Hierarchial Clustering on Water treatment data

Sumit Gupta

December 22, 2017

Loading libraries:

```r
library(data.table)
library(ggplot2)
library(fpc)
```

```r
# Loading the dataset
water_data <- read.table("water-treatment.data.txt", header = F, sep=",", na.strings = c("?")) # insert
setDT(water_data)
# str(water_data)
```

```r
# Checking missing values
colSums(is.na(water_data))
```

```
##  V1  V2  V3  V4  V5  V6  V7  V8  V9 V10 V11 V12 V13 V14 V15 V16 V17 V18
##   0  18   3   0  23   6   1  11  25   0   0  40   0  11  24   0   0  28
## V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31 V32 V33 V34 V35 V36
##   9   2  13  25   0   1  23  18   5  17  28   1  62   4  27  40  26  36
## V37 V38 V39
##  25   8  31
```

```r
#impute missing values with median

for(i in colnames(water_data)[!(colnames(water_data) %in% c("V1"))])
set(x = water_data,i = which(is.na(water_data[[i]])), j = i, value = median(water_data[[i]], na.rm = T))
```

```
## Warning in set(x = water_data, i = which(is.na(water_data[[i]])), j = i, :
## Coerced 'double' RHS to 'integer' to match the column's type; may have
## truncated precision. Either change the target column to 'double' first
## (by creating a new 'double' vector length 527 (nrows of entire table) and
## assign that; i.e. 'replace' column), or coerce RHS to 'integer' (e.g. 1L,
## NA_[real|integer]_, as.*, etc) to make your intent clear and for speed. Or,
## set the column type correctly up front when you create the table and stick
## to it, please.
```

```r
# Scaling the variables except V1
scaled_wd <- scale( water_data[,-c("V1"),with=F])
```

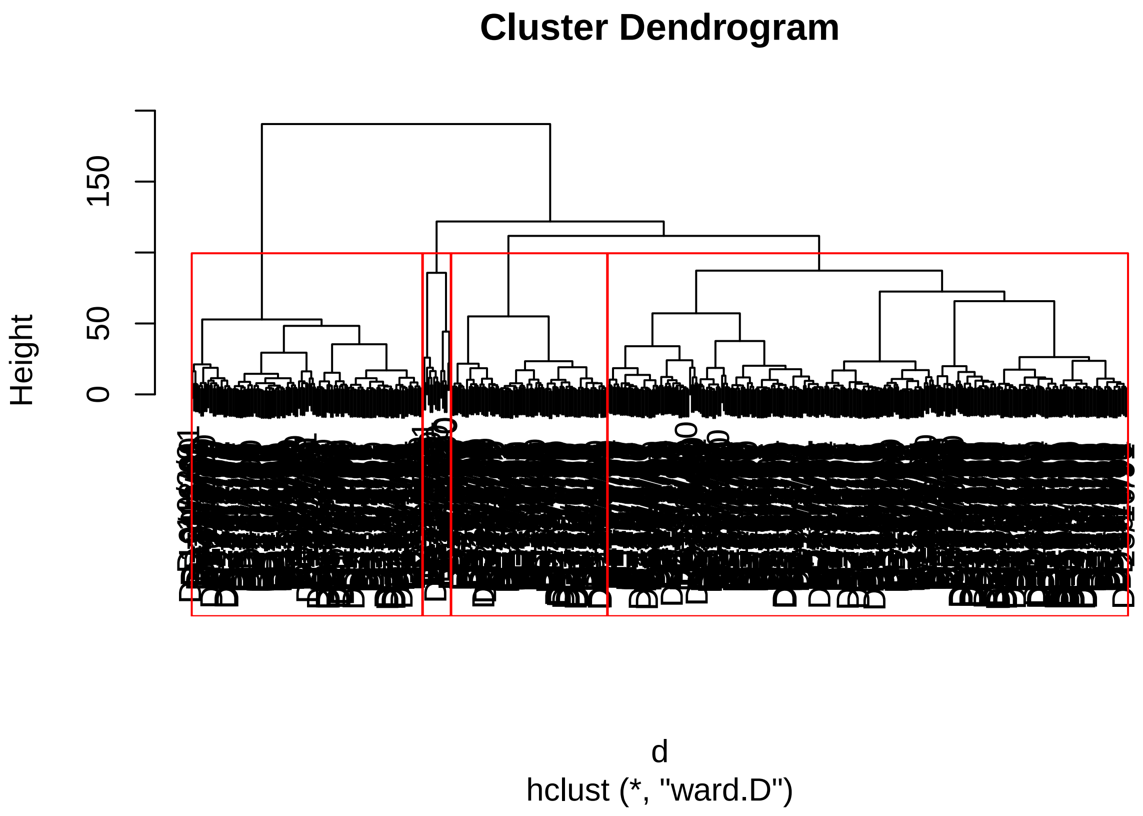
Now, our data is ready for clustering! For hierarchical clustering, we'll first calculate a distance matrix based on Euclidean measure. Then using the hclust function, we can implement hierarchical clustering.

```r
# distance matrix
d <- dist(scaled_wd, method = "euclidean")

# Hierarchical Clustering
h_clust <- hclust(d, method = "ward") #clustering
```

```
## The "ward" method has been renamed to "ward.D"; note new "ward.D2"
```

```r
# Plotting the dendrogram
plot(h_clust, labels = water_data$V1)
```

```r
# All the leaves at the bottom carry one observation each, which are then merged into similar values as
# Now, how can you estimate the number of clusters?
# Going by the logic of horizontal cut, 4 clusters are evident.
rect.hclust(h_clust,k=4)
```



Lets see which observation is into which cluster:

```r
groups <- cutree(h_clust,k=4)
groups
```

```
##   [1] 1 2 2 2 2 1 1 1 2 1 3 3 3 2 2 2 2 1 2 1 2 1 2 2 2 2 2 2 2 1 1 2 1 1 2
##  [36] 1 2 1 1 1 2 1 1 1 1 2 2 1 2 4 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 1 1 2
##  [71] 2 2 1 1 2 2 4 4 3 2 2 1 2 3 4 2 4 2 4 2 2 2 2 2 2 2 3 4 2 2 2 1 2 2 2
## [106] 2 2 1 2 2 2 1 1 2 1 2 4 2 4 4 2 2 2 4 2 4 2 4 4 4 2 2 4 2 2 2 2 2 2 2
## [141] 2 4 4 4 2 2 2 2 2 2 2 4 2 2 2 2 2 1 4 2 2 2 2 4 2 1 1 2 2 2 3 2 4 2 2
## [176] 2 4 2 2 2 2 2 4 4 4 2 2 3 2 4 2 2 2 2 2 4 4 2 2 2 2 2 2 4 4 4 4 4 2 2
## [211] 4 4 4 2 2 2 2 2 2 2 2 4 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1 2 2
## [246] 1 2 2 1 4 1 4 1 2 2 2 4 4 2 2 2 1 2 2 2 2 2 1 2 1 2 2 1 1 1 2 4 1 2 2
## [281] 4 2 4 2 4 4 4 2 4 2 2 2 3 4 4 2 2 4 2 2 1 1 2 1 2 2 2 2 1 2 1 1 1 2 1
## [316] 1 1 1 1 2 4 2 2 2 2 1 2 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 2 2 1 1 1 1 2 2
## [351] 1 1 2 1 2 1 2 2 2 2 1 1 1 1 2 1 1 1 1 2 1 1 1 1 1 2 4 2 2 1 1 4 4 2 4
## [386] 4 2 1 2 2 2 4 2 2 2 3 4 2 3 2 2 3 2 1 1 1 2 2 2 1 1 1 2 4 2 1 2 2 2 2
## [421] 2 2 2 2 2 4 3 2 4 2 2 2 2 1 2 2 2 1 4 2 2 3 3 3 2 2 1 2 2 2 2 2 2 2 2
## [456] 2 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 4 4 2 4 2
## [491] 2 2 4 2 2 2 2 2 2 2 2 1 2 2 2 4 4 4 4 4 4 4 4 4 2 4 4 4 4 4 4 2 4 4 2
## [526] 2 2
```

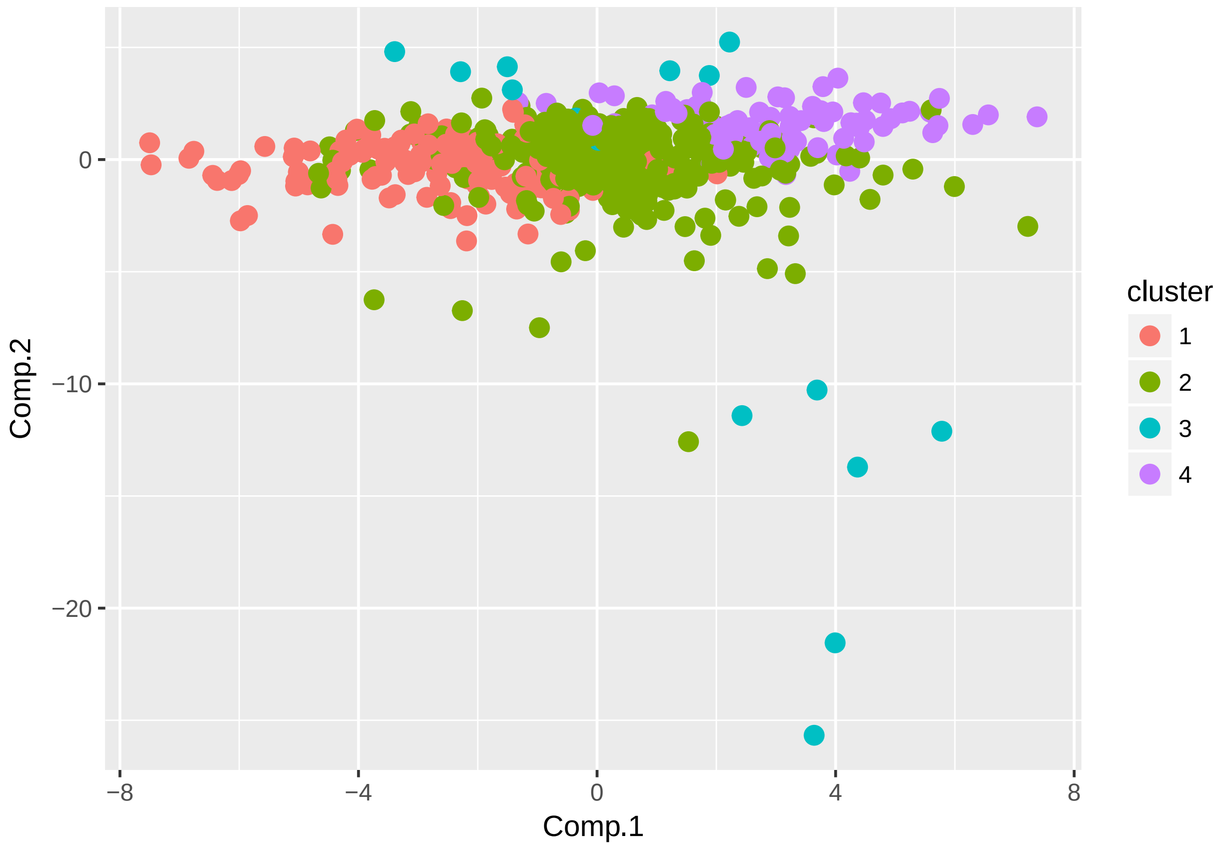
Let's visualize the clusters in a better way. However, visualizing this high dimensional data in one plot has its own challenges. A smart way could be, to visualize the cluster on principal components of this data. This way we would be able to capture most of information by reducing the data dimension.

To implement PCA, we'll use princomp base function. For our convenience, we'll take only the first two components.

```
# pca
pcmp <- princomp(scaled_wd)
pred_pc <- predict(pcmp, newdata = scaled_wd)[,1:2]
```

Now, we'll create a data frame having pc values and their corresponding clusters. Then, using ggplot2 we'll create the plot.

```
comp_dt <- cbind(as.data.table(pred_pc),cluster = as.factor(groups), Labels = water_data$V1) # Combinin

ggplot(comp_dt,aes(Comp.1,Comp.2))+ geom_point(aes(color = cluster),size=3)
```



Let's now proceed to k means clustering. We'll use the base function k means with 100 iterations to converge:

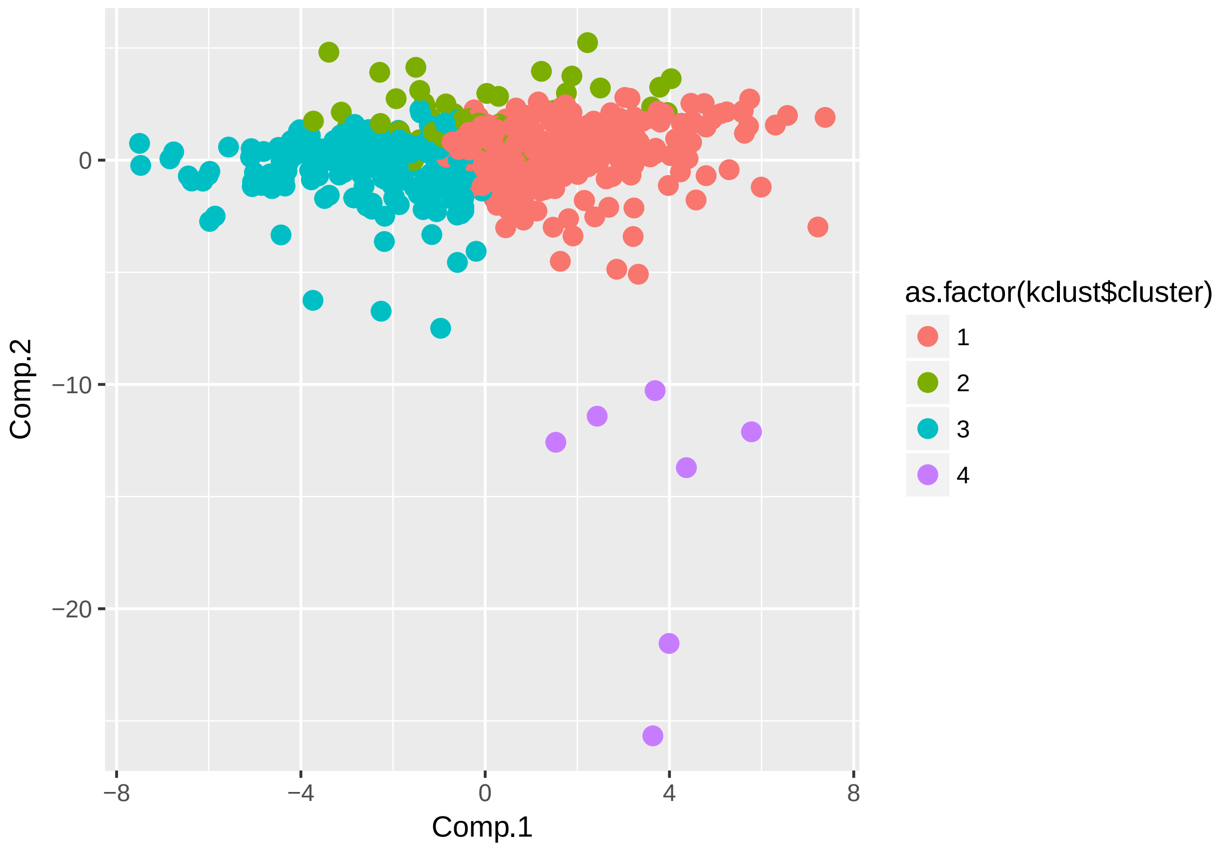
```
#kmeans

kclust <- kmeans(scaled_wd,centers = 4,iter.max = 100)
```

You can check out the number of observations comprised by each cluster using kclust$size. Again, let's create a plot to understand this clustering better. The parameter.

```
kclust$size
```

```
## [1] 265  57 198   7
```

```
# Lets visualize
ggplot(comp_dt,aes(Comp.1,Comp.2))+ geom_point(aes(color = as.factor(kclust$cluster)),size=3)
```



As seen above, both the techniques have partitioned the observations in same clusters. Is 4 the optimal number of clusters in k means ? Let's find out. To pick the best value of k, we'll use kmeansruns function from fpc package. This function is enabled with two distance metrics: Average silhouette width and Calinski-Harabasz.

Let's try to use both the methods and check out the best k value:

```
tunek <- kmeansruns(scaled_wd,krange = 1:10,criterion = "ch")
tunek$bestk #3
```

```
## [1] 3
```

```
tunekw <- kmeansruns(scaled_wd,krange = 1:10,criterion = "asw")
tunekw$bestk #4
```

```
## [1] 4
```