

Intro to Computer Science Data Analysis

Jonathan Mendes de Almeida

jonathanalmd@gmail.com

jonathan@aluno.unb.br

@jonydddev (github)

Mar 29, 2018

Useful links

- Pseudocode for k-means: <http://www.devmedia.com.br/data-mining-na-pratica-algoritmo-k-means/4584>
- Similar algorithm: <http://www.di.fc.ul.pt/~jpn/r/spectralclustering/spectralclustering.html> (uses knn plus step one and do not executes the step five) Usa knn junto no passo 1 e nao executa passo 4 do algoritmo do artigo, meio diferente mas funciona

PETAL LENGTH/WIDTH DATA

Init

Get data from file and init vars

```
Dados <- read.csv("iris.data.csv", header=FALSE)
precision <- 5 # float precision
my.data <- as.matrix(Dados[,c(3,4)])

n <- nrow(my.data) # n = number of columns from dataset
S <- my.data # data to S
A <- matrix(rep(0,n^2), nrow = n, ncol=n) # create empty matrix
# sigma2 <- sum((S - mean(S))^2) / (n) #var pop
sigma2 <- 3 # set a sigma (0 to 5 are good values)
D <- diag(n) # create a diagonal matrix
```

Spectral Clustering Algorithm

Step 1: Compute A Matrix (Affinity Matrix)____

```
for (i in 1:n){
  for(j in 1:n){
    if (i != j){
      # Euclidean distance
      A[i,j] <- exp( - sqrt(sum((S[i,]-S[j,])^2)) / 2*sigma2)
      #A[i,j] <- exp(- norm(as.matrix(S[i,]-S[j,]), type="F"))
    }
  }
}
```

```
# set float precision (5)
round(A[1:8,1:8],precision)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
## [1,] 0.00000 1.00000 0.86071 0.86071 1.00000 0.58226 0.86071 0.86071
## [2,] 1.00000 0.00000 0.86071 0.86071 1.00000 0.58226 0.86071 0.86071
## [3,] 0.86071 0.86071 0.00000 0.74082 0.86071 0.51129 0.80886 0.74082
## [4,] 0.86071 0.86071 0.74082 0.00000 0.86071 0.65425 0.80886 1.00000
## [5,] 1.00000 1.00000 0.86071 0.86071 0.00000 0.58226 0.86071 0.86071
## [6,] 0.58226 0.58226 0.51129 0.65425 0.58226 0.00000 0.62229 0.65425
## [7,] 0.86071 0.86071 0.80886 0.80886 0.86071 0.62229 0.00000 0.80886
## [8,] 0.86071 0.86071 0.74082 1.00000 0.86071 0.65425 0.80886 0.00000
```

Step 2: Compute D Matrix

2.1 Calcular matriz D

```
for (i in 1:n){
  # sum of each row and insert into the diagonal matrix D
  D[i,i] <- sum (A[i,])
}
# set float precision (5)
round(D[1:8,1:8],precision)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
## [1,] 39.1257  0.0000  0.00000  0.00000  0.0000  0.00000  0.00000  0.00000
## [2,]  0.0000 39.1257  0.00000  0.00000  0.0000  0.00000  0.00000  0.00000
## [3,]  0.0000  0.0000 36.39367  0.00000  0.0000  0.00000  0.00000  0.00000
## [4,]  0.0000  0.0000  0.00000 39.14224  0.0000  0.00000  0.00000  0.00000
## [5,]  0.0000  0.0000  0.00000  0.00000 39.1257  0.00000  0.00000  0.00000
## [6,]  0.0000  0.0000  0.00000  0.00000  0.0000 31.77652  0.00000  0.00000
## [7,]  0.0000  0.0000  0.00000  0.00000  0.0000  0.00000 37.99526  0.00000
## [8,]  0.0000  0.0000  0.00000  0.00000  0.0000  0.00000  0.00000 39.14224
```

Step 3: Compute L Matrix (using D)

```
# get sqrt of each matrix element
raiz.D <- sqrt (D)          #obs1 : raiz.D %*% raiz.D = D
                                # raiz.D x raiz.D = D (mult matrix)

# solve() para pegar inversa
# solve() function to get inverse matrix
Inv.raiz.D <- solve(raiz.D)  #obs2 : compute inverse matrix = get sqrt from inverse matrix
                                # sqrt (solve (D)) = solve (sqrt (D))

# Compute L
L <- Inv.raiz.D %*% A %*% Inv.raiz.D

# Set float precision (5)
round(L[1:8,1:8],precision)
```

```
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
## [1,] 0.00000 0.02556 0.02281 0.02199 0.02556 0.01651 0.02232 0.02199
## [2,] 0.02556 0.00000 0.02281 0.02199 0.02556 0.01651 0.02232 0.02199
## [3,] 0.02281 0.02281 0.00000 0.01963 0.02281 0.01503 0.02175 0.01963
```

```
## [4,] 0.02199 0.02199 0.01963 0.00000 0.02199 0.01855 0.02097 0.02555
## [5,] 0.02556 0.02556 0.02281 0.02199 0.00000 0.01651 0.02232 0.02199
## [6,] 0.01651 0.01651 0.01503 0.01855 0.01651 0.00000 0.01791 0.01855
## [7,] 0.02232 0.02232 0.02175 0.02097 0.02232 0.01791 0.00000 0.02097
## [8,] 0.02199 0.02199 0.01963 0.02555 0.02199 0.01855 0.02097 0.00000
```

Step 4: Compute eigenvector and set matrix X with the k first eigenvectors from L

```
#get eigenvector
autovet <- eigen(L)$vectors
#autoval <- eigen(L)$values

# set the number of classes (k)
k <- 3

# get the 3 first eigenvectors (k first eigenvalues)
X <- autovet[, (1 : k)]
X
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.09071344 -0.117764013 -0.0075799337
## [2,] 0.09071344 -0.117764013 -0.0075799337
## [3,] 0.08748901 -0.113841916 -0.0076610337
## [4,] 0.09073261 -0.117188034 -0.0068078243
## [5,] 0.09071344 -0.117764013 -0.0075799337
## [6,] 0.08175112 -0.102488451 -0.0024294243
## [7,] 0.08939336 -0.115708328 -0.0070411595
## [8,] 0.09073261 -0.117188034 -0.0068078243
## [9,] 0.09071344 -0.117764013 -0.0075799337
## [10,] 0.08786924 -0.113501100 -0.0066272665
## [11,] 0.09073261 -0.117188034 -0.0068078243
## [12,] 0.08819103 -0.112989140 -0.0054847214
## [13,] 0.08734128 -0.113329680 -0.0072318283
## [14,] 0.07607694 -0.099166977 -0.0068919918
## [15,] 0.08271085 -0.107725846 -0.0073810220
## [16,] 0.08638552 -0.110629953 -0.0053391753
## [17,] 0.08333237 -0.107708733 -0.0064088126
## [18,] 0.08939336 -0.115708328 -0.0070411595
## [19,] 0.08366202 -0.105539709 -0.0032454314
## [20,] 0.08944027 -0.115131015 -0.0062335893
## [21,] 0.08376751 -0.106075137 -0.0037408987
## [22,] 0.08638552 -0.110629953 -0.0053391753
## [23,] 0.07215749 -0.094012043 -0.0064877975
## [24,] 0.07853041 -0.097605882 -0.0013939651
## [25,] 0.07447741 -0.091073467 0.0003416743
## [26,] 0.08819103 -0.112989140 -0.0054847214
## [27,] 0.08471351 -0.107524126 -0.0040694162
## [28,] 0.09073261 -0.117188034 -0.0068078243
## [29,] 0.09071344 -0.117764013 -0.0075799337
## [30,] 0.08819103 -0.112989140 -0.0054847214
## [31,] 0.08819103 -0.112989140 -0.0054847214
## [32,] 0.08638552 -0.110629953 -0.0053391753
## [33,] 0.08786924 -0.113501100 -0.0066272665
```

```

## [34,] 0.09071344 -0.117764013 -0.0075799337
## [35,] 0.08786924 -0.113501100 -0.0066272665
## [36,] 0.08271085 -0.107725846 -0.0073810220
## [37,] 0.08748901 -0.113841916 -0.0076610337
## [38,] 0.08786924 -0.113501100 -0.0066272665
## [39,] 0.08748901 -0.113841916 -0.0076610337
## [40,] 0.09073261 -0.117188034 -0.0068078243
## [41,] 0.08658263 -0.112392751 -0.0072447052
## [42,] 0.08658263 -0.112392751 -0.0072447052
## [43,] 0.08748901 -0.113841916 -0.0076610337
## [44,] 0.07624159 -0.095087502 -0.0017926887
## [45,] 0.07371805 -0.088862328 0.0018130972
## [46,] 0.08939336 -0.115708328 -0.0070411595
## [47,] 0.08819103 -0.112989140 -0.0054847214
## [48,] 0.09071344 -0.117764013 -0.0075799337
## [49,] 0.09073261 -0.117188034 -0.0068078243
## [50,] 0.09071344 -0.117764013 -0.0075799337
## [51,] 0.08866706 0.069178853 0.0564500310
## [52,] 0.08999799 0.069114332 0.0775695766
## [53,] 0.08752052 0.069854531 0.0184859771
## [54,] 0.08093147 0.055643439 0.1248276779
## [55,] 0.09006478 0.069939025 0.0641853011
## [56,] 0.08749210 0.066315650 0.0870229569
## [57,] 0.08940215 0.070461187 0.0405007538
## [58,] 0.05977525 0.021396324 0.0998161680
## [59,] 0.08713435 0.066847791 0.0742326416
## [60,] 0.07681623 0.051719638 0.1168283648
## [61,] 0.06442034 0.030957716 0.1116615439
## [62,] 0.08365724 0.061245639 0.1008110244
## [63,] 0.07490156 0.049358014 0.1192251908
## [64,] 0.08866706 0.069178853 0.0564500310
## [65,] 0.06773771 0.038889599 0.1120583944
## [66,] 0.08851538 0.066643016 0.0944899740
## [67,] 0.08999799 0.069114332 0.0775695766
## [68,] 0.07609660 0.051711809 0.1153111767
## [69,] 0.08999799 0.069114332 0.0775695766
## [70,] 0.07574186 0.048978379 0.1250706352
## [71,] 0.08773876 0.070415033 0.0053454640
## [72,] 0.08093147 0.055643439 0.1248276779
## [73,] 0.08752052 0.069854531 0.0184859771
## [74,] 0.08278227 0.063730828 0.0621365538
## [75,] 0.08643926 0.063562551 0.1083189175
## [76,] 0.08851538 0.066643016 0.0944899740
## [77,] 0.08720750 0.068655830 0.0410231563
## [78,] 0.08770080 0.071173989 -0.0151872774
## [79,] 0.08999799 0.069114332 0.0775695766
## [80,] 0.06442034 0.030957716 0.1116615439
## [81,] 0.07317926 0.045286194 0.1235948263
## [82,] 0.06883456 0.039258843 0.1180821693
## [83,] 0.07730259 0.050800767 0.1261136693
## [84,] 0.08532541 0.069394184 -0.0206260865
## [85,] 0.08999799 0.069114332 0.0775695766
## [86,] 0.08835372 0.068157802 0.0670441706
## [87,] 0.08965473 0.070323153 0.0495923362

```

##	[88,]	0.08724541	0.065209473	0.0984581523
##	[89,]	0.08321938	0.058723590	0.1216972690
##	[90,]	0.08093147	0.055643439	0.1248276779
##	[91,]	0.08459724	0.062697343	0.0987678615
##	[92,]	0.08932527	0.068979729	0.0706870305
##	[93,]	0.07991815	0.054295565	0.1260328156
##	[94,]	0.05977525	0.021396324	0.0998161680
##	[95,]	0.08506040	0.061368895	0.1161033253
##	[96,]	0.08326068	0.059490535	0.1166388437
##	[97,]	0.08506040	0.061368895	0.1161033253
##	[98,]	0.08643926	0.063562551	0.1083189175
##	[99,]	0.05319589	0.006702448	0.0757059456
##	[100,]	0.08321938	0.058723590	0.1216972690
##	[101,]	0.06309104	0.054321401	-0.1436183223
##	[102,]	0.08629403	0.071040977	-0.0492991358
##	[103,]	0.07193013	0.061597315	-0.1460390494
##	[104,]	0.07785519	0.065533736	-0.1048212730
##	[105,]	0.07372406	0.063070506	-0.1457280861
##	[106,]	0.05275707	0.045803918	-0.1446435029
##	[107,]	0.08602007	0.066635334	0.0557087870
##	[108,]	0.05938471	0.051036456	-0.1343466965
##	[109,]	0.07310239	0.061995413	-0.1206577574
##	[110,]	0.06075797	0.052405997	-0.1435113449
##	[111,]	0.08468410	0.069914962	-0.0570826067
##	[112,]	0.08364387	0.069621528	-0.0773786834
##	[113,]	0.07995226	0.067560335	-0.1174786575
##	[114,]	0.08461954	0.069410372	-0.0423231265
##	[115,]	0.07299475	0.060713968	-0.0739101167
##	[116,]	0.07720033	0.064848560	-0.0988379576
##	[117,]	0.08001359	0.067061511	-0.0945875179
##	[118,]	0.04953223	0.043080762	-0.1401591995
##	[119,]	0.04294485	0.037398954	-0.1242690690
##	[120,]	0.08574524	0.068945152	0.0030428915
##	[121,]	0.07406799	0.063254137	-0.1408019619
##	[122,]	0.08385186	0.068276391	-0.0274909234
##	[123,]	0.04976397	0.043205154	-0.1371696607
##	[124,]	0.08805097	0.071243300	-0.0096883908
##	[125,]	0.07670363	0.065311598	-0.1362697759
##	[126,]	0.06786848	0.057912894	-0.1303887603
##	[127,]	0.08773876	0.070415033	0.0053454640
##	[128,]	0.08805097	0.071243300	-0.0096883908
##	[129,]	0.07856415	0.066660527	-0.1281032360
##	[130,]	0.06965593	0.058698969	-0.1008039138
##	[131,]	0.06607377	0.056653191	-0.1402980714
##	[132,]	0.05804394	0.050161858	-0.1454278390
##	[133,]	0.07742465	0.065813683	-0.1319909565
##	[134,]	0.08364646	0.067710781	-0.0117921874
##	[135,]	0.06958982	0.057687364	-0.0641113612
##	[136,]	0.06463070	0.055709444	-0.1506636491
##	[137,]	0.07293261	0.062129078	-0.1317536015
##	[138,]	0.08001359	0.067061511	-0.0945875179
##	[139,]	0.08773876	0.070415033	0.0053454640
##	[140,]	0.08096648	0.068102321	-0.1050871074
##	[141,]	0.07293261	0.062129078	-0.1317536015

```
## [142,] 0.07652196 0.063573172 -0.0724524481
## [143,] 0.08629403 0.071040977 -0.0492991358
## [144,] 0.06988244 0.059996416 -0.1493266975
## [145,] 0.06838047 0.058440888 -0.1332987362
## [146,] 0.07710166 0.064427853 -0.0858867645
## [147,] 0.08662257 0.070846811 -0.0342376639
## [148,] 0.08406067 0.069800936 -0.0713677783
## [149,] 0.07692306 0.064924906 -0.1111551895
## [150,] 0.08665450 0.071074334 -0.0399197845
```

```
#dec.espec <- autovet %*% diag(autoval) %*% t(autovet) -> return to L matrix
```

Step 5: Compute Y Matrix using X Matrix: Normalize Y Matrix

```
# Create a matrix with n rows and k columns
Y <- matrix (0,nrow=n,ncol=k)

# For each element from X, div(elem)/sqrt(each element from row squared)
for(i in 1:n){
  for(j in 1:k){
    Y[i,j] <- X[i,j] / (sqrt (sum(X[i,j])^2))
  }
}
# Normalized Y matrix: only 1 and -1 values
Y
```

```
##      [,1] [,2] [,3]
## [1,]    1   -1   -1
## [2,]    1   -1   -1
## [3,]    1   -1   -1
## [4,]    1   -1   -1
## [5,]    1   -1   -1
## [6,]    1   -1   -1
## [7,]    1   -1   -1
## [8,]    1   -1   -1
## [9,]    1   -1   -1
## [10,]    1   -1   -1
## [11,]    1   -1   -1
## [12,]    1   -1   -1
## [13,]    1   -1   -1
## [14,]    1   -1   -1
## [15,]    1   -1   -1
## [16,]    1   -1   -1
## [17,]    1   -1   -1
## [18,]    1   -1   -1
## [19,]    1   -1   -1
## [20,]    1   -1   -1
## [21,]    1   -1   -1
## [22,]    1   -1   -1
## [23,]    1   -1   -1
## [24,]    1   -1   -1
## [25,]    1   -1    1
## [26,]    1   -1   -1
```

##	[27,]	1	-1	-1
##	[28,]	1	-1	-1
##	[29,]	1	-1	-1
##	[30,]	1	-1	-1
##	[31,]	1	-1	-1
##	[32,]	1	-1	-1
##	[33,]	1	-1	-1
##	[34,]	1	-1	-1
##	[35,]	1	-1	-1
##	[36,]	1	-1	-1
##	[37,]	1	-1	-1
##	[38,]	1	-1	-1
##	[39,]	1	-1	-1
##	[40,]	1	-1	-1
##	[41,]	1	-1	-1
##	[42,]	1	-1	-1
##	[43,]	1	-1	-1
##	[44,]	1	-1	-1
##	[45,]	1	-1	1
##	[46,]	1	-1	-1
##	[47,]	1	-1	-1
##	[48,]	1	-1	-1
##	[49,]	1	-1	-1
##	[50,]	1	-1	-1
##	[51,]	1	1	1
##	[52,]	1	1	1
##	[53,]	1	1	1
##	[54,]	1	1	1
##	[55,]	1	1	1
##	[56,]	1	1	1
##	[57,]	1	1	1
##	[58,]	1	1	1
##	[59,]	1	1	1
##	[60,]	1	1	1
##	[61,]	1	1	1
##	[62,]	1	1	1
##	[63,]	1	1	1
##	[64,]	1	1	1
##	[65,]	1	1	1
##	[66,]	1	1	1
##	[67,]	1	1	1
##	[68,]	1	1	1
##	[69,]	1	1	1
##	[70,]	1	1	1
##	[71,]	1	1	1
##	[72,]	1	1	1
##	[73,]	1	1	1
##	[74,]	1	1	1
##	[75,]	1	1	1
##	[76,]	1	1	1
##	[77,]	1	1	1
##	[78,]	1	1	-1
##	[79,]	1	1	1
##	[80,]	1	1	1

##	[81,]	1	1	1
##	[82,]	1	1	1
##	[83,]	1	1	1
##	[84,]	1	1	-1
##	[85,]	1	1	1
##	[86,]	1	1	1
##	[87,]	1	1	1
##	[88,]	1	1	1
##	[89,]	1	1	1
##	[90,]	1	1	1
##	[91,]	1	1	1
##	[92,]	1	1	1
##	[93,]	1	1	1
##	[94,]	1	1	1
##	[95,]	1	1	1
##	[96,]	1	1	1
##	[97,]	1	1	1
##	[98,]	1	1	1
##	[99,]	1	1	1
##	[100,]	1	1	1
##	[101,]	1	1	-1
##	[102,]	1	1	-1
##	[103,]	1	1	-1
##	[104,]	1	1	-1
##	[105,]	1	1	-1
##	[106,]	1	1	-1
##	[107,]	1	1	1
##	[108,]	1	1	-1
##	[109,]	1	1	-1
##	[110,]	1	1	-1
##	[111,]	1	1	-1
##	[112,]	1	1	-1
##	[113,]	1	1	-1
##	[114,]	1	1	-1
##	[115,]	1	1	-1
##	[116,]	1	1	-1
##	[117,]	1	1	-1
##	[118,]	1	1	-1
##	[119,]	1	1	-1
##	[120,]	1	1	1
##	[121,]	1	1	-1
##	[122,]	1	1	-1
##	[123,]	1	1	-1
##	[124,]	1	1	-1
##	[125,]	1	1	-1
##	[126,]	1	1	-1
##	[127,]	1	1	1
##	[128,]	1	1	-1
##	[129,]	1	1	-1
##	[130,]	1	1	-1
##	[131,]	1	1	-1
##	[132,]	1	1	-1
##	[133,]	1	1	-1
##	[134,]	1	1	-1


```
## [135,] 1 1 -1
## [136,] 1 1 -1
## [137,] 1 1 -1
## [138,] 1 1 -1
## [139,] 1 1 1
## [140,] 1 1 -1
## [141,] 1 1 -1
## [142,] 1 1 -1
## [143,] 1 1 -1
## [144,] 1 1 -1
## [145,] 1 1 -1
## [146,] 1 1 -1
## [147,] 1 1 -1
## [148,] 1 1 -1
## [149,] 1 1 -1
## [150,] 1 1 -1
```

K-Means (Step 6)

Step 6.1: Set vars and set centroids

```
xnew <- Y
obs <- as.numeric()

# seta centroides iniciais (criando combinacao de -1 e 1) -> RUIM
#center1<- sample(seq(-1,1,by=0.1),3,replace=T)
#center2<- sample(seq(-1,1,by=0.1),3,replace=T)
#center3<- sample(seq(-1,1,by=0.1),3,replace=T)

# seta centroides iniciais com heuristica (sabendo que esses 3 pontos sao disitntos)
#center1 <- xnew[1,]
#center2 <- xnew[70,]
#center3 <- xnew[149,]

# trocar -1 por 0 para ficar tudo com valores 0 ou 1
#for(n in 1:150){
#  for(m in 1:3){
#    if (xnew[n,m] == -1){
#      xnew[n,m] = 0
#    }
#  }
#}

# get different centroids
flag <- TRUE
while(flag){
  center1 <- xnew[sample(1:150,1),]
  center2 <- xnew[sample(1:150,1),]
  center3 <- xnew[sample(1:150,1),]
  if (!(all(center1 == center2) || all(center1 == center3) || all(center2 == center3)) ){
    icenter1 <- center1
    icenter2 <- center2
    icenter3 <- center3
  }
}
```

```

    flag <- FALSE
  }
}

```

Step 6.2: K-Means Algorithm and Plot Graphs

```

for(n in 1:40000){ # upgrade centroids
  for(i in 1:150){ # 150 instances
    dist1<- sum((xnew[i,]-center1)^2)
    dist2<- sum((xnew[i,]-center2)^2)
    dist3<- sum((xnew[i,]-center3)^2)

    if(dist1<=dist2 && dist1<=dist3){
      obs[i]<-1
    }
    else if(dist2<=dist1 && dist2<=dist3){
      obs[i]<-2
    }
    else{
      obs[i]<-3
    }
  }

  grupo1<-xnew[(obs == 1),]
  grupo2<-xnew[(obs == 2),]
  grupo3<-xnew[(obs == 3),]

  d1 <- dim(grupo1)[1]
  d2 <- dim(grupo2)[1]
  d3 <- dim(grupo3)[1]

  # Check if different class
  if (d1 != 0){
    center1<-c(mean(grupo1[,1]),mean(grupo1[,2]),mean(grupo1[,3]))
  }

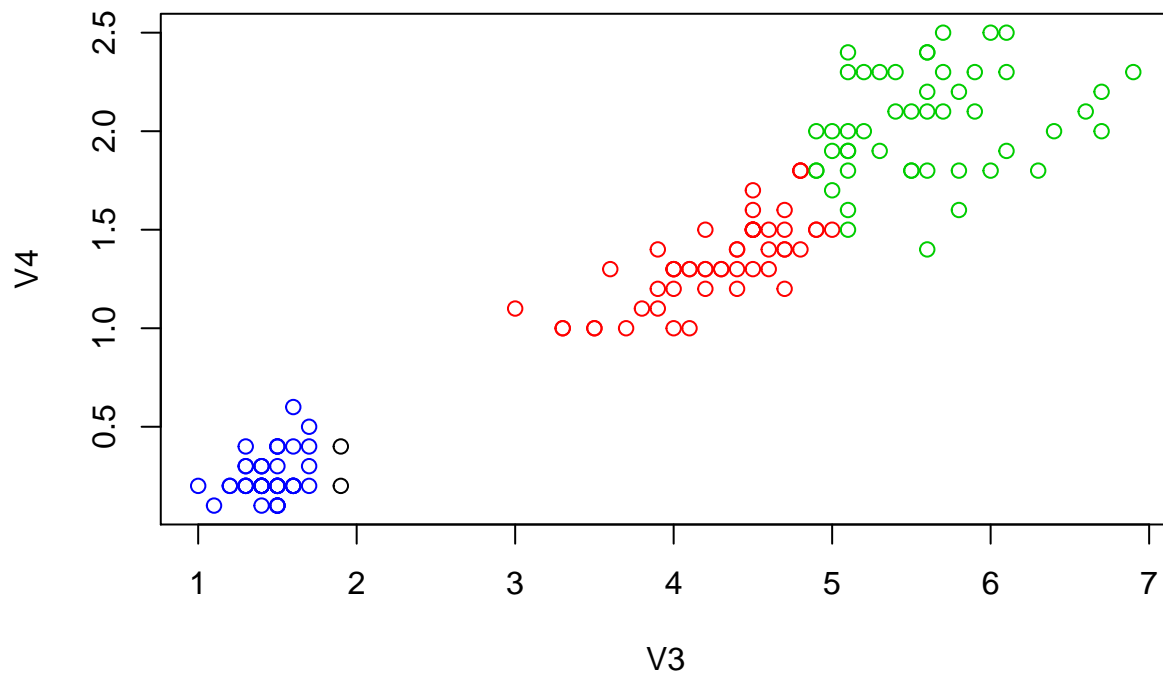
  if (d2 != 0){
    center2<-c(mean(grupo2[,1]),mean(grupo2[,2]),mean(grupo2[,3]))
  }

  if (d3 != 0){
    center3<-c(mean(grupo3[,1]),mean(grupo3[,2]),mean(grupo3[,3]))
  }

}

# R kmeans
km <- kmeans (Y,4,nstart=20)
#km <- kmeans (X,3,nstart=20)
plot(S, col=km$cluster)

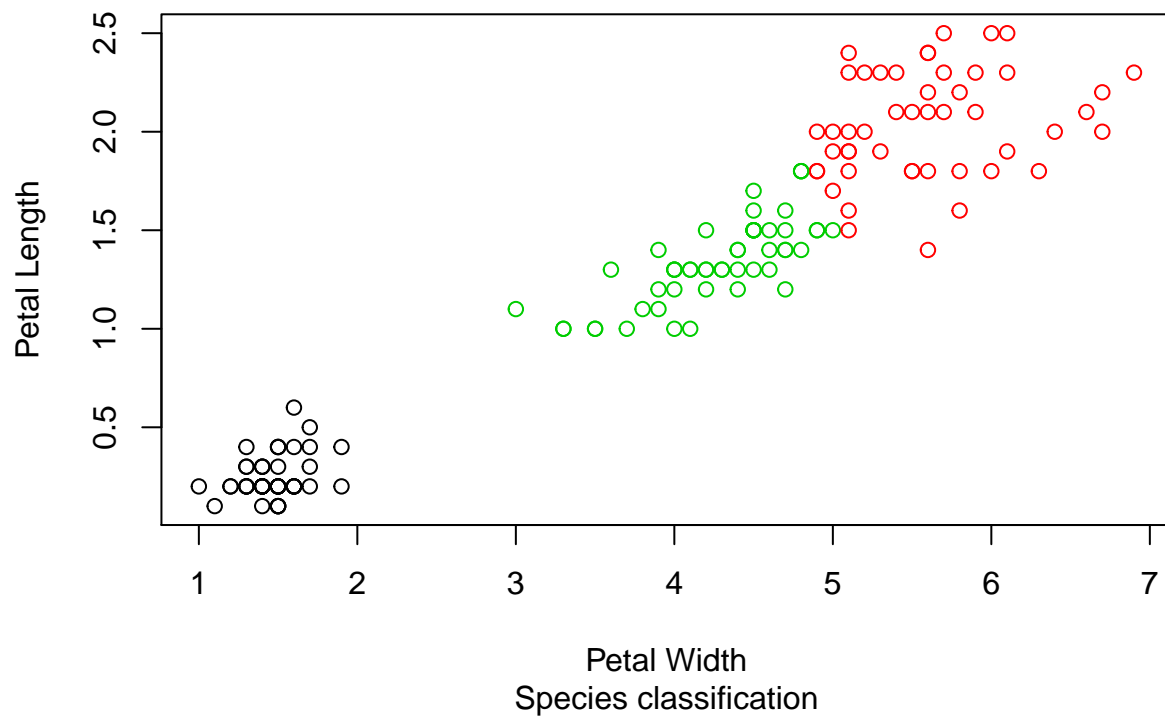
```



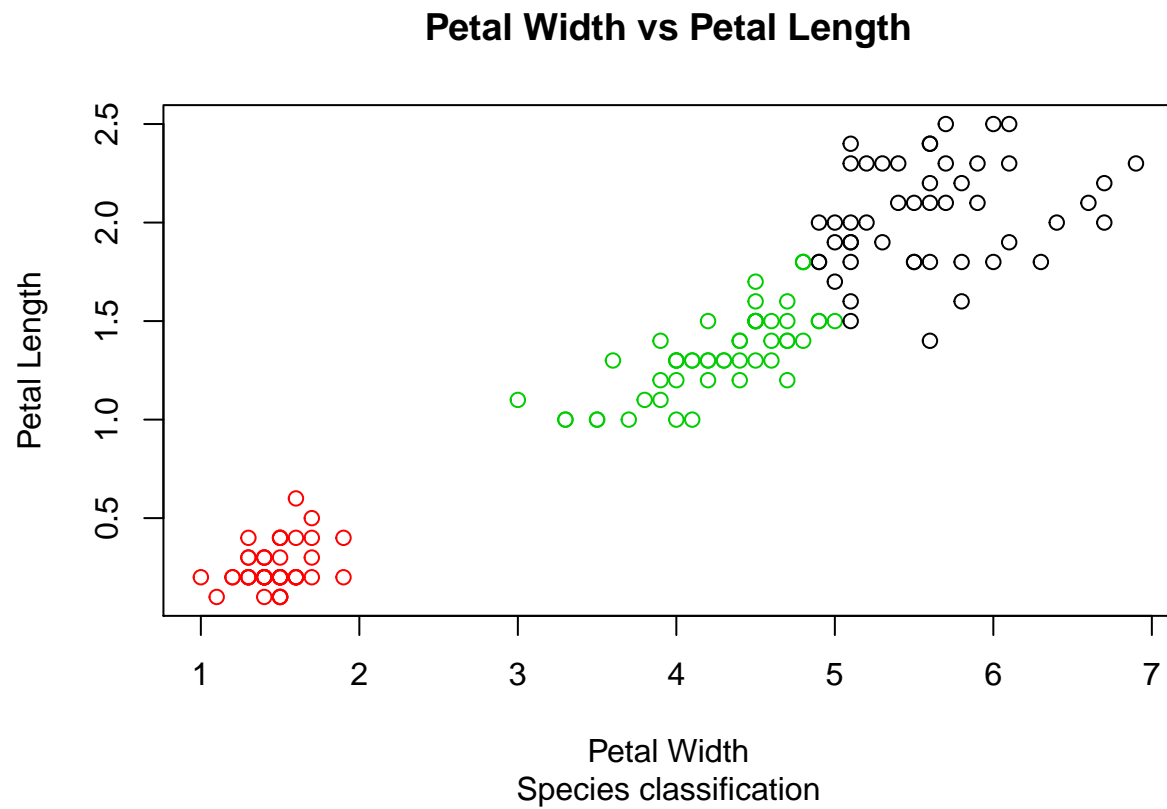
```
km <- kmeans (Y,3,nstart=20)
#km <- kmeans (X,3,nstart=20)

# Plot both
plot(S, col=km$cluster, main = 'Petal Width vs Petal Length', sub = 'Species classification', xlab = 'P
```

Petal Width vs Petal Length



```
plot(S, col=obs, main = 'Petal Width vs Petal Length', sub = 'Species classification', xlab = 'Petal Width
```



Compare centroinds values (initial and final values)

```
center1
```

```
## [1] 1 1 -1
```

```
center2
```

```
## [1] 1.00 -1.00 -0.92
```

```
center3
```

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

```
icenter1
```

```
## [1] 1 1 -1
```

```
icenter2
```

```
## [1] 1 -1 -1
```

```
icenter3
```

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