

```

1. # calcul des proportions
2. prop.app = function(Xapp, zapp) {
3.     zapp = factor(zapp)
4.     g = length(levels(zapp)) # nombre de classes
5.     n = dim(Xapp)[1] # n = nombre d'individus total
6.     p = dim(Xapp)[2] # p = nombre de variables
7.     prop = list()
8.     for (k in 1:g) {
9.         prop[[k]] = table(zapp)[k] / length(zapp)
10.    }
11.    return(prop)
12. }
13.
14.
15. # calcul des centres de gravité
16. mu.app = function(Xapp, zapp) {
17.     zapp = factor(zapp)
18.     g = length(levels(zapp)) # nombre de classes
19.     n = dim(Xapp)[1] # n = nombre d'individus total
20.     p = dim(Xapp)[2] # p = nombre de variables
21.     mu = list()
22.     for (k in 1:g) {
23.         dataClassK = Xapp[zapp == levels(zapp)[k],]
24.         mu[[k]] = apply(dataClassK, MARGIN = 2, mean)
25.     }
26.     return(mu)
27. }
28.
29.
30. # calcul des matrices de covariance
31. sigma.app = function(Xapp, zapp) {
32.     zapp = factor(zapp)
33.     g = length(levels(zapp))
34.     n = dim(Xapp)[1]
35.     p = dim(Xapp)[2]
36.     sigma = list()
37.     for (k in 1:g) {
38.         dataClassK = Xapp[zapp == levels(zapp)[k],]
39.         sigma[[k]] = var(dataClassK)
40.     }
41.     return(sigma)
42. }
43.
44.
45.
46.
47.
48.
49.
50.
51.
52.
53.
54.
55.
56.
57.
58.

```

```

59.
60.
61.
62.
63.
64.
65. adq.app <- function(Xapp, zapp) {
66.   zapp = factor(zapp)
67.   params = list()
68.   params[["pi"]] = prop.app(Xapp, zapp)
69.   params[["mu"]] = mu.app(Xapp, zapp)
70.   params[["sigma"]] = sigma.app(Xapp, zapp)
71.   return(params)
72. }
73.
74.
75. adl.app <- function(Xapp, zapp) {
76.   zapp = factor(zapp)
77.   g = length(levels(zapp))
78.   p = dim(Xapp)[2]
79.   params = list()
80.   prop = prop.app(Xapp, zapp)
81.   mu = mu.app(Xapp, zapp)
82.   classes_sigma = sigma.app(Xapp, zapp)
83.   sigma = matrix(0, p, p)
84.   for (k in 1:g) {
85.     sigma = sigma + (prop[[k]] * classes_sigma[[k]])
86.   }
87.   for (k in 1:g)
88.     params[["sigma"]][[k]] = sigma
89.   params[["pi"]] = prop
90.   params[["mu"]] = mu
91.   return(params)
92. }
93.
94.
95. nba.app <- function(Xapp, zapp) {
96.   zapp = factor(zapp)
97.   g = length(levels(zapp))
98.   params = list()
99.   prop = prop.app(Xapp, zapp)
100.  mu = mu.app(Xapp, zapp)
101.  classes_sigma = sigma.app(Xapp, zapp)
102.  for (k in 1:g)
103.    classes_sigma[[k]] = diag(diag(classes_sigma[[k]]))
104.  params[["pi"]] = prop
105.  params[["mu"]] = mu
106.  params[["sigma"]] = classes_sigma
107.  return(params)
108. }
109.
110.
111.
112.
113.
114.
115.
116.

```

```

117.
118.
119.
120.
121.
122.
123.
124.
125.
126.
127.
128. ad.val <- function(params, Xtst) {
129.     n = nrow(Xtst)
130.
131.     f1 = mvdnorm(Xtst, params$mu[[1]], params$sigma[[1]])
132.     f2 = mvdnorm(Xtst, params$mu[[2]], params$sigma[[2]])
133.
134.     discrimination = list()
135.     discrimination[["pw1"]] = vector(length = n)
136.     discrimination[["pw2"]] = vector(length = n)
137.     discrimination[["pred"]] = vector(length = n)
138.
139.     for (i in 1 : n) {
140.         discrimination[["pw1"]][i] = f1[i]*params$pi[[1]]/(f1[i]*params$pi[[1]]+
f2[i]*params$pi[[2]])
141.         discrimination[["pw2"]][i] = f2[i]*params$pi[[2]]/(f1[i]*params$pi[[1]]+
f2[i]*params$pi[[2]])
142.
143.         if(discrimination[["pw1"]][i] > discrimination[["pw2"]][i])
144.             discrimination[["pred"]][i] = 1
145.         else
146.             discrimination[["pred"]][i] = 2
147.     }
148.     discrimination[["pred"]] = factor(discrimination[["pred"]])
149.     return(discrimination)
150. }

```