Apendix A Computer code in ‘R’ for the algorithm of least squares approach of nonparametric discovery process model.

```
nonpara <- function(y, grou , ini){
# y is the observed discovery series
# grou contains the upper end points for the K intervals
# ini contains the initial values for lambda and beta, ini = (0 .1, 0) is tested to
work well
k <- length(grou) # find out how many classes there are
n <- length(y) # find out the length of the discovery series
b <- c(O, grou[-k]) # aO =0, a1 , 00 ', a_(K-1)
e<-grou #a1 , a2, oo. , aK
nj <- rep(O ,k) # create a vector of length k, filled with value zero
vj <- (b+e)/2 # middle values of the K intervals
adjy <- rep(O,n) # create a vector of length n as the space to hold the adjusted
#discovery series
forU in 1 :k) {
d <- Y >= b[j] & Y < e[j]
njU] <- sum(d)
adjy[d]<-vj[j]
}
si <- cumsum(adjy)
# identify the y values that fall into interval j
# find nj for interval j
# assign the value vj to all the y values that fall into interval j
# si contains the cumulative discovered volume for i = 1, 2, 00' ,
#n
# below is the target function for applying the method of least squares to estimate
lambda
# and beta. In Fortran , fn is the target function with two arguments lambda and
beta, and
# n, nj, vj #and si are shared in the commom block
fn <- function(theta , n, nj, vj, si){
lam<- theta[1] # assign lambda
b <- theta[2] # assign beta
w <- vtb # assign K weights into vector w
ti <- rep(O ,n) # create a vector to hold t1 , t2, 00 ' , tn
njhat <- nj/(1-exp(-w*lam)) # find estimates nj-hat of nj
mjO <- njhat # compute starting values mj(O) to solve (2) and (3) to get mj(i)
qjO <- w*mjO/sum(w*mjO) # compute starting values qj(O) to solve (2) and (3) to
get mj(i)
for(i in 1 :n) {
mj 1 <- mjO-qjO
mj1 [mj1 <0]<-0
mjO <- mj1
```

```
qjO <- w*mjO/sum(w*mjO)
ti[i] <- sum«njhat-mj1 )*vj)
}
sum«si-ti)1\2)
10 <- c(0 .0005,-2)
up <- c(1 0,4)
}
# make sure that the computed mj(i) >= 0
# update mj(i)
# update qj(i)
# compute the sum of squares target function value
# call routine "optim" to minimize the target function fn , ini denotes the initial
values for
#lambda and beta
out <- optim(ini, fn , method="Nelder-Mead", control=list(maxit=1 000), n=n ,
nj=nj, vj=vj ,
si=si)
Isest <- out$par
lam hat <- Isest[1]
bhat <- Isest[2]
w <- vj"bhat
# the least squares estimates of lambda and beta
njhat <- nj/(1-exp(-w*lamhat))
Nhat <- round(sum(njhat))
Rhat <- sum(round(njhat)*vj)
ti <- rep(O, Nhat)
# compute the final estimated nj
# find the estimate of N, rounded to a integer
# find the estimate of the total volume R
# create a vector to hold t1 , t2 , ... , t_Nhat
mjO <- njhat
qjO <- w*mjO/sum(w*mjO)
for(i in 1 :Nhat) {
mj1 <- mjO-qjO
mj1 [mj1 <0]<-0
mjO <- mj1
qjO <- w*mjO/sum(w*mjO)
ti[i] <- sum((njhat-mj1 )*vj)
}
# compute starting values mj(O) to solve (2) and (3) to get mj(i)
# compute starting values qj(O) to solve (2) and (3) to get mj(i)
# make sure that the computed mj(i) >= 0
# update mj(i)
# update qj(i)
# plot the mean cumulative discovered volume
```

```
plot(ti ,type="l", xlab="Order of Discovery", ylab="Cumulative Volume",col=4
,lwd=3)
points(1 :n, si, col=2) # overlay the observed cumulative discovered volume
# output least squares estimates of lambda and beta, njhat, Nhat and Rhat
print( c("lambda", "beta"))
print(round(lsest,3))
print("Estimated Nj")
print(round(njhat))
print("Estimated total N")
print(N hat)
print("Estimated total volume")
print(Rhat)
list(out$value, out$convergence)
}
```