**README**

Thank you for checking out the **foxy\_qtl\_pipeline**. It was written by Max Feldman and Rachel Paul with conceptual support from Ivan Baxter and Andrew Leakey.

This is a series of computer scripts that performs linkage-mapping analysis of biparental genetic mapping populations to identify quantitative trait loci (QTL). Essentially, it is a series of **R/qtl** [1] scripts called by a python program generally called on a Linux based computer server.

It can also be run on a MacBook or other Linux desktop but the analysis of many traits in parallel (server mode) is not recommended.  
  
This pipeline can very easily be adapted to perform similar analysis on RIL populations in other organisms.

Currently it automates several types of analysis

1) Perform a genome scan with a single QTL model (significance: p-value < 0.05, based upon 1000 permutations) using Haley-Knott regression. See description of the R/qtl scanone function.

2) A two-dimensional genome scan using a 2-QTL model (significance: p-value < 0.05, based upon 100 permutations). See description of the R/qtl scantwo function.

3) A stepwise model selection (significance based upon penalized LOD score, estimated in either the first 2 analysis methods). This automatically seeds that model selection procedure with significant QTL from the results from the single genome scan (scanone). This analysis selects the best model based upon penalized LOD score. The penalties are derived from the results of scantwo analysis. Currently, the pipeline will assess models with up to 25 cofactors and considers pairs of loci (scan.pairs =T) during the analysis. See the description of the R/qtl stepwiseqtl function.

4) Single QTL model genome scan and stepwise model analysis of function valued traits with up to 9 cofactors (temporal or spatial) [2]. See the vignette for the ‘funqtl’ package.

Dependencies

This software pipeline uses several python and R libraries that need to be installed.

Python libraries:

csv

glob

multiprocessing

numpy

optparse

os.path

re

subprocess

sys

R libraries:

funqtl

ggplot2

lattice

lme4

qtl

Assumptions

It is assumed that the phenotypic trait data input is appropriately blocked and normally distributed. The user should check the normality assumption using a Shapiro-Wilks test before data is input. If data is not at least approximately normal Box-Cox transformation is recommended. If individuals within treatment blocks are replicated, the program will perform partitioning of variance and submit the mean of trait values for each genotype values nested within treatment.   
  
Users are recommended to perform BLUP/BLUE modeling or other types of regression/smoothing before analysis using this QTL pipeline.

How to launch an analysis

1) Download all relevant files from GitHub:

<https://github.com/maxjfeldman>

You’ll want to run this program on a large computer server. Our current implementation requires 1-3 GB of RAM per trait at time of analysis. This can be an issue if you want to analyze many traits in parallel on a small computer.

The efficiency could be improved rather easily but I’ve never needed to analyze more than 50-100 traits in parallel and the computing infrastructure at my current institution supports this.

2) Format your input data.

Current format this is a .csv file that contains a few fields that aren't necessary for the analysis. Please see file named: “qtl\_data\_format\_example.csv”.

\*\* An important detail is that trait names cannot contain spaces or periods. Periods are currently used as a field delimiter to distinguish between the same trait measured in different treatments.

Column headers:

Obs – This is just an number entry number sometimes recorded by an instrument but can be as simple as row number

experiment – This is usually an identifier that specifies a 2 character description of experiment and the year.

year – What year is the data from?

treatment – Is there a contrast being performed (wet v. dry | dense v. sparse)? If no treatment just specify the same string for each entry (“none” for example).

plot – Setaria grow outs can be summarized by overall plot. If you experiment does not contain multiple plot you can just specify the plot using the same string for each entry (“none” for example)

subplot\_id – This is a string that associates the individual plant location within the plot (can also specify “none”).

id – This is the name of the RIL. The nomenclature you use must match the names in the genetic map. We generally use the format “RIL\_001”.

sampling – This is a subcategory that can be used to distinguish between identical plants sampled at different time points (usually, this is imputed as “none”).

The remaining columns contain phenotypic values (height\_25, height\_46, height\_67). Notice that these trait names do NOT contain periods.

3) Call the program.

Change into the directory where you have downloaded the script and use the following function call:

[user@computer~]$ python foxy\_qtl\_pipeline.py –i input\_file.csv -o name\_of\_output\_directory -c [y|n] -m name\_of\_genetic\_map -s [y|n] -t riself

There are several arguments that need to be specified:

i – This is the name of the input file (see qtl\_data\_format\_example.csv)

o – This the name of the directory where you will store the results

c – Are you doing a comparison between treatments (wet v. dry) or not? “y” indicates that the treatment field in the input file contains 2 different levels

m – File name of your genetic map formatted for R/qtl csvs input.

s – Are you running this analysis on a server? If you select “y” traits will be analyzed in parallel. If you select “n” they will be run consecutively.

t – Type of mapping population. Currently I am working with a F7 RIL mapping population so I use ‘riself’.

d – distribution/model type (is this a normal, binary, or 2-part or non-parametric) Note that choosing binary distribution will eliminate the possibility to do a comparison analysis

q – Method to use for QTL analysis (currently all is done using Haley-Knott regression)

Other arguments that will be added in the future:

f – Is this a functional trait (time-series)? [y|n] \*\* Currently we have a separate set of scripts to do this type of analysis. See below.

Example:

As an example we will perform an analysis of plant height data from the Setaria A10 X B100 recombinant inbred line population. This population was grown at two planting densities (dense = 5 cm planting density, sparse = 25 cm planting density) in the field at the University of Illinois. Plant height was measured using handheld barcode scanners at 3 time points representative of the plant life cycle (See Feldman et al., 2016. in prep).

First, examine the format/structure of the data by opening the file named ‘qtl\_data\_format\_example.csv’. You will see that there are three traits measured ‘height\_25’, ‘height\_46’ and ‘height\_67’. Do you understand what the columns are specifying? Do you see any period characters (‘.’) within the trait names?

If you examine the entries in the treatment column how many unique entries do you see?

Because there is more than 1 (‘dense’ and ‘sparse’) a comparison can be performed.

Also included are two genetic map files named “GBS\_map\_A10xB100\_v0.96.csv” and “2013setariamapJGI.csv”. The first is a genetic map that contains some genotype calls for all or most of the individuals within the RIL population as described in (Feldman et al., 2016. in prep.) whereas the latter is directly from Mauro-Herrera et al., 2013.

Lets run the analysis.

Open your terminal window and move into the foxy\_qtl\_pipeline directory. If you list the contents of this directory you should see the following files.

Type the following command (Don’t copy and paste):

[user@computer~]$ python foxy\_qtl\_pipeline.py –i qtl\_data\_format\_example.csv –o example\_output -c y -m GBS\_map\_A10xB100\_v0.96.csv -s y -t riself

This command tells the computer to launch the python program named: foxy\_qtl\_pipeline.py  
  
You’ve input the file named ‘qtl\_data\_format\_example.csv’  
The results will be in a directory named ‘example\_output’

You have indicated you’d like to do a comparison [-c y] (treatment column contains 2 factors: ‘dense’ & ‘sparse’)

You have told the program to use a genetic map named ‘GBS\_mapA10xB100\_v0.96.csv

By setting the -s y (server) flag yes you have indicated you are running this job on a computer cluster and it will run all the traits in parallel

You have told the program you are analyzing an advanced RIL population by setting the –t argument to ‘riself’

The pipeline typically takes ~2-3 hours to perform the analysis on a single trait. So the ability to parallelize this analysis using a computer server is critical to the analysis of large trait sets.

Results

The standard output the foxy\_qtl\_pipeline is a structured hierarchy of directories. The hierarchy will change slightly if the user indicates they are doing a comparison between treatments (wet v. dry | dense v. sparse) or a trait from a single environment. In this case (comparison flag is ‘y’; -c y), the mathematical difference, the relative difference and ratio between each trait is calculated and run as an additional trait set.

The top level directory is the analysis name given as the –o (output) flag when calling the program (ex/ ‘**example\_output’**). Within this directory are some files that contain summary statistics:

* heritability\_table.csv [table of variance partitioning between factors done using lme4]
* treatment\_heritability\_table.csv [table within treatment, of only proportion of variance associated with genotype]
* trait\_correlation\_table.csv [table of pearson’s correlation coefficient calculated between traits]

There are also a series of other files that are used for downstream processing.

R workspace files:

* cross.object.raw.Rdata [data from building R/qtl cross.object]
* cross.object.diff.Rdata [only if comparison flag is indicated]
* format\_and\_EDA.Rdata [initial formatting and summary statistic calculation]

Text files:

* qtl.phenotypes.raw.csv [input for qtl analysis of traits on their own]
* qtl.phenotypes.diff.csv [input for qtl analysis of difference traits]
* diff\_phenotype\_by\_treatment.csv [intermediate output]
* trait.phe.csv [intermediate output]
* phe.r.txt [listing of traits to analyze]
* phe.d.txt [difference version]

You will also see a set of sub directories that contain the results of individual traits (example: height\_46\_DN14). Note: A 4 character descriptor, in this case DN14, is added to the end of each trait name to make it easier to merge many traits grown in different conditions + years for meta analysis.

Lets examine ‘~/**example\_output /height\_46\_DN14’** as an example. Because a comparison type analysis was performed you see three sub-directories, one for each treatment, and one for comparison between treatments.

Look into the ‘~/**example\_output/height\_46\_DN14/dense**’ directory:

This folder contains two subdirectories, one for each analysis type:

* scanone.out – results from marker regression or interval mapping depending upon flavor selected. Also contains results from scantwo analysis (marker x single marker regression
* mqm.out – results from a multiple QTL model built using the R/qtl stepwiseqtl() fxn.

It also contains a histogram of the trait distribution values (‘histogram \_of\_trait.pdf), a list of RIL phenotype values ordered by magnitude (‘ordered\_phe\_vals\_trait.csv’) and an R object that contains all the results from both scanone and mqm analysis (‘cross.obj\_trait\_raw.Rdata’)

Look inside the ‘~/**example\_output /height\_46\_DN14/dense/scanone.out’** directory:

Within this directory there are many files that are partially redundant with the contents of the ‘summary.table.so.trait.csv’ file, but this is a table that consolidates most of that information. Lets look at it:

The first column is a list of marker(s) found significantly associated with the trait.

The columns in order are: Chromosome, location on genetic map (cM), LOD score, p-value), proportion of variance explained by the marker, additive allelic effect on the average phenotype, the standard error associated with effect size, L.CI\_maker is the marker associated with the lower position of the LOD 1.5 confidence interval, whereas R.CI\_marker is the marker associated with the upper base pair limit.

Additional, LOD score plots of scanone and scantwo out put are included as .pdf documents (scan[one|two].qtls.trait.pdf} as are plots of the effect size across each position in the genome (so.effect\_size.trait.pdf), significant interaction between the most significant markers (QTL.interactionplot.trait.pdf), and plot of how the distribution of the phenotype looks at each genotype (so.phenotypeXgenotype.trait.pdf).

Look inside the ‘~/**example\_output /height\_46\_DN14/dense/mqm.out’** directory

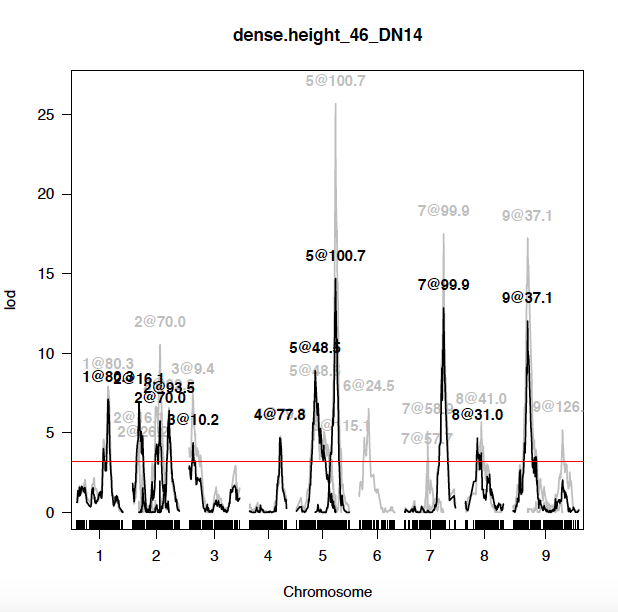
The format of results in this directory are quite similar to the results in the scanone.out directory. Once again, the user is probably most interested in the ‘summary.table.mqm.trait.csv’ file, and visualizing the LOD score plots in the ‘mqm.LOD.profile.trait.pdf’ file.

Also included is a file named ‘genes\_in\_qtl\_CI\_trait.csv’. This is a list of the all the genes, and their rice and Arabidopsis annotation found within the 1.5 LOD QTL confidence interval.   
  
This file is sorted so SNPs closest to the top have the shortest absolute distance to the QTL SNP.

\*\* Here is an important note:

When we perform the stepwiseqtl analysis we do the analysis with 2 different significance penalties, a stringent penalty of p-value < 0.05 (that is used to determine significant results) and a permissive penalty of p < 0.25 for model selection. This way we can determine if there are any potential QTL that may be just below our threshold of detection.

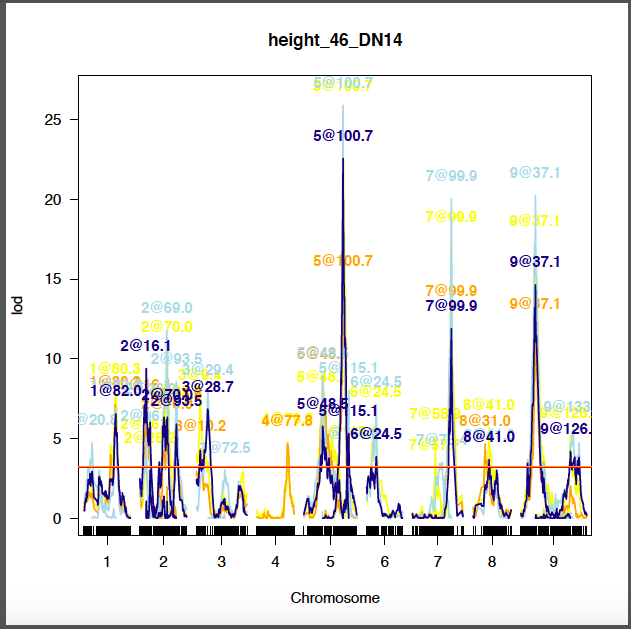
This is visualized on the ‘mqm.LOD.profile.trait.pdf’ plot. The line printed in black is associated with QTLs that are called significant, the grey line is the analysis performed using the lighter penalty. The red line plotted is of no consequence in the analysis but illustrates the scanone LOD score significance threshold based upon 1000 permutations.



Finally, examine the results in the ‘~/**example\_output/height\_46\_DN14/comparison’** directory.

The results here are formatted very similar to the results of the other directories, with some small differences. Boxplots and histograms are compared between treatments here in the files (boxplot.trait.pdf and histogram\_of\_trait.pdf), and the difference between treatments is reflected as a p-value. There is also a histogram of the difference between treatments (condition1 – condition2) plotted in the ‘histogram of trait[dense.sparse].pdf’ file.

In the ‘mqm.qtl.trait.pdf’ file you will find LOD score plots of the QTL of each the traits from each treatment plotted on top of each other. As a rule, the darker lines (navy blue and orange) are indicative of the significant QTL whereas the lighter lines (light blue and yellow) are indicative of QTL detected with a more permissive penalty structure. The orange/yellow coloring corresponds to the trait that comes first in alpha numeric order while navy/light blue correspond to the later (dry = orange; wet = blue).



Results within the ‘‘~/**example\_output/height\_46\_DN14/comparison/scanone.out’** and the ‘‘~/**example\_output/height\_46\_DN14/comparison/mqm.out’** are structured like their single trait counterparts but perform the same analysis on the numerical difference between the trait in each treatment.

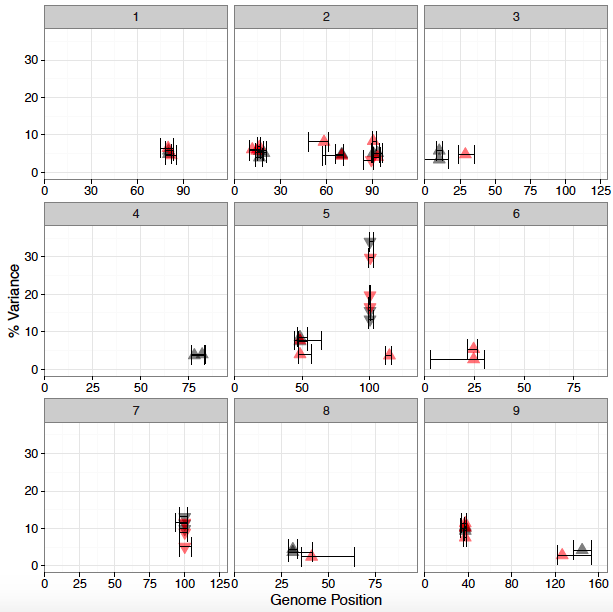
Aggregating QTL results and analyzing function valued trait analysis

We have written an add-on function to aggregate the summary tables from individual traits/timepoints after run of the foxy\_qtl\_pipeline. This program also makes a plot all QTL identified in the experiment colored by treatment and launches our function valued trait QTL pipeline

The function to perform this analysis is named ‘plot\_common\_qtl.py’, it asks for three arguments, base directory (in this case it is **‘~/example\_output**’) comparison (c -y) and time series (t -y). Here is an example function call:  
  
user@computer~]$ python plot\_common\_qtl.py -i example\_output -c y –t y

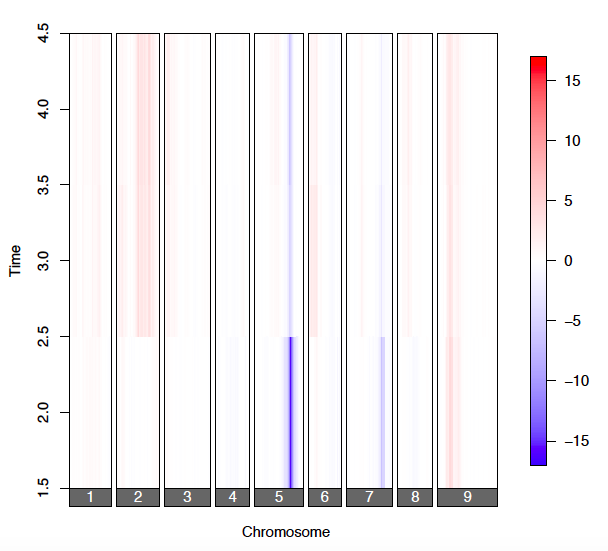
This will produce a file named ‘example\_output\_concatenated\_summary\_table.csv’. This table is identical to the summary tables for the individual traits but contains additional columns. The tables contain columns that provide the name of the trait (height\_46\_DN14), treatment (dense), experiment (DN14), year (2014), and type. Type = raw means that this QTL was detected in a standard trait. Type = comp means that this QTL was detected by using the numerical difference, the relative difference and the ratio between treatments as a trait (can be comp\_diff, comp\_rel\_diff, or comp\_ratio).

This script generates the famous banan banners first conceptualized and produced by Darshi Banan. It plots the location of each ‘raw’ type QTL along the x-axis faceting by chromosome. The y-axis is the % of variance that is explained by the locus. The color of plotting characters corresponds to their treatment (red = dense, black = sparse) and the direction indicates the directional effect of the B100 allele. The bars correspond to the 1.5 LOD confidence interval.



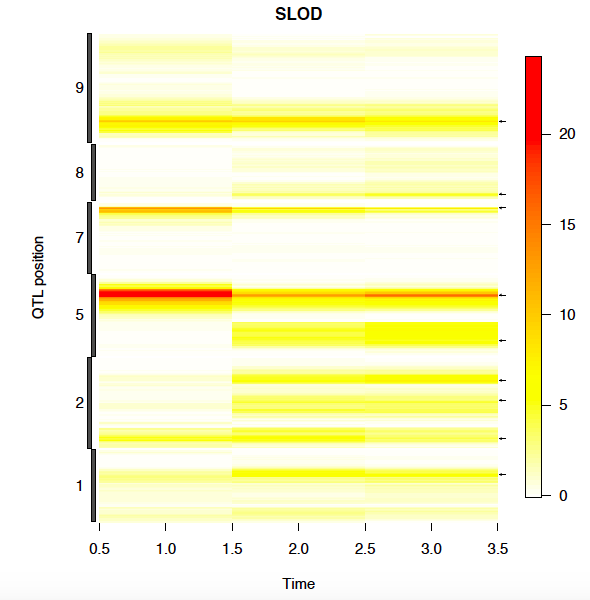
If the time series flag is set to ‘y’ (-t y) the function valued QTL pipeline will be performed. The output will be written into two directories named ‘timeseries.treatment’ (timeseries.dense).

This performs a single QTL scan at each time point using the scanoneF() function. The results of which are written to a file named: ‘qtl.table.so.F\_output\_example.dense’. This file lists the QTLs identified as rows and the significance based upon two different types of analysis, slod and mlod.

SLOD corresponds to significant (p-value < 0.05; 1000 permutations) QTL based upon the average LOD score across all time points. MLOD corresponds to significant QTL based upon the maximum LOD score across all time points. The program also produces plots of these QTL through time (ex/ ‘so\_timeseries\_qtl\_output\_example.dense.pdf’). Red corresponds to a positively signed LOD score and blue corresponds to a negatively signed LOD score based upon the effect of the B100 allele.   


The program then performs stepwiseqtl multiple QTL model selection using the stepwiseqtlF function. Penalties for an additive QTL model are based upon the the output from the scanoneF permutations performed above. The maximum model size is limited to 9 QTL.

This analysis is performed within treatments and is performed based upon the SLOD, MLOD and upon a model selected upon all major QTL identified across the entire experimental treatment (KLOD). The QTLs identified and their effect size at different time points are found in tables named ‘type\_treatment\_basedirectory.csv’ (ex/ slodeff\_dense\_example\_output.csv’). The results are plotted in a pdf document named (ex/ mqm\_timeseries\_qtl\_example\_output.dense.slod.pdf). A plot the the effect size of these QTL is in a pdf named (ex/mqm\_timeseries\_fx\_size\_slod\_example\_output.dense.pdf).



QTL analysis with prior marker sets

Also available is a QTL pipeline that does this sort of analysis give QTL that you may have prior knowledge about. Launching these jobs and the structure of the results is almost identical to our standard pathway.   
  
The pipeline names are appended with the text string ‘\_known\_markers’. It also depends upon the existence of a file named ‘markers\_for\_fixed\_model.csv’. See the file included for an example of how to format this table.  
  
If anything is unclear please don’t hesitate to send me an e-mail.

Max Feldman

mfeldman@danforthcenter.org

Thank you for checking out our foxy\_qtl\_pipeline! =)

References:   
1] Broman KW, Wu H, Sen Ś, Churchill GA (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

2] Kwak et al. (2014) A simple regression-based method to map quantitative trait loci underlying function-valued phenotypes. Genetics 197: 1409-1416