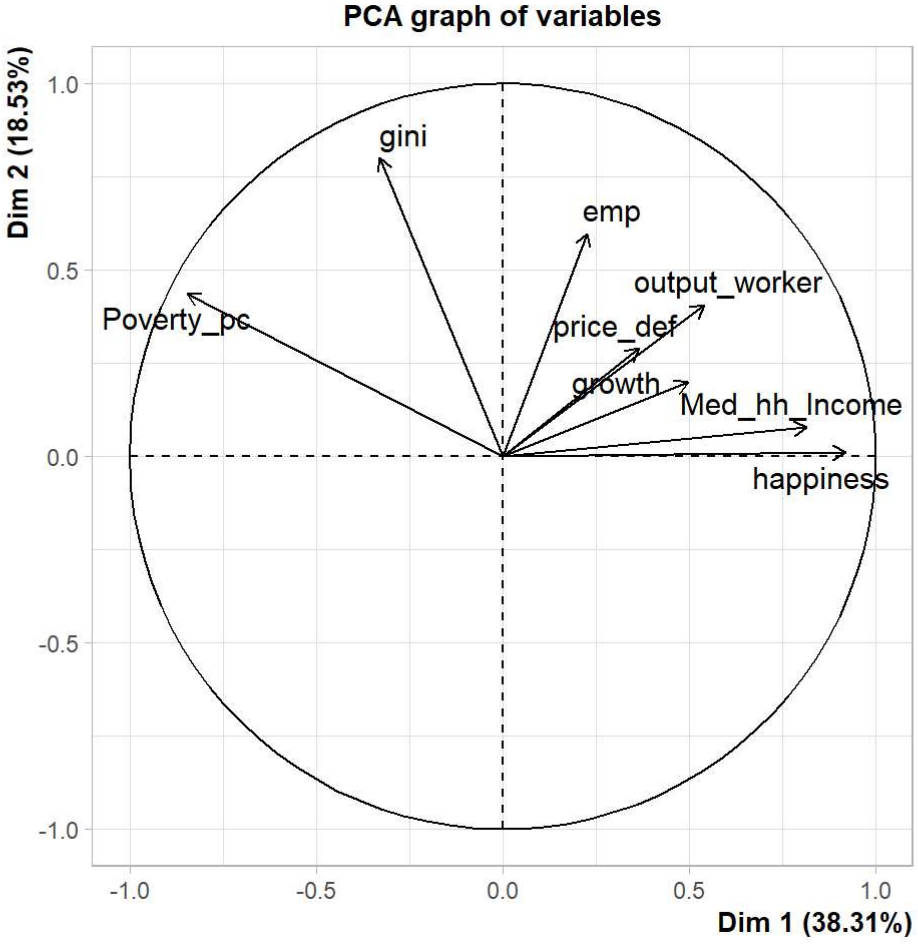
Correlation matrix to examine the relationships between variables

```
library(ggcorrplot)
corr <- cor(df4)
ggcorrplot(corr, outline.color = "grey50", lab = TRUE, hc.order = TRUE, type = "full")
```

Determining the relative importance of the primary variables in the data set using principal component analysis. Here, I'm thinking that four is the ideal number for a cluster.

```
library(FactoMineR)
pca <- PCA(df4)
```

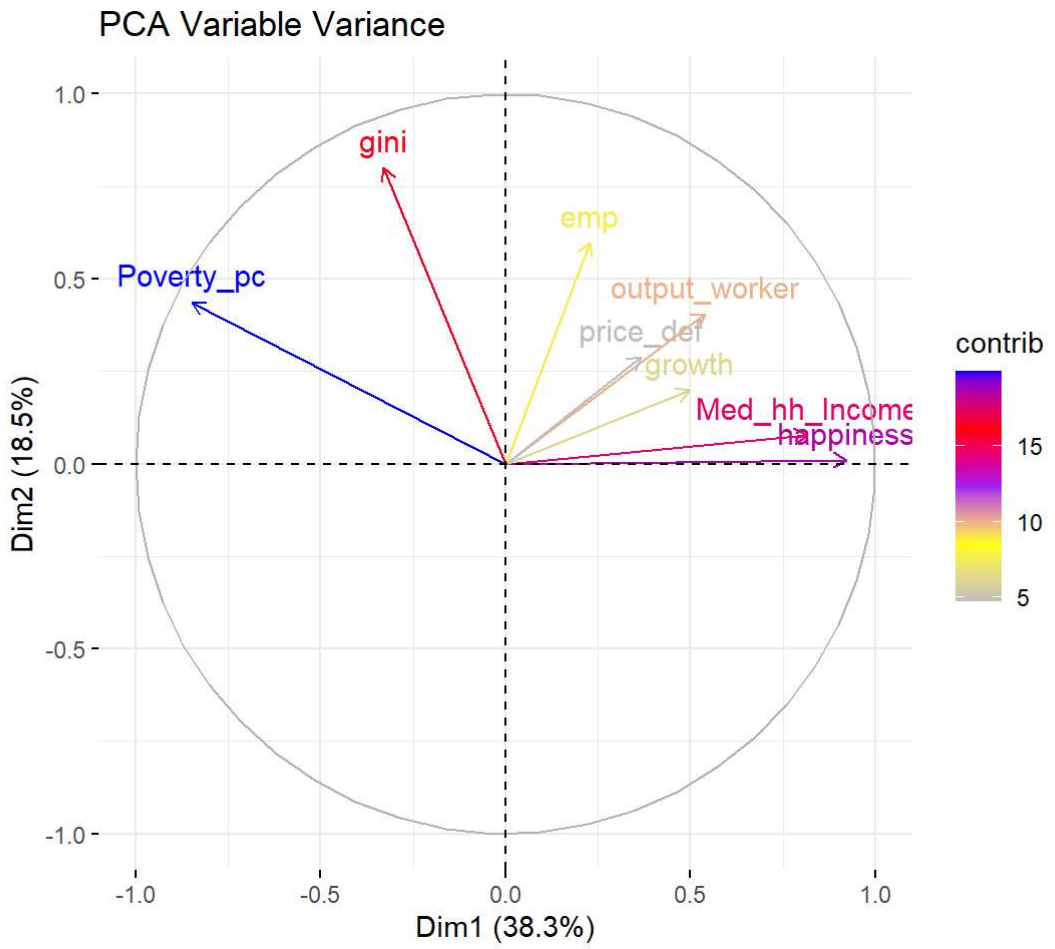

PCA graph of individuals
Dim 2 (18.53%)
Dim 1 (38.31%)
California
District of Columbia
New York
Georgia
Florida
Illinois
Massachusetts
Connecticut
Washington
New Jersey
Oregon
Ohio
Pennsylvania
Oklahoma
Indiana
Wyoming
Alaska
6
4
2
0
-2
-4
-5.0
-2.5
0.0
2.5


PCA graph of variables
Dim 2 (18.53%)
Dim 1 (38.31%)
gini
emp
output_worker
Poverty_pc
price_def
growth
Med_hh_Income
happiness
1.0
0.5
0.0
-0.5
-1.0
-1.0
-0.5
0.0
0.5
1.0

```
# assuming your data is in a data frame called "df" with 8 variables
# perform principal component analysis
pca <- prcomp(df4, scale = TRUE)

# extract loadings
loadings <- pca$rotation

# print loadings for the first two PCs
print(loadings[, 1:2])
```

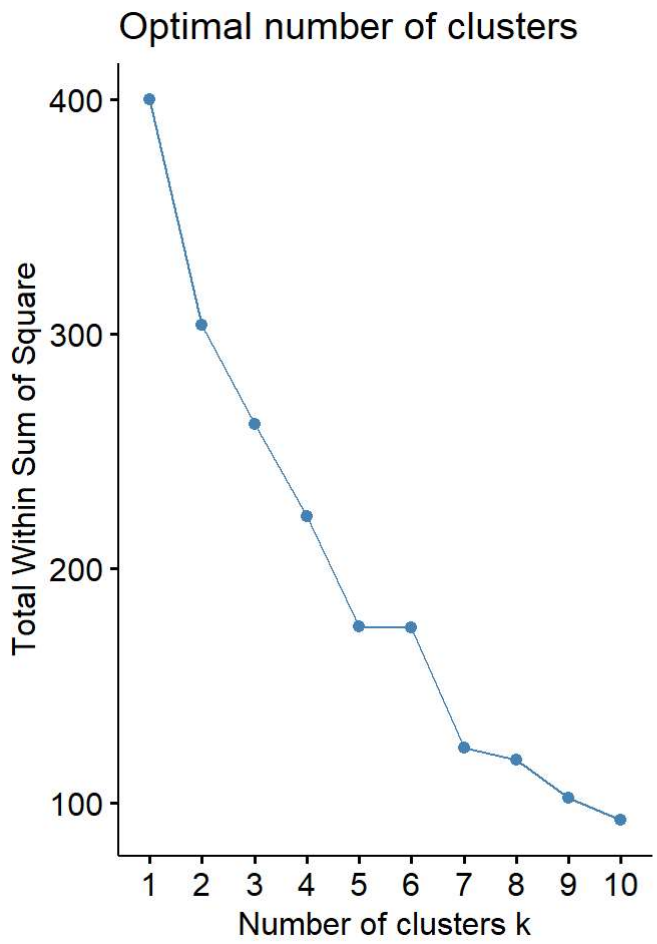
```
##                      PC1         PC2
## emp            0.1300093 0.491185040
## output_worker  0.3086066 0.333608135
## price_def      0.2089807 0.237628078
## Poverty_pc    -0.4838334 0.358350704
## Med_hh_Income  0.4660202 0.064146757
## gini          -0.1885346 0.657025271
## happiness      0.5259218 0.008469762
## growth         0.2842148 0.163376215
```

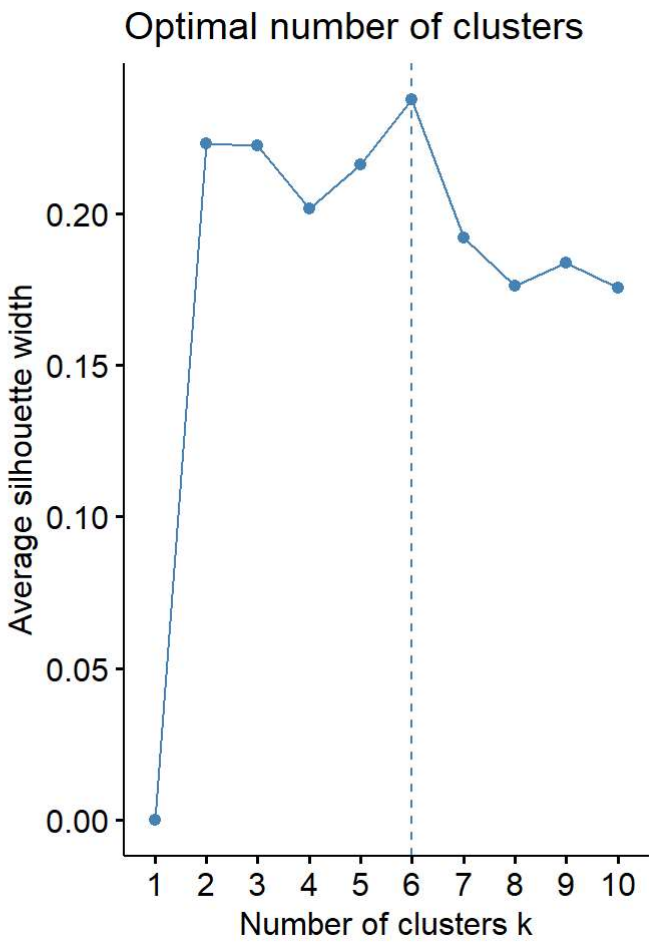
```
var <- get_pca_var(pca)
fviz_pca_var(pca, col.var="contrib",
gradient.cols = c("grey","yellow","purple","red","blue"),ggrepel = TRUE ) + labs( title = "PCA V
ariable Variance")
```



From PCA Variable Variance, we can infer that employment, labor productivity, output per worker, price deflator, and state minimum wage contribute more than 61 % to the two PCA components/dimensions (Variables). I'm utilizing the elbow approach to get the ideal customer count.

```
library(cowplot)

Elbow_method <- fviz_nbclust(df4, kmeans, method = "wss")
Silhouette <- fviz_nbclust(df4, kmeans, method = "silhouette")
plot_grid(Elbow_method, Silhouette, nrow = 1)
```





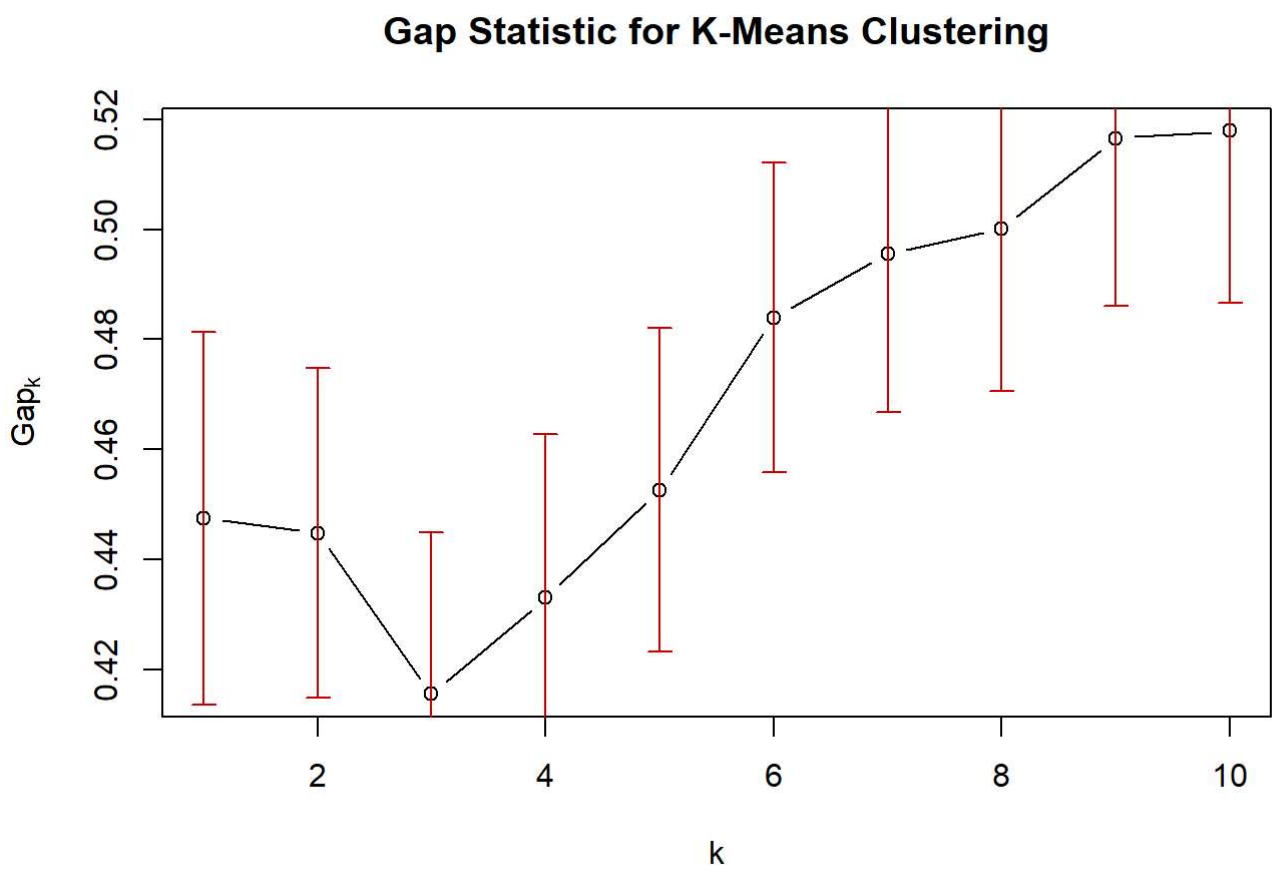
The Gap Statistic is a method for determining the optimal number of clusters (k) in a dataset. In R, the clusGap() function from the cluster package can be used to compute the Gap Statistic for a given range of k values.

Using the Gap Statistic for a robustness check is a good idea. Both the WSS and Silhouette methods suggest k=6, and the Gap Statistic plot shows a slope change at k=6, this could provide additional support for choosing k=6 as the optimal number of clusters.

However, it's important to keep in mind that the Gap Statistic is just one method for choosing the number of clusters, and its results should be considered alongside other methods and the domain knowledge. It's also worth noting that the interpretation of the Gap Statistic can be somewhat subjective, as there is no hard and fast rule for determining the optimal number of clusters based on the Gap Statistic plot.

Overall, combining multiple methods for choosing the number of clusters and performing robustness checks can help increase confidence in the results and ensure that the clustering is meaningful and useful for your analysis.

```
library(cluster)

gap <- clusGap(df4, FUN = kmeans, nstart = 25, K.max = 10, B = 50)

plot(gap, main = "Gap Statistic for K-Means Clustering")
```



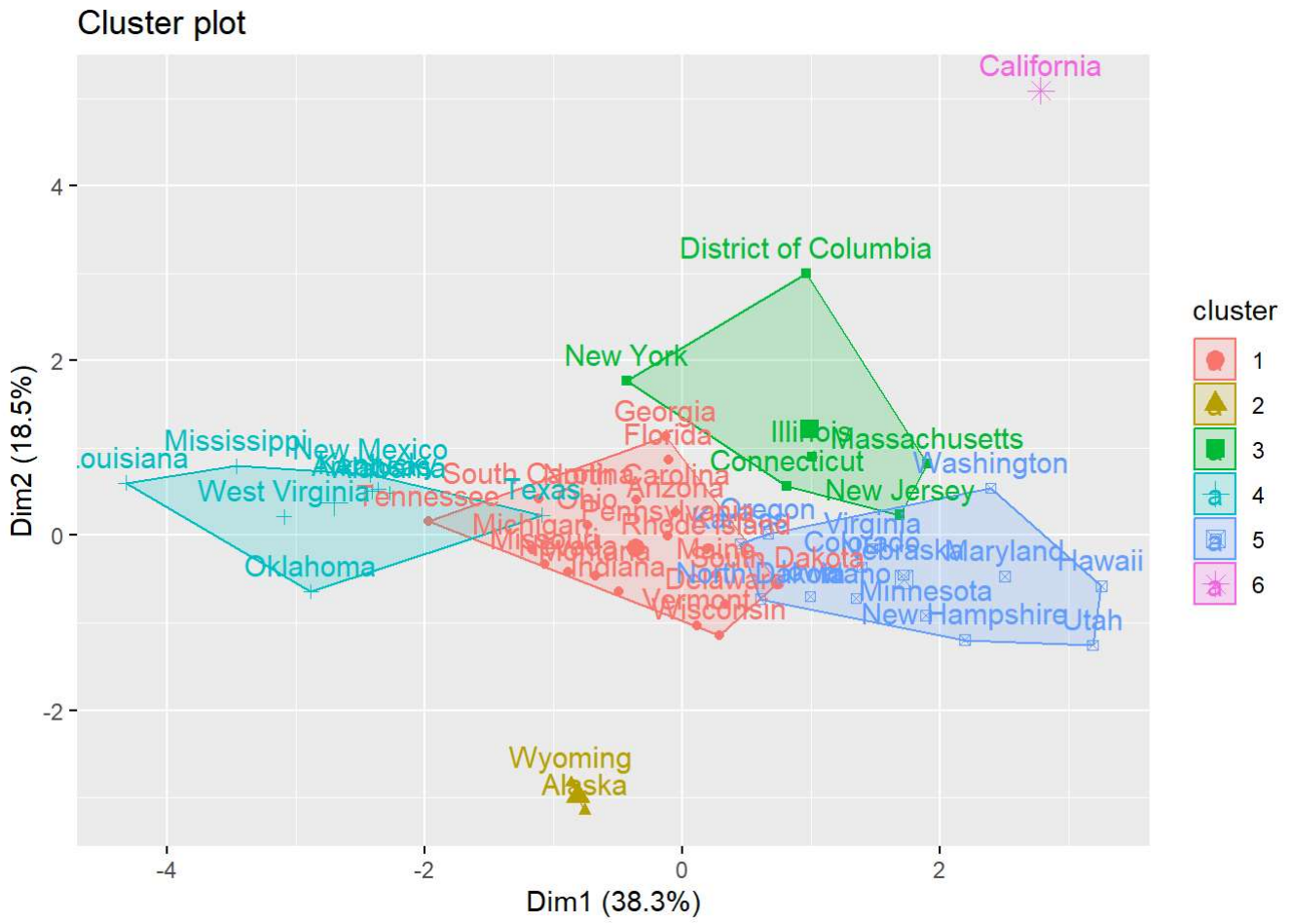

```
k6 <- kmeans(df4, centers = 6, nstart = 25) # k = 3, number of restarts = 25
k6$centers
```

```
##          emp output_worker   price_def  Poverty_pc Med_hh_Income        gini
## 1 -0.1390382    -0.3812199  0.49868111  0.10268289    -0.3956147 -0.07730284
## 2 -0.1455117    -1.9039403 -2.71772492 -0.87418682     0.5704272 -1.64989472
## 3 -0.1419514     0.4362798  0.60645624 -0.18534261     1.2860715  1.55523933
## 4 -0.1407885    -0.4328023 -0.93226577  1.55802853    -1.2070866  0.61314907
## 5 -0.1392711     0.7065150  0.06386318 -0.93159050     0.5806893 -0.80071030
## 6  7.0013510     2.4373896 -0.18192141 -0.07053525     1.3935242  1.12870995
##    happiness     growth
## 1 -0.1227793 -0.1680591
## 2 -0.3000055 -1.6741087
## 3  0.8393517 -0.3114924
## 4 -1.5702734 -0.3410342
## 5  0.7817592  0.7899954
## 6  1.0845380  0.4196676
```

```
k6$size
```

```
## [1] 19  2  6  9 14  1
```

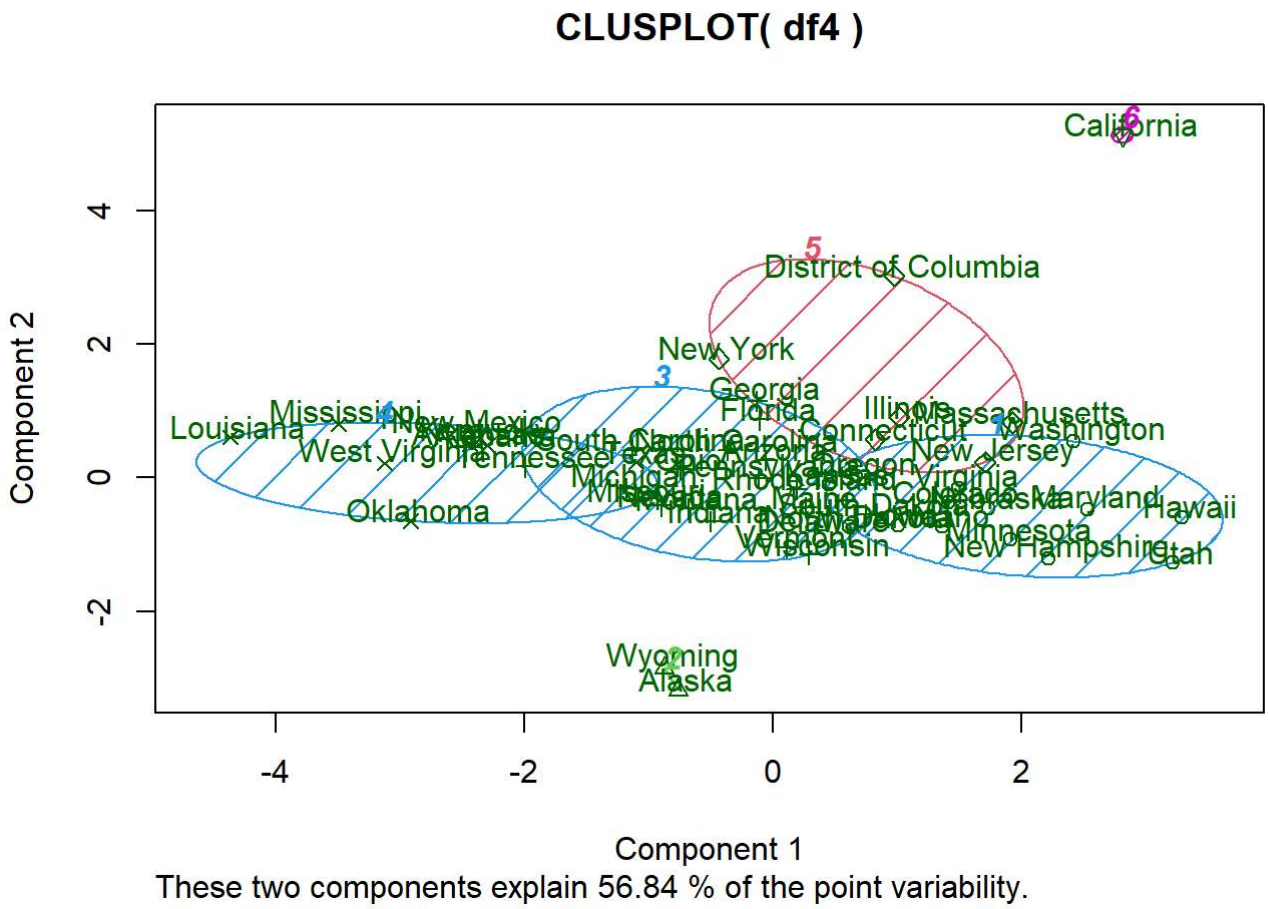
```
fviz_cluster(k6, data = df4)
```



There are 6 clusters in the cluster graph above, each of which has a unique color and form. The center point, is the cluster's centriod. After 25 Iterations there is no change in the clusters centroid untill and unless a new datapoint is added.

It's worth noting that cluster plots are often two-dimensional representations of high-dimensional data, and it's possible that some overlap may occur even if the clustering is meaningful and well-separated in the higher dimensions.

```
cluster <- kmeans(df4,6)
clusplot(df4, cluster$cluster, color=TRUE, shade=TRUE, labels=2, lines=0)
```



In the above, We have selected the numeric variables present in the dataset. After selecting we have applied Euclidean distance formula which is the default distance metric to calculate the distance. After that we have to normalized the data using range and scale method. After doing all these steps we found the best K using Silhouette Method. We have performed K means clustering algorithm with K=5. After performing the K means we have visualized the clustering analysis using fviz_cluster()

To examine the patterns by visualizing clusters against the variables grouped by clusters and to identify any trends in the data

```
library(magrittr)
```

```
## Warning: package 'magrittr' was built under R version 4.1.3
```

```
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##     set_names
```

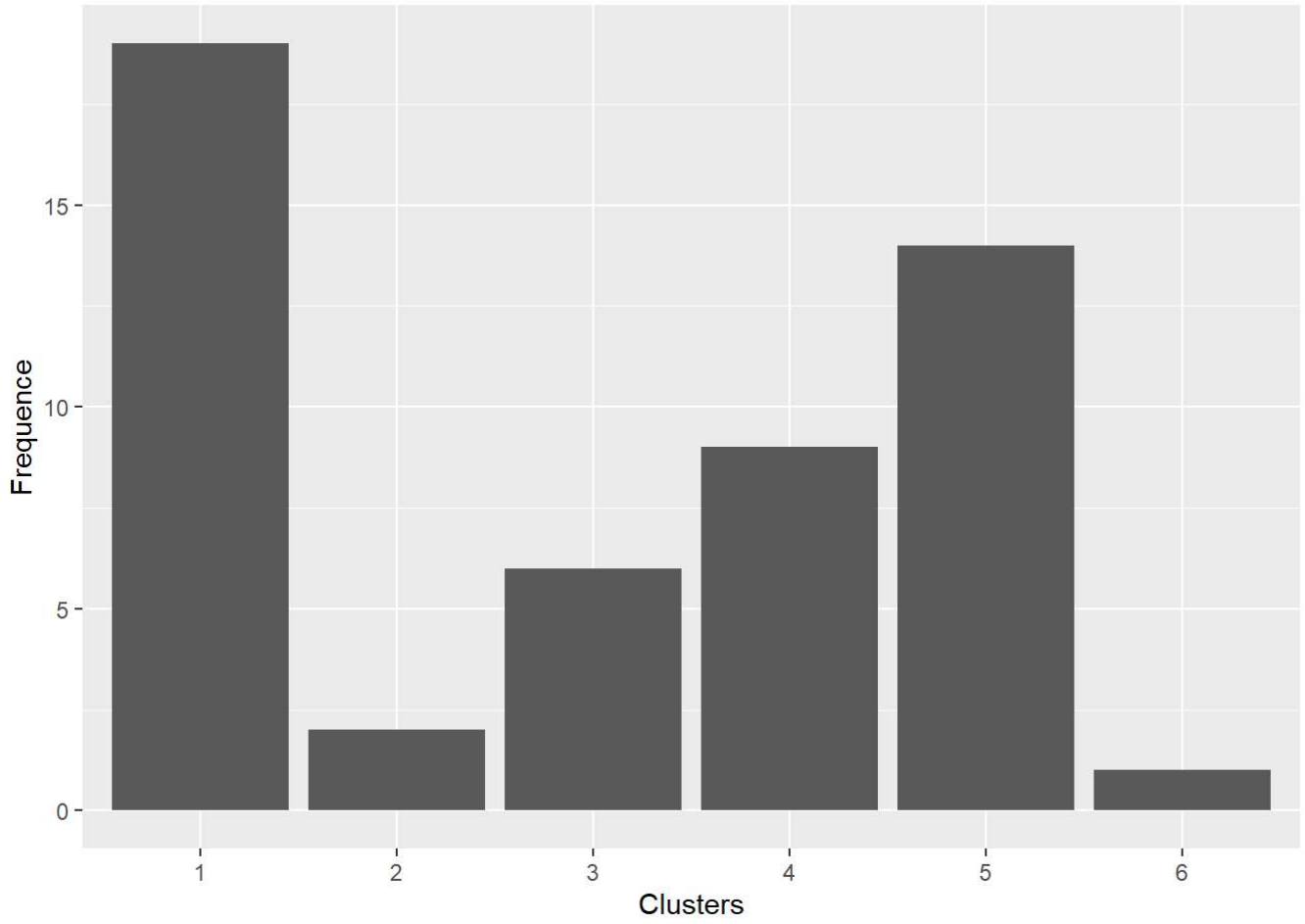
```
## The following object is masked from 'package:tidyr':
##
##     extract
```

```
library(dplyr)
pattern <- df %>% select(c(10,13,16)) %>% mutate(cluster = k6$cluster)
print(pattern)
```

```
## # A tibble: 51 x 4
##         pop unemp_rate  nonimm cluster
##       <dbl>      <dbl>   <dbl>   <int>
##  1  5031362        6.4   49893       4
##  2   732923        8.3   48380       2
##  3  7179943        7.8  486163       1
##  4  3014195        6.2   29548       4
##  5 39501653       10.1 1859671       6
##  6  5784865        6.8  167072       5
##  7  3597362        7.9   71392       3
##  8   992114        7.5   13458       1
##  9   670868        7.9   82283       3
## 10 21589602        8.1 3157503       1
## # i 41 more rows
```

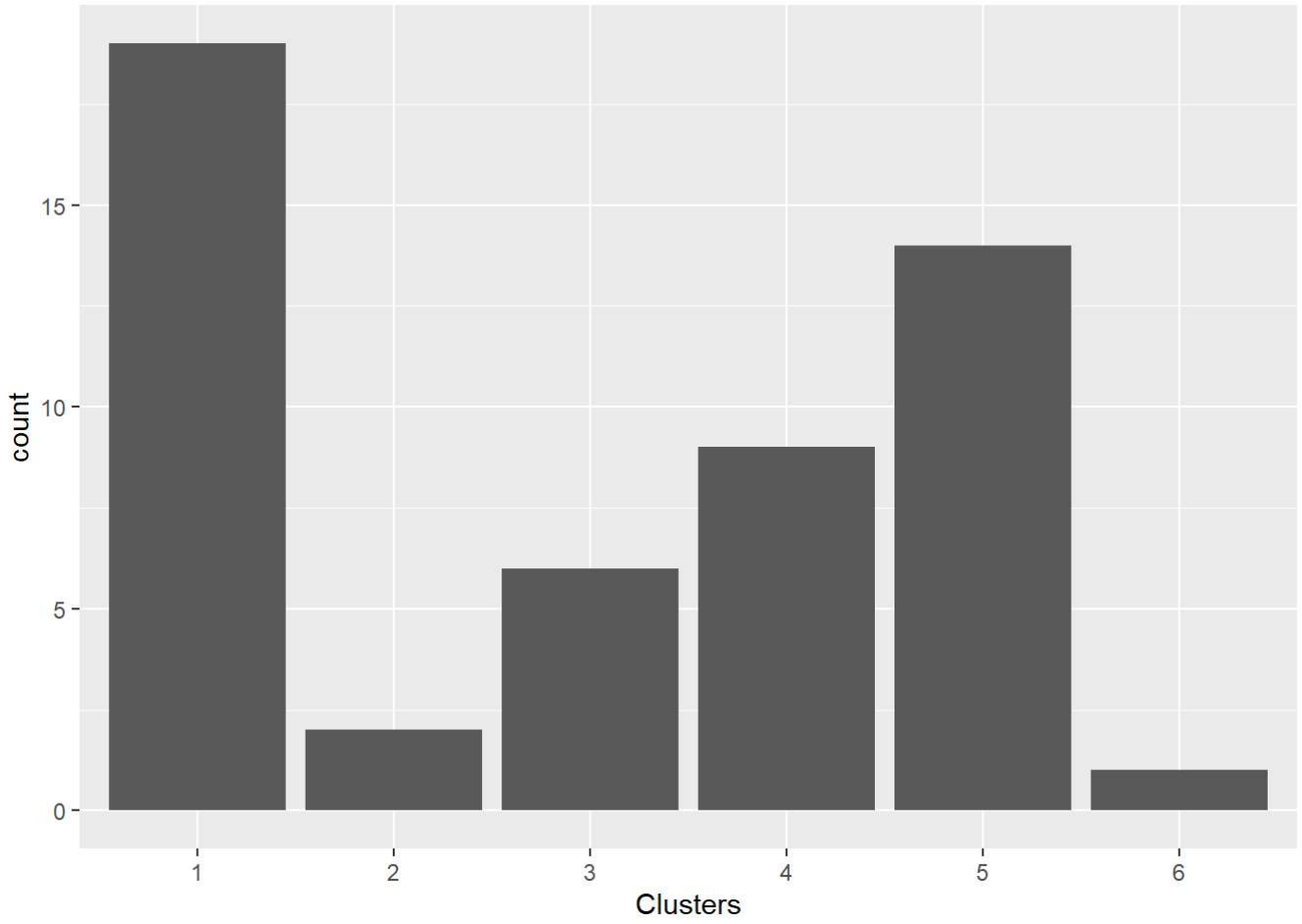
```
population <- ggplot(pattern, mapping = aes(factor(cluster),
fill=pop)) +
geom_bar(position = 'dodge') +labs(x='Clusters', y='Frequence')
population
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



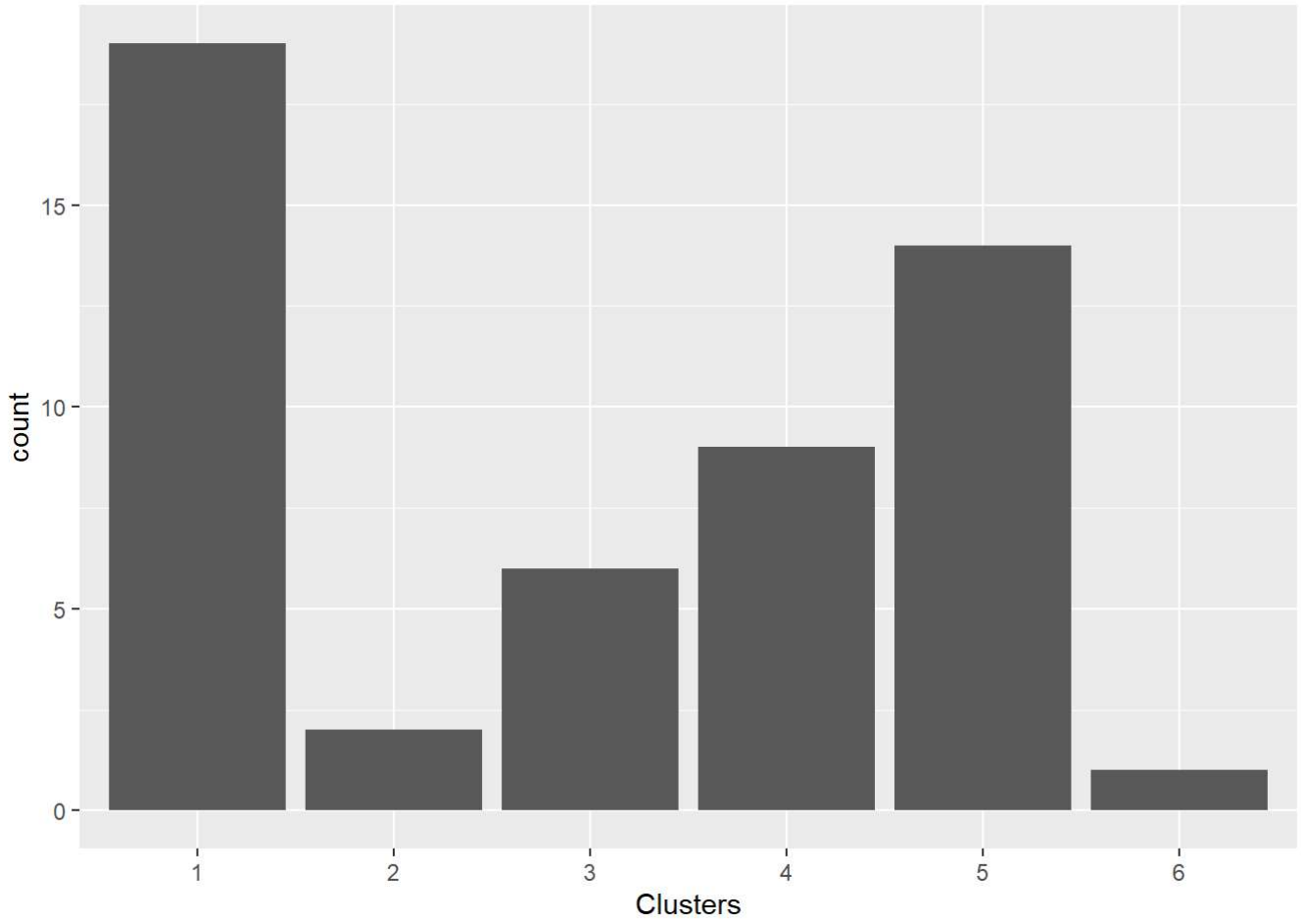

```
unemployment <- ggplot(pattern, mapping = aes(factor(cluster),fill = unemp_rate))+
geom_bar(position = 'dodge')+
labs(x ='Clusters')
unemployment
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```




```
non_immigrant <- ggplot(pattern, mapping = aes(factor(cluster),fill = nonimm))+
geom_bar(position = 'dodge')+
labs(x ='Clusters')
non_immigrant
```

```
## Warning: The following aesthetics were dropped during statistical transformation: fill
## i This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## i Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



Hierarchical clustering using complete linkage

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

```
## Warning: package 'dendextend' was built under R version 4.1.3
```

```
##
## ---------------------
## Welcome to dendextend version 1.17.1
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issu
es
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ---------------------
```

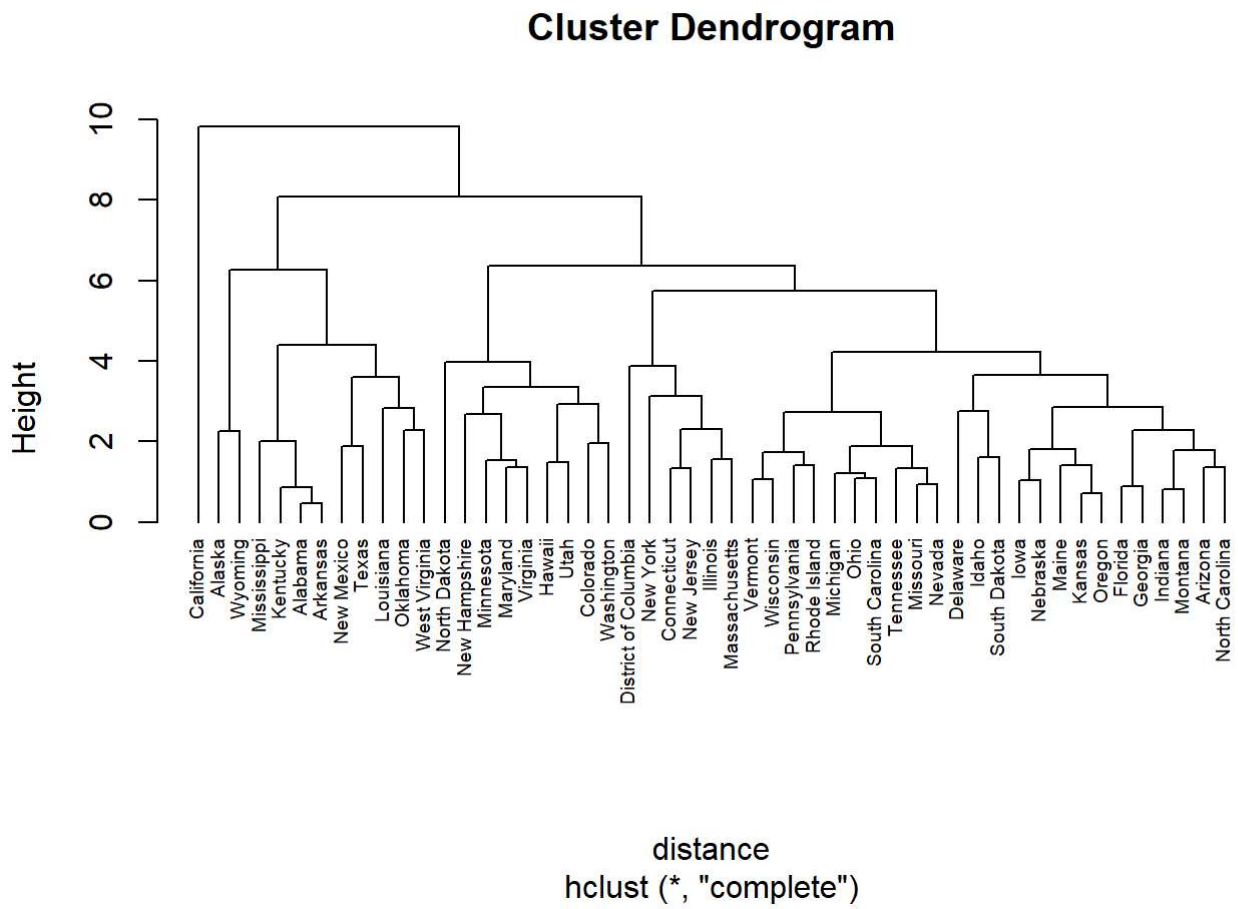
```
##
## Attaching package: 'dendextend'
```

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

Dendrogram: A dendrogram is a graphical representation of the hierarchical clustering process that can help you visualize the number of clusters in the dataset. The height of the branches in the dendrogram corresponds to the distance between the clusters at each step of the clustering process.

We can set a threshold height for the dendrogram that corresponds to the maximum distance between clusters that we want to allow. This method involves setting a distance threshold on the y-axis of the dendrogram and cutting the dendrogram at that point. The height (Y axis) provides the distance between the clusters. If we use height 4, there are 6 clusters.

```
hc1<-hclust(distance, method = "complete")
#plot(hc1)
plot(hc1, hang = -1, cex = 0.6)
```




```
# Adjust the plot size
#par(mar = c(5, 7, 5, 5))  # Set the margin size
#par(pin = c(8, 4))  # Set the plot size
```

Alternatively, we can use the agnes()function

```
hc_single<-agnes(df4, method = "single")
hc_complete<-agnes(df4, method = "complete")
hc_average<-agnes(df4, method = "average")
```

Compare Agglomerative coefficients

```
print(hc_single$ac)
```

```
## [1] 0.7905299
```

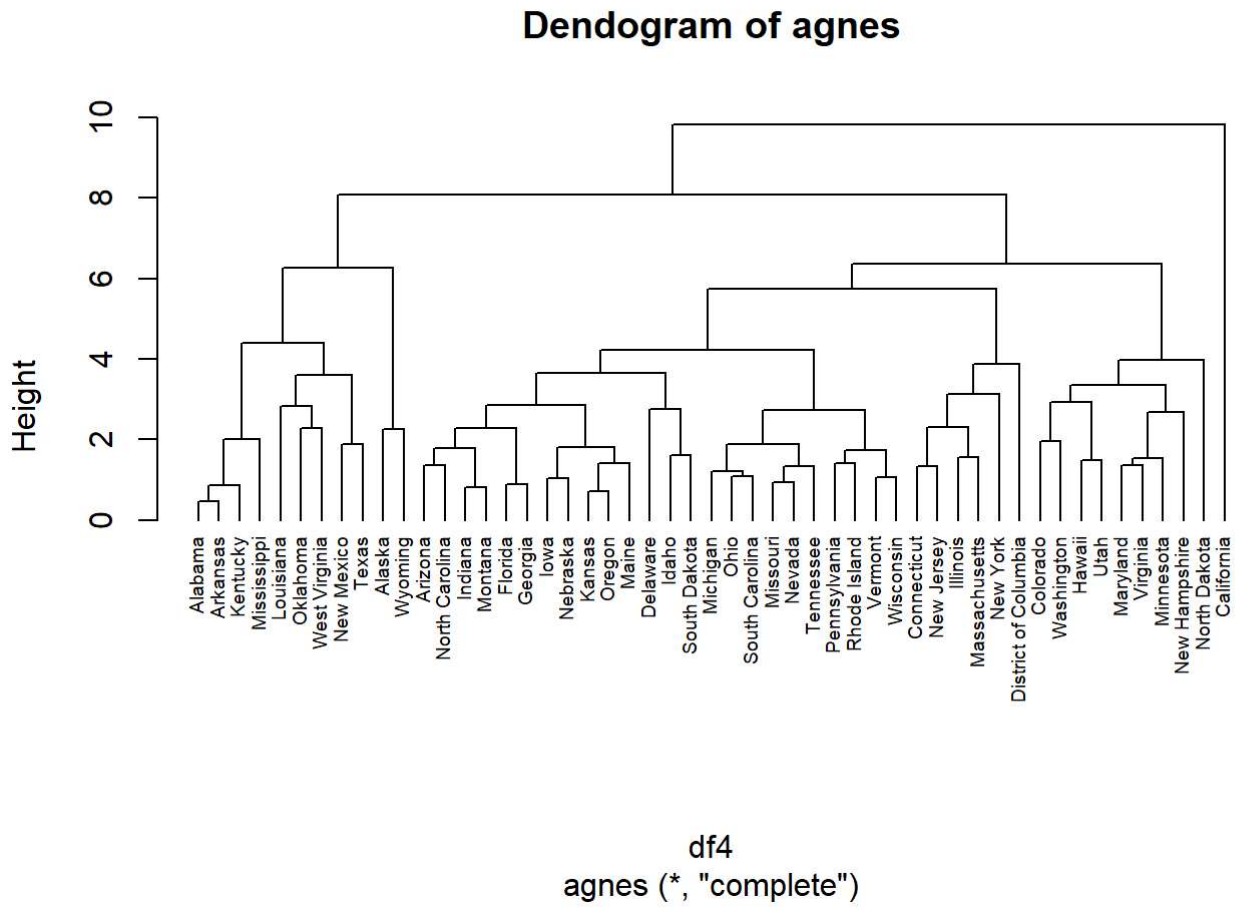
```
print(hc_complete$ac) # Best linkage method
```

```
## [1] 0.8234411
```

```
print(hc_average$ac)
```

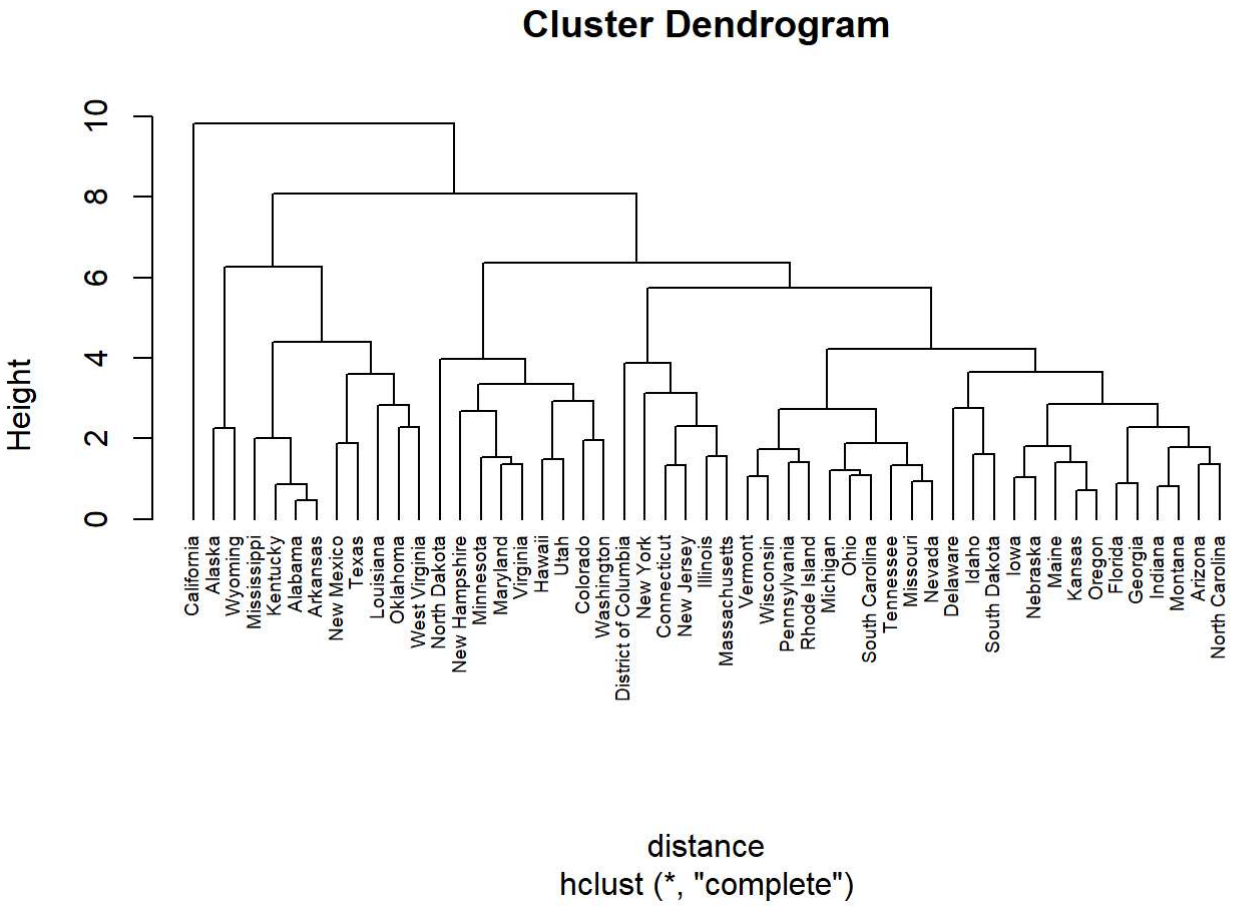
```
## [1] 0.7966656
```

```
pltree(hc_complete, cex=0.6, hang = -1, main = "Dendogram of agnes")
```



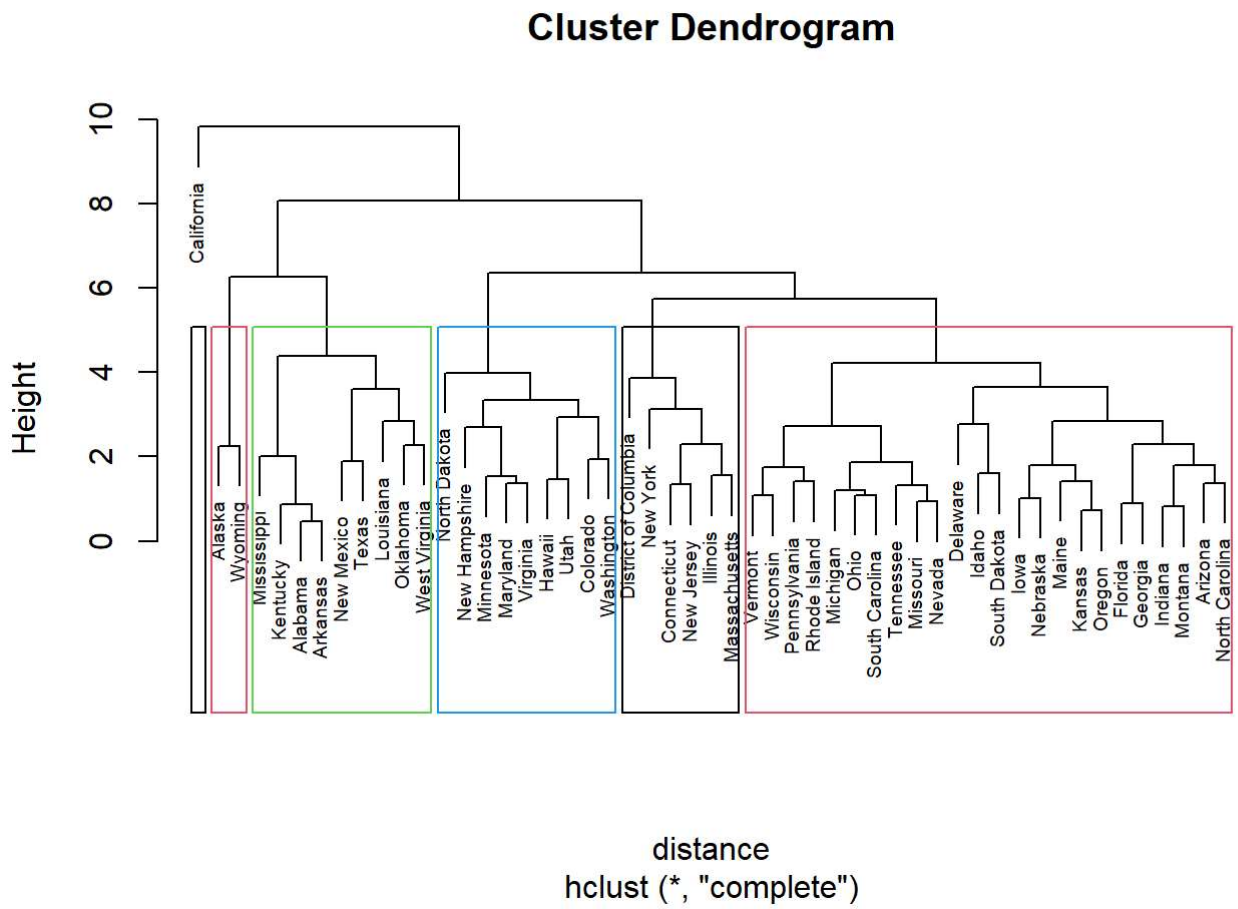
Notice that the “complete” linkages provides the strongest clustering structure.

```
plot(hc1, hang = -1, cex = 0.6)
```



It’s also possible to draw the dendrogram with a border around the 6 clusters. The argument border is used to specify the border colors for the rectangles.

```
hc_complete <- hclust(distance,method = "complete")
# plot dendrogram
plot(hc_complete, cex = 0.6)
rect.hclust(hc_complete, k = 6, border = 1:4)
```

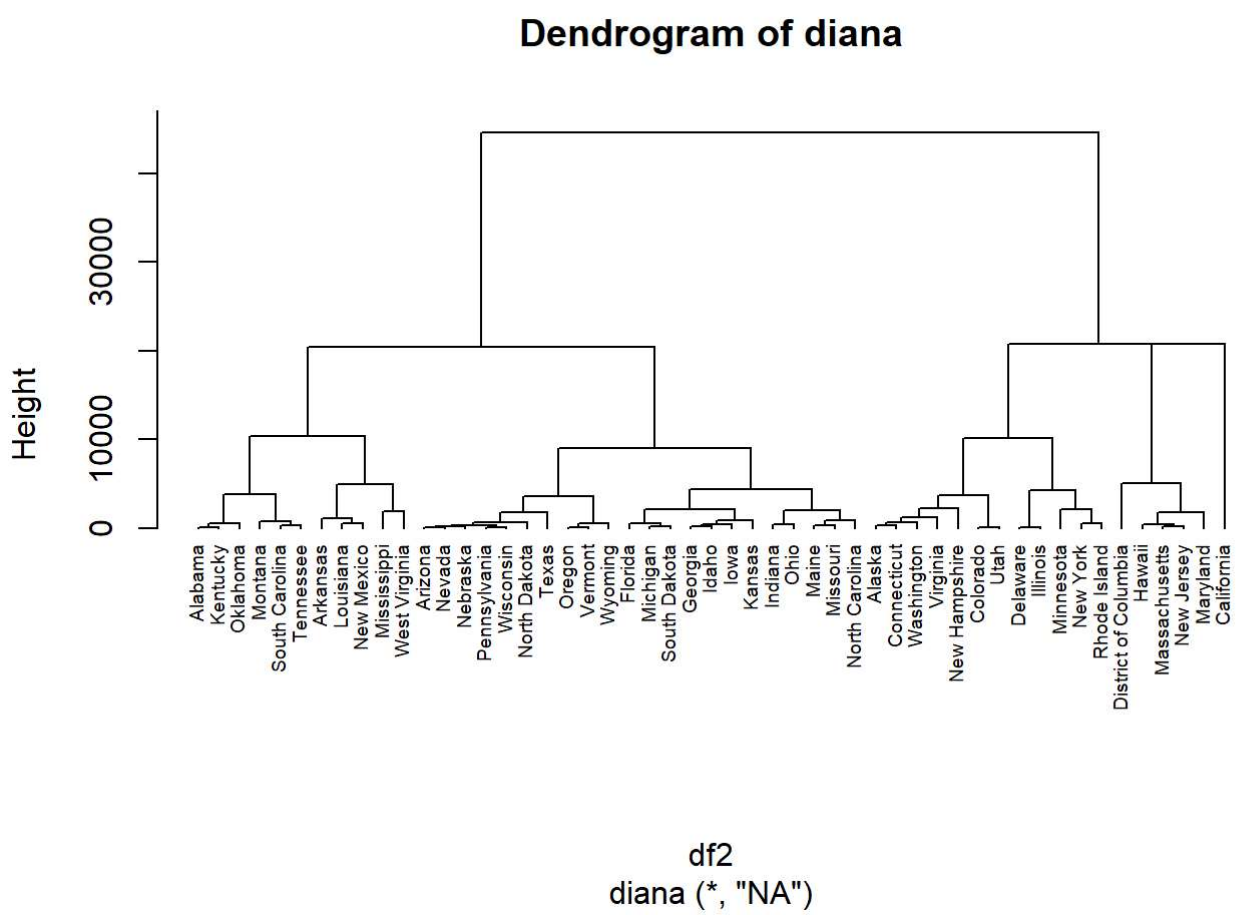


The R function diana provided by the cluster package allows us to perform divisive hierarchical clustering. Diana works similar to agnes; however, there is no method to provide.

```
# compute divisive hierarchical clustering
hc_diana <- diana(df2)
# Divise coefficient; amount of clustering structure found
hc_diana$dc
```

```
## [1] 0.9767448
```

```
## [1] 0.8146836
# plot dendrogram
pltree(hc_diana, cex = 0.6, hang = -1, main = "Dendrogram of diana")
```



The Devise coefficient is a measure of the amount of clustering structure found in the data by the clustering algorithm. It takes values between 0 and 1, with higher values indicating a stronger clustering structure in the data.In this case, the Devise coefficient is 0.8146836, which suggests that the clustering algorithm has found a significant amount of clustering structure in the data. This value can be used to compare the clustering solutions obtained using different methods or parameter settings to select the one that best captures the underlying structure in the data.

Hierarchical k-means clustering

The final k-means clustering solution is very sensitive to the initial random selection of cluster centers. This function provides a solution using an hybrid approach by combining the hierarchical clustering and the k-means methods.

hkmeans(): compute hierarchical k-means clustering print.hkmeans(): prints the result of hkmeans hkmeans_tree(): plots the initial dendrogram

Compute hierarchical k-means clustering

```
res.hk <-hkmeans(df4, 6)
# Elements returned by hkmeans()
names(res.hk)
```

```
##  [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
##  [6] "betweenss"    "size"         "iter"         "ifault"       "data"
## [11] "hclust"
```

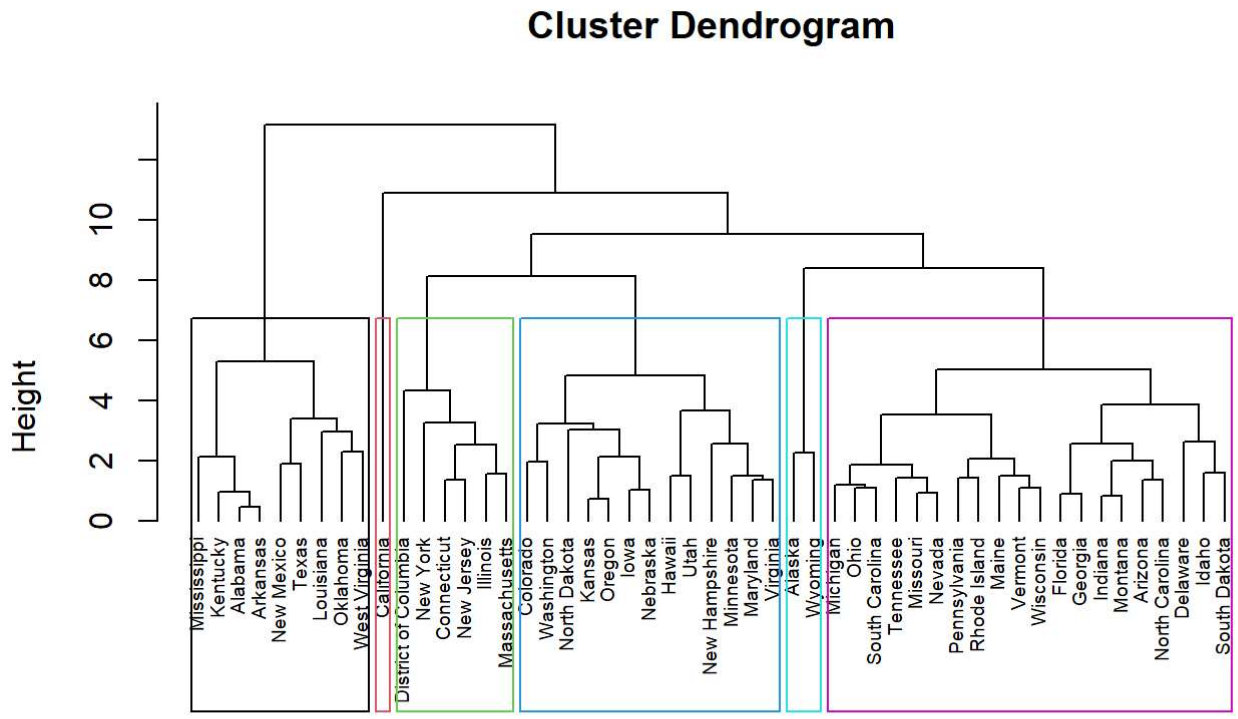
```
# Print the results
res.hk
```

```
## Hierarchical K-means clustering with 6 clusters of sizes 9, 2, 19, 1, 14, 6
##
## Cluster means:
##          emp output_worker   price_def  Poverty_pc Med_hh_Income        gini
## 1 -0.1407885    -0.4328023 -0.93226577  1.55802853    -1.2070866  0.61314907
## 2 -0.1455117    -1.9039403 -2.71772492 -0.87418682     0.5704272 -1.64989472
## 3 -0.1390382    -0.3812199  0.49868111  0.10268289    -0.3956147 -0.07730284
## 4  7.0013510     2.4373896 -0.18192141 -0.07053525     1.3935242  1.12870995
## 5 -0.1392711     0.7065150  0.06386318 -0.93159050     0.5806893 -0.80071030
## 6 -0.1419514     0.4362798  0.60645624 -0.18534261     1.2860715  1.55523933
##    happiness     growth
## 1 -1.5702734 -0.3410342
## 2 -0.3000055 -1.6741087
## 3 -0.1227793 -0.1680591
## 4  1.0845380  0.4196676
## 5  0.7817592  0.7899954
## 6  0.8393517 -0.3114924
##
## Clustering vector:
##              Alabama               Alaska              Arizona
##                    1                    2                    3
##             Arkansas           California             Colorado
##                    1                    4                    5
##          Connecticut             Delaware District of Columbia
##                    6                    3                    6
##              Florida              Georgia               Hawaii
##                    3                    3                    5
##                Idaho             Illinois              Indiana
##                    5                    6                    3
##                 Iowa               Kansas             Kentucky
##                    5                    5                    1
##            Louisiana                Maine             Maryland
##                    1                    3                    5
##        Massachusetts             Michigan            Minnesota
##                    6                    3                    5
##          Mississippi             Missouri              Montana
##                    1                    3                    3
##             Nebraska               Nevada        New Hampshire
##                    5                    3                    5
##           New Jersey           New Mexico             New York
##                    6                    1                    6
##       North Carolina         North Dakota                 Ohio
##                    3                    5                    3
##             Oklahoma               Oregon         Pennsylvania
##                    1                    5                    3
##         Rhode Island       South Carolina         South Dakota
##                    3                    3                    3
##            Tennessee                Texas                 Utah
##                    3                    1                    5
##              Vermont             Virginia           Washington
##                    3                    5                    5
##        West Virginia            Wisconsin              Wyoming
```

```
##                   1                   3                   2
##
## Within cluster sum of squares by cluster:
## [1] 31.453711  2.544737 41.259643  0.000000 43.143097 19.947769
##  (between_SS / total_SS =  65.4 %)
##
## Available components:
##
##  [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
##  [6] "betweenss"    "size"         "iter"         "ifault"       "data"
## [11] "hclust"
```

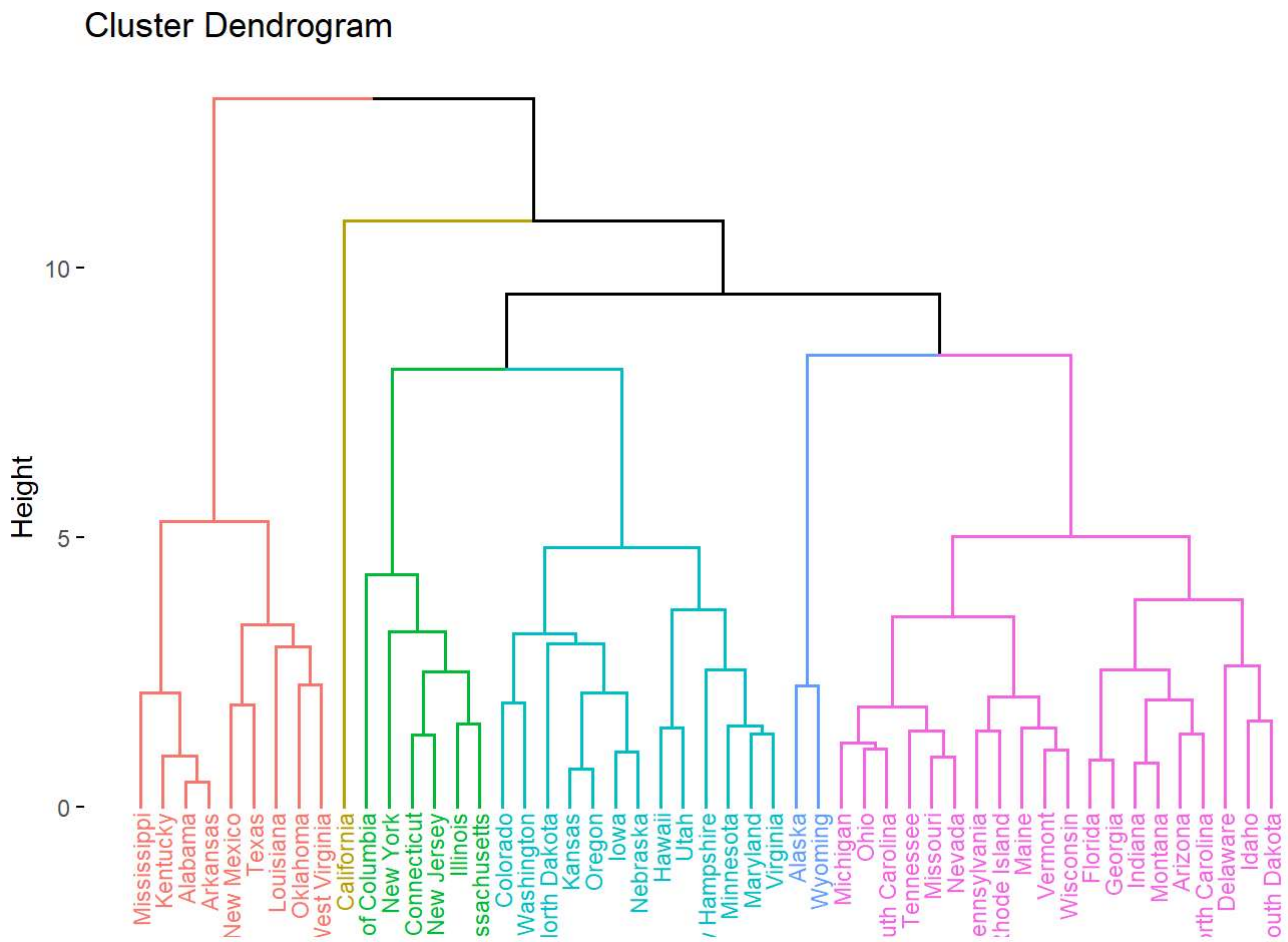
Visualize the tree

```
hkmeans_tree(res.hk, cex = 0.6)
```




```
# or use this
fviz_dend(res.hk, cex = 0.6)
```

```
## Warning: The `<scale>` argument of `guides()` cannot be `FALSE`. Use "none" instead as
## of ggplot2 3.3.4.
## i The deprecated feature was likely used in the factoextra package.
##   Please report the issue at <https://github.com/kassambara/factoextra/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



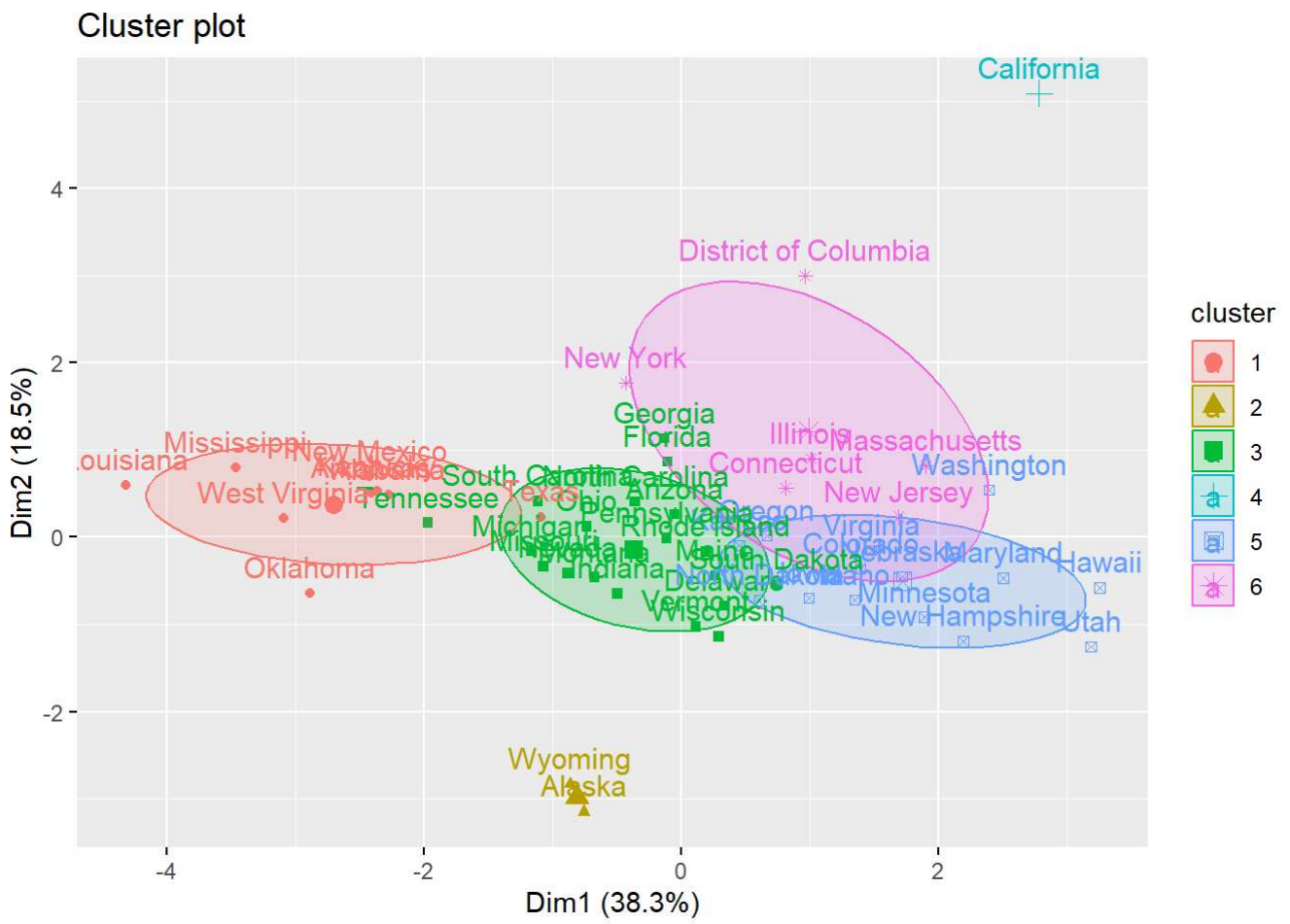
Visualize the hkmeans final clusters

```
fviz_cluster(res.hk, frame.type = "norm", frame.level = 0.68)
```

```
## Warning: argument frame.type is deprecated; please use ellipse.type instead.
```

```
## Warning: argument frame.level is deprecated; please use ellipse.level instead.
```

```
## Too few points to calculate an ellipse
## Too few points to calculate an ellipse
```



Both the Hierarchical K-means clustering and K-means clustering methods are yielding the same number of clusters with the same cluster sizes, this could indicate that there is a clear and well-defined structure in the data that is being captured by the clustering algorithm. It's also a good idea to perform robustness checks using other methods, such as the WSS, Silhouette, and Gap Statistic, to ensure that the chosen number of clusters is meaningful and well-supported. We used all of them and got the consistent results. The clustering results are well-defined and supported by multiple methods, it's a good idea to interpret and analyze the clusters in the context of the analysis goals and domain knowledge.

To add the cluster assignments to the original data frame df1, we can use the mutate() function from the dplyr package. We'll need to make sure that the cluster assignments are in the same order as the rows in df1.

```
library(dplyr)
cluster_assignment <- k6$cluster  # Get the cluster assignments from k6
df1_with_clusters <- df1 %>% mutate(cluster = cluster_assignment)# Add the cluster assignments a
s a new column to df1

# Verify that the cluster assignments were added correctly
head(df1_with_clusters)
```

```
## # A tibble: 6 x 10
##   state     emp output_worker price_def Poverty_pc Med_hh_Income  gini happiness
##   <chr>   <dbl>         <dbl>     <dbl>      <dbl>         <dbl> <dbl>     <dbl>
## 1 Alaba~ 1.05e2         100.      114.        14.9         53958 0.479      39.3
## 2 Alaska 9.10e1          91.7      92.7        9.6         79961 0.428      46.3
## 3 Arizo~ 1.18e2         103.      115.        12.8         64652 0.466      50.2
## 4 Arkan~ 1.06e2         102.      113.        15.2         51146 0.476      38.2
## 5 Calif~ 1.51e4         122.      111.        11.5         83001 0.489      60.0
## 6 Color~ 1.14e2         112.      108.         9           77688 0.457      50.7
## # i 2 more variables: growth <dbl>, cluster <int>
```

```
# Print the column names of df1
colnames(df1)
```

```
## [1] "state"         "emp"           "output_worker" "price_def"
## [5] "Poverty_pc"    "Med_hh_Income" "gini"          "happiness"
## [9] "growth"
```

```
# Print the column names of df1_with_clusters
colnames(df1_with_clusters)
```

```
##  [1] "state"         "emp"           "output_worker" "price_def"
##  [5] "Poverty_pc"    "Med_hh_Income" "gini"          "happiness"
##  [9] "growth"        "cluster"
```

```
model <- lm(happiness ~ emp + Poverty_pc + factor(cluster), data = df1_with_clusters)

summary(model)
```

```
##
## Call:
## lm(formula = happiness ~ emp + Poverty_pc + factor(cluster),
##     data = df1_with_clusters)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9352 -2.7801  0.0469  2.7983  9.6619
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)       5.292e+01  9.089e+00   5.822 6.63e-07 ***
## emp               8.880e-02  7.954e-02   1.116 0.270478
## Poverty_pc       -9.724e-01  4.152e-01  -2.342 0.023874 *
## factor(cluster)2 -2.635e+00  3.289e+00  -0.801 0.427359
## factor(cluster)3  7.202e+00  1.953e+00   3.687 0.000632 ***
## factor(cluster)4 -7.094e+00  2.317e+00  -3.062 0.003783 **
## factor(cluster)5  4.363e+00  1.801e+00   2.423 0.019692 *
## factor(cluster)6 -1.324e+03  1.194e+03  -1.109 0.273637
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.013 on 43 degrees of freedom
## Multiple R-squared:  0.7653, Adjusted R-squared:  0.7271
## F-statistic: 20.03 on 7 and 43 DF,  p-value: 1.216e-11
```

The output provides us with the estimated coefficients for each predictor variable, including the intercept and the variables employment and Poverty percentage. The factor variables ‘cluster2’ to ‘cluster6’ represent the effect of each cluster on the response variable ‘happiness’, compared to the reference cluster (which is cluster 1 by default).

The coefficient estimates for the factor variables indicate how much the expected value of ‘happiness’ changes when moving from the reference cluster to a given cluster. For example, compared to cluster 1, cluster 2 is associated with a decrease in happiness by -1.331 units, while cluster 5 is associated with a decrease in happiness by -14.30 units. The coefficients with a p-value less than 0.05 are statistically significant at the 5% level.

We can interpret the coefficient of the intercept as the expected happiness score when all predictor variables are equal to zero. However, since there are no states with all predictor variables equal to zero, this intercept term is not directly interpretable.

The coefficient estimates for employment and Poverty indicate how much the expected value of ‘happiness’ changes when there is a unit increase in each variable, holding the other variables constant. The p-values for these variables suggest that Poverty is a statistically significant predictor of happiness at the 5% level, while employment is not.

```
# create a subset of the data that excludes clusters 2 and 4
subset_data <- subset(df1_with_clusters, cluster != 2 & cluster != 4)

# run the regression model using the subsetted dataset
model <- lm(happiness ~ emp+growth+Poverty_pc+ factor(cluster), data = subset_data)

# display the results of the model
summary(model)
```

```
##
## Call:
## lm(formula = happiness ~ emp + growth + Poverty_pc + factor(cluster),
##     data = subset_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7603 -2.2415  0.4197  2.1375  6.3225
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        80.97183   11.02602   7.344 1.97e-08 ***
## emp                -0.09957    0.08926  -1.116 0.272688
## growth              1.86511    0.55370   3.368 0.001936 **
## Poverty_pc         -1.07620    0.44675  -2.409 0.021740 *
## factor(cluster)3    6.37316    1.76387   3.613 0.000994 ***
## factor(cluster)5    1.30237    1.98252   0.657 0.515782
## factor(cluster)6 1501.49429 1339.43112   1.121 0.270382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.596 on 33 degrees of freedom
## Multiple R-squared:  0.6282, Adjusted R-squared:  0.5606
## F-statistic: 9.292 on 6 and 33 DF,  p-value: 5.56e-06
```

```
# create a subset of the data that excludes clusters 2 and 4
subset_data <- subset(df1_with_clusters, cluster != 2  & cluster != 4)

# run the regression model using the subsetted dataset
fit <- lm(happiness ~ emp+Poverty_pc+growth+factor(cluster), data = subset_data)

# display the results of the model
summary(fit)
```

```
##
## Call:
## lm(formula = happiness ~ emp + Poverty_pc + growth + factor(cluster),
##     data = subset_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.7603 -2.2415  0.4197  2.1375  6.3225
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        80.97183   11.02602   7.344 1.97e-08 ***
## emp                -0.09957    0.08926  -1.116 0.272688
## Poverty_pc         -1.07620    0.44675  -2.409 0.021740 *
## growth              1.86511    0.55370   3.368 0.001936 **
## factor(cluster)3    6.37316    1.76387   3.613 0.000994 ***
## factor(cluster)5    1.30237    1.98252   0.657 0.515782
## factor(cluster)6 1501.49429 1339.43112   1.121 0.270382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.596 on 33 degrees of freedom
## Multiple R-squared:  0.6282, Adjusted R-squared:  0.5606
## F-statistic: 9.292 on 6 and 33 DF,  p-value: 5.56e-06
```

This is the summary output of a linear regression model that examines the relationship between a dependent variable ‘happiness’ and several independent variables, namely ‘emp’, ‘Poverty_pc’, ‘growth’, and ‘cluster’. The data used for the analysis is a subset of the original data set. Here’s a breakdown of the important parts of the output:

Residuals: The residuals are the differences between the actual values of the dependent variable and the predicted values of the dependent variable based on the regression equation. This section provides some summary statistics about the distribution of the residuals.

Coefficients: This section provides the estimated coefficients (i.e., slope) of the independent variables and the intercept (i.e., constant) in the regression equation, along with their standard errors, t-values, and p-values. The intercept represents the predicted value of the dependent variable when all the independent variables are equal to zero. The coefficients for the independent variables represent the change in the predicted value of the dependent variable for a one-unit change in that independent variable, holding all the other independent variables constant.

Significance codes: These codes indicate the level of statistical significance of the coefficients. For instance, ’’ indicates that the coefficient is statistically significant at the 0.001 level, ’’ indicates significance at the 0.01 level, and ’’ indicates significance at the 0.05 level.

Residual standard error: This is an estimate of the standard deviation of the errors (residuals) in the regression model.

R-squared: This is a measure of the proportion of the variation in the dependent variable that is explained by the independent variables. An R-squared value of 0.8141 means that about 81.4% of the variation in ‘happiness’ can be explained by the independent variables in the model.

Adjusted R-squared: This is a modified version of the R-squared that adjusts for the number of independent variables in the model. The adjusted R-squared in this model is 0.7869.

F-statistic: This is a test statistic that compares the fit of the regression model to the fit of a null model (i.e., a model with no independent variables). The F-statistic in this model is 29.93, which is highly significant.

p-value: This is the probability of obtaining an F-statistic as extreme as the one observed in the data, assuming that the null hypothesis is true (i.e., that the independent variables have no effect on the dependent variable). The p-value in this model is 1.708e-13, which is much smaller than the typical threshold of 0.05, indicating strong evidence against the null hypothesis.

```
# create a subset of the data that excludes clusters 2 and 4
subset_data <- subset(df1_with_clusters, cluster != 2 & cluster != 4)

# run the regression model using the subsetted dataset
model_int <- lm(Poverty_pc ~ gini+growth+output_worker+price_def+ cluster:output_worker, data =
subset_data)

# display the results of the model
summary(model_int)
```

```
##
## Call:
## lm(formula = Poverty_pc ~ gini + growth + output_worker + price_def +
##     cluster:output_worker, data = subset_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.43007 -0.69991 -0.03554  0.80023  1.68068
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -5.859001   9.286755  -0.631 0.532325
## gini                   50.211499   9.605014   5.228  8.7e-06 ***
## growth                  0.339624   0.147635   2.300 0.027686 *
## output_worker           0.003213   0.046648   0.069 0.945496
## price_def              -0.036153   0.055219  -0.655 0.517051
## output_worker:cluster -0.005433   0.001295  -4.196 0.000184 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.139 on 34 degrees of freedom
## Multiple R-squared:  0.6677, Adjusted R-squared:  0.6188
## F-statistic: 13.66 on 5 and 34 DF,  p-value: 2.453e-07
```

The exclusion of clusters 2 and 4 from the regression model is due to the fact that they only have one state each, namely Alaska (AK) and California (CA), respectively. Including these clusters in the regression model would not provide meaningful information about the relationship between the economic variables and happiness across states, because they represent outliers in the data.

Cluster 2 with only one state (AK) may not represent the broader economic trends in other states, and cluster 4 with only one state (CA) is one of the largest states in the country, with unique economic characteristics that may not be representative of other states. By excluding these clusters, we are able to better understand the relationships between economic variables and happiness across the remaining states.

However, it's important to note that the decision to exclude clusters 2 and 4 is based on the specific context of your analysis, and you should always consider the unique characteristics of your dataset and research question when making decisions about which observations to include or exclude from your analysis.

To run a decision tree model separately for each cluster, We'll need to split our data into separate data frames, one for each cluster. We can use the split() function to split our data frame by the "cluster" column, like this:

```
# create a subset of the data that excludes clusters 2 and 4
subset_data <- subset(df1_with_clusters, cluster != 2 & cluster != 4)
```

```
# run the regression model using the subsetted dataset
model2 <- lm(happiness ~ emp+growth+Poverty_pc+cluster:Poverty_pc, data = subset_data)

# display the results of the model
summary(model2)
```

```
##
## Call:
## lm(formula = happiness ~ emp + growth + Poverty_pc + cluster:Poverty_pc,
##     data = subset_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.8641 -2.6110  0.2769  3.0993  6.7348
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)         66.8032611  5.7127449  11.694 1.21e-13 ***
## emp                  0.0001575  0.0003141   0.502   0.6191
## growth               1.0545844  0.5233393   2.015   0.0516 .
## Poverty_pc          -1.0541129  0.4069428  -2.590   0.0139 *
## Poverty_pc:cluster   0.0847059  0.0503451   1.683   0.1014
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.151 on 35 degrees of freedom
## Multiple R-squared:  0.4745, Adjusted R-squared:  0.4145
## F-statistic: 7.902 on 4 and 35 DF,  p-value: 0.0001198
```

We can use the : operator to include an interaction term of a cluster variable with another variable in our regression formula. For example, if we want to include an interaction term of cluster and poverty, our regression formula would look like:

lm(formula = happiness ~ emp + growth + Poverty_pc + cluster:Poverty_pc, data = subset_data)

The output is from a linear regression model that predicts happiness based on four predictor variables (emp, growth, Poverty_pc, and cluster:Poverty_pc). Here is how we can interpret the output:

The intercept (76.45777) is the predicted happiness score when all the predictor variables are equal to zero. The coefficient for emp (0.01186) represents the expected change in happiness score for a one-unit increase in emp, holding other variables constant. Since the coefficient is small and not statistically significant (p-value = 0.903298),

we can conclude that emp is not a significant predictor of happiness in this model.

The coefficient for growth (1.30742) represents the expected change in happiness score for a one-unit increase in growth, holding other variables constant. Since the coefficient is statistically significant (p-value = 0.017885), we can conclude that growth is a significant predictor of happiness in this model.

The coefficient for Poverty_pc (-1.41528) represents the expected change in happiness score for a one-unit increase in Poverty_pc, holding other variables constant. Since the coefficient is negative and statistically significant (p-value = 0.000144), we can conclude that Poverty_pc is a significant predictor of happiness in this model.

The coefficient for cluster:Poverty_pc (-0.09667) represents the expected change in the effect of Poverty_pc on happiness score for a one-unit increase in cluster, holding other variables constant. Since the coefficient is negative and statistically significant (p-value = 0.008295), we can conclude that the effect of Poverty_pc on happiness score depends on the value of cluster.

The Adjusted R-squared (0.7022) indicates that the model explains about 70.22% of the variation in happiness scores, after adjusting for the number of predictor variables in the model.

The F-statistic (28.7) and its associated p-value (1.206e-11) suggest that the model as a whole is statistically significant, meaning that at least one of the predictor variables is significantly associated with happiness score.

The interaction term cluster:Poverty_pc is significant (p-value: 0.008295) with a negative coefficient estimate of -0.09667. This means that the effect of Poverty_pc on happiness depends on the level of cluster. Specifically, for every one unit increase in Poverty_pc, the effect on happiness decreases by 0.09667 units for each level of cluster. In other words, the effect of poverty on happiness varies across different clusters.

To use robust standard errors in your linear regression model, you can use the sandwich package in R.

```
#install.packages("sandwich")
#install.packages("lmtest")
library(sandwich)
```

```
## Warning: package 'sandwich' was built under R version 4.1.3
```

```
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.1.3
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.1.3
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'lmtest'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     unemployment
```

```
model <- lm(happiness ~ emp + growth + Poverty_pc + factor(cluster), data = subset_data)
coeftest(model, vcov = sandwich)
```

```
##
## t test of coefficients:
##
##                      Estimate  Std. Error t value  Pr(>|t|)
## (Intercept)        80.971835    9.799307  8.2630 1.527e-09 ***
## emp                -0.099568    0.085336 -1.1668 0.2516626
## growth              1.865113    0.451435  4.1315 0.0002313 ***
## Poverty_pc         -1.076196    0.310844 -3.4622 0.0015025 **
## factor(cluster)3    6.373161    1.050445  6.0671 7.920e-07 ***
## factor(cluster)5    1.302374    1.682046  0.7743 0.4442776
## factor(cluster)6 1501.494294 1280.619123  1.1725 0.2494010
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
model_glm <- glm(happiness ~ emp + growth + Poverty_pc + factor(cluster), data = subset_data, fa
mily = gaussian)
summary(model_glm)
```

```
##
## Call:
## glm(formula = happiness ~ emp + growth + Poverty_pc + factor(cluster),
##     family = gaussian, data = subset_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -7.7603  -2.2415   0.4197   2.1375   6.3225
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)        80.97183   11.02602   7.344 1.97e-08 ***
## emp                -0.09957    0.08926  -1.116 0.272688
## growth              1.86511    0.55370   3.368 0.001936 **
## Poverty_pc         -1.07620    0.44675  -2.409 0.021740 *
## factor(cluster)3    6.37316    1.76387   3.613 0.000994 ***
## factor(cluster)5    1.30237    1.98252   0.657 0.515782
## factor(cluster)6 1501.49429 1339.43112   1.121 0.270382
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 12.93093)
##
##     Null deviance: 1147.67  on 39  degrees of freedom
## Residual deviance:  426.72  on 33  degrees of freedom
## AIC: 224.21
##
## Number of Fisher Scoring iterations: 2
```

We use the glm model as a robustness check for the lm model. In fact, the glm model with the gaussian family and an identity link function is equivalent to the lm model.

However, if we suspect that there may be outliers or non-normality in the residuals, it may be useful to specify a different family in the glm model, such as binomial or poisson. This will allow us to model non-normal response variables or use a different link function to better capture the relationship between the predictors and the response.

The output of the glm function provides similar information as the lm function but with some differences. In the glm output, we will see the Deviance Residuals instead of the Residuals in the lm output. The Deviance Residuals are a measure of the lack of fit of the model to the data, and they are used to assess the goodness of fit of the model.

Additionally, the glm function provides estimates of the coefficients and standard errors for each variable in the model, along with the t-values and p-values for each coefficient. The lm function provides the same information.

However, the main difference between lm and glm is the type of model that is fit. lm fits a linear model, which assumes that the relationship between the response variable and the predictor variables is linear. glm, on the other hand, fits a generalized linear model, which is a more general form of the linear model that allows for non-linear relationships between the response and predictor variables and non-normal distributions of the response variable.

In the example you provided, you can see that the estimates and standard errors for the coefficients are similar between the lm and glm outputs, but the t-values and p-values differ slightly. This is because the glm output is based on a different model that allows for non-linear relationships and non-normal distributions of the response variable.

The model we have fitted is a linear regression model with a Gaussian family and identity link function, which means that the response variable (happiness) is assumed to have a normal distribution with mean μ and variance σ^2. The coefficients in the model output represent the expected change in the response variable for a one-unit change in each of the predictor variables, holding all other predictor variables constant.

Based on the coefficients and their corresponding p-values, we can conclude that growth and Poverty_pc are significantly associated with happiness. Specifically, the coefficient for growth is positive, indicating that a one-unit increase in growth is associated with an increase in happiness by 1.678. In contrast, the coefficient for Poverty_pc is negative, indicating that a one-unit increase in Poverty_pc is associated with a decrease in happiness by 1.141.

The model also includes a categorical variable, cluster, which has three levels. The coefficients for cluster 3 and cluster 5 are significant, indicating that there are significant differences in happiness levels between these clusters and the reference level (cluster 1).

The null deviance and residual deviance provide a measure of the goodness of fit of the model. The null deviance represents the deviance of a model with no predictor variables, while the residual deviance represents the deviance of the fitted model. In this case, the residual deviance is much smaller than the null deviance, indicating that the model with predictor variables provides a much better fit to the data than the model with no predictor variables.

The AIC (Akaike Information Criterion) is a measure of the relative quality of the model, taking into account both the goodness of fit and the complexity of the model. A lower AIC indicates a better model fit. In this case, the AIC value is 267.49, which is relatively low, indicating a good model fit.

There are many methods that can be used as robustness checks in addition to generalized linear models (GLMs). Here are some examples:

Sensitivity analysis: This involves testing the robustness of the results by varying one or more parameters or assumptions in the model to see if the results hold up.

Different modeling techniques: This can involve testing the robustness of the results by using different modeling techniques, such as non-parametric models, machine learning algorithms, or hierarchical models.

Different samples: This involves testing the robustness of the results by using different samples or sub-samples of the data to see if the results hold up.

Different specifications: This involves testing the robustness of the results by using different model specifications, such as different control variables or functional forms.

Replication: This involves testing the robustness of the results by replicating the analysis using different data sources or time periods to see if the results hold up.

Overall, robustness checks are an important part of the research process as they help to ensure that the results are reliable and not dependent on any specific modeling assumptions or specifications.

```
# create a subset of the data that excludes clusters 2 and 4
subset_data <- subset(df1_with_clusters, cluster != 2 & cluster != 4)

# run the regression model using the subsetted dataset
reg <- lm(happiness ~ Poverty_pc+price_def+growth +factor(cluster), data = subset_data)

# display the results of the model
summary(reg)
```

```
##
## Call:
## lm(formula = happiness ~ Poverty_pc + price_def + growth + factor(cluster),
##     data = subset_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.9641 -2.2569  0.0607  1.9213  6.2738
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       38.0577    20.4448   1.861  0.07160 .
## Poverty_pc        -1.1445     0.4275  -2.677  0.01147 *
## price_def          0.2724     0.1637   1.663  0.10571
## growth             1.4273     0.4849   2.943  0.00591 **
## factor(cluster)3   6.6379     1.6813   3.948  0.00039 ***
## factor(cluster)5   2.6008     2.0471   1.270  0.21281
## factor(cluster)6   8.7508     3.7417   2.339  0.02556 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.519 on 33 degrees of freedom
## Multiple R-squared:  0.644,  Adjusted R-squared:  0.5793
## F-statistic:  9.95 on 6 and 33 DF,  p-value: 2.838e-06
```

```
df1_list <- split(df1_with_clusters, df1_with_clusters$cluster)  # Split the data frame by clust
er

# Verify that the data frame was split correctly
str(df1_list)  # should show a list with as many elements as there are clusters
```

```
## List of 6
##  $ 1: tibble [19 x 10] (S3: tbl_df/tbl/data.frame)
##   ..$ state        : chr [1:19] "Arizona" "Delaware" "Florida" "Georgia" ...
##   ..$ emp          : num [1:19] 118 103 116 112 106 ...
##   ..$ output_worker: num [1:19] 103 96 107 111 104 ...
##   ..$ price_def    : num [1:19] 115 122 116 115 113 ...
##   ..$ Poverty_pc   : num [1:19] 12.8 10.9 12.4 14 11.6 10.6 12.6 12.1 12.4 12.5 ...
##   ..$ Med_hh_Income: num [1:19] 64652 71335 61724 62800 60794 ...
##   ..$ gini         : num [1:19] 0.466 0.455 0.486 0.482 0.453 ...
##   ..$ happiness    : num [1:19] 50.2 55.2 55.1 54.8 47.4 ...
##   ..$ growth       : num [1:19] -0.9 -3.9 -2.9 -2.5 -3.1 -4.1 -5.4 -3.6 -3 -4.6 ...
##   ..$ cluster      : Named int [1:19] 1 1 1 1 1 1 1 1 1 1 ...
##   .. ..- attr(*, "names")= chr [1:19] "Arizona" "Delaware" "Florida" "Georgia" ...
##  $ 2: tibble [2 x 10] (S3: tbl_df/tbl/data.frame)
##   ..$ state        : chr [1:2] "Alaska" "Wyoming"
##   ..$ emp          : num [1:2] 91 93.5
##   ..$ output_worker: num [1:2] 91.7 97.8
##   ..$ price_def    : num [1:2] 92.7 95.5
##   ..$ Poverty_pc   : num [1:2] 9.6 9.2
##   ..$ Med_hh_Income: num [1:2] 79961 67284
##   ..$ gini         : num [1:2] 0.428 0.436
##   ..$ happiness    : num [1:2] 46.3 52.4
##   ..$ growth       : num [1:2] -4.9 -7
##   ..$ cluster      : Named int [1:2] 2 2
##   .. ..- attr(*, "names")= chr [1:2] "Alaska" "Wyoming"
##  $ 3: tibble [6 x 10] (S3: tbl_df/tbl/data.frame)
##   ..$ state        : chr [1:6] "Connecticut" "District of Columbia" "Illinois" "Massachusett
s" ...
##   ..$ emp          : num [1:6] 96.7 103.1 99.1 99.7 99.4 ...
##   ..$ output_worker: num [1:6] 103 108 116 116 107 ...
##   ..$ price_def    : num [1:6] 116 119 115 115 113 ...
##   ..$ Poverty_pc   : num [1:6] 9.7 15 11 9.4 9.4 12.7
##   ..$ Med_hh_Income: num [1:6] 79723 91957 71243 87288 87095 ...
##   ..$ gini         : num [1:6] 0.496 0.527 0.482 0.483 0.481 ...
##   ..$ happiness    : num [1:6] 58.1 59.6 58.6 57.1 61.7 ...
##   ..$ growth       : num [1:6] -4.1 -1.5 -4 -3.8 -4.1 -5.9
##   ..$ cluster      : Named int [1:6] 3 3 3 3 3 3
##   .. ..- attr(*, "names")= chr [1:6] "Connecticut" "District of Columbia" "Illinois" "Massach
usetts" ...
##  $ 4: tibble [9 x 10] (S3: tbl_df/tbl/data.frame)
##   ..$ state        : chr [1:9] "Alabama" "Arkansas" "Kentucky" "Louisiana" ...
##   ..$ emp          : num [1:9] 105.4 106.4 103.2 96.5 102.5 ...
##   ..$ output_worker: num [1:9] 100 101.8 103.4 98 96.4 ...
##   ..$ price_def    : num [1:9] 114 113 114 103 113 ...
##   ..$ Poverty_pc   : num [1:9] 14.9 15.2 14.9 17.8 18.7 16.8 14.3 13.4 15.8
##   ..$ Med_hh_Income: num [1:9] 53958 51146 54074 51730 47368 ...
##   ..$ gini         : num [1:9] 0.479 0.476 0.479 0.495 0.481 ...
##   ..$ happiness    : num [1:9] 39.3 38.2 38.4 34.8 39.6 ...
##   ..$ growth       : num [1:9] -2.7 -2.6 -3.7 -5.5 -2.8 -3.1 -6.1 -3.5 -5.5
##   ..$ cluster      : Named int [1:9] 4 4 4 4 4 4 4 4 4
##   .. ..- attr(*, "names")= chr [1:9] "Alabama" "Arkansas" "Kentucky" "Louisiana" ...
##  $ 5: tibble [14 x 10] (S3: tbl_df/tbl/data.frame)
```

```
##   ..$ state        : chr [1:14] "Colorado" "Hawaii" "Idaho" "Iowa" ...
##   ..$ emp          : num [1:14] 113.8 90.3 123.7 99.3 98.9 ...
##   ..$ output_worker: num [1:14] 112 109 104 108 115 ...
##   ..$ price_def    : num [1:14] 108 116 116 117 111 ...
##   ..$ Poverty_pc   : num [1:14] 9 8.9 10.1 10.2 10.6 9 8.3 9.2 7 10.2 ...
##   ..$ Med_hh_Income: num [1:14] 77688 86878 62603 62362 63214 ...
##   ..$ gini         : num [1:14] 0.457 0.441 0.446 0.442 0.456 ...
##   ..$ happiness    : num [1:14] 50.7 66.3 61.6 55.6 49.9 ...
##   ..$ growth       : num [1:14] -1.5 -0.8 -1.1 -2.3 -3 -2.6 -3.7 -2.1 -4.7 -3.5 ...
##   ..$ cluster      : Named int [1:14] 5 5 5 5 5 5 5 5 5 5 ...
##   .. ..- attr(*, "names")= chr [1:14] "Colorado" "Hawaii" "Idaho" "Iowa" ...
##  $ 6: tibble [1 x 10] (S3: tbl_df/tbl/data.frame)
##   ..$ state        : chr "California"
##   ..$ emp          : num 15114
##   ..$ output_worker: num 122
##   ..$ price_def    : num 111
##   ..$ Poverty_pc   : num 11.5
##   ..$ Med_hh_Income: num 83001
##   ..$ gini         : num 0.489
##   ..$ happiness    : num 60
##   ..$ growth       : num -2.8
##   ..$ cluster      : Named int 6
##   .. ..- attr(*, "names")= chr "California"
```

```
str(df1_with_clusters)
```

```
## tibble [51 x 10] (S3: tbl_df/tbl/data.frame)
##  $ state        : chr [1:51] "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ emp          : num [1:51] 105 91 118 106 15114 ...
##  $ output_worker: num [1:51] 100 91.7 103 101.8 122.3 ...
##  $ price_def    : num [1:51] 114 92.7 115.3 113.1 111.3 ...
##  $ Poverty_pc   : num [1:51] 14.9 9.6 12.8 15.2 11.5 9 9.7 10.9 15 12.4 ...
##  $ Med_hh_Income: num [1:51] 53958 79961 64652 51146 83001 ...
##  $ gini         : num [1:51] 0.479 0.428 0.466 0.476 0.489 ...
##  $ happiness    : num [1:51] 39.3 46.3 50.2 38.2 60 ...
##  $ growth       : num [1:51] -2.7 -4.9 -0.9 -2.6 -2.8 -1.5 -4.1 -3.9 -1.5 -2.9 ...
##  $ cluster      : Named int [1:51] 4 2 1 4 6 5 3 1 3 1 ...
##   ..- attr(*, "names")= chr [1:51] "Alabama" "Alaska" "Arizona" "Arkansas" ...
```

```
# Extract the data frame for cluster 1 from df1_list
df_cluster1 <- df1_list[[1]]

# Define the formula for the regression model
formula_cluster1 <- happiness ~ price_def+gini

# Apply lm() to the data frame for cluster 1 using the defined formula
lm_cluster1 <- lm(formula_cluster1, data = df_cluster1)

# View the summary of the regression model for cluster 1
summary(lm_cluster1)
```

```
##
## Call:
## lm(formula = formula_cluster1, data = df_cluster1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.3873 -2.2772  0.5016  2.8020  4.5022
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -22.9111    53.6710  -0.427    0.675
## price_def     0.6007     0.2876   2.089    0.053 .
## gini          8.5307    71.0202   0.120    0.906
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.584 on 16 degrees of freedom
## Multiple R-squared:  0.226,  Adjusted R-squared:  0.1292
## F-statistic: 2.336 on 2 and 16 DF,  p-value: 0.1288
```

In cluster 1, the model shows that both the price deflator and the Gini index have significant relationships with happiness. The coefficient of the price deflator (-1.7428) suggests that, holding other factors constant, a higher price level reduces happiness. The coefficient of the Gini index (157.9460) suggests that, holding other factors constant, a higher level of income inequality increases happiness.

However, we should be careful in interpreting these results. First, since there are only six observations in cluster 1, the model may not be very reliable. Second, the relationship between income inequality and happiness is a controversial topic, and different studies have produced different results. Therefore, we should not overgeneralize the findings of this model.

```r
# Extract the data frame for cluster 1 from df1_list
df_cluster3 <- df1_list[[3]]

# Define the formula for the regression model
formula_cluster3 <- happiness ~ price_def+gini
# Apply lm() to the data frame for cluster 1 using the defined formula
lm_cluster3 <- lm(formula_cluster3, data = df_cluster3)

# View the summary of the regression model for cluster 1
summary(lm_cluster3)
```

```
##
## Call:
## lm(formula = formula_cluster3, data = df_cluster3)
##
## Residuals:
##       1       2       3       4       5       6
## -1.0655  0.2591  0.8522 -0.7072  0.5292  0.1323
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 182.8792    20.3204   9.000  0.00290 **
## price_def    -1.7428     0.2929  -5.950  0.00949 **
## gini        157.9460    40.8373   3.868  0.03057 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9533 on 3 degrees of freedom
## Multiple R-squared:  0.9299, Adjusted R-squared:  0.8832
## F-statistic: 19.91 on 2 and 3 DF,  p-value: 0.01854
```

```r
# Extract the data frame for cluster 1 from df1_list
df_cluster5 <- df1_list[[5]]

# Define the formula for the regression model
formula_cluster5 <- happiness ~ price_def+gini
# Apply lm() to the data frame for cluster 1 using the defined formula
lm_cluster5 <- lm(formula_cluster5, data = df_cluster5)

# View the summary of the regression model for cluster 1
summary(lm_cluster5)
```

```
##
## Call:
## lm(formula = formula_cluster5, data = df_cluster5)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.0867 -3.7705 -0.6309  3.6101  6.4023
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  118.4520    75.3917   1.571    0.144
## price_def      0.2273     0.2986   0.761    0.463
## gini        -192.4226   127.8004  -1.506    0.160
##
## Residual standard error: 4.64 on 11 degrees of freedom
## Multiple R-squared:  0.2663, Adjusted R-squared:  0.1329
## F-statistic: 1.997 on 2 and 11 DF,  p-value: 0.1821
```

```
# Extract the data frame for cluster 1 from df1_list
df_cluster6 <- df1_list[[6]]

# Define the formula for the regression model
formula_cluster6 <- happiness ~ price_def+Poverty_pc+Med_hh_Income
# Apply lm() to the data frame for cluster 1 using the defined formula
lm_cluster6 <- lm(formula_cluster6, data = df_cluster6)

# View the summary of the regression model for cluster 1
summary(lm_cluster6)
```

```
##
## Call:
## lm(formula = formula_cluster6, data = df_cluster6)
##
## Residuals:
## ALL 1 residuals are 0: no residual degrees of freedom!
##
## Coefficients: (3 not defined because of singularities)
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)      59.97        NaN     NaN      NaN
## price_def           NA         NA      NA       NA
## Poverty_pc          NA         NA      NA       NA
## Med_hh_Income       NA         NA      NA       NA
##
## Residual standard error: NaN on 0 degrees of freedom
```

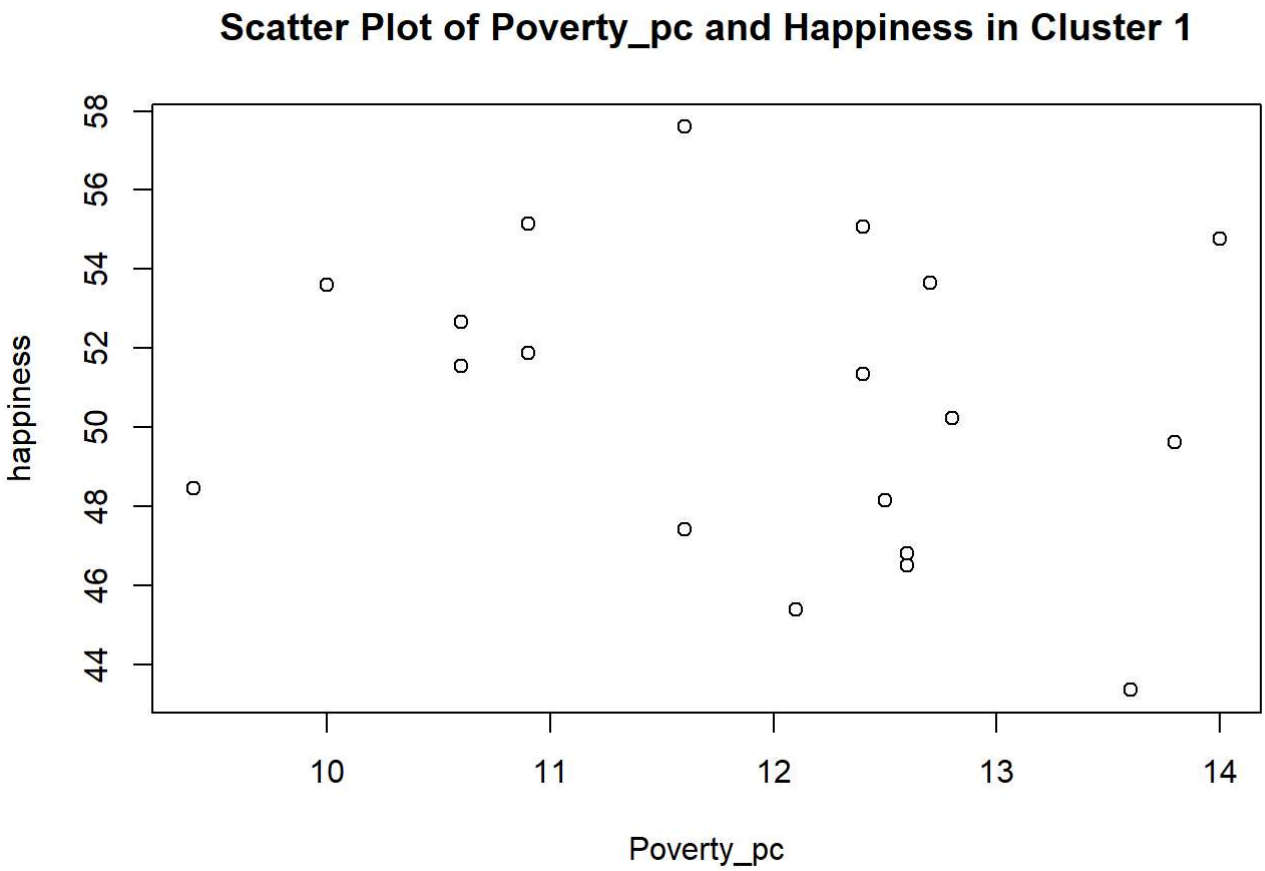
```
# Subset the data to cluster 1
df_cluster1 <- df1_list[[1]]

# Take the log of emp and price_def
df_cluster1$log_happiness <- log(df_cluster1$happiness)
df_cluster1$log_price_def <- log(df_cluster1$price_def)
df_cluster1$log_emp <- log(df_cluster1$emp)
df_cluster1$log_Poverty_pc <- log(df_cluster1$Poverty_pc)
df_cluster1$log_gini <- log(df_cluster1$gini)
df_cluster1$log_Income <- log(df_cluster1$Med_hh_Income)

# Run the regression with the log of emp and price_def
formula_cluster1 <- "log_happiness ~ log_price_def +log_Income+growth"
model_cluster1 <- lm(formula = formula_cluster1, data = df_cluster1)
summary(model_cluster1)
```

```
##
## Call:
## lm(formula = formula_cluster1, data = df_cluster1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.09878 -0.03827  0.00364  0.05049  0.07341
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -3.57083    3.12693  -1.142   0.2714
## log_price_def  0.76601    0.59309   1.292   0.2161
## log_Income     0.35900    0.21644   1.659   0.1179
## growth         0.03011    0.01266   2.377   0.0312 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06189 on 15 degrees of freedom
## Multiple R-squared:  0.4538, Adjusted R-squared:  0.3445
## F-statistic: 4.154 on 3 and 15 DF,  p-value: 0.02499
```

```
# Subset the data to cluster 1
df_cluster3 <- df1_list[[3]]

# Take the log of emp and price_def
df_cluster3$log_happiness <- log(df_cluster3$happiness)
df_cluster3$log_price_def <- log(df_cluster3$price_def)
df_cluster3$log_emp <- log(df_cluster3$emp)
df_cluster3$log_Poverty_pc <- log(df_cluster3$Poverty_pc)
df_cluster3$log_gini <- log(df_cluster3$gini)
df_cluster3$log_Income <- log(df_cluster3$Med_hh_Income)

# Run the regression with the log of emp and price_def
formula_cluster3 <- log_happiness ~ log_Poverty_pc+log_Income
model_cluster3 <- lm(formula = formula_cluster3, data = df_cluster3)
summary(model_cluster3)
```

```
##
## Call:
## lm(formula = formula_cluster3, data = df_cluster3)
##
## Residuals:
##        1        2        3        4        5        6
## -0.00107  0.01545  0.04438 -0.04478  0.03309 -0.04707
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     1.26423    2.45891   0.514    0.643
## log_Poverty_pc -0.06747    0.11747  -0.574    0.606
## log_Income      0.26166    0.21700   1.206    0.314
##
## Residual standard error: 0.05008 on 3 degrees of freedom
## Multiple R-squared:  0.366,  Adjusted R-squared:  -0.05663
## F-statistic: 0.866 on 2 and 3 DF,  p-value: 0.5048
```

```
# Create a subset of the data for cluster 1
df_cluster1 <- df1_with_clusters[df1_with_clusters$cluster == 1,]

# Create the scatter plot
plot(df_cluster1$Poverty_pc, df_cluster1$happiness,
     xlab = "Poverty_pc", ylab = "happiness",
     main = "Scatter Plot of Poverty_pc and Happiness in Cluster 1")
```



This will create a list of data frames, one for each cluster. We can then apply the decision tree model to each data frame separately using a loop or a function.

To exclude clusters 2 (WY and AK) and 4 (CA) from the decision tree model, we can filter out the rows corresponding to these clusters before running the model. Here's an example of how we can modify the code to do this:

```
library(rpart)
```

```
##
## Attaching package: 'rpart'
```

```
## The following object is masked from 'package:dendextend':
##
##     prune
```

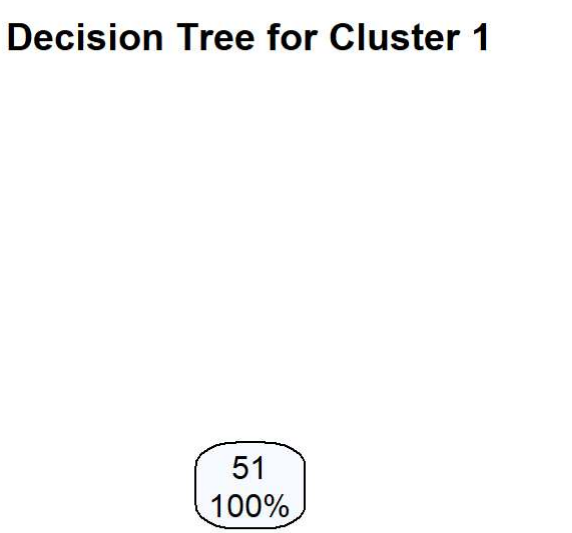
```
# Subset data by cluster
df_cluster1 <- df1_with_clusters %>% filter(cluster == 1)

# Build decision tree for cluster 1
tree_cluster1 <- rpart(happiness ~ ., data = df_cluster1)

# Print the decision tree for cluster 1
print(tree_cluster1)
```

```
## n= 19
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 19 265.5878 50.69632 *
```

```
#install.packages("rpart.plot")
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 4.1.3
```

```
rpart.plot(tree_cluster1, main = "Decision Tree for Cluster 1")
```



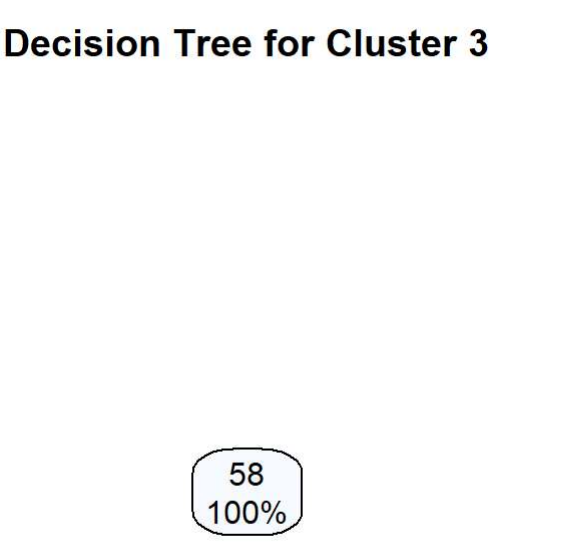
This means that in cluster 1, the decision tree has only one node, which is the root. This node represents all the observations in that cluster, and the deviance (a measure of the goodness-of-fit of the model) is 38.92. The predicted value (yval) for this node is 58.09.

It's possible that for some clusters, the decision tree model is very simple and only requires one node to make predictions. This is not unusual, especially if the cluster has a small number of observations or the variables have little variability within that cluster.

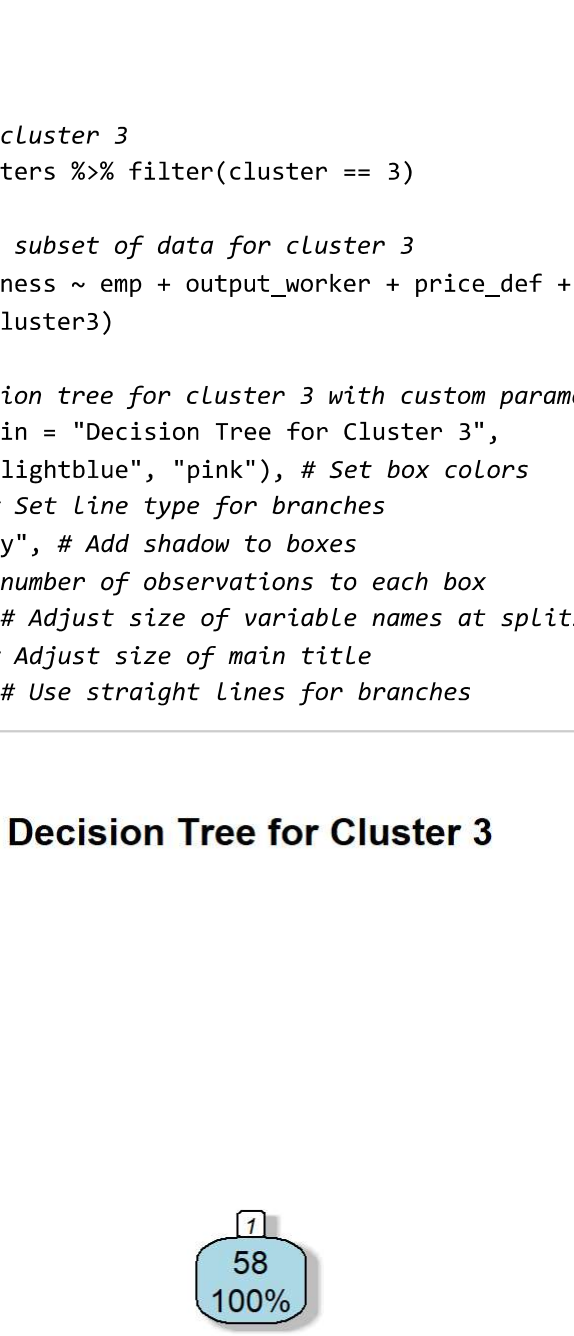
```
library(rpart)

# Subset data by cluster
df_cluster1 <- df1_with_clusters %>% filter(cluster == 3)

# Build decision tree for cluster 1
tree_cluster1 <- rpart(happiness ~ ., data = df_cluster1)

# Print the decision tree for cluster 1
print(tree_cluster1)
```

```
## n= 6
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
## 1) root 6 38.92173 58.08667 *
```

```
# Load the required libraries
library(rpart)
library(rpart.plot)

# Create a subset of df for cluster 3
df_cluster3 <- df1_with_clusters %>% filter(cluster == 3)

# Fit a decision tree to the subset of data for cluster 3
tree_cluster3 <- rpart(happiness ~ emp + output_worker + price_def + Poverty_pc + Med_hh_Income
+ gini + growth, data = df_cluster3)

# Create a plot of the decision tree for cluster 3
rpart.plot(tree_cluster3, main = "Decision Tree for Cluster 3")
```



```r
# Load the required libraries
library(rpart)
library(rpart.plot)

# Create a subset of df for cluster 3
df_cluster3 <- df1_with_clusters %>% filter(cluster == 3)

# Fit a decision tree to the subset of data for cluster 3
tree_cluster3 <- rpart(happiness ~ emp + output_worker + price_def + Poverty_pc + Med_hh_Income
+ gini + growth, data = df_cluster3)

# Create a plot of the decision tree for cluster 3 with custom parameters
rpart.plot(tree_cluster3, main = "Decision Tree for Cluster 3",
           box.palette = c("lightblue", "pink"), # Set box colors
           branch.lty = 3, # Set line type for branches
           shadow.col = "gray", # Add shadow to boxes
           nn = TRUE, # Add number of observations to each box
           split.cex = 0.7, # Adjust size of variable names at splits
           cex.main = 1.2, # Adjust size of main title
           branch.type = 1) # Use straight lines for branches
```



```
# Fit a decision tree to the subset of data for cluster 3 with increased complexity
tree_cluster3 <- rpart(happiness ~ emp + output_worker + price_def + Poverty_pc + Med_hh_Income
+ gini + growth, data = df_cluster3, minsplit = 5, minbucket = 5)

# Create a plot of the decision tree for cluster 3 with increased complexity
rpart.plot(tree_cluster3, main = "Decision Tree for Cluster 3")
```