Lab Report

Joel R. Herrera

Updated to: $04 March, 2022$

Contents


```
library(dplyr)
library(knitr)
library(kableExtra)
library(fastqcr)
library(ggplot2)
library(factoextra)
```

```
library(VennDiagram)
library(DESeq2)
library(readxl)
library(seqinr)
library(stringr)
library(M3C)
library(xlsx)
library(edgeR)
library(orthologr)
library(clusterProfiler)
library(org.At.tair.db)
library(DOSE)
library(forcats)
```

1 Workflow overview

1.1 Up-regulated genes in meristem

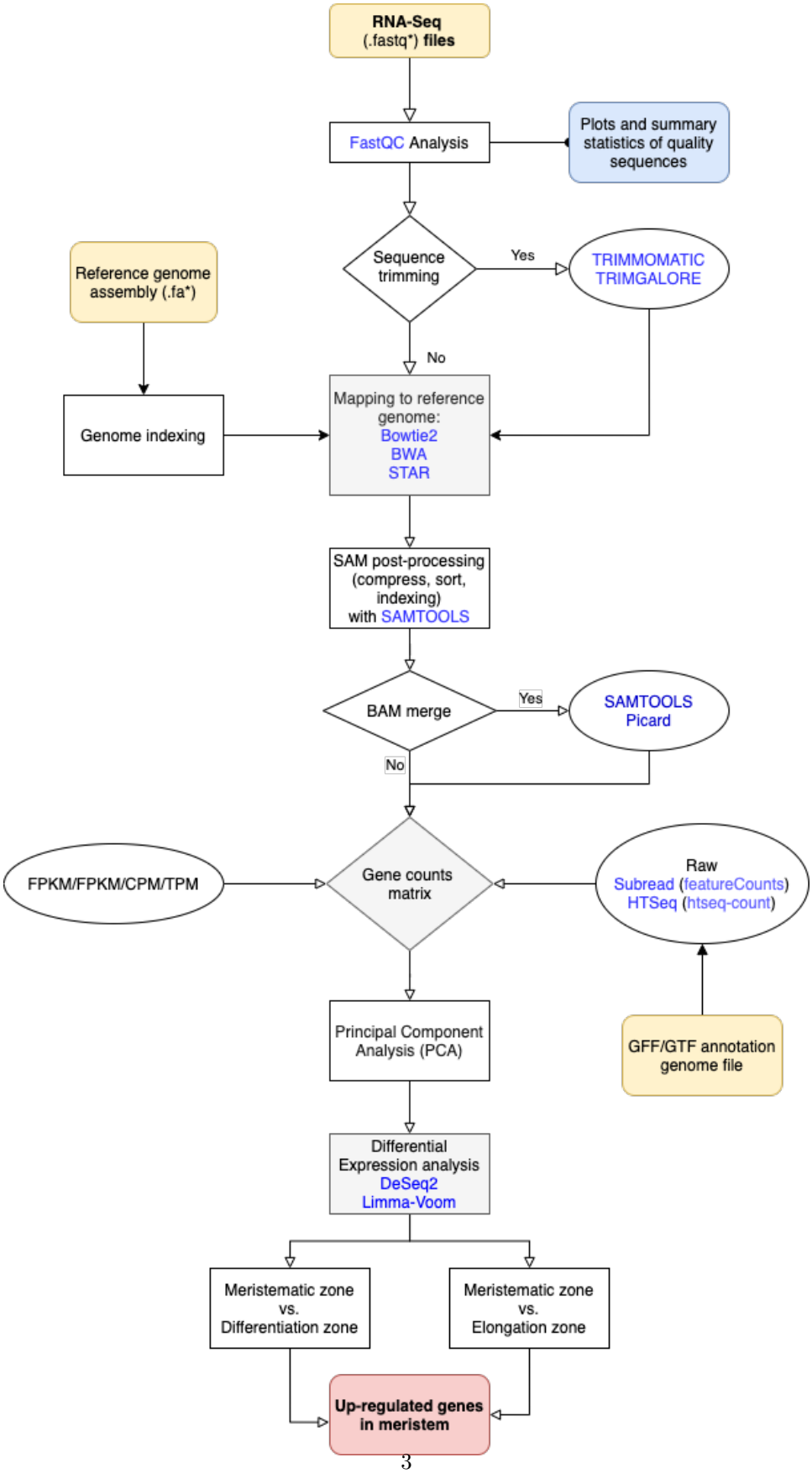


Figure 1: Workflow of the first part.

1.2 Global functions

```
pca_go <- function(pca_contrib,zone, samp=100){
  pca_f <- rownames(pca_contrib)[1:samp]
  ego_f <- enrichGO(gene = pca_f, keyType = "TAIR", OrgDb = org.At.tair.db, ont = "BP",
↪  pAdjustMethod = "BH", pvalueCutoff = 0.05, qvalueCutoff = 0.05)
  ego_f <- mutate(ego_f,
                  log2foldEnrichment = log2((as.numeric(sub("/\\d+", "", GeneRatio)) /
                  ↪  as.numeric(sub("\\d+/", "", GeneRatio))) / (as.numeric(sub("/\\d+",
                  ↪  "", BgRatio)) / as.numeric(sub("\\d+/", "", BgRatio)))))

    plot <- ggplot(ego_f, showCategory = 10, aes(log2foldEnrichment,
↪  fct_reorder(Description, log2foldEnrichment))) +
  geom_segment(aes(xend=0, yend = Description)) +
  geom_point(aes(color=p.adjust, size = Count)) +
  scale_color_gradientn(colours=c("#f7ca64", "#46bac2", "#7e62a3"),
  trans = "log10",
  guide=guide_colorbar(reverse=TRUE, order=1)) +
  scale_size_continuous(range=c(2, 10)) +
  theme_dose(12) +
  xlab("Log2(Fold-Enrichment)") +
  ylab(NULL) +
  ggtitle(paste("BP GO enriched for", zone)) +
      scale_y_discrete(labels = function(x) str_wrap(x, 45))
  theme(axis.text.y = element_text(size = 10))

    return(plot)
}
```

```
plotExpVar <- function(exp_var, org){
  color <- grDevices::colors()[grep('gr(a|e)y', grDevices::colors(), invert = T)]
  color <- sample(color, length(exp_var$nPC), replace = T)
  plot <- ggplot(exp_var) +
    geom_bar(aes(x = nPC, y = exp_var), stat = "identity", fill = color) +
    geom_line(aes(x = nPC, y = var_cum)) +
    geom_point(aes(x = nPC, y = var_cum)) +
    scale_x_continuous(breaks = 1:length(exp_var$nPC)) +
    geom_hline(yintercept = 90, color = 'Green') +
    annotate(geom = "text", x=length(exp_var$nPC), y=93, label="90%", size=3) +
    geom_hline(yintercept = 75, color = 'Blue') +
    annotate(geom = "text", x=length(exp_var$nPC), y=78, label="75%", size=3) +
    geom_hline(yintercept = 50, color = 'Red') +
    annotate(geom = "text", x=length(exp_var$nPC), y=53, label="50%", size=3) +
    labs(title = paste("Explained variance by each PC for", org,"dataset"), x =
    ↪  "Principal Component (PC)", y = "Explained variance (%)") +
    geom_text(aes(label=paste0(round(exp_var, 0), "%"), x=nPC, y=exp_var), vjust=-0.5,
    ↪  size=3)

  return(plot)
  }
```

```
plotPCA <- function(pca_object, exp_var){
  PC <- as.data.frame(pca_object$x)
  PC$Sample <- rep(c('MZ', 'EZ', 'DZ'), each = 3)
```

```
  plot <- ggplot(PC, aes(x = PC1, y = PC2, color = Sample)) +
    geom_point() +
    labs(title = "Principal Component Analysis",
         x = paste0("PC1: ", round(exp_var$exp_var[1]), "% Variance"),
         y = paste0("PC2: ", round(exp_var$exp_var[2]), "% Variance"), color = "Sample
         ↪  zone")
  return(plot)
}
```

```
expVar <- function(pca_object){
  exp_var <- pca_object$sdev^2
  exp_var <- (exp_var / sum(exp_var)) * 100 # Proporcion de la varianza explicada
  nPC <- 1:length(pca_object$sdev) # Numero de componentes
  exp_var <- data.frame(exp_var, nPC)
  exp_var$var_cum <- cumsum(exp_var$exp_var)
  return(exp_var)
}
```

```
orts <- function(ref, orts){
  orts$query.id.gene <- sapply(orts$query_id, function(id) substr(id, 1, nchar(id)-2))
  orts$subject.id.gene <- sapply(orts$subject_id, function(id) substr(id, 1,
↪  nchar(id)-2))
  temp <- sapply(ref, function(id){
    t <- orts$query.id.gene[orts$subject.id.gene == id]
    if(length(t) == 0) return(NA)
    else return(t[1])
    })
  return(temp)
}
```

```
filt_blasp <- function(cl){
  c <- lapply(unique(cl$query.id), function(query){
    temp <- cl[cl$query.id == query,]
    return(temp[which.min(temp$evalue),])
  })
  filt <- c[[1]]
  for(row in c[2:length(c)]){
    filt <- rbind(filt, row)
  }
  return(filt)
}
```

```
vplot <- function(res, f1, f2, xl=8, yl=20){

  res <- na.omit(as.data.frame(res))

  res = mutate(res, sig=ifelse(res$padj < 0.05, "Sig", "Not DE"))
  res$sig[res$sig == "Sig" & res$log2FoldChange > log2(1.5)] <- "Up"
  res$sig[res$sig == "Sig" & res$log2FoldChange < -log2(1.5)] <- "Down"
  res$id <- rownames(res)

  plot <- ggplot(res, aes(x=log2FoldChange, y=-log10(padj))) +
  geom_point(aes(col=sig), alpha = 0.4) +
```

```
  geom_text(data=res %>% filter(log2FoldChange > 3 & padj < 0.01),
            aes(label=id), check_overlap = T, na.rm = T, size = 2.8) +
  theme_minimal() +
  scale_color_brewer(palette="Accent") +
  geom_hline(yintercept=-log10(0.05),linetype = "longdash", col="red") +
  annotate(geom = "text", x=-xl+2, y=2, label="-log10(p = 0.05)", color="red") +
  geom_vline(xintercept=c(-log2(1.5), log2(1.5)), linetype = "dashed") +
  annotate(geom = "text", x=log2(1.5)+0.5, y=yl-2, label="log2(1.5)", angle=270) +
  scale_y_continuous(breaks = seq(0,yl,5), limits = c(0,yl)) +
  scale_x_continuous(breaks = seq(-xl,xl, 2), limits = c(-xl,xl)) +
  labs(title = paste("Differential Expression:", f1, "compared with", f2),
       x = "Log2(Fold-Change)",
       y = "-Log10(P-adjusted value)", color = "Expression")

  return(plot)
}
```

```
# this function is for normalize the raw counts for PCA visualization

rlog_norm <- function(counts, metadata, minCounts = 1, NminSamples = 3){
  x <- counts[ which( apply( cpm(DGEList(counts = counts)), 1,
                             function(y) sum(y>=minCounts)) >= NminSamples), ]
  x <- DESeqDataSetFromMatrix(countData=x, colData = metadata, design=~dex)
  x <- estimateSizeFactors(x)
  x <- rlog(x, blind=TRUE)
  x <- t(assay(x))
  PCA <- prcomp(x)
  return(PCA)
}
```

```
wfasta <- function(seq, id, region, start, end, strand, len, out_file){
  name <- paste(paste(id, region, sep = "_"), " from", -start, "to", -end, "Strand:",
↪  strand, "Length:", len)
  write.fasta(sequences = seq,
                names = name,
                open = "a",
                file.out = out_file)
  return(name)
}
```

```
retrieve_seq <- function(id, seq_file, ngb_len, input_df, out_file){
  seq <- read.fasta(seq_file)
  names(seq) <- sapply(names(seq), function(nam) str_split(nam, pattern = "\\|")[[1]][1])
  start <- input_df$start[input_df$gen_id == id]
  end <- input_df$end[input_df$gen_id == id]
  len <- length(seq[id][[1]])+1
  strand <- input_df$strand[input_df$gen_id == id]
  names(seq) <- sapply(names(seq), function(x) str_split(x, pattern = "\\|")[[1]][1])
  seq <- seq[id][[1]]

  new_end <- (abs(len+start))-1

  if(new_end < ngb_len) slide <- new_end
```

```
  else slide <- ngb_len

  new_start <- (abs(len+start))-slide

  new_start2 <- (abs(len+end))+1

  if((len - new_start2) < ngb_len) slide2 <- len - new_start2
  else slide2 <- ngb_len

  new_end2 <- (abs(len+end)) + slide2


  if(strand == "+"){
     seq_up <- seq[new_start:new_end]
     seq_down <- seq[new_start2:new_end2]
  }else{
    seq <-comp(seq)
    seq_up <- rev(seq[new_start:new_end])
    seq_down <- rev(seq[new_start2:new_end2])
  }

  if(length(seq_up) >= 3){
    sup <- wfasta(seq_up, id, "upstream", len-new_start, len-new_end, strand,
↪  length(seq_up), out_file)
    }else sup <- NA
  if(length(seq_down) >= 3){
    dup <- wfasta(seq_down, id, "downstream", len-new_start2, len-new_end2, strand,
↪  length(seq_down), out_file)
    }else dup <- NA

  return(c(sup,dup))
}
```

```
read_sea <- function(sea_file, sea_seq_file, seqs_dframe){

  sea <- read.table(sea_file, sep="\t", header = 1)
  sea_seq <- read.table(sea_seq_file, sep="\t", header = 1)

  sea_seq <- sea_seq[sea_seq$seq_Class == "tp",]

  seq_match <- sapply(sea$ID, function(id){
    seqs <- sea_seq$seq_ID[sea_seq$motif_ID == id]
    return(paste(seqs, collapse = ","))
  })


  sea <- cbind(sea[,c(1,3,4,5,12)], seq_match)

  gen_id <- sapply(sea_seq$seq_ID, function(x) str_split(x, pattern = "_")[[1]][1])
  region <- sapply(sea_seq$seq_ID, function(x) str_split(x, pattern = "_")[[1]][2])
  family <- sapply(sea_seq$motif_ID, function(x) str_split(x, pattern = "_")[[1]][1])
  sea_seq <- cbind(sea_seq, gen_id, region, family)
```

```
  region_ngb <- sapply(seqs_dframe$gen_id, function(gen){
    rgb <- paste(sea_seq$region[sea_seq$gen_id == gen], collapse = ";")
    if(nchar(rgb) == 0) rgb <- NA
    return(rgb)
  })

  motif_ngb <- sapply(seqs_dframe$gen_id, function(gen){
    rgb <- paste(sea_seq$motif_ALT_ID[sea_seq$gen_id == gen], collapse = ";")
    if(nchar(rgb) == 0) rgb <- NA
    return(rgb)
  })

  family_ngb <- sapply(seqs_dframe$gen_id, function(gen){
    rgb <- paste(sea_seq$family[sea_seq$gen_id == gen], collapse = ";")
    if(nchar(rgb) == 0) rgb <- NA
    return(rgb)
  })

  seqs_dframe <- cbind(seqs_dframe, region_ngb, family_ngb, motif_ngb)

  return(seqs_dframe)
}
```

```
gseDEGs <- function(res){
  res_enrich <- res$log2FoldChange
  names(res_enrich) <- rownames(res)
  res_enrich <- na.omit(res_enrich)
  res_enrich <- sort(res_enrich, decreasing = T)

  res_gse <- gseGO(geneList = res_enrich, ont = "BP", keyType = "TAIR", OrgDb =
↪   org.At.tair.db, pvalueCutoff = 0.05, seed = T, verbose = T, nPermSimple = 1000, eps =
↪   0, minGSSize = 10, maxGSSize = 10000)

  p <- dotplot(res_gse, showCategory=10, split=".sign") +
    facet_grid(.~.sign)
 return(p)
}
```

2 Arabidopsis thaliana

2.1 FASTQC Quality analysis

2.1.1 Summary

Table 1: Quality report for fastq files before cleaning.

	1	2	3	4	5	6	7	8	9
Basic Statistics	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Per base sequence quality	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Per tile sequence quality	WARN	WARN	PASS	WARN	WARN	PASS	WARN	WARN	PASS
Per sequence quality scores	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Per base sequence content	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL
Per sequence GC content	PASS	PASS	WARN	PASS	PASS	PASS	PASS	PASS	PASS
Per base N content	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS

Sequence Length Distribution	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Sequence Duplication Levels	WARN	FAIL	FAIL	WARN	FAIL	FAIL	WARN	FAIL	FAIL
Overrepresented sequences	WARN	WARN	FAIL	WARN	WARN	FAIL	WARN	WARN	WARN
Adapter Content	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS

Files: [1] SRR1740401 [2] SRR1740402 [3] SRR1740403 [4] SRR1740404 [5] SRR1740405 [6] SRR1740406 [7] SRR1740407 [8] SRR1740408 [9] SRR1740409

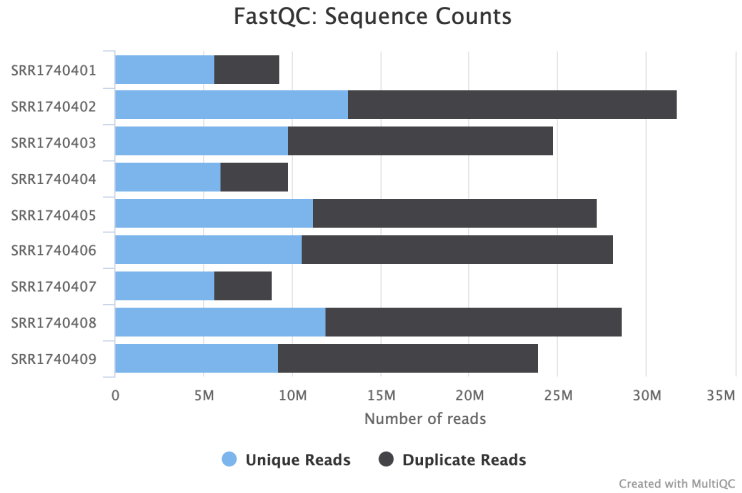


Figure 2: MultiQC: reads in files.

2.1.2 Plots

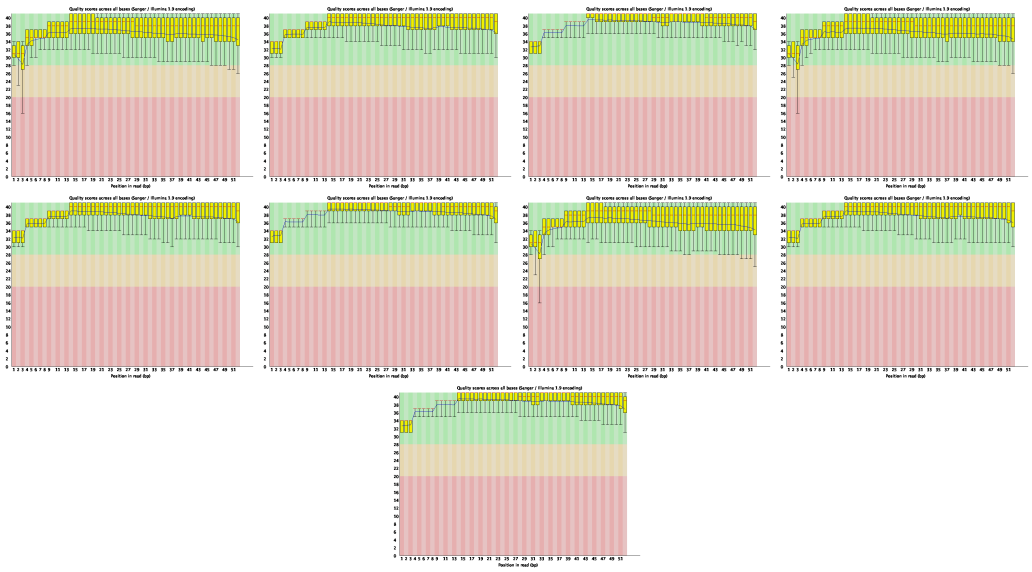


Figure 3: Per Base Sequence Quality

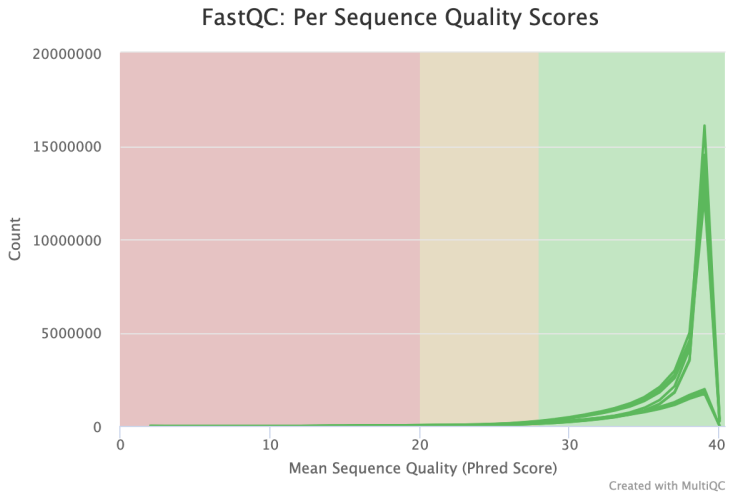


Figure 4: MultiQC Per Sequence Quality Scores.

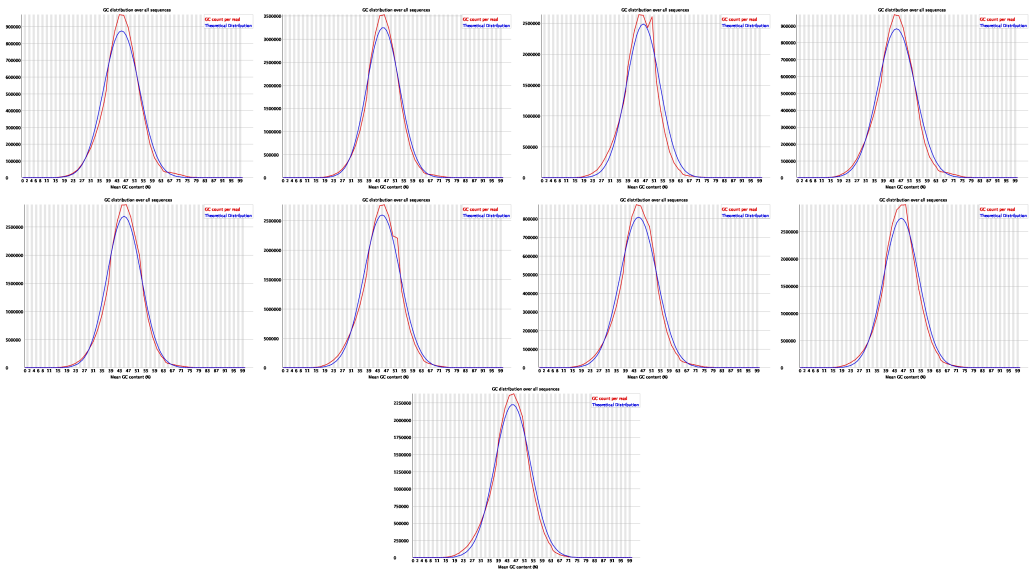


Figure 5: Per Sequence GC Content

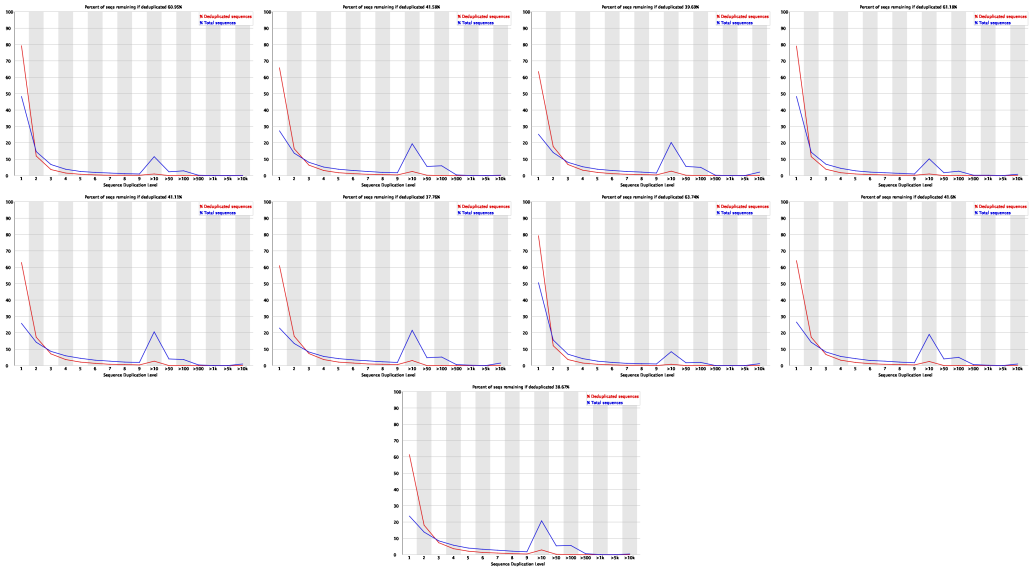


Figure 6: Sequence duplication levels

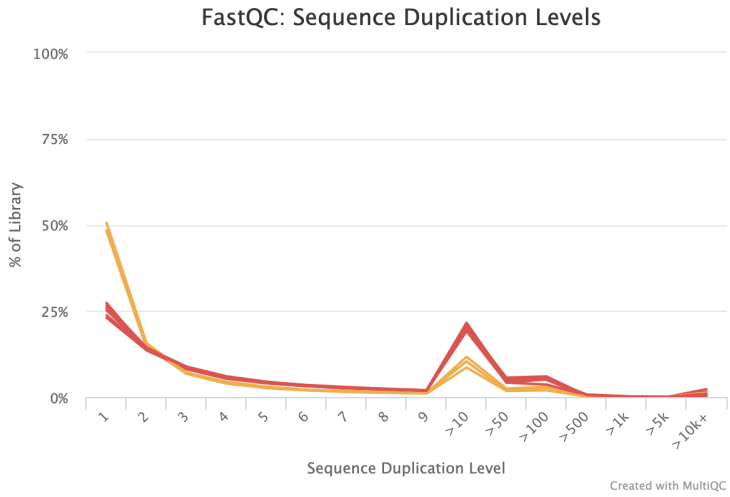


Figure 7: MultiQC Sequence Duplication Levels.

Table 2: Quality report for fastq files after cleaning.

	1	2	3	4	5	6	7	8	9
Basic Statistics	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Per base sequence quality	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Per tile sequence quality	WARN	WARN	PASS	WARN	WARN	PASS	WARN	WARN	PASS
Per sequence quality scores	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Per base sequence content	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL	FAIL
Per sequence GC content	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Per base N content	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Sequence Length Distribution	WARN	WARN	WARN	WARN	WARN	WARN	WARN	WARN	WARN
Sequence Duplication Levels	WARN	FAIL	FAIL	WARN	FAIL	FAIL	WARN	FAIL	FAIL
Overrepresented sequences	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS
Adapter Content	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS	PASS

Files: [1] SRR1740401 [2] SRR1740402 [3] SRR1740403 [4] SRR1740404 [5] SRR1740405 [6] SRR1740406 [7] SRR1740407 [8] SRR1740408 [9] SRR1740409

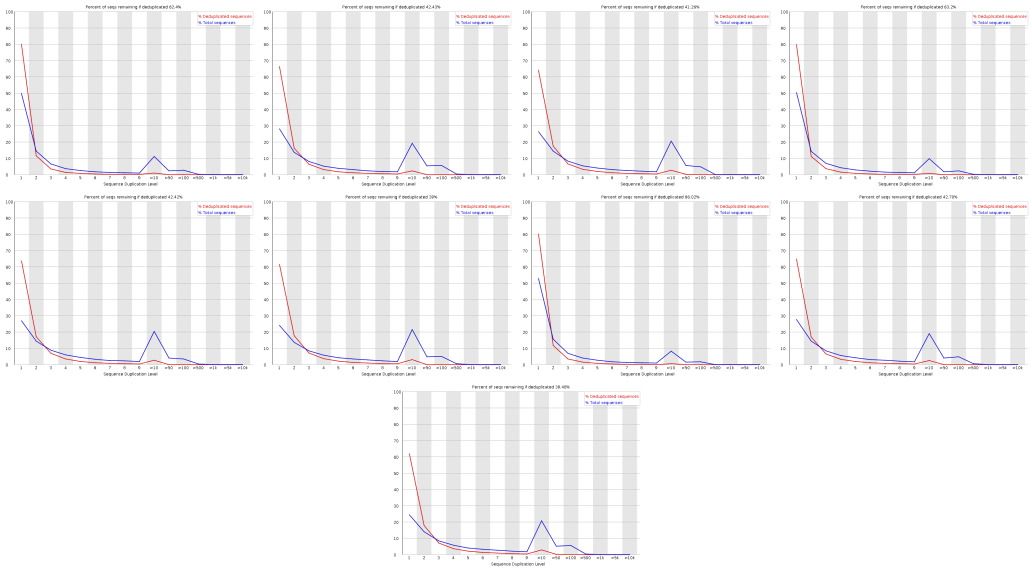


Figure 8: Sequence duplication levels

2.1.2.0.1 FastQC

2.2 Count matrix

Table 3: Count matrix

gene_id	MZ_rep1	MZ_rep2	MZ_rep3	EZ_rep1	EZ_rep2	EZ_rep3	DZ_rep1	DZ_rep2	DZ_rep3
AT1G01010	33	97	97	110	128	1035	595	1681	1536
AT1G01020	1144	4062	3457	1075	4019	4121	957	4131	3370
AT1G01030	15	61	62	23	76	58	19	45	14
AT1G01040	215	919	636	278	783	1078	372	1289	887
AT1G01050	910	3248	2834	1564	4907	3935	938	3042	2083
AT1G01060	1305	4987	4127	1133	3700	3784	1532	4290	3912

Note: Only showed first 6 levels of:

```
length(rownames(count_matrix))
```

```
## [1] 21200
```

```
write.csv(count_matrix, file = "ara/ara_count.csv", row.names = F)

rownames(count_matrix) <- count_matrix$gene_id
count_matrix <- count_matrix[,-1]

colnames(count_matrix) <- c('ara_MZ_rep1', 'ara_MZ_rep2', 'ara_MZ_rep3', 'ara_EZ_rep1',
↪  'ara_EZ_rep2', 'ara_EZ_rep3','ara_DZ_rep1', 'ara_DZ_rep2', 'ara_DZ_rep3')
```

2.3 PCA

```
metadata <- data.frame(
  dex = as.factor(rep(c('MZ', 'EZ', 'DZ'), each = 3)),
  sample = rep.int(c(1,2,3), 3)
)
rownames(metadata) <- colnames(count_matrix)
metadata
```

2.3.0.1 Metadata

```
##             dex sample
## ara_MZ_rep1  MZ      1
## ara_MZ_rep2  MZ      2
## ara_MZ_rep3  MZ      3
## ara_EZ_rep1  EZ      1
## ara_EZ_rep2  EZ      2
## ara_EZ_rep3  EZ      3
## ara_DZ_rep1  DZ      1
## ara_DZ_rep2  DZ      2
## ara_DZ_rep3  DZ      3
```

```
pca_ara <- rlog_norm(count_matrix, metadata)
exp_var_ara <- expVar(pca_ara)
```

```
plotPCA(pca_ara, exp_var_ara)
```

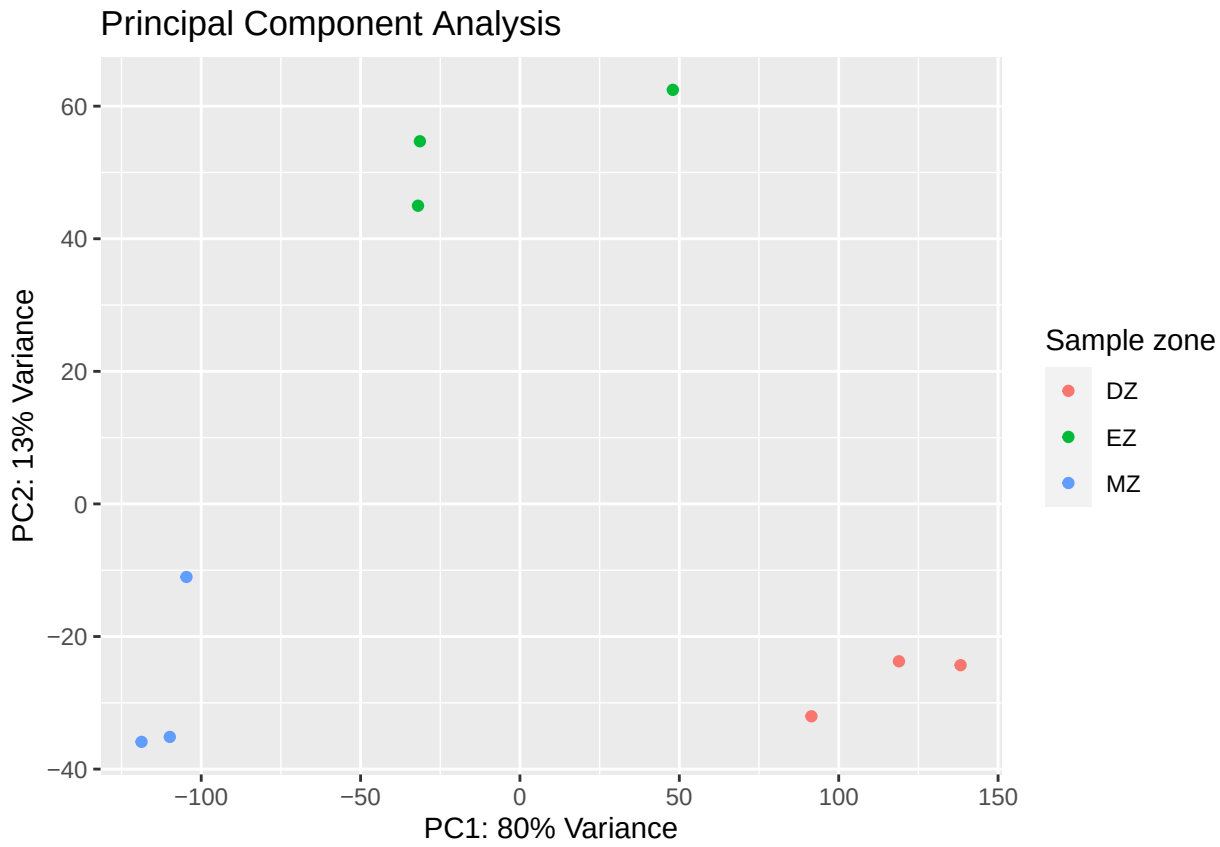
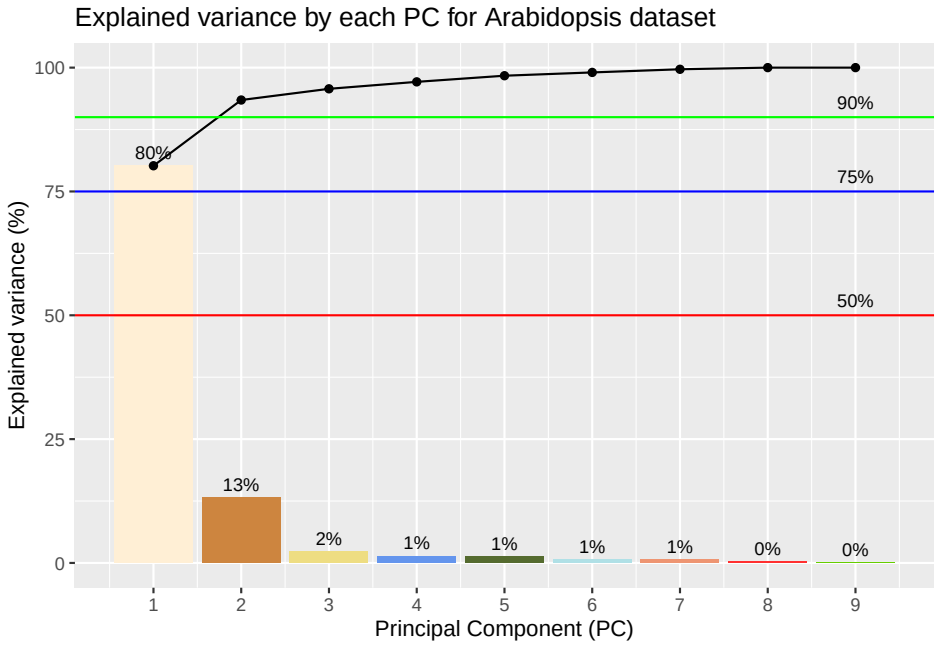


Figure 9: First two dimensions of PCA.

2.3.1 Proportion of explained variance

```
plotExpVar(exp_var_ara, "Arabidopsis")
```



2.3.2 Contribution of variables to PCA

```
var_ara <- get_pca_var(pca_ara)
ara_pca_contrib <- as.data.frame(var_ara$contrib[order(var_ara$contrib[,1],
↪  var_ara$contrib[,2],decreasing = TRUE),c(1,2)])
colnames(ara_pca_contrib) <- c('Dim.1.Contrib', 'Dim.2.Contrib')
varcor <- as.data.frame(var_ara$cor)
ara_pca_contrib <- cbind(ara_pca_contrib, do.call(rbind,
↪  lapply(rownames(ara_pca_contrib), function(gen){
  return(varcor[which(rownames(varcor) == gen), c(1,2)])
})))
```

```
samplesNames <- rep(c('MZ', 'EZ', 'DZ'), each = 3)
fviz_pca_biplot(pca_ara,
                geom.ind = "point",
                fill.ind = samplesNames, pointsize = 5,
                pointshape = 21, repel = TRUE, select.var = list(contrib = 10),
                col.var = "contrib") +
 scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Variables most contributing to the variance of samples",
       x = paste("PC1 ", "(", round(exp_var_ara$exp_var[1], 2), "%)", sep =""),
       y = paste("PC2 ", "(", round(exp_var_ara$exp_var[2], 2), "%)", sep =""),
       color = "Contribution", fill = "Zone")

fviz_pca_ind(pca_ara, pointsize = "cos2",
             pointshape = 21, fill = "#E7B800",
             repel = TRUE)
```

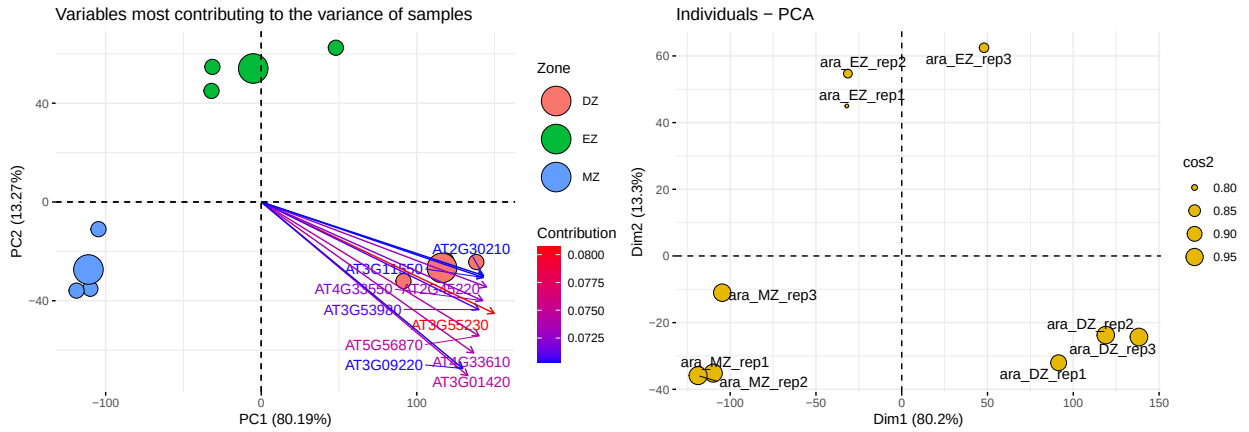


Figure 10: Distribution of indiviudals and variables in PCA by their qualities of representation: cos2, ontribution and coordinates

2.3.3 Enrichment BP GO per cluster

```
fviz_pca_biplot(pca_ara,
                geom.ind = "point",
                fill.ind = samplesNames, pointsize = 5,
                pointshape = 21, repel = TRUE,
                select.ind = list(name = c("ara_DZ_rep1", "ara_DZ_rep2", "ara_DZ_rep3")),
                ↪  select.var = list(contrib = 10),
                col.var = "contrib") +
 scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Variables most contributing to the variance of DZ",
       x = paste("PC1 ", "(", round(exp_var_ara$exp_var[1], 2), "%)", sep =""),
       y = paste("PC2 ", "(", round(exp_var_ara$exp_var[2], 2), "%)", sep =""),
       color = "Contribution", fill = "Zone")  +
    theme(legend.position="bottom", legend.text = element_text( size=6)) +
    ↪  coord_fixed(ratio = 3/2)
```

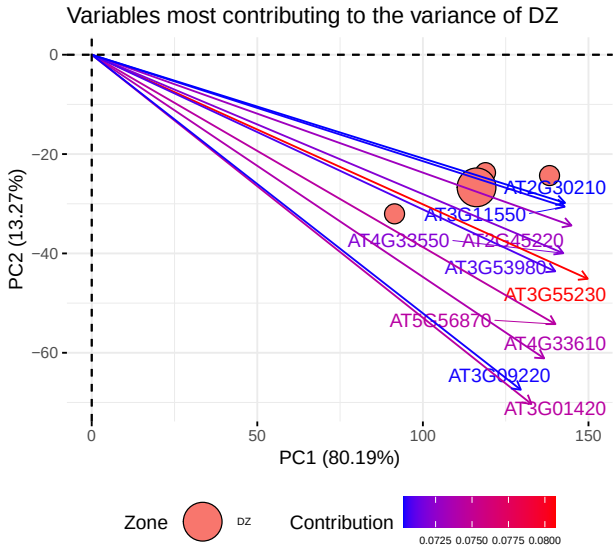


Figure 11: Variables representation per sample cluster

```
pca_go(ara_pca_contrib[ara_pca_contrib$Dim.1 > 0 & ara_pca_contrib$Dim.2 < 0, ], "DZ",
↪  100)
```

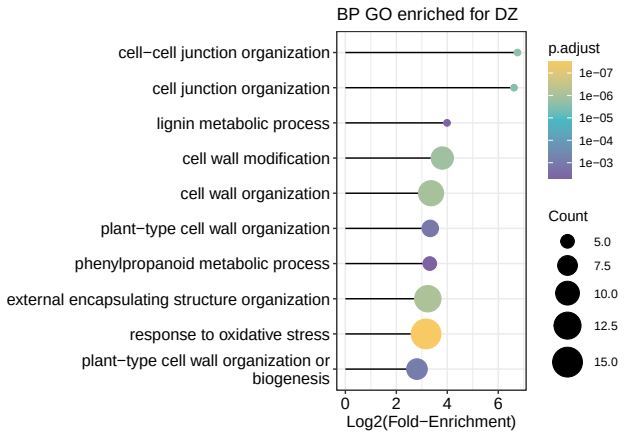


Figure 12: Enrichment of 100 variables most representative per sample cluster

```
fviz_pca_biplot(pca_ara,
                geom.ind = "point",
                fill.ind = samplesNames, pointsize = 5,
                pointshape = 21, repel = TRUE,
                select.ind = list(name = c("ara_MZ_rep1", "ara_MZ_rep2", "ara_MZ_rep3")),
                ↪  select.var = list(contrib = 550),
                col.var = "contrib") +
  lims(x = c(-120,0), y = c(-80,0)) +
 scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Variables most contributing to the variance of MZ",
       x = paste("PC1 ", "(", round(exp_var_ara$exp_var[1], 2), "%)", sep =""),
       y = paste("PC2 ", "(", round(exp_var_ara$exp_var[2], 2), "%)", sep =""),
       color = "Contribution", fill = "Zone") +
    theme(legend.position="bottom", legend.text = element_text( size=8)) +
    ↪  coord_fixed(ratio = 3/2.5)
```

```
## Warning: Removed 540 rows containing missing values (geom_text_repel).
```

```
## Warning: Removed 540 rows containing missing values (geom_segment).
```

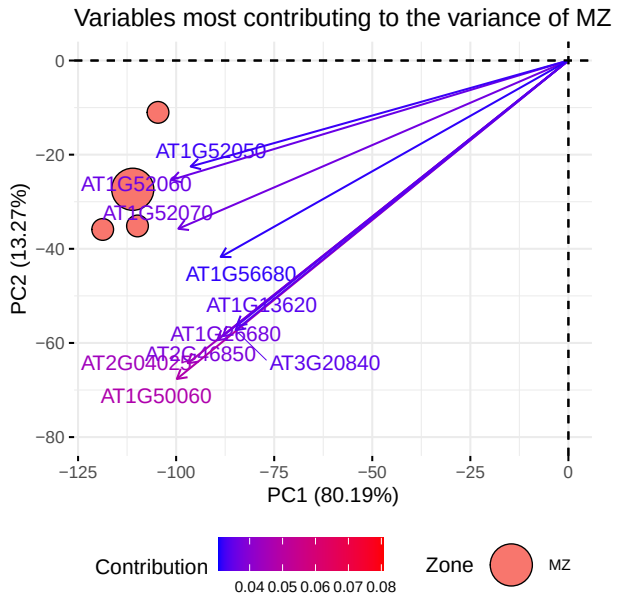


Figure 13: Variables representation per sample cluster

```
pca_go(ara_pca_contrib[ara_pca_contrib$Dim.1 < 0 & ara_pca_contrib$Dim.2 < 0, ], "MZ",
↪  100)
```

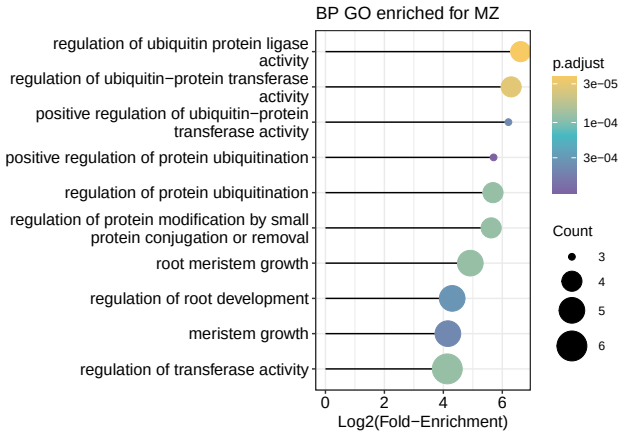


Figure 14: Enrichment of 100 variables most representative per sample cluster

```
fviz_pca_biplot(pca_ara,
                geom.ind = "point",
                fill.ind = samplesNames, pointsize = 5,
                pointshape = 21, repel = T,
                select.ind = list(name = c("ara_EZ_rep1", "ara_EZ_rep2", "ara_EZ_rep3")),
                ↪  select.var = list(contrib = 1250),
                col.var = "contrib") +
  lims(x = c(-60,50), y = c(0, 150)) +
 scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Variables most contributing to the variance of EZ",
       x = paste("PC1 ", "(", round(exp_var_ara$exp_var[1], 2), "%)", sep =""),
       y = paste("PC2 ", "(", round(exp_var_ara$exp_var[2], 2), "%)", sep =""),
```

```
      color = "Contribution", fill = "Zone") +
  theme(legend.position="bottom", legend.text = element_text( size=8)) +
  ↪  coord_fixed(ratio = 6/12)
```

```
## Warning: Removed 1240 rows containing missing values (geom_text_repel).
```

```
## Warning: Removed 1240 rows containing missing values (geom_segment).
```

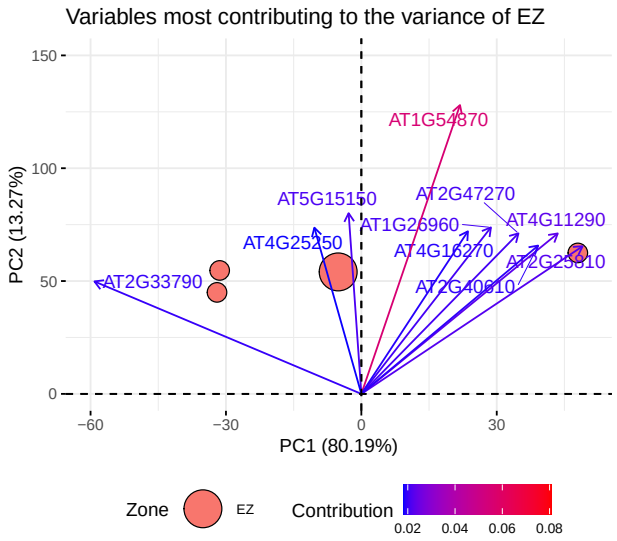


Figure 15: Variables representation per sample cluster

```
pca_go(ara_pca_contrib[ara_pca_contrib$Dim.1 < 70 & ara_pca_contrib$Dim.1 > -80 &
↪  ara_pca_contrib$Dim.2 > 0, ], "EZ", 100)
```

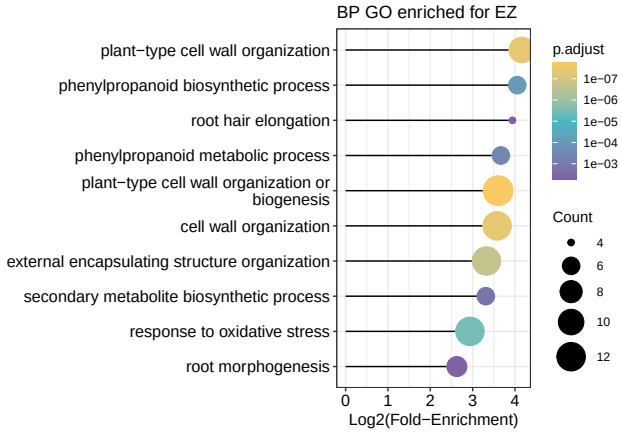


Figure 16: Enrichment of 100 variables most representative per sample cluster

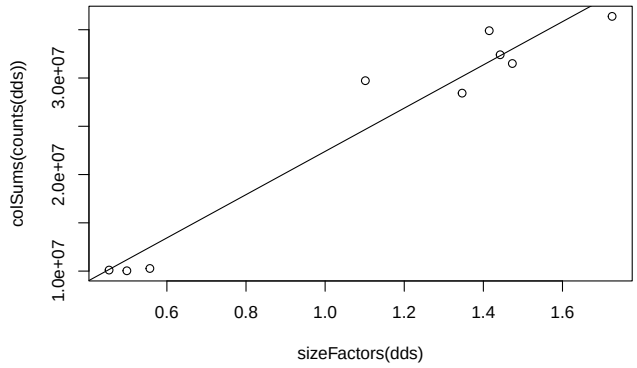
2.4 Diferential Expression Analysis

2.4.1 DESeq Matrix

```
dds <- DESeqDataSetFromMatrix(countData=count_matrix, colData = metadata, design=~dex)
```

2.4.2 Normalization factors

```
dds <- estimateSizeFactors(dds)
plot(sizeFactors(dds), colSums(counts(dds)))
abline(lm(colSums(counts(dds)) ~ sizeFactors(dds) + 0))
normlzd_dds <- counts(dds, normalized=T)
```



2.4.3 MZ vs EZ

```
dds <- DESeq(dds)

contrast <- c("dex", "MZ", "EZ")
res_melong <- results(dds, name = "MZ vs EZ",
                contrast = contrast,
                alpha=0.05, lfcThreshold = log2(1.5))
summary(res_melong)
```

```
##
## out of 20677 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0.58 (up)    : 2170, 10%
## LFC < -0.58 (down) : 3027, 15%
## outliers [1]       : 281, 1.4%
## low counts [2]     : 401, 1.9%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
res_melong_ara <- as.data.frame(res_melong)
```

2.4.4 MZ vs DZ

```
contrast <- c("dex", "MZ", "DZ")
res_mdif <- results(dds, name = "MZ vs DZ",
                contrast = contrast,
                alpha=0.05, lfcThreshold = log2(1.5))
summary(res_mdif)
```

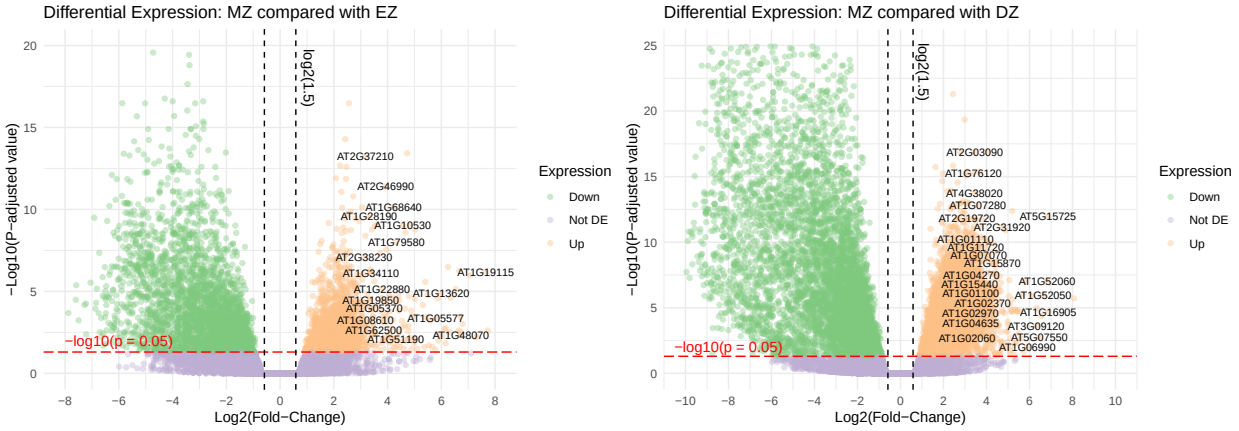
```
##
## out of 20677 with nonzero total read count
## adjusted p-value < 0.05
```

```
## LFC > 0.58 (up)     : 4943, 24%
## LFC < -0.58 (down) : 5654, 27%
## outliers [1]       : 281, 1.4%
## low counts [2]     : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
res_mdif_ara <- as.data.frame(res_mdif)
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```

```
## Warning: Removed 190 rows containing missing values (geom_point).
```



2.4.5 DEGs GO enrichment

```
gseDEGs(res_melong_ara)
```

2.4.5.1 Up and down-regulated in MZ vs. EZ

```
## preparing geneSet collections...
```

```
## GSEA analysis...
```

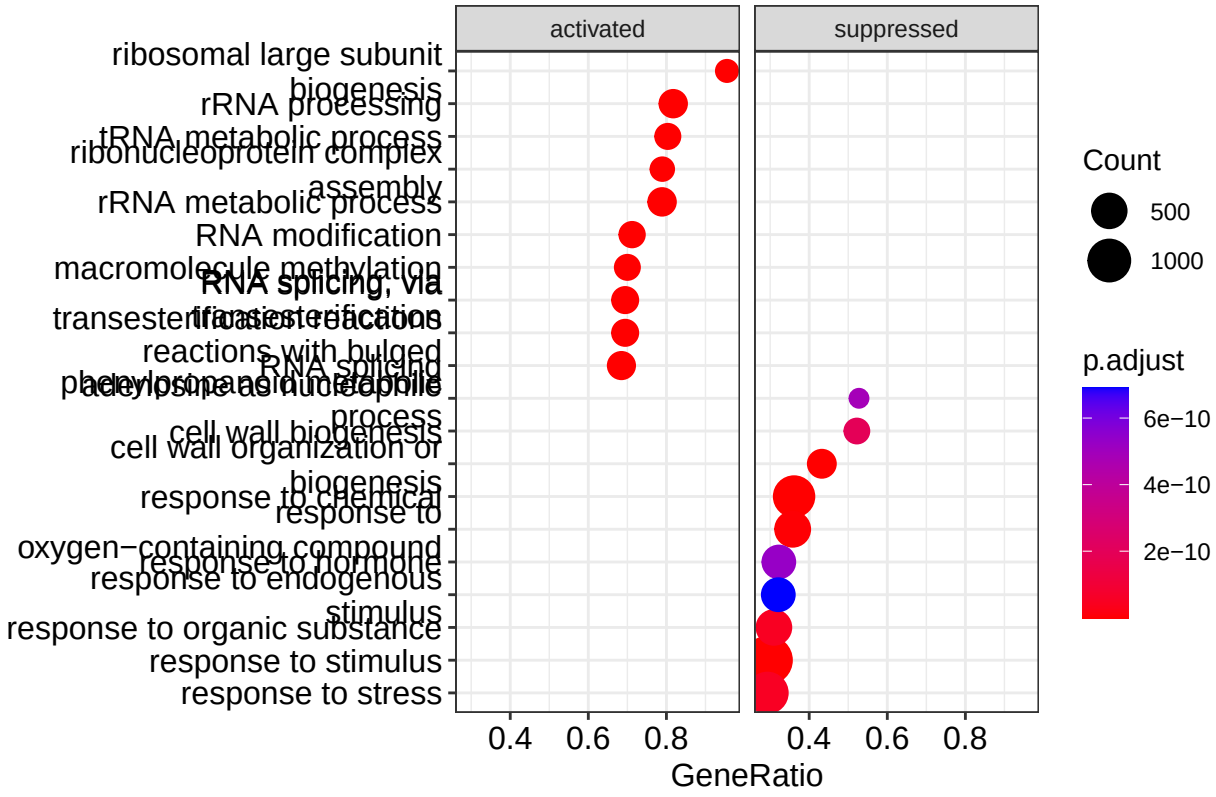
```
## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.
```

```
## Warning in fgseaMultilevel(...): There were 78 pathways for which P-values were
## not calculated properly due to unbalanced (positive and negative) gene-level
## statistic values. For such pathways pval, padj, NES, log2err are set to NA. You
## can try to increase the value of the argument nPermSimple (for example set it
## nPermSimple = 10000)
```

```
## Warning in fgseaMultilevel(...): For some of the pathways the P-values were
## likely overestimated. For such pathways log2err is set to NA.
```

```
## leading edge analysis...
```

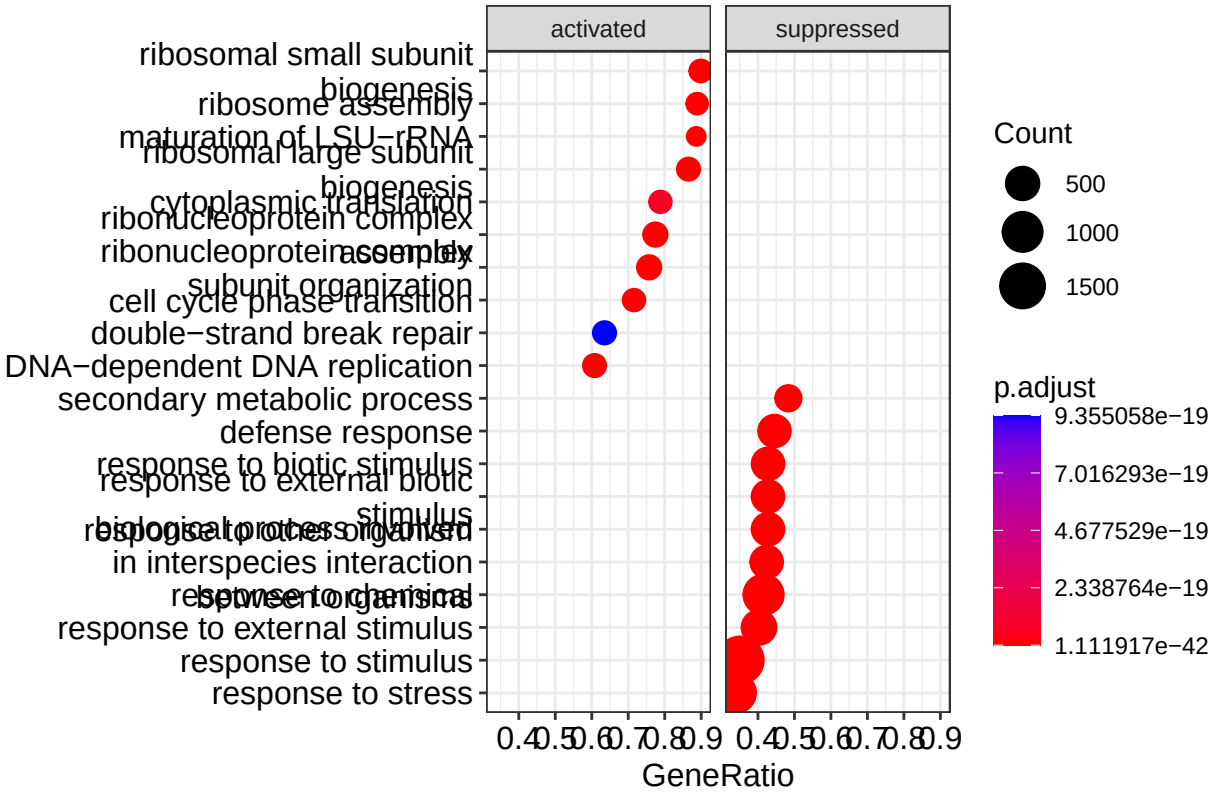
```
## done...
```




```
gseDEGs(res_mdif_ara)
```

2.4.5.2 Up and down-regulated in MZ vs. DZ

```
## preparing geneSet collections...

## GSEA analysis...

## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : There are ties in
## The order of those tied genes will be arbitrary, which may produce unexpected results.

## Warning in fgseaMultilevel(...): There were 139 pathways for which P-values were
## not calculated properly due to unbalanced (positive and negative) gene-level
## statistic values. For such pathways pval, padj, NES, log2err are set to NA. You
## can try to increase the value of the argument nPermSimple (for example set it
## nPermSimple = 10000)

## Warning in fgseaMultilevel(...): For some of the pathways the P-values were
## likely overestimated. For such pathways log2err is set to NA.

## leading edge analysis...

## done...
```

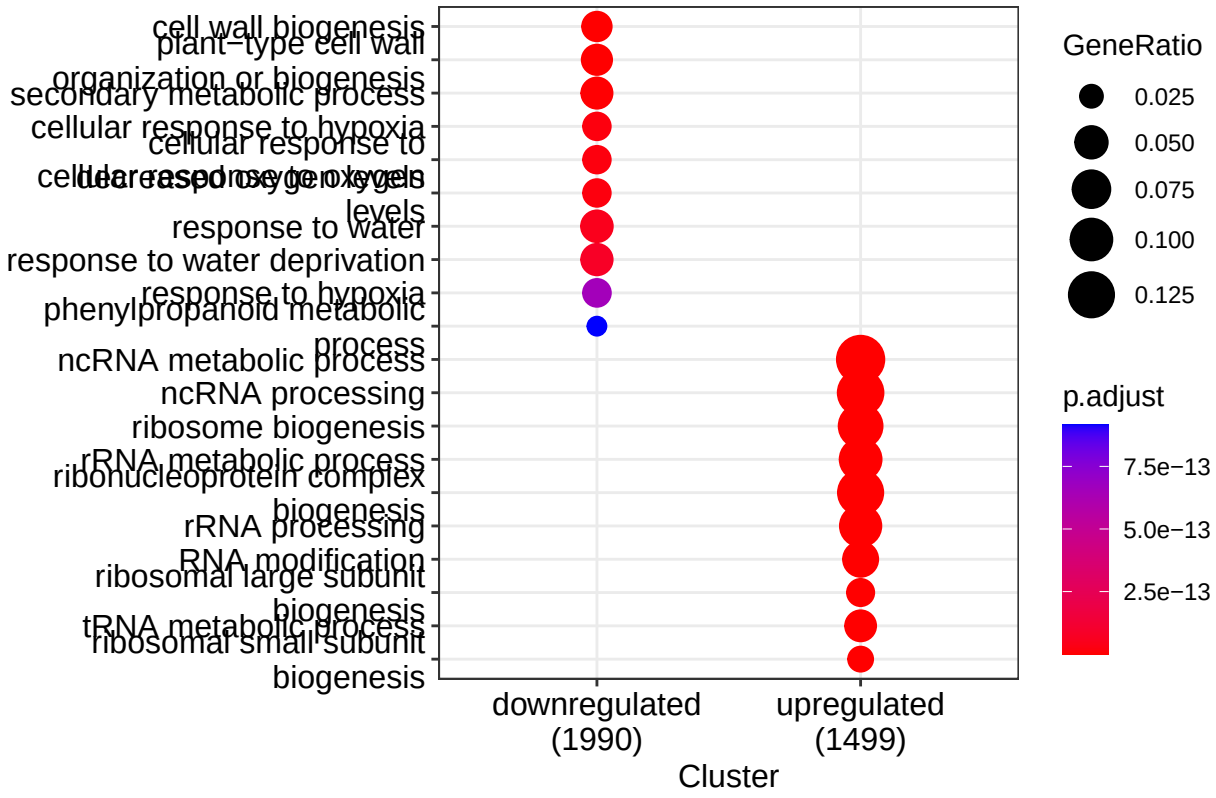


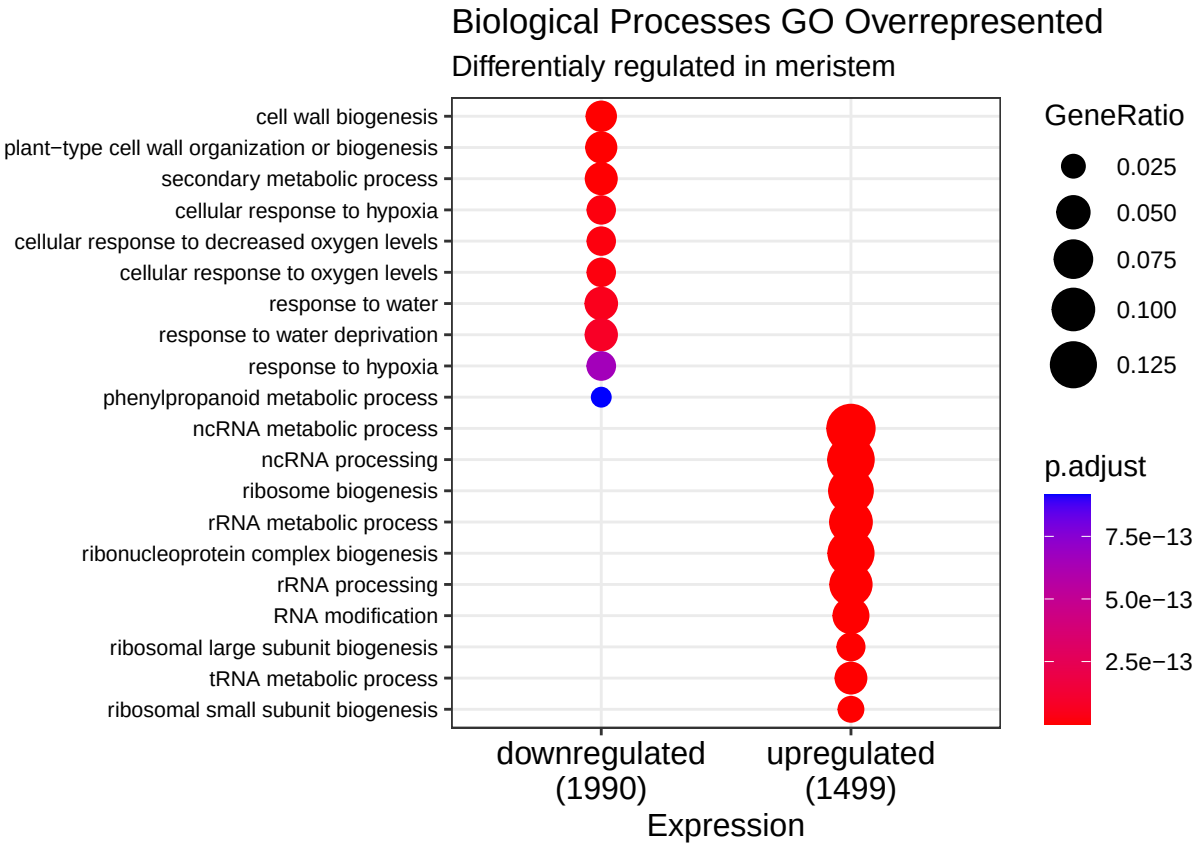

```
dea <- data.frame(tair = rownames(count_matrix),
                  FC.MZEZ = res_melong_ara$log2FoldChange,
                  P.MZEZ = res_melong_ara$padj,
                  FC.MZDZ = res_mdif_ara$log2FoldChange,
                  P.MZDZ = res_mdif_ara$padj)

dea$MZEZ <- NA
dea$MZEZ[dea$FC.MZEZ > log2(1.5) & dea$P.MZEZ < 0.05] <- "upregulated"
dea$MZEZ[dea$FC.MZEZ < log2(1.5) & dea$P.MZEZ < 0.05] <- "downregulated"
dea$MZDZ <- NA
dea$MZDZ[dea$FC.MZDZ > log2(1.5) & dea$P.MZDZ < 0.05] <- "upregulated"
dea$MZDZ[dea$FC.MZDZ < log2(1.5) & dea$P.MZDZ < 0.05] <- "downregulated"

dea$expression <- NA
dea$expression[dea$MZEZ == "upregulated" & dea$MZDZ == "upregulated"] <- "upregulated"
dea$expression[dea$MZEZ == "downregulated" & dea$MZDZ == "downregulated"] <-
↪  "downregulated"

dea <- dea[,c(1,8)]
dea <- na.omit(dea)

res_go_dea <- compareCluster(tair~expression, data=dea, fun = "enrichGO", OrgDb =
↪  org.At.tair.db, keyType = "TAIR", ont = "BP", pAdjustMethod = "BH", pvalueCutoff =
↪  0.05, qvalueCutoff = 0.01)

dotplot(res_go_dea, showCategory=10) +
  theme(axis.text.y = element_text(size = 8)) + scale_y_discrete(labels = function(x)
  ↪  str_wrap(x, 75)) +
```

```
  coord_fixed(ratio = 3.5/28) +
  labs(x = "Expression") +
  ggtitle("Biological Processes GO Overrepresented", subtitle = "Differentialy regulated
  ↪  in meristem")
```

2.4.5.3 Meristem DEGs BP enrichment

```
## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```





2.4.6 Meristem upregulated DEGs

```
res_melong_up <- subset(res_melong, log2FoldChange > log2(1.5))
res_melong_up <- subset(res_melong_up, padj<.05)
res_melong_up <- res_melong_up[order(res_melong_up$padj),]
res_mdif_up <- subset(res_mdif, log2FoldChange > log2(1.5))
res_mdif_up <- subset(res_mdif_up, padj<.05)
res_mdif_up <- res_mdif_up[order(res_mdif_up$padj),]

genes_melong_up <- rownames(res_melong_up)
genes_mdif_up <- rownames(res_mdif_up)

gene_meristem <- intersect(genes_melong_up, genes_mdif_up)
length(gene_meristem)
```

2.4.6.1 Intersect MZ vs DZ and MZ vs EZ

```
## [1] 1813
```

Table 4: Description of genes

transcript_id	defline
AT1G01010.1	NAC domain containing protein 1
AT1G01020.1	Arv1-like protein
AT1G01020.2	Arv1-like protein
AT1G01030.1	AP2/B3-like transcriptional factor family protein
AT1G01040.1	dicer-like 1
AT1G01040.2	dicer-like 1

```
l2fc_melong <- sapply(gene_meristem, function(gen){
  res_melong_up$log2FoldChange[rownames(res_melong_up) ==  gen]})
l2fc_mdif <- sapply(gene_meristem, function(gen){
  res_mdif_up$log2FoldChange[rownames(res_mdif_up) ==  gen]})
padj_melong <- sapply(gene_meristem, function(gen){
  res_melong_up$padj[rownames(res_melong_up) ==  gen]})
padj_mdif <- sapply(gene_meristem, function(gen){
  res_mdif_up$padj[rownames(res_mdif_up) ==  gen]})
defline <- sapply(gene_meristem, function(gen){
  if(gen %in% gene_defline$gene_id){
    defs <- gene_defline$defline[gene_defline$gene_id == gen]
    return(paste(defs[!duplicated(defs)], collapse = ";"))
  }
  else NA
})


meristem <- data.frame(
  "gene_id" = gene_meristem,
  "log2FoldChange_MeristemElong" = l2fc_melong,
  "log2FoldChange_MeristemDif" = l2fc_mdif,
  "padj_MeristemElong" = padj_melong,
  "padj_MeristemDif" = padj_mdif,
  "log2FoldChange_mean" = rowMeans(data.frame(l2fc_mdif, l2fc_melong)),
  "padj_mean" = rowMeans(data.frame(padj_mdif, padj_melong)),
  "def" = defline
  )

meristem <- meristem[order(meristem$log2FoldChange_mean, decreasing = T), ]
write.csv(meristem, file = 'ara/gen_meristem.csv', row.names = F)
```

2.4.7 Meristem downregulated genes

```
res_melong_down <- subset(res_melong, log2FoldChange <= log2(1.5))
res_melong_down <- subset(res_melong_down, padj<=.05)
res_melong_down <- res_melong_down[order(res_melong_down$padj),]
res_mdif_down <- subset(res_mdif, log2FoldChange <= log2(1.5))
res_mdif_down <- subset(res_mdif_down, padj<=.05)
res_mdif_down <- res_mdif_down[order(res_mdif_down$padj),]

genes_melong_down <- rownames(res_melong_down)
genes_mdif_down <- rownames(res_mdif_down)

gene_meristem_down <- intersect(genes_melong_down, genes_mdif_down)

length(gene_meristem_down)
```

2.4.7.1 Intersect MZ vs DZ and MZ vs EZ

```
## [1] 2496
```

```
l2fc_melong <- sapply(gene_meristem_down, function(gen){
  res_melong_down$log2FoldChange[rownames(res_melong_down) ==  gen]})
```

```
l2fc_mdif <- sapply(gene_meristem_down, function(gen){
  res_mdif_down$log2FoldChange[rownames(res_mdif_down) ==  gen]})
padj_melong <- sapply(gene_meristem_down, function(gen){
  res_melong_down$padj[rownames(res_melong_down) ==  gen]})
padj_mdif <- sapply(gene_meristem_down, function(gen){
  res_mdif_down$padj[rownames(res_mdif_down) ==  gen]})
defline <- sapply(gene_meristem_down, function(gen){
  if(gen %in% gene_defline$gene_id){
    defs <- gene_defline$defline[gene_defline$gene_id == gen]
    return(paste(defs[!duplicated(defs)], collapse = ";"))
  }
  else NA
})


meristem_down <- data.frame(
  "gene_id" = gene_meristem_down,
  "log2FoldChange_MeristemElong" = l2fc_melong,
  "log2FoldChange_MeristemDif" = l2fc_mdif,
  "padj_MeristemElong" = padj_melong,
  "padj_MeristemDif" = padj_mdif,
  "log2FoldChange_mean" = rowMeans(data.frame(l2fc_mdif, l2fc_melong)),
  "padj_mean" = rowMeans(data.frame(padj_mdif, padj_melong)),
  "def" = defline
  )

meristem_down <- meristem_down[order(meristem_down$log2FoldChange_mean, decreasing = T),
↪  ]
write.csv(meristem_down, file = 'ara/gen_meristem_down.csv', row.names = F)
```

Transcript with more log2FoldChange for

```
plotCounts(dds, gene=meristem$gene_id[1], intgroup = c('dex'))
plotCounts(dds, gene=meristem_down$gene_id[1], intgroup = c('dex'))
```

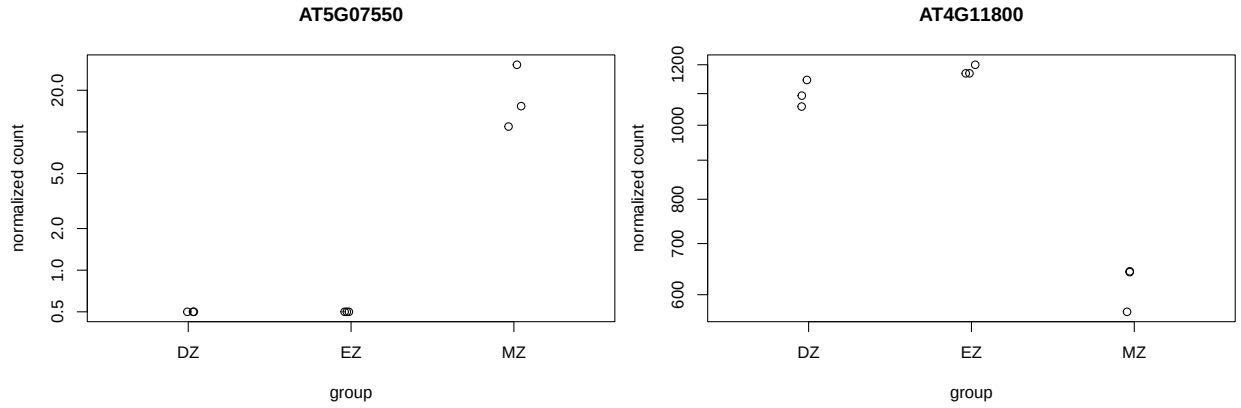


Figure 17: Meristem DEGs

```
grid.newpage()
```

```
draw.quad.venn(area1 = length(genes_melong_up),
               area2 = length(genes_mdif_up),
               area3 = length(genes_melong_down),
               area4 = length(genes_mdif_down),
               n12 = length(gene_meristem),
               n13 = length(intersect(genes_melong_up, genes_melong_down)),
               n14 = length(intersect(genes_melong_up, genes_mdif_down)),
               n23 = length(intersect(genes_mdif_up, genes_melong_down)),
               n24 = length(intersect(genes_mdif_up, genes_mdif_down)),
               n34 = length(gene_meristem_down),
               n123 = length(intersect(gene_meristem, genes_melong_down)),
               n124 = length(intersect(gene_meristem, genes_mdif_down)),
               n134 = length(intersect(gene_meristem_down, genes_melong_up)),
               n234 = length(intersect(gene_meristem_down, genes_mdif_up)),
               n1234 = length(intersect(gene_meristem, gene_meristem_down)),
               category = c("MZ-EZ(up)", "MZ-DZ(up)", "MZ-EZ(down)", "MZ-DZ(down)"),
               lty = "blank", fill = c("skyblue", "pink1", "mediumorchid", "orange"),)
```

```
## (polygon[GRID.polygon.1368], polygon[GRID.polygon.1369], polygon[GRID.polygon.1370], polygon[GRID.pol
```

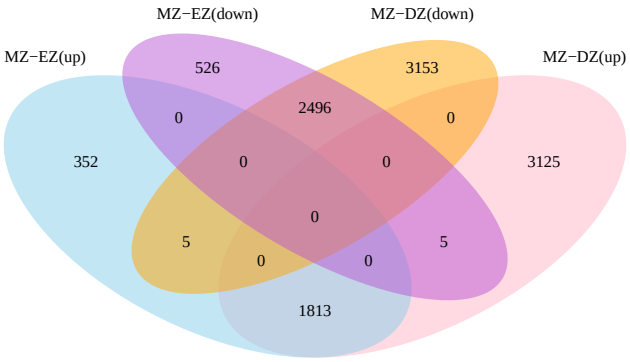


Figure 18: Intersection between upregulated genes in meristem in DE analysis

2.5 Meristem DEGs regulated by PLETHORA

2.5.1 Santuari PLT regulated genes compendium

Table 5: Santuari's Supplemental Dataset

AGI	PLT1	PLT2	PLT3	PLT4	PLT5	PLT7
AT1G01010	0	0	0	0	0	0
AT1G01030	0	0	0	0	0	0
AT1G01040	0	0	0	0	0	0
AT1G01050	0	0	0	0	0	0
AT1G01060	0	0	0	0	0	0
AT1G01070	-1	-1	0	-1	0	-1

Only genes upregulated or downregulated by some of the PLETHORA are filtered:

```
regulated_gen <- apply(santuari[,21:26], MARGIN = 1, function(gen) 1%in%gen | -1%in%gen)
plt_gens <- santuari[regulated_gen,]
head(plt_gens[,c(1,21:26)])  %>%
  kable(caption = "Santuari's Supplemental Dataset filtered", longtable = T, booktabs =
  ↪  TRUE, align = 'c', digits = 2) %>%
```

```
    kable_styling(font_size = 7, full_width = F)
```

Table 6: Santuari's Supplemental Dataset filtered

AGI	PLT1	PLT2	PLT3	PLT4	PLT5	PLT7
AT1G01070	-1	-1	0	-1	0	-1
AT1G01110	1	1	0	0	0	0
AT1G01120	-1	-1	-1	-1	0	-1
AT1G01140	-1	0	0	0	0	-1
AT1G01320	-1	0	0	0	0	0
AT1G01390	-1	-1	-1	-1	0	-1

2.5.2 Meristem upregulated DEGs regulated by PLETHORA

```
plt_gen_mz <- meristem[meristem$gene_id%in%plt_gens$AGI,]
plt_gen_mz_comp <- sapply(meristem$gene_id, function(locus) which(locus == plt_gens$AGI))
plt_gen_mz_comp <- unlist(Filter(length,plt_gen_mz_comp))
plt_gen_mz_comp <-plt_gens[plt_gen_mz_comp,]
plt_gen_mz <- cbind(plt_gen_mz, plt_gen_mz_comp)
```

Table 7: Meristem upregulated genes regulated by PLETHORA

gene_id	log2FoldChange_MeristemElong	def
AT5G57420	6.54	indole-3-acetic acid inducible 33
AT5G15725	5.02	NA
AT1G32900	6.67	UDP-Glycosyltransferase superfamily protein
AT5G58784	3.27	Undecaprenyl pyrophosphate synthetase family protein
AT2G04025	6.35	PTHR36313:SF1 - ROOT MERISTEM GROWTH FACTOR 1-RELATED (1 of 3)
AT1G26680	5.48	transcriptional factor B3 family protein

```
length(rownames(plt_gen_mz))
```

```
## [1] 661
```

2.5.3 Meristem downregulated DEGs regulated by PLETHORA

```
plt_gen_mz_down <- meristem_down[meristem_down$gene_id%in%plt_gens$AGI,]
plt_gen_mz_comp_down <- sapply(meristem_down$gene_id, function(locus) which(locus ==
↪  plt_gens$AGI))
plt_gen_mz_comp_down <- unlist(Filter(length,plt_gen_mz_comp_down))
plt_gen_mz_comp_down <-plt_gens[plt_gen_mz_comp_down,]
plt_gen_mz_down <- cbind(plt_gen_mz_down, plt_gen_mz_comp_down)
```

Table 8: Meristem upregulated genes regulated by PLETHORA

gene_id	log2FoldChange_MeristemElong	def
AT2G18160	-1.23	basic leucine-zipper 2
AT2G18162	-1.23	conserved peptide upstream open reading frame 1
AT3G01930	-1.20	Major facilitator superfamily protein
AT2G44060	-1.18	Late embryogenesis abundant protein, group 2
AT1G64530	-1.19	Plant regulator RWP-RK family protein
AT2G47490	-1.11	NAD+ transporter 1

```
length(rownames(plt_gen_mz_down))
```

```
## [1] 909
```

2.6 PLT binding motifs in DEGs regulated by PLETHORA

2.6.1 Retrieve upstream (promoter region)

First, we need to retrieve upstream sequences of the DEGs upregulated and downregulated in meristem and regulated at the same time by some of the PLETHORA (which have a conserved binding motif).

For this, we use RSAT Sequence Tools from the regulatory sequence analysis tools (RSAT) web server [6] starting from a list of genes for retrieve the default upstream region (**-1000 to -1**) from the start codon which should include the promotor sequences. Additionally, the option for prevent overlap with neighbour genes (**noorf**) was activated.

With the weight matrices downloaded from the Plant Promoter Analysis Navigator for the PLT bindind sites in Arabidpopsis and the fasta file of the meristem DEGs we implemented a scan of the sequences in search of binding motifs. For this, we use RSAT matrix-scan service.

The scan settings includes an order 2 markov model and a genomic background corresponding to the TAIR10 assembly.

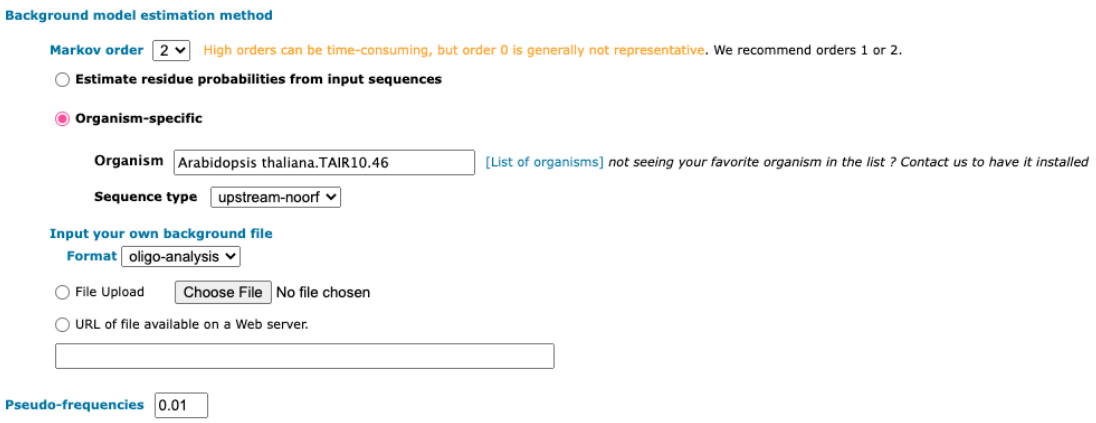


Figure 19: Use of RSAT

2.6.2 Binding motifs in upregulated DEGs regulated by PLETHORA

```
write.csv(plt_gen_mz, file = "ara/plt_gen.csv", row.names = F)
```

```
gm_scan_snt <-
↪  read.table("ara/tfbs/results/matrix-scan_jasparmotifs_plt1234567_p001_3000bp.gff3.txt",
↪  sep = "\t", col.names = c('gen_id',  'source',   'ft_type',  'start',    'end',
↪  'score',    'strand',   'frame',    'attribute'))
gm_scan_snt <- gm_scan_snt[gm_scan_snt$ft_type == 'site',]
```

```
# Separate attribute fields in seqname and seq columns
attrib <- str_split(gm_scan_snt$attribute, pattern = ";N")
plt_bind <- sapply(attrib, function(x) str_split(x[1], pattern = "=")[[1]][2])
seq_plt_motif <- sapply(attrib, function(x) str_split(x[2], pattern = "=")[[1]][2])
name <- str_split(gm_scan_snt$gen_id, pattern = "\\|", n = 2)
```

```
gen_id <- sapply(name, function(x) x[1])
gen_name <- sapply(name, function(x) x[2])
gm_scan_snt <- cbind(gen_id, gen_name, gm_scan_snt[c(-1, -9)], plt_bind, seq_plt_motif)
```

```
length(unique(gm_scan_snt$gen_id))
```

```
## [1] 432
```

```
grid.newpage()
draw.pairwise.venn(area1 = length(plt_gens$AGI),
                   area2 = length(meristem$gene_id),
                   cross.area = length(plt_gen_mz$gene_id),
                   category = c("Compendium of\nPLT regulated genes",
                                "Meristem\nupregulated DEGs"),
                   fill = c("skyblue", "pink1"),
                   lty = "blank", cat.cex = 0.9, scaled = F, cat.pos = 180)
```

```
## (polygon[GRID.polygon.1395], polygon[GRID.polygon.1396], polygon[GRID.polygon.1397], polygon[GRID.pol
```

```
grid.newpage()
draw.triple.venn(area1 = length(plt_gens$AGI),
                   area2 = length(meristem$gene_id),
                   area3= length(unique(gm_scan_snt$gen_id)),
                   n12 = length(plt_gen_mz$gene_id),
                   n13 = length(intersect(plt_gens$AGI, unique(gm_scan_snt$gen_id))),
                   n23 = length(intersect(meristem$gene_id, unique(gm_scan_snt$gen_id))),
                   n123 = length(
                     intersect(
                       intersect(plt_gens$AGI,unique(gm_scan_snt$gen_id)),
                       meristem$gene_id)),
                   category = c("Compendium of\nPLT regulated genes",
                                "Meristem\nupregulated DEGs",
                                "Meristem upregulated DEGs\nwith PLT binding motif"),
                   scaled = T,
                   fill = c("skyblue", "pink1", "mediumorchid"),
                   lty = "blank", cat.cex = 0.9, cat.pos = 180)
```

```
## (polygon[GRID.polygon.1404], polygon[GRID.polygon.1405], polygon[GRID.polygon.1406], polygon[GRID.pol
```

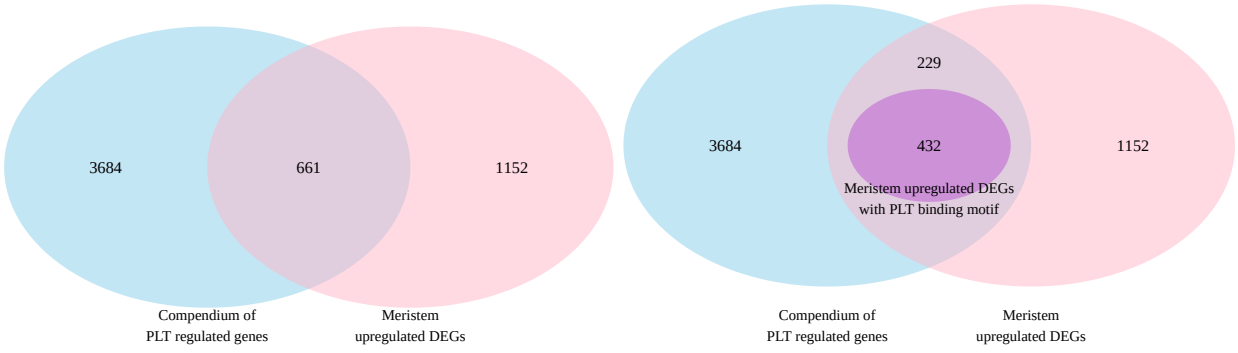


Figure 20: Finding PLT binding sites in meristem upregulated DEGs

```
## take the sites with the best score per query
c <- lapply(unique(gm_scan_snt$gen_id), function(query){
  temp <- gm_scan_snt[gm_scan_snt$gen_id == query,]
  return(temp[which.max(temp$score),])
})
filt_gen_ms_sant <- c[[1]]
for(row in c[2:length(c)]){
  filt_gen_ms_sant <- rbind(filt_gen_ms_sant, row)
}
```

```
ggplot(filt_gen_ms_sant, aes(x=start)) +
  geom_density()
```

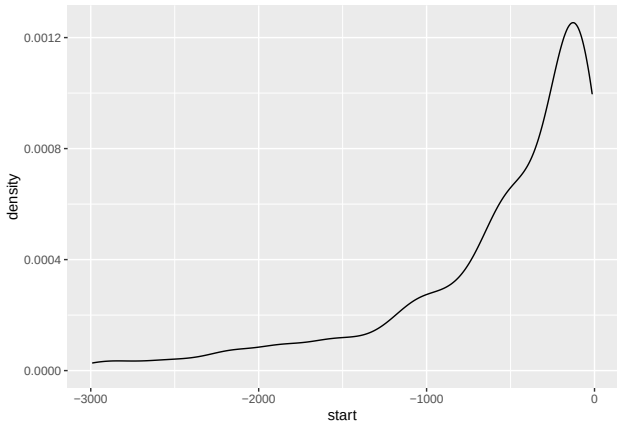


Figure 21: PLT binding sites distribution along the promoter sequences

2.6.3 Binding motifs in downregulated DEGs regulated by PLETHORA

```
write.csv(plt_gen_mz_down, file = "ara/plt_gen_down.csv", row.names = F)
```

```
gm_scan_snt <-
↪  read.table("ara/tfbs/results/matrix-scan_jasparmotifs_plt1234567_p001_down.gff3.txt",
↪  sep = "\t", col.names = c('gen_id',    'source',   'ft_type',  'start',    'end',
↪  'score',    'strand',   'frame',    'attribute'))
gm_scan_snt <- gm_scan_snt[gm_scan_snt$ft_type == 'site',]
```

```
# Separate attribute fields in seqname and seq columns
attrib <- str_split(gm_scan_snt$attribute, pattern = ";N")
plt_bind <- sapply(attrib, function(x) str_split(x[1], pattern = "=")[[1]][2])
seq_plt_motif <- sapply(attrib, function(x) str_split(x[2], pattern = "=")[[1]][2])
name <- str_split(gm_scan_snt$gen_id, pattern = "\\|", n = 2)
gen_id <- sapply(name, function(x) x[1])
gen_name <- sapply(name, function(x) x[2])
gm_scan_snt_down <- cbind(gen_id, gen_name, gm_scan_snt[c(-1, -9)], plt_bind,
↪  seq_plt_motif)
```

```
length(unique(gm_scan_snt_down$gen_id))
```

```
## [1] 797
```

```
c <- lapply(unique(gm_scan_snt_down$gen_id), function(query){
  temp <- gm_scan_snt[gm_scan_snt_down$gen_id == query,]
  return(temp[which.max(temp$score),])
})
filt_gen_ms_sant_down <- c[[1]]
for(row in c[2:length(c)]){
  filt_gen_ms_sant_down <- rbind(filt_gen_ms_sant_down, row)
}
```

```
ggplot(filt_gen_ms_sant_down, aes(x=start)) +
  geom_density()
```

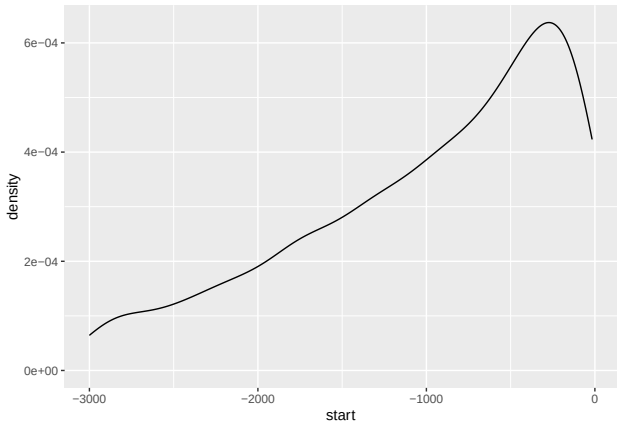


Figure 22: PLT binding sites distribution along the promoter sequences

```
grid.newpage()
draw.pairwise.venn(area1 = length(plt_gens$AGI),
                   area2 = length(meristem_down$gene_id),
                   cross.area = length(plt_gen_mz_down$gene_id),
                   category = c("Compendium of\nPLT regulated genes",
                                "Meristem\ndownregulated DEGs"),
                   fill = c("skyblue", "pink1"),
                   lty = "blank", cat.cex = 0.9, scaled = F, cat.pos = 180)
```

```
## (polygon[GRID.polygon.1499], polygon[GRID.polygon.1500], polygon[GRID.polygon.1501], polygon[GRID.pol
```

```
grid.newpage()
draw.triple.venn(area1 = length(plt_gens$AGI),
                   area2 = length(meristem_down$gene_id),
                   area3= length(unique(gm_scan_snt_down$gen_id)),
                   n12 = length(plt_gen_mz_down$gene_id),
                   n13 = length(intersect(plt_gens$AGI, unique(gm_scan_snt_down$gen_id))),
                   n23 = length(intersect(meristem_down$gene_id,
                   ↪  unique(gm_scan_snt_down$gen_id))),
                   n123 = length(
                     intersect(
                       intersect(plt_gens$AGI,unique(gm_scan_snt_down$gen_id)),
                       meristem_down$gene_id)),
                   category = c("Compendium of\nPLT regulated genes",
                                "Meristem\ndownregulated DEGs",
```

```
                          "Meristem downregulated DEGs\nwith PLT binding motif"),
                scaled = T,
                fill = c("skyblue", "pink1", "mediumorchid"),
                lty = "blank", cat.cex = 0.9, cat.pos = 180)
```

```
## (polygon[GRID.polygon.1508], polygon[GRID.polygon.1509], polygon[GRID.polygon.1510], polygon[GRID.pol
```

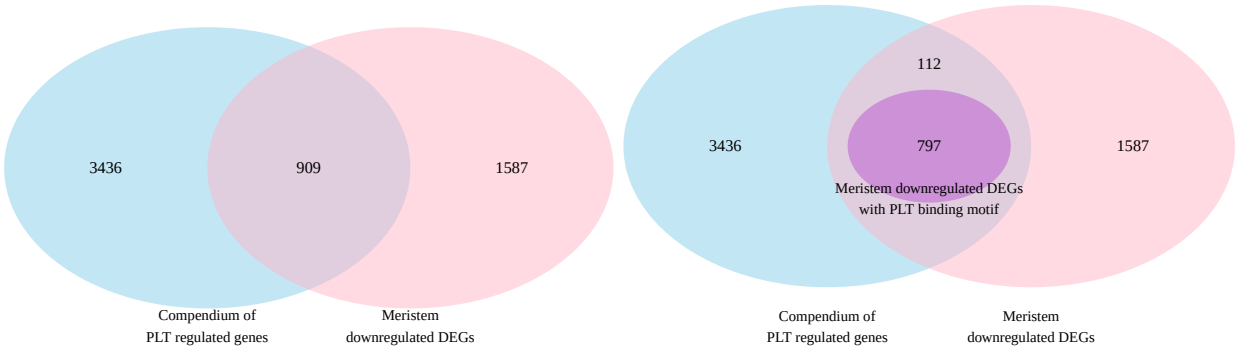


Figure 23: Finding PLT binding sites in meristem upregulated DEGs

2.7 Transcription Factors in Meristem

```
tfs <- read.table(file = 'ara/genome/Ath_TF_list.txt', sep = '\t', header = 1)
```

```
tfs_meristem <- meristem[meristem$gene_id %in% tfs$Gene_ID,]
fam <- sapply(meristem$gene_id[meristem$gene_id %in% tfs$Gene_ID], function(x)
↪  tfs$Family[x == tfs$Gene_ID][1])
tfs_meristem <- cbind(tfs_meristem, fam)
```

Table 9: Meristem TFs

D	1	2	3	4	5	6
AT4G17710	5.85	5.59	0.00	0.00	5.72	0.00
AT1G26680	5.48	5.14	0.00	0.00	5.31	0.00
AT1G33280	4.88	5.19	0.00	0.00	5.03	0.00
AT1G65620	5.11	4.60	0.03	0.03	4.85	0.03
AT3G20840	5.43	3.89	0.00	0.01	4.66	0.01
AT4G01460	3.53	5.68	0.00	0.00	4.61	0.00

Data type: [1] log2FoldChange_MeristemElong
[2] log2FoldChange_MeristemDif [3] padj_MeristemElong
[4] padj_MeristemDif [5] log2FoldChange_mean
[6] padj_mean

Number of TFs upregulated in meristem:

```
length(rownames(tfs_meristem))
```

```
## [1] 68
```

```
write.csv(tfs_meristem, file = 'ara/tfs_ara_meristem.csv', row.names = F)
```

2.7.1 Families of TFs expressed in meristem

```
tfs_fams <- unique(tfs_meristem$fam)
tfs_fams
```

```
##  [1] "HD-ZIP"      "B3"          "NAC"         "LBD"         "AP2"
##  [6] "bHLH"        "SAP"         "TCP"         "bZIP"        "GATA"
## [11] "GRF"         "C2H2"        "ARF"         "MYB"         "C3H"
## [16] "HSF"         "MIKC_MADS"   "MYB_related" "Whirly"      "GeBP"
## [21] "HB-other"    "E2F/DP"      "Trihelix"    "ERF"         "G2-like"
## [26] "GRAS"        "CPP"         "WRKY"        "NF-YC"       "STAT"
```

2.8 TFs upregulated in meristem and regulated by PLT

```
filt_gen_ms_sant <- read_excel("ara/results.xlsx")
tfs_matrix_scan_sant <- filt_gen_ms_sant[filt_gen_ms_sant$gen_id %in% tfs$Gene_ID,]
fam <- sapply(tfs_matrix_scan_sant$gen_id, function(x) tfs$Family[x == tfs$Gene_ID][1])
tfs_matrix_scan_sant <- cbind(tfs_matrix_scan_sant, fam)
```

Table 10: Matrix-scan filtered result

gen_id	gen_name	TF	plt_bind
AT1G26680	AT1G26680	B3	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT4G01460	BHLH57	bHLH	plt1
AT3G19500	BHLH113	bHLH	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT1G68640	PAN	bZIP	plt1
AT4G36930	SPT	bHLH	plt1
AT1G19850	ARF5	ARF	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT2G28450	AT2G28450	C3H	plt1
AT1G77570	AT1G77570	HSF	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT1G17460	TRFL3	MYB_related	plt1
AT5G10510	AIL6	AP2	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT2G36010	E2F3	E2F/DP	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT5G63420	emb2746	Trihelix	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT5G53290	CRF3	ERF	plt1
AT4G23750	CRF2	ERF	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT5G17490	RGL3	GRAS	plt1;_plt2;_plt3;_plt4;_plt5;_plt7
AT4G25210	GEBPL	GeBP	plt1
AT1G56170	NFYC2	NF-YC	plt1;_plt2;_plt3;_plt4;_plt5;_plt7

```
unique(tfs_matrix_scan_sant$fam)
```

```
##  [1] "B3"          "bHLH"        "bZIP"        "ARF"         "C3H"
##  [6] "HSF"         "MYB_related" "AP2"         "E2F/DP"      "Trihelix"
## [11] "ERF"         "GRAS"        "GeBP"        "NF-YC"
```

```
length(rownames(tfs_matrix_scan_sant))
```

```
## [1] 17
```

2.9 Nwtwork tables

```
plt_ids <- read.table("ara/tfbs/plt_ids.txt", sep = "\t", col.names = c("plt_name",
↪  "plt_id"))
plt_gens_filt <- plt_gens[plt_gens$AGI %in% filt_gen_ms_sant$gen_id, ]

int_table <- do.call(rbind, lapply(plt_gens_filt$AGI, function(gen){
  temp <- as.list(plt_gens_filt[plt_gens_filt$AGI == gen, c(21:26)])
```

```
  names(temp) <- sapply(names(temp), function(plt){
    plt_name <- tolower(str_split(plt, "_")[[1]][1])
    return(plt_ids$plt_id[plt_ids$plt_name == plt_name])
    })
  plts <- na.omit(sapply(temp, function(plt){
    if(!is.na(plt)){
    if(plt != 0){
      if(plt == 1) return("A")
      else return("R")
    }
    else return(NA)
    }else return(NA)}))
  return(data.frame(
    source = noquote(names(plts)),
    interaction = plts,
    target = noquote(rep(gen, length(plts)))
  ))
}))

write.csv(int_table, file = "ara/network_table.csv", row.names = F, quote = F)

goslim <- read.table("ara/genome/ATH_GO_GOSLIM.txt", sep = "\t", comment.char = "!",
↪  quote = "")
getgo <- read.table("ara/getgo.txt", sep = "\t", comment.char = "!", quote = "", header =
↪  1)

normlzd_dds <- as.data.frame(normlzd_dds)

prop_table <- do.call(rbind, lapply(filt_gen_ms_sant$gen_id, function(gen){
  common <- filt_gen_ms_sant$gen_name[filt_gen_ms_sant$gen_id == gen]
  product_type <- getgo$GO.term[getgo$Locus == gen & getgo$category == 'func']
  if(length(product_type) == 0) product_type <- goslim$V5[goslim$V1 == gen & (goslim$V4
  ↪  == "involved in")][1]
  if(length(product_type) == 0) product_type <- goslim$V5[goslim$V1 == gen & (goslim$V4
  ↪  == "enables")][1]
  if(length(product_type) == 0) product_type <- goslim$V5[goslim$V1 == gen & (goslim$V4
  ↪  == "acts upstream of or within")][1]
  mz <- rowMeans(normlzd_dds[rownames(normlzd_dds) == gen, c(1:3)])
  ez <- rowMeans(normlzd_dds[rownames(normlzd_dds) == gen, c(4:6)])
  dz <- rowMeans(normlzd_dds[rownames(normlzd_dds) == gen, c(7:9)])
  return(cbind(data.frame(
    gen_id = gen,
    alias = common,
    product_type = paste(product_type, collapse = '; '),
    MZ = mz, EZ = mz, DZ = dz
  ), normlzd_dds[rownames(normlzd_dds) == gen,]))
}))

write.csv(prop_table, file = "ara/prop_table.csv", row.names = F)
```