Practical No.4

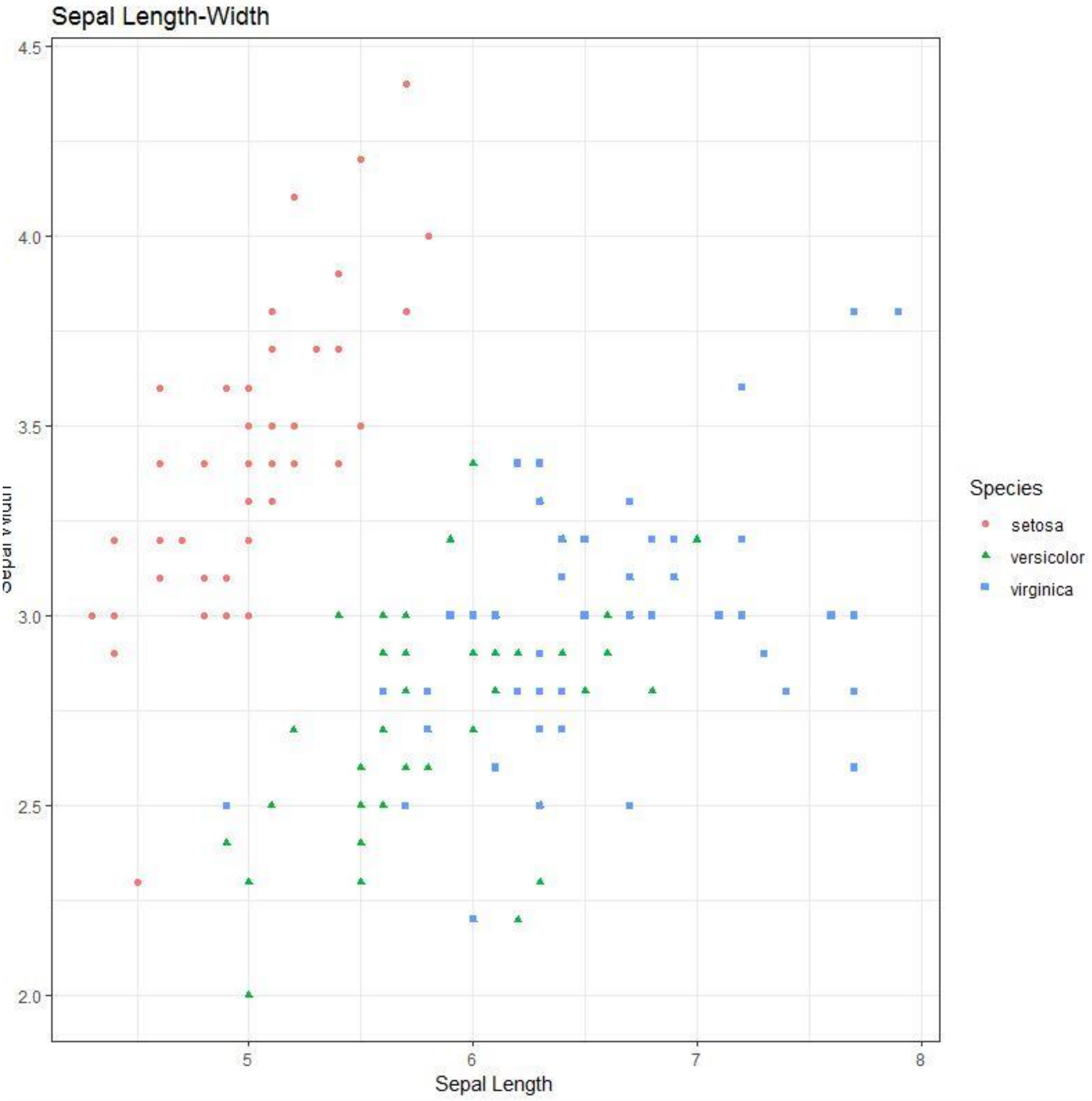
AIM: Practical of Clustering.

Theory: This dataset is very commonly used for Overview of data, Data Visualization and Clustering model. It includes three iris species with 50 samples each as well as some properties about each flower. One flower species is linearly separable from the other two, but the other two are not linearly separable from each other.

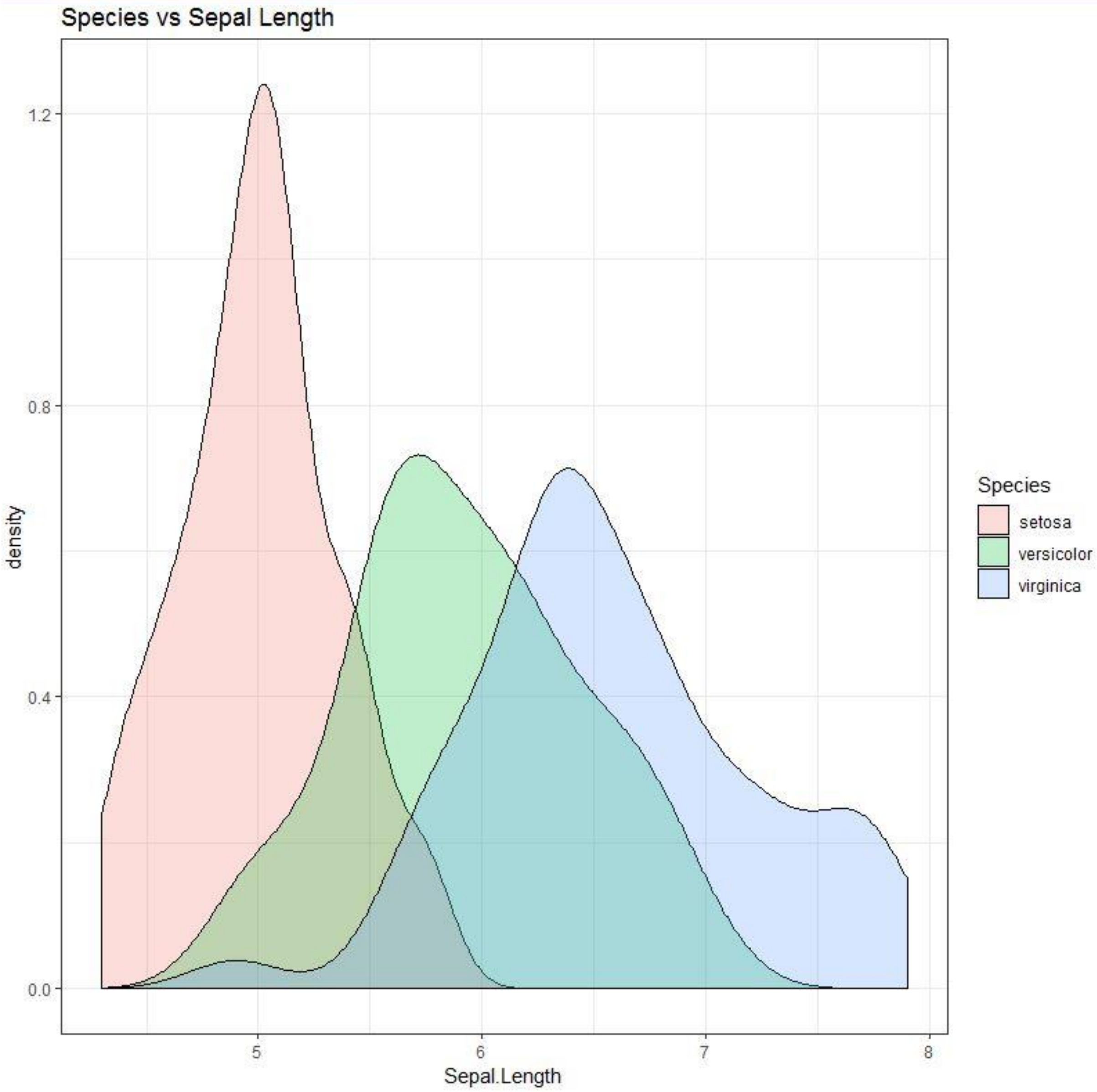
The given columns in this dataset are:
i> Id
ii> SepalLength (Cm)
iii>SepalWidth (Cm)
iv> PetalLength (Cm)
v> PetalWidth (Cm)
vi> Species

Lets visualize this dataSet and Cluster with kmeans
Solution approach –
IRIS Data, Basic Visualization before Clustering

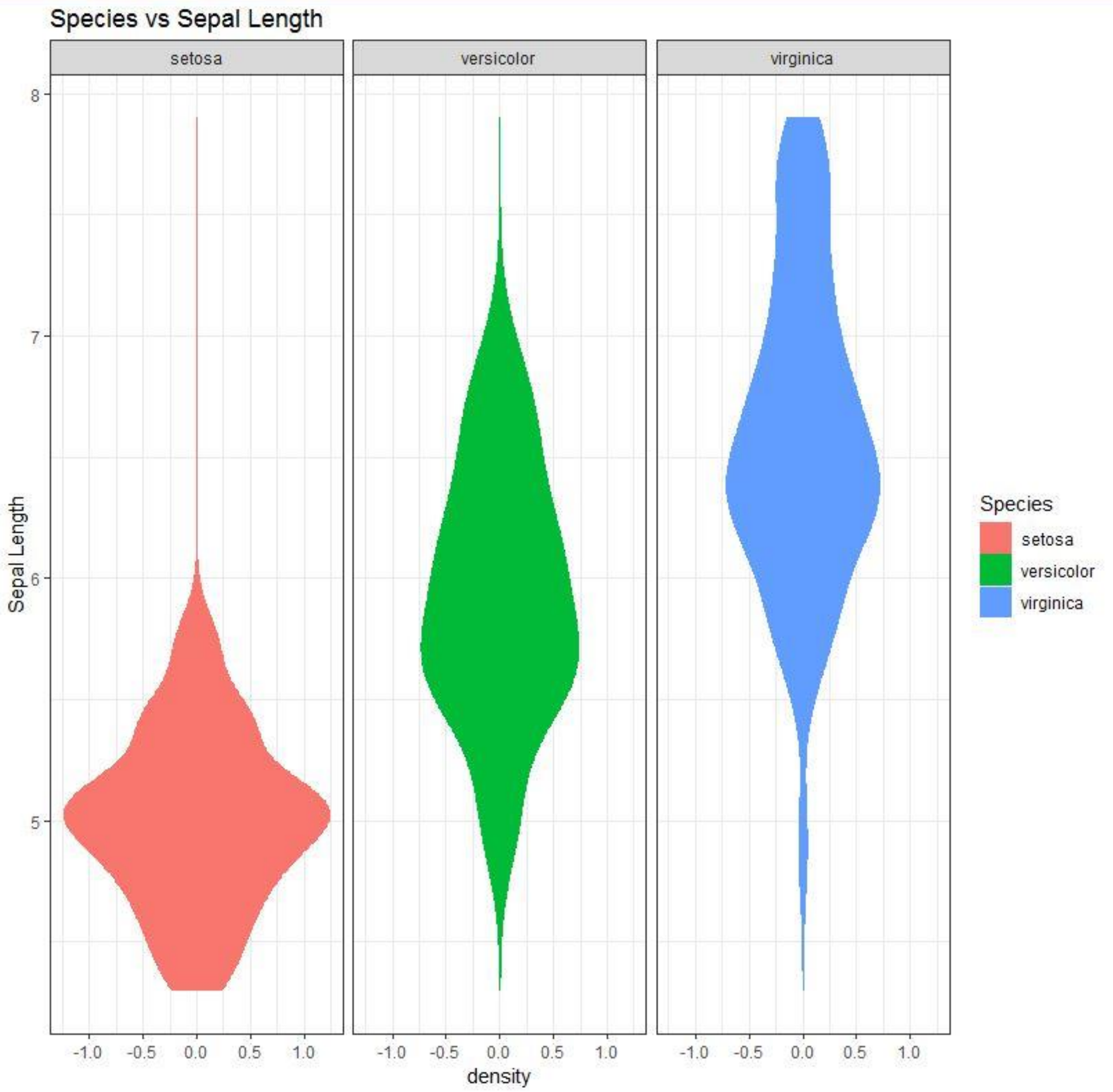
```
> install.packages("ggplot2")
> library(ggplot2)

> scatter <- ggplot(data=iris, aes(x = Sepal.Length, y = Sepal.Width))
> scatter + geom_point(aes(color=Species, shape=Species)) +
+   theme_bw()+
+   xlab("Sepal Length") +  ylab("Sepal Width") +
+   ggtitle("Sepal Length-Width")
```



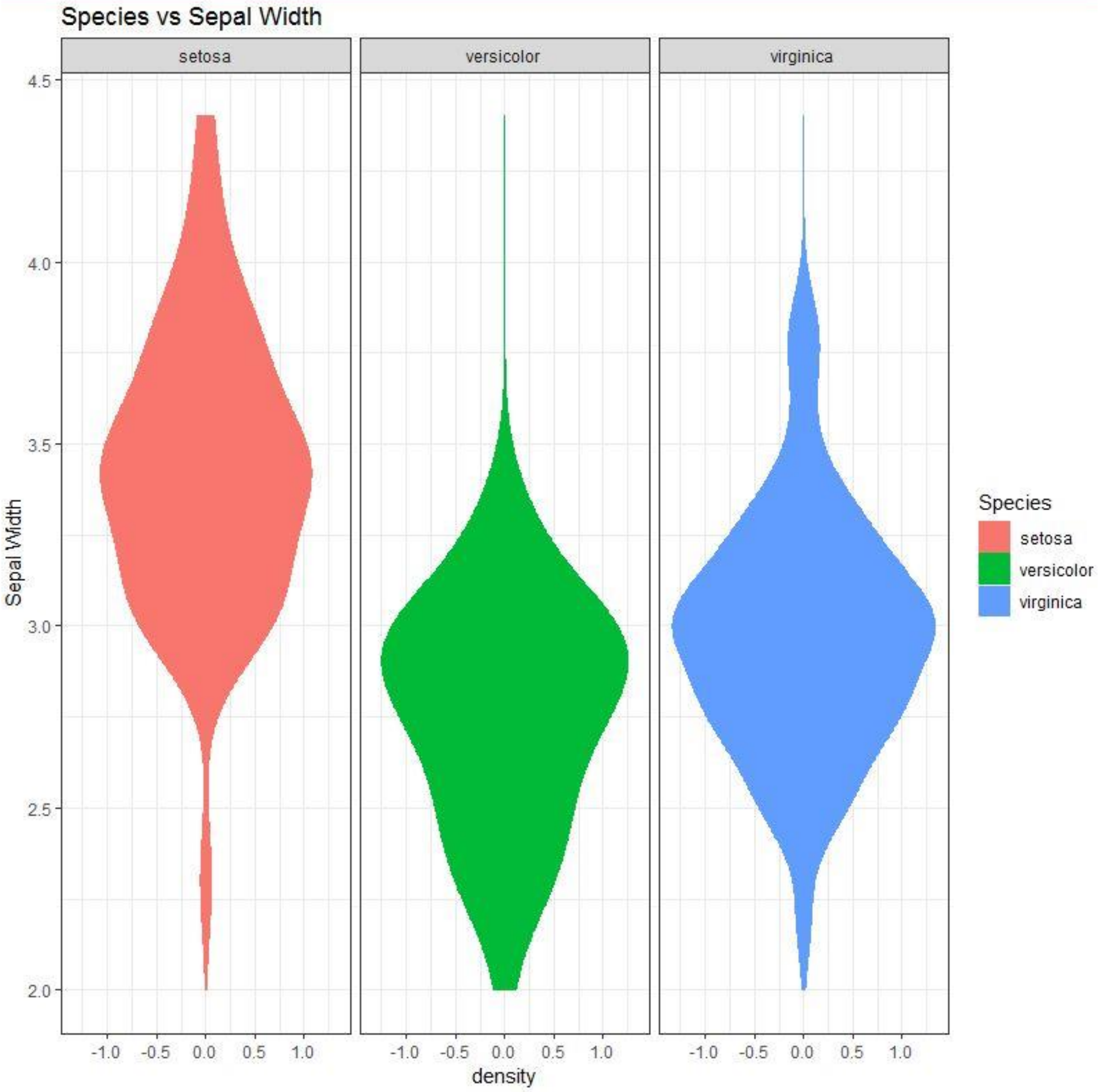
```
> ggplot(data=iris, aes(Sepal.Length, fill = Species))+
+   theme_bw()+
+   geom_density(alpha=0.25)+
+   labs(x = "Sepal.Length", title="Species vs Sepal Length")
```



```
> vol <- ggplot(data=iris, aes(x = Sepal.Length))

> vol + stat_density(aes(ymax = ..density..,  ymin = -..density..,
+                         fill = Species, color = Species),
+                     geom = "ribbon", position = "identity") +
+   facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Length", title="Spec
ies vs Sepal Length")
```

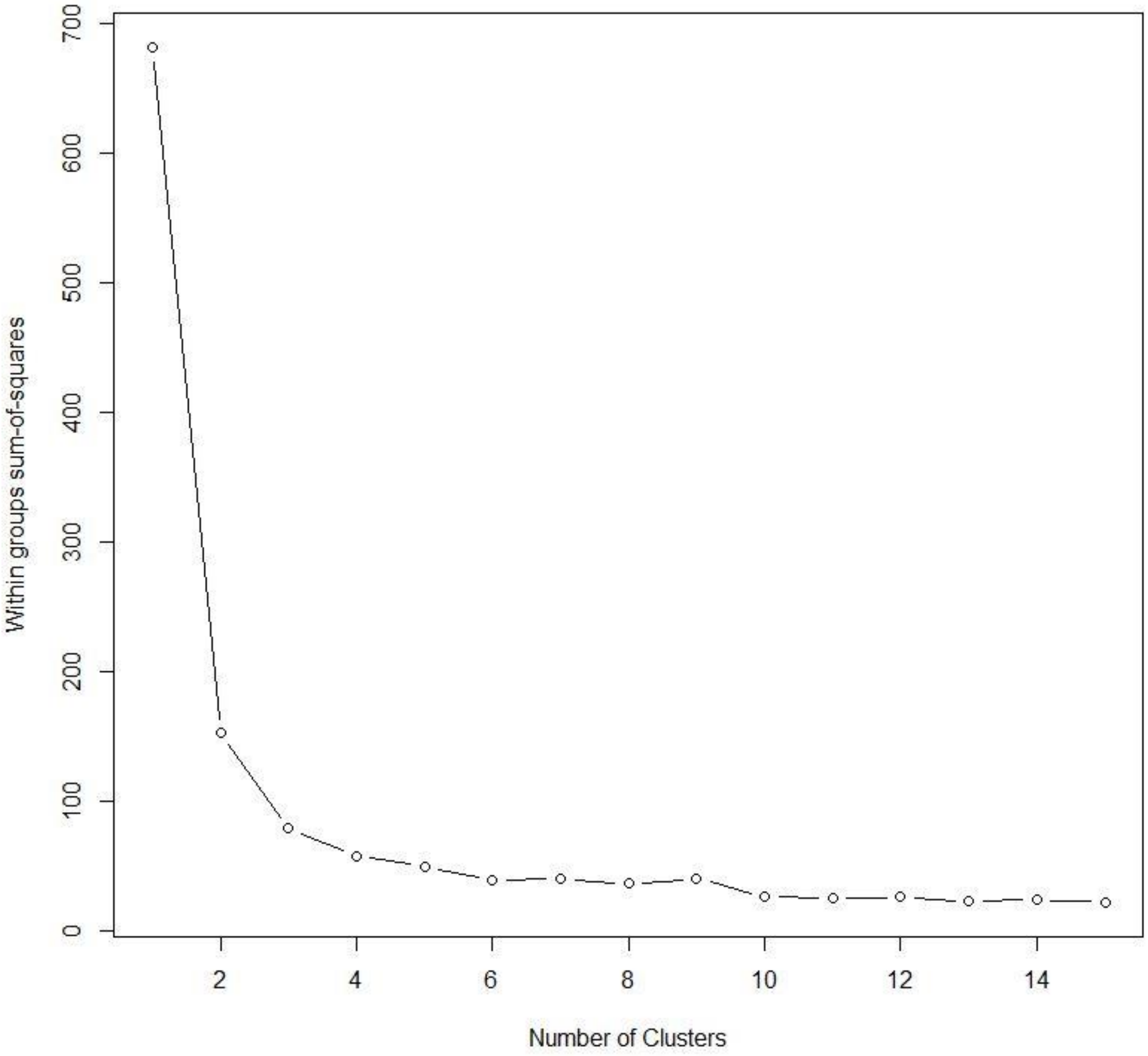


```
> vol <- ggplot(data=iris, aes(x = Sepal.Width))
> vol + stat_density(aes(ymax = ..density..,  ymin = -..density..,
+                        fill = Species, color = Species),
+                     geom = "ribbon", position = "identity") +
+   facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Width", title="Species vs Sepal Width")
```



Clustering Data :: Method-1

```
> irisData <- iris[,1:4]
> totalwSS<-c()

# kmeans clustering for 15 times in a loop
> for (i in 1:15)
+ {
+   clusterIRIS <- kmeans(irisData, centers=i)
+   totalwSS[i]<-clusterIRIS$tot.withinss
+ }

# Scree plot - Use plot function to plot values of tot_wss against no-of-clusters
> plot(x=1:15,                          # x= No of clusters, 1 to 15
+      y=totalwSS,                        # tot_wss for each
+      type="b",                        # Draw both points as also connect them
+      xlab="Number of Clusters",
+      ylab="Within groups sum-of-squares")
```



Clustering Data :: Method-2
Using NbClust - Uses huge no of cluster suitability measuring critera

```
> install.packages("NbClust")
> library(NbClust)
```

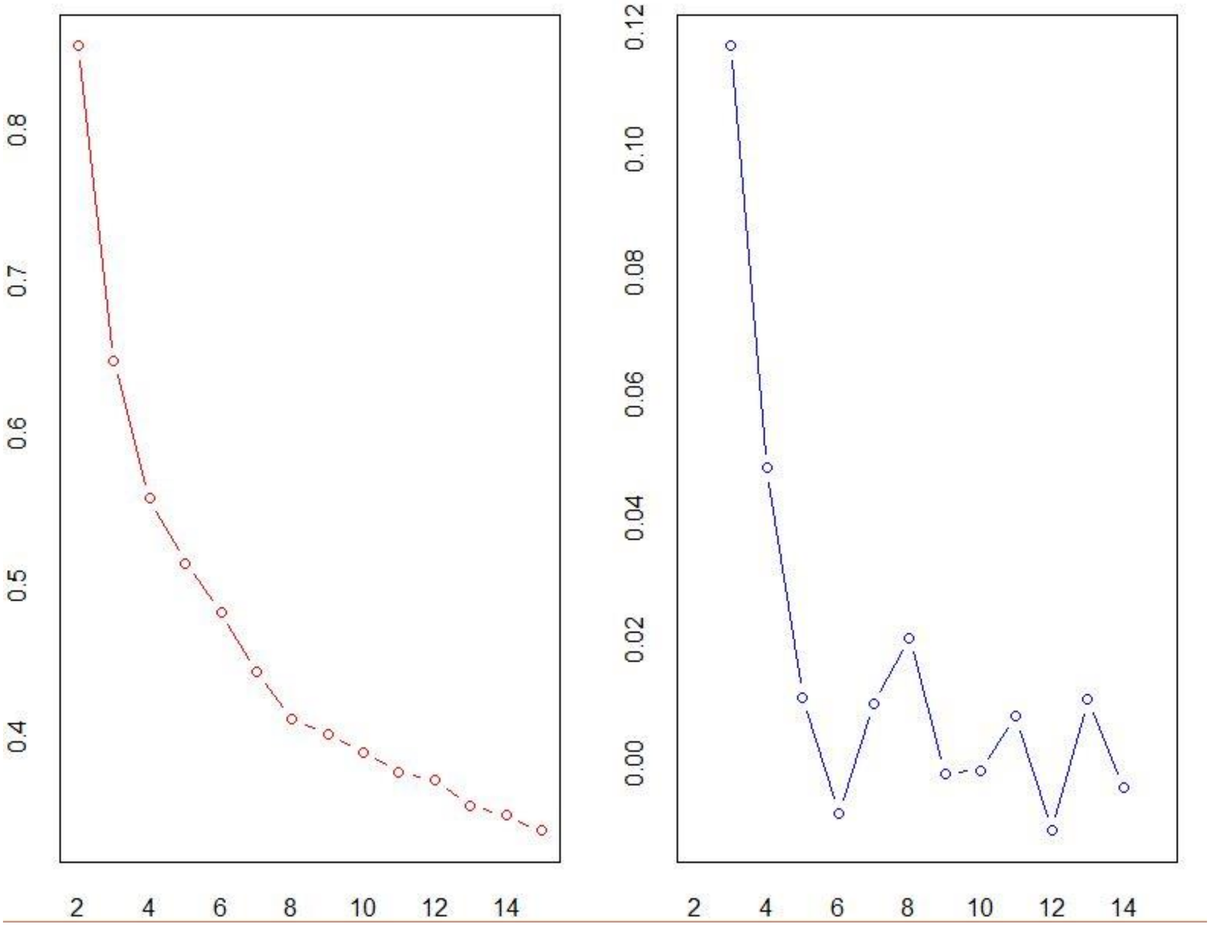
# Set margins as: c(bottom, left, top, right)

```
> par(mar = c(2,2,2,2))
```

# NbClust measures appropirateness of cluster on a number of indices. # By default, it checks from 2 clusters to 15 clusters

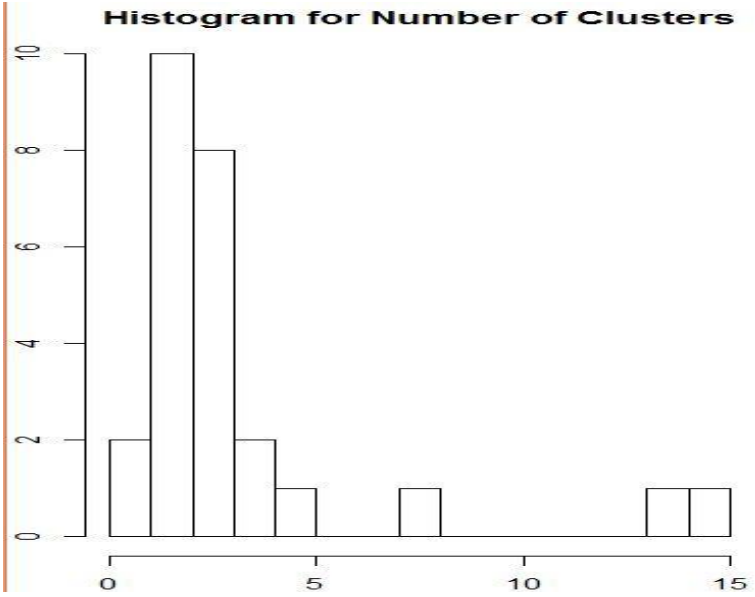
```
> nb <- NbClust(irisData, method = "kmeans") # Takes time
```

```
*** : The Hubert index is a graphical method of determining the number of clusters.
                In the plot of Hubert index, we seek a significant knee that corresponds to a
                significant increase of the value of the measure i.e the significant peak in Hubert
                index second differences plot.

*** : The D index is a graphical method of determining the number of clusters.
                In the plot of D index, we seek a significant knee (the significant peak in Dindex
                second differences plot) that corresponds to a significant increase of the value of
                the measure.

******************************************************************
* Among all indices:
* 10 proposed 2 as the best number of clusters
* 8 proposed 3 as the best number of clusters
* 2 proposed 4 as the best number of clusters
* 1 proposed 5 as the best number of clusters
* 1 proposed 8 as the best number of clusters
* 1 proposed 14 as the best number of clusters
* 1 proposed 15 as the best number of clusters

                   ***** Conclusion *****

* According to the majority rule, the best number of clusters is  2


******************************************************************
```



# Draw a histogram denoting how various indices have voted for number of clusters.
# Out of 26 indicies, most (10) voted for 2 clusters, eight voted
# for 3 clusters and remaining eight (26-10-8) for other no of clusters
# Histogram, breaks =15 as our algorithm checks from 2 to 15 clusters

```
> hist(nb$Best.nc[1,], breaks = 15, main="Histogram for Number of Clusters")
```



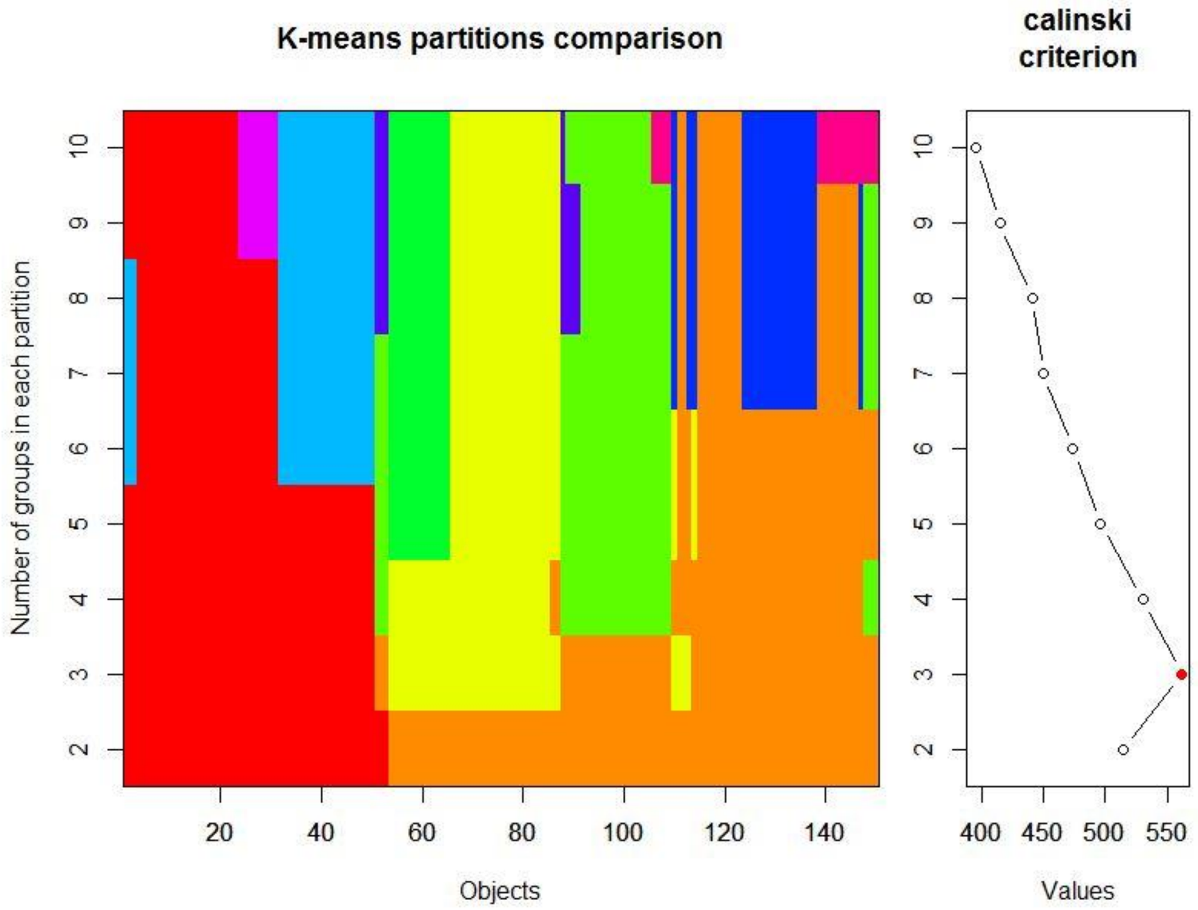
Clustering Data :: Method-3

calinski criterion is similar to finding ratio of between-cluster-variance/within-cluster variance

```
> install.packages("vegan")
> library(vegan)
Loading required package: permute
Loading required package: lattice
This is vegan 2.5-4
```

# Test for clusters 1 to 10

```
> modelData <- cascadeKM(irisData, 1, 10, iter = 100)

> plot(modelData, sortg = TRUE)
```



# Groups against BC/WC values

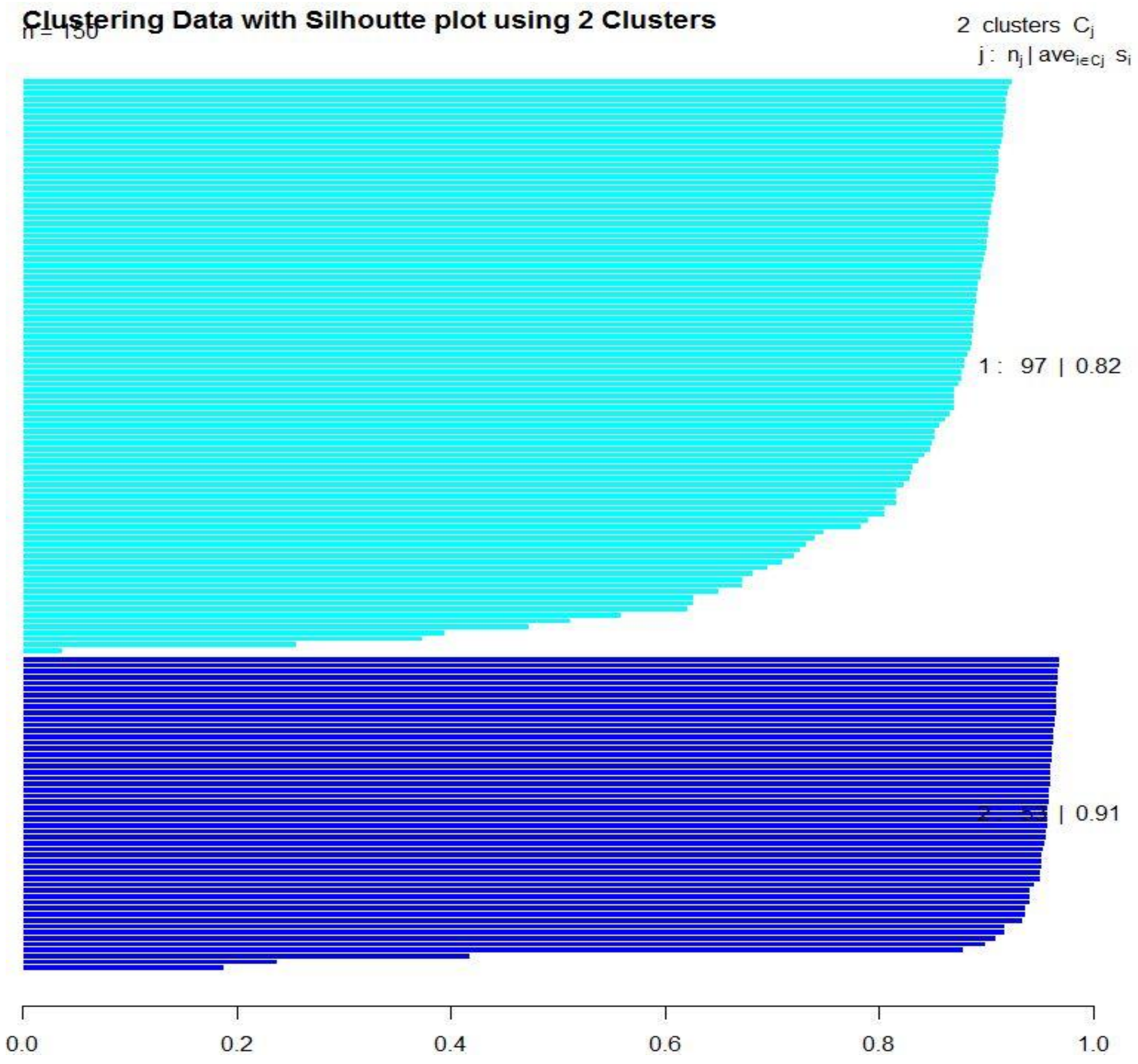
```
> modelData$results[2,]
 1 groups  2 groups  3 groups  4 groups  5 groups  6 groups  7 groups  8 groups  9 groups 10 g
roups
       NA  513.9245  561.6278  530.7658  495.5415  473.8506  449.6410  440.6205  414.5753
394.7207
```

```
> which.max(modelData$results[2,])
3 groups
       3
```

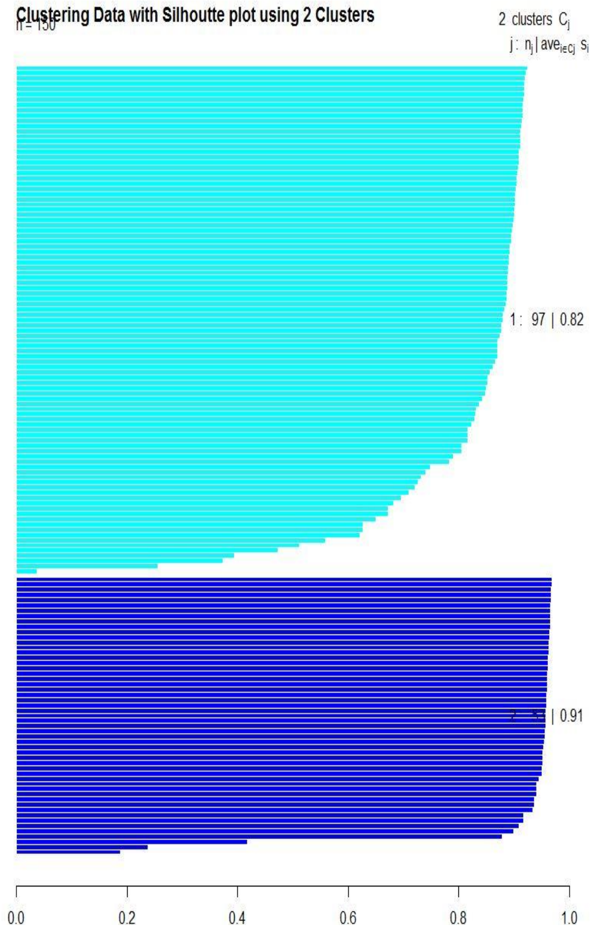
Clustering Data with Silhoutte plot :: Method-4

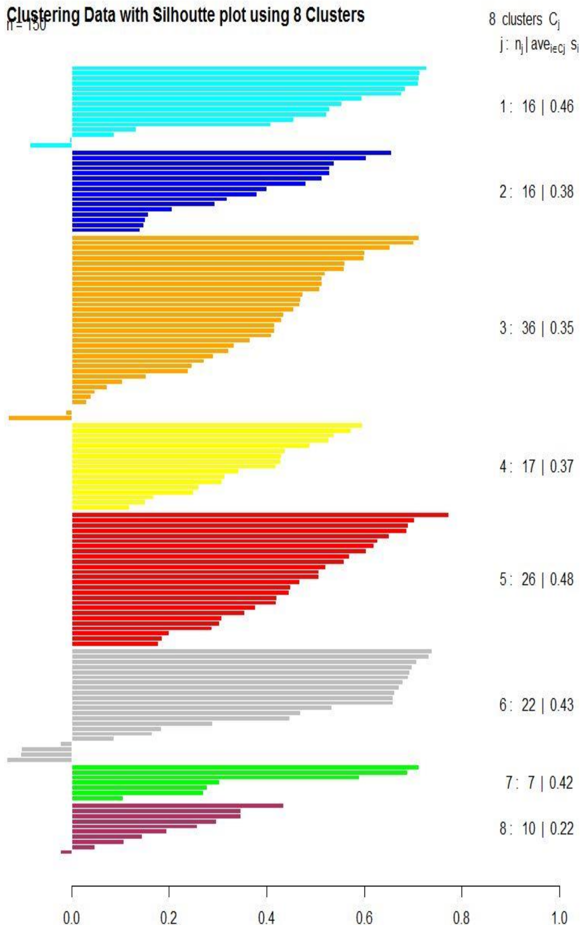
Try with 2 Clusters first —

```
# For silhoutte()
> library(cluster)
> cl <- kmeans(iris[,-5], 2)

# Compute and returns the distance matrix computed by using euclidean distance measure to
compute # the distances between the rows of a data matrix.
> dis <- dist(iris[,-5])^2

# Get silhoutte coefficient
> sil = silhouette (cl$cluster, dis)

> plot(sil, main = "Clustering Data with Silhoutte plot using 2 Clusters", col = c("cyan", "blu
e"))
```



```
> library(cluster)   # For silhoutte()
> cl <- kmeans(iris[,-5], 8)
# Compute and returns the distance matrix computed by using euclidean distance measure to
compute

# the distances between the rows of a data matrix.
> dis <- dist(iris[,-5])^2

# Get silhoutte coefficient
> sil = silhouette (cl$cluster, dis)

> plot(sil, main = "Clustering Data with Silhoutte plot using 8 Clusters", col = c("cyan", "blu
e", "orange", "yellow", "red", "gray", "green", "maroon"))
```





Analyze Clustering Tendency

Calculate Hopkin’s statistic for iris and random dataset

```
# get_clust_tendency() assesses hopkins stat
> install.packages("factoextra")

> library(factoextra)
Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at https://goo.gl/13EFC
Z

# 1. Given a vector of numbers or a column of a dataframe

# Generate uniform random numbers as per its min and max values
> install.packages("clustertend")

# Another package for hopkins() function
> library(clustertend)

# 1. Given a vector of numbers or a column of a dataframe

# Generate uniform random numbers as per its min and max values
> genx<-function(x){
+   runif(length(x), min(x), (max(x)))
+ }

# 2. Generate random data by applying function over each column
> random_df <- apply(iris[,-5], 2, genx)
> random_df <- as.data.frame(random_df)


# 3. Standardize both data sets
> iris[,-5] <- scale(iris[,-5])  # By default, center = T, scale = T
> random_df <- scale(random_df)

# 4. Compute Hopkins statistic for iris dataset
> res <- get_clust_tendency(iris[,-5],
+                          n = nrow(iris) -1 ,
+                          graph = FALSE)

> res$hopkins_stat
[1] 0.1815219

# 5. Also calculate using function, hopkins(),
# of clustertend package

> hopkins(iris[,-5], n = nrow(iris) -1)
$H
[1] 0.1903924
```

```
# 6. Compute Hopkins statistic for a random dataset
> res <- get_clust_tendency(random_df, n = nrow(random_df)-1,
+                          graph = FALSE)
> res$hopkins_stat
[1] 0.4980571
```

All Command:

```
install.packages("ggplot2")
library(ggplot2)
scatter <- ggplot(data=iris, aes(x = Sepal.Length, y = Sepal.Width))
scatter + geom_point(aes(color=Species, shape=Species)) +
  theme_bw()+
  xlab("Sepal Length") +  ylab("Sepal Width") +
  ggtitle("Sepal Length-Width")

ggplot(data=iris, aes(Sepal.Length, fill = Species))+
  theme_bw()+
  geom_density(alpha=0.25)+
  labs(x = "Sepal.Length", title="Species vs Sepal Length")

vol <- ggplot(data=iris, aes(x = Sepal.Length))

vol + stat_density(aes(ymax = ..density..,  ymin = -..density..,
                   fill = Species, color = Species),
               geom = "ribbon", position = "identity") +
  facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Length",
title="Species vs Sepal Length")

vol <- ggplot(data=iris, aes(x = Sepal.Width))

vol + stat_density(aes(ymax = ..density..,  ymin = -..density..,
                   fill = Species, color = Species),
               geom = "ribbon", position = "identity") +
  facet_grid(. ~ Species) + coord_flip() + theme_bw()+labs(x = "Sepal Width",
title="Species vs Sepal Width")

irisData <- iris[,1:4]
totalwSS<-c()

for (i in 1:15)
{
  clusterIRIS <- kmeans(irisData, centers=i)
  totalwSS[i]<-clusterIRIS$tot.withinss
}
```

```
plot(x=1:15,                    # x= No of clusters, 1 to 15
    y=totalwSS,                  # tot_wss for each
    type="b",                   # Draw both points as also connect them
    xlab="Number of Clusters",
    ylab="Within groups sum-of-squares")


install.packages("NbClust")
library(NbClust)
par(mar = c(2,2,2,2))
nb <- NbClust(irisData, method = "kmeans")
hist(nb$Best.nc[1,], breaks = 15, main="Histogram for Number of Clusters")


install.packages("vegan")
library(vegan)

modelData <- cascadeKM(irisData, 1, 10, iter = 100)  # Test for clusters 1 to 10
plot(modelData, sortg = TRUE)

modelData$results[2,]

which.max(modelData$results[2,])

library(cluster)
cl <- kmeans(iris[,-5], 2)
dis <- dist(iris[,-5])^2
sil = silhouette (cl$cluster, dis)
plot(sil, main = "Clustering Data with Silhoutte plot using 2 Clusters", col = c("cyan",
"blue"))
library(cluster)
cl <- kmeans(iris[,-5], 8)
dis <- dist(iris[,-5])^2
sil = silhouette (cl$cluster, dis)

plot(sil, main = "Clustering Data with Silhoutte plot using 8 Clusters", col = c("cyan",
"blue", "orange", "yellow", "red", "gray", "green", "maroon"))
install.packages("factoextra")
library(factoextra)
install.packages("clustertend")
library(clustertend)
genx<-function(x){
  runif(length(x), min(x), (max(x)))
}
random_df <- apply(iris[,-5], 2, genx)
random_df <- as.data.frame(random_df)

iris[,-5] <- scale(iris[,-5])
random_df <- scale(random_df)

res <- get_clust_tendency(iris[,-5],
                          n = nrow(iris) -1 ,
                          graph = FALSE)
res$hopkins_stat

hopkins(iris[,-5], n = nrow(iris) -1)
res <- get_clust_tendency(random_df, n = nrow(random_df)-1,
                          graph = FALSE)

res$hopkins_stat
```