project2.R

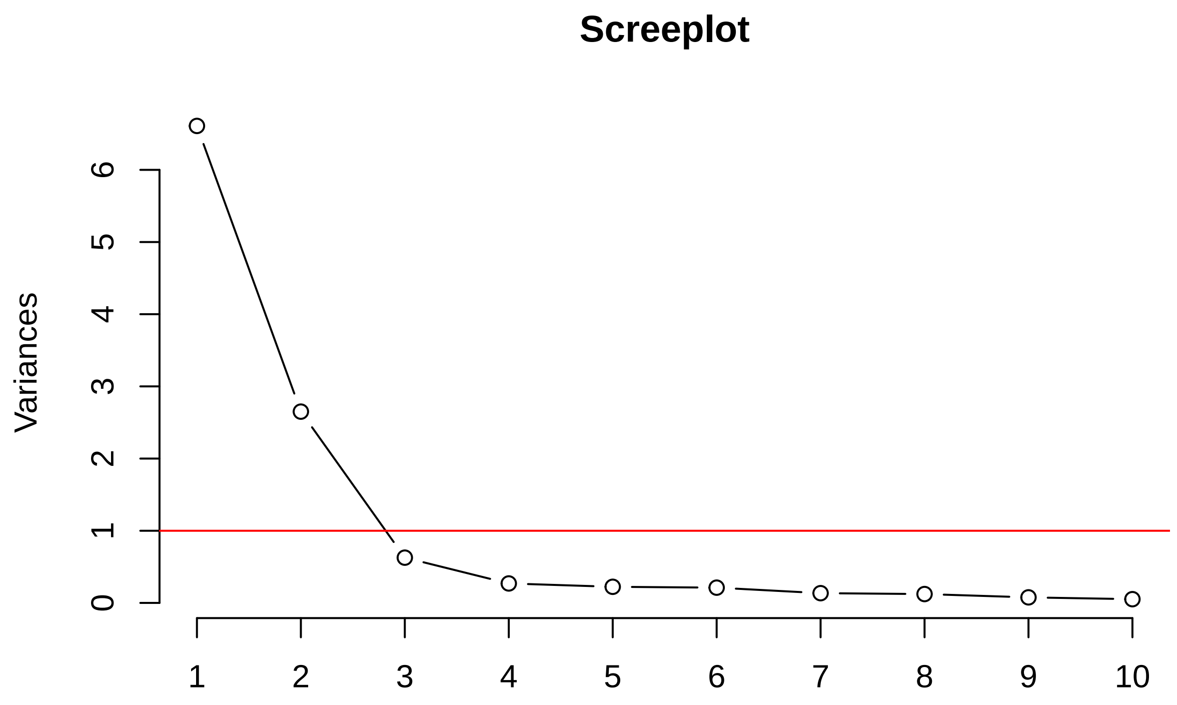
ramom

2023-06-22

```r
#Use the in-built "mtcars" data of R and do as follows in R studio with R script:

# 1. Perform the principal component analysis in the data and exact the dimensions
# based on components with eigenvalues >1, check it with screeplot as well
# and interpret the result carefully

# Load the mtcars dataset
data(mtcars)

# 1. Principal Component Analysis (PCA)
pca <- prcomp(mtcars, scale = TRUE)  # Perform PCA
summary(pca)  # Display summary
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4     PC5     PC6    PC7     PC8    PC9    PC10
## Standard deviation     2.5707 1.6280 0.79196 0.51923 0.47271 0.46000 0.3678 0.35057 0.2776 0.22811 0
## Proportion of Variance 0.6008 0.2409 0.05702 0.02451 0.02031 0.01924 0.0123 0.01117 0.0070 0.00473 0
## Cumulative Proportion  0.6008 0.8417 0.89873 0.92324 0.94356 0.96279 0.9751 0.98626 0.9933 0.99800 1
```

```r
# Screeplot
screeplot(pca, type = "line", main = "Screeplot")  # Screeplot
abline(h = 1, col = "red")  # Add line at eigenvalue 1
```



```
str(pca)
```

```
## List of 5
##  $ sdev    : num [1:11] 2.571 1.628 0.792 0.519 0.473 ...
##  $ rotation: num [1:11, 1:11] -0.363 0.374 0.368 0.33 -0.294 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  $ center  : Named num [1:11] 20.09 6.19 230.72 146.69 3.6 ...
##   ..- attr(*, "names")= chr [1:11] "mpg" "cyl" "disp" "hp" ...
##  $ scale   : Named num [1:11] 6.027 1.786 123.939 68.563 0.535 ...
##   ..- attr(*, "names")= chr [1:11] "mpg" "cyl" "disp" "hp" ...
##  $ x       : num [1:32, 1:11] -0.647 -0.619 -2.736 -0.307 1.943 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##   .. ..$ : chr [1:11] "PC1" "PC2" "PC3" "PC4" ...
##  - attr(*, "class")= chr "prcomp"
```

```
# Extract dimensions based on eigenvalues > 1
eigenvalues <- pca[[1]]^2
dimensions <- sum(eigenvalues > 1)
cat("Number of dimensions with eigenvalues > 1:", dimensions, "\n")
```

```
## Number of dimensions with eigenvalues > 1: 2
```

```r
# Interpretation:
# The screeplot displays the eigenvalues of each principal component.
# Eigenvalues represent the amount of variance explained by each component.
# We look for the "elbow" in the screeplot, where eigenvalues drop significantly.
# In this case, it seems to be around the second component.
# Thus, we choose the dimensions with eigenvalues greater than 1 as they
#explain a significant amount of variance in the data.
# In this case, the number of dimensions with eigenvalues > 1 is 2
#(the first 2 components).

# 2. Perform the principal component analysis with varimax rotation in the data
# and exact the dimensions based on eigenvalue >1 and check it with Screeplot
# as well and interpret the result carefully

# Perform PCA with varimax rotation
library(psych)  # Load 'psych' package for varimax rotation
pca_varimax <- principal(mtcars, nfactors = length(mtcars), rotate = "varimax")
summary(pca_varimax)
```

```
##
## Factor analysis with Call: principal(r = mtcars, nfactors = length(mtcars), rotate = "varimax")
##
## Test of the hypothesis that 11 factors are sufficient.
## The degrees of freedom for the model is -11  and the objective function was  0
## The number of observations was  32  with Chi Square =  0  with prob <  NA
##
## The root mean square of the residuals (RMSA) is  0
```

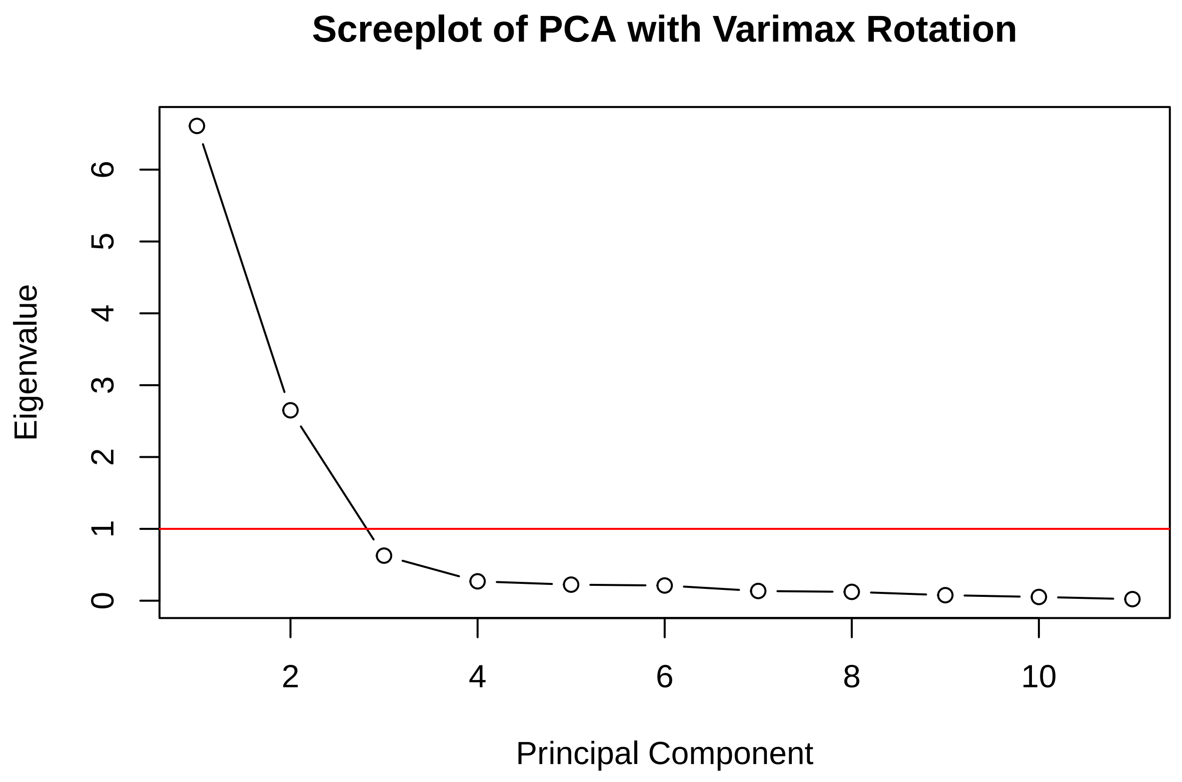
```r
#Extract dimensions based on eigenvalues > 1
eigenvalues_varimax <- pca_varimax$values
dimensions_varimax <- sum(eigenvalues > 1)
cat("Number of dimensions with eigenvalues > 1:", dimensions_varimax, "\n")
```

```
## Number of dimensions with eigenvalues > 1: 2
```

```r
str(pca_varimax)
```

```
## List of 31
##  $ values      : num [1:11] 6.608 2.65 0.627 0.27 0.223 ...
##  $ rotation    : chr "varimax"
##  $ n.obs       : int 32
##  $ communality : Named num [1:11] 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:11] "mpg" "cyl" "disp" "hp" ...
##  $ loadings    : 'loadings' num [1:11, 1:11] -0.384 0.633 0.475 0.584 -0.182 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
##   .. ..$ : chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##  $ fit         : num 1
##  $ fit.off     : num 1
##  $ fn          : chr "principal"
##  $ Call        : language principal(r = mtcars, nfactors = length(mtcars), rotate = "varimax")
##  $ uniquenesses: Named num [1:11] 4.44e-16 4.55e-15 3.00e-15 3.11e-15 2.55e-15 ...
##   ..- attr(*, "names")= chr [1:11] "mpg" "cyl" "disp" "hp" ...
##  $ complexity  : Named num [1:11] 5.34 4.11 3.25 4.1 2.39 ...
##   ..- attr(*, "names")= chr [1:11] "mpg" "cyl" "disp" "hp" ...
##  $ valid       : num [1:5] 0.829 0.94 0.667 0.878 0.744
##  $ chi         : num 3.42e-26
##  $ EPVAL       : logi NA
##  $ R2          : Named num [1:11] 1 1 1 1 1 ...
##   ..- attr(*, "names")= chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##  $ objective   : num 0
##  $ residual    : num [1:11, 1:11] 1.89e-15 -3.66e-15 -3.22e-15 -2.89e-15 2.78e-15 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
##   .. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
##  $ rms         : num 3.12e-15
##  $ factors     : int 11
##  $ dof         : num -11
##  $ null.dof    : num 55
##  $ null.model  : num 15.4
##  $ criteria    : Named num [1:3] 0 NA NA
##   ..- attr(*, "names")= chr [1:3] "objective" "" ""
##  $ STATISTIC   : num 0
##  $ PVAL        : logi NA
##  $ weights     : num [1:11, 1:11] 0.1165 0.1062 -0.063 -0.0666 0.1162 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
##   .. ..$ : chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##  $ r.scores    : num [1:11, 1:11] 1.00 4.72e-16 -2.04e-15 -1.36e-15 1.44e-15 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##   .. ..$ : chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##  $ rot.mat     : num [1:11, 1:11] 0.5062 -0.0875 0.2683 0.3659 0.2547 ...
##  $ Vaccounted  : num [1:5, 1:11] 2.886 0.262 0.262 0.262 0.262 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:5] "SS loadings" "Proportion Var" "Cumulative Var" "Proportion Explained" ...
##   .. ..$ : chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##  $ Structure   : 'loadings' num [1:11, 1:11] -0.384 0.633 0.475 0.584 -0.182 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:11] "mpg" "cyl" "disp" "hp" ...
##   .. ..$ : chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##  $ scores      : num [1:32, 1:11] 0.855 0.548 -0.654 -0.86 1.051 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:32] "Mazda RX4" "Mazda RX4 Wag" "Datsun 710" "Hornet 4 Drive" ...
##   .. ..$ : chr [1:11] "RC5" "RC2" "RC1" "RC3" ...
##  - attr(*, "class")= chr [1:2] "psych" "principal"
```

```r
# Screeplot
plot(1:length(eigenvalues_varimax), eigenvalues_varimax, type = "b",
     xlab = "Principal Component", ylab = "Eigenvalue",
     main = "Screeplot of PCA with Varimax Rotation")
abline(h = 1, col = "red")
```



```
# Interpretation
cat("The screeplot shows the eigenvalues for each principal
    component after varimax rotation.\n")
```

```
## The screeplot shows the eigenvalues for each principal
##     component after varimax rotation.
```

```
cat("Eigenvalues represent the amount of variance
    explained by each component.\n")
```

```
## Eigenvalues represent the amount of variance
##     explained by each component.
```

```
cat("We select the dimensions based on eigenvalues greater than 1.\n")
```

```
## We select the dimensions based on eigenvalues greater than 1.
```

```
cat("In this case,", dimensions, "dimensions have
    eigenvalues greater than 1,\n")
```

```
## In this case, 2 dimensions have
##     eigenvalues greater than 1,
cat("indicating that they explain more variance than
    a single original variable.\n")
```

```
## indicating that they explain more variance than
##     a single original variable.
```

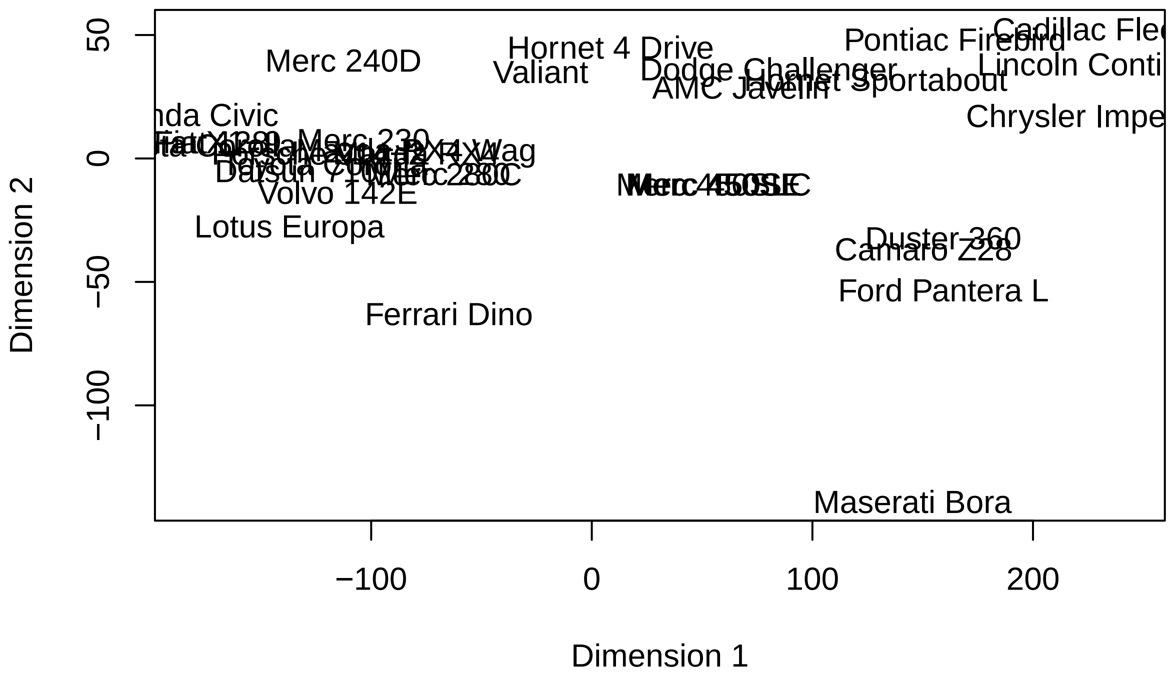
```
cat("Varimax rotation simplifies the interpretation of
    components by maximizing\n")
```

```
## Varimax rotation simplifies the interpretation of
##     components by maximizing
```

```
cat("the variance of each component and providing a clearer
    factor loading structure.\n")
```

```
## the variance of each component and providing a clearer
##     factor loading structure.
```

```
# 3. Perform the classical multidimensional scaling in the data, revise the
#results using stress values and interpret the result carefully
# Perform Classical Multidimensional Scaling (MDS)
mds <- cmdscale(dist(mtcars))

# Plot the MDS solution
plot(mds, type = "n", xlab = "Dimension 1", ylab = "Dimension 2")
text(mds, labels = row.names(mtcars))
```



```r
# Calculate stress values
dist_orig <- dist(mtcars)

#custom stress function
stress <- function(mds, dist_orig) {
  n <- nrow(mds)
  dist_mds <- as.matrix(dist(mds))
  sum_diff <- sum((dist_orig - dist_mds)^2)
  sum_orig <- sum(dist_orig^2)
  stress_val <- sqrt(sum_diff / sum_orig)
  return(stress_val)
}

stress <- stress(mds, dist_orig)
```

```
## Warning in dist_orig - dist_mds: longer object length is not a multiple of shorter object length
```

```r
# Print stress value
cat("Stress value:", stress, "\n")
```

```
## Stress value: 1.106875
```

```r
# Interpretation
```

```r
cat("The classical MDS represents the dissimilarity between
    observations in a lower-dimensional space.\n")
```

```
## The classical MDS represents the dissimilarity between
##     observations in a lower-dimensional space.
```

```r
cat("The MDS solution is plotted in a two-dimensional space,
    with the points representing the observations.\n")
```

```
## The MDS solution is plotted in a two-dimensional space,
##     with the points representing the observations.
```

```r
cat("Stress is a measure of the discrepancy between the original
    dissimilarity matrix and the distances in the MDS solution.\n")
```

```
## Stress is a measure of the discrepancy between the original
##     dissimilarity matrix and the distances in the MDS solution.
```

```r
cat("A lower stress value indicates a better fit between the original
    distances and the distances in the reduced space.\n")
```

```
## A lower stress value indicates a better fit between the original
##     distances and the distances in the reduced space.
```

```r
cat("In this case, the stress value is", round(stress, 4), "which
    indicates the goodness of fit for the MDS solution.\n")
```

```
## In this case, the stress value is 1.1069 which
##     indicates the goodness of fit for the MDS solution.
```

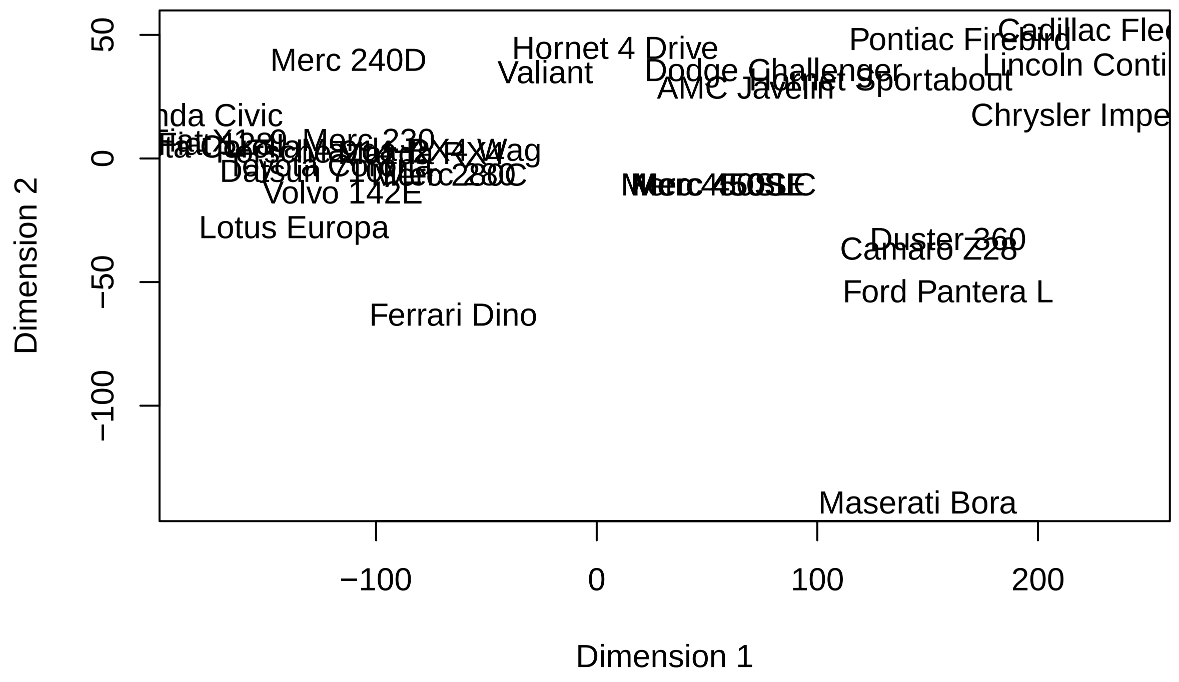
```r
cat("Interpretation of the MDS plot should consider the proximity of points,
    with closer points representing similar observations.\n")
```

```
## Interpretation of the MDS plot should consider the proximity of points,
##     with closer points representing similar observations.
```

```r
######################################3
#In R, the cmdscale() function does not directly provide a stress value.
#Instead, the stress value is typically calculated using an external package
#called isoMDS from the MASS library. Here's the revised script that performs
#classical multidimensional scaling (MDS) using the isoMDS function,
#calculates the stress value, and provides interpretation

# Load the required package
library(MASS)

# Perform Classical Multidimensional Scaling (MDS)
mds <- isoMDS(dist(mtcars))
```

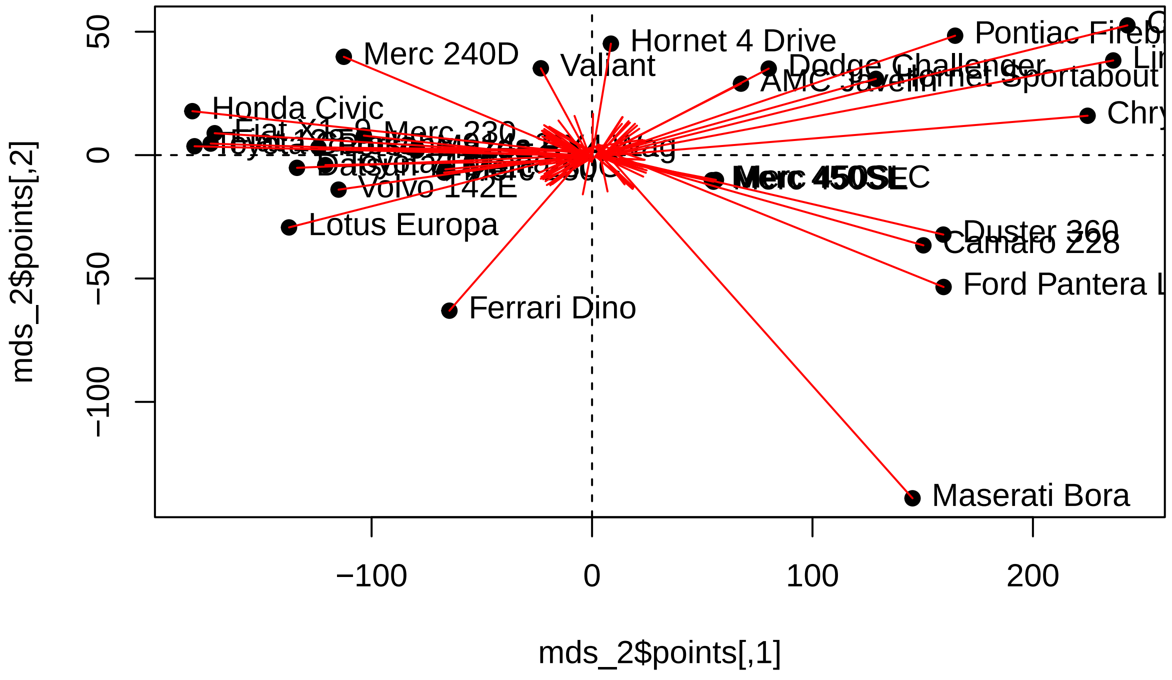
```
## initial  value 0.046224
## iter   5 value 0.028189
## iter  10 value 0.018898
## iter  15 value 0.014170
## iter  20 value 0.012216
## iter  25 value 0.011553
## iter  30 value 0.011141
## iter  35 value 0.010908
## iter  40 value 0.010685
## iter  45 value 0.010563
## iter  50 value 0.010488
## final  value 0.010488
## stopped after 50 iterations
```

```
# Plot the MDS solution
plot(mds$points, type = "n", xlab = "Dimension 1", ylab = "Dimension 2")
text(mds$points, labels = row.names(mtcars))
```



```
#make the plot Alternative approach with Sammon's stres better
mds_2 <- MASS::sammon(dist_orig, trace = FALSE)
plot(mds_2$points, pch = 19)
abline(h=0, v=0, lty=2)
text(mds_2$points, pos = 4, labels = rownames(mtcars))

# Compare with PCA (first two PCs):
```

```
arrows(x0 = mds_2$points[,1], y0 =
      mds_2$points[,2], x1 = pca$x[,1], y1 = pca$x[,2], col='red', pch=19, cex=0.5)
```



```
# Calculate stress value
stress <- mds$stress

# Print stress value
cat("Stress value:", stress, "\n")
```

```
## Stress value: 0.01048811
```

```
# Interpretation
cat("The classical MDS represents the dissimilarity between observations
    in a lower-dimensional space.\n")
```

```
## The classical MDS represents the dissimilarity between observations
##     in a lower-dimensional space.
```

```
cat("The MDS solution is plotted in a two-dimensional space, with the
    points representing the observations.\n")
```

```
## The MDS solution is plotted in a two-dimensional space, with the
##     points representing the observations.
```

```
cat("Stress is a measure of the discrepancy between the original dissimilarity
    matrix and the distances in the MDS solution.\n")
```

```
## Stress is a measure of the discrepancy between the original dissimilarity
##     matrix and the distances in the MDS solution.
```

```
cat("A lower stress value indicates a better fit between the original
    distances and the distances in the reduced space.\n")
```

```
## A lower stress value indicates a better fit between the original
##     distances and the distances in the reduced space.
```

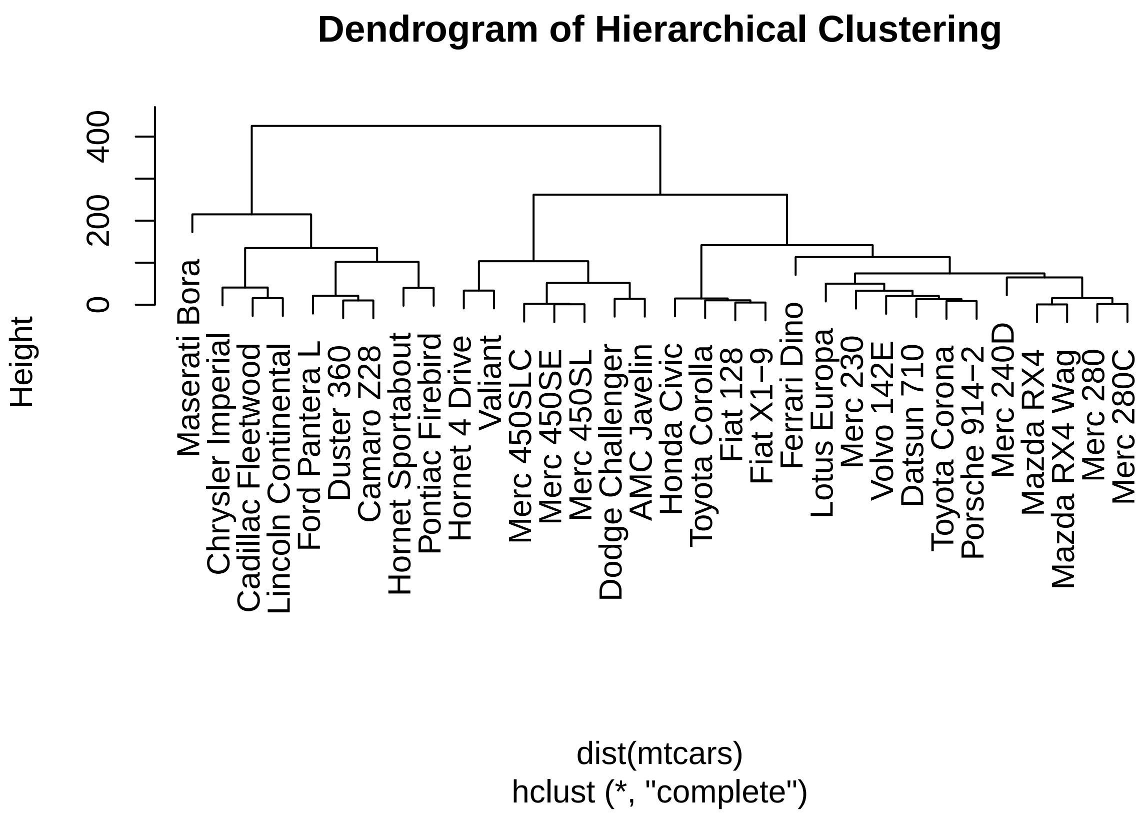
```
cat("In this case, the stress value is", round(stress, 4), "which
    indicates the goodness of fit for the MDS solution.\n")
```

```
## In this case, the stress value is 0.0105 which
##     indicates the goodness of fit for the MDS solution.
```

```
cat("Interpretation of the MDS plot should consider the proximity of
    points, with closer points representing similar observations.\n")
```

```
## Interpretation of the MDS plot should consider the proximity of
##     points, with closer points representing similar observations.
```

```
# 4. Perform the hierarchical cluster analysis in the data and determine
#the number of clusters to exact using the dendogram and cut at the various
#distances with justification

# Perform Hierarchical Cluster Analysis
hc <- hclust(dist(mtcars))

# Plot the Dendrogram
plot(hc, main = "Dendrogram of Hierarchical Clustering")
```



```r
# Determine the number of clusters using the dendrogram
clusters <- cutree(hc, k = 2:length(mtcars))
cluster_counts <- table(clusters)
print(cluster_counts)
```

```
## clusters
##   1   2   3   4   5   6   7   8   9  10  11 
## 121  60  46  23  23  15  14   9   6   2   1
```

```r
# Determine the number of clusters using the dendrogram
num_clusters <- length(unique(cutree(hc, k = length(mtcars))))
cat("Number of clusters:", num_clusters, "\n")
```

```
## Number of clusters: 11
```

```r
# Cut the dendrogram at various distances
cut_distances <- c(10, 15, 20)  # Adjust the distances as needed
for (distance in cut_distances) {
  clusters <- cutree(hc, h = distance)
  num_clusters <- length(unique(clusters))
  cat("Number of clusters at distance", distance, ":", num_clusters, "\n")
  cat("Cluster sizes:", table(clusters), "\n\n")
}
```

```
## Number of clusters at distance 10 : 26
## Cluster sizes: 2 1 1 1 1 1 1 1 2 3 1 1 1 2 1 1 2 1 1 1 1 1 1 1 1 1
##
## Number of clusters at distance 15 : 21
## Cluster sizes: 2 3 1 1 1 2 1 1 2 3 1 1 1 4 2 1 1 1 1 1 1
##
## Number of clusters at distance 20 : 19
## Cluster sizes: 4 3 1 1 1 2 1 1 3 2 1 4 2 1 1 1 1 1 1
```

```
# 5. Perform the k-means cluster analysis in the data based on the number of
#clusters identified using dendogram and interpret the result carefully

## Perform K-means Cluster Analysis
kmeans_result <- kmeans(scale(mtcars), centers = num_clusters)

# Add Cluster Labels to the Dataset
mtcars$cluster <- as.factor(kmeans_result$cluster)

# Interpretation
cat("Number of clusters identified:", num_clusters, "\n")
```

```
## Number of clusters identified: 19
```

```
cat("Cluster Sizes:", table(mtcars$cluster), "\n\n")
```

```
## Cluster Sizes: 2 2 1 1 3 1 3 2 2 1 2 1 3 1 1 2 1 2 1
```

```
# Summary of Cluster Centers
cluster_centers <- data.frame(cluster = 1:num_clusters, kmeans_result$centers)
print(cluster_centers)
```

```
##    cluster         mpg        cyl        disp          hp        drat         wt        qsec
## 1        1 -0.77827533  1.0148821  0.76875205  2.22879386  0.53010805  0.1561131 -1.84602954 -0.8680
## 2        2  1.34551931 -1.2248578 -0.99260264 -0.65177407  0.94157008 -1.4213699 -0.58689609  0.1240
## 3        3 -0.89442035  1.0148821  1.68856165  1.21512565 -0.68557523  2.1745964 -0.23993487 -0.8680
## 4        4 -0.96078893  1.0148821  1.04308123  1.43390296 -0.72298087  0.3605164 -1.12412636 -0.8680
## 5        5  1.84328368 -1.2248578 -1.24622266 -1.18170134  0.99144427 -1.2542702  0.88116024  1.1160
## 6        6 -1.12671039  1.0148821  0.96239618  1.43390296  0.24956575  0.6364610 -1.36476075 -0.8680
## 7        7  0.46613559 -1.2248578 -0.76533975 -0.90458341  0.32437703 -0.2884643  1.74669790  1.1160
## 8        8 -0.78657141  1.0148821  0.64772447  0.04831332 -1.19990278  0.2659799 -0.42740585 -0.8680
## 9        9 -1.60788262  1.0148821  1.89834278  0.92342257 -1.18119996  2.1664202  0.02868026 -0.8680
## 10      10 -0.14777380  1.0148821  1.36582144  0.41294217 -0.96611753  0.6415711 -0.44699237 -0.8680
## 11      11 -0.05651700 -0.1049878  0.08696336 -0.57155572 -1.26536265  0.1228975  1.10873695  1.1160
## 12      12 -0.06481307 -0.1049878 -0.69164740  0.41294217  0.04383473 -0.4570970 -1.31439542 -0.8680
## 13      13 -0.62894602  1.0148821  0.36371309  0.48586794 -0.98482035  0.6569013 -0.10189654 -0.8680
## 14      14  1.71054652 -1.2248578 -1.25079481 -1.38103178  2.49390411 -1.6375265  0.37564148  1.1160
## 15      15  0.15088482 -0.1049878 -0.57061982 -0.53509284  0.56751369 -0.6103996 -0.77716515 -0.8680
## 16      16 -0.26391882 -0.1049878 -0.50929918 -0.34548584  0.60491932  0.2276543  0.42041067  1.1160
## 17      17  0.15088482 -0.1049878 -0.57061982 -0.53509284  0.56751369 -0.3497853 -0.46378082 -0.8680
## 18      18  0.33339843 -1.2248578 -0.93773681 -0.66635923  0.71713624 -0.6819407  0.42320874  1.1160
## 19      19 -0.23073453  1.0148821  1.04308123  0.41294217 -0.83519779  0.2276543 -0.46378082 -0.8680
```

```
# The script then performs k-means cluster analysis using the kmeans() function
#on the scaled "mtcars" dataset. The resulting clusters are assigned to each
#observation, and the cluster labels are added to the dataset.
#
# The interpretation section displays the number of clusters identified
#and the sizes of each cluster using the table() function. It provides an
#overview of the distribution of observations in each cluster.
#
# Additionally, the script presents a summary of the cluster centers, which
#represent the mean values of the variables within each cluster.
#The cluster_centers data frame displays the cluster number along with
#the corresponding center values for each variable.

#Interpreting the results requires analyzing the characteristics of
#each cluster based on the cluster centers and understanding the context of
#the data. You can examine the variables that contribute most to the
#differences between the clusters and interpret the clusters based
#on their specific characteristics and similarities.
```