Elucidation and Analyses of the Regulatory Networks of Upland and Lowland Ecotypes of Switchgrass in Response to Drought and Salt Stresses

Working

PLOS One

Chunman Zuo[1], Yuhong Tang[2], Hao Fu[3], Yiming Liu[4], Xunzhong Zhang[4], Bingyu Zhao[5], Ying Xu[1]

[1]College of Computer Science and Technology, Jilin University, Changchun, China, [2]Noble Research Institute, LLC., Ardmore, OK, USA, [3]North Automatic Control Technology Institute, Taiyuan, China, [4]Department of Crop and Soil Environmental Science, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA, [5]Department of Horticulture, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, USA

Sep 25, 2018 dx.doi.org/10.17504/protocols.io.saxeafn

Chunman Zuo
Jilin University

EXTERNAL LINK

https://doi.org/10.1371/journal.pone.0204426

THIS PROTOCOL ACCOMPANIES THE FOLLOWING PUBLICATION

Zuo C, Tang Y, Fu H, Liu Y, Zhang X, Zhao B, Xu Y (2018) Elucidation and analyses of the regulatory networks of upland and lowland ecotypes of switchgrass in response to drought and salt stresses. PLoS ONE 13(9): e0204426. doi: 10.1371/journal.pone.0204426

PROTOCOL STATUS

Working

SAFETY WARNINGS

This command take hours to complete: 'screen' or 'nohup' command to avoid process being killed if the connection to the server is lost.

1 Download Illumina-based sequencing data from NCBI.

DATASET
Illumina Hiseq 2500 RNA-seq data NCBI BioProject Accession: PRJN

DATASET
Illumina Hiseq 2500 RNA-seq data NCBI BioProject Accession: PRJN

SOFTWARE
NCBI SRA Toolkit 2.8.2-1

COMMAND

```
my $fastq_dump = $project_dir."/sratoolkit/2.8.2-1"."/bin/fastq-dump";
foreach my $code (@SRR_codes)
{
  system("$fastq_dump -I --split-files $code -o $out_dir");
}
```

For each sample of each project, the SRR code and corresponding information are mannually collected. Then, the script 'fastq-dump' was used to download each sample based on SRR code in the current directory, with fastq format. Note: $project_dir and $out_dir indicate the software installation directory and fastq file output directory, respectively.

Unix - Perl script

2 Adapter-trimming, quality-trimming, filtering and contaminant-filtering, were applied to each fastq file

SOFTWARE
BBDuk tools 35.82

COMMAND

```
my $bbduk = $project_dir."/bbmap/35.82"."/bbduk.sh";
my $contam_seq =  $project_dir."/bbmap/35.82"."/resources/phix174_ill.ref.fa.gz";
my $adapter_seq = $project_dir."/bbmap/35.82"."/resources/adapters.fa";

foreach my $code (@SRR_codes)
{
  my $fq_file_1 = $out_dir.$code."_1.fastq";
  my $fq_file_2 = $out_dir.$code."_2.fastq";

  my $fq_file_trim_1 = $out_dir.$code."_1_trim.fastq";
  my $fq_file_trim_2 = $out_dir.$code."_2_trim.fastq";

  my $fq_file_final_1 = $out_dir.$code."_1_final.fastq";
  my $fq_file_final_2 = $out_dir.$code."_2_final.fastq";

  my $fq_file_m_1 = $out_dir.$code."_1_match.fastq";
  my $fq_file_m_2 = $out_dir.$code."_2_match.fastq";

  my $fq_match_status = $out_dir.$code."_status.txt";

  # adapter-trimming, quality-trimming, filtering
  system("$bbduk -Xmx5g in=$fq_file_1 in2=$fq_file_2 out=$fq_file_trim_1 out2=$fq_file_trim_2 ref=$adapter_seq threads=10 qtrim=rl trimq=10 ktrim=r k=23 mink=11 hdist=1 tpe tbo");
  # contaminant-filtering
  system("$bbduk -Xmx5g in=$fq_file_trim_1 in2=$fq_file_trim_2 out=$fq_file_final_1 out2=$fq_file_final_2 outm=$fq_file_m_1 outm2=$fq_file_m_2 ref=$contam_seq threads=10 k=31 hdist=1 stats=$fq_match_status");
}
```

Regarding each SRR sample, bbduk tools were used to simultaneously process two fastq files.

Unix - Perl script

3 Mapping the high-quality reads to the genome

SOFTWARE
Hisat2 2.1.0

COMMAND

```
my $Hisat2_mapp = $project_dir."/hisat2/2.1.0/hisat2";
my $hisat2_build = $project_dir."/hisat2/2.1.0"."/bin/hisat2-build";

# go into genome directory, and generate one directory to save genome index files
system("cd $genome_dir");
system("mkdir index");

# build index files for genome of switchgrass (indicated by variable '$genome_fa')
my $genome_index = $genome_dir."/index/switchgrass";
system("$hisat2_build -f $genome_fa $genome_index");

# for each sample, hisat2 was used to map the raw reads into genome
foreach my $code (@SRR_codes)
{
  my $fq_1 = $out_dir.$code."_1_final.fastq";
  my $fq_2 = $out_dir.$code."_2_final.fastq";

  my $file1 = $out_dir.$code."_mapping.sam";

  #mapping
  system("$Hisat2_mapp -x $genome_index -q -t -k 1 -p 10 -1 $fq_1  -2 $fq_2 -s $file1");
}
```

Firstly, the script 'hisat2-build' was used to build HISAT2 index for the genome fasta file. Then, for each SRR sample, based on the index files, the script

Unix - Perl script

4 Summary gene-level read counts by featureCounts

SOFTWARE
Subread 1.5.0

COMMAND

```
my $subread = $project_dir."subread-1.5.0-Linux-x86_64/bin/featureCounts";

# collect all the mapping files for each project
opendir($AA, $out_dir);
my @list_ = readdir($AA);
closedir($AA);

### the variable $line was used to save all the mapping files directory
my $line = "";
foreach my $file (@list_)
{
  if( $file =~ /_mapping.sam/)
  {
    if($line eq "")
    {
      $line = $out_dir.$file;
    }else{
      $line = $line." ".$out_dir.$file;
    }
  }
}

# summary gene-level read counts
my $out_file = $out_dir."Raw_reads_count.txt";

# ref_file is the SAF formatted gene annotation file including GeneID, Chr, Start, End and Strand
system("$subread -T 10 -p -g gene_id -a $ref_file -F SAF -o $out_file $line");
```

Based on the gene annotation and raw reads mapping files, the software featureCounts was used to summary gene-level read counts.

Unix - Perl script

5 Normalization and differentially expression analysis

SOFTWARE
edgeR 3.20.9

COMMAND

```
# load edgeR package
library("edgeR")

# normalization steps: filter out genes with lower read counts and normalization by TMM methods

# variable 'reads_count' indicates the gene expression matrix, with row for genes and column for samples.
d <- DGEList(counts=reads_count)

#variable 'gene_length' indicates gene length, with the same order for rows of reads_count
d$genes$Length = gene_length

#variable 'sample_group' indicates the class (ecotype+time) for each stress type
d$samples$group = sample_group

cpm_reads = cpm(d,log = F)

# filter out genes with low reads count
keep <- rowSums(cpm_reads >2) >= ceiling(0.9*length(sample_group))

d_new = DGEList(counts= reads_count[keep,])
d_new$genes$Length <- gene_length[keep]
d_new$samples$group = sample_group

d_new1 <- calcNormFactors(d_new)

##CPM normalization
cpm_reads_new = cpm(d_new1,log = F)

### differentially expressed analysis

condition1 = c("treated1","treated2")
condtion2 = c("untreated1","untreated2")

for(i in 1:length(condtion1))
{
   treat_loc1 = NULL
   untreat_loc1 = NULL

  for(j in 1:dim(reads_count)[2])
  {
     if(regexpr(condtion1[i],colnames(reads_count)[j])[1]>-1)
     {
       treat_loc1 = c(treat_loc1,j)
     }

    if(regexpr(condtion2[i],colnames(reads_count)[j])[1]>-1)
     {
       untreat_loc1 = c(untreat_loc1,j)
     }
  }

  temp_data = reads_count[keep,c(treat_loc1, untreat_loc1)]
  colnames(temp_data) =     c(paste((rep("A",length(treat_loc1))),seq(1,length(treat_loc1)),sep=""),
                            paste((rep("B",length(untreat_loc1))),seq(1,length(untreat_loc1)),sep=""))
  row.names(temp_data) = row.names(reads_count[keep,])
  d_s = DGEList(counts=temp_data,group=c(rep("A",length(treat_loc1)),rep("B",length(untreat_loc1))))
  d_s <- calcNormFactors(d_s)

  design = model.matrix(~0+group, data=d_s$samples)
  colnames(design) = levels(d_s$samples$group)

  y <- estimateDisp(d_s, design,robust = T)
  fit <- glmQLFit(y, design, robust=TRUE)

  con = makeContrasts(contrasts= "A-B", levels=c("A","B"))
  annov = glmQLFTest(fit,contrast =con )
  annova_com = topTags(annov,n = dim(temp_data)[1])

  # up-regulated genes for treat samples relative to untreated samples, with p-value <=0.05 and fold-change >=2
  up_genes = row.names(annova_com$table)[which((as.numeric(annova_com$table[[4]])<=0.05)&(as.numeric(annova_com$table[[1]])>=log2(2)))]

  # down-regulated genes for treated samples relative to untreated samples, with p-value <=0.05 and fold-change >=2
  down_genes = row.names(annova_com$table)[which((as.numeric(annova_com$table[[4]])<=0.05)&(as.numeric(annova_com$table[[1]])<=log2(0.5)))]

}
```

Windows-R script

6 Construction of gene regulatory network

SOFTWARE

NCA algorithm 2.3

COMMAND

```
# Construction of gene regulatory network
# Construction of initial gene regulatory network based on co-expression information  measured using the Pearson correlations.
# R method 'cor.test()' was used to calculate Pearson correlation coefficient (PCC)

# variable 'TFs_diff' indicates differentially regulated TF
TF_locs = match(TFs_diff,row.names(cpm_reads_new))

# variable 'TGs_diff' indicates differentially regulated TGs
TG_locs = match(TGs_diff,row.names(cpm_reads_new))

TF_TG_matrix = matrix(0,length(TFs_diff),length(TGs_diff))

for(i in 1:length(TFs_diff))
{
   for(j in 1:length(TGs_diff))
  {
     cor_value = cor.test(cpm_reads_new[TF_locs[i],], cpm_reads_new[TG_locs[j],],method = "pearson")
     if(cor_value$p.value<0.01)
     {
       TF_TG_matrix[i,j] = cor_value$estimate
     }
  }
}

row.names(TF_TG_matrix) = TFs_diff
colnames(TF_TG_matrix) = TGs_diff

# Any pairs of TF-TG with PCC <=0.7 were filtered out
TF_TG_matrix[which(abs(TF_TG_matrix)<=0.7)] = 0

#preparation data for the input of NCA algorithm

# generate gene expression data
gene_sample_X = NULL

unique_group = unique(sample_group)
for(i in 1:length(unique_group))
{
  gene_sample_X = cbind(gene_sample_X,apply(cpm_reads_new[,which(sample_group== unique_group[i])],1,mean))
}

#generate binary regulatory matrix
gene_regula_new = TF_TG_matrix
gene_regula_new[which(gene_regula_new != 0)] = 1

gene_regulatory_input = t(gene_regula_new)
row.names(gene_regulatory_input) = colnames(TF_TG_matrix)
colnames(gene_regulatory_input) = row.nanes(TF_TG_matrix)
```

For each stress type, a pair of transcriptional factors (TF) and genes with Pearson correlation coefficients (PCC) > 0.70 was predicted to have an regulatory relationship.

Windows - R script

COMMAND

```
# run script 'RunNCAToolbox.m' and then follow each step in NCA algorithm
```

Input gene expression matrix and gene regulatory relationship into NCA algorithm, we can get transcription factor activity and control strength.

Windows - Matlab

7 Validation of predicted TGs for each TF

SOFTWARE
GOSemSim 2.0.4

SOFTWARE
org.Hs.eg.db 3.6.0

SOFTWARE
foreach 1.4.4

SOFTWARE
doParallel 1.0.11

COMMAND

```
# To quantify functional relevance of the predicted TGs for each TF, we have calculated GO semantic similarity scores of the GO terms between each pair of the co-regulated TGs, using the GOSemSim package

#here, we considered only the GO biological processes

# Here, we just mannually constructed the relationship between genes and go terms, focused on biological process (BP)

# step1, for all BP, construct this relationship.
```

```
go_annos = readLines(paste(go_paths,"GO_annotation",sep=""))
GO_idss = NULL
go_genss = NULL

for(i in 2:length(go_annos))
{
  temps = unlist(strsplit(go_annos[i],"\t"))
  if(temps[4]=="P")
  {
    GO_idss = c(GO_idss,temps[2])
    go_genss = c(go_genss,temps[1])
  }

}

unique_gens = unique(go_genss)
max_tas = 0
for(i in 1:length(unique_gens))
{
  mac_is = which(unique_gens[i]==go_genss)
  if(length(mac_is)>max_tas)
  {
    max_tas = length(mac_is)
  }

}

matrix_Genes_gos = matrix(NA,length(unique_gens),max_tas)
all_in_go = NULL

for(i in 1:length(unique_gens))
{
  mac_is = which(unique_gens[i]==go_genss)
  matrix_Genes_gos[i,1:length(mac_is)] = GO_idss[mac_is]

}

row.names(matrix_Genes_gos) = paste(unique_gens,".v4.1",sep="")

# calculate the Go function similarity for any two BP GO term

library(GOSemSim)
library("org.Hs.eg.db", lib.loc="~/R/x86_64-redhat-linux-gnu-library/3.4")

GO_datass = list(NULL,NULL,NULL)
wang_simia = list(NULL,NULL,NULL)

### all BP Go term similarity
hsGO <- godata('org.Hs.eg.db', ont="BP")

go1_BP = unique(GO_idss)
go2_BP = unique(GO_idss)

wang_simia_BP = mgoSim(go1_BP, go2_BP, semData=hsGO, measure="Wang", combine=NULL)

# The function for the calculation of go term similarity of any two genes (G1 and G2)

gene_similarity <- function(G1,G2,similarty)
{
  max_simia1 = rep(0,length(G1))
  max_simia2 = rep(0,length(G2))

  max_simia1_1 = rep(1,length(G1))
  max_simia2_2 = rep(1,length(G2))

  for(i in 1:length(G1))
  {
    G1_sima_max = similarty[which(row.names(similarty)==G1[i]),match(G2,colnames(similarty))]
    max_simia1[i] = max(G1_sima_max)
  }

  for(i in 1:length(G2))
  {
    G2_sima_max = similarty[which(row.names(similarty)==G2[i]),match(G1,colnames(similarty))]
    max_simia2[i] = max(G2_sima_max)
  }

  return((sum(c(max_simia1,max_simia2))/(sum(c(max_simia1_1,max_simia2_2)))))
}

# parallelly calculate the GO function similarity of TGs for each TF. Here, NCA_list is the variable of list, and each list is the gene regulatory matrix, with row for TF and column for TGs

library(foreach)
library(doParallel)

cl <- makeCluster(10)
registerDoParallel(cl, cores=10)

drought_tar_sima <- foreach(jjj=1:dim(NCA_list[[1]])[1], .combine='c') %dopar%
{
  temp_tae = colnames(NCA_list[[1]])[which(NCA_list[[1]][jjj,]!=0)]
  temp_gene_sima = NULL

  for(kk in 1:(length(temp_tae)))
  {
    for(kkk in (kk+1):length(temp_tae))
    {
      int1 = which(row.names(matrix_Genes_gos)==temp_tae[kk])
      int1 = which(row.names(matrix_Genes_gos)==temp_tae[kk])
      int2 = which(row.names(matrix_Genes_gos)==temp_tae[kkk])

      temp_go1 = matrix_Genes_gos[int1,][which(!is.na(matrix_Genes_gos[int1,]))]
      temp_go2 = matrix_Genes_gos[int2,][which(!is.na(matrix_Genes_gos[int2,]))]

      if((length(temp_go1)>0)&&(length(temp_go2)>0))
      {
        temp_gene_sima = c(temp_gene_sima,gene_similarity(temp_go1,temp_go2,wang_simia_total))
      }else{
        temp_gene_sima = c(temp_gene_sima,NA)
      }

    }
  }

  return(temp_gene_sima)

}

stopCluster(cl)

### generate one random network, with the number of edge is the same with the number of total connections as pairs of TGs for each TF

library("igraph")
set.seed(30)
g1_r_f <- erdos.renyi.game(25971, length(which(!is.na(drought_tar_sima))), type = "gnm")


g1_egde_r_f = get.edgelist(g1_r_f)

droght_gens = setdiff(row.names(matrix_Genes_gos),unique(c(row.names(NCA_list[[1]]),colnames(NCA_list[[1]]))))
g1_edge_names_r_f = cbind(droght_gens[g1_egde_r_f[,1]],droght_gens[g1_egde_r_f[,2]])


### parallelly calculate the GO function similarity of two genes connected by one edge.

library(foreach)
library(doParallel)

cl <- makeCluster(10)
registerDoParallel(cl, cores=10)

res2_p_r_drougt <- foreach(jjj=1:dim(g1_edge_names_r_f)[1], .combine='c') %dopar%
{

  int1 = which(row.names(matrix_Genes_gos)==g1_edge_names_r_f[,1][jjj])
  int2 = which(row.names(matrix_Genes_gos)==g1_edge_names_r_f[,2][jjj])

  temp_go1 = matrix_Genes_gos[int1,][which(!is.na(matrix_Genes_gos[int1,]))]
  temp_go2 = matrix_Genes_gos[int2,][which(!is.na(matrix_Genes_gos[int2,]))]

  if((length(temp_go1)>0)&&(length(temp_go2)>0))
  {
    return(gene_similarity(temp_go1,temp_go2,wang_simia_total))
    # genes_sima_random = c(genes_sima_random,gene_similarity(temp_go1,temp_go2,wang_simia))
  }else{
   return(0)
  }
}

stopCluster(cl)
```

Unix - R script

SOFTWARE
igraph 1.2.1

8 Validation of predicted TF-TG interactions

SOFTWARE
inparanoid 4.1

COMMAND

```
# run 'perl inparanoid.pl X.fa Y.fa'
```

The script 'inparanoid.pl' in the software 'inparanoid' was used to generate the orthologous groups between two species. We consider a predicted TF-TG interaction as validated if it has orthologous pairs of TF-TG interactions in the mannually collected predicetd relationships of other species.

Unix - Perl script

9 Dynamic regulatory maps

SOFTWARE
DREM 2.0.3

COMMAND

```
# run software 'drem.jar' file in the directory of DREM software installation directory.
```

The method of DREM takes as input fold-change (treated relative to untreated) time series data and transcription factor-gene interaction data, and produces as output a dynamic regulatory map.

Windows - Jave

10 GO enrichment analysis

SOFTWARE

topGO 2.30.1

COMMAND

```
### one defined function for GO enrichment analysis. here, ‘inter_genes’ for your interested genes

cal_enrichments <- function(GO_path,save_path_,inter_genes,codition){

  # note: here, inter_genes and all_used_gene need to have same naming rule.

  geneID2GO <- readMappings(file = paste(GO_path,"gene2_go",sep=""))
  all_used_gene = names(geneID2GO)

  geneList = factor(as.integer(all_used_gene %in% inter_genes))
  names(geneList) = all_used_gene

  # run test for Biological process

  GOdata_BP <- new("topGOdata", ontology = "BP", allGenes = geneList,
                annot = annFUN.gene2GO, gene2GO = geneID2GO)
  BP_fisher = runTest(GOdata_BP,algorithm = "classic", statistic = "fisher")
  allRes_BP <- GenTable(GOdata_BP, classic = BP_fisher,
                     orderBy = "weight", ranksOf = "classic", topNodes = length(which(BP_fisher@score< 0.01)))
  write.table(allRes_BP,file = paste(save_path_,codition,"_BP.txt",sep=""),sep="\t",quote = F,row.names = F )

  # run test for Molecular function

  GOdata_MF <- new("topGOdata", ontology = "MF", allGenes = geneList,
                annot = annFUN.gene2GO, gene2GO = geneID2GO)
  MF_fisher = runTest(GOdata_MF,algorithm = "classic", statistic = "fisher")
  allRes_MF <- GenTable(GOdata_MF, classic = MF_fisher,
                     orderBy = "weight", ranksOf = "classic", topNodes = length(which(MF_fisher@score< 0.01)))
  write.table(allRes_MF,file = paste(save_path_,codition,"_MF.txt",sep=""),sep="\t",quote = F,row.names = F )

  # run test for Cellular component

  GOdata_CC <- new("topGOdata", ontology = "CC", allGenes = geneList,
                annot = annFUN.gene2GO, gene2GO = geneID2GO)
  CC_fisher = runTest(GOdata_CC,algorithm = "classic", statistic = "fisher")
  allRes_CC <- GenTable(GOdata_CC, classic = CC_fisher,
                     orderBy = "weight", ranksOf = "classic", topNodes = length(which(CC_fisher@score< 0.01)))
  write.table(allRes_CC,file = paste(save_path_,codition,"_CC.txt",sep=""),sep="\t",quote = F,row.names = F )

}
```

GO term enrichment was conducted over a given gene set using the topGO R package, where all annotated genes in switchgrass was used as the background. A GO function is considered enriched if the p-value < 0.01.

Windows - R script