Identifying Genotype-by-Environment Interactions in the Metabolism of Germinating Arabidopsis Seeds Using Generalized Genetical Genomics

Joosen, Ronny Viktor Louis; Arends, Danny; Li, Yang; Willems, Leo A. J.; Keurentjes, Joost J. B.; Ligterink, Wilco; Jansen, Ritsert C.; Hilhorst, Henk W. M.

Published in:
Plant Physiology

DOI:
10.1104/pp.113.216176

```
######################################################################
#
#
# copyright (c) 2010-2011, Danny Arends
# Adjusted by Ronny Joosen
# last modified Jan, 2012
# first written Nov, 2010
#
#     This program is free software; you can redistribute it and/or
#     modify it under the terms of the GNU General Public License,
#     version 3, as published by the Free Software Foundation.
#
#     This program is distributed in the hope that it will be useful,
#     but without any warranty; without even the implied warranty of
#     merchantability or fitness for a particular purpose.  See the GNU
#     General Public License, version 3, for more details.
#
#     A copy of the GNU General Public License, version 3, is available
#     at http://www.r-project.org/Licenses/GPL-3
#
# Contains: Metabolite QTL mapping pipeline with for generalized genetical
genomic design experiments
#
######################################################################

######## some  Indexing to search across the long script############

# Dataloading and pre-processing

# Declare environmental specific combinations

# Map QTl procedure for a single environment
# Map QTl prcedure for multiple environments
# Actual mapping procedure, mapall
# procedure to do Permutation test to determine LOD threshold values

# Mapping single environments Y = G
# Mapping multiple environments Y = E + G:E + G
# Permutation tests

# data extraction general procedures
# procedure for peak detection
# procedure for Heatmaps
# Procedure to plot distibution for specific locus

# get the permutation results for environments
# saving LOD profiles, save heatmap and perform peakdetection

############################################################
############################################################

# Set working directory and load libraries
dir = "E:/Rqtl/mQTLmap1/mQTL mapping version 1/"
set.seed(1000)
library(qtl)
setwd(dir)

# Dataloading and pre-processing
mycross <- read.cross("csvr",file="AraClassical_Raw.csv",genotypes=c("A","H"))
mycross <- convert2riself(mycross)

cond1 <-
read.cross("csvr",file="AraClassical_GCMS_Centro_1.csv",genotypes=c("A","H"))
cond1 <- convert2riself(cond1)

cond2 <-
```

```
read.cross("csvr",file="AraClassical_GCMS_Centro_2.csv",genotypes=c("A","H"))
cond2 <- convert2riself(cond2)

cond3 <-
read.cross("csvr",file="AraClassical_GCMS_Centro_3.csv",genotypes=c("A","H"))
cond3 <- convert2riself(cond3)

cond4 <-
read.cross("csvr",file="AraClassical_GCMS_Centro_4.csv",genotypes=c("A","H"))
cond4 <- convert2riself(cond4)

germ_annot <- read.table("germ_At.txt",sep="\t",header=T,row.names=1)
havegermname <-
(1:length(rownames(germ_annot)))[-c(which(germ_annot[,1]==""),which(germ_annot[,
1]=="?"),which(germ_annot[,1]=="no hit"))]

annot <- read.table("centrotype_At.txt",sep="\t",header=T,row.names=1)
havename <-
(1:length(rownames(annot)))[-c(which(annot[,1]==""),which(annot[,1]=="?"),which(
annot[,1]=="no hit"))]

markernames <- unlist(lapply(FUN=names,pull.map(cond1)),use.names=F)
markerchr <- names(rep(nmar(cond1),nmar(cond1)))
markerdist <- unlist(pull.map(cond1),use.names=F)

# Declare environmental specific combinations
############Germination traits############

envigerm <- c(rep(1,165))
enviColgerm <- c(rep("red",165))

phenotypesgerm <- rbind(pull.pheno(mycross))
colnames(phenotypesgerm)[havegermname] <-  gsub("
","",as.character(germ_annot[havegermname,1]))
#phenotypes1 <- log10(phenotypes1) #take the log of phenotypes

####removing outliers via zscore transformation
zscores <-
apply(phenotypesgerm,2,function(x){(x-mean(x,na.rm=T))/sd(x,na.rm=T)})
cnt <- 0
  removed <- NULL
  for(x in 1:ncol(zscores)){
    n <- which(abs(zscores[,x])>3)
    cnt <- cnt + length(n)
    phenotypesgerm[n,x] = NA}
#phenotypesgerm <- rbind(phenotypesgerm)
cntgerm <- cnt

phenotypesgerm <- t(phenotypesgerm)
genotypesgerm <- rbind(pull.geno(fill.geno(mycross)))
phenonames <- rownames(phenotypesgerm)

getenvgerm <- function(x){
x[,3]
}

getgengerm <- function(x){
x[,4]
}

getintgerm <- function(x){
x[,5]
}


############1############
```

```
envi1 <- c(rep(1,44))
enviCol1 <- c(rep("red",44))

phenotypes1 <- rbind(pull.pheno(cond1))
colnames(phenotypes1)[havename] <-  gsub(" ","",as.character(annot[havename,1]))
phenotypes1 <- log10(phenotypes1) #take the log of phenotypes

####removing outliers via zscore transformation
zscores <- apply(phenotypes1,2,function(x){(x-mean(x,na.rm=T))/sd(x,na.rm=T)})
cnt <- 0
  removed <- NULL
  for(x in 1:ncol(zscores)){
    n <- which(abs(zscores[,x])>3)
    cnt <- cnt + length(n)
    phenotypes1[n,x] = NA}
cnt1 <-cnt

phenotypes1 <- t(phenotypes1)
genotypes1 <- rbind(pull.geno(fill.geno(cond1)))
phenonames <- rownames(phenotypes1)

getenv1 <- function(x){
x[,3]
}

getgen1 <- function(x){
x[,4]
}

getint1 <- function(x){
x[,5]
}

############2############

envi2 <- c(rep(2,41))
enviCol2 <- c(rep("green",41))

phenotypes2 <- rbind(pull.pheno(cond2))
colnames(phenotypes2)[havename] <-  gsub(" ","",as.character(annot[havename,1]))
phenotypes2 <- log10(phenotypes2) #take the log of phenotypes

####removing outliers via zscore transformation
zscores <- apply(phenotypes2,2,function(x){(x-mean(x,na.rm=T))/sd(x,na.rm=T)})
cnt <- 0
  removed <- NULL
  for(x in 1:ncol(zscores)){
    n <- which(abs(zscores[,x])>3)
    cnt <- cnt + length(n)
    phenotypes2[n,x] = NA}
cnt2 <-cnt

phenotypes2 <- t(phenotypes2)
genotypes2 <- rbind(pull.geno(fill.geno(cond2)))
phenonames <- rownames(phenotypes2)

getenv2 <- function(x){
x[,3]
}

getgen2 <- function(x){
x[,4]
}

getint2 <- function(x){
```

```
x[,5]
}

############3############

envi3 <- c(rep(3,41))
enviCol3 <- c(rep("green",41))

phenotypes3 <- rbind(pull.pheno(cond3))
colnames(phenotypes3)[havename] <-  gsub(" ","",as.character(annot[havename,1]))
phenotypes3 <- log10(phenotypes3) #take the log of phenotypes

####removing outliers via zscore transformation
zscores <- apply(phenotypes3,2,function(x){(x-mean(x,na.rm=T))/sd(x,na.rm=T)})
cnt <- 0
  removed <- NULL
  for(x in 1:ncol(zscores)){
    n <- which(abs(zscores[,x])>3)
    cnt <- cnt + length(n)
    phenotypes3[n,x] = NA}
cnt3 <-cnt

phenotypes3 <- t(phenotypes3)
genotypes3 <- rbind(pull.geno(fill.geno(cond3)))
phenonames <- rownames(phenotypes3)

getenv3 <- function(x){
x[,3]
}

getgen3 <- function(x){
x[,4]
}

getint3 <- function(x){
x[,5]
}

############4############

envi4 <- c(rep(4,38))
enviCol4 <- c(rep("purple",38))

phenotypes4 <- rbind(pull.pheno(cond4))
colnames(phenotypes4)[havename] <-  gsub(" ","",as.character(annot[havename,1]))
phenotypes4 <- log10(phenotypes4) #take the log of phenotypes

####removing outliers via zscore transformation
zscores <- apply(phenotypes4,2,function(x){(x-mean(x,na.rm=T))/sd(x,na.rm=T)})
cnt <- 0
  removed <- NULL
  for(x in 1:ncol(zscores)){
    n <- which(abs(zscores[,x])>3)
    cnt <- cnt + length(n)
    phenotypes4[n,x] = NA}
cnt4 <-cnt

phenotypes4 <- t(phenotypes4)
genotypes4 <- rbind(pull.geno(fill.geno(cond4)))
phenonames <- rownames(phenotypes4)

getenv4 <- function(x){
x[,3]
}

getgen4 <- function(x){
```

```
x[,4]
}

getint4 <- function(x){
x[,5]
}


############1234#################
envi1234 <- c(rep(1,44),rep(2,41),rep(3,41),rep(4,38))
enviCol1234 <- c(rep("red",44),rep("green",41),rep("blue",41),rep("purple",38))

phenotypes1234 <-
rbind(pull.pheno(cond1),pull.pheno(cond2),pull.pheno(cond3),pull.pheno(cond4))
colnames(phenotypes1234)[havename] <-  gsub("
","",as.character(annot[havename,1]))
phenotypes1234 <- log10(phenotypes1234) #take the log of phenotypes

####removing outliers via zscore transformation
zscores <-
apply(phenotypes1234,2,function(x){(x-mean(x,na.rm=T))/sd(x,na.rm=T)})
cnt <- 0
  removed <- NULL
  for(x in 1:ncol(zscores)){
    n <- which(abs(zscores[,x])>3)
    cnt <- cnt + length(n)
    phenotypes1234[n,x] = NA}
cnt1234 <-cnt

phenotypes1234 <- t(phenotypes1234)
genotypes1234 <-
rbind(pull.geno(fill.geno(cond1)),pull.geno(fill.geno(cond2)),pull.geno(fill.gen
o(cond3)),pull.geno(fill.geno(cond4)))
phenonames <- rownames(phenotypes1234)

getenv1234 <- function(x){
x[,3]
}

getgen1234 <- function(x){
x[,4]
}

getint1234 <- function(x){
x[,5]
}

### Writing the outliers to a table##
noutliers <- data.frame(cntgerm, cnt1234)
write.table(noutliers, file="noutliers.txt", col.names=NA, sep="\t")

################################################################################
#############
## Some Functions specific for the mapping procedure

# Map QTl procedure for a single environment
mapNoEnvQTL <- function(x, genotypes, phenotypes, markernames, markerchr,
markerdist){
  LODgen <- NULL
  for(marker in 1:ncol(genotypes)){
    genomodel <- lm(phenotypes[x,] ~ genotypes[,marker])
    Pvalues <- anova(genomodel)[[5]]
    LODgen <-
c(LODgen,(genomodel[[1]][2]/abs(genomodel[[1]][2]))*-log10(Pvalues[1]))
  }
  out <- rbind(as.numeric(markerchr),markerdist,LODgen)
```

```
  colnames(out) <- markernames
  rownames(out) <- c("chr","pos","genetic")
  out <- t(out)
  out <- as.data.frame(out)
  class(out) <- c(class(out),"scanone")

#plot.scanone(out,lodcolumn=c(1,2,3),ylim=c(-30,30),col=c("green","black","blue"
))
  out
}
# Map QTl prcedure for multiple environments
mapEnvQTL <- function(x,genotypes, phenotypes, enviroment, markernames,
markerchr, markerdist){
  LODenv <- NULL
  LODgen <- NULL
  LODint <- NULL
  for(marker in 1:ncol(genotypes)){
    genomodel <- lm(terms(phenotypes[x,] ~ enviroment +
enviroment:genotypes[,marker] + genotypes[,marker], keep.order=FALSE))
    Pvalues <- anova(genomodel)[[5]]
    LODenv <-
c(LODenv,(genomodel[[1]][2]/abs(genomodel[[1]][2]))*-log10(Pvalues[1]))
    LODint <-
c(LODint,(genomodel[[1]][3]/abs(genomodel[[1]][3]))*-log10(Pvalues[2]))
    LODgen <-
c(LODgen,(genomodel[[1]][4]/abs(genomodel[[1]][4]))*-log10(Pvalues[3]))
  }
  out <- rbind(as.numeric(markerchr),markerdist,LODenv,LODgen,LODint)
  colnames(out) <- markernames
  rownames(out) <- c("chr","pos","env","env:genetic","genetic")
  out <- t(out)
  out <- as.data.frame(out)
  class(out) <- c(class(out),"scanone")

#plot.scanone(out,lodcolumn=c(1,2,3),ylim=c(-30,30),col=c("green","black","blue"
))
  out
}

# Actual mapping procedure, mapall
mapall <- function(genotypes, phenotypes, enviroment,
markernames,markerchr,markerdist, pheno.col= 1:nrow(phenotypes), verbose =
TRUE){
  results <- vector("list", nrow(phenotypes))
  for(x in pheno.col){
    if(missing(enviroment)){
      results[[x]] <- mapNoEnvQTL(x, genotypes, phenotypes, markernames,
markerchr, markerdist)
    }else{
      results[[x]] <- mapEnvQTL(x, genotypes, phenotypes, enviroment,
markernames, markerchr, markerdist)
    }
    if(verbose) cat("Done",x,"/",nrow(phenotypes),"\n")
  }
  results
}

get_env <- function(x){
  x[,3]
}

get_gen <- function(x){
  x[,5]
}

get_int <- function(x){
```

```
  x[,4]
}

# procedure to do Permutation test to determine LOD threshold values

#Set the number of permutations
nperm <- 1

### Permutation test to determine LOD threshold values
# Permutation test for the Genetic component
permallR_GENETIC <- function(g, p, e,
markernames,markerchr,markerdist,n.perm=nperm){
  perm_results <- NULL
  for(x in 1:n.perm){
    geno <- g[sample(nrow(g)),]
    results <- mapall(geno, p, e,
markernames,markerchr,markerdist,verbose=FALSE)
    if(missing(e)){
      mymatrix <- do.call(rbind,lapply(FUN=get_env,results))
    }else{
      mymatrix <- do.call(rbind,lapply(FUN=get_gen,results))
    }
    perm_results <- rbind(perm_results,apply(mymatrix,1,max))
   cat("Done",x,"/",n.perm,"\n")
  }
  perm_results
}

# Permutation test for the Genetic:Environment component
permallR_GXE <- function(g, p, e,
markernames,markerchr,markerdist,n.perm=nperm){
  perm_results <- NULL
  for(x in 1:n.perm){
    if(missing(e)){
      stop("No environment !!!!!")
    }else{
      geno <- g[sample(nrow(g)),]
      results <- mapall(geno, p, e,
markernames,markerchr,markerdist,verbose=FALSE)
      mymatrix <- do.call(rbind,lapply(FUN=get_int,results))
    }
    perm_results <- rbind(perm_results,apply(mymatrix,1,max))
   cat("Done",x,"/",n.perm,"\n")
  }
  perm_results
}

################################################################################
################################################################################

# Mapping single environments Y = G
 germ1 <-
mapall(genotypesgerm,phenotypesgerm,markernames=markernames,markerchr=markerchr,
markerdist=markerdist)
 res1 <-
mapall(genotypes1,phenotypes1,markernames=markernames,markerchr=markerchr,marker
dist=markerdist)
 res2 <-
mapall(genotypes2,phenotypes2,markernames=markernames,markerchr=markerchr,marker
dist=markerdist)
 res3 <-
mapall(genotypes3,phenotypes3,markernames=markernames,markerchr=markerchr,marker
dist=markerdist)
 res4 <-
mapall(genotypes4,phenotypes4,markernames=markernames,markerchr=markerchr,marker
dist=markerdist)
```

```
# Mapping multiple environments Y = E + G:E + G
  res1234env <- mapall(genotypes1234,
phenotypes1234,envi1234,markernames,markerchr,markerdist)

# Permutation tests
#Permutation for the Genetic component
 permgerm <-
permallR_GENETIC(genotypesgerm,phenotypesgerm,markernames=markernames,markerchr=
markerchr,markerdist=markerdist)
 perm1 <-
permallR_GENETIC(genotypes1,phenotypes1,markernames=markernames,markerchr=marker
chr,markerdist=markerdist)
 perm2 <-
permallR_GENETIC(genotypes2,phenotypes2,markernames=markernames,markerchr=marker
chr,markerdist=markerdist)
 perm3 <-
permallR_GENETIC(genotypes3,phenotypes3,markernames=markernames,markerchr=marker
chr,markerdist=markerdist)
 perm4 <-
permallR_GENETIC(genotypes4,phenotypes4,markernames=markernames,markerchr=marker
chr,markerdist=markerdist)

#Permutation for the Genetic:Environment component
 perm1234gxe <-
permallR_GXE(genotypes1234,phenotypes1234,envi1234,markernames=markernames,marke
rchr=markerchr,markerdist=markerdist)

##############################################################################
##############################################################################
# data extraction general procedures

# procedure for peak detection
getpeaks <- function(qtlprofiles, cutoff = 3.0){
  cat("Starting peak detection above",cutoff,"\n")
  mmatrix <- NULL
  for(x in 1:nrow(qtlprofiles)){
    peak <- FALSE
    curmax <- 0
    curmaxindex <- 1
    marker <- 1
    maximums <- NULL
    mrow <- rep(0,ncol(qtlprofiles))
    for(ab in (qtlprofiles[x,]>cutoff | qtlprofiles[x,]<(-cutoff))){
      if(ab){
        peak <- TRUE
        if(qtlprofiles[x,marker]/abs(qtlprofiles[x,marker]) > 0){
          if(qtlprofiles[x,marker] > curmax){
            curmax <- qtlprofiles[x,marker]
            curmaxindex <- marker
          }
        }else{
          if(qtlprofiles[x,marker] < (-curmax)){
            curmax <- qtlprofiles[x,marker]
            curmaxindex <- -marker
          }
        }
        if(ncol(qtlprofiles)==marker){
          if(curmax!=0) maximums <- c(maximums,curmaxindex)
        }
      }else{
        if(curmax!=0) maximums <- c(maximums,curmaxindex)
        peak <- FALSE
        curmax <- 0
      }
      marker <- marker+1
```

```
    }
    mrow[which(qtlprofiles[x,] > cutoff)] <- 1
    mrow[which(qtlprofiles[x,] < -cutoff)] <- -1
    for(a in which(maximums>0)){
      mrow[maximums[a]] <- 2
    }
    for(b in which(maximums<0)){
      mrow[(-maximums[b])] <- -2
    }
    mmatrix <- rbind(mmatrix,mrow)
  }
  mmatrix
}
# procedure for Heatmaps
plotheatmap <- function(FUN=getenvsampleA,main="Heatmap",allres,phenonames){
  data <- do.call(rbind,lapply(FUN=FUN,allres))
  col <- c("green","blue","lightblue","white","yellow","orange","red")
  breaks <- c(-100,-10,-5,-2.5,2.5,5,10,100)
  rownames(data) <- phenonames
  heatmap(data,main=main,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
}

# Procedure to plot distibution for specific locus
plotdistr <- function(trait=1,marker=1,genotypes,phenotypes,envi){
  data <- cbind(envi,phenotypes[trait,],genotypes[,marker])
  col <- genotypes[,marker]
  shift <- genotypes[,marker]
  col[col==1] <- "red"
  shift[shift==1] <- -0.1
  col[col==2] <- "blue"
  shift[shift==2] <- 0.1
  col[is.na(col)] <- "gray"
  shift[is.na(shift)] <- 0
  #envi <- envi+shift

plot(cbind(envi,phenotypes[trait,]),col=col,main=paste(rownames(phenotypes)[trai
t],"at marker",marker),xaxt="n",pch=20,cex=2)
  colrz <- c("red","blue")
  res <- NULL
  for(x in 1:4){
    ab <- NULL
    for(y in 1:2){
      a <- which(data[,1]==x)
      b <- which(data[,3]==y)
      t <- mean(data[a[a%in%b],2])
      ab <- c(ab,t)
      cat("Cond",x,"Geno",y,"Val=",t,"\n");
      #points(cbind(x,t),col=colrz[y],pch=y,lwd=5)
    }
    res <- rbind(res,ab)
  }
  res <- cbind(1:4,res)
  lines(res[,c(1,2)],col="red",lwd=3)
  lines(res[,c(1,3)],col="blue",lwd=3)
  res
}

################################################################################
################################################################################

# get the permutation results for environments
signl <- .95
pgerm <- sort(unlist(permgerm))[signl*length(unlist(permgerm))]
p1 <- sort(unlist(perm1))[signl*length(unlist(perm1))]
p2 <- sort(unlist(perm2))[signl*length(unlist(perm2))]
p3 <- sort(unlist(perm3))[signl*length(unlist(perm3))]
```

```
p4 <- sort(unlist(perm4))[signl*length(unlist(perm4))]
p1234gxe <- sort(unlist(perm1234gxe))[signl*length(unlist(perm1234gxe))]

# Writing the results to a table
presults <- data.frame(pgerm, p1, p2, p3,p4,p1234gxe)
write.table(presults, file="permresults.txt", col.names=NA, sep="\t")

# saving LOD profiles, save heatmap and perform peakdetection
# define heatmap colors and breaks
col <- c("green","blue","lightblue","white","yellow","orange","red")
breaks <- c(-100,-10,-8,-5,5,8,10,100)

qtlsmaingerm <- do.call(rbind,lapply(FUN=get_env,germ1))
write.table(qtlsmaingerm, file="qtlsmaingerm.txt", col.names=NA, sep="\t")
header <- "qtlsmaingerm"
pdf("qtlsmaingerm.pdf")
heatmap(qtlsmaingerm
,main=header,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
dev.off()
peaksgerm <- getpeaks(abs(qtlsmaingerm), cutoff =pgerm)
image(1:ncol(peaksgerm),1:nrow(peaksgerm),t(peaksgerm))
write.table(peaksgerm, file="peaksgerm.txt", col.names=NA, sep="\t")


qtlsmain1 <- do.call(rbind,lapply(FUN=get_env,res1))
write.table(qtlsmain1, file="qtlsmain1.txt", col.names=NA, sep="\t")
header <- "qtlsmain1"
pdf("qtlsmain1.pdf")
heatmap(qtlsmain1
,main=header,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
dev.off()
peaks1 <- getpeaks(abs(qtlsmain1), cutoff =p1)
image(1:ncol(peaks1),1:nrow(peaks1),t(peaks1))
write.table(peaks1, file="peaks1.txt", col.names=NA, sep="\t")

qtlsmain2 <- do.call(rbind,lapply(FUN=get_env,res2))
write.table(qtlsmain2, file="qtlsmain2.txt", col.names=NA, sep="\t")
header <- "qtlsmain2"
pdf("qtlsmain2.pdf")
heatmap(qtlsmain2
,main=header,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
dev.off()
peaks2 <- getpeaks(abs(qtlsmain2), cutoff =p2)
image(1:ncol(peaks2),1:nrow(peaks2),t(peaks2))
write.table(peaks2, file="peaks2.txt", col.names=NA, sep="\t")

qtlsmain3 <- do.call(rbind,lapply(FUN=get_env,res3))
write.table(qtlsmain3, file="qtlsmain3.txt", col.names=NA, sep="\t")
header <- "qtlsmain3"
pdf("qtlsmain3.pdf")
heatmap(qtlsmain3
,main=header,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
dev.off()
peaks3 <- getpeaks(abs(qtlsmain3), cutoff =p3)
image(1:ncol(peaks3),1:nrow(peaks3),t(peaks3))
write.table(peaks3, file="peaks3.txt", col.names=NA, sep="\t")

qtlsmain4 <- do.call(rbind,lapply(FUN=get_env,res4))
write.table(qtlsmain4, file="qtlsmain4.txt", col.names=NA, sep="\t")
header <- "qtlsmain4"
pdf("qtlsmain4.pdf")
heatmap(qtlsmain4
,main=header,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
dev.off()
peaks4 <- getpeaks(abs(qtlsmain4), cutoff =p4)
image(1:ncol(peaks4),1:nrow(peaks4),t(peaks4))
```

```
write.table(peaks4, file="peaks4.txt", col.names=NA, sep="\t")

qtlsmain1234 <- do.call(rbind,lapply(FUN=get_gen,res1234env))
write.table(qtlsmain1234, file="qtlsmain1234.txt", col.names=NA, sep="\t")
header <- "qtlsmain1234"
pdf("qtlsmain1234.pdf")
heatmap(qtlsmain1234
,main=header,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
dev.off()
peaks1234 <- getpeaks(abs(qtlsmain1234), cutoff =p1234gxe)
image(1:ncol(peaks1234),1:nrow(peaks1234),t(peaks1234))
write.table(peaks1234, file="peaks1234.txt", col.names=NA, sep="\t")

qtlsmain1234GxE <- do.call(rbind,lapply(FUN=get_int,res1234env))
write.table(qtlsmain1234GxE, file="qtlsmain1234GxE.txt", col.names=NA, sep="\t")
header <- "qtlsmain1234GxE"
pdf("qtlsmain1234GxE.pdf")
heatmap(qtlsmain1234GxE
,main=header,scale="none",Colv=NA,Rowv=NA,col=col,breaks=breaks)
dev.off()
peaks1234GxE <- getpeaks(abs(qtlsmain1234GxE), cutoff =p1234gxe)
image(1:ncol(peaks1234GxE),1:nrow(peaks1234GxE),t(peaks1234GxE))
write.table(peaks1234GxE, file="peaks1234GxE.txt", col.names=NA, sep="\t")

qtlsmain1234E <- do.call(rbind,lapply(FUN=get_env,res1234env))
write.table(qtlsmain1234E[,1], file="qtlsmain1234E.txt", col.names=NA, sep="\t")
header <- "qtlsmain1234E"
pdf("qtlsmain1234E.pdf")
heatmap(abs(qtlsmain1234E),main=header,scale="none",Colv=NA,Rowv=NA,col=col,brea
ks=breaks)
dev.off()

save.image("All with permutation.RData")
```