Segunda entrega de projeto

Karen Kaori Yonea - 10349471

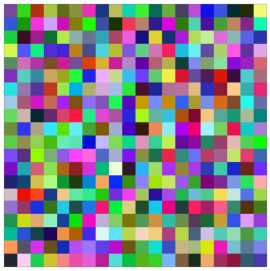
1 Projeto4

1.1 Implementação

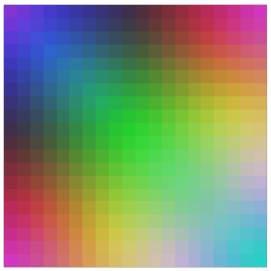
```
SOM <- function(train, n) {
  train <<- train
  n <<- n
  n_train <<- dim(train)[1]
  grid <- meshgrid(1:n, 1:n)
  c <- matrix(c(as.vector(t(grid$X)), as.vector(t
      (grid$Y))), ncol = 2)

  #Definir a saída
  res <- list()

  #Configuração inicial: pesos aleatórios
  nodes <<- rand(n*n, dim(train)[2])
  res$init = nodes

  #Pesos atualizados
  res$trained <- training(c)

  #Retorna a matriz de pesos inicial e final
  return(res)
}

training <- function(c) {
  for (i in 1:dim(train)[1]) {
    #Encontrar o neuronio "vencedor"
    match = matching(train[i,])
    #Distâncias
    d <<- rowNorms(c - c[match], method='
        euclidean')
    L = learning_rate(i)
    S = learning_radius(i)
    aux = train[i,] - nodes
    #Atualizar pesos
    nodes = nodes + L*(S * aux)
  }
  return(nodes)
}

matching <- function(teachers) {
  norms = rowNorms(nodes - teachers, method='
      euclidean')
  match = which.min(norms)
  return(match)
}

neighbourhood <- function(t) {
  hfl = n_train/4
  ini = n/2
  nei = ini*exp(-t/hfl)
  return(nei)
}

learning_rate <- function(t) {
  hfl = n_train/4
  ini = 0.1
  rat = ini*exp(-t/hfl)
  return(rat)
}

learning_radius <- function(t) {
  s = neighbourhood(t)
  rad = exp(-d^2/(2*s^2))
  return(rad)
}
```

1.2 Paleta de cores

Para testar o algoritmo foi utilizado dados gerados de forma aleatório em escala de cor RGB:



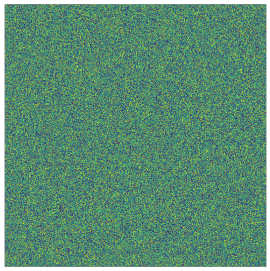
(a) Antes de aplicar o SOM



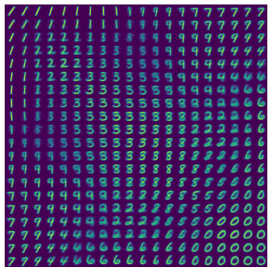
(b) Depois de aplicar o SOM

1.3 Base MNIST(caracteres manuscritos)

Utilizando a base de dados de imagens de teste do MNIST com 10 categorias obteve-se:



(a) Antes de aplicar o SOM



(b) Depois de aplicar o SOM

2 Projeto 5

2.1 Implementação

2.1.1 KNN

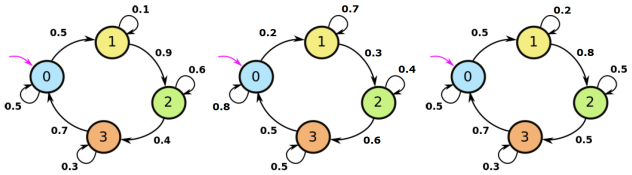
```
#Analise da base de treino
guess.knn <- function(x, train, trainlabels, k) {
  #Distância euclidiana
  xmatrix <- matrix(as.numeric(x), nrow=nrow(
      train), ncol=length(x), byrow=T)
  xmatrix <- (abs(as.matrix(train)-xmatrix))^2
  diffs <- (rowSums(xmatrix))^(1/2)
  #Ordena as distâncias
  diffs <- data.frame(dist=diffs,label=
      trainlabels)
  diffs <- (diffs[order(diffs$dist),])
  diffs <- diffs[1:k,]

  guess <- names(sort(-table(diffs$label)))[1]

  return(guess)
}

knn <- function(train, test, trainlabels, k) {
  #Formatando a base de treino
  subsample <- sample(1:nrow(train), nrow(train),
       replace=F)
  train <- train[subsample,]
  trainlabels <- trainlabels[subsample]

  #Atribuir os valores das classes da base teste
  kn <- apply(test, 1, function(x) guess.knn(x,
      train, trainlabels, k))

  return(kn)
}
```

2.1.2 K-means

```
k_means <- function(dataset, k, threshold=1e-3) {
  #Definindo os centroides iniciais
  ids = sample(1:nrow(dataset), size=k)
  centroid = as.data.frame(dataset[ids,])
  div = 2 * threshold

  #Divergente > que o limiar
  while (div > threshold) {
    # Para cada centroide
    dists = NULL
    for (i in 1:k) {
      # calcular a distancia euclidiana de cada
          ponto do
      # espaço em relação a cada centroide
      euclidean = apply(dataset, 1, function(row)
          {sqrt(sum((row - centroid[i,])^2))})
      dists = cbind(dists, euclidean)
    }

    # Descobrir qual centroide cada ponto está
        associado
    clusters = apply(dists, 1, function(row) {
        which.min(row) } )

    # Atualizar a posição dos centroides
    div = 0
    for (i in 1:k) {
      ids = which(clusters == i)
      position = colMeans(as.data.frame(dataset[
          ids,]))
      div = div + sqrt(sum((centroid[i,] -
          position)^2))
      centroid[i,] = position
    }

    div = div / k
  }

  km = list()
  km$k = k
  km$centroid = centroid
  km$cluster = clusters

  return(km)
}
```

2.2 Sinal 1D - Autômatos probabilísticos

Para essa parte foi gerado 500 padrões de tamanho $M = 500$ para cada autômato, tendo um total de 1500 dados.



(a) (b) (c)

Figura 3: Autômatos utilizados

A escolha de dois autômatos muito similares para testar os métodos foi proposital.
Atributos escolhidos:

- Média do tamanho dos bursts
- Entropia do tamanho dos bursts
- Média das distâncias intersímbolos
- Entropia das distâncias intersímbolos

2.2.1 KNN

Foram selecionados aleatoriamente 250 índices para treino e outros 250 para teste.

Matriz de Confusão

	a	b	c
a	82	0	13
b	0	82	0
c	12	0	61

2.2.2 K-means

Utilizando 250 índices foram obtidos resultados pouco assertivos, então foi tomada a escolha de analisar 500 valores.

Matriz de Confusão

	a	b	c
a	134	0	26
b	1	164	0
c	11	0	164

Os melhores resultados foram obtidos utilizando entre $(500, 750)$ índices.

2.3 Base MNIST(caracteres manuscritos)

2.3.1 KNN

Para essa parte foram selecionados aleatoriamente 250 índices da base de treino e 250 da base de teste.

Matriz de Confusão

	0	1	2	3	4	5	6	7	8	9
0	20	0	0	0	0	0	0	0	0	0
1	0	32	0	0	0	0	0	0	2	0
2	2	8	12	0	2	0	0	1	1	0
3	0	3	0	15	0	1	0	0	0	3
4	0	4	0	0	21	0	1	0	0	0
5	3	1	0	5	0	9	0	1	1	1
6	0	0	0	0	2	0	20	0	0	0
7	0	3	0	0	2	0	0	19	0	7
8	1	3	0	7	1	0	0	1	5	3
9	0	2	0	0	6	0	0	0	1	21

2.3.2 K-means

Para essa parte foram selecionados aleatoriamente 250 índices da base de teste e os atributos foram pré-processados com a função nearZeroVar que diagnostica preditores que possuem um valor único (ou seja, preditores de variação zero) ou preditores que possuem as duas características a seguir: eles têm muito poucos valores exclusivos em relação ao número de amostras e a razão da frequência do valor mais comum para a frequência do segundo valor mais comum é grande.

Matriz de Confusão

	0	1	2	3	4	5	6	7	8	9
0	26	0	0	0	0	1	0	1	0	0
1	0	27	0	0	0	0	0	0	2	0
2	0	7	12	0	6	0	0	0	1	0
3	0	5	0	9	1	2	0	1	2	1
4	0	0	0	0	12	1	0	1	5	11
5	0	1	0	8	1	4	0	0	5	2
6	0	0	0	0	8	1	12	0	1	0
7	1	1	0	0	0	0	0	16	0	5
8	0	5	0	2	0	6	0	1	9	1
9	0	0	0	0	4	2	0	2	0	18

Alguns grupos ficam bem definidos como o zero enquanto outros ficam indistinguíveis

2.4 Base IRIS

2.4.1 KNN

Foram selecionados aleatoriamente 75 índices da base de dados como treino e os outros 75 índices como teste.

Matriz de Confusão

	setosa	versicolor	virginica
setosa	25	0	0
versicolor	0	25	0
virginica	0	5	20

2.4.2 K-means

Utilizou-se todas as entradas da base de dados e não foi feito nenhum tipo de pré-processamento.

Matriz de Confusão

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	14	36

3 Projeto 6

3.1 Implementação

PCA:

```
pca <- function(p) {
  k = cov(p)
  Q = eigen(k)
  p = scale(p)
  for (i in 1:dim(p)[1]) {
    x = t(p[i,]) %*% Q$vectors
    p[i,] = x
  }
  return(p)
}
```

Método descrito por Newman:

```
spectral_clustering <- function(dt, k, sig) {
  #Definindo variáveis
  S <- dt
  n <- nrow(dt)
  W <- matrix(rep(0,n^2) ,nrow = n ,ncol=n)
  D <- diag(n)

  #Calculando matriz W
  for (i in 1:n){
    for(j in 1:n){
      if (i != j){
        W[i,j] <- exp( - sqrt(sum((S[i,]-S[j,])
            ^2)) / 2*sig)
      }
    }
  }

  #Calculando matriz diagonal
  for (i in 1:n){
    D[i,i] <- sum (W[i,])
  }

  #Definindo L
  aux = solve(sqrt(D))
  L <- aux %*% W %*% aux

  #Autovetores de L (funcao eigen ja ordena)
  eig <- eigen(L)$vectors
  X <- eig[,(1 : k)]

  #Definindo matriz de comunidade
  Y <- matrix(0,nrow=n,ncol=k)
  com <- matrix(0,nrow=n,ncol=1)

  #Encontra as comunidades
  for(i in 1:n){
    for(j in 1:k){
      Y[i,j] <- X[i,j] / (sqrt (sum(X[i,j])^2))
    }
  }

  return(Y)
}
```

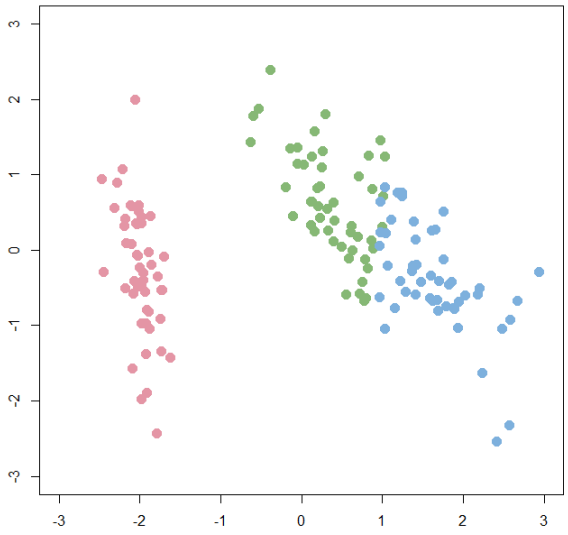


Figura 4: PCA com a separação das comunidades

3.2 Resultados

Matriz de Confusão Método de Newman

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	44	6
virginica	0	6	44

Matriz de Confusão K-means

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	2
virginica	0	14	36

No geral ambos os métodos diferenciaram os grupos de forma satisfatória, porém, o custo computacional da implementação do k-means para essa precisão é muito maior, sendo assim, compensa nesse caso utilizar o método descrito por Newman.

4 Projeto 7

No projeto foram utilizadas as seguintes bibliotecas disponibilizadas do respositório CRAN:

- cluster: algoritmos de clusterização
- dendextrend: comparar dois dendrogramas

4.1 Dendrogramas

Todos os dendrogramas foram gerados à partir da base de dados IRIS e utilizando a distância euclidiana entre os pontos.

4.1.1 Single-linkage

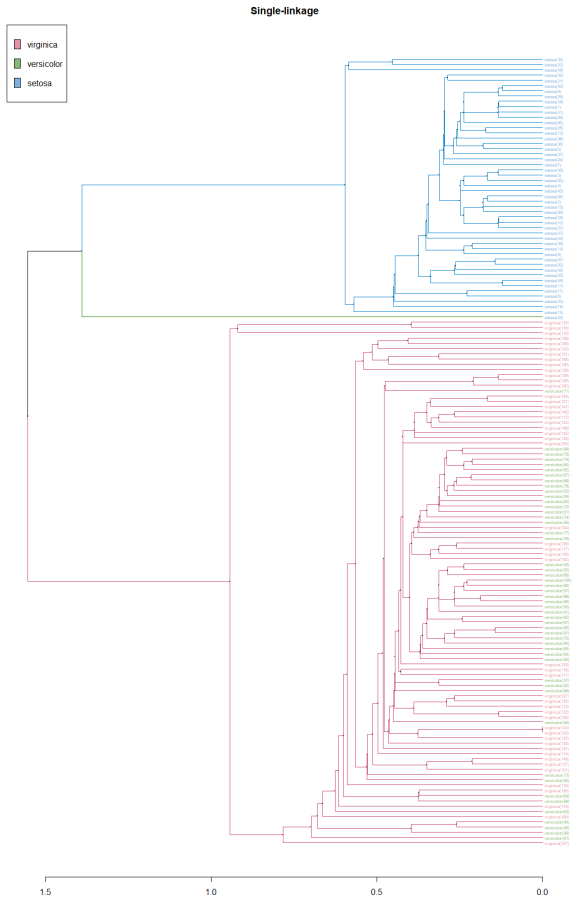


Figura 5: Dendrograma single-linkage

Matriz de Confusão

	setosa	versicolor	virginica
setosa	49	1	0
versicolor	0	0	50
virginica	0	0	50

4.1.2 Complete-linkage

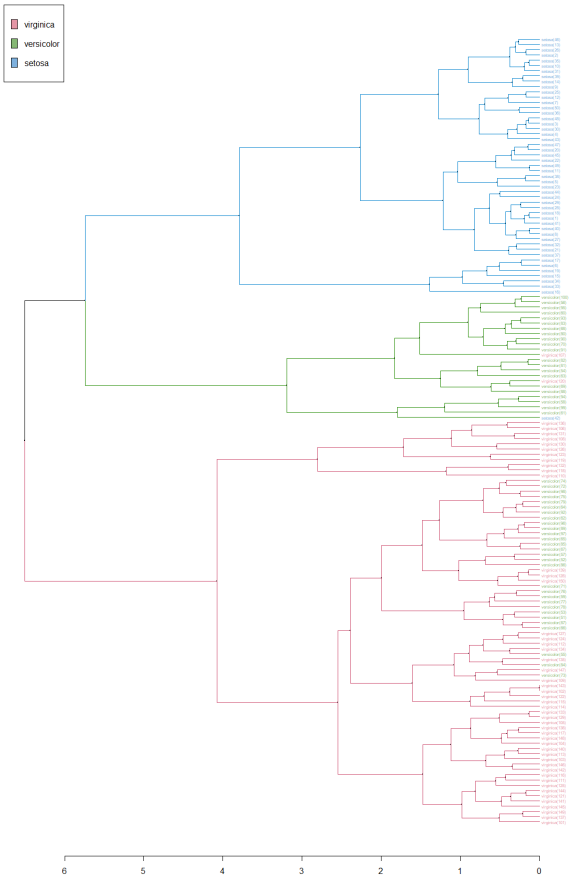


Figura 6: Dendrograma complete-linkage

Matrix de Confusão

	setosa	versicolor	virginica
setosa	49	1	0
versicolor	0	21	29
virginica	0	2	48

4.1.3 Median

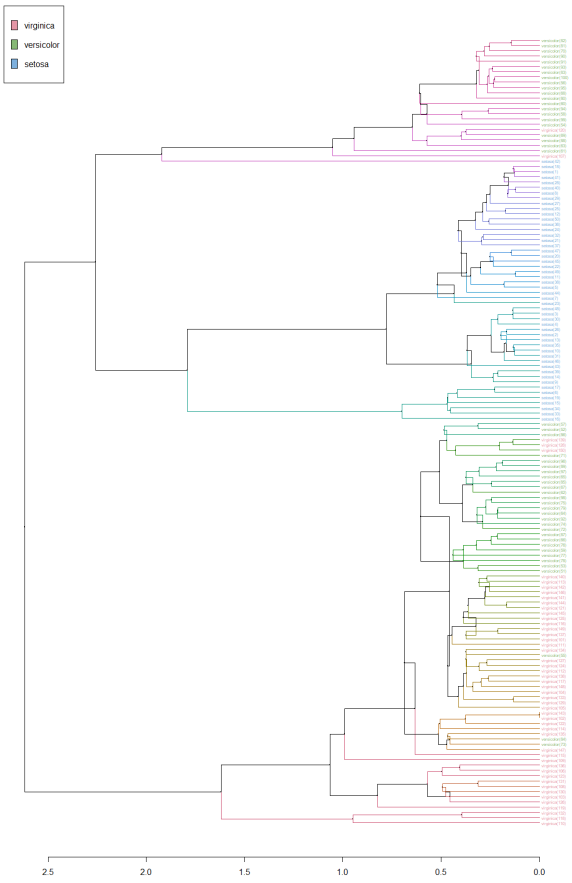


Figura 7: Dendrograma median

Matriz de Confusão

	setosa	versicolor	virginica
setosa	49	1	0
versicolor	0	21	29
virginica	0	2	48

4.1.4 Ward

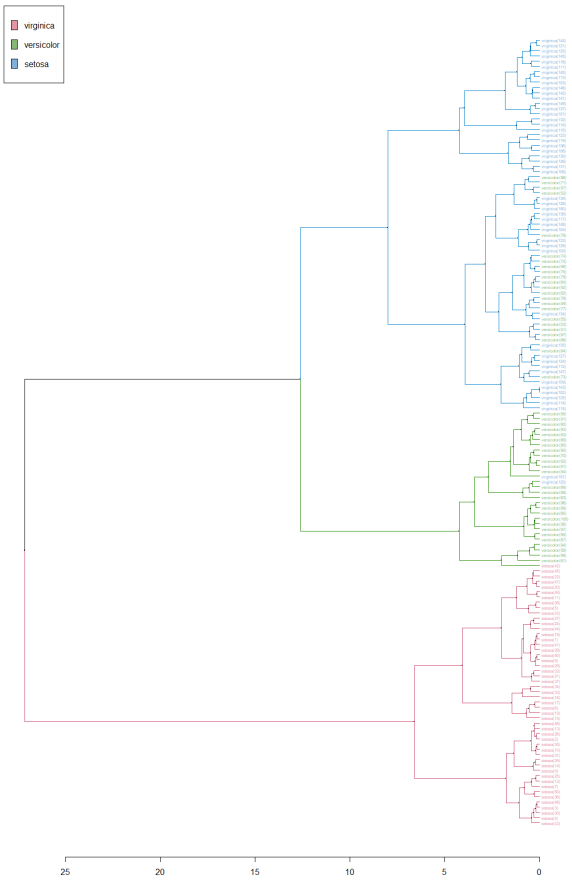


Figura 8: Dendrograma ward.D2

Matriz de Confusão

	setosa	versicolor	virginica
setosa	49	1	0
versicolor	0	27	23
virginica	0	2	48

4.2 Relação entre os dendrogramas

Para mensurarmos a relação entre os dendrogramas gerados pelos diferentes métodos, calcularemos o emaranhamento(entanglement) entre eles, essa medida assume valores entre 0 e 1, quanto menor, mais alinhados são os dendrogramas.

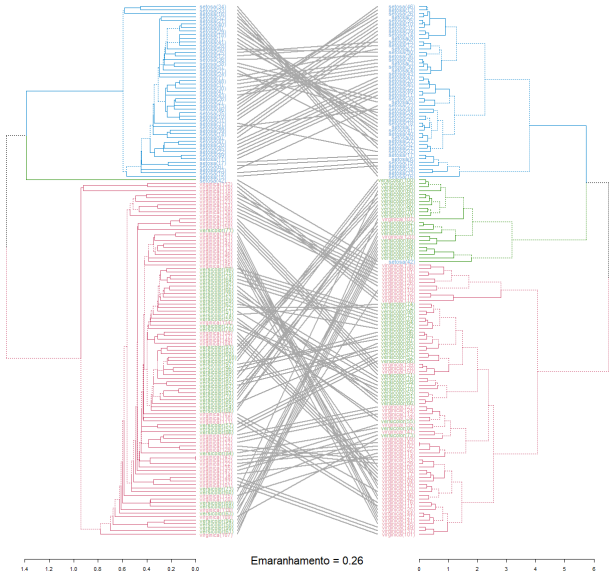


Figura 9: single e complete

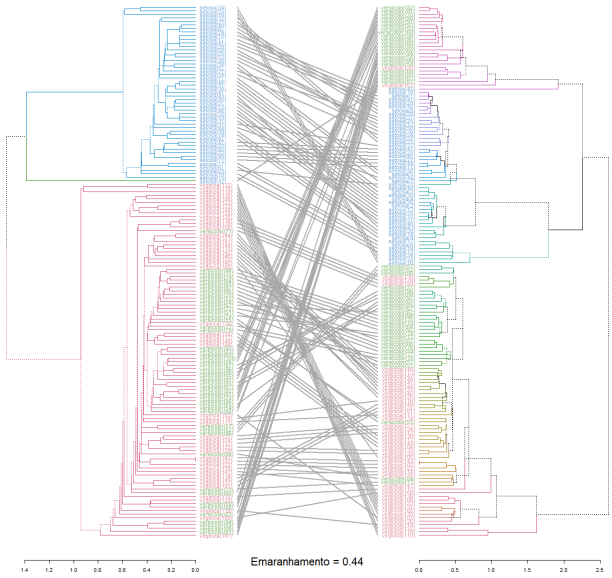


Figura 10: single e median

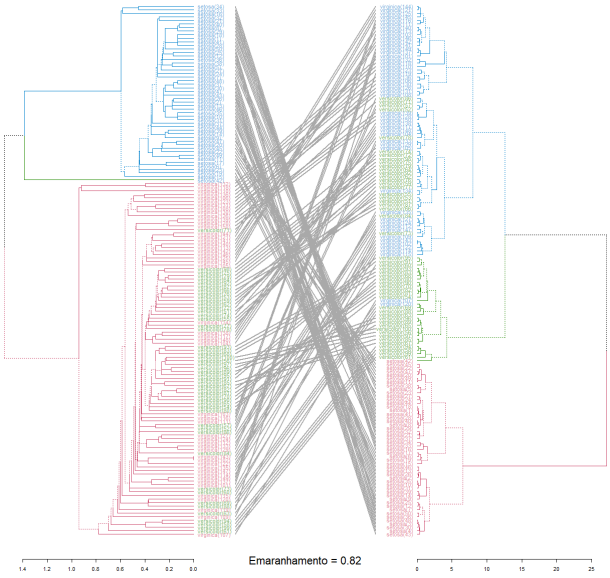


Figura 11: single e ward

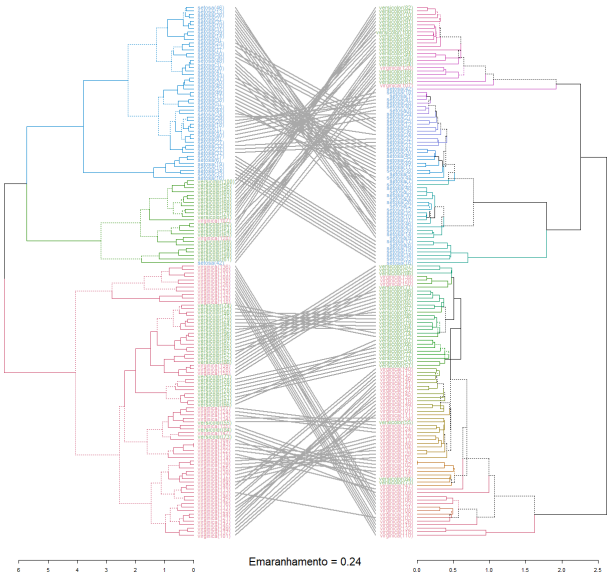


Figura 12: Complete e median

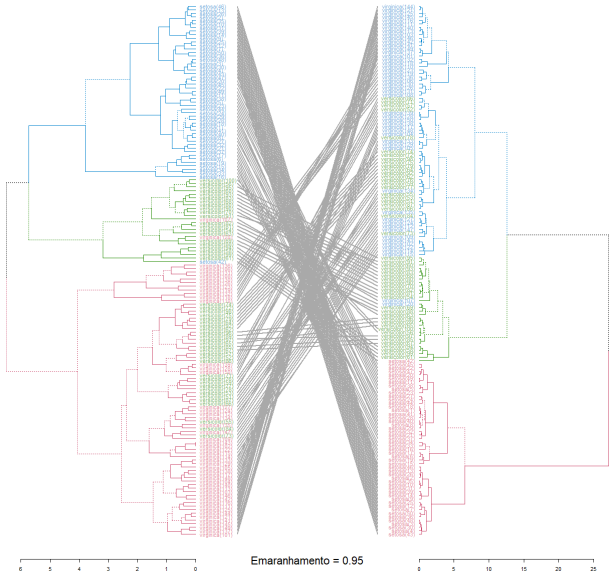


Figura 13: Complete e ward

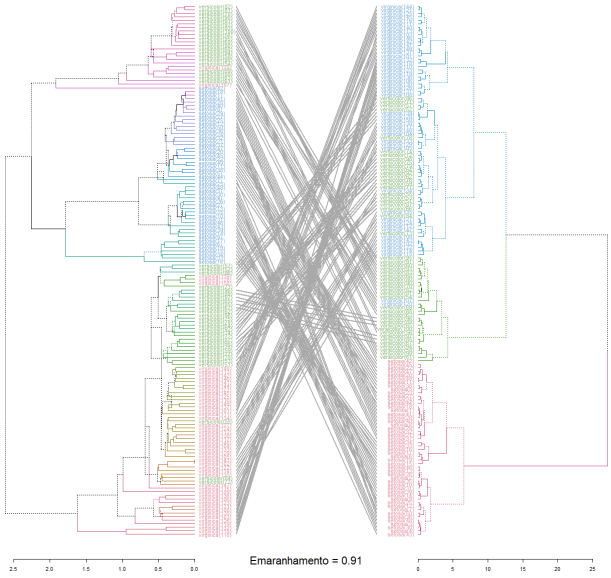


Figura 14: Median e ward

Ao obter todos os valores de emaranhamento par a par dos 4 métodos é tirada a conclusão de que os métodos: single-linkage, complete-linkage e median; agrupam os itens de forma parecida e completamente diferente da forma que o método ward.