Clustering - Kmeans

Método Particional e Realocação

Ao contrário dos tradicionais *métodos hierárquicos*, em que os grupos não são revisitados depois de terem sidos construídos, os algortimos de realocação melhoraram gradualmente os grupos, resultando em clusters de melhor qualidade.

Basicamente, os algoritmos de particionamento e realocação, dividem os dados em vários subgrupos. Entretanto, a verificação de todos os possíveis subgrupos é computacionalmente inviável. Deste modo, foram desenvolvidas diferentes Heurísticas que executam a realocação de forma iterativa, re-atribuindo as observações entre os grupos.

Algoritmos K-means

Entre os algoritmos de agrupamento do tipo partição, o algoritmo *k-means* é destacadamente o mais importante. O algoritmo *k-means* é amplamente utilizado nas mais diversas aplicações. Na área de mineração de dados, por exemplo, entre inúmeros algoritmos, foi classificado como o segundo mais importante em data mining. O algoritmo *k-means* trata do problema de agrupamento segundo o critério de mínima soma de quadrados.

O algoritmo k-means é um método iterativo simples para particionar um conjunto de dados em um número de grupos especificado pelo usuário. Basicamente o algoritmo, em suas diversas formas, possui a seguinte estrutura de passos.

1. Seleção de **k** pontos para sementes iniciais dos centróides dos k grupos.
2. Cada observação é associada a um cluster, para o qual a dissimilaridade entre o objeto e o centro do cluster é menor que as demais.
3. Os centros dos clusters são recalculados, redefinindo cada um em função da média de todas as observações atríbuídas ao grupo.
4. retorna ao passo 2 até que os centros dos clusters se estabilizem.

Distância euclidiana

```
euclidean.distance<-function(x, y){
    resp<-sum((x-y)^2)
    resp
}
```

K mais próximo.

```
closest.k<-function(data, centroids){
    euclidean.dist<-c()
    for(i in 1:nrow(centroids)){
        euclidean.dist[i]<-euclidean.distance(data, centroids[i,])
    }

    which.min(euclidean.dist)
}
```

Funçao para execução do kmeans

```
k.means<-function(dataset, k=2){
  # cria uma lista para receber a resposta
    resp<-list()
    cat("Inicializando centroides...\n");
    index<-sample(1:nrow(dataset), k)
    # aplica os centrois no datset
    centroids<-dataset[index, ]

    #
    clustering<-rep(0,nrow(dataset))
    stop.crit<-matrix(0, nrow=nrow(centroids), ncol=ncol(centroids))

    while(euclidean.distance(stop.crit, centroids) > 0){

        #organiza os objetos em clusters iniciais
        cat("Centroides escolhidos:\n")
        for(i in 1:k){
            cat("[",i,"] = ", as.double(centroids[i,]), "\n")
        }

        for(i in 1:nrow(dataset)){
            clustering[i]<-closest.k(dataset[i,], centroids)
        }
        stop.crit<-centroids

        resp$data<-dataset
        resp$centroids<-centroids
        resp$clustering<-clustering

        #atualiza centroide...
        for(i in 1:nrow(centroids)){
            centroids[i,]<-colMeans(dataset[which(clustering == i),])
        }
    }
    resp
}
```

Plotar Kmeans

```
plot.kmeans<-function(resp){
    plot(resp$data)
    for(i in sort(unique(resp$clustering))){
        points(resp$data[which(resp$clustering==i),], col=(i+1), pch=19)
        points(resp$centroids[i,], pch=8, col=(i+1))
    }
}
```

Após declarar as funções, importamos a base para realização do agrupamento.

```
# read the dataset
iris <- read.csv('../data/Iris.csv')

# pegando somente as features do meu conjunto de dados, variável contínua
iris.features <- iris[,2:5]
```

```r
# obtem os centroides do meu cluster com o agrupamento kmeans
cluster.kmeans <- k.means(iris.features, 3)
```

```
## Inicializando centroides...
## Centroides escolhidos:
## [ 1 ] =  4.9 3.1 1.5 0.1
## [ 2 ] =  4.8 3.4 1.6 0.2
## [ 3 ] =  6.2 2.2 4.5 1.5
## Centroides escolhidos:
## [ 1 ] =  4.826316 3.042105 1.478947 0.2315789
## [ 2 ] =  5.115625 3.6125 1.503125 0.278125
## [ 3 ] =  6.273737 2.875758 4.925253 1.681818
## Centroides escolhidos:
## [ 1 ] =  4.752 3.016 1.648 0.292
## [ 2 ] =  5.232143 3.667857 1.485714 0.2857143
## [ 3 ] =  6.301031 2.886598 4.958763 1.695876
## Centroides escolhidos:
## [ 1 ] =  4.752 2.964 1.732 0.324
## [ 2 ] =  5.224138 3.655172 1.482759 0.2827586
## [ 3 ] =  6.314583 2.895833 4.973958 1.703125
## Centroides escolhidos:
## [ 1 ] =  4.741667 2.954167 1.754167 0.3291667
## [ 2 ] =  5.216667 3.64 1.473333 0.28
## [ 3 ] =  6.314583 2.895833 4.973958 1.703125
```

```r
# cluster.means -> $data - $centroids - $clustering
print('classificação de cada atributo')
```

```
## [1] "classificação de cada atributo"
```

```r
cluster.kmeans$clustering
```

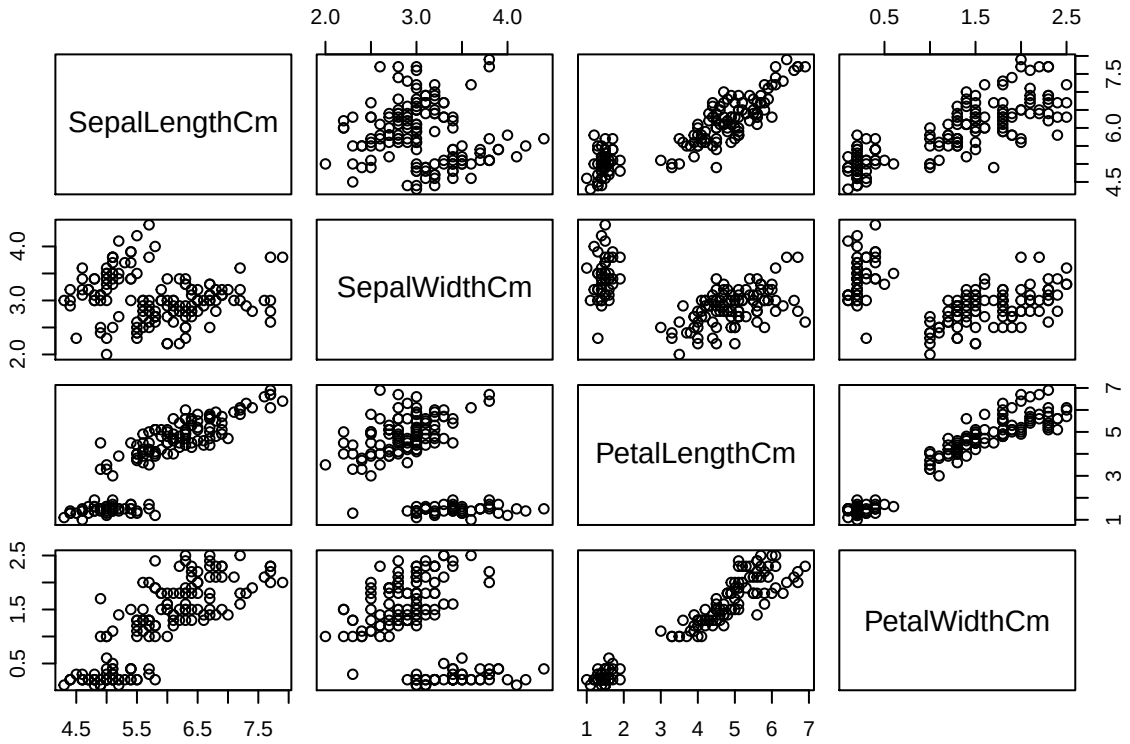
```
##   [1] 2 1 1 1 2 2 1 2 1 1 2 1 1 1 2 2 2 2 2 2 2 2 2 1 1 2 2 2 1 1 2 2 2 1
##  [36] 2 2 1 1 2 2 1 1 2 2 1 2 1 2 2 3 3 3 3 3 3 3 1 3 3 1 3 3 3 3 3 3 3 3
##  [71] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 1 3 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3
```

```r
# centroides dos minhas 3 classes de atributos
cluster.kmeans$centroids
```

```
##    SepalLengthCm SepalWidthCm PetalLengthCm PetalWidthCm
## 38      4.741667     2.954167      1.754167    0.3291667
## 12      5.216667     3.640000      1.473333    0.2800000
## 69      6.314583     2.895833      4.973958    1.7031250
```

visulizando os primeiros resultados

```r
# visualizando
plot.kmeans(cluster.kmeans)
```



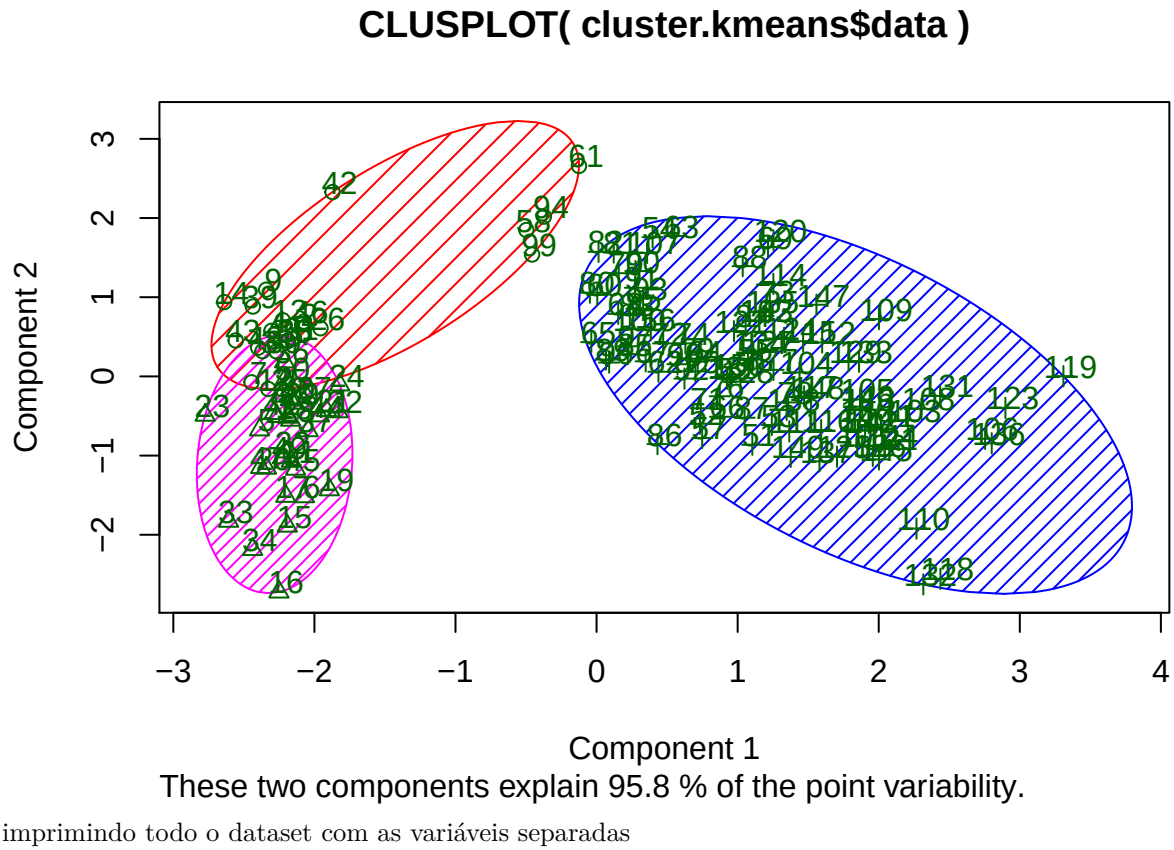
kmeans Interativo

```
k.means.interativo<-function(dataset, k=2, sleep.time=1){
    resp<-list()
    cat("Inicializando centroides...\n");
    Sys.sleep(sleep.time)
    index<-sample(1:nrow(dataset), k)
    centroids<-dataset[index, ]

    clustering<-rep(0,nrow(dataset))
    stop.crit<-matrix(0, nrow=nrow(centroids), ncol=ncol(centroids))

    while(euclidean.distance(stop.crit, centroids) > 0){

        #organiza os objetos em clusters iniciais
        cat("Centroides escolhidos:\n")
        for(i in 1:k){
            cat("[",i,"] = ", as.double(centroids[i,]), "\n")
        }

        for(i in 1:nrow(dataset)){
            clustering[i]<-closest.k(dataset[i,], centroids)
        }
        stop.crit<-centroids

        resp$data<-dataset
        resp$centroids<-centroids
        resp$clustering<-clustering
```

```
        plot.kmeans(resp)
        Sys.sleep(sleep.time)

        #atualiza centroide...
        for(i in 1:nrow(centroids)){
            centroids[i,]<-colMeans(dataset[which(clustering == i),])
        }
    }
    resp
}
```

visualizado com o clusplot com todos as 4 variáveis do grupo

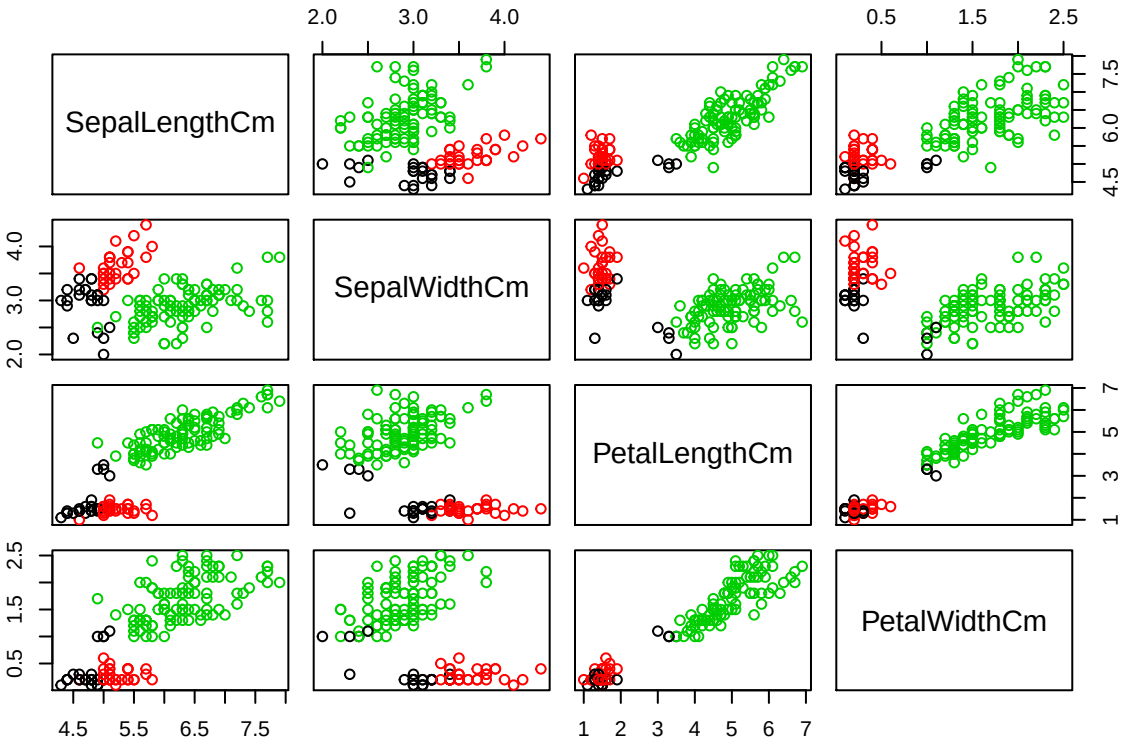
```
library("cluster")
clusplot(cluster.kmeans$data, cluster.kmeans$clustering, color = TRUE, shade= TRUE, labels = 3, lines =
```



###

imprimindo todo o dataset com as variáveis separadas

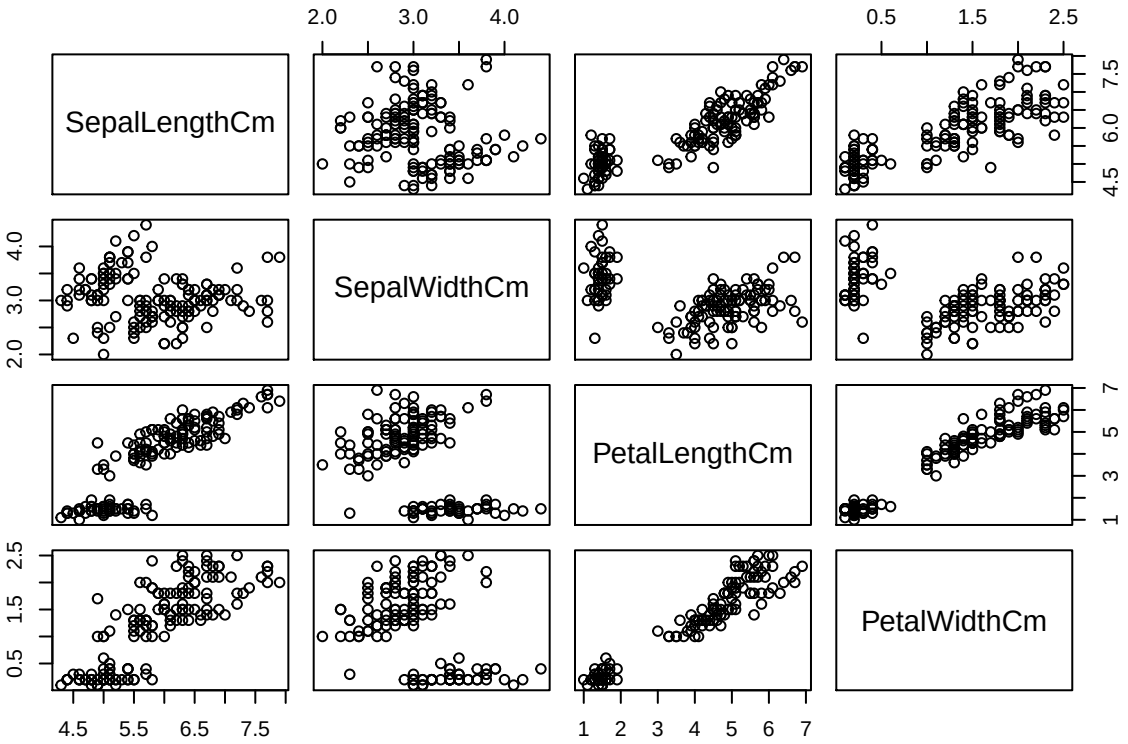
```
with(iris, pairs(cluster.kmeans$data, col=c(1:3)[cluster.kmeans$clustering]))
```



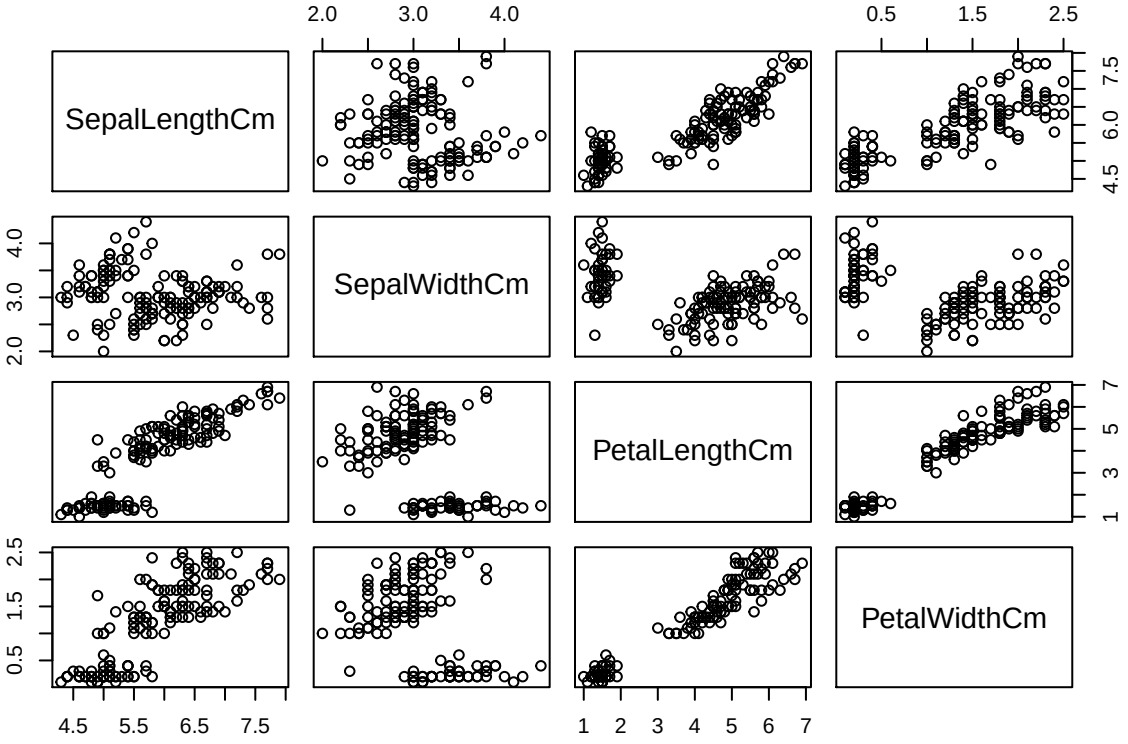
Kmeans interativo

```
#
c.k.means <- k.means.interativo(iris.features, 3)
```

```
## Inicializando centroides...
## Centroides escolhidos:
## [ 1 ] =  4.7 3.2 1.6 0.2
## [ 2 ] =  6.1 3 4.6 1.4
## [ 3 ] =  5.7 2.9 4.2 1.3
```

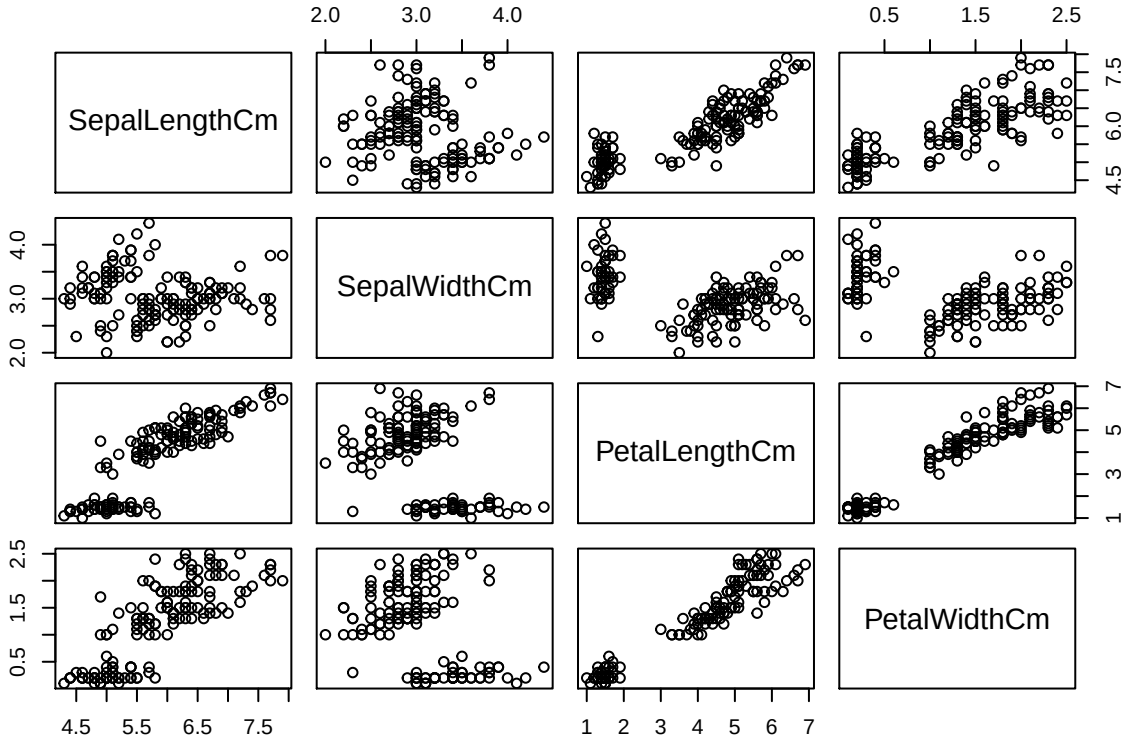



```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.545833 2.963889 5.273611 1.85
## [ 3 ] =  5.532143 2.635714 3.960714 1.228571
```

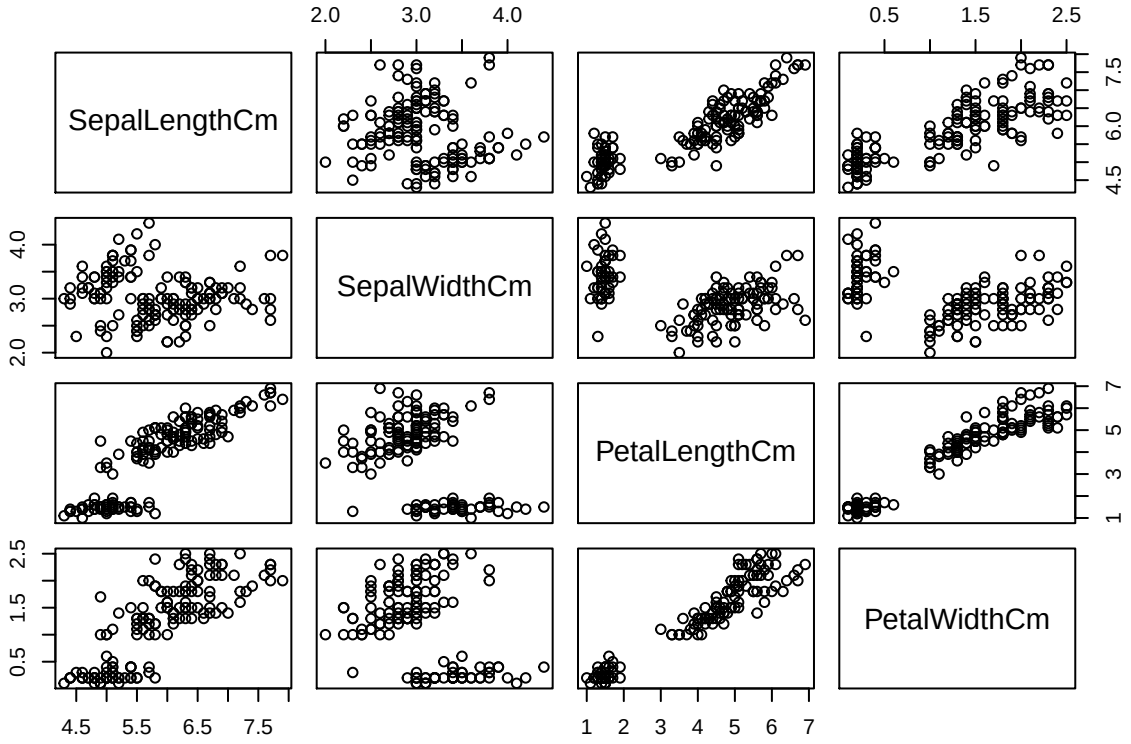



```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
```

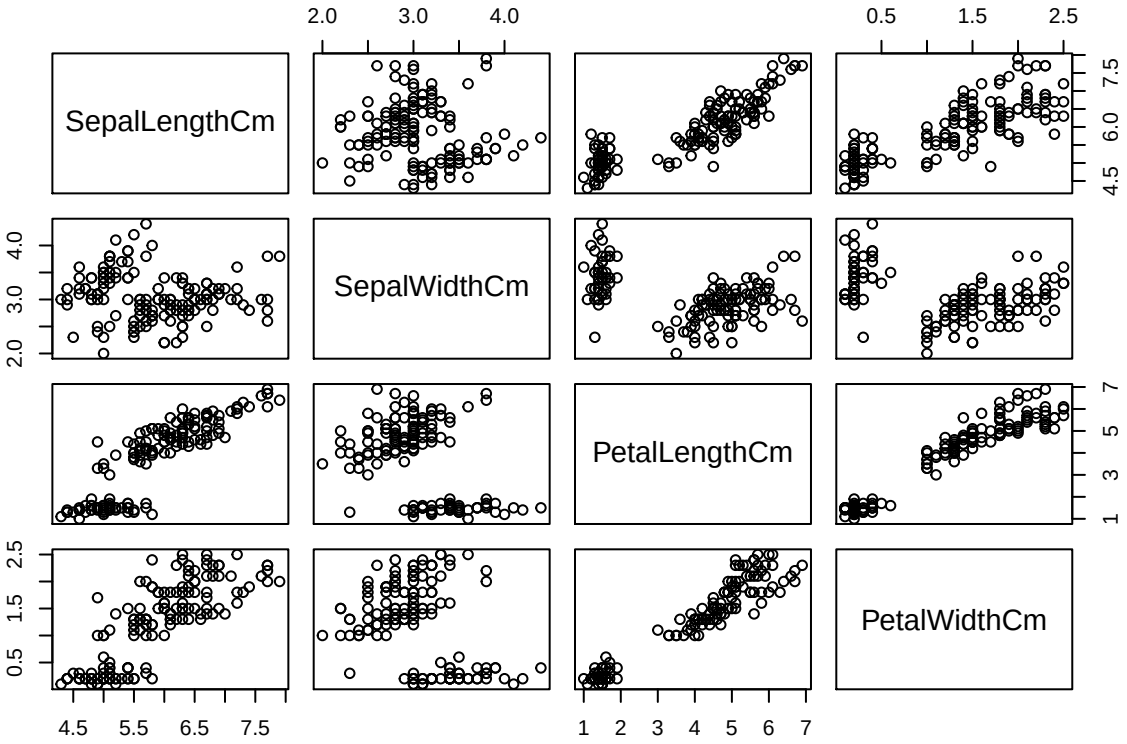
```
## [ 2 ] =  6.577273 2.990909 5.348485 1.895455
## [ 3 ] =  5.65 2.641176 4.047059 1.25
```



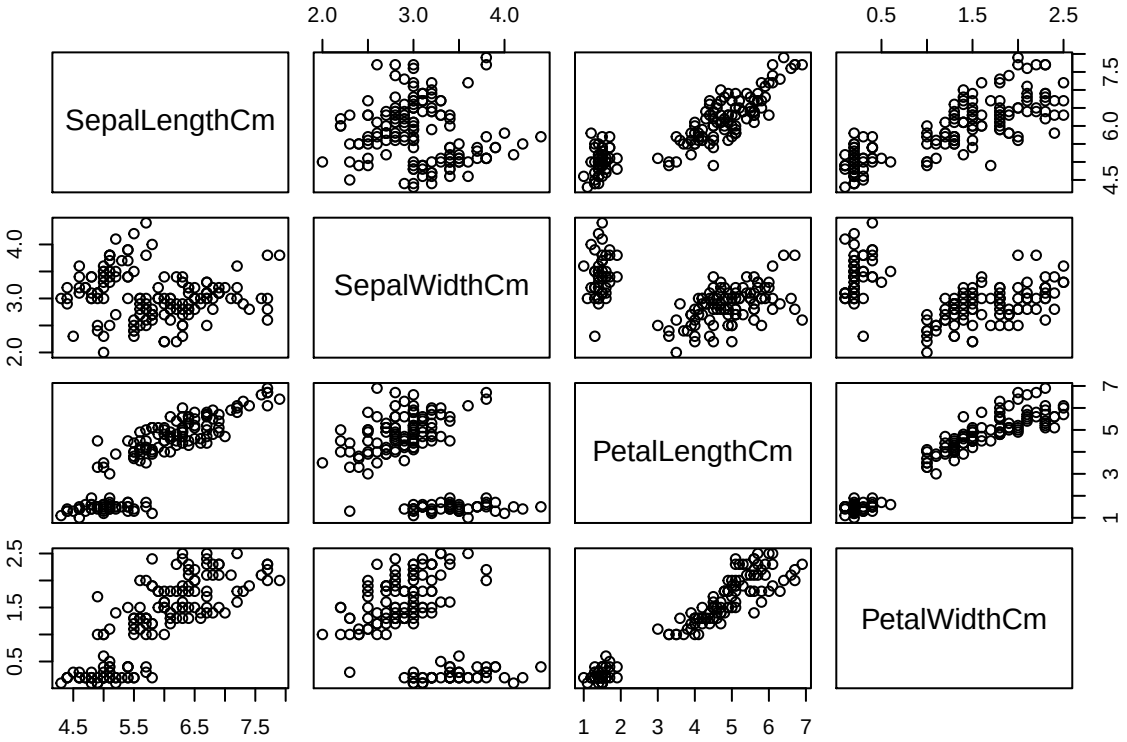

```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.601587 2.985714 5.384127 1.915873
## [ 3 ] =  5.683784 2.678378 4.091892 1.267568
```



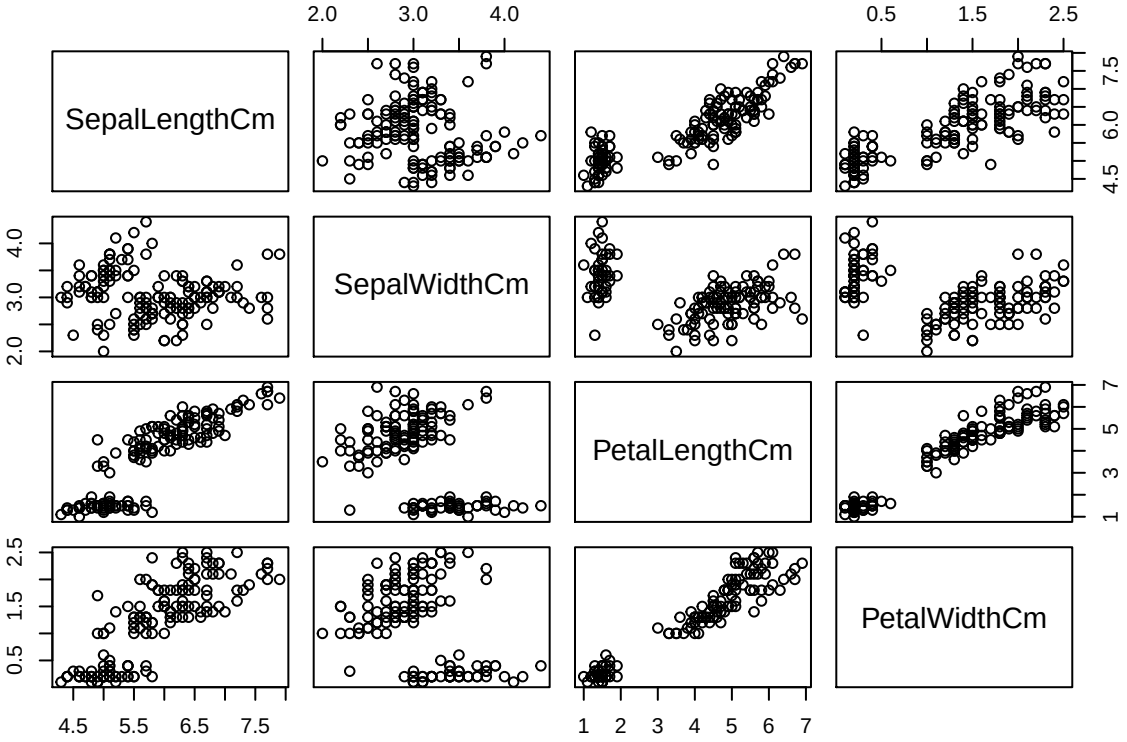
```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.632203 2.998305 5.430508 1.937288
## [ 3 ] =  5.729268 2.690244 4.15122 1.3
```




```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.664815 3.007407 5.5 1.968519
## [ 3 ] =  5.78913 2.713043 4.208696 1.332609
```

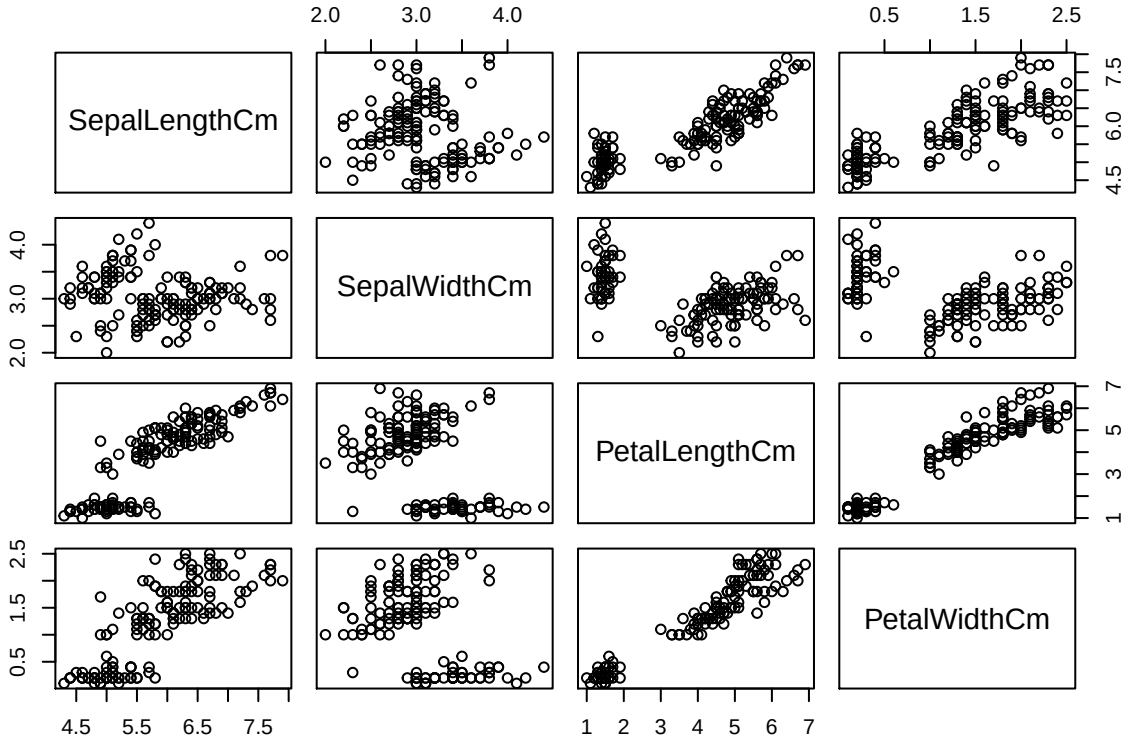



```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.702 3.016 5.556 1.992
## [ 3 ] =  5.822 2.728 4.256 1.36
```

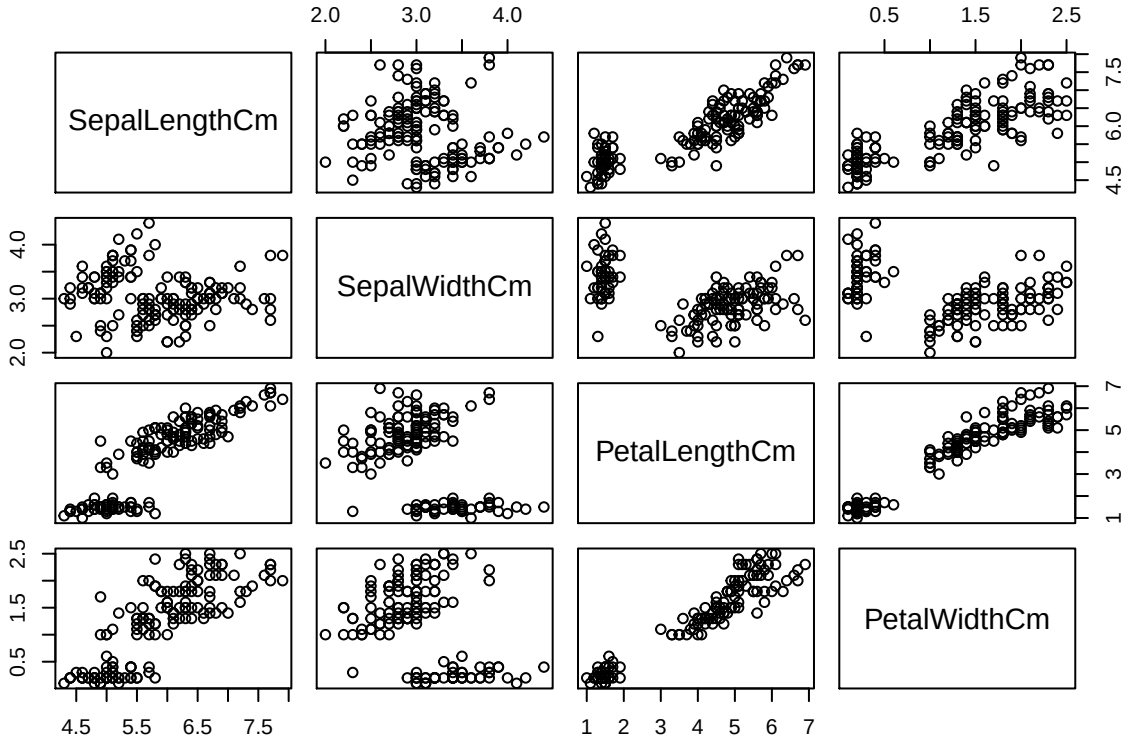



```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
```

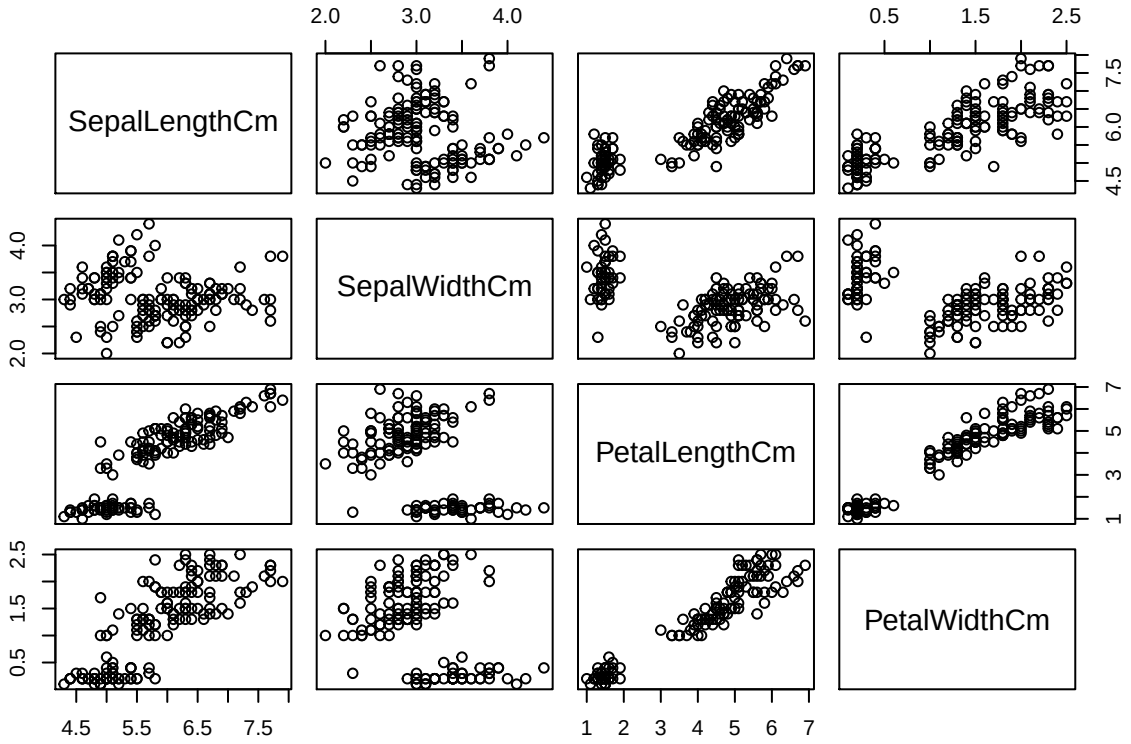
```
## [ 2 ] =  6.769565 3.036957 5.6 2.008696
## [ 3 ] =  5.82963 2.731481 4.314815 1.392593
```



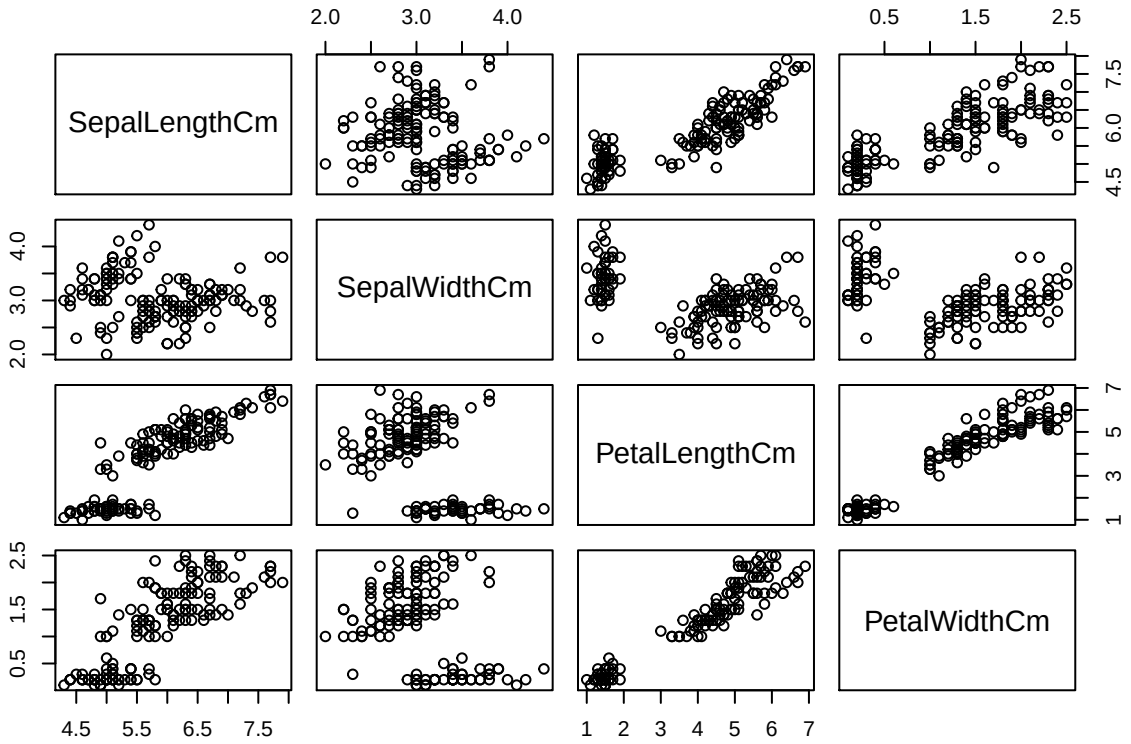

```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.802326 3.044186 5.648837 2.030233
## [ 3 ] =  5.854386 2.742105 4.345614 1.408772
```



```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.8275 3.07 5.7 2.0625
## [ 3 ] =  5.885 2.74 4.376667 1.418333
```




```
## Centroides escolhidos:
## [ 1 ] =  5.006 3.418 1.464 0.244
## [ 2 ] =  6.853846 3.076923 5.715385 2.053846
## [ 3 ] =  5.883607 2.740984 4.388525 1.434426
```


SepalLengthCm
SepalWidthCm
PetalLengthCm
PetalWidthCm
2.0
3.0
4.0
0.5
1.5
2.5
4.5
6.0
7.5
1
3
5
7
4.5
5.5
6.5
7.5
1
2
3
4
5
6
7