Complete QTL and Candidate Search Tutorial

John T. Lovell

2017-08-11

Contents

---

email: johntlovell@gmail.com – website: lovelleeb.weebly.com – github: github.com/jtlovell/qtlTools

---

1 Setup

For this tutorial, we need a couple datasets and R packages.

1. The `R/qtl` R pacakge.

- `install.packages("qtl")`
- `library(qtl)`

2. The `qtlTools` R package, in development.

- If you don't have the devtools package, run:
  - `install.packages("devtools")`
- `devtools::install_github("jtlovell/qtlTools")`
- `library(qtlTools)`

3. The `ggplot2` R package

- `install.pacakges("ggplot2")`
- `library(ggplot2)`

4. The tutorial dataset which contains an R image with the following files:

- A `cross` object, which stores the genetic map, genotype and phenotype data needed to do QTL mapping
- A `gff` file with the physical positions of all genes
- A `vcf_summary` file with the counts of various types of mutations in each gene
- `parentExp` and `parentInfo` files with gene expression and experimental design data for the parental genotypes
- A `f2Exp` file with gene expression for the F2 population
- A `map` file that contains the physical AND mapping positions of all markers.

To get the tutorial dataset directly from R, run the following:

```
download.file(paste0("https://github.com/jtlovell/Rqtl_tutorials/blob/master/", # website
                     "completeQTL_tutorial_data.rda", # data file name
                     "?raw=true"), # use the 'raw' data
              destfile = "~/Desktop/completeQTL_tutorial_data.rda",
              method="wget", quiet = TRUE)
load("~/Desktop/completeQTL_tutorial_data.rda")
```

Here are the files names:

```
ls()
```

```
## [1] "cr"          "f2Exp"       "gff"         "map"         "parentExp"
## [6] "parentInfo"  "vcf_summary"
```

The R/qtl cross object summary (cr):

```
summary(cr)
```

```
##     F2 intercross
##
##     No. individuals:    244
##
##     No. phenotypes:     3
##     Percent phenotyped: 100 100 100
##
##     No. chromosomes:    9
##         Autosomes:      1 2 3 4 5 6 7 8 9
##
##     Total markers:      884
##     No. markers:        105 127 121 75 119 95 65 49 128
##     Percent genotyped:  99.8
##     Genotypes (%):      AA:22.2  AB:50.5  BB:27.3  not BB:0.0  not AA:0.0
```

The genetic and physical positions of markers (map):

```
knitr::kable(head(map))
```

	marker.name	chr	pos	bp
2	Pahal.A00074	1	0.0000000	668055
1	Pahal.A00017	1	0.6089281	1096202
3	Pahal.A00093	1	1.2178410	1259952
4	Pahal.A00138	1	1.8267386	1478629
5	Pahal.A00205	1	2.6393248	1864895
6	Pahal.A00225	1	3.2482224	1999342

The genome feature file annotation (gff):

```
knitr::kable(head(gff))
```

	seqname	source	feature	start	end	score	strand	frame	attribute
1	Chr01	JGI	gene	1173370	1175655	.	+	.	Name=Pahal.A00001
5	Chr01	JGI	gene	1170853	1172427	.	+	.	Name=Pahal.A00002
11	Chr01	JGI	gene	1170257	1172027	.	-	.	Name=Pahal.A00003
18	Chr01	JGI	gene	1149751	1150794	.	-	.	Name=Pahal.A00004
22	Chr01	JGI	gene	1148106	1149292	.	-	.	Name=Pahal.A00005
32	Chr01	JGI	gene	1136662	1142260	.	+	.	Name=Pahal.A00006

The parental expression (parentExp) and experimental design (info) data:

```
knitr::kable(parentExp[1:5,1:5])
```

	F1_H040_457	F1_H005_732	F1_H140_842	F1_H148_386	F1_H151_243
Pahal.C00791	0	0	7	0	0
Pahal.C00795	2	27	19	2	15
Pahal.C00796	7	27	28	1	8
Pahal.C00799	136	163	163	134	196
Pahal.C00803	14	19	23	29	37

```
knitr::kable(head(parentInfo))
```

	Treatment	type	libname
F1_H040_457	Dry	F1	F1_H040_457
F1_H005_732	Dry	F1	F1_H005_732
F1_H140_842	Dry	F1	F1_H140_842
F1_H148_386	Dry	F1	F1_H148_386
F1_H151_243	Dry	F1	F1_H151_243
F1_G969_327	Dry	F1	F1_G969_327

It's important to note that these datasets match ...

```
identical(rownames(parentInfo), colnames(parentExp))
```

```
## [1] TRUE
```

The gene-by-gene summary of non-synonymous DNA variants (vcf_summary):

```
knitr::kable(head(vcf_summary))
```

geneID	HIGH	LOW	MODERATE
Pahal.C00791	0	30	44
Pahal.C00792	0	11	9
Pahal.C00793	0	17	11
Pahal.C00794	1	12	9
Pahal.C00795	0	10	40
Pahal.C00796	0	12	12

The expression dataset for the mapping population (`f2Exp`):

```
knitr::kable(f2Exp[1:5,1:5])
```

Pahal.C00795	Pahal.C00796	Pahal.C00804	Pahal.C00807	Pahal.C00809
2.8157860	3.0563745	0.7344464	2.885006	0.5368464
0.7063020	0.8478562	-1.1387233	3.010271	1.4010099
1.3796110	2.3551993	0.6033860	2.703474	-0.5541553
2.1359740	2.0772516	1.1644626	2.349532	1.1804927
0.5689717	1.0819229	0.9387283	3.107259	1.6401327

It's important the the line IDs in the cross object match the row names of the f2Exp dataset

```
identical(rownames(f2Exp), as.character(getid(cr)))
```

```
## [1] TRUE
```

2 Overview - developing expectations for QTL interval sizes and number of candidate genes.

One goal of QTL mapping is to determine the genes that have the potential to affect phenotypic trait variation . . . candidate genes. While highly accurate, the precision of linkage mapping in experimental populations is limitted by the scale of recombination - QTL often span 100's of kb's and contain many gene models.

To illustrate this, let's model the size of confidence intervals for a single large effect QTL in a 200 individual F2 population.

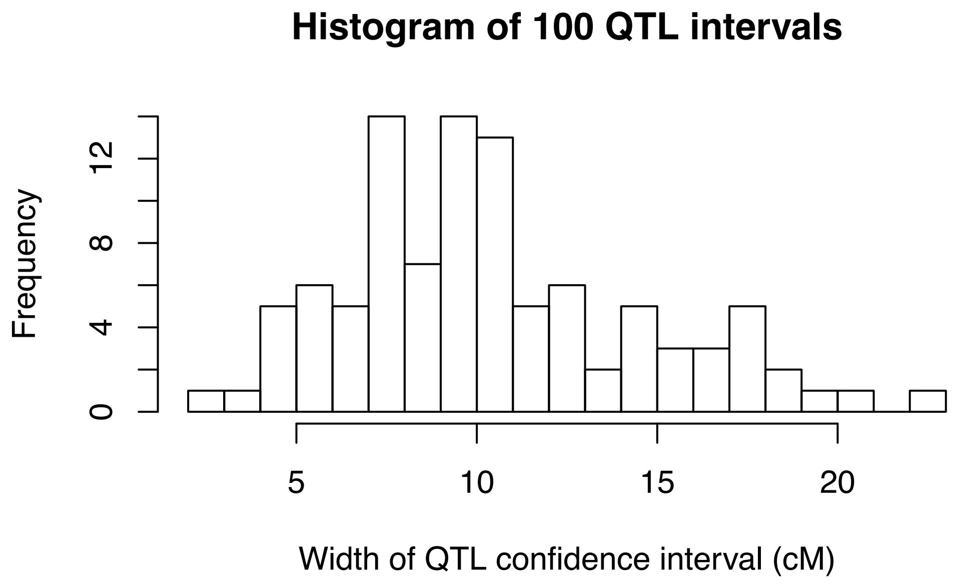


Figure 1: Distribution of QTL confidence intervals (calculated as the 1-LOD drop interval) from 100 simulated linkage maps. For each map, a single QTL was simulated with a LOD score of approximately 25 (Additive effect = 1, dominance effect = 0)

Note that most QTL's are 5-20cM wide. A typical *A. thaliana* genetic map is 500cM and 212Mb . . . ~400Kb / cM. If the ~30k genes are distributed evenly, we might expect 300-1200 genes under any given QTL with a LOD of 25. As such, there is considerable work remaining after finding a QTL peak.

Note *Higher LOD QTL, more recombination (e.g. RIL/MAGIC populations) or more individuals will reduce the width of QTL intervals. If there is little additional information in a mapping population, it is best to get information from as many recombinant genotypes as possible (e.g. subsequent fine mapping or very large populations).*

3 The phenotype data

Here we will look at an F2 population of Panicum hallii - a genetic model for C4 grasses. Importantly, there are two environmental treatments (Dry and Wet)… these are covariates, which are stored as `covar` and included in all analyses.

```r
covar<-data.frame(
  covar = as.numeric( # turn treatment classification into 1/2's
    as.factor( # turn character "wet/dry" into a factor
      pull.pheno(cr, pheno.col = "Treatment")))) # pull the phenotype data from cr
```

```r
ggplot(pull.pheno(cr),
       aes(x = phenotype, fill = Treatment))+
  scale_fill_manual(values = c("darkred","cornflowerblue"))+
  geom_histogram(binwidth = .15)+
  theme_jtlbar()
```

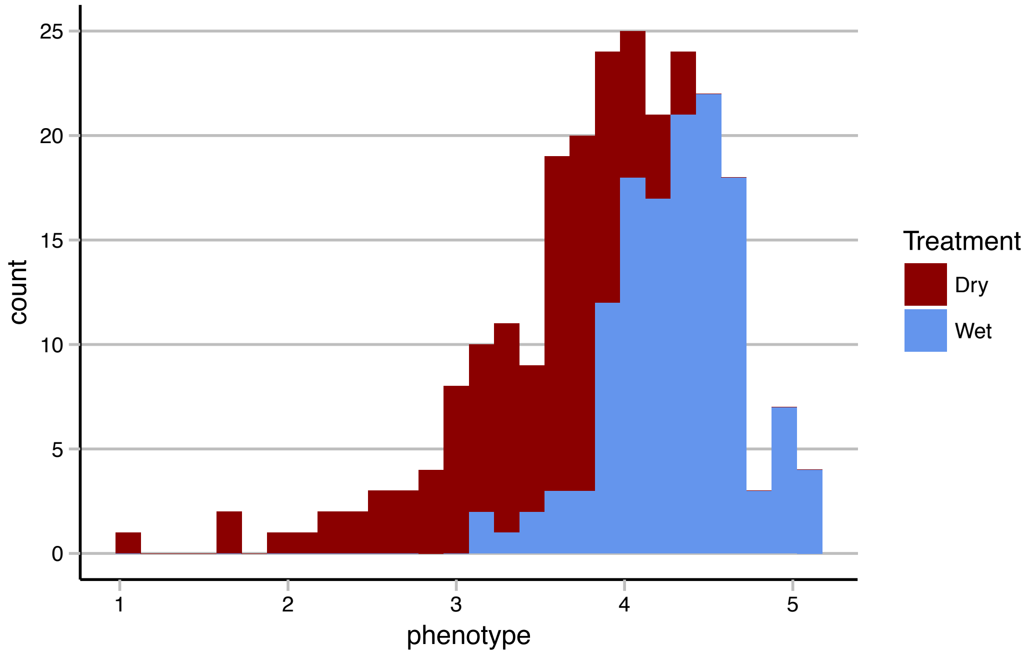


Figure 2: Distribution of phenotypic variation. Data is skewed, but this isn't really a problem if we are looking at a single QTL … We use permutations to test significance, which account for non-normal distributions.

4 Run a 1-way QTL scan

This is the first step in QTL mapping … for each marker / pseudo marker in the genome, we test the hypothesis H0: no QTL vs. HA: 1 QTL. Since here we also have the added complexity of an experimental covariate, our explicit likelihood ratio test is: H0: y ~ 1 … HA: y ~ QTL + Treatment + QTL*Treatment

```r
s1<-scanone(cr,
            method = "hk", # use the haley-knott regression
            pheno.col = "phenotype",
            intcovar = covar, # force treatment to interact with the QTL
            addcovar = covar) # for there to be an additive effect of treatment
```

To test for significance, we must run permutations...

```
s1p<-scanone(cr,
             n.perm = 100,
             method = "hk", # use the haley-knott regression
             pheno.col = "phenotype",
             intcovar = covar, # force treatment to interact with the QTL
             addcovar = covar, # for there to be an additive effect of treatment
             verbose = F)
```

```
plot(s1p)
abline(v = quantile(s1p,.95), col = "red", lty=2)
```

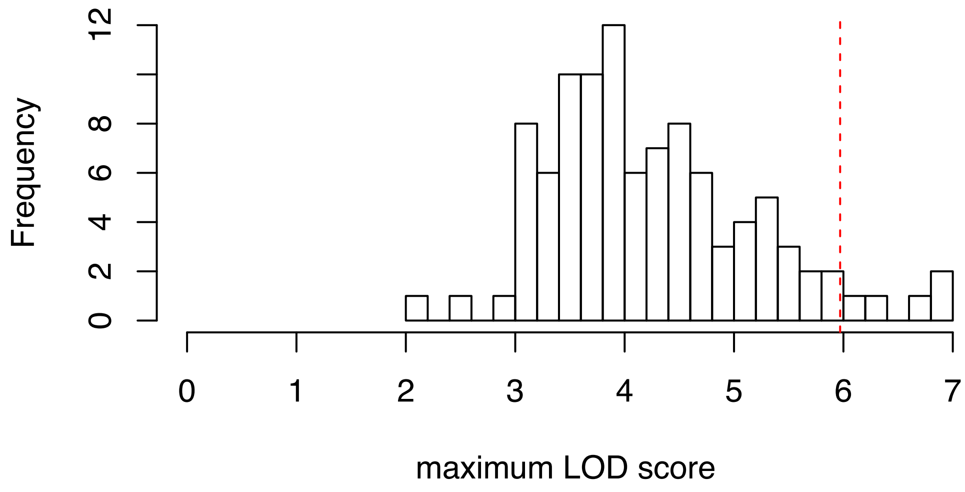


Figure 3: The distribution of maximum LOD scores from scanone permutations (100). The 90th%ile of the permutations is presented by the vertical

To see if we have a QTL, we plot the one-way scan results along with the permutation threshold

```
plot(s1, ylab = "LOD")
add.threshold(out = s1, perms = s1p, col = "red", lty=2)
```

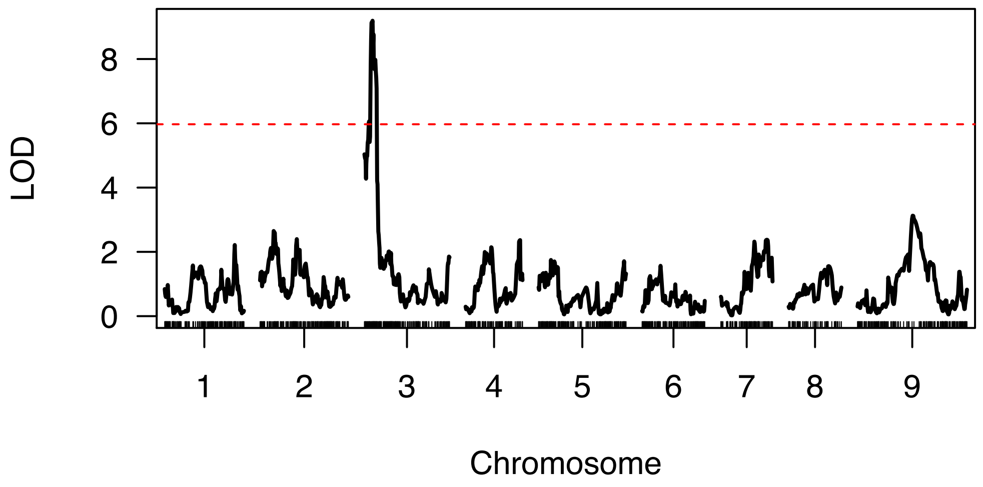


Figure 4: One-way QTL scan for loci associated with phenotypic variation.

5 Finding a QTL peak and confidence interval

We now want to determine the width of the QTL interval. To do this, we calculate the LOD profile. In this case, since we only have one QTL, the LOD profile will be identical to the one-way QTL scan.

```
# make a QTL model
q<-makeqtl(cr,
           chr = max(s1)$chr,
           pos = max(s1)$pos,
           what = "prob")
# generate a LOD profile
r<-refineqtl(cross = cr,
             qtl = q, # the qtl model
             formula = y~Q1+covar+Q1*covar, # the formula, w/ covariate interactions
             method = "hk", # haley-knott regression
             covar = covar, # covariate dataset
             pheno.col = "phenotype",
             verbos = F)
```

Using the LOD profile, determine the QTL confidence interval as the 90% approximate Bayesian credible interval.

```
cis<-calcCis(cross = cr,
             mod = r, # the qtl model with calculated LOD profile
             pheno.col = "phenotype",
             lodint = F, # calculate bayes confidence intervals
             prob = .9 # use 90% probability
             )
```

Finally, plot the LOD profile and add the confidence interval around the QTL peak.

```
plotLodProfile(r,
               xlab = "Chr03 mapping position",
               ylab = "LOD")
segmentsOnPeaks(cr,
                calcCisOutput = cis,
                chr = 3,
                mod = r,
                int.y = 8,
                col = rgb(1,0,0,.5),
                lwd = 3, pt.cex = 1)
```

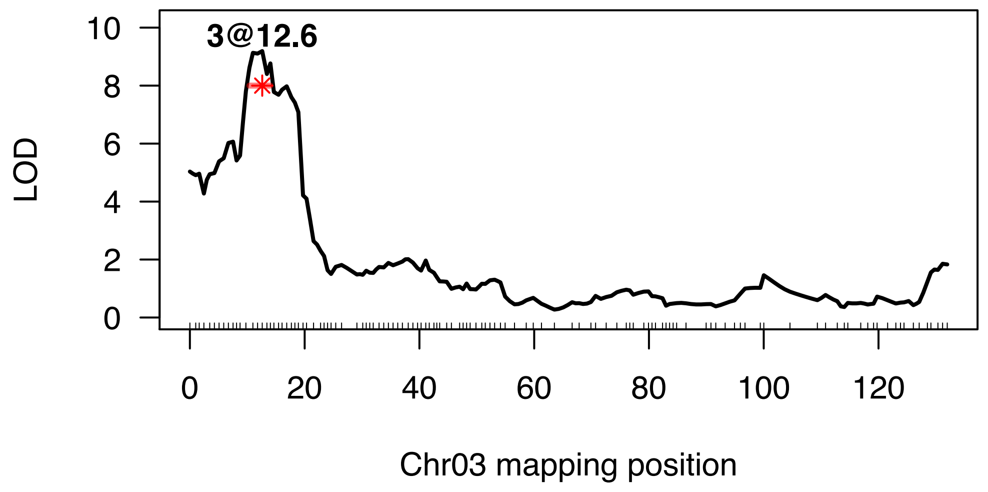


Figure 5: Chromosome 3 LOD profile including the position of the QTL peak and confidence interval (the red point and line segment).

6 Finding genes under a QTL

The first step towards finding candidate genes is finding out which gene models lie within the interval. To do this, we need two datasets:

- An annotation (e.g. gff3) file with the location of all genes
- The mapping and physical position of markers

Using these datasets, we can determine the mapping postion of all genes

This can be calculated using any number of approaches. Here we employ a marker-by-marker linear model. In short, we pull two adjacent markers and calculate the slope and intercept of the relationship between physical (bp) and genetic mapping (cM) positions. We then pull all genes that have physical positions between those markers and predict their mapping from physical positions.

```
geneCM<-findGenecM(cross = cr, # the cross object
                   marker.info = map, # the genetic map with
                   # a column including bp position
                   gff = gff[gff[,3]=="gene",], # the gff file.
                   #For speed I cull to only 'genes'
                   attributeParse = "Name=",
                   # the strings to drop from the gene name
                   seqnameParse = "Chr0"
                   # the strings to drop from the Chr identifier
                   # so that they match the chromosome names in the cross.
                   )
```

```
## parsing attributes column of gff file
## culling chromosomes to those in the cross
## inferring mapping position for:
## chr 1 (n. features = 4172)
## chr 2 (n. features = 4751)
## chr 3 (n. features = 4774)
## chr 4 (n. features = 3108)
## chr 5 (n. features = 5013)
## chr 6 (n. features = 2758)
## chr 7 (n. features = 3172)
## chr 8 (n. features = 2267)
## chr 9 (n. features = 6110)
```

```
kable(head(geneCM[,-c(2,6,7,8)]))
```

	chr	feature	start	end	attribute	geneID	bp	pos
1467	1	gene	662182	665960	Name=Pahal.A00075	Pahal.A00075	664071.0	0
1477	1	gene	654463	661391	Name=Pahal.A00076	Pahal.A00076	657927.0	0
1511	1	gene	653322	654261	Name=Pahal.A00077	Pahal.A00077	653791.5	0
1519	1	gene	650510	652209	Name=Pahal.A00078	Pahal.A00078	651359.5	0
79828	1	gene	57826	60205	Name=Pahal.D02104	Pahal.D02104	59015.5	0
79838	1	gene	60485	61987	Name=Pahal.D02105	Pahal.D02105	61236.0	0

```
ggplot(map, aes(x = bp/1000000, y = pos))+
  geom_point(col = "red", pch = 8, cex = .5)+
  geom_point(data = geneCM, pch = ".")+
  facet_wrap(~chr)+theme_jtl()+
  labs(x = "physical position (Mbp)",
       y = "genetic mapping position (cM)")
```

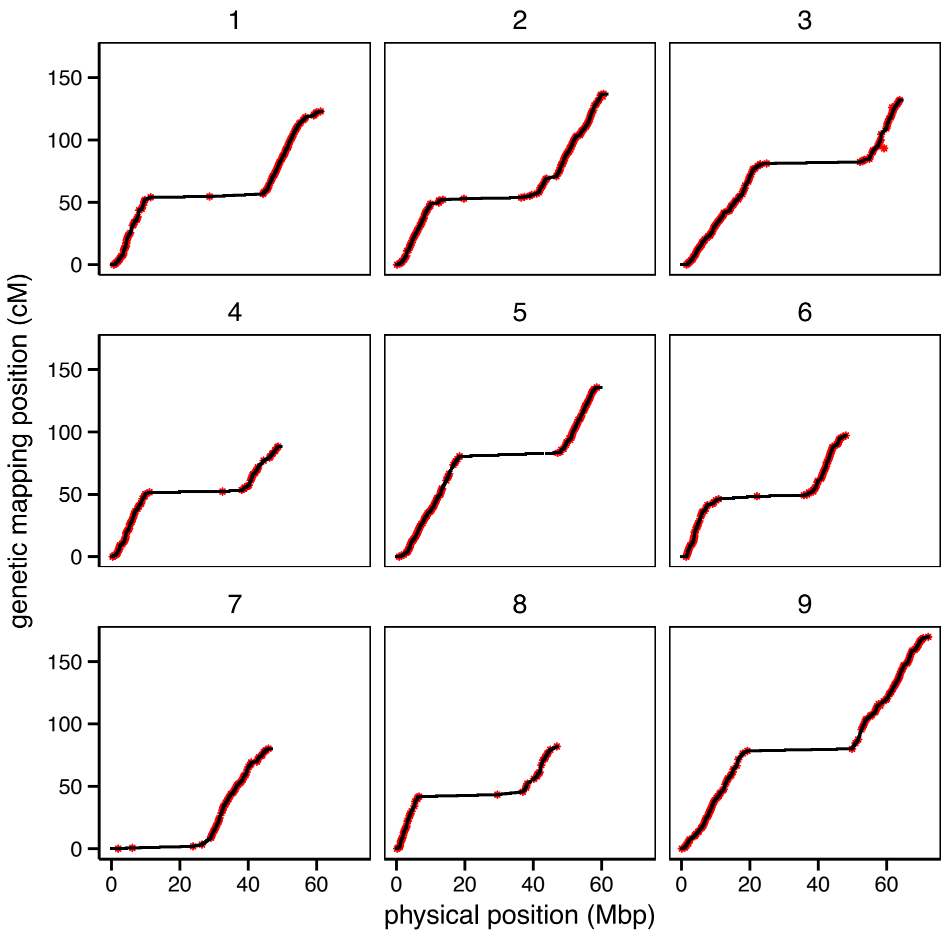


Figure 6: The relationship between phyical and mapping positions of all genes. The red points are the markers in the genetic map. The black points are all genes in the annotation. Note the low recombination rates in the pericentromeric regions . . . not surprising given that this is a grass.

6.1 Pull out genes in the intervals

```
candGenes<-findGenesInterval(findGenecM.output = geneCM,
                             calcCis.output = cis)
```

```
print(candGenes)
```

```
##   [1] "Pahal.C00791" "Pahal.C00792" "Pahal.C00793" "Pahal.C00794"
##   [5] "Pahal.C00795" "Pahal.C00796" "Pahal.C00797" "Pahal.C00798"
##   [9] "Pahal.C00799" "Pahal.C00800" "Pahal.C00801" "Pahal.C00802"
##  [13] "Pahal.C00803" "Pahal.C00804" "Pahal.C00805" "Pahal.C00806"
##  [17] "Pahal.C00807" "Pahal.C00808" "Pahal.C00809" "Pahal.C00810"
##  [21] "Pahal.C00811" "Pahal.C00812" "Pahal.C00813" "Pahal.C00814"
##  [25] "Pahal.C00815" "Pahal.C00816" "Pahal.C00817" "Pahal.C00818"
##  [29] "Pahal.C00819" "Pahal.C00820" "Pahal.C00821" "Pahal.C00822"
##  [33] "Pahal.C00823" "Pahal.C00824" "Pahal.C00825" "Pahal.C00826"
##  [37] "Pahal.C00827" "Pahal.C00828" "Pahal.C00829" "Pahal.C00830"
##  [41] "Pahal.C00831" "Pahal.C00832" "Pahal.C00833" "Pahal.C00834"
##  [45] "Pahal.C00835" "Pahal.C00836" "Pahal.C00837" "Pahal.C00838"
##  [49] "Pahal.C00839" "Pahal.C00840" "Pahal.C00841" "Pahal.C00842"
##  [53] "Pahal.C00843" "Pahal.C00844" "Pahal.C00845" "Pahal.C00846"
##  [57] "Pahal.C00847" "Pahal.C00848" "Pahal.C00849" "Pahal.C00850"
##  [61] "Pahal.C00851" "Pahal.C00852" "Pahal.C00853" "Pahal.C00854"
##  [65] "Pahal.C00855" "Pahal.C00856" "Pahal.C00857" "Pahal.C00858"
##  [69] "Pahal.C00859" "Pahal.C00860" "Pahal.C00861" "Pahal.C00862"
##  [73] "Pahal.C00863" "Pahal.C00864" "Pahal.C00865" "Pahal.C00866"
##  [77] "Pahal.C00867" "Pahal.C00868" "Pahal.C00869" "Pahal.C00870"
##  [81] "Pahal.C00871" "Pahal.C00872" "Pahal.C00873" "Pahal.C00874"
##  [85] "Pahal.C00875" "Pahal.C00876" "Pahal.C00877" "Pahal.C00878"
##  [89] "Pahal.C00879" "Pahal.C00880" "Pahal.C00881" "Pahal.C00882"
##  [93] "Pahal.C00883" "Pahal.C00884" "Pahal.C00885" "Pahal.C00886"
##  [97] "Pahal.C00887" "Pahal.C00888" "Pahal.C00889" "Pahal.C00890"
## [101] "Pahal.C00891" "Pahal.C00892" "Pahal.C00893" "Pahal.C00894"
## [105] "Pahal.C00895" "Pahal.C00896" "Pahal.C00897" "Pahal.C00898"
## [109] "Pahal.C00899" "Pahal.C00900" "Pahal.C00901" "Pahal.C00902"
## [113] "Pahal.C00903" "Pahal.C00904" "Pahal.C00905" "Pahal.C00906"
## [117] "Pahal.C00907" "Pahal.C00908" "Pahal.C00909" "Pahal.C00910"
## [121] "Pahal.C00911" "Pahal.C00912" "Pahal.C00913" "Pahal.C00914"
## [125] "Pahal.C00915" "Pahal.C00916" "Pahal.C00917" "Pahal.C00918"
## [129] "Pahal.C00919" "Pahal.C00920" "Pahal.C00921" "Pahal.C00922"
## [133] "Pahal.C00923" "Pahal.C00924" "Pahal.C00925" "Pahal.C00926"
## [137] "Pahal.C00927" "Pahal.C00928" "Pahal.C00929" "Pahal.C00930"
## [141] "Pahal.C00931" "Pahal.C00932" "Pahal.C00933" "Pahal.C00934"
## [145] "Pahal.C00935" "Pahal.C00936" "Pahal.C00937" "Pahal.C00938"
## [149] "Pahal.C00939" "Pahal.C00940" "Pahal.C00941" "Pahal.C00942"
## [153] "Pahal.C00943" "Pahal.C00944" "Pahal.C00945" "Pahal.C00946"
## [157] "Pahal.C00947" "Pahal.C00948" "Pahal.C00949" "Pahal.C00950"
## [161] "Pahal.C00951" "Pahal.C00952"
```

Our list of candidate genes is now 162

7 Culling candidate gene list by expression of the parents

To definitively clone a QTL, we need to get the list of candidates down to a managable number . . . this number depends on how easy it is to get and assay transgenics in your system. The first step towards narrowing down the candidate gene list is to determine what molecular mechanism you think could potentially be causing phenotypic variation. Here, we have data suggesting that the phenotype is driven by transcript abundance of a regulatory gene. Therefore, we expect the parents to have differential expression of the causal gene in the QTL interval. This means that any genes with NO differential expression between the parents can be disregarded in our candidate gene search.

Here, we test differential expression using the `DESeq2` R package and a pipeline in the `deTools` development version R package

```
devtools::install_github("jtlovell/deTools")
suppressPackageStartupMessages(library(deTools))

# if you don't have DESeq2 installed ... run this:
# source("https://bioconductor.org/biocLite.R")
# biocLite("DESeq2")
suppressPackageStartupMessages(library(DESeq2))
```

```
de.parent<-pipeDESeq2(counts = parentExp, info = parentInfo,
           formula = " ~ type + Treatment + type*Treatment",
           reduced = " ~ Treatment",
           testNames = "geno")
```

```
## running Likelihood ratio tests for results for:
##  ~ type + Treatment + type*Treatment  vs.   ~ Treatment
```

```
de.test<-de.parent$LRT_results
de.test$signif<-ifelse(de.test$padj_geno<=0.01, "sig","NS")
de.candidates<-de.test$gene[de.test$signif == "sig"]
print(de.candidates)
```

```
##  [1] "Pahal.C00795" "Pahal.C00796" "Pahal.C00804" "Pahal.C00807"
##  [5] "Pahal.C00812" "Pahal.C00821" "Pahal.C00825" "Pahal.C00826"
##  [9] "Pahal.C00827" "Pahal.C00829" "Pahal.C00831" "Pahal.C00832"
## [13] "Pahal.C00836" "Pahal.C00837" "Pahal.C00838" "Pahal.C00840"
## [17] "Pahal.C00841" "Pahal.C00845" "Pahal.C00846" "Pahal.C00851"
## [21] "Pahal.C00852" "Pahal.C00853" "Pahal.C00854" "Pahal.C00856"
## [25] "Pahal.C00857" "Pahal.C00858" "Pahal.C00861" "Pahal.C00862"
## [29] "Pahal.C00864" "Pahal.C00865" "Pahal.C00867" "Pahal.C00874"
## [33] "Pahal.C00877" "Pahal.C00878" "Pahal.C00880" "Pahal.C00882"
## [37] "Pahal.C00883" "Pahal.C00885" "Pahal.C00890" "Pahal.C00898"
## [41] "Pahal.C00900" "Pahal.C00903" "Pahal.C00904" "Pahal.C00906"
## [45] "Pahal.C00908" "Pahal.C00910" "Pahal.C00915" "Pahal.C00917"
## [49] "Pahal.C00920" "Pahal.C00925" "Pahal.C00927" "Pahal.C00929"
## [53] "Pahal.C00930" "Pahal.C00933" "Pahal.C00934" "Pahal.C00937"
## [57] "Pahal.C00938" "Pahal.C00939" "Pahal.C00942" "Pahal.C00943"
## [61] "Pahal.C00950" "Pahal.C00952"
```

We now have 62 candidates

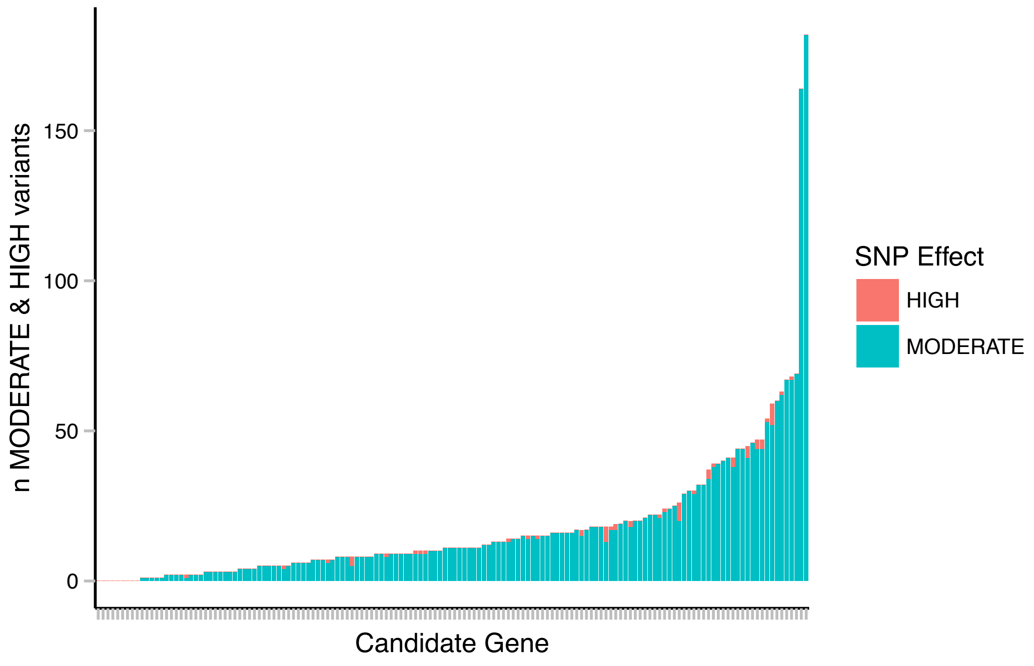
8 Culling candidates by DNA sequence variation

We may also expect phenotypes to be driven by *non-synonymous variants* ... In this case, one may generate a vcf (variant call format) matrix that details all sequence variants between the parents of the linkage mapping population, then annotate this file using snpeff and count the number of non-synonymous variants for each gene. Those genes with >0 non-synonymous variants could be your culled candidate gene list.

Here, we have annotated the genes in the interval and qualified the effects of variants as `LOW` (synonymous), `MODERATE` (non-synonymous or similar), `HIGH` (frameshift, premature stop, etc.).

```
## Need another packages here ... reshape2
vcf_toplot<-reshape2::melt(vcf_summary[,c("geneID","HIGH","MODERATE")])

nmodhigh<-with(vcf_toplot, tapply(value, geneID, sum))
nmodhigh<-nmodhigh[order(nmodhigh)]
vcf_toplot$geneID<-factor(vcf_toplot$geneID, levels = names(nmodhigh))
ggplot(vcf_toplot, aes(y = value, x = geneID, fill = variable))+
  geom_bar(stat = "identity")+
  theme_jtlbar()+
  theme(axis.text.x = element_blank())+
  labs(x = "Candidate Gene", y = "n MODERATE & HIGH variants", fill = "SNP Effect")
```



So, we need to set some threshold for a potential candidate gene ... lets just arbitrarily say that we need >3 MODERATE effect OR >0 HIGH effect variant

```
seq_candidates<-vcf_summary$geneID[vcf_summary$HIGH>=1 | vcf_summary$MODERATE>3]
```

How many of the differnetial expression candidates are also DNA sequence candidates?

```
table(de.candidates %in% seq_candidates)
```

```
##
## FALSE  TRUE
##    14    48
```

```r
de_and_seq_candidates<-de.candidates[de.candidates %in% seq_candidates]
print(de_and_seq_candidates)
```

```
##  [1] "Pahal.C00795" "Pahal.C00796" "Pahal.C00804" "Pahal.C00812"
##  [5] "Pahal.C00821" "Pahal.C00825" "Pahal.C00826" "Pahal.C00827"
##  [9] "Pahal.C00829" "Pahal.C00831" "Pahal.C00832" "Pahal.C00836"
## [13] "Pahal.C00837" "Pahal.C00838" "Pahal.C00841" "Pahal.C00845"
## [17] "Pahal.C00846" "Pahal.C00851" "Pahal.C00852" "Pahal.C00853"
## [21] "Pahal.C00854" "Pahal.C00856" "Pahal.C00858" "Pahal.C00861"
## [25] "Pahal.C00862" "Pahal.C00864" "Pahal.C00865" "Pahal.C00867"
## [29] "Pahal.C00877" "Pahal.C00878" "Pahal.C00880" "Pahal.C00885"
## [33] "Pahal.C00890" "Pahal.C00898" "Pahal.C00900" "Pahal.C00903"
## [37] "Pahal.C00906" "Pahal.C00908" "Pahal.C00910" "Pahal.C00915"
## [41] "Pahal.C00917" "Pahal.C00920" "Pahal.C00925" "Pahal.C00927"
## [45] "Pahal.C00929" "Pahal.C00933" "Pahal.C00934" "Pahal.C00937"
```

Our list of candidates is down to 48

9 Culling by expression of the mapping population.

If we believe that a QTL is caused by a gene expression polymorphism, then we are explicitly only interested in candidate genes that have *cis*-eQTL. That is, a local variant drives expression of the candidate gene.

To find this, we load normalized expression data from all genes in the confidence interval. And treat these expression values as phenotypes

```r
cr2<-cr
cr2$pheno<-f2Exp

allCands<-colnames(f2Exp)

s1.cand<-scanone(cr2,
                 pheno.col = allCands, # all the genes in the region
                 method = "hk",
                 addcovar = covar,
                 intcovar = covar,
                 chr = 3) # just look at Chr3 (speeds things up)
s1.cand.perm <- scanone(cr2,
                        pheno.col = allCands,
                        method = "hk",
                        addcovar = covar,
                        intcovar = covar,
                        chr = 3,
                        n.perm = 100, verbose =F)
```

Now lets look and see which of these candidate genes have a good cis-eQTL in the region

```r
thresholds = summary(s1.cand.perm, alpha = .05) # calculate permutation thresholds
max.s1s<-apply(s1.cand[,-c(1:2)], 2, max) # calculate QTL peak LOD
max.pos<-s1.cand$pos[apply(s1.cand[,-c(1:2)], 2, which.max)] # find the position of the QTL Peak
plot(max.pos,max.s1s,
     xlab = "Mapping position of QTL peak",
     ylab = "LOD score of QTL peak")
rect(xleft = cis$lowposition, xright = cis$highposition,
     ybottom = min(thresholds), ytop = max(max.s1s),
     col = rgb(1,0,0,.2))
```

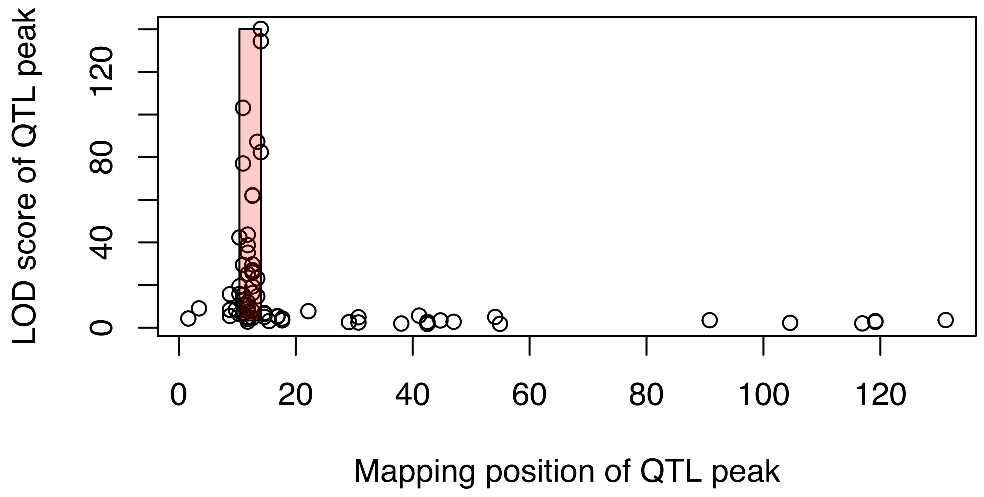


Figure 7: scanone plots for all of the genes that are found in the qtl interval on Chr3

Now, find all genes in the QTL interval with a significant QTL in the interval.

```
cisEqtl_candidates<-allCands[which(max.s1s>=thresholds &
        max.pos>=cis$lowposition &
        max.pos <= cis$highposition)]
table(de_and_seq_candidates %in% cisEqtl_candidates)
```

```
##
## FALSE  TRUE
##    24    24
```

```
seqExpCis_candidates<-de_and_seq_candidates[de_and_seq_candidates %in% cisEqtl_candidates]
print(seqExpCis_candidates)
```

```
##  [1] "Pahal.C00796" "Pahal.C00812" "Pahal.C00821" "Pahal.C00829"
##  [5] "Pahal.C00836" "Pahal.C00837" "Pahal.C00838" "Pahal.C00841"
##  [9] "Pahal.C00851" "Pahal.C00852" "Pahal.C00853" "Pahal.C00854"
## [13] "Pahal.C00861" "Pahal.C00862" "Pahal.C00877" "Pahal.C00878"
## [17] "Pahal.C00890" "Pahal.C00900" "Pahal.C00903" "Pahal.C00908"
## [21] "Pahal.C00910" "Pahal.C00917" "Pahal.C00925" "Pahal.C00937"
```

** Our list of candidate genes is down to 24**

10 Rank Candidate genes.

Here we use the covariance of expression and trait of interest in mapping population (Lovell et al. 2015, Plant Cell) to rank the potential effects of a candidate gene. To do so, we iteratively add, then remove the gene expression of a candidate to the QTL model. Those that reduce the signal of the focal QTL are better candidates than those which do not.

```
cand.mat<-pull.pheno(cr2,seqExpCis_candidates)
```

```r
covs<-covScanQTL(cross = cr,
                 pheno.col = "phenotype",
                 qtl = q, # the qtl model from the initial scan
                 expression.covariates = cand.mat, # matrix of gene expression data
                 qtl.method = "hk",
                 nperm = 20, # how many permutation to do to test for significance?
                 addcovar = covar,
                 intcovar = covar,
                 ylab = "LOD")
```

```
## creating the marker covariate dataset
## running covariate scan
```

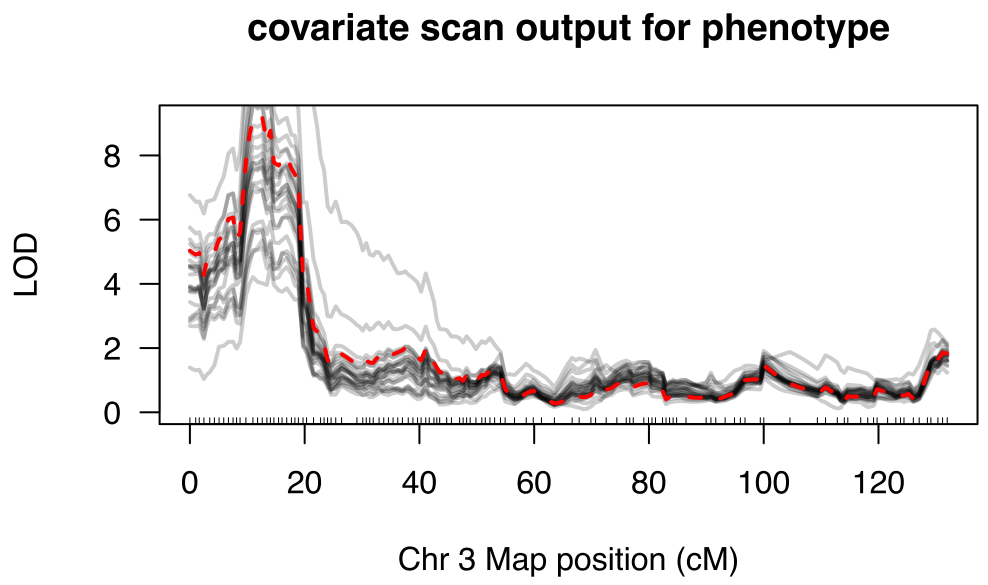


Figure 8: Output from the candidate gene search. The red line indicates the intial LOD profile. Each grey line represents the LOD profile when that gene is added to the model as a covariate. Lower values at the QTL peak are indicative of potentially greater effect on the focal phenotype.

```
## running permutation: 10 20
## done
```

Here are the top 6 candidates:

```r
kable(head(covs))
```

phenotype	candidateID	lodAtPeak	diffAtPeak	rank	perm.p
phenotype	Pahal.C00812	4.020582	5.170218	1	0
phenotype	Pahal.C00854	5.127274	4.063527	2	0
phenotype	Pahal.C00917	5.356228	3.834573	3	0
phenotype	Pahal.C00853	5.662500	3.528300	4	0
phenotype	Pahal.C00852	5.704658	3.486142	5	0
phenotype	Pahal.C00837	5.951539	3.239261	6	0

And all significant candidates:

```
covs$candidateID[covs$perm.p<=0.05]
```

```
##  [1] "Pahal.C00812" "Pahal.C00854" "Pahal.C00917" "Pahal.C00853"
##  [5] "Pahal.C00852" "Pahal.C00837" "Pahal.C00878" "Pahal.C00937"
##  [9] "Pahal.C00841" "Pahal.C00796" "Pahal.C00910" "Pahal.C00838"
## [13] "Pahal.C00925" "Pahal.C00890" "Pahal.C00829" "Pahal.C00908"
```

11 Conclusions

So, thats really it. The next step is to look at these candidates, figure out which ones might make sense … then do the molecular biology to acutally nail down causation. GOOD LUCK!