**Department of Animal and Plant Sciences**

**University of Sheffield**

Yellow stickers

**Registration Number**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 8 | 0 | 1 | 6 | 3 | 9 | 3 | 1 |

**Module number**

|  |  |
| --- | --- |
| APS | 61013 |

**Title**

|  |
| --- |
| APS61013 Lab Notebook |

**Word count**

|  |
| --- |
| 6173 |

*By submitting this document I certify that the work submitted is all my own, except where specifically indicated (e.g. by referencing, or quotation), and confirm that I have read and understood the University's rules relating to plagiarism. I understand that the Department reserves the right to run spot checks on all coursework using plagiarism software.*

**APS61013 - Joe’s methodology notebook.**

\*For references to programmes used and the sources for data collected see main paper.

\*Orthologues/orthogroups refer to groups of genes with related ancestry.

*Alloteropsis semialata* is a species of grass which is known to have received genes via lateral gene transfer (LGTs) from various different grass lineages. It is geographically diverse and reference genomes have been collected from a multitude of distinct accessions.

However, the regulation of LGTs present within *A. semialata* is still unexplored. This project aims to rectify this by quantifying the expression of LGTs in *A. semialata* and compare their expression to the native genes of their original donor species *(Setaria italica* or *Themeda triandra*), and the native genes of the *A. semialata* accession which received the LGT. Ultimately asking 1) is the expression of LGTs more similar to donor or recipient orthologues and 2) are LGTs uniquely diverged from their donor orthologues in terms of expression. The first question lets us understand if regulation is more influenced by the gene (if its closer to donor expression) or the host (if its closer to recipient expression). The second question can answer whether LGTs are differentially regulated after arriving in a new species.

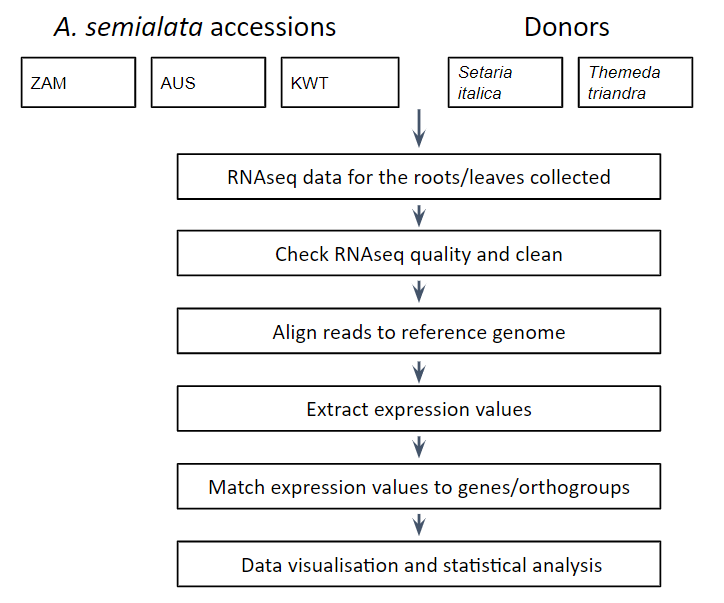


Figure 1**.** A basic schematic of steps taken during this project.

**Data collected from previous studies:**

Prior to this project RNAseq data was extracted from replicants of various *A. semialata* accessions and donors (Table 1). FASTA files for both the root and leaf were prepared and then used in our analysis (Figure 2). RNAseq analysis enables us to determine the transcriptome (the full range of messenger RNA, or mRNA, molecules expressed by an organism) for each sample.

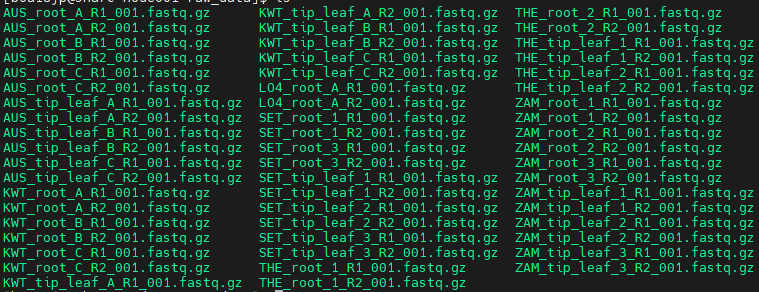


Figure 2. A list of raw FASTA files acquired at the start of this project. L04 data was not analysed in this study. R1 indicates forward reads and R2 paired reads. Each accession is further identified by area (root or leaf) and replicate (1, 2, 3 or A, B, C).

|  |  |  |  |
| --- | --- | --- | --- |
| Species | Root replicates | Tip leaf replicates | Reference genome |
| *A. semialata* (AUS) | 3 | 3 | *A. semialata* (AUS) |
| *A. semialata* (ZAM) | 3 | 3 | *A. semialata* (ZAM) |
| *A. semialata* (KWT) | 3 | 3 | *A. semialata* (KWT) |
| *Setaria italica* | 2 | 3 | *Setaria italica* |
| *Themeda triandra* | 2 | 2 | *Sorghum bicolor* |

Table 1. Species, the number of replicates and the reference genome the species was aligned to in our study. AUS = Australia, ZAM = Zambia, KWT = South Africa.

|  |  |  |
| --- | --- | --- |
| Species | Reference genome(s) | Availability |
| *A. semialata* (AUS) | ASEM\_C4\_v1.0.CDS.fasta | Dunning *et al.* (2019) |
| *A. semialata* (KWT) | KWT\_v1.0.all.maker.CDS.fasta | Raimondeau *et al.* (*in press*) |
| *A. semialata* (ZAM) | ASEM\_ZAM15-05-10.v1.0.all.maker.cds.fasta | Raimondeau *et al.* (*in press*) |
| *S. italica* | Setaria\_italica.JGIv2.0.cds.all.fa | http://ftp.ensemblgenomes.org/pub/plants/release-37/fasta/ |
| *T. triandra* | Sorghum\_bicolor.Sorghum\_bicolor\_v2.cds.all.fa  Sbicolor\_454\_v3.1.1.cds.fa | <http://ftp.ensemblgenomes.org/pub/plants/release-37/fasta/>  https://phytozome-next.jgi.doe.gov/info/Sbicolor\_v3\_1\_1 |

**Table 2.** A list of reference genomes used in this study. *A. semialata* reference genomes were received directly from the author, while both *S. italica* and *S. bicolor* reference genomes are available online.

**After data was collected, the following steps were performed (also see figure 1):**

1. Filtering and trimming sequence reads -> in this step we checked the quality of data (using FastQC), trimmed the data to remove low quality bases/adapter sequences (using trimmomatic) and then checked the quality again.
   * Adapter sequences are short oligonucleotides used to be ligated to the ends of DNA fragments of interest, so that they can be combined with primers for amplification.
2. Aligning reads to a reference genome -> in this step we aligned paired reads to a reference genome (using bowtie2), which were composed of coding sequence data (.cds).
3. Quantifying expression of reads -> in this step we quantified expression for the roots and leaves of all samples (using eXpress).
4. Assigning gene names and orthogroups -> in this step we matched our quantified expression data to genes, donor lineages and orthogroups.
5. Data visualisation and statistical analysis -> in this step we visualised data in graphs and performed appropriate statistical tests relevant to our hypothesis.

Steps 1-3 were performed using the University of Sheffield’s high performance computing cluster (<https://docs.hpc.shef.ac.uk/en/latest/>). This uses a BASH shell (linux). Steps 4-5 were performed using excel, perl and R (with the tidyverse package).

**Acquiring data from hpc:**

Raw data was acquired from the ‘raw\_data’ and ‘references’ directories in ‘/fastdata/bo1lpg/RNAseq-alloteropsis’ on the hpc.

The ‘raw\_data’ directory contains .fastq.gz data for 56 different accessions of grass. There are six different accessions (AUS, KWT, LO4, SET, THE, ZAM) and these accessions include multiple samples (A, B, C or 1, 2, 3), different sample regions (leaf or root) and forward or reverse reads (R1 or R2 respectively).

The ‘references’ directory contains four .fasta and one .fa file (AUS(C4), LO4B (not used in this study, but name refers to previous member of lab), ZAM. KWT, SET (*Setaria italica)* and a *Sorghum bicolor* reference). These provide the reference (CDS) data which allows us to align a sequence to a reference genome.

Files from these two were copied into my personal directory ‘/fastdata/boa18jp/boa18jp/RNAseq-alloteropsis’ either into the relevant /accession/ directory or the /reference\_genomes/ directory.

**Cleaning data:**

The first step in processing this data is to check its quality using fastqc, a quality control tool for raw sequence data coming from high throughput sequencing pipelines. The raw data for each accession is placed in the relevant accession directory’s /raw/ directory (E.g. AUS/raw, ZAM/raw/ and so on), while the fastqc output is placed within the relevant directory's /raw\_fastqc/ directory. All scripts are in the /scripts/ directory of each accession.

The script is as follows (fastqc.sh).

#!/bin/bash

#$ -l h\_rt=0:30:00

#$ -l rmem=2G

#$ -j y

#$ -o fastqc.log

#$ -pe smp 8

source /usr/local/extras/Genomics/.bashrc

DIR="/fastdata/boa18jp/masters/RNAseq-alloteropsis/accessions"

DIR2= "{PWD##\*}"

inputString=""

space=" "

for i in $DIR/$DIR2 raw/\*.fastq.gz

do

inputString=$inputString$i$space

done

fastqc -o /fastdata/boa18jp/masters/RNAseq-alloteropsis/accessions/LO4/raw\_fastqc/ -t 4 $inputString

* Couldn’t figure out how to make the output folder automatically be the subdirectory, so manually did it, change LO4 to other accessions/species as needed.
  + I did later do a better automation script for an eXpress script however, so check that for insight on how you might do it for fastqc.

Note each accession directory is identical to each other, aside from raw data files. This script allows fastqc processing for each accession.

**Trimming data (and recleaning):**

The following script was used to run trimmomatic (trimmomatic.sh).

#!/bin/bash

#$ -l h\_rt=0:30:00

#$ -l rmem=2G

#$ -j y

#$ -o trimmomatic.log

#$ -t 1-12

source /usr/local/extras/Genomics/.bashrc

#set the path to trimmomatic

ProgramPath="/usr/local/extras/Genomics/apps/trimmomatic/current"

#get a list of all the forward reads

SAMPLE1=($(ls raw/\*R1\_001.fastq.gz))

#get a list of all the reverse reads

SAMPLE2=($(ls raw/\*R2\_001.fastq.gz))

#set up an index for the reads/tasks

INDEX=$((SGE\_TASK\_ID-1))

# run trimmomatic on the 3 sets

java -jar $ProgramPath/trimmomatic-0.38.jar PE -phred33 ${SAMPLE1[$INDEX]} ${SAMPLE2[$INDEX]} ${SAMPLE1[$INDEX]}.out\_paired\_50bp.fastq.gz ${SAMPLE1[$INDEX]}.out\_unpaired\_50bp.fastq.gz ${SAMPLE2[$INDEX]}.out\_paired\_50bp.fas$

For this script you have to manually move the output of trimmomatic (in my case from ‘raw’ to ‘trimmed’). Afterwards fastqc the trimmed data using the same script as fastqc.sh but replace the directory ‘raw’ with ‘trimmed’ and the output ‘raw\_fastqc’ with ‘trimmed\_fastqc’.

**Align RNA-seq reads to reference:**

Use bowtie2 -> an aligner that takes reads from next-generation sequencing and maps them to a reference genome. This is so we can identify genes, in other studies this could be used to monitor mutations between an individual and their reference genome.

The first step in bowtie2 is to index reference genomes:

Index reference genomes: bowtie2-build ASEM\_L04B\_v1.0.all.maker.cds.fasta L04 (or other reference genome).

The output is: L04.1.bt2, L04.2.bt2, L04.3.bt2, L04.4.bt2, L04.ref.1.bt2, L04.rev.2.bt2

Do for each accession! Run in aligned directory too.

bowtie2 works like: bowtie2 [options]\* <index file> <read1> <read 2>.

E.g. bowtie2 --very-fast-local -x KWT -1 ../trimmed/KWT\_root\_B\_R1\_001.fastq.gz.out\_paired\_50bp.fastq.gz -2 ../trimmed/KWT\_root\_B\_R2\_001.fastq.gz.out\_paired\_50bp.fastq.gz -S ../aligned/SAM/KWT\_root\_B.sam

--- For other accessions (varies by sample name, sample letter/number, root or leaf\_tip)

bowtie2 --very-fast-local -x SET \

-1 ../trimmed/SET\_tip\_leaf\_3\_R1\_001.fastq.gz.out\_paired\_50bp.fastq.gz \

-2 ../trimmed/SET\_tip\_leaf\_3\_R2\_001.fastq.gz.out\_paired\_50bp.fastq.gz \

-S ../aligned/SAM/SET\_tip\_leaf\_3.sam

---

bowtie2 --very-fast-local -x SET \

-1 ../trimmed/SET\_root\_3\_R1\_001.fastq.gz.out\_paired\_50bp.fastq.gz \

-2 ../trimmed/SET\_root\_3\_R2\_001.fastq.gz.out\_paired\_50bp.fastq.gz \

-S ../aligned/SAM/SET\_root\_3.sam

bowtie2

This outputs a L04.sam file (make sure to name these clearly, L04 only has one pair has data extracted from roots).

For the purposes of this study we don’t need to convert SAM files to BAM.

**Reading expression data using express:**

\*This section contains both a description of my original script (that requires manually moving files to the appropriate location), and an updated version along with how the fixes work.

Find express here (<https://pachterlab.github.io/eXpress/manual.html>) or acquire on hpc using:

wget<https://pachterlab.github.io/eXpress/downloads/express-1.5.1/express-1.5.1-linux_x86_64.tgz> tar xzvf express-1.5.1-linux\_x86\_64.tgz express-1.5.1-linux\_x86\_64/./express

Express then can be run by typing express-1.5.1-linux\_x86\_64/./express wherever it is installed on hpc.

File Usage: express [options] <target\_seqs.fa> <hits.(sam/bam)>

express-1.5.1-linux\_x86\_64/express ASEM\_L04B\_v1.0.all.maker.cds.fasta SAM/LO4\_root\_A.sam

--- Another example:(editing as easier than typing in console)

express-1.5.1-linux\_x86\_64/express ASEM\_C4\_v1.0.CDS.fasta SAM

SAM/AUS\_root\_B.sam -o express/AUS\_root\_B --no-bias-correct

--- (creates new directory but doesn’t rename files)

Produces the following files: params.xprs, results.xprs

(try add -o /express/ to see if it sends output directly to that directory, ALSO see if can change names of params.xprs and results.xprs

All express data processed, all has some results! Also, made a script to work express on all files (but it's a bit dodgy in terms of output location).

express.sh

#!/bin/bash

#$ -pe mpi 3

#$ -t 3

#$ -l h\_rt=1:00:00

#$ -l rmem=4G

#$ -j y

#$ -P training

#$ -q training.q

#$ -wd /fastdata/boa18jp/masters/RNAseq-alloteropsis/accessions/THE/aligned/

source /usr/local/extras/Genomics/.bashrc

sam=(/fastdata/boa18jp/masters/RNAseq-alloteropsis/accessions/THE/aligned/SAM/\*)

n=(0)

one=(1)

for i in $(ls /fastdata/boa18jp/masters/RNAseq-alloteropsis/accessions/THE/aligned/SAM/\*);

do

express-1.5.1-linux\_x86\_64/express Sbicolor\_454\_v3.1.1.cds.fa ${sam[$n]} -o express/${sam[$n]##\*/} --no-bias-correct

n=$(($n + $one))

done;

#code works as follows:

#get list of alignment .sam files ($sam)

#assign n to 0

#assign one to 1

#make loop to run express for all the alignment files (for i) on the specified reference genome, beginning at file '0' in the list and increasing by one (as n increases by 1 each loop)

#output directory created in the /express/ directory, with each output file having the same name as the respective .sam file, minus the directory path (##\*/ does this)

The issue with this script is it recreates all the directories of the relevant .sam file in its output. To fix this issue, move the specific directories which contain the params.xprs and results.xprs files to the /express/ directory and delete the subdirectories created by the script. I also removed the .sam from the directory names to keep consistency, although this isn’t necessary (E.g. mv THE\_tip\_leaf\_2.sam/ THE\_tip\_leaf\_2).

Also note I changed my results.xprs filenames to show which accession it was for (E.g. THE\_tip\_leaf\_2\_results.xprs). I done this for all files after downloading them.

#FIXED! By changing the output to -o express/${sam[$n]##\*/} the code outputs correctly. The first / is required as otherwise the output directory becomes expressSAMFILENAME, this makes it express/SAMFILENAME and the ##\*/ is what specifically removes the basepath (what I otherwise call the directory)

Files were further modified using excel as .csv files, and grouped by replicate and type of gene for example ‘AUS\_LGT.csv’ contains LGT expression data for each AUS replicate, including both root and leaf tip values (which were also calculated as averages). See supplementary material of the main paper for the full list of files. Additionally, genes were identified from a list of genes present in TableS4 of Raimondeau et al. (*in press*) and grouped using trees also from Raimondeau et al. with ‘retrieve\_names.pl’ which can be found in the supplementary material. This script was made by Dr. Pascal Antoine-Christin.

After data was organised I compared them to table S4 (description found below) from Raimondeau et al. in order to get donors, and I checked the gene IDs against tree data from Raimondeau. This allowed me to select the correct reference genomes (a list of which can be found on the main paper, with the source included) and in the case of *T triandra* two reference genomes was used to accommodate the tree data in Raimondeau et al.

Afterwards, I used excel to calculate mean root and mean leaf expression for each species (an average of replicate expression) as well as standard errors and standard deviations (though se and sds weren’t used in the project.

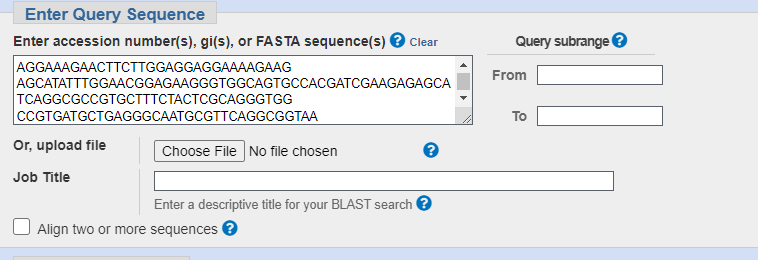
Data was then visualised using R and various non-parametric tests were performed. This R script is available in the supplementary information of the main paper and structured in order of graphs plotted, while also containing all statistical tests. The code runs perfectly line by line as of 08/06/22. Some parts are perhaps inefficiently done, but it works fine. A description of the R script can be found below.

Identifying genes:

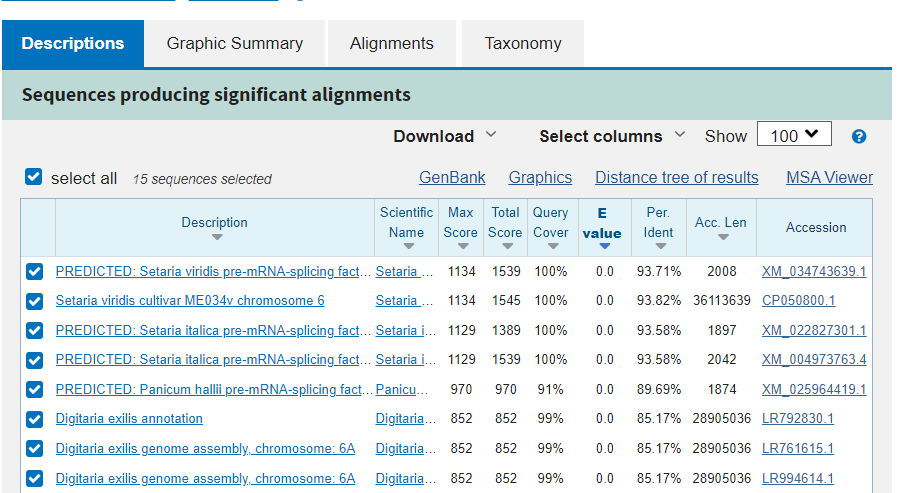
Genes were identified using NCBI BLAST [Nucleotide BLAST: Search nucleotide databases using a nucleotide query (nih.gov)](https://blast.ncbi.nlm.nih.gov/Blast.cgi?PROGRAM=blastn&BLAST_SPEC=GeoBlast&PAGE_TYPE=BlastSearch). To use this, find the gene you want to identify, for example ‘LGT-KWT3-06264’, open the KWT3 reference genome and CTRL + F ‘06264’ (because genes are just genes, not LGTs in the reference genome).



Then copy the nucleotide sequence into BLAST



… click BLAST at the bottom and wait.



Now you have a list of related genes, this gene has a Cenchrinae donor in our study so it appears to have worked.



Congratulations, you’ve used BLAST successfully! **Using R for statistical analysis and data visualisation.**

**\***Note only select examples were included as screenshots, as explanations are fairly similar across different parts of the script.

\*In my opinion it’s much easier to understand the R script by viewing the R script rather than this document. I elaborated on the goals and purpose of each section of the script here to give an overview of the statistical analysis as its specified in the lab book criteria.

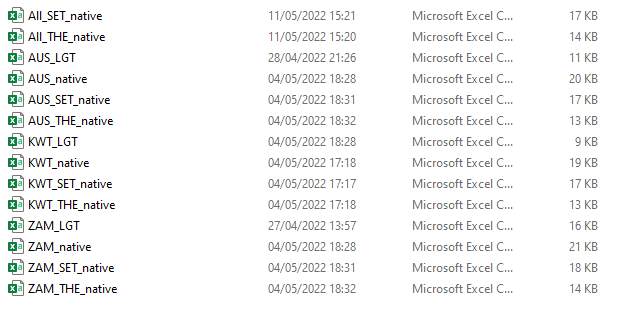
\*If you’re just here for the stats, CTRL+F [stats] will guide you.

\*I saved the ‘All\_accessions\_with\_donor’ dataframe as a .csv, which contains all expression data for every gene quantified in this study, this is so the R script doesn’t have to be run to look at expression data.

\*Some of this code is messy as most of this code I had to learn during my analysis

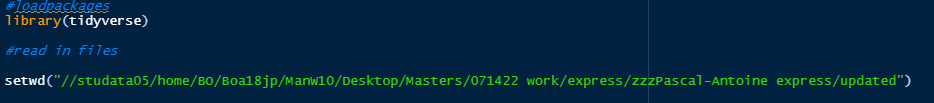
\* R version 4.1.0 was used

The following files are required for this script:



Each .csv file contains 179 rows (excluding headings) which correlates to an orthogroup. For example, the first row of results is orthogroup 1 for all accessions. This allows easy comparison between matching orthogroups. These files contained columns for genes, expression values for the roots and tip leaf of different samples, average expression for the roots and tip leaf, donor name, as well as se and sd values.

I made different native donor files, because in my original results I accidentally merged donor data in the distance plots. I’m not sure if this is needed, but it was a simple way to separate orthogroups by donor and accession then later merge them. Note that the ‘donor’ column is filled in with respect to orthogroups that are known to be donated. As these .csvs were later filtered by their respective donor, extra donors not used in this study were left in but are effectively the same as a blank space. The All\_\*\*\*\_native files contain donor data for all accessions.



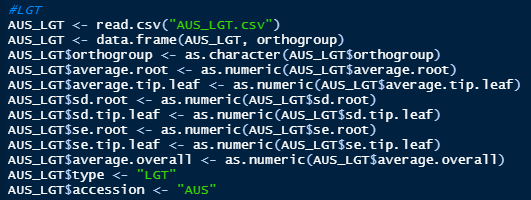
In terms of packages, only ‘tidyverse’ was used. I also added a new column ‘orthogroup’ to each file using R itself to let me assign a value to rows (this let me pair/match orthogroups across different gene types/accessions). After I loaded all the data from my main working directory I then changed my working directory so I could organise my figures into different folders. Although this is just for organisation and is not needed.



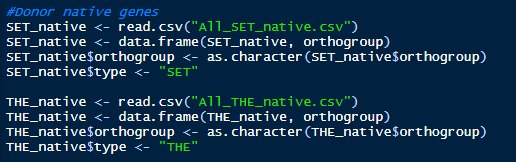
After loading tidyverse and setting my working directory, I then made a data frame for orthogroups, as each .csv was ordered in orthogroup this was a nice way to pair the data for later.



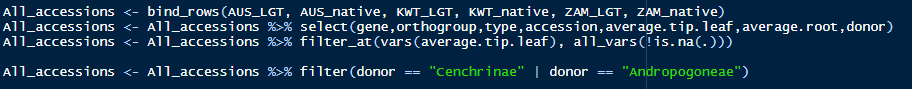
I had to set the orthogroup dataframe ‘as.character’ so it would be categorical and not numeric. If you don’t do this the code won’t work. Additionally certain .csv files required me to set the average root and tip leaf expression values to numeric (I set more, but these aren’t important as it’s much easier to calculate these values in R itself). Furthermore I set each respective .csv to an appropriate type and accession. Type is just the ‘orthogroup type’ so ‘Recipient’, ‘LGT’, ‘Donor’. There is a donor column in the actual .csv files which lists by ‘Andropogoneae’ or ‘Cenchrinae’ so I could filter by donor later on.



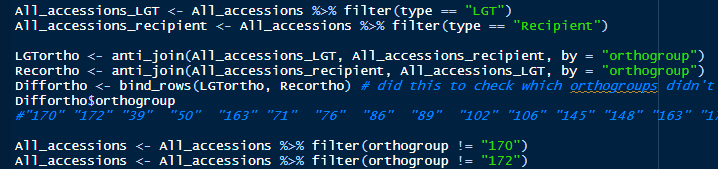
However, while the ‘<accession>\_SET\_native’ and <accession>\_THE\_native’ dataframes were set as type ‘Donor’… I did also set the SET\_native and THE\_native dataframes to either ‘SET’ or ‘THE’. This is because of how I originally organised my data, which I later changed. I kept in the old versions because I already had results for the expression plots so didn’t see the need to change it. It’s important to do this when I want to compare by ‘type’, as not to merge the SET/THE into the same dataset. All other dataframes created in the data preparation phase followed these general structures.



After these initial steps of data preparation, I made a new dataframe called ‘All\_accessions’. As you can see, I just bound the rows of all the LGT and native *A. semialata* files. I highly recommend selecting the columns I did below for the sake of clarity as I had a lot of columns by default. I also filtered to remove ‘na’ rows as these were useless, if a dataframe was ‘na’ for tip leaf it was na for root. Lastly, I filtered to only include donors from our two favourite clades ‘Cenchrinae’ and ‘Andropogoneae’.

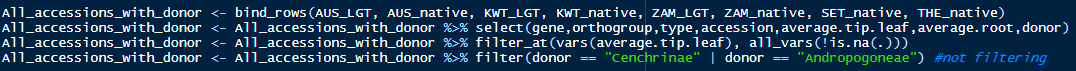


I also made new dataframes filtering for either ‘LGT’ or ‘Recipient’ orthologues, and then I used ‘anti\_join’ to check which groups didn’t match between these groups. I also used the count function to double check this (I used the count function for ALL filtering). Then I manually removed each orthogroup that wasn’t found in both the LGT and Recipient dataframes. There were 20 in total, see the full script for the whole list.

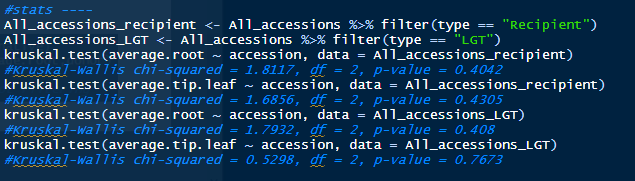


Afterwards I made ‘All\_accessions\_with\_donor’ which contains all accession data from donors, recipients and LGT .csv. I filtered this group to remove na, selected certain values and filtered to only include Cenchrinae and Andropogoneae.

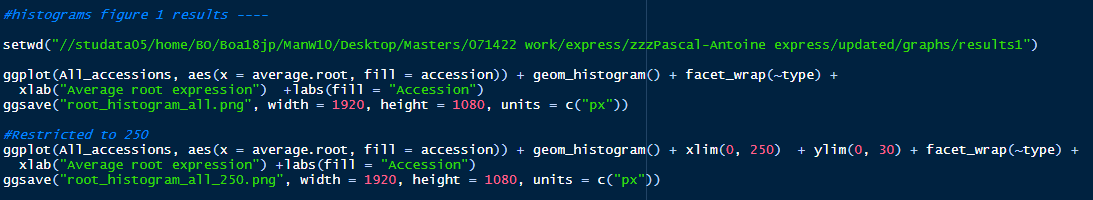
The #not filtering comment refers to the fact I did not filter this dataframe. This lets me have a consistent ‘unfiltered’ reference if I want to quickly look up any genes of interest. I also decided to save this data frame as a .csv and put it in the supplementary material, just so the entire R script doesn’t have to be run to view my raw data (although I always viewed the data on R as it was simple to use the filter feature on view(All\_accessions\_with\_donor)).



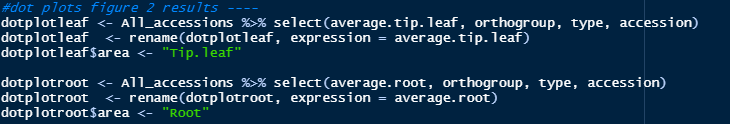
Now for our first [stats]. These stats are for figure 1 of the main paper. I chose a Kruskal-Wallis test as it’s non parametric (as the data was not normalised) and there are very few LGT orthogroups found across accessions (see figure 2 of main paper for just how few). Anyway, what this does is check the rank order of all my data, and I found there were no significant differences across accession data for the roots or tip leaf of LGTs or recipient genes. Success! Now we know it’s okay to use the accession data as one data frame.

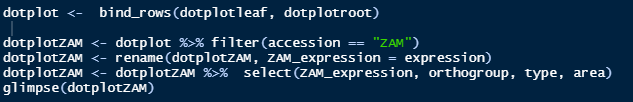


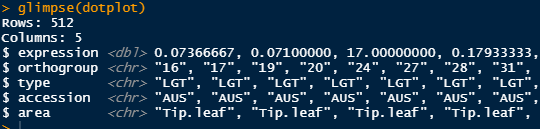
Right below these stats are how to make the histograms for figure 1. I used the All\_accession dataframe, so remember this was filtered to only include shared orthogroups and I also used xlim and ylim to filter the axis for visibility. I used ggsave 1920 x 1080 for every plot as it’s much easier to resize a big plot, then save a small plot then have to increase the size. Change average.root to average.tip.leaf to plot tip leaf data.



Now it’s time to prepare figure 2 – where I plot only data found in two of the three accessions. First I used the select function to make data frames which only had tip leaf or root expression, and then renamed them to ‘expression’ and added a new column to specify an area.

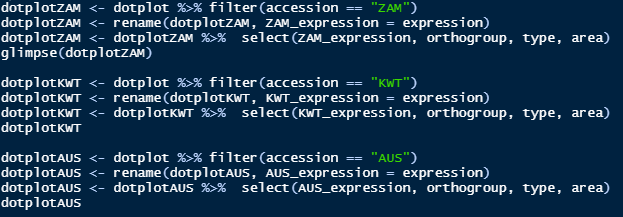
After this, I made another dataframe ‘dotplot’, which combined these two rows.





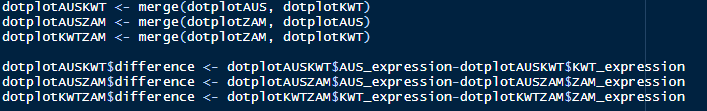
Now we have the combine rows, we can filter by accession I recommend keeping names consistent. A lot of my script I just copied from one accession and replaced the ‘ZAM’ with ‘KWT’ for example. This saves lots of time and let me write code for one accession at a time and then just repeat it.

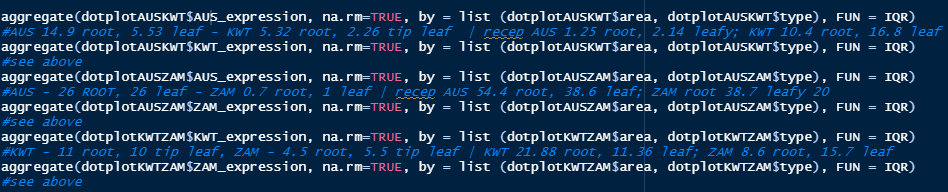
I also renamed the expression values to distinguish expression by accession.



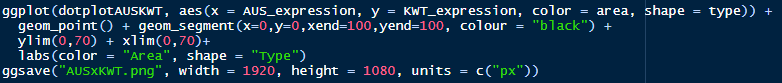
Then to get our accession vs accession dotplots, I merged all three combinations that were possible. I didn’t test differences for significance (these sample sizes are very small) but did look at the interquartile range of each accession to see how the patterns looked. The aggregate feature is very useful for sorting functions (IQR, sum, standard error, standard deviation among others). Initially I wrote giant scripts to do these calculations and it took me hours… just use aggregate for calculations!

The ‘merge’ feature also has the neat benefit of removing all orthogroups that were not found in both the dataframes I was merging. If bind\_rows() doesn’t do the job, sometimes merge() will.

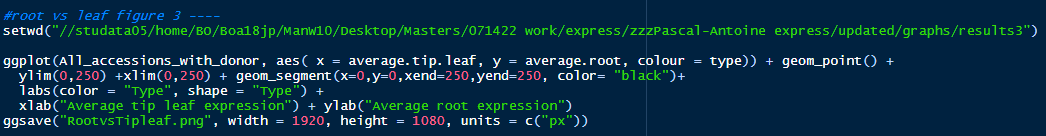




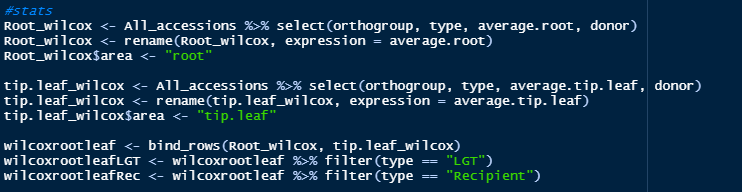
The dot plots for the accession plot is as follows, it’s very simple and I changed the x and y axis to best visualise the data (all of this data was represented in the graphs, just some of the expression varied). Change dotplotAUSKWT to one of the others to do the other two graphs.



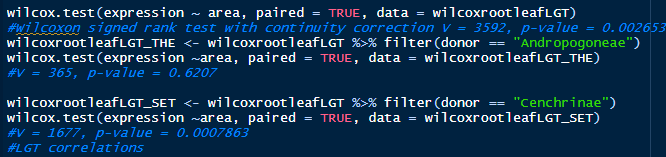
Right below this is how to do figure 3, very similar to figure 2 but just with root and tip leaf data against each other.



However the stats for this figure were a bit more complicated code-wise. I first filtered the All\_accession data and renamed the average.root/tipleaf to ‘expression’ as well as making ‘area’ columns. Then I bound these rows and filtered to specifiy LGT or recipient.



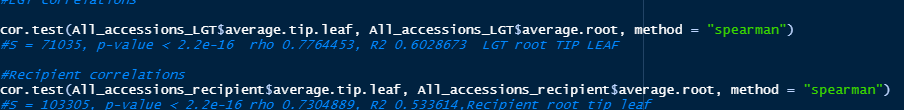
Note because I previously filtered All\_accessions, these statistical tests were paired. I performed three wilcox tests here (non-parametric). First for LGTs derived from BOTH Andropogoneae/Cenchrinae and then one test for LGTs derived from only Andropogoneae or Cenchrinae. This was important as I thought there might be some differences.



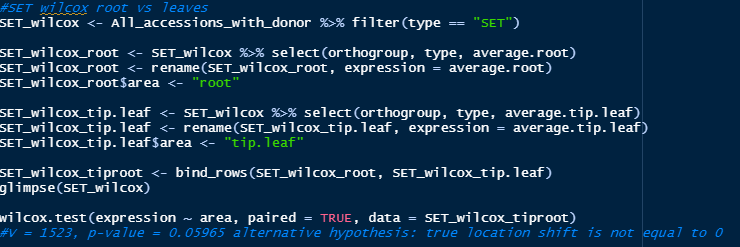
I also tested recipient expression by area, note this was not filtered per donor.



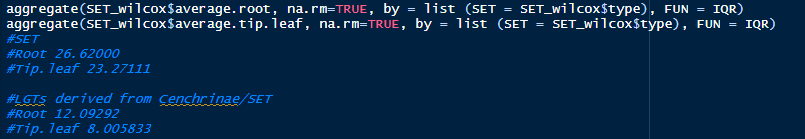
Now for spearman’s correlation tests (I chose these as non-parametric). My two initial tests looked at the root/leaf correlations of LGTs and recipients for orthogroups which had both LGTs/recipient genes. Ignore SET/THE\_spear as they were made without properly filtering the data.



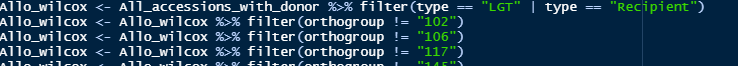
Here is the set up for the *S. italica* wilcox tests. The ‘SET\_wilcox\_tiproot’ dataframe has tip leaf and root data combined, let me pair the the data for the results associated with figure 3. The exact same was done for ‘THE’.



The interquartile ranges were also calculated from the SET/THE\_wilcox dataframes, as seen below (and used in Table 2 of main paper). All\_accessions wasn’t used for the IQR data for LGT/recipients in this table as I wanted to only include orthogroups also found in both donors and recipients.



Instead I prepared another data frame ‘Allo\_wilcox’ for this, filtered to only include accessions which had both LGT, recipient and donor orthologue data. I used the count() function to manually count and check which orthogroups matched… paired tests working indicated they were completely filtered.

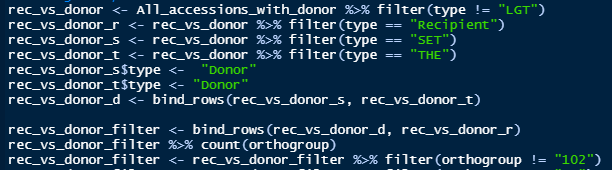


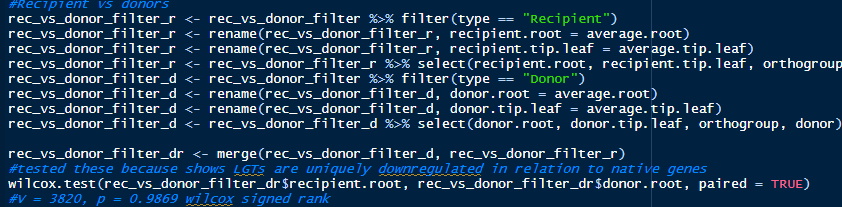


Correspondingly this was also used to test for significance between LGT and recipient root expression, pairing the data.

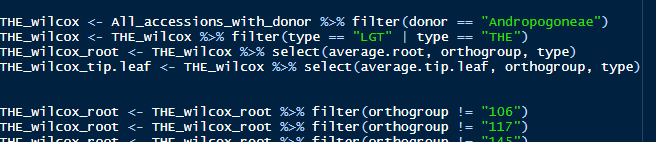


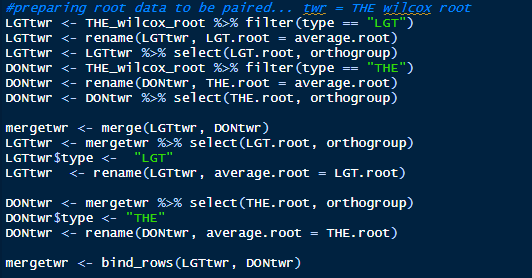
To test recipient vs donors I did similar filters for each orthogroup and recipient, and after some renaming of this data they were then suitable for paired Wilcox tests. I chose to do this additional test because I noticed the only Cenchrinae-derived LGTs had significantly higher root than leaf expression, while the Andropogoneae ones didn’t and actually had numerically higher leaf expression. By comparing the donors and recipients via a wilcox test I could see whether this could be due to unique expression divergences between the two groups. Significance was not found.





The next step of the script is re-preparing data for proper correlation tests and some LGT vs donor comparisons. Filtering to only include orthogroups with both LGT/donor data. This script was done for both THE (i.e. twl = THE wilcox root) and SET, and individually for the roots and tip leaf.

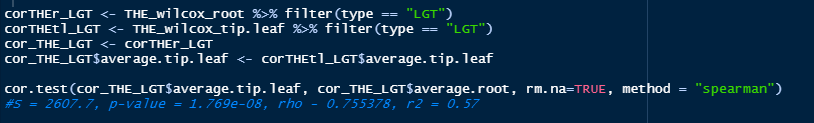




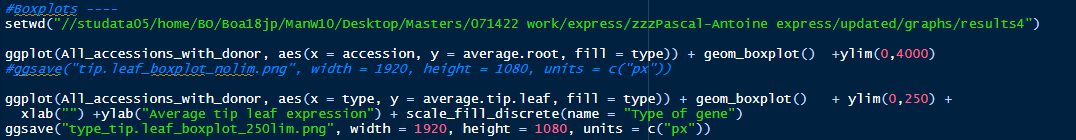
Now data was paired we could look at root comparisons for the LGTs and THE donors in this case (same done for tip leaf and SET, see code).



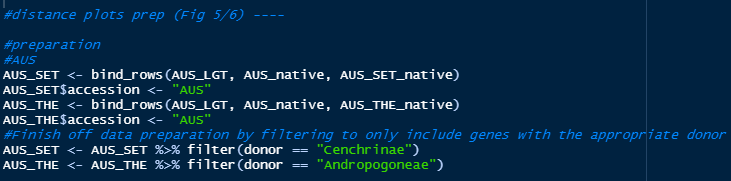
Now correlation tests were performed with some additional filters and recombining of columns to ensure the data matched. By adding the tip leaf data to the root data frame… we had matched tip leaf and root expression for our correlations. Some extra wilcox tests were also performed for recipients vs SET/THE but aren’t useful to the study.



After all that data preparation, some simple box plots were made for figure 4 (which plotted root and tip leaf expression across accessions). Again certain filters were used, and remember boxplots natively plot the IQR data (I didn’t and spent hours trying to figure out how to do this with geom\_point).



Now it’s time for the preparation for the distance plots of figure 5/6. Each accession was combined with its LGT, recipient and relevant THE/SET\_native.csv. This is messy code, but these were actually the first plots created and I didn’t want to have to re-write them. The reason I distinguished AUS\_SET\_native and so on was because our initial results triplicate the data (so it looked like Cenchrinae LGTs were significantly diverged… after correcting this our p value was 0.505, and a 94.995% confidence interval is not significant, but it’s a sign that with more data there could be significance).



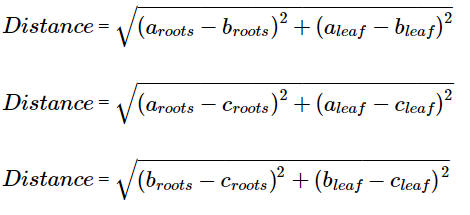
Afterwards, distances were calculated by the input below, and the following calculation (and the equation below I made using latex – see https://www.overleaf.com/ for more details).

This calculated the distance of LGTs to recipients/donors, and the distance of recipients to LGTs/donors, used in figures 5/6 (and our first and second aim of the study) respectively.

I also re-assigned orthogroups as these datasets were filtered, I think this wouldn’t be needed if I wrote the data a bit tidier, but the orthogroups are just ways to group genes and their actual numbers are unimportant for the distances. There were 111 distance measurements for SET and 62 for THE… if you run the code line by line there isn’t any issues despite me making dist\_count\_orthogroups for both dataframes.







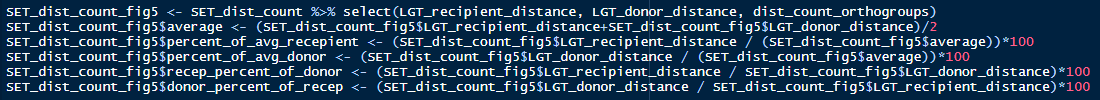
ALL\_SET$LGT\_recipient\_distance <- sqrt((SET\_LGT$average.root-SET\_recipient$average.root)^2+(SET\_LGT$average.tip.leaf-SET\_recipient$average.tip.leaf)^2)

ALL\_SET$LGT\_donor\_distance<- sqrt((SET\_LGT$average.root-SET\_donor$average.root)^2+(SET\_LGT$average.tip.leaf-SET\_donor$average.tip.leaf)^2)

ALL\_SET$donor\_recipient\_distance <- sqrt((SET\_donor$average.root-SET\_recipient$average.root)^2+(SET\_donor$average.tip.leaf-SET\_recipient$average.tip.leaf)^2)

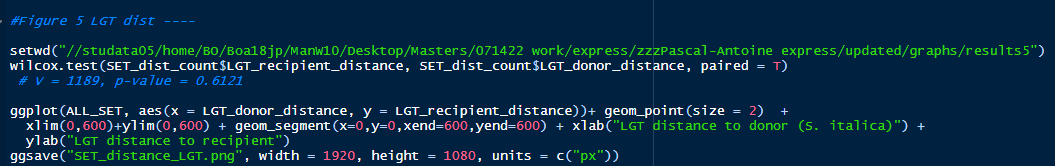
For calculating the percentage values (i.e. how many recipient/donor distances to LGTs were within 20% of the average distance value to LGTs of that orthogroup) and how many genes were more than twice as close to one of the other two orthogroups, the following calculations was used.

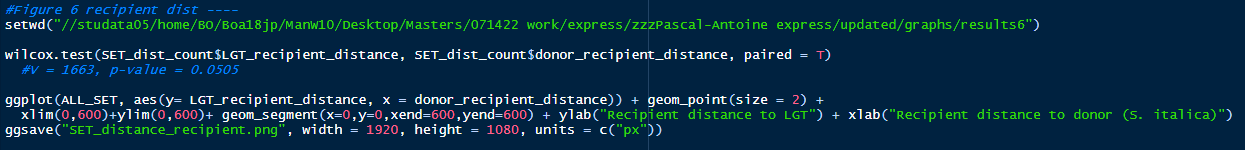
I decided to make data frames split for figure 5 and 6 as its much easier to sort.



The plots for figure 5 and 6 were very simple too. Remember, geom\_segment was used as a line of equidistance, as it’s nice knowing where things are equal on the graph.

I also performed some more Wilcoxon signed-rank tests (as data was paired) to test for significance. These are simply directly comparing the distances as below.





And that is the complete list of my code. Some of it isn’t perfectly organised as I corrected a lot of tests so I could use paired data, but if it’s run line by line all results used in the main paper can be found.

All my statistical tests are commented, so if you CTRL + F in the script the test statistic (E.g. v = 1663) mentioned in the results of the main paper, you’ll immediately find the line of code responsible for that statistical test.



**Side notes and other things I attempted which weren’t used in project:**

\*This section can be entirely ignored, none of it was used in the end. However I did successfully manage to create a script for Orthofinder which could’ve been useful.

Initial reference genomes for *S. italica* were not the same as the gene trees, so I corrected this. Gene trees used two reference genomes for *S. bicolor* so I also did this.

I didn’t create trees in this study, but this can be done using Orthofinder:

**Orthofinder guide:**

Download Orthofinder from github

Move to directory /RNAseq-alloteropsis/orthofinder\_analysis

Extract (tar xzf OrthoFinder\_glibc-2.15.tar.gz)

To run Orthofinder for nucleotide sequences use -d (default is amino acids) .

Script below (orthofinder.sh) runs this:

#$ -m bea

#$ -M jpartington-smith1@sheffield.ac.uk

#$ -pe openmp 4

#$ -l h\_rt=96:00:00

#$ -v OMP\_NUM\_THREADS=4

#$ -l mem=6G

#$ -l rmem=6G

#$ -j n

source /usr/local/extras/Genomics/.bashrc

./OrthoFinder/orthofinder -f /fastdata/boa18jp/masters/RNAseq-alloteropsis/orthofinder\_analysis -d -t 4

This script may take a while to run.

But what it does is run orthofinder and compares all the reference genomes in the relevant directory. Put all reference genomes needed into the directory and run it. Then it produces the relevant results in a folder based on the date script is run, for example “Results\_Mar24”. “The key aspect of orthofinder is that it sorts your genes into orthogroups. So, basically, you can then easily find the genes that descend from a single copy in the common ancestor of Panicoideae” - Dr. Pascal Antoine-Christin.

Description of Raimondeau’s trees (TableS4) - used to identify donors.

* This tree is from the supplementary material of the paper ‘High turnover of lateral gene transfers diversifies a grass pangenome’
  + In this paper, 168 LGT were detected across five reference genomes for the *Alloteropsis* genus
  + Additionally, 45 whole-genome resequenced accessions were used to establish the distribution of these LGT through time.
    - This revealed many recent LGT compared and a few ancient ones, and with decay models enabled predictions over the rate of loss of LGTs compared to gains.
* This turnover was suggested to have created important intraspecific structural variants, and that most LGT occur as accessory genes in the *Alloteropsis* pangenome.
* Furthermore ancient LGT which then underwent independent selection for 10s of millions of years added to the recipient genome novelties, and their rapid turnover created standing variation which provides a good context for further adaptation

How table S4 is laid out:

The top-left of the table contains information about the LGTs themselves (example in brackets):

* LGT name (LGT-073)
* Candidate type (primary or secondary LGT )
  + Primary candidates were identified using a stringent phylogenetic pipeline
  + Secondary candidates flank primary candidates in the *Alloteropsis* genomes and either:
    - Have gene trees supporting the same LGT scenario
    - Are shown in potential donor mapping data as being acquired as part of the same LGT fragment (as the primary candidate?).
* Reference gene.
  + The names of the reference genomes are different from my AUS, L04, KWT and ZAM. The reference genomes for THE (a sorghum reference) and SET are not in this dataset.
* Donor species name (name of the species which donated the LGT)
* Genomic block, or the location the gene is found on the relevant genome.
* Gene order, or sorting LGT by appearance on its genomic block.
  + E.g. Genomic block 3 has three LGTs given the numbers ‘1’, ‘2’ and ‘3’.
* Then five rows related to genes and five more rows to their fragments. I think this can be used to show which LGTs are found on which species, and by which fragment. Some LGTs are found in multiple of the five species.
* Then a list containing individual samples and data on if an LGT (sorted by columns) was found with a binary ‘1’ or ‘0’ value.

Additionally all relevant LGTs begin with ASEM\_xxx (e.g. ZAM) which might be used in grep.

Issues with using this:

* No data for Setaria or Sorghum reference genomes (actually this is fine, we are looking to use Cenchrinae and Andropogoneae instead).

How to compare reference genomes:

Install blastn from [Index of /blast/executables/blast+/LATEST (nih.gov)](https://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/)

Move to bash and install

gunzip -d ncbi-blast-2.10.1+-x64-linux.tar.gz

tar xvpf ncbi-blast-2.10.1+-x64-linux.tar

Run blastn on two ref genomes

ncbi-blast-2.13.0+/bin/blastn -query Sorghum\_bicolor.Sorghum\_bicolor\_v2.cds.all.fa -subject z\_not\_used\_Sobic\_ref/Sbicolor\_454\_v3.1.1.cds.fa -outfmt 6 -out results.txt

**Ape**

**My (unsuccessful) attempts at extracting LGT names from Table\_S4 and then filtering my results.xprs files to only include LGTs**

#this extracts parts of table we need (LGT names)

library(tidyverse)

library(xlsx)

LGTs <- as.matrix(read.xlsx("TableS4.xlsx",

sheetIndex = 1, #sheet no1 from file

rowIndex = 4, #extract row 4, containing LGT names

colIndex = 3:145, #extract columns C:FN, containing LGT names))

header = FALSE ))

## open express read

LGTs\_vector <- as.character(as.matrix(LGTs)) #convert table to vector?

L04\_root <- read.csv("L04\_root.csv")

#next goal, figure out how to mutate(?) L04 read to only show LGTs

L04\_root\_LGT <- L04\_root %>%

filter(target\_id %in% LGTs\_vector)

write.csv(L04\_root, "L04\_root\_LGT.csv")

#removes columns that don't match TableS4 LGTs

L04\_root\_no\_LGT <- L04\_root %>%

filter(!target\_id %in% LGTs\_vector)

#removes columns that match TableS4

#including this if we were to look at the 6657 genes without LGT in a table

#as I could just filter by those and the remaining rows should technically

#all be LGTs

#Ways to improve:

#Adding donor from TableS4

#Adding primary/secondary values from TableS4

#Can do this manually, but probably a way to do it automated.

Extracting gene trees attempt:

Need to extract gene ID from the gene trees Pascal-Antoine gave me.

For each tree I want, ID for the LGT of each of the five Alloteropsis genomes, the ID of the native genes from each of the five reference genomes, the ID of the Stearia ortholog, the ID from the Sorghum orthologs.

Can do this with ape:

library(ape) #load ape

tree <- read.tree(‘ASEM\_ZAM15-05-10\_63496.phy\_phyml\_tree.txt.newick’) #read tree

tree$tip.label #all gene names found here

tree$tip.label[grepl(‘Alloteropsis’,tree$tip.label)] #get the ID of Alloteropsis genes

tree$tip.label[grepl(‘Setaria-italica’,tree$tip.label)] #get ID of Setaria italica genes

Once I have the gene IDs I can then get the trees and label by color (examples below).

Final file should look like

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Gene | M\_LGT | KWT\_LGT | L04\_LGT | ZAM\_LGT | AUS\_LGT | M\_native | K\_native | L\_native |
| Z\_native | AUS\_native | Setaria | Sorghum |  |  |  |  |  |

Potential issue - some cases maybe Sorghum gene that isn’t orthologues to the LGT, PAC says probably not but I should check the tree s too.

Don’t re-root tree, not sure why google docs kills the image quality of tree but it does. Leaving it unrooted is best as we know Poodieae-Oryzae are the outgroup.

Other genes of Alloteropsis that aren’t LGTs are native orthologues. Donor orthologues are genes from donor that are expected based on the species tree - For Setaria in this tree these are the two gene just above the LGT (use *Setaria italica*)

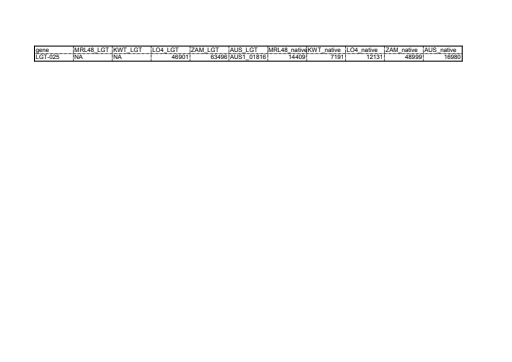
Anyway, extract expression values corresponding to gene names once we got the names - can do on R quite easily.

Setaria output is KQK99508 etc… but check if this links up to the SET99508 or something if it is on the TableS4 files. C4 and Aus1 seem to link pretty well. E.g. ASEM\_C4\_17509 and ASEM\_CR\_17509-RA are the same… I think. Gene name is 17509.

It didn’t so I changed the reference genome (see final paper).

We have to extract name of the Setaria and Sorghum genes from Newick trees, not table S4.

We should also merge all the .xprs trees one gene family per line. NOT 168 lines, but instead do like this



Gene - LGTs - natives

LGT-025 - expression levels

Keep going through the list ! This is for gene tree I posted above, will double check if my results are the same as Pascal-Antoine’s and proceed to do the rest.

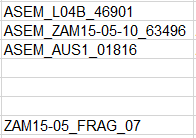
To do this I need to understand the TableS4 LGT dataset

Info for LGT-025 from TableS4:

It’s a primary LGT with the name ASEM\_ZAM15-05-10\_63496 and has a Cenchrinae donor.

* Can see on the table Pascal-Antoine Christin made this links up identically with his results.
* ZAM\_63496 LGT matches
* L04B\_46901 matches
* AUS1\_01816 matches
* There are no KWT LGT
* There are no MRL LGT (but we don’t need this anyway)

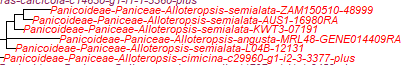
Also shows fragment but not sure significance of that



Find LGTs in trees as they are named LGT.



But how do we get the native genes? Check tree for other Alloteropsis genes not named as ‘LGT’



* ZAM\_150510-48999 - yes, but why is it 48999
* AUS1-16980 - yes
* KWT-07191 match (but PAC removed the 0, probably google sheets did auto)
* MRL-GENE014409RA - yes
* L04B-12131 - yes
* Setaria - Si028712 #closest to group
* Sorghum - EER97252 #WAS EER07252 but presume typo due to the awful image quality of the tree i sent

But now we have the dataset for one gene tree (i think) we just need to do this for all gene trees… but how to do this automatically?

These are the lines of code you need to extract the data.

setlabelSet <- tree$tip.label[grepl('Set',tree$tip.label)]

setlabelSet

setlabelLGT <- tree$tip.label[grepl('LGT',tree$tip.label)]

setlabelLGT

setlabelSor <- tree$tip.label[grepl('Sor', tree$tip.label)]

setlabelSor

setlabelAllo <- tree$tip.label[grepl('Alloteropsis', tree$tip.label)]

setlabelAllo #note we can probably just ignore the setlabelLGT as this includes the LGTs… I think all LGTs are in allo but still

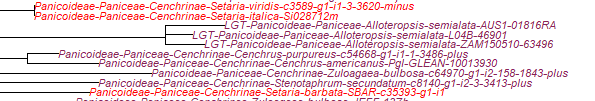
Things need to know -> should we record primary and secondary trees? Yes

We don’t need angusta

We don’t have express data for… Sorghum? Can’t find the EER97252



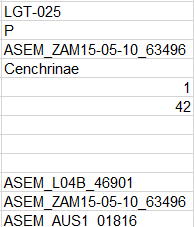
Si028712m for Setaria… now does the express data exist for this?



KQL28712 does exist

…problem of sorghum BUT we can probably… maybe start merging the files

Why do we know it is LGT-025?

 Because tableS4 tells me it is.

How to merge the files?

Strategy 1 -> Do trees one by one…

Strategy 2 -> Script to do this automatically (perl script to do this was made for me by PAC).