```r
# calcul des proportions
prop.app = function(Xapp, zapp) {
    zapp = factor(zapp)
    g = length(levels(zapp)) # nombre de classes
    n = dim(Xapp)[1] # n = nombre d'individus total
    p = dim(Xapp)[2] # p = nombre de variables
    prop = list()
    for (k in 1:g) {
        prop[[k]] = table(zapp)[k] / length(zapp)
    }
    return(prop)
}


# calcul des centres de gravité
mu.app = function(Xapp, zapp) {
    zapp = factor(zapp)
    g = length(levels(zapp)) # nombre de classes
    n = dim(Xapp)[1] # n = nombre d'individus total
    p = dim(Xapp)[2] # p = nombre de variables
    mu = list()
    for (k in 1:g) {
        dataClassK = Xapp[zapp == levels(zapp)[k],]
        mu[[k]] = apply(dataClassK, MARGIN = 2, mean)
    }
    return(mu)
}


# calcul des matrices de covariance
sigma.app = function(Xapp, zapp) {
    zapp = factor(zapp)
    g = length(levels(zapp))
    n = dim(Xapp)[1]
    p = dim(Xapp)[2]
    sigma = list()
    for (k in 1:g) {
        dataClassK = Xapp[zapp == levels(zapp)[k],]
        sigma[[k]] = var(dataClassK)
    }
    return(sigma)
}
```

```r


adq.app <- function(Xapp, zapp) {
    zapp = factor(zapp)
    params = list()
    params[["pi"]] = prop.app(Xapp, zapp)
    params[["mu"]] = mu.app(Xapp, zapp)
    params[["sigma"]] = sigma.app(Xapp, zapp)
    return(params)
}


adl.app <- function(Xapp, zapp) {
    zapp = factor(zapp)
    g = length(levels(zapp))
    p = dim(Xapp)[2]
    params = list()
    prop = prop.app(Xapp, zapp)
    mu = mu.app(Xapp, zapp)
    classes_sigma = sigma.app(Xapp, zapp)
    sigma = matrix(0, p, p)
    for (k in 1:g) {
        sigma = sigma + (prop[[k]] * classes_sigma[[k]])
    }
    for (k in 1:g)
        params[["sigma"]][[k]] = sigma
    params[["pi"]] = prop
    params[["mu"]] = mu
    return(params)
}


nba.app <- function(Xapp, zapp) {
    zapp = factor(zapp)
    g = length(levels(zapp))
    params = list()
    prop = prop.app(Xapp, zapp)
    mu = mu.app(Xapp, zapp)
    classes_sigma = sigma.app(Xapp, zapp)
    for (k in 1:g)
        classes_sigma[[k]] = diag(diag(classes_sigma[[k]]))
    params[["pi"]] = prop
    params[["mu"]] = mu
    params[["sigma"]] = classes_sigma
    return(params)
}


```

```r
ad.val <- function(params, Xtst) {
    n = nrow(Xtst)

    f1 = mvdnorm(Xtst, params$mu[[1]], params$sigma[[1]])
    f2 = mvdnorm(Xtst, params$mu[[2]], params$sigma[[2]])

    discrimination = list()
    discrimination[["pw1"]] = vector(length = n)
    discrimination[["pw2"]] = vector(length = n)
    discrimination[["pred"]] = vector(length = n)

    for (i in 1 : n) {
        discrimination[["pw1"]][i] = f1[i]*params$pi[[1]]/(f1[i]*params$pi[[1]]+
f2[i]*params$pi[[2]])
        discrimination[["pw2"]][i] = f2[i]*params$pi[[2]]/(f1[i]*params$pi[[1]]+
f2[i]*params$pi[[2]])

        if(discrimination[["pw1"]][i] > discrimination[["pw2"]][i])
            discrimination[["pred"]][i] = 1
        else
            discrimination[["pred"]][i] = 2
    }
    discrimination[["pred"]] = factor(discrimination[["pred"]])
    return(discrimination)
}
```