Supplemental Material

Tittle: Xylem systems genetics analysis reveals a key regulator of lignin biosynthesis in *Populus deltoides*

BALMANT, KM[1]; NOBLE, JD[2]; COUTO, FA[3]; DERVINIS, C[1]; CONDE, D[1]; SCHMIDT, HW[1]; VAZQUEZ, AI[3]; BARBAZUK, WB[2,4,5]; DE LOS CAMPOS, G[3,6]; RESENDE Jr., MFR[2,7]; KIRST, M[1,2,5]

AFFILIATIONS

[1]School of Forest Resources and Conservation, University of Florida, Gainesville, FL 32611, USA.

[2]Plant Molecular and Cellular Biology Graduate Program, University of Florida, Gainesville, FL 32611, USA.

[3]Department of Epidemiology and Biostatistics, Michigan State University, East Lansing, MI 48824, USA.

[4]Department of Biology, University of Florida, Gainesville, FL 32611, USA.

[5]Genetics Institute, University of Florida, Gainesville, FL 32611, USA.

[6]Statistics Department, Michigan State University, East Lansing, MI 48824, USA.

[7]Horticulture Sciences Department, University of Florida, Gainesville, FL 32611, USA.

Table of Contents:

Supplemental Tables

Table S1: Gene Ontology (GO) enrichment of the genes with heritability higher than 0.5. Refer to Tab 1 in Supplemental_Tables.xlsx.

Table S2: Genomic Location of SNPs. Refer to Tab 2 in Supplemental_Tables.xlsx.

Table S3: Genes present in the modules identified in the WGCNA analysis. Refer to Tab 3 in Supplemental_Tables.xlsx.

Table S4: Genes present in the modules identified in the WGCNA analysis using only the genes present in the modules black and red (1st analysis). Refer to Tab 4 in Supplemental_Tables.xlsx.

Table S5: Eigengene-based connectivities (kME) of genes present in the turquoise module. Genes with kME>0.7 are considered hub genes. Refer to Tab 5 in Supplemental_Tables.xlsx.

Table S6: Local-SNPs with significant association with the expression of *MYB125*. Refer to Tab 6 in Supplemental_Tables.xlsx.

Table S7: Pairwise linkage disequilibrium among the SNPs located within *MYB125.* Refer to Tab 7 in Supplemental_Tables.xlsx.

Table S8: Posterior heritability (h2, 95% credibility region) and proportion of the total genetic variance (95% credibility region) explained by the MYB215's cis-regulators for each trait. Refer to Tab 8 in Supplemental_Tables.xlsx.

Table S9: Primer list. Refer to Tab 9 in Supplemental_Tables.xlsx

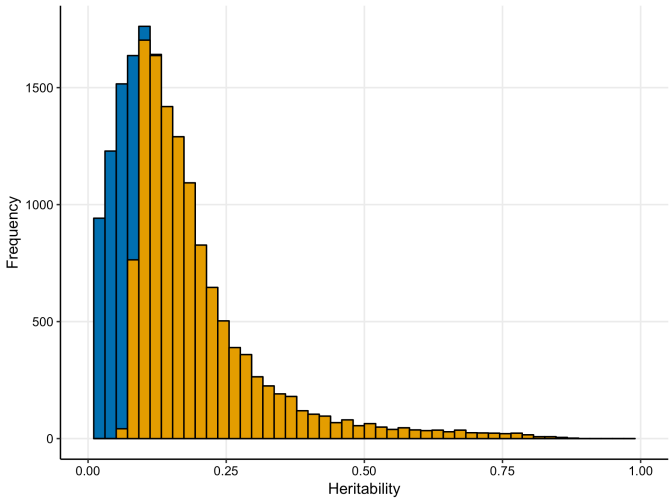

Figure S1 – Gene expression heritability. Distribution of gene expression heritability for all genes (blue) and for the subset of genes that showed heritability higher than expected by chance (yellow; permutation test, p-value < 0.01).

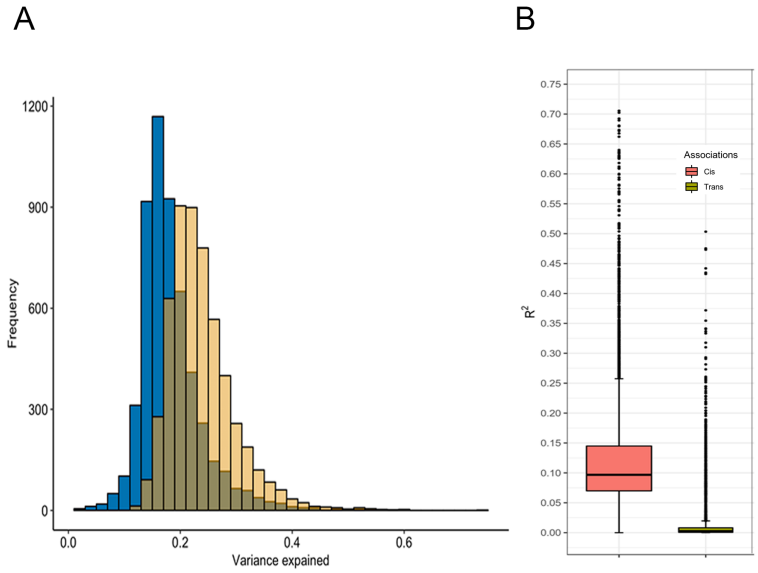

Figure S2 - Expression variance explained for individual local (cis) and distant (trans) SNPs. (A) Stacked histogram showing the distribution of the proportion of phenotypic variance (gene expression) explained by individual local (yellow) and distant (blue) SNPs. (B) Boxplot showing the variance explained by a single SNP (with significant association) for each gene. The horizontal line represents the median and the vertical lines mark the range of the minimum and maximum values.

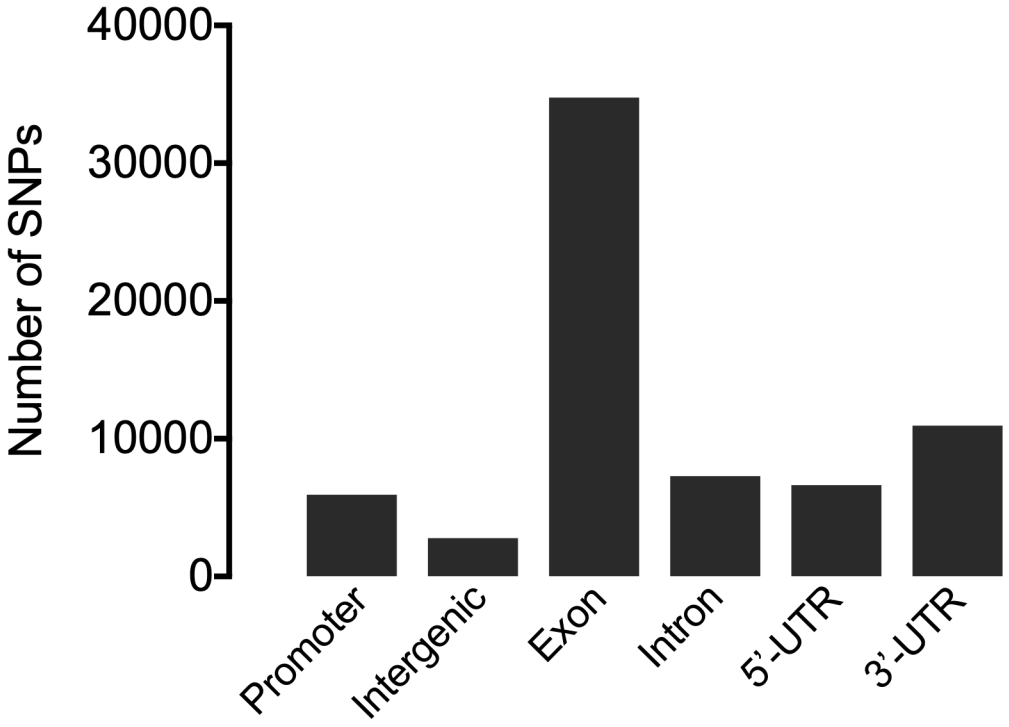

Figure S3 – Genomic location of the all the 68,480 SNPs used in the eQTL analysis. Promoters are defined as 500 base pairs upstream of the gene.

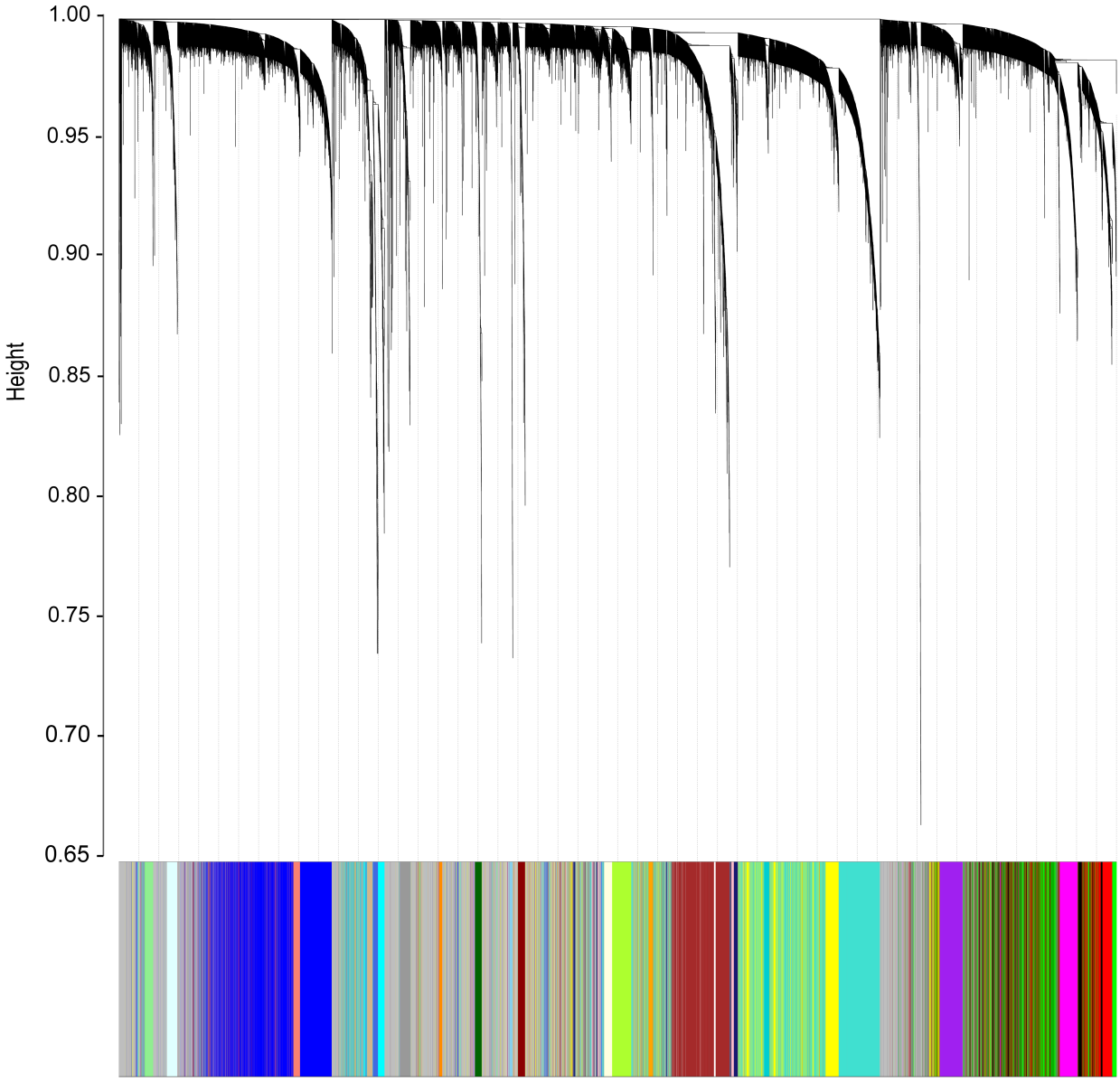

Figure S4 - Weighted gene co-expression analysis (WGCNA) of xylem transcriptome of *P. deltoides*. Hierarchical cluster tree showing 28 modules of co-expressed genes. Each of the 18,207 expressed genes is represented by a leaf in the tree, and each module by the different colors at the bottom. Note that module “Grey” is for unsigned genes.

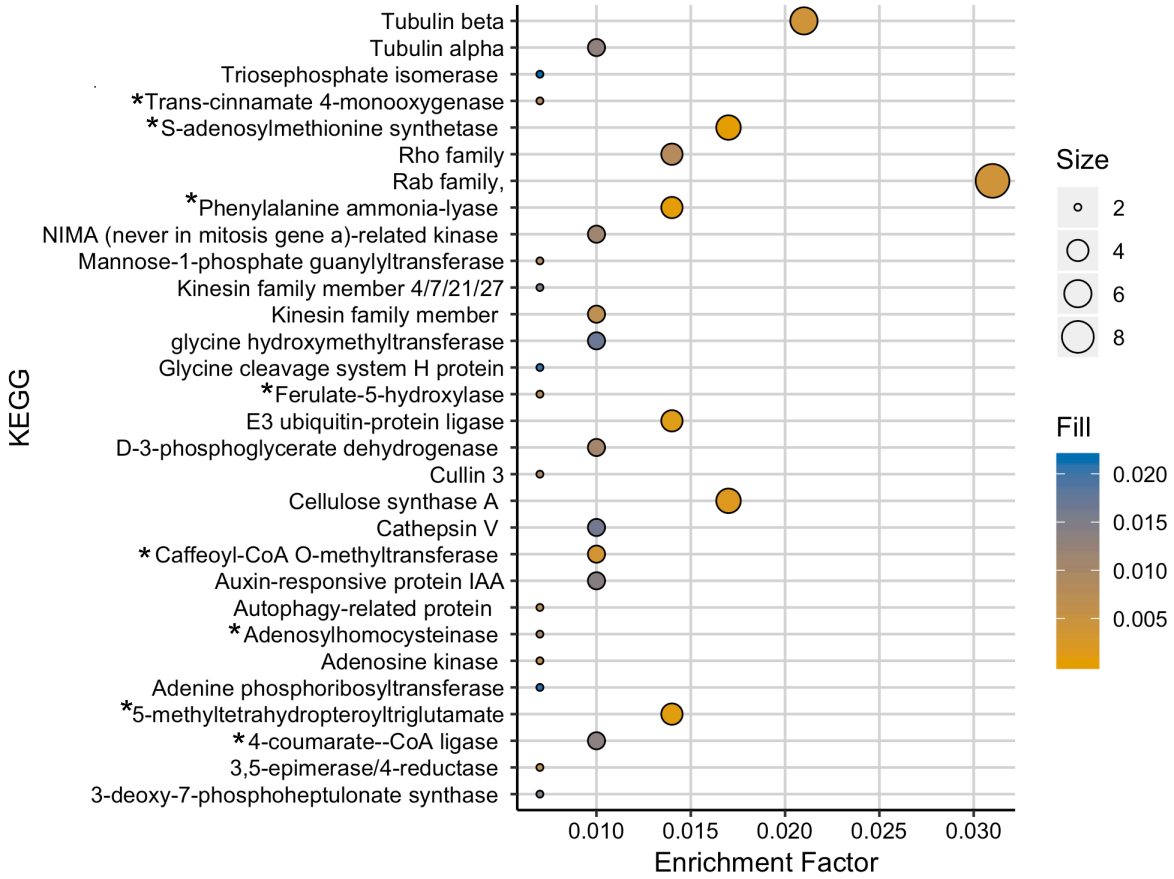

Figure S5 – KEGG orthology enrichment analysis of genes present in modules black and red. Bubble chart shows enrichment of KEGG orthology. Y-axis label represents molecular functions, and X-axis label represents enrichment factor (enrichment factor = number of genes enriched in the pathway/number of all genes in background gene set). Size and color of the bubble represent amount of differentially expressed genes enriched in pathway and enrichment significance, respectively. Asterisks indicate molecular functions related to lignin biosynthesis pathway. The enrichment analysis was performed for all the 28 modules identified in the gene co-expression analysis and only modules black and red showed KEGG enrichment for molecular functions related to lignin biosynthesis pathway.

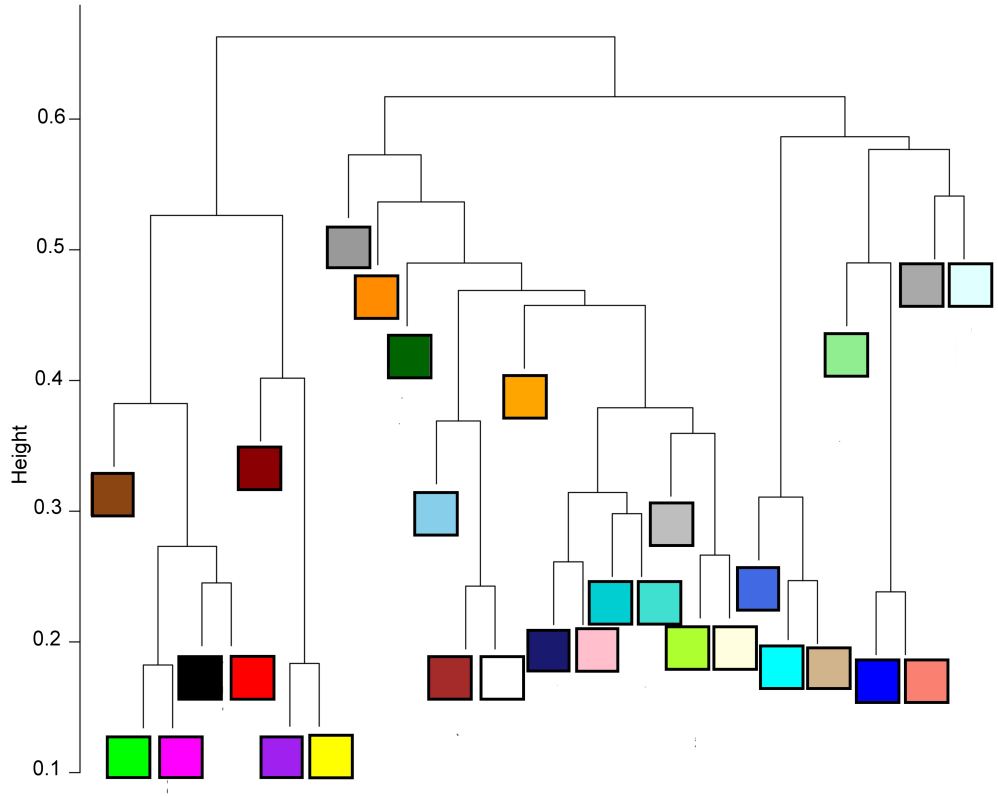

Figure S6 - Hierarchical cluster of modules eigengenes shows closely related modules.

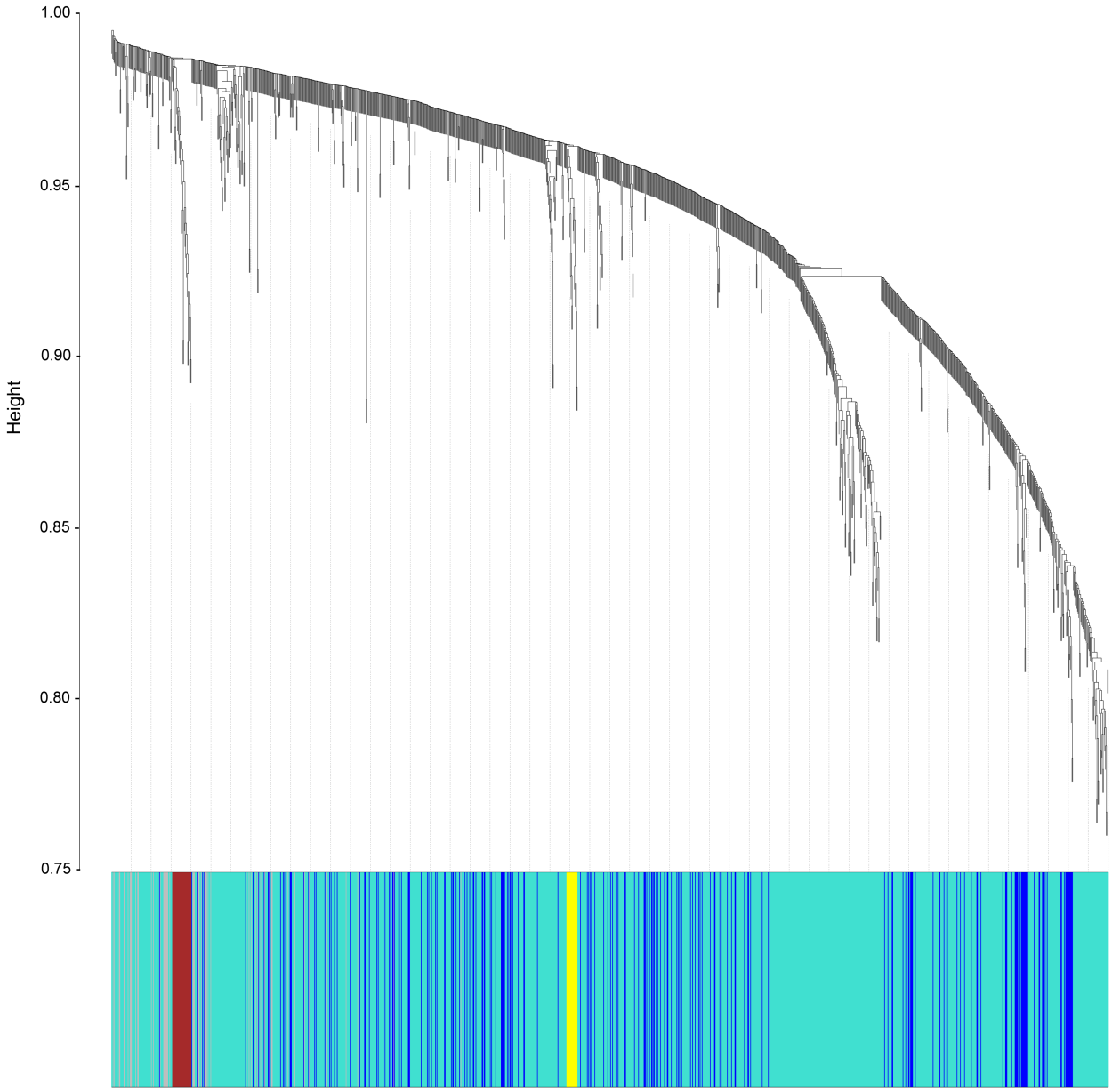

Figure S7 - Weighted gene co-expression analysis (WGCNA) of genes previously clustered in modules black and red. Hierarchical cluster tree showing five modules of co-expressed genes. Each of the 1,459 genes is represented by a leaf in the tree, and each module by the different colors at the bottom.

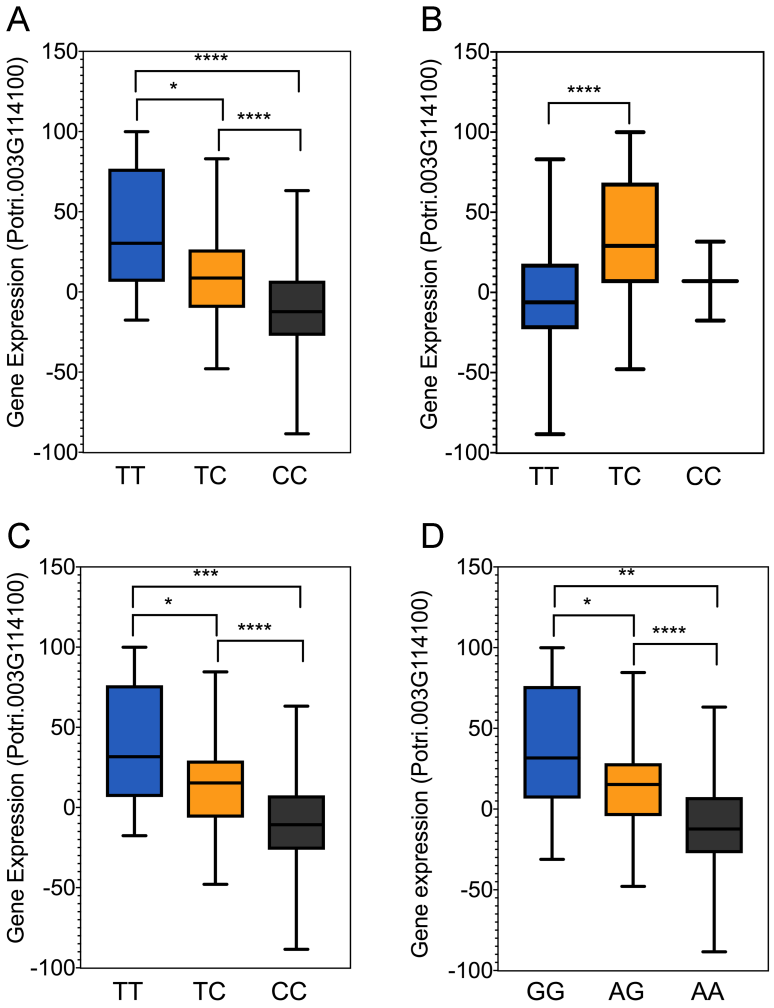

Figure S8 – Local-SNPs regulates the expression of *MYB125.* Boxplot for expression of *MYB125* plotted as an effect of genotypes at the SNPs located in the 2nd exon of *MYB125*. (A) SNP 1 (Position 13769070). (B) SNP 2 (Position 13769229). (C) SNP 3 (Position 13769271). (D) SNP 4 (Position 13769383). The horizontal line represents the median and the vertical lines mark the range of the minimum and maximum values. Data was analyzed using ANOVA followed by Tukey's multiple comparison test. Significance is indicated by asterisks: **** p-value < 0.0001, *** p-value < 0.001, ** p-value < 0.01, * p-value < 0.05.

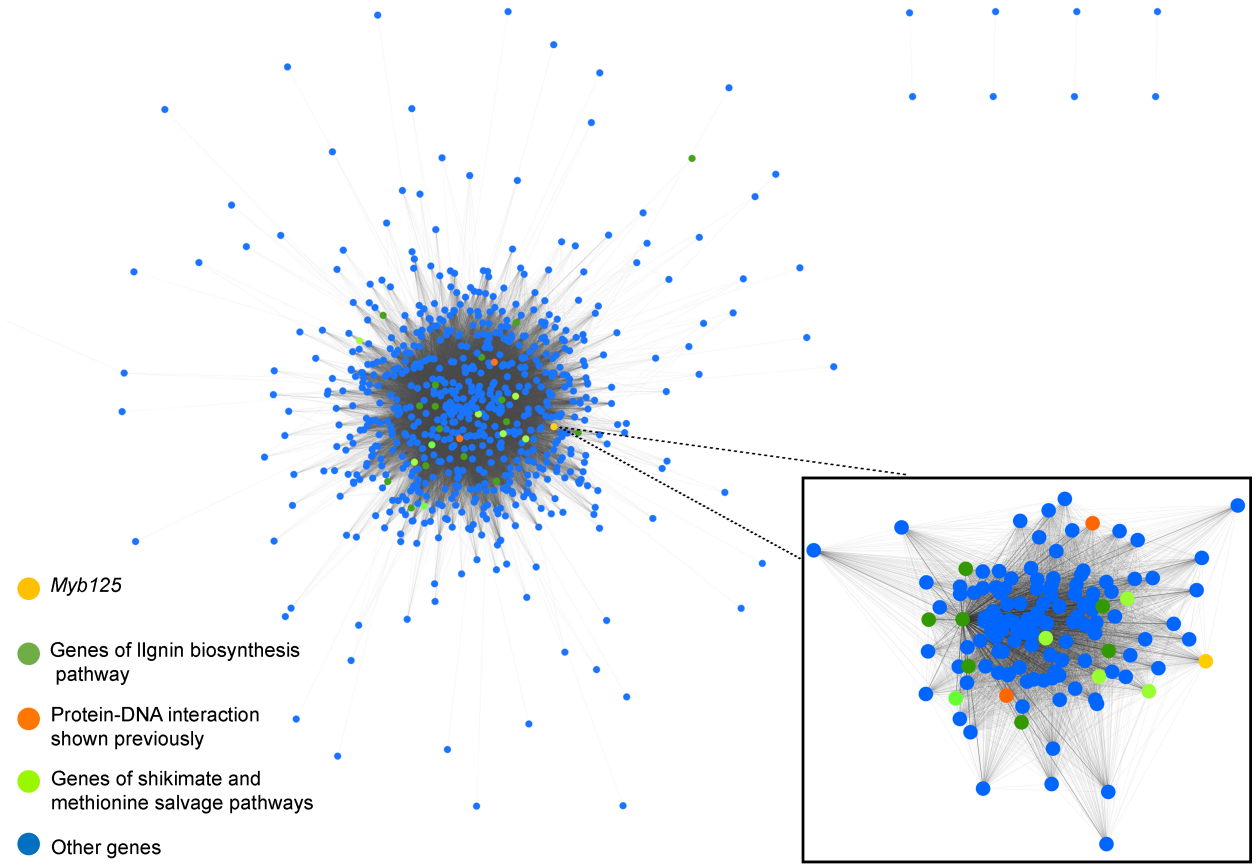

Figure S9 - Cytoscape representation of co-expressed genes network with edge weight ≥ 0.02. Nodes represent genes present in the turquoise module. Edge thickness represent how well two individual nodes are related to each other. Zoom-in region is a sub network of the only the genes directly connect to *MUB125*.

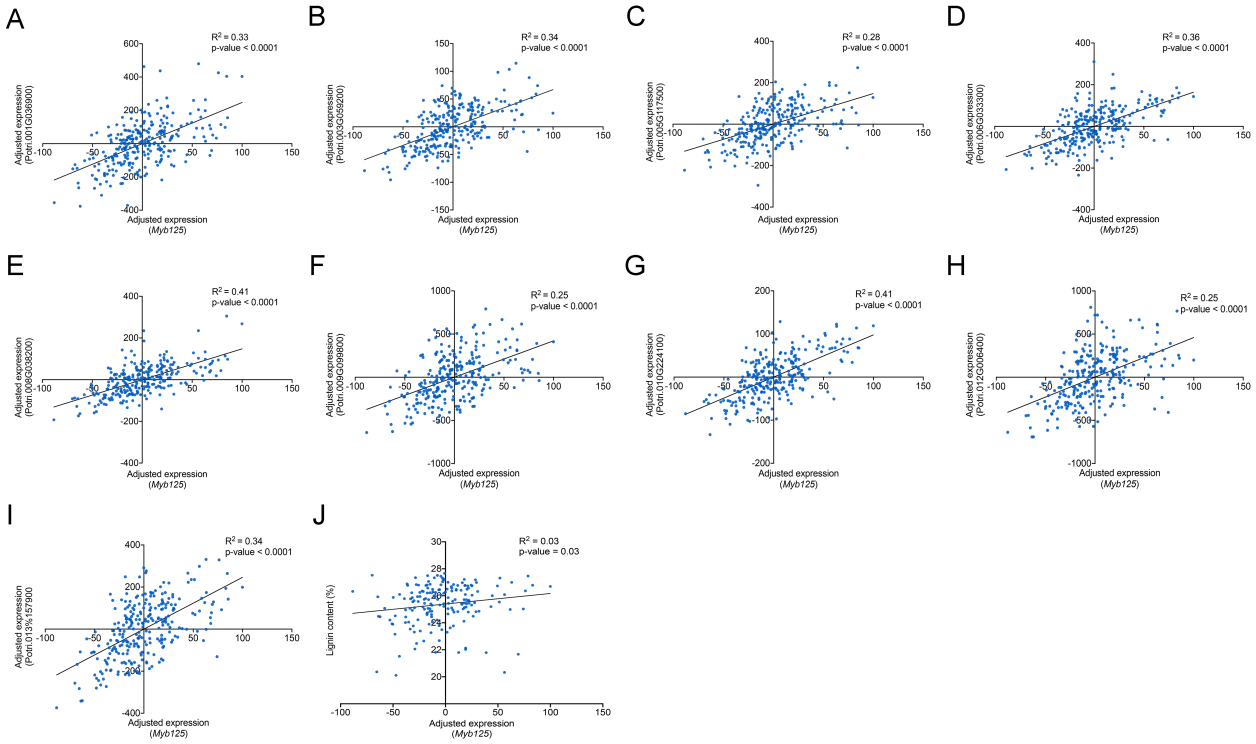

Figure S10 – Expression of *MYB125* is positively correlated with lignin. Expression of *Myb125* is positively correlated with the expression of (A) *4CL3* (B) *CSE* (C) *F5H* (D) *C3'H* (E) *PAL2* (F) *CCoAOMT1* (G) *PAL4* (H) *COMT1* (I) *C4H1*, and (J) lignin content.

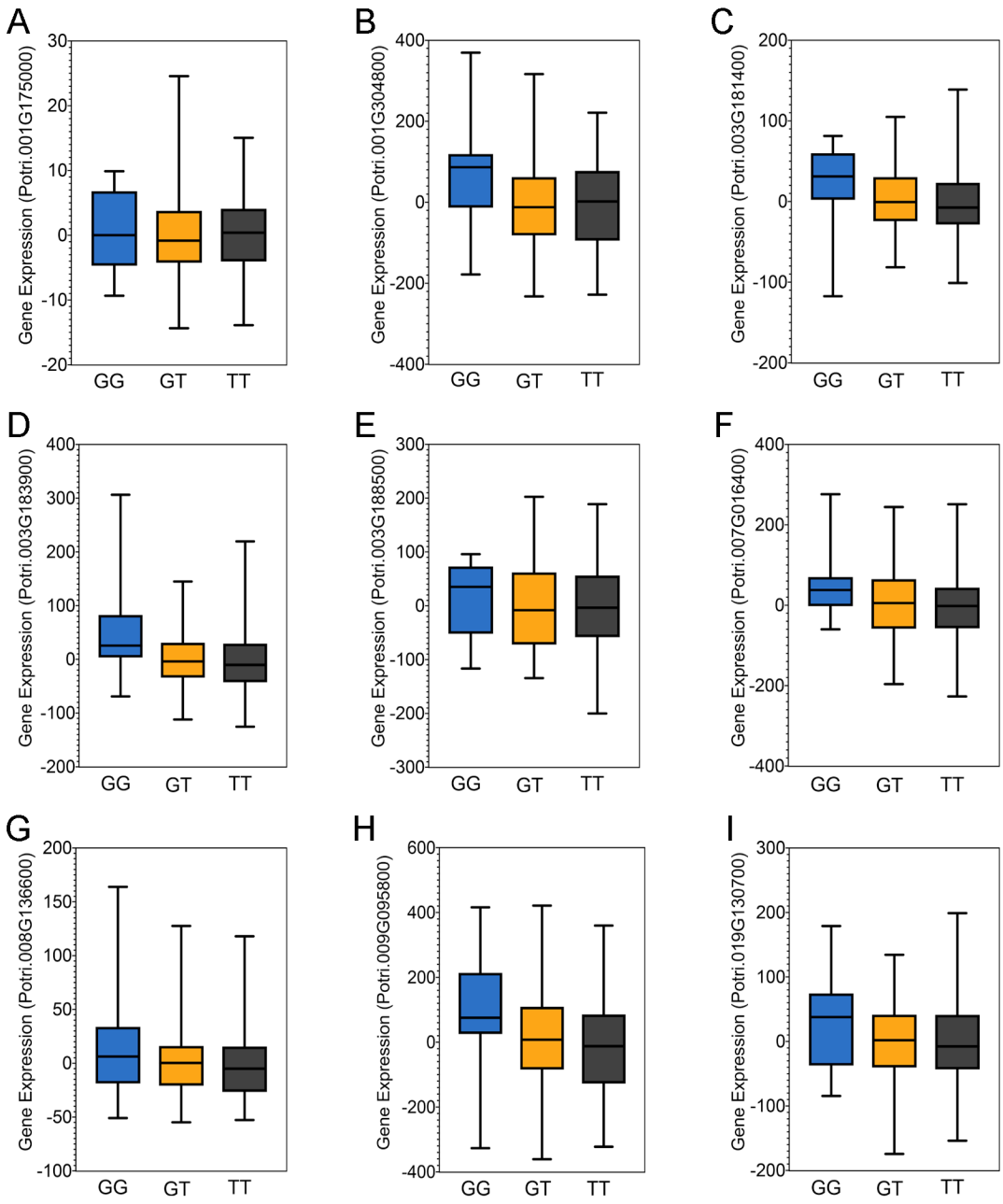

Figure S11 – Association between SNP in *MYB125* with genes of lignin biosynthesis pathway not directly connected to it in the gene expression network. Single marker regression analysis between the local-regulatory SNP of *MYB125* (with the smallest p-value) and the expression of (A) *CSE* (p-value = 0.95). (B) *CCoAOMT* (p-value = 0.07). (C) *CCR* (p-value = 0.2). (D) *shikimate O-hydroxycinnamoyltransferase* (p-value = 0.08). (E) *4CL4* (p-value = 0.2). (F) *F5H4* (p-value = 0.06). (G) *CCoAOMT* (p-value = 0.3). (H) *CAD* (p-value = 0.03). (I) *trans-cinnamate 4-monooxygenase* (p-value = 0.3). Boxplot for expression of genes from the lignin biosynthesis pathway plotted as an effect of genotypes at the SNP, with the smallest p-value, in the region of *MYB125*. The horizontal line represents the median and the vertical lines mark the range of the minimum and maximum values.

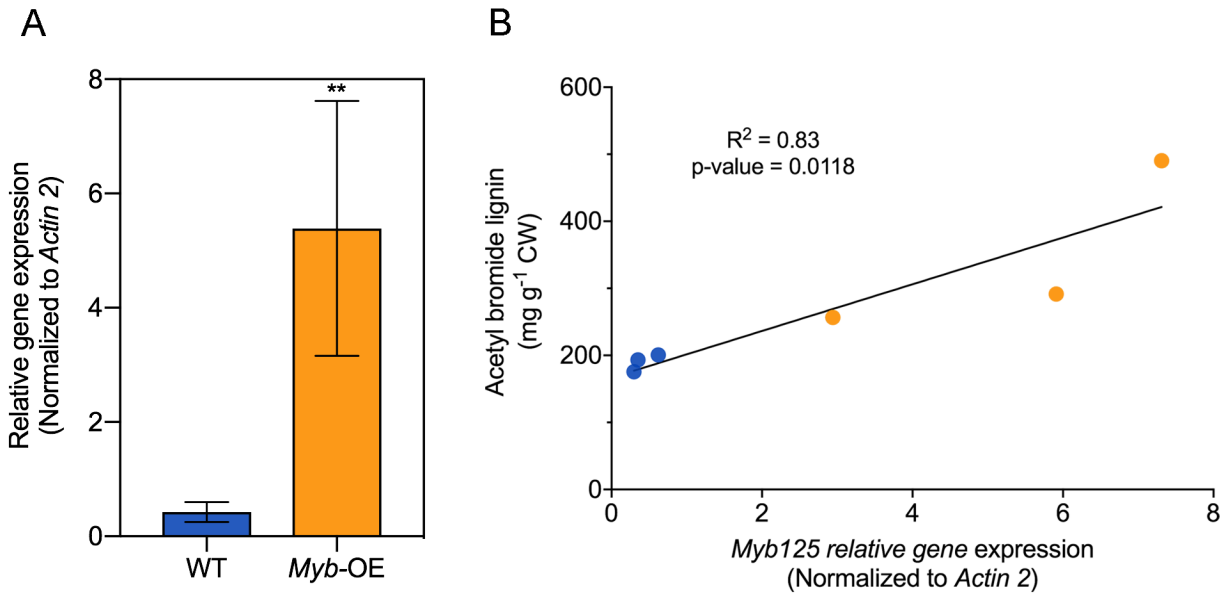

Figure S12 – Expression of *MYB125* is positively correlated with lignin content. (A) Relative expression of *MYB125* in WT and transgenic roots overexpressing *Myb125*. Relative transcript levels were quantified by RT-qPCR and normalized with the house-keeping gene *Actin2.* One tail Student's *t*-test was used to determine statistical significance; n=3. Significance is indicated by asterisks: ** p-value<0.01. (B) Pearson correlation between lignin content and *MYB125* expression. Blue dots: WT roots; orange dots: transgenic roots overexpressing *MYB125*.

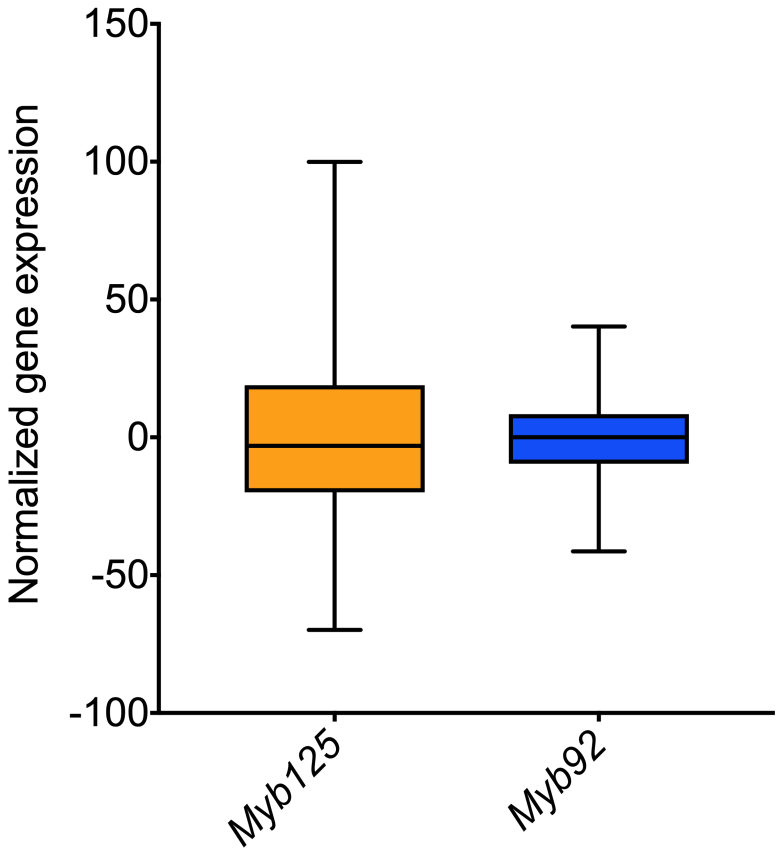

Figure S13 – Distribution of *MYB125* and *MYB92* gene expression in the *P. deltoides* population. Boxplot for expression of *MYB125* and *MYB92* genes. The horizontal line represents the median and the vertical lines mark the range of the minimum and maximum values.

Documents the liftover pipeline to convert polymorhism coordinates from one genome to another.

```
Can be done across species but is not really meant to do so.
All programs are from the Kent tools library from UCSC; 600+ programs
Loaded via: module load kent


1. make directories: old new net lift chain over blat
2. put old reference fasta in the "old" directory and the new reference fasta in the "new" directory
3. make info files for old genome: go to old directory and run faToTwoBit on the old fasta;
      run twoBitInfo on the twobit file for the old ref to get chromosome sizes:
      faToTwoBit oldref.fa oldref.2bit
      twoBitInfo oldref.2bit oldref.chromsizes
3. split new genome, make info files: go to new directory and split fasta into 3kb chunks and get needed info on the
split file using:
      faSplit -lift=new.lft size <new ref genome file name> -oneFile 3000 <naming convention of your choice for
outfile> I use name.split.fa
      faToTwoBit new.split.fa new.split.2bit

      Also need a 2bit file and chrom sizes for the whole genome:
      faToTwoBit newref.fa newref.2bit
      twoBitInfo newref.2bit newref.chromsizes

4. blat: Time consuming step #1: run blat to align the old reference to the new one:
      cd blat
      blat ../old/oldref.2bit ../new/new.split.fa oldref.psl -tileSize=12 -minScore=100 -minIdentity=98 -fastMap
      cd ..

      This will take around a day for poplar. Time probably scales with genome sizes. Blat cannot be multithreaded.
      Output file is oldref.psl; will be used downstream

5. liftup: Change the coordinates in the psl file output from 4. Current conversion coordinates are base on the split
fasta file.
      This will convert the coordinates those of the non-split (new) fasta reference. Via:
      cd lift

      liftUp -pslQ  old2new.psl <path to new>/new.lft ../blat/oldref.psl
      ### the .lft file was created in 3 when splitting the fasta file ###
      cd ..

6. chain: convert psl to chain via:
      cd chain
      axtChain -linearGap=medium -psl ../lift/old2new.psl ../old/oldref.2bit ../new/newref.2bit old2new.chain


7. chainsort: sort your new chain file, might not be necessary:
      chainSort old2new.chain old2new.chain.sorted
      cd ..
```

```
8. chainnet: identify alignable regions from chain file:
      cd net
      chainNet ../chain/old2new.chain.sorted ../old/oldref.chromsizes ../new/newref.chromsizes old2new.net /dev/null
      cd ..

9. netchainsubset (makes liftover file): Make the file you've been doing all the hard work for:
      liftOver net/old2new.net chain/old2new.chain.sorted old2new.liftover

10. liftover snp file: steps assume one is using a vcf file to start. the conversion of coordinates only works on a 3
column
      bed file. Convert vcf to 3 column bed and convert coords via:

      ml load bedops
      vcf2bed < snps.vcf > snps.bed
      awk 'BEGIN {OFS="\t"}; {print $1"\t"$2"\t"$3}' snps.bed > snps_3col.bed

      ml load crossmap
      CrossMap.py bed old2new.liftover snps_3col.bed > crossmap_lft.out

11. convert coordinates in old bed file to new locations:
      convert_coords.py -inbed <old bed file with all columns> -conversions <crossmap output file>
      output file is "updated_coords.bed"
```

```
#!/usr/bin/env python2.7
import io
import sys
import re
import argparse
from collections import defaultdict

def main():
      parser = argparse.ArgumentParser(description = 'updates old bed file with new coordinates')
      parser.add_argument('-inbed', action = 'store', type = str, required = True, help = 'old bed file')
      parser.add_argument('-conversions', action = 'store', type = str, required = True, help = 'crossmap output file')
      args = parser.parse_args()
      bedFile = args.inbed
      conversions = args.conversions
      process_files(bedFile, conversions)

def process_files(bed, conv):
      inBed = open(bed, "r")
      inConv = open(conv, "r")

      convDict = getConvInfo(inConv)
```

```
	updateCoords(inBed, convDict)

def getConvInfo(conv):
	convDict = {}
	for line in conv:
		line = line.strip()
		if re.search('Fail', line):
			continue
		else:
			linearr = line.split('->')
			for i in range(0, len(linearr)):
				linearr[i] = linearr[i].strip()
			oldPos = linearr[0].split('\t')[2]
			oldChrom = linearr[0].split('\t')[0].strip()
			oldPos = oldChrom + "\t" + oldPos
			newPos = linearr[1].split('\t')[2]
			chromo = linearr[1].split('\t')[0].strip()
			newPos = chromo + "\t" + newPos
			#print(oldPos + '\t' + newPos)
			convDict[oldPos] = newPos
	return convDict

def updateCoords(bed, convs):
	#outfile = open('updated_coords.bed', "w")
	for line in bed:
		line = line.strip()
		linearr = line.split('.', 1)
		chromo = linearr[0].split('\t')[0]
		old = linearr[0].split('\t')[2]
		old = chromo + "\t" + old

		#print(str(old))
		#print(str(convs[old]))
		#print(linearr[0] + linearr[1])
		if old in convs:
			print(convs[old] + "\t" + "." + linearr[1])
			#outfile.write(convs[old] + "\t" + "." + linearr[1] + "\n")

if __name__ == "__main__":
	main()
```

```
#Convert isoform to gene expression

#!/usr/bin/env python2.7

import sys, re ,io
from collections import defaultdict

fh = open(sys.argv[1], "r")

fpkmDict = defaultdict(dict)
transDict = defaultdict(dict)
sampleArr = []

for line in fh:
      line = line.strip()
      if(re.search('gene', line)):
            arr = line.split(',')
            for i in range(2, len(arr)):
                  sampleArr.append(arr[i])
            #print(str(len(sampleArr)))

      else:
            arr = line.split(',')
            #print(str(len(arr)))
            gene = arr[0]
            tid = arr[1]
            if(gene in transDict):
                  transDict[gene].append(tid)
            else:
                  transDict[gene] = [tid]
            for i in range(2,len(arr)):
                  fpkmDict[sampleArr[i-2]][tid] = arr[i]
fh.close()
#print(transDict["Potri.006G211100"])

isoDict = defaultdict(dict)

for geno in fpkmDict:
      for gene in transDict:
            isoVals = []
            tot = 0.0
            naCheck = ""
            for tid in transDict[gene]:
                  #print(tid)
                  #if(fpkmDict[geno][tid] != "NA"):
                        #print(fpkmDict[geno][tid])
                  if(re.search('TCONS', tid)):
```

```
                        #print(geno + '\t' + gene + '\t' + tid + '\t' + fpkmDict[geno][tid])
                        tot += float(fpkmDict[geno][tid])
                        isoVals.append(float(fpkmDict[geno][tid]))
                  #elif(fpkmDict[geno][tid] == "NA"):
                  #     naCheck = "T"
            if(naCheck != "T"):
                  for tid in transDict[gene]:
                        if(tot == 0):
                              frac = 0
                        else:
                              frac = float(fpkmDict[geno][tid]) / tot
                        isoDict[geno][gene] = tot
            elif(naCheck == "T"):
                  for tid in transDict[gene]:
                        isoDict[geno][tid] = "NA"



sys.stdout.write("gene," + str(','.join(sampleArr)) + '\n')
sys.stdout.flush()
for gene in transDict:
      #for tid in transDict[gene]:
      sys.stdout.write(gene)
      for geno in sampleArr:
            #print(geno + '\t' + gene + '\t' + tid + '\t' + str(isoDict[geno][tid]))
            sys.stdout.write(',' + str(isoDict[geno][gene]))
      sys.stdout.write('\n')
```

```
#Heritability_and_Gene_Expression

rm(list=ls())
args = commandArgs(trailingOnly = TRUE)
stopifnot(length(args)>0)
jobID=as.numeric(args[1])

## Read argumetnts jobID
permute=(jobID>1)
jobID=jobID-1

library(Matrix)
library(lattice)
library(lme4,lib.loc="/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Packages")
library(pedigreemm,lib.loc="/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Packages")

load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/SNP_Matrix_updated_coords_maf5_missing75_centered0.R
Data")
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/Experimental_design_sequential.RData")
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/matrix_fpkm_cuff_March2019_AllSamples.RData")

# matching genos with samples Z=genotype matrix, X=RNA-seq (Gene expression), SAMPLES=experimental design

samples = as.matrix(paste(SAMPLES$genotype, SAMPLES$rep, sep="_"))
rownames(SAMPLES) = samples

SAMPLES = SAMPLES[match(intersect(rownames(X),rownames(SAMPLES)),rownames(SAMPLES)),]
X = X[match(intersect(rownames(X),rownames(SAMPLES)), rownames(X)),]
Z=Z[match(names(table(as.matrix(SAMPLES$genotype))),rownames(Z)),]

stopifnot(all(rownames(X)==rownames(SAMPLES)))
p=ncol(X)

VARCOMP=matrix(nrow=ncol(X),ncol=5)
colnames(VARCOMP)=c('logLik-G','genotype','row','col','error')

SAMPLES$row=as.factor(SAMPLES$row)
SAMPLES$col=as.factor(SAMPLES$col)
SAMPLES$genotype=as.character(SAMPLES$genotype)
SAMPLES$rep=as.character(SAMPLES$rep)
SAMPLES$exp=as.factor(SAMPLES$exp)
IDs=names(table(matrix(unlist(strsplit(rownames(X),"_")),ncol = 2,byrow = T)[,1]))
GE.adjusted=matrix(nrow=length(IDs),ncol=ncol(X),NA)
rownames(GE.adjusted)=IDs
colnames(GE.adjusted)=colnames(X)
```

```
for(i in 1:nrow(VARCOMP)){
  y=as.matrix(X[,i])

  if(permute){ SAMPLES$genotype=sample(SAMPLES$genotype,replace=F,size=nrow(SAMPLES))}

  if(sum(y!=0)>=50){
    fm=lmer(y~exp+(1|row)+(1|col)+(1|genotype),data=SAMPLES)
    # Extracting variance components and log-likelihood

    tmp=as.numeric(summary(fm)@REmat[,3])
    VARCOMP[i,1]=logLik(fm)
    VARCOMP[i,2:5]=tmp

    ## Extracting BLUEs
    uHat=ranef(fm)$genotype
    eHat=residuals(fm)
    eHat=tapply(X=eHat,FUN=mean,INDEX=SAMPLES$genotype)
    tmp=match(rownames(uHat),names(eHat))
    eHat=eHat[tmp]
    stopifnot(all(names(eHat)==rownames(uHat)))
    uHat=as.vector(uHat[,1])
    eHat=as.vector(eHat)
    GE.adjusted[,i] = uHat+eHat

    if(i%%10==0){
      print(i)
    }

  }
}

setwd("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/")
if(permute){
    dir.create('permutations')
    setwd('permutations')
    save(VARCOMP,  file=paste0('VARCOMP_GE_perm_',jobID,'.RData'))
}else{
    dir.create('original')
    setwd('original')
    save(VARCOMP, GE.adjusted , file='VARCOMP_GE_AllSamples.RData')
}

setwd('/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/permutations')
load('../original/VARCOMP_GE_AllSamples.RData')
VARCOMP0=VARCOMP
H2.PERM=matrix(nrow=nrow(VARCOMP0),ncol=10000,NA)
files=list.files(pattern='VARCOMP_GE_perm')
```

```
for(i in 1:10000){
   fname=paste0('VARCOMP_GE_perm_',i,'.RData')
   if(fname%in%files){
            load(fname)
            h2=VARCOMP[,2]/rowSums(VARCOMP[,2:5])

            H2.PERM[,i]=h2
      }
      print(i)
}
rm(h2)

threshold=apply(MARGIN=1,FUN=quantile,prob=.99,X=H2.PERM,na.rm=T)
h2=VARCOMP0[,2]/rowSums(VARCOMP0[,2:5])
sum(h2>threshold,na.rm=T)
H2_Filt = as.matrix(h2[which(h2>threshold)])


save(H2_Filt,
file="/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/permutations/H2_Filtered_permutation_AllSamples.RDa
ta")

ID = colnames(GE.adjusted)
h2=as.matrix(h2)
rownames(h2)=ID

tmp=which(h2%in%H2_Filt)
h2=h2[tmp,]   ####### Genes that passed the threshold
save(h2, file="/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/permutations/genes_passed_H2_test.RData")

quit(save='no')
```

```
#eQTL_analysis

rm(list=ls())

## get args jobID
args = commandArgs(trailingOnly = TRUE)
stopifnot(length(args)==1)

chunkSize=100
jobID=as.numeric(args[1])

library(BEDMatrix, lib.loc= "/ufrc/kirst/balmant")
library(LinkedMatrix, lib.loc= "/ufrc/kirst/balmant")
library(symDMatrix,lib.loc= "/ufrc/kirst/balmant")
library(BGData, lib.loc= "/ufrc/kirst/balmant")

# Genotypes
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/SNP_Matrix_updated_coords_maf5_missing75_centered0.R
Data")

# Adjusted gene expression
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/original/VARCOMP_GE_AllSamples.RData")

ID=matrix(rownames(GE.adjusted))

tmp=which(rownames(Z)%in%ID)
Z=Z[tmp,]

tmp=which(ID%in%rownames(Z))
GE.adjusted=GE.adjusted[tmp,]
ID=matrix(rownames(GE.adjusted))

## CHUNKS
  iniGene=(jobID-1)*chunkSize+1
  endGene=min(iniGene+chunkSize-1,ncol(GE.adjusted))
  genes=iniGene:endGene
  nGenes=length(genes)

# GE in rows, SNPs in columns
OUT1=matrix(ncol=length(genes),nrow=ncol(Z),NA)
rownames(OUT1)=colnames(Z)
colnames(OUT1)=colnames(GE.adjusted)[genes]

G=tcrossprod(scale(Z,center=T,scale=F))
G=G/mean(diag(G))
EVD=eigen(G)
PC=EVD$vectors[,1:5]
```

```
setwd("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output")
dir.create('chunks')
setwd('chunks')


for( i in 1:length(genes)){
  timeIn=proc.time()[3]

  y=GE.adjusted[,genes[i]]
  useIt=mean(is.na(y))<.5
  if(useIt){
    stopifnot(all(rownames(Z)==names(y)))
    y=residuals(lm(y~PC))
    DATA=BGData(geno=Z,pheno=data.frame(y=y))
    TMP=GWAS(y~1,nCores=1,verbose=F,bufferSize=1000,nTasks=1,data=DATA,method='rayOLS')
    OUT1[,i]=TMP[,4] # stores p-values
  }
  print(c(i,proc.time()[3]-timeIn))
}

save(OUT1,file=paste0('OUT_',jobID,'.RData'))


#Combine chunks
setwd('/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/chunks')
OUT=matrix(nrow=92894, ncol=18153,NA,dimnames = list(c(1:92894),c(1:18153)))
files=list.files(pattern='OUT_AllSamples_')
chunkSize = 100
for(i in 1:length(files)){
      load(paste("OUT_",i,".RData",sep=""))
      ## CHUNKS
      iniGene=(i-1)*chunkSize+1
      endGene=min(iniGene+chunkSize-1,ncol(OUT))
      genes=iniGene:endGene
      OUT[,genes] = OUT1
      colnames(OUT)[genes] = colnames(OUT1)
      print(i)
}
rownames(OUT) = rownames(OUT1)

save(OUT, file =
"/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/eQTL_Black/chunks/OUT_pvalue_AllSamples.RData")

quit(save='no')
```

```
#Heterozygous_test

rm(list=ls())

load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/SNP_Matrix_updated_coords_maf5_missing75.RData")

getChisqOneSNP=function(x){
      n=sum(!is.na(x))
      if(n>0){
            observed=c(sum(x==0,na.rm=T),sum(x==1,na.rm=T))
            observed=c(observed,n-sum(observed))
            p=sum(observed[2]+2*observed[3])/2/n
            if(p==1|p==0){
                  chiSq=NA
            }else{
                  expected=c((1-p)^2,2*p*(1-p),p^2)*n
                  chiSq=sum(((observed-expected)^2)/expected)
            }
      }else{
            chiSq=NA
      }
      return(chiSq)
}

p=rep(NA,ncol(Z))
OH=rep(NA,ncol(Z))
Chisq=OH
for(i in 1:ncol(Z)){
      z=Z[,i]
      z[z==-1]=NA
      p[i]=mean(z,na.rm=T)/2
      OH[i]=mean(z==1,na.rm=T)
      print(i)
      Chisq[i]= getChisqOneSNP(z)
}

EH=2*p*(1-p)




HW.pvalue=pchisq(Chisq,lower.tail=F,df=1)
which(HW.pvalue<-0.00000000567)

FiltMarkers = matrix(0,nrow = nrow(HW.pvalue),ncol = 1)
FiltMarkers[which(HW.pvalue>0.01)] = 1
FiltMarkers=as.matrix(FiltMarkers)
```

```
plot(y=OH,ylim=c(0,1),xlim=c(0,1),x=EH,col=ifelse(-log10(HW.pvalue_FDR)<2,4,2),cex=.5)
abline(a=0,b=1,col='white',lwd=3)

rownames(FiltMarkers) = rownames(HW.pvalue)

save(FiltMarkers, file = "/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/Z_Filt_HW.RData")

load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/permutations/H2_Filtered_permutation.RData")
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/Z_Filt_HW.RData")
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/OUT_pvalue_GeneFilt.RData")

OUT_GeneFilt = OUT_GeneFilt[,-which(is.na(H2_Filt)==TRUE)]
OUT_GeneFilt = OUT_GeneFilt[which(FiltMarkers==1),]
OUT_FIlt_FDR = p.adjust(OUT_GeneFilt, method = "fdr")
OUT_FIlt_FDR_matrix = matrix(OUT_FIlt_FDR,ncol = ncol(OUT_GeneFilt),byrow = T)
save(OUT_FIlt_FDR_matrix, file = "/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/OUT_Filt_FDR.RData")
```

```
#Defining Cis and Trans

rm(list=ls())
library(reshape2)
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/permutations/genes_passed_H2_test.RData")
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Input/Z_Filt_HW.RData")
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/chunks/OUT_pvalue_AllSamples.RData")
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/OUT_Filt_FDR.RData")

OUT_FIlt_FDR=as.matrix(OUT_FIlt_FDR, ncol=1)
tmp=which(colnames(OUT)%in%rownames(h2))
OUT_GeneFilt=OUT[,tmp]
OUT_GeneFilt = OUT_GeneFilt[which(FiltMarkers==1),]
OUT_GeneFilt.m=melt(OUT_GeneFilt)
OUT_GeneFilt.m[,1] = gsub("scaffold_","scaffold-",OUT_GeneFilt.m[,1])

RES = data.frame(SNP_CHR = matrix(unlist(strsplit(as.character(OUT_GeneFilt.m[,1]),"_")),ncol = 2,byrow = T)[,1],
                 SNP_POS = matrix(unlist(strsplit(as.character(OUT_GeneFilt.m[,1]),"_")),ncol = 2,byrow = T)[,2],
                 GENE = OUT_GeneFilt.m[,2],
                 GENE_CHR = bed[match(OUT_GeneFilt.m[,2],bed[,4]),1],
                 GENE_POS = bed[match(OUT_GeneFilt.m[,2],bed[,4]),2],
                 P_VAL = OUT_FIlt_FDR[,1],
                 TYPE = NA)

thrs_length = 1000000000
RES[which(as.character(RES[,1])!=as.character(RES[,4])),7] = "Trans"
RES[which(as.character(RES[,1])==as.character(RES[,4]) & abs(as.numeric(RES[,5])-as.numeric(RES[,2]))<=thrs_length),7]
= "Cis"
RES[which(as.character(RES[,1])==as.character(RES[,4]) & abs(as.numeric(RES[,5])-as.numeric(RES[,2]))> thrs_length),7]
= "Trans-same-CHR"


save(RES, file="/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/RES_pvalue_1Mgb.RData")
```

```
#Multiple testing correction
rm(list=ls())

#Trans
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/RES_Trans_pvalue_1Mgb.RData")
Trans_FDR = p.adjust(Trans[,6], method = "fdr")
Trans_FDR=as.matrix(Trans_FDR, ncol=1)
colnames(Trans_FDR)="FDR"
rownames(Trans_FDR)=rownames(Trans)
Trans=cbind(Trans,Trans_FDR)
Trans=Trans[c(1,2,3,4,5,6,8,7)]
Trans_sig=Trans[which(Trans[,7]<=0.05),]

save(Trans,Trans_sig, file="/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/Results_Trans_FDR.RData")

#Cis
load("/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/RES_Cis_pvalue_1Mgb.RData")
Cis_FDR = p.adjust(Cis[,6], method = "fdr")
Cis_FDR=as.matrix(Cis_FDR, ncol=1)
colnames(Cis_FDR)="FDR"
rownames(Cis_FDR)=rownames(Cis)
Cis=cbind(Cis,Cis_FDR)
Cis=Cis[c(1,2,3,4,5,6,8,7)]
Cis_sig=Cis[which(Cis[,7]<=0.05),]

save(Cis,Cis_sig, file="/ufrc/kirst/balmant/Data_Analysis/Data_Gene_Expression/Output/Results_Cis_FDR.RData")
```

```
#Proportion of variation explained by associations in cis and trans

#This code shows how to estimate the proportion of variance explained by SNPs in Cis and Trans association for one gene
(“Potri.001G000400”)

### Package
library(BGLR)

### Loading data
load("RES_Cis_Trans_AllGenes_pvalue_1Mb.RData")  # file containing all results (all markers for all     tested genes).
This file must contain the information related to the type of association of the SNP (Cis or Trans), names of the genes
and the markers code as presented in in the last box.

load("Z_Filt_HW.RData") #Genotype matrix
load("VARCOMP_GE_AllSamples.RData") # file containing the adjusted Gene expressions

dir.create("Var_Cis_Trans")
setwd("Var_Cis_Trans")

GENE<-"Potri.001G000400"

Y<-scale(GE.adjusted[,GENE]) # Escaling and centering the gene expression for the selected gene

### Recovering the information regarding the association type between SNP and the selected gene
All<-RES[RES$GENE%in%GENE,]
All<-droplevels(All)
SNPs<-as.character(All$SNP_CHR)[grepl("scaffold",All$SNP_CHR)]
SNPs<-do.call(rbind,strsplit(SNPs,split="-"))
SNPs<-paste(SNPs[,1],SNPs[,2],sep="_")
All$SNP_CHR<-as.character(All$SNP_CHR)
All$SNP_POS<-as.character(All$SNP_POS)
All$SNP_CHR[grepl("scaffold",All$SNP_CHR)]<-SNPs
All$SNP<-paste0(All$SNP_CHR,"_",All$SNP_POS)

### Defining the BGLR parameters
burnIn=5000
nIter=35000

### Starting the analysis for the selected gene
dir.create(GENE)
setwd(GENE)

### Determining SNPs that are in Cis and Trans association with the “Potri.001G000400” gene
TRANS_SNPS<-All$SNP[All$TYPE!="Cis"]
CIS_SNPS<-All$SNP[All$TYPE=="Cis"]
nGROUP<-data.frame(nCis=length(CIS_SNPS),nTrans=length(TRANS_SNPS)) # Counting number of associations for each group
 if(length(CIS_SNPS)==0){
```

```
    TRANSt<-tcrossprod(scale(Z[,colnames(Z)%in%TRANS_SNPS],scale=F))
    G_TRANS<-TRANSt/mean(diag(TRANSt)) # G-matrix for the SNPs in Trans
          ETA<-list(TRANS=list(K=G_TRANS, model="RKHS",saveEffects=T))
          fm<-BGLR(y=Y,ETA=ETA,burnIn=burnIn,nIter=nIter,verbose=T)
    Gen_Var_Trans<- scan("ETA_TRANS_varU.dat")[-c(1:burnIn/5)]
    Error<-scan("varE.dat")[-c(1:burnIn/5)]
    herd_TRANS<-mean(Gen_Var_Trans /( Gen_Var_Trans +error)) # Reflects the proportion of the phenotypic variance
explained by Trans association for the tested gene
    Results<-data.frame(Gene=GENE,
                        Var_Trans=mean(Gen_Var_Trans),
                        Var_Cis= NA,
                        Var_Error=mean(Error),
                        h2_Trans=herd_TRANS,
                        h2_Cis=NA)}
                                 else{

    TRANSt<-tcrossprod(scale(Z[,colnames(Z)%in%TRANS_SNPS],scale=F))
    G_TRANS<-TRANSt/mean(diag(TRANSt)) # G-matrix for the SNPs in Trans
          CISt<-tcrossprod(scale(Z[,colnames(Z)%in%CIS_SNPS],scale=F))
          G_CIS<-CISt/mean(diag(CISt))
    ETA<-list(TRANS=list(K=G_TRANS, model="RKHS",saveEffects=T),
              CIS= list(K=G_CIS, model="RKHS",saveEffects=T))

     fm<-BGLR(y=Y,ETA=ETA,burnIn=burnIn,nIter=nIter,verbose=T)
    Gen_Var_Trans<- scan("ETA_TRANS_varU.dat")[-c(1:burnIn/5)]
    Gen_Var_Cis<- scan("ETA_CIS_varU.dat")[-c(1:burnIn/5)]
    Error<-scan("varE.dat")[-c(1:burnIn/5)]
    herd_TRANS<-mean(Gen_Var_Trans /( Gen_Var_Trans+ Gen_Var_Cis +error))
    herd_CIS<-mean(Gen_Var_Cis /( Gen_Var_Trans+ Gen_Var_Cis +error))

    Results<-data.frame(Gene=GENE,
                        Var_Trans=mean(Gen_Var_Trans),
                        Var_Cis= mean(Gen_Var_Cis),
                        Var_Error=mean(Error),
                        h2_Trans=herd_TRANS,
                        h2_Cis= herd_CIS)
           }

save(nGROUP,Results,fm,file="fm.RData")
```

```
#Proportion of variance explained by the top-5 principal components

#This code shows how to estimate the proportion of the variance explained by the top-5 PCs for the gene
“Potri.001G000400”
### Packages
library(lme4)
### Estimating the genomic relationship matrix
load("Z_Filt_HW.RData") #Genotype matrix
Zcent<-scale(Z,scale=F)
cross<-tcrossprod(Zcent)
G<-cross/mean(diag(cross))

### Loading experimental design and FPKM data
load("Experimental_design_sequential.RData")
load("matrix_fpkm_cuff_March2019_AllSamples.RData")
samples = as.matrix(paste(SAMPLES$genotype, SAMPLES$rep, sep="_"))
rownames(SAMPLES) = samples
SAMPLES = SAMPLES[match(intersect(rownames(X),rownames(SAMPLES)),rownames(SAMPLES)),]
SAMPLES$row<-paste0(SAMPLES$row,"_",SAMPLES$exp)
SAMPLES$col<-paste0(SAMPLES$col,"_",SAMPLES$exp)
X = X[match(intersect(rownames(X),rownames(SAMPLES)), rownames(X)),]
stopifnot(all(rownames(X)==rownames(SAMPLES)))
SAMPLES$row=as.factor(SAMPLES$row)
SAMPLES$col=as.factor(SAMPLES$col)
SAMPLES$genotype=as.factor(SAMPLES$genotype)
SAMPLES$rep=as.factor(SAMPLES$rep)
SAMPLES$exp=as.factor(SAMPLES$exp)
SAMPLES<-SAMPLES[which(as.character(SAMPLES$genotype)%in%rownames(G)),]

### Estimating RSS0 and RSS1 correcting for the 5 PCs
EVD=eigen(G)
PC=EVD$vectors[,1:5]
PC1<-data.frame()

for(i in 1:nrow(PC)){
  nrep<-nrow(SAMPLES[SAMPLES$genotype==rownames(G)[i],])
  PC1<-rbind(PC1,matrix(rep(PC[i,],each=nrep),ncol=5,byrow=F))
}
for(i in 1:5){
                SAMPLES[,paste0('PC',i)]=PC1[,i]
}


ANOVA_PC=matrix(nrow=ncol(X),ncol=3,NA)
colnames(ANOVA_PC)=c('RSS0','RSS1','R2')
```

```
y<-as.numeric(X[,"Potri.001G000400"])
SAMPLES$y<-scale(y,scale=F)
fm0=lmer(y~exp+(1|row)+(1|col),data=SAMPLES)
eHat0=residuals(fm0)
fm1=lm(eHat0~PC1+PC2+PC3+PC4+PC5,data=SAMPLES)
eHat1=residuals(fm1)
ANOVA_PC[,'RSS0']=sum(eHat0^2)
ANOVA_PC[,'RSS1']=sum(eHat1^2)
ANOVA_PC[,'R2']<-1- ANOVA_PC[i,'RSS1'] /ANOVA_PC[i,'RSS0']
ANOVA_PC
```

```
#Proportion of variance explained by the MYB215 cis associated SNPs on other 9 genes and lignin content

### Packages
library(BGLR)
library(coda)

load("Z_Filt_HW.RData") #Genotype matrix
load("VARCOMP_GE_AllSamples.RData") # file containing the adjusted Gene expressions


dir.create("VarSNPS_MYB215")
setwd("VarSNPS_MYB215")

G<-c("Potri.001G036900","Potri.003G059200","Potri.005G117500","Potri.006G033300",
"Potri.008G038200","Potri.009G099800","Potri.010G224100","Potri.012G006400","Potri.013G157900")

### Identifying on the Z matrix the markers associated to the MYB215

Loc<-c(which(grepl("13769520",colnames(Z))==T),which(grepl("13769383",colnames(Z))==T))

SNPSS<-Z [,Loc] # SNPs associated to the MYOB215
Zsel<-Z [,-Loc] # Z matrix withou MYB215 markers


GENES<-GE.adjusted[,colnames(GE.adjusted)%in%G]

library(BGLR)
burnIn=10000
nIter=60000

Z<-scale(Zsel,scale=F)
     ZZ<-tcrossprod(Z)
Gzz<-ZZ/mean(diag(ZZ))
EVD<-eigen(Gzz)
PC=EVD$vectors[,1:5]

ETA<-list(Significative=list(X=SNPSS,model="FIXED",saveEffects=TRUE),
          OtherSNPS=list(K=Gzz,model="RKHS"))

### Analysis for the gene "Potri.001G036900"

  dir.create("Potri.001G036900")
  setwd("Potri.001G036900")
  y=GENES[,"Potri.001G036900"]
  yadj<-residuals(lm(y~PC[,1]+PC[,2]+PC[,3]+PC[,4]+PC[,5]))
  fm<-BGLR(y=scale(yadj),burnIn=burnIn,nIter=nIter,ETA=ETA,verbose=T)
  betas<-read.table("ETA_Significative_b.dat",h=T)[-c(1:(burnIn/5)),]
```

```
VarOtherSNPs<- scan("ETA_OtherSNPS_varU.dat")[-c(1:(burnIn/5))]
Error<-scan("varE.dat")[-c(1:(burnIn/5))]
xbeta<-do.call(rbind,apply(betas,1,function(x) data.frame(
                                  VarSNP1=var(SNPSS[,1]*x[1]),
                                  VarSNP2=var(SNPSS[,2]*x[2]),
                                  VarSum=var(SNPSS[,1]*x[1]+SNPSS[,2]*x[2]))))

ProportionGenVar<-xbeta$VarSum/(xbeta$VarSum+VarOtherSNPs)
h2<-(xbeta$VarSum+VarOtherSNPs) /(xbeta$VarSum+VarOtherSNPs+Error)
MCMC<-as.mcmc(ProportionGenVar)
Estimates<-round(summary(MCMC)[[1]],digits=4)
Interval<-round(c(Estimates[1]+qnorm(c(0.025))*Estimates[3],
            Estimates[1]+qnorm(c(0.975))*Estimates[3]),digits=4)

mcmch2<-as.mcmc(h2)
h2est<-round(summary(mcmch2)[[1]],digits=4)

Intervalh2<-round(c(h2est[1]+qnorm(c(0.025))*h2est[3],h2est[1]+qnorm(c(0.975))*h2est[3]),digits=4)
VarSD<-data.frame(Trait="Potri.001G036900",
               ProportionGeneticvar=t(paste0(Estimates[1],"(",Interval[1],"-",Interval[2],")")),
                                  h2=t(paste0(h2est[1],"(",Intervalh2[1],"-",Intervalh2[2],")")))
Proportions<-rbind(Proportions,VarSD)
```