Feb 20, 2021

Genetic mechanisms associated with floral initiation and the repressive effect of fruit on flowering in apple (Malus x domestica Borkh.)

PLOS One

Chris Gottschalk[1], Songwen Zhang[1], Phil Schwallier[2], Sean Rogers[1], M. John Bukovac[1], Steve van Nocker[1]

[1]Michigan State University; [2]Michigan State University Extension

1 Works for me dx.doi.org/10.17504/protocols.io.bp54mq8w

gottsc33

SUBMIT TO PLOS ONE

ABSTRACT

Many apple cultivars are subject to biennial fluctuations in flowering and fruiting. It is believed that this phenomenon is caused by a repressive effect of developing fruit on the initiation of flowers in the apex of proximal bourse shoots. However, the genetic pathways of floral initiation are incompletely described in apple, and the biological nature of floral repression by fruit is currently unknown. In this study, we characterized the transcriptional landscape of bourse shoot apices in the biennial cultivar, 'Honeycrisp', during the period of floral initiation, in trees bearing a high fruit load and in trees without fruit. Trees with high fruit load produced almost exclusively vegetative growth in the subsequent year, whereas the trees without fruit produced flowers on the majority of the potential flowering nodes. Using RNA-based sequence data, we documented gene expression at high resolution, identifying >11,000 transcripts that had not been previously annotated, and characterized expression profiles associated with vegetative growth and flowering. We also conducted a census of genes related to known flowering genes, organized the phylogenetic and syntenic relationships of these genes, and compared expression among homeologs. Several genes closely related to *AP1, FT, FUL, LFY,* and *SPLs* were more strongly expressed in apices from non-bearing, floral-determined trees, consistent with their presumed floral-promotive roles. In contrast, a homolog of *TFL1* exhibited strong and persistent up-regulation only in apices from bearing, vegetative-determined trees, suggesting a role in floral repression. Additionally, we identified four *Gibberellic Acid (GA) 2 oxidase* genes that were expressed to relatively high levels in apices from bearing trees. These results define the flowering-related transcriptional landscape in apple, and strongly support previous studies implicating both gibberellins and *TFL1* as key components in repression of flowering by fruit.

EXTERNAL LINK

https://doi.org/10.1371/journal.pone.0245487

THIS PROTOCOL ACCOMPANIES THE FOLLOWING PUBLICATION

Gottschalk C, Zhang S, Schwallier P, Rogers S, Bukovac MJ, van Nocker S. Genetic mechanisms associated with floral initiation and the repressive effect of fruit on flowering in apple (Malus x domesica Borkh.). PLoS One (in review)

DOI

dx.doi.org/10.17504/protocols.io.bp54mq8w

EXTERNAL LINK

https://doi.org/10.1371/journal.pone.0245487

PROTOCOL CITATION

Chris Gottschalk, Songwen Zhang, Phil Schwallier, Sean Rogers, M. John Bukovac, Steve van Nocker 2021. Genetic mechanisms associated with floral initiation and the repressive effect of fruit on flowering in apple (Malus x domestica Borkh.). **protocols.io**
https://dx.doi.org/10.17504/protocols.io.bp54mq8w

MANUSCRIPT CITATION please remember to cite the following publication along with this protocol

Gottschalk C, Zhang S, Schwallier P, Rogers S, Bukovac MJ, van Nocker S. Genetic mechanisms associated with floral initiation and the repressive effect of fruit on flowering in apple (Malus x domesica Borkh.). PLoS One (in review)

LICENSE

CREATED

Nov 27, 2020

LAST MODIFIED

Feb 20, 2021

PROTOCOL INTEGER ID

44956

GUIDELINES

The scripts provided in this protocol provide the framework for the work conducted. Subsets of data were regularly generated and executed in these scripts to decrease computation time. If you have questions concerning the methods please contact Chris Gottschalk.

MATERIALS TEXT

This work was conducted using the RNAseq libraries that can be retrieved from https://www.ncbi.nlm.nih.gov/sra/?term=SAMN04239699.

Index reference genome

1 Download reference genome Source:

High-quality apple reference genome

2 Index the reference genome fasta using HISAT2

hisat2-build

```
hisat2-build GDDH13_1-1_formatted.fasta GDDH13
```

index reference genome

HISAT 2.1.0
source by Kim et al.

Read Alignment

3 Retrieve RNAseq libraries

Files are locally maintained but can be retrieved from NCBI SRA database

SRA Files

4 Filter RNAseq reads for quality and removal of the adapter sequences

Fastq-mcf 1.04.636
source by http://www.q2labsolutions.com

fastq-mcf

```
fastq-mcf -t 0.10 -p 15 -l 20 -q 25 <adapters.fa> <reads.fq>
```

Filtering RNAseq reads for quality and removal of the adapter sequences

5 Align RNAseq reads to reference genome assembly

Read Alignment

```
hisat2 -q --dta-cufflinks --un-conc-gz /unmapped/<RNAseq_library>.fq -x <path to reference genome index> -1 <forward_read_file>.mcf -2 <reverse_read_file>.mcf -S <output alignment file>.sam
```

HISAT 2.1.0
source by Kim et al.

Transcriptome Assembly

6 Convert SAM alignment files into binary format and sort

SAMtools 1.3.1
source by Li et al.

SAM to BAM converstion and alignment sorting

```
samtools sort -o <sorted and converted alignment file name>_sort.bam
<alignment file>.sam
```

Converting SAM to BAM format and sort

7 Assembly of transcripts and annotation of individual transcriptomes

StringTie 1.3.3
source by Pertea et al.

Assembly of transcriptome

```
stringtie <sorted alignment file>.bam -G
M_domestica_genomes/GDDH13_v1.1/gene_models_20170606.gff3 -o <output
trancriptome for individual library>.gtf
```

Assembling transcripts using the reference transcriptome as a guide

8 Assembling merged transcriptome

stringtie -merge

```
stringtie --merge -G
M_domestica_genomes/GDDH13_v1.1/gene_models_20170606.gff3 -o <output
merged transcriptome>.gtf <text file of individual transcriptome annotation
files>.txt
```

Merging individual transcriptomes using reference annotation as a guide

StringTie 1.3.3
source by Pertea et al.

Transcriptome Statistics and Characterization

9

gffcompare

```
gffcompare -r M_domestica_genomes/GDDH13_v1.1/gene_models_20170606.gff3 -
o <merged honeycrisp trasncriptome>.gtf
```

Comparing Honeycrisp transcriptome to the reference transcriptome

gffcompare 0.9.12
source by Pertea and Pertea

10 Quality assessment of the transcriptome assembly

RNA seq QC

```
rnaseqc <merged Honeycrisp annotation>.gtf <individual sorted alignment
files>.bam <output folder name>
```

Assessing the quality of transcriptome assembly

RNA-SeQC 1.1.8
source by Getz et al.

11 Predicting coding potential of assembled transcripts

Filtering for novel features

```
#first we want to extract all class codes associated with novel features that do
not represent mapping or assembly errors
#This includes the following list:
#u= unknown/intergenic
#o= same strand with overlap on reference exon
#i= contained within reference intron
#x= exonic overlap but on opposite strand
#e= singleton, overlapping intron, possible pre-mRNA
#y= contains a reference within intron(s)

awk '/"u"|"o"|"i"|"x"|"e"|"y"/ { print }' gffcompare_out.annotated.gtf >
novel_annotations.gtf
awk '/"c"|"s"|"k"|"m"|"n"|"j"/ { print }' gffcompare_out.annotated.gtf >
isoform_annotations.gtf

#11,264 transcripts were collected





#second we will filter list into two files, one contains only transcripts of >200 nt
and the other contains transcripts of <200 nt

awk '{ if ($5-$4>200) print $0 }'  novel_annotations.gtf >
novel_annotations_200.gtf
#11,205 transcripts were collected

awk '{ if ($5-$4<201) print $0 }'  novel_annotations.gtf >
novel_annotations_smallRNA.gtf
#59 transcripts were collected

#extract the sequences for these novel annotations >200nt for use in
classification
gffread -w novel_annotations_200.fa -g ~/Genomes/GDDH13_v1.1/GDDh13.fa
novel_annotations_200.gtf

#reference CDS, coding, and noncoding gene models sequences

awk '!/ncRNA/ { print }' gene_models_20170606.gff3 > ref_coding.gff3
awk '/ncRNA/ { print }' gene_models_20170606.gff3 > ref_nc.gff3
gffread ref_nc.gff3 -T -o ref_nc.gtf
gffread ref_coding.gff3 -T -o ref_coding.gtf
awk '/CDS/ {print}' ref_coding.gtf > CDS.gtf
gffread -w ref_CDS.fa -g GDDH13.fa CDS.gtf
gffread -w ref_nc_seqs.fa -g GDDH13.fa ref_nc.gtf
gffread -w ref_coding.fa -g GDDH13.fa ref_coding.gtf
```

Filtering transcriptome for novel features based on the gffcompare annotations

gffread 0.11.7
source by Pertea and Pertea

11.1 Predicting coding potential using CPC2

CPC2

```
cd $CPC_HOME && python ./bin/CPC2.py -r -i /novel_annotations_200.fa -o CPC2_output.txt
```

Predicting coding potential using CPC2

CPC2 beta
by Kang et al.

11.2 Predicting coding potential using PLEK

PLEK

```
python PLEK.py -range maize_ens_linli.range -model maize_ens_linli.model -fasta novel_annotations_200.fa -out PLEK_output.txt
```

Predicting coding potential using PLEK

PLEK 1.2
by Li and Zhang

11.3 Predicting coding potential using CPAT

CPAT

```
make_hexamer_tab.py -c ref_CDS.fa -n ref_nc_seqs.fa > GDDH13_hexamer.tsv
make_logitModel.py -x GDDH13_hexamer.tsv -c ref_coding.fa -n
ref_nc_seqs.fa -o GDDH13_logitmodel
cpat.py -g novel_annotations_200.fa -d
~/Genomes/GDDH13_v1.1/GDDH13_logitmodel.logit.RData -x
~/Genomes/GDDH13_v1.1/GDDH13_hexamer.tsv -o CPAT_output
```

Predicting coding potential using CPAT

CPAT 1.2.4
source by Wang et al.

Calculate Transcript Abundance and Differential Expression

12 Calculate transcript abundance

cuffquant

```
cuffquant -o <output folder> <merged honeycrisp transcriptome>.gtf <sorted
alignment file>.bam
```

Quantify transcriptome adundance

cufflinks 2.2.1
source by Trapnell et al.

13 Perform DEG analysis

cuffdiff

```
cuffdiff -o <output folder name> -b < reference sequence GDDH13.fa> -L NTFB,TF
HC_GDDH13_Transcriptome/cuffquant/HC_C30_cuffq_out/abundances.cxb HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C4_cuffq_out/abundances.cxb HC_GDDH1
HC_GDDH13_Transcriptome/cuffquant/HC_C15_cuffq_out/abundances.cxb,HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C22_cuffq_out/abundances.cxb,HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C14_cuffq_out/abundances.cxb,HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C27_cuffq_out/abundances.cxb,HC_GDDH
```

Differential expression analysis

cufflinks 2.2.1
source by Trapnell et al.

14 Normalize the quantified expression

Normalize the quantified expression

```
cuffnorm -o <output folder name> -b < reference sequence GDDH13.fa> -L NTFB,T
HC_GDDH13_Transcriptome/cuffquant/HC_C30_cuffq_out/abundances.cxb HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C4_cuffq_out/abundances.cxb HC_GDDH1
HC_GDDH13_Transcriptome/cuffquant/HC_C15_cuffq_out/abundances.cxb,HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C22_cuffq_out/abundances.cxb,HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C14_cuffq_out/abundances.cxb,HC_GDDH
HC_GDDH13_Transcriptome/cuffquant/HC_C27_cuffq_out/abundances.cxb,HC_GDDH
```

cufflinks 2.2.1
source by Trapnell et al.

General transcript annotation

15 Retrieve transcript FASTA sequences

Transcript sequence retrieval

```
gffread -w transcripts_HC.fa -g M_domestica
_genomes/GDDH13_v1.1/GDDH13.fa <merged Honeycrisp transcriptome>.gtf
```

Retrieve transcript FASTA sequence from transcriptome

gffread 0.11.7
source by Pertea and Pertea

16 Identify homologous peptide sequences between Honeycrisp/Apple and Arabidopsis

TAIR10_pep_20101214_updated 2012-04-16

blastx

```
blastx -query transcripts_HC.fa -db ~/blast/db/TAIR10_pep/TAIR10_pep -out
hc_blast_out -evalue 10 -outfmt "6 qseqid sseqid evalue bitscore" -
max_target_seqs 1
```

Identification of homology between Honeycrisp and Arabidopsis

BLAST+ 2.7.1
source by Camacho et al.

17 Reciprocal blast to identify high confidence homologs of flowering genes

This was done using in-house scripts. If interested please contact Steve van Nocker @ vannocke@msu.edu

18 Predicting coding sequences of target transcripts identified by blast homology

Model Longest ORFs and predict representative peptides

```
TransDecoder.LongOrfs –t HC_GDDH13/transcript_models/merged_transcripts.fa

TransDecoder.Predict -t HC_GDDH13/transcript_models/merged_transcripts.fa

cdna_alignment_orf_to_genome_orf.pl
HC_GDDH13/transcript_models/merged_transcripts.fa.transdecoder.gff3
Transcriptome/HC_GDDH13/transcript_models/ merged.gff3
HC_GDDH13/transcript_models/merged_transcripts.fa >
HC_GDDH13/transcript_models/ transcript.fasta.transdecoder.mfusca.gff3
```

Transdecoder 5.5
source by Haas

19 Phylogentics of target genes

Phylogenetic tree construction

```
xvfb-run ete3 build -w standard_fasttree -a AGL24_compiled.pep.fa -o AGL24_tree
```

Construction of a phylogenetic tree for a specific gene family (example script)

ETE3 3.1.1
source by Huerta-Cepas et al.

20 Syntenic Analysis

BLAST+ 2.7.1
source by Camacho et al.

MCScanX
by Wang et al.

20.1

Homology file creation

```
MCScanX_h HC_GDDH13/annotation/collinearity/ATxGDDH13
```

Concatenate a homology file that is a two column text file the has homologs from both
species documented ATxGDDH13.homology this contains my blastp tophits and blast results generated earlier
ex. AT1G00000 MD00G0000000
need to have a concatenated gff file of Arabidopsis and GDDH13

20.2

Interspecific synteny analysis

```
blastall -i HC_GDDh13.pep -d
~/blast/Malus_domestica/HC_GDDH13_pep/HC_GDDH13_pep -p blastp -e 1e-10
-b 5 -v 5 -m 8 -o MdxMd.blast
MCScanX /HC_GDDH13/annotation/collinearity/MdxMd
```

Intraspecific analysis
Step 1
perform a within species synteny analysis
blast p all honeycrisp apple longest ORFs vs apple longest ORFs
reformat a gff so that it only contains chr gene id start stop = MdxMd.gff

20.3

Duplicate gene cluster classification

```
duplicate_gene_classifier HC_GDDH13/annotation/collinearity/ATxGDDH13
```

Classifying duplicate gene clusters

20.4

Detect collinearity within gene families

```
perl detect_collinearity_within_gene_families.pl -i
HC_GDDH13/annotation/collinearity/gene_family.txt -d
HC_GDDH13/annotation/collinearity/MdxMd.collinearity -o
HC_GDDH13/annotation/collinearity/col_families.txt
```

Detect collinearity within gene families

de novo unmapped transcript assembly

21 Assembly of transcripts from reads that failed to map to the reference genome

de novo transcript assembly

```
Trinity --seqType fq --max_memory 20G --samples_file <list of unmapped fastq
files>.txt --output /HC_denovo/Trinity
```

Reads that failed to align to the reference genome (unmapped) were de novo assembled using Trinity.

Trinity Assembler 2.4.0
source by Haas et al.

22 Construct gene and isoform models

Model genes and isoforms

```
Trinity_gene_splice_modeler.py --trinity_fasta Trinity.fasta
```

Trinity_gene_splice_modeler.py
source by Haas et al.

Index de novo transcripts

23 Create index using de novo transcripts

HISAT2-build de novo

```
hisat2-build
/Users/vannockerlab/ChrisG/HC_GDDH13/unmapped_reads_assembly/gene_transcri
HC_denovo
```

Creating index of de novo transcripts for RNAseq read alignment

HISAT 2.1.0
source by Kim et al.

Align unmapped reads to de novo transcripts

24 Alignment of unmapped reads

Align unmapped reads

```
hisat2 -q --dta-cufflinks -x
HC_GDDH13/unmapped_reads_assembly/trinity_fasta_index/HC_denovo -1
HC_GDDH13/unmapped_reads_assembly/mapping/unmapped_reads/<unmapped
reads forward>.fq.gz -2
HC_GDDH13/unmapped_reads_assembly/mapping/unmapped_reads/<unmapped
reads reverse>.fq.gz -S <alignment output>.bam
```

HISAT 2.1.0
source by Kim et al.

25 Sort the alignments

SAM to BAM converstion and alignment sorting

```
samtools sort -o <sorted and converted alignment file name>_sort.bam
<alignment file>.sam
```

Converting SAM to BAM format and sort

samtools 1.9
source by http://htslib.org/

Quantify expression de transcripts

26 Quantify expression per unmapped read library using cuffquant

cuffquant

```
cuffquant -p 4 -o cuffquauntout
unmapped_reads_assembly/gene_transcript_models/trinity_genes.gtf
unmapped_reads_assembly/mapping/alignment_to_trinity/<library>unmapped_out.b
```

Quantification of de novo transcripts

cufflinks 2.2.1
source by Trapnell et al.

27 Retrieve normalized expression data

cuffnorm

```
cuffnorm -o norm_out -L NTFB,TFB,NT15,T15,NT35,T35,NT50,T50,NT70,T70
unmapped_reads_assembly/gene_transcript_models/trinity_genes.gtf
./cuffquant/C30/abundances.cxb ./cuffquant/C31/abundances.cxb ./cuffquant/C21/a
./cuffquant/C4/abundances.cxb ./cuffquant/C28/abundances.cxb,./cuffquant/C2/abu
./cuffquant/C15/abundances.cxb,./cuffquant/C16/abundances.cxb ./cuffquant/C23/a
./cuffquant/C22/abundances.cxb,./cuffquant/C3/abundances.cxb
./cuffquant/C14/abundances.cxb,./cuffquant/C24/abundances.cxb
./cuffquant/C27/abundances.cxb,./cuffquant/C13/abundances.cxb,./cuffquant/C25/a
```

Normalize quantified expression

cufflinks 2.2.1
source by Trapnell et al.

Transcript annotation of de novo transcripts

28 Assigning annotation using BLAST from various databases

BLAST+ 2.7.1
source by Camacho et al.

28.1 TAIR10 Arabidopsis annotation

BLASTx

```
blastx -query unmapped_reads_assembly/gene_transcript_models/trinity_genes.fasta -db TAIR10_pep/TAIR10_pep -out hc_denovo_blast_out -evalue 10 -outfmt "6 qseqid sseqid evalue bitscore" -max_target_seqs 1 -num_threads 2
```

BLASTx de novo transcripts vs TAIR10 peptides

28.2

BLASTn

```
blastn -query trinity_genes.fasta -db ~/blast/db/nt/nt -out blastn_out -outfmt "6 qseqid sseqid evalue bitscore score pident nident length staxids " -max_target_seqs 1
```

BLASTn de novo transcripts vs NCBI nt database

28.3

Retrieve nt annotation descriptions

```
blastdbcmd -db ../blast/db/nt/nt -dbtype nucl -entry_batch ./HC_denovo_blastn_out_id.txt -out hc_de_novo_nt__descript_out -outfmt "%a %t"
```

28.4

BLASTx

```
blastx -query trinity_genes.fasta -db ~/blast/db/nr/nr -out blastx_out -outfmt "6 qseqid sseqid evalue bitscore score pident nident length staxids " -max_target_seqs 1
```

BLASTx de novo transcripts vs NCBI nr database

29 Model longest ORFs and predict representative transcripts from de novo transcript dataset

Transdecoder de novo transcripts

```
TransDecoder.LongOrfs -t unmapped_reads_assembly/trinity_genes.fasta

TransDecoder.Predict -t unmapped_reads_assembly/trinity_genes.fasta

cdna_alignment_orf_to_genome_orf.pl
unmapped_reads_assembly/trinity_genes.fasta.transdecoder.gff3
unmapped_reads_assembly/trinity_genes.gff3
unmapped_reads_assembly/trinity_genes.fasta >
../transcripts.fasta.transdecoder.denovofusca.gff3
```

Transdecoder 5.5
source by Haas

R commands for transcriptome analysis

30

R Studio Desktop 1.1.463
by The R Studio, Inc.

R programming language 3.3.3 of later
source by The R Foundation

CummeRbund 2.24.0
source by Goff et al.

CummeRbund

```
library(cummeRbund)
setwd("HC_GDDH13/cuffdiff/processed_cuffdiff_files/cuffdiff_edited") #setworking
#creation of a cufflinks db for cummeRbund
#cuff<-readCufflinks(dir =
"HC_GDDH13/cuffdiff/processed_cuffdiff_files/cuffdiff_edited", gtfFile =
"HC_GDDH13/transcript_models/merged.gtf", genome =
"M_domestica/GDDH13_v1.1/GDDH13")
#reload already created cuff database
#cuff <- readCufflinks(File = "cuffData.db", gtfFile =
"HC_GDDH13/transcript_models/merged.gtf", genome =
"M_domestica/GDDH13_v1.1/GDDH13")
#check cuffset instances; should contain genes and isoforms other features
could be missing
#cuff
#check cuffset is functioning by buidling dendrogram of genes in treatments
#dend<-csDendro(genes(cuff))
#dend
```

This is the primary scripts used to set up the CummeRbund and R based analyses.

30.1

Clustering of DEGs in the thinned apices

```
#targets with foldchange >1 <-1
fctargets1 <- subset(subsample_initiation_get_genes, log2_fold_change >=
1.5 | log2_fold_change <= -1.5)
fctargets2 <- unique(fctargets1$gene_id)
subsamplefinal_initiation_get_genes <- getGenes(cuff,fctargets2,
sampleIdList = samplelist)
initiation_sig_genes_cluster5 <-
csCluster(subsamplefinal_initiation_get_genes, k=5)
initiation_sig_genes_cluster5
thinned_sig_genes_clusterplot5 <-
csClusterPlot(initiation_sig_genes_cluster5)
thinned_sig_genes_clusterplot5
cluster_assignment_initiation5 <- initiation_sig_genes_cluster5$clustering
write.csv(cluster_assignment_initiation5, file =
"initiating_gene_cluster_assignment5_fc1.5", sep = ",", eol = "\n")
```

Clustering of DEGs in the thinned apices

30.2

Heatmap for thinned apices

```
#heatmap for sig genes ordered by cluster assignment using this to sort by
cluster for the heatmap (reorganized the steve annotation/heatmap.txt file)
#import csv to data to heatmap.txt
row.names <- (heatmap_updated$Homolog)
heat_map_data <- data.matrix(heatmap_updated)
gene_exp <- heat_map_data[,2:6]
row.names <- heatmap_updated$Homolog
row.names(gene_exp) <- row.names
my_palette <- colorRampPalette(c("White", "Red"))(n = 299)
hm_heatmap <- heatmap.2(gene_exp, Rowv = NULL, Colv = NULL, col =
my_palette, trace = "none", density.info = "none", margins = c(16,12))
hm_heatmap
```

30.3

MdTFL1 expression plots

```
tfl1 <- data.frame(tfl1.2_plot)
tfl1_2_plot <- ggplot(tfl1, aes(x=Timepoint, y=Exp, color=Source,
group=Source)) + geom_line(stat = "Identity") + scale_colour_brewer(type =
"div") + geom_errorbar(aes(ymin=Exp-StdE, ymax=Exp+StdE, width=.2))
tfl1_2_plot

cor.test(~ PCR + RNAseq, data = cor, method = "pearson", conf.level = 0.95)

tfl1_rna <- data.frame(tfl1.2_plot_rna)
tfl1_qpcr <- data.frame(tfl1.2_plot_qpcr)

#tfl1_qpcr_pval <- compare_means(Relative.Expression ~ Source, group.by =
"Timepoint", data = tfl1_qpcr_all, p.adjust.method = "bon")
#tfl1_qpcr_pval

tfl1_2_plot1 <- ggplot(tfl1_rna, aes(x=Timepoint, y=FPKM, group=Source,
fill=Source)) + geom_bar(stat = "Identity", position = position_dodge()) +
geom_errorbar(aes(ymin=FPKM-StdE, ymax=FPKM+StdE, width=.2), position
= position_dodge(width=0.9)) + theme_gray()#+ geom_text()
tfl1_2_plot2 <- ggplot(tfl1_qpcr, aes(x=Timepoint, y=Relative.Expression,
color=Source, group=Source)) + geom_line(stat = "Identity") +
geom_errorbar(aes(ymin=Relative.Expression-StdE,
ymax=Relative.Expression+StdE, width=.2)) + theme_gray() #+
stat_compare_means(method = "t.test")



tfl1_2_plot1
tfl1_2_plot2

tfl1_plots <- ggarrange(tfl1_2_plot2, tfl1_2_plot1, ncol = 1, nrow = 2, labels =
c("A","B"))
tfl1_plots

###### TFL1-1 plots
tfl1_1 <- data.frame(tfl1.1_plot)
tfl1_1_plot <- ggplot(tfl1_1, aes(x=Timepoint, y=Exp, color=Source,
group=Source)) + geom_line(stat = "Identity") + scale_colour_brewer(type =
"div") + geom_errorbar(aes(ymin=Exp-StdE, ymax=Exp+StdE, width=.2))
tfl1_1_plot

cor.test(~ PCR + RNAseq, data = cor2, method = "pearson", conf.level =
0.95)

tfl1_1_rna <- data.frame(tfl1.1_plot_rna)
tfl1_1_qpcr <- data.frame(tfl1.1_plot_qpcr)

#tfl1_1_qpcr_pval <- compare_means(Exp ~ Source, group.by = "Timepoint",
data = tfl1.1_plot_qpcr_all, p.adjust.method = "bon")
#tfl1_1_qpcr_pval

tfl1_1_plot1 <- ggplot(tfl1_1_rna, aes(x=Timepoint, y=Exp, group=Source,
fill=Source)) + geom_bar(stat = "Identity", position = position_dodge()) +
geom_errorbar(aes(ymin=Exp-StdE, ymax=Exp+StdE, width=.2), position =
position_dodge(width=0.9)) + theme_gray() #+ geom_text()
tfl1_1_plot2 <- ggplot(tfl1_1_qpcr, aes(x=Timepoint, y=Exp, color=Source,
group=Source)) + geom_line(stat = "Identity") +
geom_errorbar(aes(ymin=Exp-StdE, ymax=Exp+StdE, width=.2)) +
theme_gray() #+ stat_compare_means(method = "t.test")


tfl1_1_plot1
tfl1_1_plot2

tfl1_1_plots <- ggarrange(tfl1_1_plot2, tfl1_1_plot1, ncol = 1, nrow = 2, labels
= c("A","B"))
tfl1_1_plots

#### TFL1 plots all

Figure_6 <- ggarrange(tfl1_2_plot2,tfl1_1_plot2,tfl1_2_plot1,tfl1_1_plot1, ncol
= 2, nrow = 2, labels = c("A","B", "C", "D"))
Figure_6
```

30.4

Clustering for DEG in response to crop load

```
ic <- csCluster(targets, k=5)
head(ic$cluster)
icp <- csClusterPlot(ic)
icp
```

Used the same scripts as the clustering for the initiating apices (csCluster). However, the dataset here was manually curated to cluster by fold change of only the DEGs. Simplified R commands found below.

30.5

Heatmap for crop load fold change

```
row.names <- (heatmap_cropload_final$Ref_gene)
cl_fc <- data.matrix(heatmap_cropload_final)
gene_exp_cl_fc <- cl_fc[,2:6]
row.names <- cl_fc$Ref_gene
row.names(gene_exp_cl_fc) <- row.names
my_palette <- colorRampPalette(c("Blue", "White", "Red"))(n = 299)
cl_fc_heatmap <- heatmap.2(gene_exp_cl_fc, Rowv = TRUE, Colv = NULL, col
= my_palette, trace = "none", density.info = "none",  key = "True", margins =
c(8,6))
```

Heatmap for GA2OX genes

```
row.namesGAox2 <- (GAox2$gid)
heat_map_dataGAox2 <- data.matrix(GAox2)
gene_expGAox2 <- heat_map_dataGAox2[,2:6]
row.namesGAox2 <- GAox2$gid
row.names(gene_expGAox2) <- row.namesGAox2
my_palette <- colorRampPalette(c("Blue","White", "Red"))(n = 299)
hm_heatmapGAox2 <- heatmap.2(gene_expGAox2, Rowv = NULL, Colv =
NULL, col = my_palette, trace = "none", density.info = "none", margins =
c(16,12))
hm_heatmapGAox2
```

WGCNA

31

Co-expression analysis script

```
setwd("~/R/WGCNA/")
setwd("~/R/WGCNA/")

library("WGCNA")
library("flashClust")


options(stringsAsFactors = FALSE)
enableWGCNAThreads()
wgcna_cropload <- read.delim("./rep_data.txt", row.names=1)
traitsData <- read.delim("./datTraits.txt", row.names=1)
datExpr = as.data.frame(t(wgcna_cropload))

gsg = goodSamplesGenes(datExpr, verbose = 3)
gsg$allOK
#if last statement is False using following to remove offending genes
if (!gsg$allOK)
  {if (sum(!gsg$goodGenes)>0)
      printFlush(paste("Removing genes:", paste(names(datExpr)
[!gsg$goodGenes], collapse= ", ")));
      if (sum(!gsg$goodSamples)>0)
         printFlush(paste("Removing samples:", paste(rownames(datExpr)
[!gsg$goodSamples], collapse=", ")))
      datExpr= datExpr[gsg$goodSamples, gsg$goodGenes]
      }

Samples = rownames(datExpr);
allTraits = traitsData[, c(1:3) ];

traitRows = match(Samples, allTraits$sample);
datTraits = allTraits[traitRows,];
rownames(datTraits) = datTraits[, 1]



sampleTree = hclust(dist(datExpr), method = "average")
traitColors = numbers2colors(traitRows, signed = FALSE);
pdf(file = "SampleClustering.pdf");
plotDendroAndColors(sampleTree, traitColors,
groupLabels = names(traitRows),
main = "Sample dendrogram and trait heatmap")
dev.off()

save(datExpr, datTraits, file = "Single_Block_data_entry_check.RData")

# Choose a set of soft-thresholding powers
powers = c(c(1:10), seq(from = 12, to=30, by=2))
# Call the network topology analysis function
sft = pickSoftThreshold(datExpr, powerVector = powers, verbose = 5)
# Plot the results:
sizeGrWindow(9, 5)
par(mfrow = c(1,2));
cex1 = 0.9;
# Scale-free topology fit index as a function of the soft-thresholding power
```

```
pdf(file = "Soft_power_calc.pdf");
plot(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
xlab="Soft Threshold (power)",ylab="Scale Free Topology Model Fit,signed
R^2",type="n",
main = paste("Scale independence"));
text(sft$fitIndices[,1], -sign(sft$fitIndices[,3])*sft$fitIndices[,2],
labels=powers,cex=cex1,col="red");
# this line corresponds to using an R^2 cut-off of h
abline(h=0.90,col="red")
# Mean connectivity as a function of the soft-thresholding power
plot(sft$fitIndices[,1], sft$fitIndices[,5],
xlab="Soft Threshold (power)",ylab="Mean Connectivity", type="n",
main = paste("Mean connectivity"))
text(sft$fitIndices[,1], sft$fitIndices[,5], labels=powers, cex=cex1,col="red")
dev.off()

#We select a soft power of 4 as  is the lowest power for which the scale-free
topology fit index reaches 0.90

softPower = 4;
adjacency = adjacency(datExpr, power = softPower);
TOM = TOMsimilarity(adjacency);
dissTOM = 1-TOM

# Call the hierarchical clustering function
geneTree = hclust(as.dist(dissTOM), method = "average");
# Plot the resulting clustering tree (dendrogram)
pdf(file = "Clustering_dendro.pdf");
sizeGrWindow(12,9)
plot(geneTree, xlab="", sub="", main = "Gene clustering on TOM-based
dissimilarity",
labels = FALSE, hang = 0.04);
dev.off()

# We like large modules, so we set the minimum module size relatively high:
minModuleSize = 30;
# Module identification using dynamic tree cut:
dynamicMods = cutreeDynamic(dendro = geneTree, distM = dissTOM,
deepSplit = 2, pamRespectsDendro = FALSE,
minClusterSize = minModuleSize);
table(dynamicMods)

# Convert numeric lables into colors
dynamicColors = labels2colors(dynamicMods)
table(dynamicColors)
# Plot the dendrogram and colors underneath
sizeGrWindow(8,6)
pdf(file = "Clustering_dendro_with_colors.pdf");
plotDendroAndColors(geneTree, dynamicColors, "Dynamic Tree Cut",
dendroLabels = FALSE, hang = 0.03,
addGuide = TRUE, guideHang = 0.05,
main = "Gene dendrogram and module colors")
dev.off()
```

```
genesMatchcolors <- cbind(dynamicColors, names(datExpr))
write.csv(genesMatchcolors, file="gene_names_dynamicColors_match.csv")


# Calculate eigengenes
MEList = moduleEigengenes(datExpr, colors = dynamicColors, softPower = 4)
MEs = MEList$eigengenes
# Calculate dissimilarity of module eigengenes
MEDiss = 1-cor(MEs);
# Cluster module eigengenes
METree = hclust(as.dist(MEDiss), method = "average");
# Plot the result
sizeGrWindow(7, 6)
plot(METree, main = "Clustering of module eigengenes",
xlab = "", sub = "")

MEDissThres = 0.25
# Plot the cut line into the dendrogram
abline(h=MEDissThres, col = "red")
# Call an automatic merging function
merge = mergeCloseModules(datExpr, dynamicColors, cutHeight = MEDissThres,
verbose = 3)
# The merged module colors
mergedColors = merge$colors;
# Eigengenes of the new merged modules:
mergedMEs = merge$newMEs;

sizeGrWindow(12, 9)
pdf(file = "Clustering eigengenes.pdf", wi = 9, he = 6)
plotDendroAndColors(geneTree, cbind(dynamicColors, mergedColors),
c("Dynamic Tree Cut", "Merged dynamic"),
dendroLabels = FALSE, hang = 0.03,
addGuide = TRUE, guideHang = 0.05)
dev.off()


# Rename to moduleColors
moduleColors = mergedColors
# Construct numerical labels corresponding to the colors
colorOrder = c("grey", standardColors(50));
moduleLabels = match(moduleColors, colorOrder)-1;
MEs = mergedMEs;
# Save module colors and labels for use in subsequent parts
save(MEs, moduleLabels, moduleColors, geneTree, file = "Single_Block-
networkConstruction-stepByStep.RData")

write.csv(cbind(MEList, names(datExpr)), file="gene_names_module_match.csv")
write.csv(MEs, file="Module_Eigengenes_dataframe.csv")
```

These scripts were executed on the HPCC at the MSU.

WGCNA

source by Zhang and Horvath

R programming language 3.3.3 of later

source by The R Foundation