```
## PROGRAMA ALGORITMO GENÉTICO, FUNÇÃO OBJETIVO:RESÍDUO SSE e 221 individuos
## probabilidade de mutacao 0.4 e 100 solucoes, 100 GERACOES, 21 cromossomos e 182 marcadores
## ------------------------------------------------------------------------------------------------------
library(genalg)
library(MASS)
## ------------------------------------------------------------------------------------------------------
## 1 - ROTINA DE LEITURA
## ------------------------------------------------------------------------------------------------------
## 1.1 - ROTINA PARA LEITURA DO VETOR TRAÇOS
## ------------------------------------------------------------------------------------------------------
# traco: vetor que guarda os valores do traço quantitativo a ser estudado
traco<-read.table("e://programaatualizado//tracosbpsratos.txt")
tracot<-t(traco)
n<-length(tracot)

#n: numero de individuos existentes na amostra
n
dim(traco)
## ------------------------------------------------------------------------------------------------------
## 1.2 - ROTINA PARA LEITURA DA MATRIZ DE MARCADORES
## ------------------------------------------------------------------------------------------------------
marcadores<-read.table("e://programaatualizado//marcadoresratos.txt")
# marcadores: marcadores com dados completos
attach(marcadores)
dim(marcadores)

## ------------------------------------------------------------------------------------------------------
## 1.3 - ROTINA PARA LEITURA DO VETOR DE POSIÇÕES DO GENOMA COM RESPOSTA BINÁRIA,
##         SE EXISTE OU NÃO MARCADOR EM CADA POSIÇÃO
## ------------------------------------------------------------------------------------------------------
# existemarcador:vetor cuja posicao recebe 1 quando tem marcador e 0 caso contrário
existemarcador<-read.table("e://programaatualizado//existemarcadorratos.txt")
dim(existemarcador)

## ------------------------------------------------------------------------------------------------------
## 1.4 - ROTINA PARA LEITURA DA MATRIZ COMPRIMENTO DE CADA CROMOSSOMO E NUMERO DE
MARCADORES
## ------------------------------------------------------------------------------------------------------

# chrmarc: matriz que guarda o comprimento de cada cromossomo em cM e o numero de marcadores em cada um deles.
chrmarc<-read.table("e://programaatualizado//chrmarcratos.txt")
dim(chrmarc)

# compgenoma: comprimento do genoma completo para cada cromossomo
compgenoma<-sum(chrmarc[,1])+length(chrmarc[,1])
compgenoma
# compgenoma1,...,compgenoma10: comprimento de cada cromossomo em cM.
compgenoma1<-chrmarc[1,1]+1
compgenoma2<-chrmarc[2,1]+1
compgenoma3<-chrmarc[3,1]+1
compgenoma4<-chrmarc[4,1]+1
compgenoma5<-chrmarc[5,1]+1
compgenoma6<-chrmarc[6,1]+1
compgenoma7<-chrmarc[7,1]+1
compgenoma8<-chrmarc[8,1]+1
compgenoma9<-chrmarc[9,1]+1
compgenoma10<-chrmarc[10,1]+1
compgenoma11<-chrmarc[11,1]+1
compgenoma12<-chrmarc[12,1]+1
compgenoma13<-chrmarc[13,1]+1
```

```
compgenoma14<-chrmarc[14,1]+1
compgenoma15<-chrmarc[15,1]+1
compgenoma16<-chrmarc[16,1]+1
compgenoma17<-chrmarc[17,1]+1
compgenoma18<-chrmarc[18,1]+1
compgenoma19<-chrmarc[19,1]+1
compgenoma20<-chrmarc[20,1]+1
compgenoma21<-chrmarc[21,1]+1

nmarcadorescromossomo1<-chrmarc[1,2]
nmarcadorescromossomo2<-chrmarc[2,2]
nmarcadorescromossomo3<-chrmarc[3,2]
nmarcadorescromossomo4<-chrmarc[4,2]
nmarcadorescromossomo5<-chrmarc[5,2]
nmarcadorescromossomo6<-chrmarc[6,2]
nmarcadorescromossomo7<-chrmarc[7,2]
nmarcadorescromossomo8<-chrmarc[8,2]
nmarcadorescromossomo9<-chrmarc[9,2]
nmarcadorescromossomo10<-chrmarc[10,2]
nmarcadorescromossomo11<-chrmarc[11,2]
nmarcadorescromossomo12<-chrmarc[12,2]
nmarcadorescromossomo13<-chrmarc[13,2]
nmarcadorescromossomo14<-chrmarc[14,2]
nmarcadorescromossomo15<-chrmarc[15,2]
nmarcadorescromossomo16<-chrmarc[16,2]
nmarcadorescromossomo17<-chrmarc[17,2]
nmarcadorescromossomo18<-chrmarc[18,2]
nmarcadorescromossomo19<-chrmarc[19,2]
nmarcadorescromossomo20<-chrmarc[20,2]
nmarcadorescromossomo21<-chrmarc[21,2]
## ------------------------------------------------------------------------------------------------
## 2 - ROTINA QUE GUARDA MATRIZ COM POSIÇÕES DO GENOMA, CROMOSSOMO E POSIÇÕES NO CROMOSSOMO
## ------------------------------------------------------------------------------------------------

# chrpos: posicao no genoma de 1 em 1 cM, cromossomo e posição no cromossomo
chrpos<-matrix(data = 0, nrow = compgenoma, ncol = 3)
dim(chrpos)
i<-1
while (i<=compgenoma){
chrpos[i,1]<-i
i<-i+1
}
chr1<-rep(1,chrmarc[1,1]+1)
chr2<-rep(2,chrmarc[2,1]+1)
chr3<-rep(3,chrmarc[3,1]+1)
chr4<-rep(4,chrmarc[4,1]+1)
chr5<-rep(5,chrmarc[5,1]+1)
chr6<-rep(6,chrmarc[6,1]+1)
chr7<-rep(7,chrmarc[7,1]+1)
chr8<-rep(8,chrmarc[8,1]+1)
chr9<-rep(9,chrmarc[9,1]+1)
chr10<-rep(10,chrmarc[10,1]+1)
chr11<-rep(11,chrmarc[11,1]+1)
chr12<-rep(12,chrmarc[12,1]+1)
chr13<-rep(13,chrmarc[13,1]+1)
chr14<-rep(14,chrmarc[14,1]+1)
chr15<-rep(15,chrmarc[15,1]+1)
chr16<-rep(16,chrmarc[16,1]+1)
chr17<-rep(17,chrmarc[17,1]+1)
```

```
chr18<-rep(18,chrmarc[18,1]+1)
chr19<-rep(19,chrmarc[19,1]+1)
chr20<-rep(20,chrmarc[20,1]+1)
chr21<-rep(21,chrmarc[21,1]+1)

chrpos[,2]<-cbind(t(chr1),t(chr2),t(chr3),t(chr4),t(chr5),t(chr6),t(chr7),t(chr8),t(chr9),t(chr10),t(chr11),t(chr12),t(chr13),t(
chr14),t(chr15),t(chr16),t(chr17),t(chr18),t(chr19),t(chr20),t(chr21))
chrpos
chr1a<-c(1:(chrmarc[1,1]+1))
chr2a<-c(1:(chrmarc[2,1]+1))
chr3a<-c(1:(chrmarc[3,1]+1))
chr4a<-c(1:(chrmarc[4,1]+1))
chr5a<-c(1:(chrmarc[5,1]+1))
chr6a<-c(1:(chrmarc[6,1]+1))
chr7a<-c(1:(chrmarc[7,1]+1))
chr8a<-c(1:(chrmarc[8,1]+1))
chr9a<-c(1:(chrmarc[9,1]+1))
chr10a<-c(1:(chrmarc[10,1]+1))
chr11a<-c(1:(chrmarc[11,1]+1))
chr12a<-c(1:(chrmarc[12,1]+1))
chr13a<-c(1:(chrmarc[13,1]+1))
chr14a<-c(1:(chrmarc[14,1]+1))
chr15a<-c(1:(chrmarc[15,1]+1))
chr16a<-c(1:(chrmarc[16,1]+1))
chr17a<-c(1:(chrmarc[17,1]+1))
chr18a<-c(1:(chrmarc[18,1]+1))
chr19a<-c(1:(chrmarc[19,1]+1))
chr20a<-c(1:(chrmarc[20,1]+1))
chr21a<-c(1:(chrmarc[21,1]+1))
chrpos[,3]<-cbind(t(chr1a),t(chr2a),t(chr3a),t(chr4a),t(chr5a),t(chr6a),t(chr7a),t(chr8a),t(chr9a),t(chr10a),t(chr11a),t(chr12
a),t(chr13a),t(chr14a),t(chr15a),t(chr16a),t(chr17a),t(chr18a),t(chr19a),t(chr20a),t(chr21a))
chrpos
chrposem<-cbind(chrpos,existemarcador)
vetoralg<-chrposem
i<-1
contmarcadorcromo<-1
posicao<-1
## ------------------------------------------------------------------------------------------------
## 3 - ROTINA QUE GUARDA ALÉM DA MATRIZ DO ITEM 2, GUARDA TAMBÉM
##     O VETOR EXISTENCIA DE MARCADOR EM CADA POSIÇÃO DO GENOMA
## ------------------------------------------------------------------------------------------------

vetorposicaomarcador<-matrix(data = 0, nrow = contmarcadorcromo, ncol = 1)
while (i<=compgenoma) {
if(vetoralg[i,4] == 1) {
vetorposicaomarcador[contmarcadorcromo]<-posicao
contmarcadorcromo<-contmarcadorcromo+1
}
posicao<-posicao+1
i<-i+1
}

print(vetorposicaomarcador)

print(contmarcadorcromo)

print(vetoralg)

## ------------------------------------------------------------------------------------------------
## 4 - ROTINA QUE GUARDA POSIÇÕES NO GENOMA, CROMOSSOMO,
```

```
## POSIÇÃO NO CROMOSSOMO E EXISTENCIA DE MARCADOR.
## ------------------------------------------------------------------------------------------------------

# mcp: guarda as posições dos cromossomos e marcadores selecionados
mcp<-matrix(data = 0, nrow = 1, ncol = 8)
val1<-matrix(data = 0, nrow = 1, ncol = 4)
val2<-matrix(data = 0, nrow = 1, ncol = 4)
pos1<-1
pos2<-2
contexec<-1
while (abs(pos1-pos2)<=9) {

pos1<-round(runif(1,0.51,compgenoma + 0.49))
pos2<-round(runif(1,0.51,compgenoma + 0.49))
if (pos2 < pos1)
{
auxiliar<-pos1
pos1<-pos2
pos2<-auxiliar
}
val1<-chrposem[pos1,]
val2<-chrposem[pos2,]

contexec<-contexec+1
}

## ---------------------------------------------------------------------
## 5 - ROTINA DA FUNÇÃO OBJETIVA PARA RESÍDUO SSE
## ---------------------------------------------------------------------

cria.testes.aux.MRN<-function(pos1,pos2)
{

# X, X2:matrizes que guardam uma coluna de 1's, uma segunda coluna com valorres de Xa
# e uma outra com valores de Xd para o primeiro e segundo qtl's, respectivamente, considerando
# o modelo completo
# Xea, Xea2, Xed, Xed2: matriz que guarda o efeito aditivo.
# Xf, Xeaf, Xea2f, Xed2f e X2f: matrizes empilhamento
Xf<-matrix(data = 0, nrow = n, ncol = 3)
X<-matrix(data = 0, nrow = n, ncol = 3)
Xea<-matrix(data = 0, nrow = n, ncol = 2)
Xea2<-matrix(data = 0, nrow = n, ncol = 2)
Xeaf<-matrix(data = 0, nrow = n, ncol = 2)
Xea2f<-matrix(data = 0, nrow = n, ncol = 2)
X2f<-matrix(data = 0, nrow = n, ncol = 3)
X2<-matrix(data = 0, nrow = n, ncol = 3)

## ---------------------------------------------------------------------
## 5.1 - ROTINA PARA ESCOLHA DAS POSIÇÕES E CÁLCULOO DA DISTANCIA MAIS PRÓXIMA DO
MARCADOR PARA O PRIMEIRO QTL.
## ---------------------------------------------------------------------

# valor1: guarda o primeiro cromossomo e o primeiro loco dentro deste cromossomo
valor1<-chrposem[pos1,]

# valor2: guarda o segundo cromossomo e o segundo loco dentro deste cromossomo
valor2<-chrposem[pos2,]

# ------------------------------------------------------------------------------------
# mcp: guarda as posições dos cromossomos e marcadores selecionados
```

```
mcp<-cbind(pos1,valor1,pos2,valor2)

# rotina para localização do primeiro intervalo
# ------------------------------------------------------------------------------------

# se o arquivo de dados chegaram ao fim

contador1<-0
while ((mcp[1,5]==0) && (chrposem[pos1+contador1,4] == 0)  && (pos1+contador1 < compgenoma))
contador1<-contador1+1

ifelse(mcp[1,5]==1,contador1<-0,contador1)

contador2<-0
 while ((mcp[1,5]==0) && (pos1-contador2 > 0) && (chrposem[pos1+contador2,4] == 0)) contador2<-contador2-1

ifelse(mcp[1,5]==1,contador2<-0,contador2)

posicaodomarcador1<-1

if (abs(contador1) <= abs(contador2)) posicaodomarcador1<-pos1+contador1
if (abs(contador2) < abs(contador1)) posicaodomarcador1<-pos1+contador2

## ------------------------------------------------------------------
## 5.2 - ROTINA PARA ESCOLHA DAS POSIÇÕES E CÁLCULO DA DISTANCIA MAIS PRÓXIMA DO
MARCADOR PARA O SEGUNDO QTL.
## ------------------------------------------------------------------

# ------------------------------------------------------------------------------------
# rotina para localização do segundo intervalo

contador1a<-0
while ((mcp[1,10]==0) && (chrposem[pos2+contador1a,4] == 0) && (pos2+contador1a < compgenoma))
contador1a<-contador1a+1
ifelse(mcp[1,10] == 1,contador1a<-0,contador1a)

contador2a<-0
while ((mcp[1,10]==0) && (pos2-contador2a > 0)  &&  (chrposem[pos2+contador2a,4] == 0)) contador2a<-contador2a-1

ifelse(mcp[1,10]==1,contador2a<-0,contador2a)

posicaodomarcador2<-1
if (abs(contador1a) <= abs(contador2a)) posicaodomarcador2<-pos2+contador1a
if (abs(contador2a) < abs(contador1a)) posicaodomarcador2<-pos2+contador2a

contmarcador1<-0
i<-1
while (i <= posicaodomarcador1){
if (chrposem[i,4] == 1)
contmarcador1<-contmarcador1+1
i<-i+1
}
contmarcador1
contmarcador2<-0
i<-1
while (i <= posicaodomarcador2){
if (chrposem[i,4] == 1)
contmarcador2<-contmarcador2+1
i<-i+1
```

```
}
contmarcador2
marcador1<-marcadores[,contmarcador1]
marcador2<-marcadores[,contmarcador2]
marcador1<-as.matrix(marcador1)
dim(marcador1)
marcador2<-as.matrix(marcador2)
dim(marcador2)

matrizmarcadoresselecionados<-cbind(traco,marcador1,marcador2)
dados<-matrizmarcadoresselecionados

r<-abs(contador1 - contador2)*0.01
if (r == 0) r<-0.01
npassos<-r/0.01
r1 <- 0
j<-1

tabela<-matrix(data = 0, nrow = npassos, ncol = 14)
tabelaea<-matrix(data = 0, nrow = npassos, ncol = 12)
while (j<=npassos) {

## ------------------------------------------------------------------------
## 5.3 - ROTINA PARA CÁLCULO DAS MATRIZES DE PLANEJAMENTO X
##       E DOS AJUSTES PARA O PRIMEIRO QTL PERCORRENDO O INTERVALO
##       CALCULADO EM 5.1 DE 1 EM 1 cM.
## ------------------------------------------------------------------------

# modelo (Haley and Knott, 1992)
# r é a frequencia de recombinação entre os marcadores
# r1 é a frequencia de recombinação entre o marcador M1 e o QTL
# r2 é a frequencia de recombinação entre o marcador M2 e o QTL

if ((1-2*r1 == 0)||((1-r)^2==0)||(r*(1-r)==0)||(r^2==0)||(r^2+(1-r)^2==0)) r1<-r1+0.01

r2<-(r-r1)/(1-2*r1)
r2


# probabilidades condicionais associadas aos genótipos (pcag) de um possível QTL no
# intervalo entre os marcadores M1 e M2 em uma população F2, considerando
# interferência nula
pcag<-matrix(data = 0, nrow = 9, ncol = 3)
pcag[1,1]<-((1-r1)^2*(1-r2)^2)/(1-r)^2
pcag[1,2]<-(2*r1*(1-r1)*r2*(1-r2))/(1-r)^2
pcag[1,3]<-(r1^2*r2^2)/(1-r)^2
pcag[2,1]<-(2*(1-r1)^2*r2*(1-r2))/(2*r*(1-r))
pcag[2,2]<-(2*r1*(1-r1)*(r2^2+(1-r2)^2))/(2*r*(1-r))
pcag[2,3]<-(2*r1^2*r2*(1-r2))/(2*r*(1-r))
pcag[3,1]<-((1-r1)^2*r2^2)/r^2
pcag[3,2]<-(2*r1*(1-r1)*r2*(1-r2))/r^2
pcag[3,3]<-(r1^2*(1-r1)^2)/r^2
pcag[4,1]<-(2*r1*(1-r1)*(1-r2)^2)/(2*r*(1-r))
pcag[4,2]<-(2*(r1^2+(1-r1)^2)*r2*(1-r2))/(2*r*(1-r))
pcag[4,3]<-(2*r1*(1-r1)*r2^2)/(2*r*(1-r))
pcag[5,1]<-(4*r1*(1-r1)*r2*(1-r2))/(2*(r^2+(1-r)^2))
pcag[5,2]<-(2*(r1^2+(1-r1)^2)*(r2^2+(1-r2)^2))/(2*(r^2+(1-r)^2))
pcag[5,3]<- (4*r1*(1-r1)*r2*(1-r2))/(2*(r^2+(1-r)^2))
pcag[6,1]<-(2*r1*(1-r1)*r2^2)/(2*r*(1-r))
pcag[6,2]<-(2*(r1^2+(1-r1)^2)*r2*(1-r2))/(2*r*(1-r))
```

```
pcag[6,3]<-(2*r1*(1-r1)*(1-r2)^2)/(2*r*(1-r))
pcag[7,1]<-(r1^2*(1-r2)^2)/r^2
pcag[7,2]<-(2*r1*(1-r1)*r2*(1-r2))/r^2
pcag[7,3]<-((1-r1)^2*r2^2)/r^2
pcag[8,1]<-(2*r1^2*r2*(1-r2))/(2*r*(1-r))
pcag[8,2]<-(2*r1*(1-r1)*(r2^2+(1-r2)^2))/(2*r*(1-r))
pcag[8,3]<-(2*(1-r1)^2*r2*(1-r2))/(2*r*(1-r))
pcag[9,1]<-(r1^2*r2^2)/(1-r)^2
pcag[9,2]<-(2*r1*(1-r1)*r2*(1-r2))/(1-r)^2
pcag[9,3]<-((1-r1)^2*(1-r2)^2)/(1-r)^2

# matriz de valores de Xa e xd
XaXd<-matrix(data = 0, nrow = 9, ncol = 2)
XaXd[1,1]<-pcag[1,1] - pcag[1,3]
XaXd[1,2]<-pcag[1,2]
XaXd[2,1]<-pcag[2,1] - pcag[2,3]
XaXd[2,2]<-pcag[2,2]
XaXd[3,1]<-pcag[3,1] - pcag[3,3]
XaXd[3,2]<-pcag[3,2]
XaXd[4,1]<-pcag[4,1] - pcag[4,3]
XaXd[4,2]<-pcag[4,2]
XaXd[5,1]<-pcag[5,1] - pcag[5,3]
XaXd[5,2]<-pcag[5,2]
XaXd[6,1]<-pcag[6,1] - pcag[6,3]
XaXd[6,2]<-pcag[6,2]
XaXd[7,1]<-pcag[7,1] - pcag[7,3]
XaXd[7,2]<-pcag[7,2]
XaXd[8,1]<-pcag[8,1] - pcag[8,3]
XaXd[8,2]<-pcag[8,2]
XaXd[9,1]<-pcag[9,1] - pcag[9,3]
XaXd[9,2]<-pcag[9,2]
XaXd

# montando a matriz de planejamento X para testar o modelo completo (yj = mi + aXj+ dZj+epsilon)
# contra o modelo reduzido (yj = mi +epsilon).
# Trata-se do teste H3/H0, cujo efeito signifitivo indica presenca de efeito aditivo e/ou de dominancia.

# início da rotina de dados completos para QTL 1.
i<-1

# montando a matriz de planejamento X
while (i<=n) {
X[i,1]<-1
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 2) {
X[i,2]<-XaXd[1,1]
X[i,3]<-XaXd[1,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 0) {
```

```
if(dados[i,2] == 0) {
X[i,2]<-XaXd[9,1]
X[i,3]<-XaXd[9,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 1) {
X[i,2]<-XaXd[4,1]
X[i,3]<-XaXd[4,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 2) {
X[i,2]<-XaXd[2,1]
X[i,3]<-XaXd[2,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 0) {
X[i,2]<-XaXd[7,1]
X[i,3]<-XaXd[7,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 1) {
X[i,2]<-XaXd[5,1]
X[i,3]<-XaXd[5,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 2) {
X[i,2]<-XaXd[3,1]
X[i,3]<-XaXd[3,2]
}
}
```

```
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 0) {
X[i,2]<-XaXd[8,1]
X[i,3]<-XaXd[8,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 1) {
X[i,2]<-XaXd[6,1]
X[i,3]<-XaXd[6,2]
}
}
i<-i+1
}

# fim da rotina de dados completos para QTL 1.

# início da rotina de dados incompletos para QTL 1.
i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 1) {
X[i,2]<-(XaXd[4,1]+2*XaXd[5,1]+XaXd[6,1])/4
X[i,3]<-(XaXd[4,2]+2*XaXd[5,2]+XaXd[6,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == -1) {
X[i,2]<-(XaXd[2,1]+2*XaXd[5,1]+XaXd[8,1])/4
X[i,3]<-(XaXd[2,2]+2*XaXd[5,2]+XaXd[8,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 0) {
X[i,2]<-(XaXd[7,1]+2*XaXd[8,1]+XaXd[9,1])/4
X[i,3]<-(XaXd[7,2]+2*XaXd[8,2]+XaXd[9,2])/4
}
}
```

```
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == -1) {
X[i,2]<-(XaXd[3,1]+2*XaXd[6,1]+XaXd[9,1])/4
X[i,3]<-(XaXd[3,2]+2*XaXd[6,2]+XaXd[9,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 10) {
X[i,2]<-(2*XaXd[4,1]+4*XaXd[5,1]+2*XaXd[6,1]+XaXd[7,1]+2*XaXd[8,1]+XaXd[9,1])/12
X[i,3]<-(2*XaXd[4,2]+4*XaXd[5,2]+2*XaXd[6,2]+XaXd[7,2]+2*XaXd[8,2]+XaXd[9,2])/12
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == -1) {
X[i,2]<-(2*XaXd[2,1]+XaXd[3,1]+4*XaXd[5,1]+2*XaXd[6,1]+2*XaXd[8,1]+XaXd[9,1])/12
X[i,3]<-(2*XaXd[2,2]+XaXd[3,2]+4*XaXd[5,2]+2*XaXd[6,2]+2*XaXd[8,2]+XaXd[9,2])/12
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 12) {
X[i,2]<-(XaXd[1,1]+2*XaXd[2,1]+XaXd[3,1]+2*XaXd[4,1]+4*XaXd[5,1]+2*XaXd[6,1])/12
X[i,3]<-(XaXd[1,2]+2*XaXd[2,2]+XaXd[3,2]+2*XaXd[4,2]+4*XaXd[5,2]+2*XaXd[6,2])/12
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == -1) {
X[i,2]<-(XaXd[1,1]+2*XaXd[2,1]+2*XaXd[4,1]+4*XaXd[5,1]+XaXd[7,1]+2*XaXd[8,1])/12
X[i,3]<-(XaXd[1,2]+2*XaXd[2,2]+2*XaXd[4,2]+4*XaXd[5,2]+XaXd[7,2]+2*XaXd[8,2])/12
}
}
```

```
i<-i+1
}



i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 0) {
X[i,2]<-(2*XaXd[8,1]+XaXd[9,1])/3
X[i,3]<-(2*XaXd[8,2]+XaXd[9,2])/3
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 10) {
X[i,2]<-(2*XaXd[6,1]+XaXd[9,1])/3
X[i,3]<-(2*XaXd[6,2]+XaXd[9,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 1) {
X[i,2]<-(4*XaXd[5,1]+2*XaXd[4,1])/6
X[i,3]<-(4*XaXd[5,2]+2*XaXd[4,2])/6
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 10) {
X[i,2]<-(4*XaXd[5,1]+2*XaXd[8,1])/6
X[i,3]<-(4*XaXd[5,2]+2*XaXd[8,2])/6
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 2) {
X[i,2]<-(2*XaXd[2,1]+XaXd[3,1])/3
X[i,3]<-(2*XaXd[2,2]+XaXd[3,2])/3
}
}
```

```
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 10) {
X[i,2]<-(2*XaXd[4,1]+XaXd[7,1])/3
X[i,3]<-(2*XaXd[4,2]+XaXd[7,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 0) {
X[i,2]<-(XaXd[7,1]+2*XaXd[8,1])/3
X[i,3]<-(XaXd[7,2]+2*XaXd[8,2])/3

}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 12) {
X[i,2]<-(XaXd[3,1]+2*XaXd[6,1])/3
X[i,3]<-(XaXd[3,2]+2*XaXd[6,2])/3
}
}
i<-i+1
}



i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 1) {
X[i,2]<-(4*XaXd[5,1]+2*XaXd[6,1])/6
X[i,3]<-(4*XaXd[5,2]+2*XaXd[6,2])/6
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 12) {
X[i,2]<-(2*XaXd[2,1]+4*XaXd[5,1])/6
X[i,3]<-(2*XaXd[2,2]+4*XaXd[5,2])/6
```

```
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 2) {
X[i,2]<-(XaXd[1,1]+2*XaXd[2,1])/3
X[i,3]<-(XaXd[1,2]+2*XaXd[2,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 12) {
X[i,2]<-(XaXd[1,1]+2*XaXd[4,1])/3
X[i,3]<-(XaXd[1,2]+2*XaXd[4,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 10) {
X[i,2]<-(2*XaXd[4,1]+4*XaXd[5,1]+XaXd[7,1]+2*XaXd[8,1])/9
X[i,3]<-(2*XaXd[4,2]+4*XaXd[5,2]+XaXd[7,2]+2*XaXd[8,1])/9
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 12) {
X[i,2]<-(2*XaXd[2,1]+4*XaXd[5,1]+XaXd[6,1]+2*XaXd[3,1])/9
X[i,3]<-(2*XaXd[2,2]+4*XaXd[5,2]+XaXd[6,2]+2*XaXd[3,1])/9
}
}
i<-i+1
}
# início da rotina de dados incompletos para QTL 2.

i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == -1) {
X[i,2]<-(XaXd[1,1]+2*XaXd[2,1]+XaXd[3,1]+2*XaXd[4,1]+4*XaXd[5,1]+2*XaXd[6,1]+XaXd[7,1]+2*XaXd[8,1]+
XaXd[9,1])/16
X[i,3]<-(XaXd[1,2]+2*XaXd[2,2]+XaXd[3,2]+2*XaXd[4,2]+4*XaXd[5,2]+2*XaXd[6,2]+XaXd[7,2]+2*XaXd[8,2]+
```

```
XaXd[9,2])/16
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 10) {
X[i,2]<-(4*XaXd[5,1]+2*XaXd[6,1]+2*XaXd[8,1]+XaXd[9,1])/9
X[i,3]<-(4*XaXd[5,2]+2*XaXd[6,2]+2*XaXd[8,2]+XaXd[9,2])/9
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 12) {
X[i,2]<-(XaXd[1,1]+2*XaXd[2,1]+2*XaXd[4,1]+4*XaXd[5,1])/9
X[i,3]<-(XaXd[1,2]+2*XaXd[2,2]+2*XaXd[4,2]+4*XaXd[5,2])/9
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 2) {
X[i,2]<-(XaXd[1,1]+2*XaXd[2,1]+XaXd[3,1])/4
X[i,3]<-(XaXd[1,1]+2*XaXd[2,2]+XaXd[3,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == -1) {
X[i,2]<-(XaXd[1,1]+2*XaXd[4,1]+XaXd[7,1])/4
X[i,3]<-(XaXd[1,1]+2*XaXd[4,2]+XaXd[7,2])/4
}
}
i<-i+1
}


# calculo do vetor betachapeu para os parâmetros mi, a e d.
# Y vetor que armazena a coluna que contem os valores para o traço em estudo
Y<-dados[,1]
```

```
# H1/H0: testa a presença do efeito aditivo (H1: a diferente de zero, d = 0 versusu H0: a = 0, d = 0)
Xea<-cbind(X[,1],X[,2])
determ<-determinant(t(Xea)%*%Xea)
if (determ$modulus == 0) Xea[1,1]<-Xea[1,1]+0.01
betachapeuea<-qr.solve(t(Xea)%*%Xea,tol = 1e-1000)%*%t(Xea)%*%Y

# SQTNC: soma dos quadrados não corrigido

SQTNC<- t(Y)%*%Y

# SQTC: soma de quadrados corrigido
SQTC<-SQTNC - sum(Y)^2/n

# SQPea : soma de quadrados de parâmetros
SQPea <- t(betachapeuea)%*%t(Xea)%*%Y

# SQRegea: soma de quadrados da regressão para o modelo aditivo
SQRegea <- t(betachapeuea)%*%t(Xea)%*%Y - sum(Y)^2/n

# SQDr: soma de quadrados dos desvios para o modelo reduzido
SQDr <- SQTC

# SQDcea: soma de quadrados dos desvios para o modelo aditivo
SQDcea <- t(Y)%*%Y - t(betachapeuea)%*%t(Xea)%*%Y

# CDNCea: coeficiente de determinação não corrigido para o modelo aditivo
CDNCea <- (t(betachapeuea)%*%t(Xea)%*%Y)/(t(Y)%*%Y)

# CDCea: coeficiente de determinação corrigido o modelo aditivo
CDCea <- (t(betachapeuea)%*%t(Xea)%*%Y - sum(Y)^2/n)/(t(Y)%*%Y - sum(Y)^2/n)

# LRq1ea: estatística razão de verossimilhanças pelo coeficiente de determinação corrigido
# Esse teste é calculado considerando o modelo aditivo e o modelo reduzido
# (modelo que inclui apenas a média).

LRq1ea <- n*log(SQTC/SQDcea)

R2ea<- SQRegea/SQTC
nsLRq1ea<-1-pchisq(LRq1ea,df=2)
# LODea: estatística Lod score
LODea <- -n*log10(1-CDCea)/2

R2NCea<-(t(betachapeuea)%*%t(Xea)%*%Y - sum(Y)^2/n)/((t(Y)%*%Y- sum(Y)^2/n))
p<-length(Xea[1,])
AICea<--2*log(LRq1ea)+2*p
BICea<--2*log(LRq1ea)-p*log(n)
tabelaea[j,1]<- r1
tabelaea[j,2]<-betachapeuea[1,1]
tabelaea[j,3]<-betachapeuea[2,1]
tabelaea[j,4]<-betachapeuea[2,1]/betachapeuea[1,1]
tabelaea[j,5]<-CDCea
tabelaea[j,6]<-R2ea
tabelaea[j,7]<-LRq1ea
tabelaea[j,8]<-nsLRq1ea
tabelaea[j,9]<-LODea
tabelaea[j,10]<-AICea
tabelaea[j,11]<-BICea
tabelaea[j,12]<-SQDcea

j<-j+1
```

```
r1<-r1 + 0.01
Xf<-cbind(Xf,X)
Xeaf<-cbind(Xeaf,Xea)
}

Xf<-Xf[,-1]
Xf<-Xf[,-1]
Xf<-Xf[,-1]

Xeaf<-Xeaf[,-1]
Xeaf<-Xeaf[,-1]

## ------------------------------------------------------------------------
## 5.4 - ROTINA PARA CÁLCULO DAS MATRIZES DE PLANEJAMENTO X
##       E DOS AJUSTES PARA O SEGUNDO QTL PERCORRENDO O INTERVALO
##       CALCULADO EM 5.2 DE 1 EM 1 cM.
## ------------------------------------------------------------------------

# rotina para o segundo qtl
ra<-abs(contador1a - contador2a)*0.01
if (ra == 0) ra<-0.01
npassos2<-ra/0.01
r1a <- 0
j2<-1
tabelaea2<-matrix(data = 0, nrow = npassos2, ncol = 12)
tabela2<-matrix(data = 0, nrow = npassos, ncol = 14)
while (j2<=npassos2) {

# rotina para a localizacao do segundo qtl no segundo intervalo
# modelo (Haley and Knott, 1992)
# ra é a frequencia de recombinação entre os marcadores
# r1a é a frequencia de recombinação entre o marcador M1 e o segundo QTL
# r2a é a frequencia de recombinação entre o marcador M2 e o segundo QTL

if ((1-2*r1a == 0)||((1-ra)^2==0)||(ra*(1-ra)==0)||(ra^2==0)||(ra^2+(1-ra)^2==0)) r1a<-r1a+0.01

r2a<-(ra-r1a)/(1-2*r1a)


# probabilidades condicionais associadas aos genótipos (pcag) de um possível segundo
# QTL no intervalo entre os marcadores M1 e M2 em uma população F2, considerando
# interferência nula
pcag2<-matrix(data = NA, nrow = 9, ncol = 3)
pcag2[1,1]<-((1-r1a)^2*(1-r2a)^2)/(1-ra)^2
pcag2[1,2]<-(2*r1a*(1-r1a)*r2a*(1-r2a))/(1-ra)^2
pcag2[1,3]<-(r1a^2*r2a^2)/(1-ra)^2
pcag2[2,1]<-(2*(1-r1a)^2*r2a*(1-r2a))/(2*ra*(1-ra))
pcag2[2,2]<-(2*r1a*(1-r1a)*(r2a^2+(1-r2a)^2))/(2*ra*(1-ra))
pcag2[2,3]<-(2*r1a^2*r2a*(1-r2a))/(2*ra*(1-ra))
pcag2[3,1]<-((1-r1a)^2*r2a^2)/ra^2
pcag2[3,2]<-(2*r1a*(1-r1a)*r2a*(1-r2a))/ra^2
pcag2[3,3]<-(r1a^2*(1-r1a)^2)/ra^2
pcag2[4,1]<-(2*r1a*(1-r1a)*(1-r2a)^2)/(2*ra*(1-ra))
pcag2[4,2]<-(2*(r1a^2+(1-r1a)^2)*r2a*(1-r2a))/(2*ra*(1-ra))
pcag2[4,3]<-(2*r1a*(1-r1a)*r2a^2)/(2*ra*(1-ra))
pcag2[5,1]<-(4*r1a*(1-r1a)*r2a*(1-r2a))/(2*(ra^2+(1-ra)^2))
pcag2[5,2]<-(2*(r1a^2+(1-r1a)^2)*(r2a^2+(1-r2a)^2))/(2*(ra^2+(1-ra)^2))
pcag2[5,3]<- (4*r1a*(1-r1a)*r2a*(1-r2a))/(2*(ra^2+(1-ra)^2))
pcag2[6,1]<-(2*r1a*(1-r1a)*r2a^2)/(2*ra*(1-ra))
```

```
pcag2[6,2]<-(2*(r1a^2+(1-r1a)^2)*r2a*(1-r2a))/(2*ra*(1-ra))
pcag2[6,3]<-(2*r1a*(1-r1a)*r2a^2)/(2*ra*(1-ra))
pcag2[7,1]<-(r1a^2*(1-r2a)^2)/ra^2
pcag2[7,2]<-(2*r1a*(1-r1a)*r2a*(1-r2a))/ra^2
pcag2[7,3]<-((1-r1a)^2*r2a^2)/ra^2
pcag2[8,1]<-(2*r1a^2*r2a*(1-r2a))/(2*ra*(1-ra))
pcag2[8,2]<-(2*r1a*(1-r1a)*(r2a^2+(1-r2a)^2))/(2*ra*(1-ra))
pcag2[8,3]<-(2*(1-r1a)^2*r2a*(1-r2a))/(2*ra*(1-ra))
pcag2[9,1]<-(r1a^2*r2a^2)/(1-ra)^2
pcag2[9,2]<-(2*r1a*(1-r1a)*r2a*(1-r2a))/(1-ra)^2
pcag2[9,3]<-((1-r1a)^2*(1-r2a)^2)/(1-ra)^2

# matriz de valores de Xa e xd
XaXd2<-matrix(data = NA, nrow = 9, ncol = 2)
XaXd2[1,1]<-pcag2[1,1] - pcag2[1,3]
XaXd2[1,2]<-pcag2[1,2]
XaXd2[2,1]<-pcag2[2,1] - pcag2[2,3]
XaXd2[2,2]<-pcag2[2,2]
XaXd2[3,1]<-pcag2[3,1] - pcag2[3,3]
XaXd2[3,2]<-pcag2[3,2]
XaXd2[4,1]<-pcag2[4,1] - pcag2[4,3]
XaXd2[4,2]<-pcag2[4,2]
XaXd2[5,1]<-pcag2[5,1] - pcag2[5,3]
XaXd2[5,2]<-pcag2[5,2]
XaXd2[6,1]<-pcag2[6,1] - pcag2[6,3]
XaXd2[6,2]<-pcag2[6,2]
XaXd2[7,1]<-pcag2[7,1] - pcag2[7,3]
XaXd2[7,2]<-pcag2[7,2]
XaXd2[8,1]<-pcag2[8,1] - pcag2[8,3]
XaXd2[8,2]<-pcag2[8,2]
XaXd2[9,1]<-pcag2[9,1] - pcag2[9,3]
XaXd2[9,2]<-pcag2[9,2]

# início da rotina de dados completos para QTL 2.

i<-1
# montando a matriz de planejamento X2

while (i<=n) {
X2[i,1]<-1
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 2) {
X2[i,2]<-XaXd2[1,1]
X2[i,3]<-XaXd2[1,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 0) {
X2[i,2]<-XaXd2[7,1]
X2[i,3]<-XaXd2[7,2]
```

```
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 0) {
X2[i,2]<-XaXd2[9,1]
X2[i,3]<-XaXd2[9,2]
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 1) {
X2[i,2]<-XaXd2[4,1]
X2[i,3]<-XaXd2[4,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 2) {
X2[i,2]<-XaXd2[2,1]
X2[i,3]<-XaXd2[2,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 0) {
X2[i,2]<-XaXd2[8,1]
X2[i,3]<-XaXd2[8,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 1) {
X2[i,2]<-XaXd2[5,1]
X2[i,3]<-XaXd2[5,2]
}
}
i<-i+1
}
```

```
i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 2) {
X2[i,2]<-XaXd2[3,1]
X2[i,3]<-XaXd2[3,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 0) {
X2[i,2]<-XaXd2[7,1]
X2[i,3]<-XaXd2[7,2]
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 1) {
X2[i,2]<-XaXd2[6,1]
X2[i,3]<-XaXd2[6,2]
}
}
i<-i+1
}
# fim da rotina de dados completos para QTL 2.

# início da rotina de dados incompletos para QTL 2.

i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 1) {
X2[i,2]<-(XaXd2[4,1]+2*XaXd2[5,1]+XaXd2[6,1])/4
X2[i,3]<-(XaXd2[4,2]+2*XaXd2[5,2]+XaXd2[6,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == -1) {
X2[i,2]<-(XaXd2[2,1]+2*XaXd2[5,1]+XaXd2[8,1])/4
X2[i,3]<-(XaXd2[2,2]+2*XaXd2[5,2]+XaXd2[8,2])/4
}
}
i<-i+1
}
```

```
i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 0) {
X2[i,2]<-(XaXd2[7,1]+2*XaXd2[8,1]+XaXd2[9,1])/4
X2[i,3]<-(XaXd2[7,2]+2*XaXd2[8,2]+XaXd2[9,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == -1) {
X2[i,2]<-(XaXd2[3,1]+2*XaXd2[6,1]+XaXd2[9,1])/4
X2[i,3]<-(XaXd2[3,2]+2*XaXd2[6,2]+XaXd2[9,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 10) {
X2[i,2]<-(2*XaXd2[4,1]+4*XaXd2[5,1]+2*XaXd2[6,1]+XaXd2[7,1]+2*XaXd2[8,1]+XaXd2[9,1])/12
X2[i,3]<-(2*XaXd2[4,2]+4*XaXd2[5,2]+2*XaXd2[6,2]+XaXd2[7,2]+2*XaXd2[8,2]+XaXd2[9,2])/12
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == -1) {
X2[i,2]<-(2*XaXd2[2,1]+XaXd2[3,1]+4*XaXd2[5,1]+2*XaXd2[6,1]+2*XaXd2[8,1]+XaXd2[9,1])/12
X2[i,3]<-(2*XaXd2[2,2]+XaXd2[3,2]+4*XaXd2[5,2]+2*XaXd2[6,2]+2*XaXd2[8,2]+XaXd2[9,2])/12
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 12) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[2,1]+XaXd2[3,1]+2*XaXd2[4,1]+4*XaXd2[5,1]+2*XaXd2[6,1])/12
X2[i,3]<-(XaXd2[1,2]+2*XaXd2[2,2]+XaXd2[3,2]+2*XaXd2[4,2]+4*XaXd2[5,2]+2*XaXd2[6,2])/12
}
}
i<-i+1
}
```

```
i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == -1) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[2,1]+2*XaXd2[4,1]+4*XaXd2[5,1]+XaXd2[7,1]+2*XaXd2[8,1])/12
X2[i,3]<-(XaXd2[1,2]+2*XaXd2[2,2]+2*XaXd2[4,2]+4*XaXd2[5,2]+XaXd2[7,2]+2*XaXd2[8,2])/12
}
}
i<-i+1
}



i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 0) {
X2[i,2]<-(2*XaXd2[8,1]+XaXd2[9,1])/3
X2[i,3]<-(2*XaXd2[8,2]+XaXd2[9,2])/3
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 10) {
X2[i,2]<-(2*XaXd2[6,1]+XaXd2[9,1])/3
X2[i,3]<-(2*XaXd2[6,2]+XaXd2[9,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 1) {
X2[i,2]<-(4*XaXd2[5,1]+2*XaXd2[4,1])/6
X2[i,3]<-(4*XaXd2[5,2]+2*XaXd2[4,2])/6
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 10) {
X2[i,2]<-(4*XaXd2[5,1]+2*XaXd2[8,1])/6
X2[i,3]<-(4*XaXd2[5,2]+2*XaXd2[8,2])/6
}
}
i<-i+1
}
```

```
i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 2) {
X2[i,2]<-(2*XaXd2[2,1]+XaXd2[3,1])/3
X2[i,3]<-(2*XaXd2[2,2]+XaXd2[3,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 10) {
X2[i,2]<-(2*XaXd2[4,1]+XaXd2[7,1])/3
X2[i,3]<-(2*XaXd2[4,2]+XaXd2[7,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 0) {
X2[i,2]<-(XaXd2[7,1]+2*XaXd2[8,1])/3
X2[i,3]<-(XaXd2[7,2]+2*XaXd2[8,2])/3

}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 0) {
if(dados[i,2] == 12) {
X2[i,2]<-(XaXd2[3,1]+2*XaXd2[6,1])/3
X2[i,3]<-(XaXd2[3,2]+2*XaXd2[6,2])/3
}
}
i<-i+1
}



i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 1) {
X2[i,2]<-(4*XaXd2[5,1]+2*XaXd2[6,1])/6
X2[i,3]<-(4*XaXd2[5,2]+2*XaXd2[6,2])/6
}
}
i<-i+1
```

```
}


i<-1
while (i<=n) {
if(dados[i,3] == 1) {
if(dados[i,2] == 12) {
X2[i,2]<-(2*XaXd2[2,1]+4*XaXd2[5,1])/6
X2[i,3]<-(2*XaXd2[2,2]+4*XaXd2[5,2])/6
}
}
i<-i+1
}

i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 2) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[2,1])/3
X2[i,3]<-(XaXd2[1,2]+2*XaXd2[2,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == 12) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[4,1])/3
X2[i,3]<-(XaXd2[1,2]+2*XaXd2[4,2])/3
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 10) {
X2[i,2]<-(2*XaXd2[4,1]+4*XaXd2[5,1]+XaXd2[7,1]+2*XaXd2[8,1])/9
X2[i,3]<-(2*XaXd2[4,2]+4*XaXd2[5,2]+XaXd2[7,2]+2*XaXd2[8,1])/9
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 12) {
X2[i,2]<-(2*XaXd2[2,1]+4*XaXd2[5,1]+XaXd2[6,1]+2*XaXd2[3,1])/9
X2[i,3]<-(2*XaXd2[2,2]+4*XaXd2[5,2]+XaXd2[6,2]+2*XaXd2[3,1])/9
}
}
i<-i+1
}
```

```
i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == -1) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[2,1]+XaXd2[3,1]+2*XaXd2[4,1]+4*XaXd2[5,1]+2*XaXd2[6,1]+XaXd2[7,1]+2*Xa
Xd2[8,1]+ XaXd2[9,1])/16
X2[i,3]<-(XaXd2[1,2]+2*XaXd2[2,2]+XaXd2[3,2]+2*XaXd2[4,2]+4*XaXd2[5,2]+2*XaXd2[6,2]+XaXd2[7,2]+2*Xa
Xd2[8,2]+ XaXd2[9,2])/16
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 10) {
if(dados[i,2] == 10) {
X[i,2]<-(4*XaXd2[5,1]+2*XaXd2[6,1]+2*XaXd2[8,1]+XaXd2[9,1])/9
X[i,3]<-(4*XaXd2[5,2]+2*XaXd2[6,2]+2*XaXd2[8,2]+XaXd2[9,2])/9
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 12) {
if(dados[i,2] == 12) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[2,1]+2*XaXd2[4,1]+4*XaXd2[5,1])/9
X2[i,3]<-(XaXd2[1,2]+2*XaXd2[2,2]+2*XaXd2[4,2]+4*XaXd2[5,2])/9
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == -1) {
if(dados[i,2] == 2) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[2,1]+XaXd2[3,1])/4
X2[i,3]<-(XaXd2[1,1]+2*XaXd2[2,2]+XaXd2[3,2])/4
}
}
i<-i+1
}


i<-1
while (i<=n) {
if(dados[i,3] == 2) {
if(dados[i,2] == -1) {
X2[i,2]<-(XaXd2[1,1]+2*XaXd2[4,1]+XaXd2[7,1])/4
X2[i,3]<-(XaXd2[1,1]+2*XaXd2[4,2]+XaXd2[7,2])/4
}
}
```

```
i<-i+1
}

# fim da rotina de dados incompletos para QTL 2.
X2f<-cbind(X2f,X2)
# Y vetor que armazena a coluna que contem os valores para o traço em estudo
Y<-dados[,1]


# H1/H0: testa a presença do efeito aditivo (H1: a diferente de zero, d = 0 versusu H0: a = 0, d = 0)

Xea2<-cbind(X2[,1],X2[,2])

Xea2f<-cbind(Xea2f,Xea2)
determ2<-determinant(t(Xea2)%*%Xea2)
if (determ2$modulus == 0) Xea2[1,1]<-Xea2[1,1]+0.01
betachapeuea2<-qr.solve(t(Xea2)%*%Xea2,tol = 1e-1000)%*%t(Xea2)%*%Y

# SQTNC: soma dos quadrados não corrigido

SQTNC<- t(Y)%*%Y

# SQTC: soma de quadrados corrigido
SQTC<-SQTNC - sum(Y)^2/n

# SQPea : soma de quadrados de parâmetros
SQPea2 <- t(betachapeuea2)%*%t(Xea2)%*%Y

# SQRegea2: soma de quadrados da regressão para o modelo aditivo
SQRegea2 <- t(betachapeuea2)%*%t(Xea2)%*%Y - sum(Y)^2/n

# SQDr: soma de quadrado do desvio para o modelo reduzido
SQDr <- SQTC

# SQDcea2: soma de quadrados do desvio para o modelo aditivo
SQDcea2 <- t(Y)%*%Y - t(betachapeuea2)%*%t(Xea2)%*%Y

# CDNCea2: coeficiente de determinação não corrigido para o modelo aditivo
CDNCea2 <- (t(betachapeuea2)%*%t(Xea2)%*%Y)/(t(Y)%*%Y)

# CDCea2: coeficiente de determinação corrigido o modelo aditivo
CDCea2 <- (t(betachapeuea2)%*%t(Xea2)%*%Y - sum(Y)^2/n)/(t(Y)%*%Y - sum(Y)^2/n)

# LRq1ea:2 estatística razão de verossimilhanças pelo coeficiente de determinaçxão corrigido
# Esse teste é calculado considerando o modelo aditivo e o modelo reduzido
# (modelo que inclui apenas a média).

LRq1ea2 <- n*log(SQTC/SQDcea2)
R2ea2<- SQRegea2/SQTC
nsLRq1ea2<-1-pchisq(LRq1ea2,df=2)
# LODea2: estatística Lod score
LODea2 <- -n*log10(1-CDCea2)/2

R2NCea2<-(t(betachapeuea2)%*%t(Xea2)%*%Y - sum(Y)^2/n)/((t(Y)%*%Y- sum(Y)^2/n))
p<-length(Xea2[1,])
AICea2<- -2*log(LRq1ea2)+2*p
BICea2<--2*log(LRq1ea2)-p*log(n)
tabelaea2[j2,1]<- r1a
tabelaea2[j2,2]<-betachapeuea2[1,1]
tabelaea2[j2,3]<-betachapeuea2[2,1]
```

```
tabelaea2[j2,4]<-betachapeuea2[2,1]/betachapeuea2[1,1]
tabelaea2[j2,5]<-CDCea2
tabelaea2[j2,6]<-R2ea2
tabelaea2[j2,7]<-LRq1ea2
tabelaea2[j2,8]<-nsLRq1ea2
tabelaea2[j2,9]<-LODea2
tabelaea2[j2,10]<-AICea2
tabelaea2[j2,11]<-BICea2
tabelaea2[j2,12]<-SQDcea2

j2<-j2+1
r1a<-r1a + 0.01


}

X2f<-X2f[,-1]
X2f<-X2f[,-1]
X2f<-X2f[,-1]

Xea2f<-Xea2f[,-1]
Xea2f<-Xea2f[,-1]

## ------------------------------------------------------------------------
## 5.5 - AJUSTE DO MODELO ADITIVO PARA DOIS QTLs E INTERAÇÃO.
## ------------------------------------------------------------------------

# modelo 2, modelo que considera todos os efeitos aditivos e interações

i<-1
Y<-as.matrix(Y)
tabelabeta2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 4)
poslinmat<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
# betaaux2<-matrix(data = 0, nrow = 4, ncol = 1)
rj<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
rk<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
SSE2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
# nsSSE2: nível de significancia para o SSE2
nsSSE2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
SSR2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
MSR2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
MSE2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
s2b<-matrix(data = 0, nrow =(length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
# nsF2 nivel de significancia do teste F no modelo com apenas efeitos aditivos
F2<-matrix(data = 0, nrow =(length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
nsF2<-matrix(data = 0, nrow =(length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
LR2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
nsLR2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
AIC2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
BIC2<-matrix(data = 0, nrow = (length(Xf[1,])/3)*(length(X2f[1,])/3), ncol = 1)
XfX2f2<-matrix(data = 0, nrow = n, ncol =4)
XfX2faux2<-matrix(data = 0, nrow = n, ncol = 4)
j<-0
while (j<length(Xf[1,])) {
k<-0
while (k<length(X2f[1,])) {
XfX2f2[,1]<-Xf[,1+j]*X2f[,1+k]
XfX2f2[,2]<-Xf[,1+j]*X2f[,2+k]
XfX2f2[,3]<-Xf[,2+j]*X2f[,1+k]
XfX2f2[,4]<-Xf[,2+j]*X2f[,2+k]
```

```
determinantes2<-det(t(XfX2f2)%*%XfX2f2,tol=1e-10000)
## determinantes2<-determinantes2$modulus
XfX2faux2<-rbind(XfX2faux2,XfX2f2)
if (abs(determinantes2) > 0){
## if (determinant(t(XfX2f2)%*%XfX2f2)==0) XfX2f2[1,1]<-XfX2f2[1,1]+0.01
beta2<-qr.solve(t(XfX2f2)%*%XfX2f2,tol = 1e-10000)%*%t(XfX2f2)%*%Y
## print("beta2")
tabelabeta2[i,1]<-beta2[1,1]
tabelabeta2[i,2]<-beta2[2,1]
tabelabeta2[i,3]<-beta2[3,1]
tabelabeta2[i,4]<-beta2[4,1]
rj[i,]<-j/3
rk[i,]<-k/3
SSR2[i,]<-t(beta2)%*%t(XfX2f2)%*%Y - (sum(Y))^2/n
p<-1
MSR2[i,]<-SSR2[i,]/(p-1)
SSE2[i,]<-abs(t(Y)%*%Y-t(beta2)%*%t(XfX2f2)%*%Y)
nsSSE2[i,]<-1-pchisq(SSE2[i,],df=n-p)
MSE2[i,]<-SSE2[i,]/(n-p)
inversa<-qr.solve(t(XfX2f2)%*%XfX2f2,tol = 1e-10000)
s2b[i,]<-(MSE2[i,]/inversa[4,4])
F2[i,]<-(beta2[4,1])^2/s2b[i,]
nsF2[i,]<-1-pf(F2[i,], df1= 1, df2 = n-p)
LR2[i,]<-n*log(1+p/(n-p-1)*F2[i,])
nsLR2[i,]<-1-pchisq(LR2[i,],df=1)
AIC2[i,]<--2*log(LR2[i,])+2*p
BIC2[i,]<--2*log(LR2[i,])-p*log(n)
poslinmat[i,]<-i
i<-i+1
}

k<-k+3}

j<-j+3}



resultados2<-cbind(poslinmat,rj,rk,SSE2,nsSSE2,MSE2,F2,nsF2,LR2,nsLR2,AIC2,BIC2)
resultados2<-resultados2[-length(resultados2[,1]),]
resultados2<-resultados2[-length(resultados2[,1]),]

## ------------------------------------------------------------------------
## 5.5.1 - AS POSIÇÕES SELECIONADAS ESTÃO EM CIMA DOS MARCADORES
## ------------------------------------------------------------------------
linha<-1
if (existemarcador[pos1,] == 1) if (existemarcador[pos2,] == 1)
{
vetorposicaomarcador<-as.matrix(vetorposicaomarcador)
aux<-matrix(1,n,1)
tm<-cbind(traco,marcadores)
contp1<-1
while (pos1 != vetorposicaomarcador[contp1,]) contp1<-contp1+1
## print("posicao")
## print(contp1)

contp2<-1
while (pos2 != vetorposicaomarcador[contp2,]) contp2<-contp2+1
## print("posicao")
## print(contp2)
```

```
XfX2f2<-cbind(aux,tm[,contp1+1],tm[,contp2+1],tm[,contp1+1]*tm[,contp2+1])
beta2<-qr.solve(t(XfX2f2)%*%XfX2f2,tol=1e-1000000)%*%t(XfX2f2)%*%Y
p<-1
SSR2<-t(beta2)%*%t(XfX2f2)%*%Y - (sum(Y))^2/n
 MSR2<-SSR2/(p-1)
SSE2<-abs(t(Y)%*%Y-t(beta2)%*%t(XfX2f2)%*%Y)
nsSSE2<-1-pchisq(SSE2,df=n-p)
MSE2<-SSE2/(n-p)
SSTO<-(t(XfX2f2)%*%XfX2f2)%*%t(XfX2f2) - (sum(Y))^2/n
inversa<-qr.solve(t(XfX2f2)%*%XfX2f2,tol=1e-1000000)
s2b<-MSE2*inversa[4,4]
F2<-(beta2[4,1])^2/s2b
nsF2<-1-pf(F2, df1= 1, df2 = n-p)
LR2<-n*log(1+p/(n-p-1)*F2)
nsLR2<-1-pchisq(LR2,df=1)
AIC2<--2*log(LR2)+2*p
BIC2<--2*log(LR2)-p*log(n)
resultados2<-cbind(linha,pos1,pos2,SSE2,nsSSE2,MSE2,F2,nsF2,LR2,nsLR2,AIC2,BIC2)
linha<-linha+1
}


matriz.aux<-list(mcp=mcp,dados=dados,tabelaea=tabelaea,tabelaea2=tabelaea2,resultados2=resultados2)
return(matriz.aux)
}




nsolucoes<-1000
 geracoes<-10000

todasatualizacoes<-matrix(data = 0, nrow = geracoes, ncol = nsolucoes)



solucoes<-matrix(0,nsolucoes,5)
ns<-1
while (ns<=nsolucoes)
{

## 6 - IMPLEMENTAÇÃO DO ALGORITMO GENÉTICO

## ---------------------------------------------------------------------
## 6.1 - ROTINA PARA MATRIZ DE INICIALIZAÇÃO/ AVALIAÇÃO DE QUATRO PARES
##       DE POSIÇÕES COM OS CÁLCULOS DOS RESPECTIVOS RESÍDUOS SSE PELA FUNÇÃO
##       OBJETIVO CALCULADA EM 5.
## ---------------------------------------------------------------------

cv<-1
aleatorizacao<-matrix(data = 0, nrow = 10,ncol =  5)
a<-matrix(0,2,1)
for(i in 1:10){
a1<-1
a2<-10
contler<-1
  while (abs(a1-a2)<=9) {
  a1<-sample(1:compgenoma, 1)
  a[1,1]<-a1
  a2<-sample(1:compgenoma, 1)
```

```
  a[2,1]<-a2
contler<-contler+1
  }
a<-as.matrix(a)
if (a[2,1] < a[1,1])
{
auxiliar<-a[1,1]
a[1,1]<-a[2,1]
a[2,1]<-auxiliar
}

  aleatorizacao[i,1]<-a[1,1]
  aleatorizacao[i,2]<-vetoralg[aleatorizacao[i,1],2]
  aleatorizacao[i,3]<-a[2,1]
  aleatorizacao[i,4]<-vetoralg[aleatorizacao[i,3],2]
  testes<-cria.testes.aux.MRN(aleatorizacao[i,1],aleatorizacao[i,3])
  SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
## if (BIC == Inf) BIC<-99999
  aleatorizacao[i,5]<-SSE
print(cv)
cv<-cv+1
}

## ------------------------------------------------------------------
## 6.2 - DECLARAÇÃO DAS MATRIZES UTILIZADAS NO ALGORITMO GENÉTICO
## ------------------------------------------------------------------

 atualizacao<-matrix(data = 99999999999, nrow = 1, ncol = 5)
SSE<-matrix(data = 0, nrow = 1, ncol = 5)
SSE2<-matrix(data = 0, nrow = 1, ncol = 5)
quadrorecombimutacao<-matrix(data = 0, nrow = 1, ncol = 5)
matrizalfabeta<-matrix(data=0, nrow = geracoes, ncol = 4)
 solucao<-matrix(data = 0, nrow = 1, ncol = 5)
 selecoes<-matrix(data = 0, nrow = geracoes, ncol = 11)
 selecoes2<-matrix(data = 0, nrow = geracoes, ncol = 6)
 recombinacoes<-matrix(data = 0, nrow = geracoes, ncol = 7)
atualizacoes<-matrix(data = 0, nrow = geracoes, ncol = 7)

 ger<-1

 while (ger <= geracoes)
 {
## ------------------------------------------------------------------
## 6.3 - ROTINA PARA FORMAÇÃO DA MATRIZ INICIALIZAÇÃO/AVALIAÇÃO
## ------------------------------------------------------------------

simulacoes<-4
inicializacaoavaliacao<-matrix(data=0,nrow=simulacoes,ncol=5)
simul<-1
contexec<-1
while (simul <= simulacoes)
{
if (aleatorizacao[contexec,2] != aleatorizacao[contexec,4]) {

inicializacaoavaliacao[simul,1]<-aleatorizacao[contexec,1]
inicializacaoavaliacao[simul,2]<-aleatorizacao[contexec,2]
inicializacaoavaliacao[simul,3]<-aleatorizacao[contexec,3]
inicializacaoavaliacao[simul,4]<-aleatorizacao[contexec,4]
inicializacaoavaliacao[simul,5]<-aleatorizacao[contexec,5]
contexec<-contexec+1
```

```
simul<-simul+1
}
contexec<-contexec+1

}


## ------------------------------------------------------------------
## 6.4 - ROTINA PARA PRIMEIRA SELEÇÃO POR TORNEIOS.
## ------------------------------------------------------------------

## ----------------------------------------------------
## SELEÇÃO DE MODELOS - torneio
##
## este método é aplicado para cada dois conjuntos dos quatro
## obtidos anteriormente e os vencedores seguem para as
## próximas etapas.
## ----------------------------------------------------
selecao<-matrix(data = 0, nrow =length(inicializacaoavaliacao[,1])/2, ncol = 5)
i<-1
j<-1

while (j<=length(selecao[,1])) {
if (inicializacaoavaliacao[i,5] <= inicializacaoavaliacao[i+1,5]) vencedor<-inicializacaoavaliacao[i,]
if (inicializacaoavaliacao[i,5] > inicializacaoavaliacao[i+1,5]) vencedor<-inicializacaoavaliacao[i+1,]
vencedor<-as.matrix(vencedor)
vencedor<-t(vencedor)
selecao[j,1]<-vencedor[1,1]
selecao[j,2]<-vencedor[1,2]
selecao[j,3]<-vencedor[1,3]
selecao[j,4]<-vencedor[1,4]
selecao[j,5]<-vencedor[1,5]

i<-i+2
j<-j+1
}

selecoes[ger,1]<-ger
selecoes[ger,2]<-selecao[1,1]
selecoes[ger,3]<-selecao[1,2]
selecoes[ger,4]<-selecao[1,3]
selecoes[ger,5]<-selecao[1,4]
selecoes[ger,6]<-selecao[1,5]
selecoes[ger,7]<-selecao[2,1]
selecoes[ger,8]<-selecao[2,2]
selecoes[ger,9]<-selecao[2,3]
selecoes[ger,10]<-selecao[2,4]
selecoes[ger,11]<-selecao[2,5]

## ------------------------------------------------------------------
## 6.5 - ROTINA PARA SEGUNDA SELEÇÃO POR TORNEIOS.
## ------------------------------------------------------------------

selecao2<-matrix(data = 0, nrow =length(selecao[,1])/2, ncol = 5)
if (selecao[1,5] <= selecao[2,5]) selecao2<-selecao[1,]
if (selecao[1,5]  > selecao[2,5]) selecao2<-selecao[2,]

selecao2<-as.matrix(selecao2)

selecao2<-t(selecao2)
```

```
selecoes2[ger,1]<-ger
selecoes2[ger,2]<-selecao2[1,1]
selecoes2[ger,3]<-selecao2[1,2]
selecoes2[ger,4]<-selecao2[1,3]
selecoes2[ger,5]<-selecao2[1,4]
selecoes2[ger,6]<-selecao2[1,5]

quadrorecombimutacao[1,1]<-selecao2[1,1]
quadrorecombimutacao[1,2]<-selecao2[1,2]
quadrorecombimutacao[1,3]<-selecao2[1,3]
quadrorecombimutacao[1,4]<-selecao2[1,4]
quadrorecombimutacao[1,5]<-selecao2[1,5]
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999
## ------------------------------------------------------------------
## 6.6 - ROTINA QUE ESTABELECE RECOMBINAÇÃO BLX - alfa OU MUTAÇÃO LIMITE.
## ------------------------------------------------------------------

probabilidade <- runif(1)
if (probabilidade <= 0.60)
        {
opcao<-0
pos1<-1
pos2<-11
contsimula<-1
while (abs(pos1-pos2)<=9) {
alfa<-runif(1)
while (vetoralg[pos1,2]==vetoralg[pos2,2])
while (pos1 <= 0)
while (pos1 > compgenoma)
while (pos2 <= 0)
while (pos2 > compgenoma)
{
beta<-runif(2,-alfa,1+alfa)
beta<-as.matrix(beta)
pos1<-selecao2[1,1]+beta[1,1]*(selecao2[1,3]-selecao2[1,1])
pos2<-selecao2[1,1]+beta[2,1]*(selecao2[1,3]-selecao2[1,1])
}
contsimula<-contsimula+1
}
testes<-cria.testes.aux.MRN(pos1,pos2)
SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
quadrorecombimutacao[1,1]<-pos1
quadrorecombimutacao[1,2]<-vetoralg[pos1,2]
quadrorecombimutacao[1,3]<-pos2
quadrorecombimutacao[1,4]<-vetoralg[pos2,2]
quadrorecombimutacao[1,5]<-SSE
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999

## fim da rotina de recombinação
}

## ------------------------------------------------------------------
## 6.6.2 - ROTINA DE MUTAÇÃO
## ------------------------------------------------------------------
##
if (probabilidade > 0.6) {
pmlmd<-runif(1)

if (pmlmd <= 0.5)
{
```

```
## print("MUTAÇÃO")
## ------------------------------------------------------------------
## 6.6.2.1 - ROTINA DE MUTAÇÃO LIMITE -  MUTAÇÃO DRÁSTICA
## ------------------------------------------------------------------
##
opcao<-1
pmd<-runif(1)

if (pmd <= 0.5)
{
## print("LIMITE INFERIOR")
if (selecao2[1,1] <= selecao2[1.3]) {
pos1<-1
pos2<-selecao2[1,3]
quadrorecombimutacao[1,1]<-pos1
quadrorecombimutacao[1,2]<-vetoralg[pos1,2]
quadrorecombimutacao[1,3]<-pos2
quadrorecombimutacao[1,4]<-vetoralg[pos2,2]
testes<-cria.testes.aux.MRN(pos1,pos2)
SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
quadrorecombimutacao[1,5]<-SSE
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999

}

if (selecao2[1,1] > selecao2[1.3]) {
pos1<-1
pos2<-selecao2[1,1]
quadrorecombimutacao[1,1]<-pos1
quadrorecombimutacao[1,2]<-vetoralg[pos1,2]
quadrorecombimutacao[1,3]<-pos2
quadrorecombimutacao[1,4]<-vetoralg[pos2,2]
testes<-cria.testes.aux.MRN(pos1,pos2)
SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
quadrorecombimutacao[1,5]<-SSE
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999
}

}
if (pmd > 0.5)
{
## print("LIMITE SUPERIOR")

if (selecao2[1,1]> selecao2[1.3]) {
pos1<-selecao2[1,3]
pos2<-compgenoma
quadrorecombimutacao[1,1]<-pos1
quadrorecombimutacao[1,2]<-vetoralg[pos1,2]
quadrorecombimutacao[1,3]<-pos2
quadrorecombimutacao[1,4]<-vetoralg[pos2,2]
testes<-cria.testes.aux.MRN(pos1,pos2)
SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
quadrorecombimutacao[1,5]<-SSE
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999

}

if (selecao2[1,1] <= selecao2[1.3]) {
pos1<-selecao2[1,1]
```

```
pos2<-compgenoma
quadrorecombimutacao[1,1]<-pos1
quadrorecombimutacao[1,2]<-vetoralg[pos1,2]
quadrorecombimutacao[1,3]<-pos2
quadrorecombimutacao[1,4]<-vetoralg[pos2,2]
testes<-cria.testes.aux.MRN(pos1,pos2)
SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
quadrorecombimutacao[1,5]<-SSE
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999

}

}
## final da rotina de mutação drástica
}

if (pmlmd > 0.5) {
opcao<-2
## print("MUTAÇÃO LEVE")
   DELTA<-round(runif(1,0.51,compgenoma+0.49))
if (abs(quadrorecombimutacao[1,1]-DELTA) <= abs(quadrorecombimutacao[1,3]-DELTA))
## (quadrorecombimutacao[1, 5] <= atualizacao[1, 5])
{
pos1<-DELTA
pos2<-selecao2[1,3]
quadrorecombimutacao[1,1]<-pos1
quadrorecombimutacao[1,2]<-vetoralg[pos1,2]
quadrorecombimutacao[1,3]<-pos2
quadrorecombimutacao[1,4]<-vetoralg[pos2,2]
testes<-cria.testes.aux.MRN(pos1,pos2)
SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
quadrorecombimutacao[1,5]<-SSE
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999

}
if (abs(quadrorecombimutacao[1,1]-DELTA) > abs(quadrorecombimutacao[1,3]-DELTA))
{


pos2<-DELTA
pos1<-selecao2[1,1]
quadrorecombimutacao[1,1]<-pos1
quadrorecombimutacao[1,2]<-vetoralg[pos1,2]
quadrorecombimutacao[1,3]<-pos2
quadrorecombimutacao[1,4]<-vetoralg[pos2,2]
testes<-cria.testes.aux.MRN(pos1,pos2)
testes
SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
quadrorecombimutacao[1,5]<-SSE
## if ((quadrorecombimutacao[1,5] == Inf) | (quadrorecombimutacao[1,5] == NA)) quadrorecombimutacao[1,5] <- 99999


}

## final da rotina de mutaçãao leve
}

}
recombinacoes[ger,1]<-ger
recombinacoes[ger,2]<-probabilidade
```

```
recombinacoes[ger,3]<-quadrorecombimutacao[1,1]
recombinacoes[ger,4]<-quadrorecombimutacao[1,2]
recombinacoes[ger,5]<-quadrorecombimutacao[1,3]
recombinacoes[ger,6]<-quadrorecombimutacao[1,4]
recombinacoes[ger,7]<-quadrorecombimutacao[1,5]


if (quadrorecombimutacao[1,5] <= atualizacao[1,5])
{
atualizacao[1,1] <- quadrorecombimutacao[1,1]
atualizacao[1,2] <- quadrorecombimutacao[1,2]
atualizacao[1,3] <- quadrorecombimutacao[1,3]
atualizacao[1,4] <- quadrorecombimutacao[1,4]
atualizacao[1,5] <- quadrorecombimutacao[1,5]
}

atualizacoes[ger,1]<-ger
atualizacoes[ger,2]<-probabilidade
atualizacoes[ger,3]<-atualizacao[1,1]
atualizacoes[ger,4]<-atualizacao[1,2]
atualizacoes[ger,5]<-atualizacao[1,3]
atualizacoes[ger,6]<-atualizacao[1,4]
atualizacoes[ger,7]<-atualizacao[1,5]
todasatualizacoes[ger,ns]<-atualizacao[1,5]


## ------------------------------------------------------------------
## 6.7 - ROTINA QUE ATUALIZA MATRIX DE ATUALIZAÇÃO PARA DOIS PARES DE
##        POSIÇÕES E SEUS RESPECTIVOS VALORES PARA O RESÍDUO SSE.
## ------------------------------------------------------------------
##
maximo<-max(inicializacaoavaliacao[,5])
i<-1
while(maximo != inicializacaoavaliacao[i,5]) i<-i+1

minimo<-min(atualizacao[,5])

if (minimo <= maximo) {
inicializacaoavaliacao[i,1]<-atualizacao[1,1]
inicializacaoavaliacao[i,2]<-atualizacao[1,2]
inicializacaoavaliacao[i,3]<-atualizacao[1,3]
inicializacaoavaliacao[i,4]<-atualizacao[1,4]
inicializacaoavaliacao[i,5]<-atualizacao[1,5]
}

print("geracao: ")
print(ger)
 ger<-ger+1
print(atualizacao)
 }
print(atualizacao)
win.graph()
plot(atualizacoes[,7])
solucao[1,1]<-atualizacoes[ger-1,3]
solucao[1,2]<-atualizacoes[ger-1,4]
solucao[1,3]<-atualizacoes[ger-1,5]
solucao[1,4]<-atualizacoes[ger-1,6]
solucao[1,5]<-atualizacoes[ger-1,7]

print("solucao")
```

```
print(ns)

solucoes[ns,1]<-solucao[1,1]
solucoes[ns,2]<-solucao[1,2]
solucoes[ns,3]<-solucao[1,3]
solucoes[ns,4]<-solucao[1,4]
solucoes[ns,5]<-solucao[1,5]
ns<-ns + 1}
print(solucoes)
plot(solucoes[,5])

## ------------------------------------------------------------------------
## 7 - IMPLEMENTAÇÃO DA BUSCA EXAUSTIVA
## CONSIDERANDO DISTANCIA DE ATÉ 1 Cm
## ------------------------------------------------------------------------
todasposicoes<-matrix(data = 0, nrow = (compgenoma-10)*(1+compgenoma-10),  ncol = 5)
pos1<-1

i<-1
while (pos1 <= compgenoma)
{

pos2<-pos1+10

while (pos2 <= compgenoma)
{

testes<-cria.testes.aux.MRN(pos1,pos2)

 AIC<-testes$resultados2[length(testes$resultados2[,1]),11]
 BIC<-testes$resultados2[length(testes$resultados2[,1]),12]
 SSE<-testes$resultados2[length(testes$resultados2[,1]),4]
## if (BIC == Inf) BIC <-99999
todasposicoes[i,1]<-pos1
todasposicoes[i,2]<-pos2
todasposicoes[i,3]<-AIC
todasposicoes[i,4]<-BIC
todasposicoes[i,5]<-SSE
print(todasposicoes[i,])
i<-i+1
pos2<-pos2+1
}
pos1<-pos1+1
}

print(todasposicoes)
write.table(todasposicoes,file="e:\\todasposicoes.dat",row.names=FALSE,sep=",")

i<-1
j<-1
buscacondicional<-matrix(0,compgenoma,4)
while (i <= compgenoma)
{

testes<-cria.testes.aux.MRN(i,j)
SSE1<-testes$tabelaea[length(testes$tabelaea[,1]),12]
buscacondicional[i,1]<-i
buscacondicional[i,2]<- SSE1
SSE2<-testes$tabelaea2[length(testes$tabelaea2[,1]),12]
buscacondicional[i,3]<-j
```

```
buscacondicional[i,4]<-SSE2
i<-i+1
print(i)
j<=j+1
print(j)
}
buscacondicional
```