Candidate Gene Search Tutorial

JT Lovell

2017-10-24

Contents

---

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

---

1 Part 1: Overview

To search for candidate genes you need four objects.

1. gff - the gene model position dataset
2. markerBp - the basepair position of markers
3. cross - the QTL cross object used to identify QTL
4. interval - the numeric confidence interval of the QTL (chr, lower ci, upper ci)

To infer the potential of a candidate gene you need at least one of 7 datasets.

1. vcf - the polymorphisms between parents, in "vcf" format. It is optimal to have this annotated by snpEff or similar.
2. parentGeneExp - results of differential expression analysis between parents
3. cisEQtl - a list of genes with cis-eQTL
4. methyl - dataset containing the degree of methylation for each gene
5. geneDescr - Gene descriptions
6. GO - GO annotations
7. geneExp - gene expression of the mapping population

With these data, one can infer whether a gene is likely to contain the causal QTN(s)

2 Part 2: Getting set up

To start, you need the qtlTools package. Get it from github.

```
library(devtools)
install_github("jtlovell/qtlTools")
library(qtlTools)
```

Load the multitrait data from R/qtl

```
data(multitrait)
```

Create some fake physical positions of the markers allowing for low recombination in the middle of the chromosomes (as would be expected in the pericentromeric region)

```
map<-pullMap(multitrait)
map$bp<-0
for(i in unique(map$chr)){
  n<-sum(map$chr==i)
  p<-sin((1:n/n)*pi)
  map$bp[map$chr==i]<-cumsum(p*1000000)
}
```

Create a fake gff file

```
gff<-data.frame(chr = rep(paste0("scaffold_",1:5),each = 200),
   feature = rep("gene",1000),
   start = rep(seq(from = 0, to = max(map$bp), length = 200), 5),
   end = rep(seq(from = 0, to = max(map$bp), length = 200))+1000,
   strand = rep("+",1000),
   attribute = paste0("gene",1:1000,";","gene",1:1000,".1"), stringsAsFactors=F)
```

3 Part 3: Infer the physical position of the genes, using the position of the markers

```
geneCM<-findGenecM(cross = multitrait, marker.info = map, gff = gff,
   gffCols = c("chr","feature","start","end","strand","attribute"))
## parsing attributes column of gff file
## culling chromosomes to those in the cross
## inferring mapping position for:
## chr 1 (n. features = 200)
## chr 2 (n. features = 200)
## chr 3 (n. features = 200)
## chr 4 (n. features = 200)
## chr 5 (n. features = 200)
```

Plots showing the bp/cM patterns

```
par(mfrow=c(2,3))
for(i in unique(map$chr)){
  with(geneCM[geneCM$chr==i,], plot(pos,bp, col="grey",
                                    ylab = "physical position (bp)",
                                    xlab = "mapping position (cM)"))
  with(map[map$chr==i,], points(pos,bp, col=i, pch = 19, cex=.8))
}
```

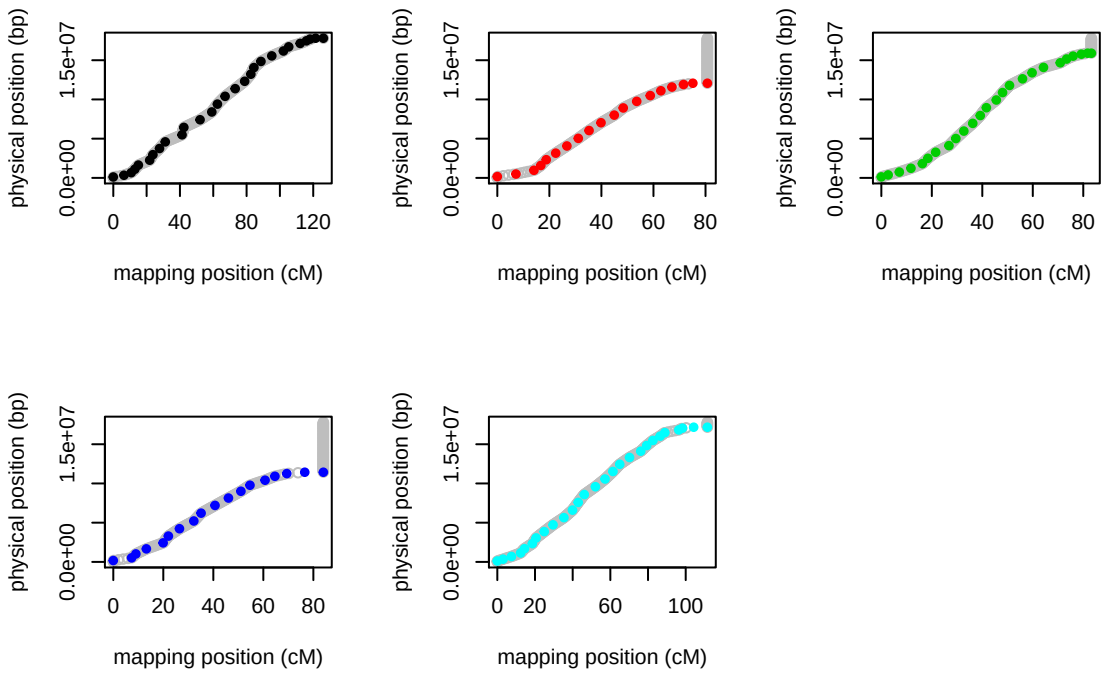


Figure 1: Physical and mapping positions of fake data for the 5 A. thaliana chromosomes.

4 Part 4: Find genes in the interval

Make qtl intervals

```
multitrait<-calc.genoprob(multitrait)
s1<-scanone(multitrait, method="hk", pheno.col=1)
perm<-scanone(multitrait, n.perm=100, method="hk",pheno.col=1, verbose=FALSE)
cis<-calcCis(cross = multitrait, s1.output=s1, perm.output=perm, drop=5)

par(mfrow = c(1,1))
plot(s1)
segmentsOnPeaks(multitrait, s1.output=s1, calcCisOutput = cis, int.y = 13.1)
```

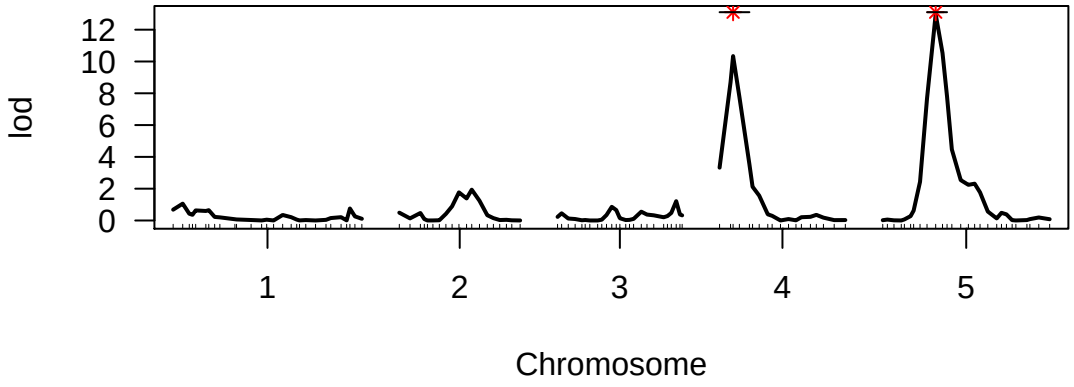


Figure 2: Scanone profile with confidence intervals.

Pull out genes in the intervals

```
candGenes<-findGenesInterval(findGenecM.output = geneCM, calcCis.output = cis)
print(candGenes)
## $`4@9`
##  [1] "gene601" "gene602" "gene603" "gene604" "gene605" "gene606" "gene607"
##  [8] "gene608" "gene609" "gene610" "gene611" "gene612" "gene613" "gene614"
## [15] "gene615" "gene616" "gene617" "gene618" "gene619" "gene620" "gene621"
## [22] "gene622" "gene623" "gene624" "gene625" "gene626" "gene627" "gene628"
##
## $`5@35`
##  [1] "gene854" "gene855" "gene856" "gene857" "gene858" "gene859" "gene860"
##  [8] "gene861" "gene862" "gene863" "gene864" "gene865" "gene866" "gene867"
## [15] "gene868" "gene869" "gene870" "gene871" "gene872" "gene873" "gene874"
## [22] "gene875" "gene876" "gene877" "gene878" "gene879" "gene880" "gene881"
## [29] "gene882" "gene883" "gene884" "gene885"
```

5 Part 5: next steps

There are a number of approaches to define how likely any gene is to be the candidate.

1. Genes with non-synonymous SNPs
2. Genes with cis-eQTL (Lowry et al. 2013, Plant Cell)
3. Genes with annotations similar to the trait of interest
4. Covariance of expression and trait of interest in mapping population (Lovell et al. 2015, Plant Cell)
5. Causal Inference testing

Here, we will explore the 4th option. First, we must simulate some gene expression data: Let's just focus on the chromosome 5 QTL and say there are 50 genes under the QTl interval. Here, we simulate expression (normalized around 0) with a few genes with correlated expression with the focal marker.

```
cross<-subset(multitrait, ind = !is.na(pull.pheno(multitrait, 1)))
phe<-pull.pheno(cross, 1)

mult.fact<-exp(seq(from = 0, to = 50, length.out = 50))
facs<-sapply(1:50, function(x){
  scale(sapply(scale(phe), function(y) rnorm(n = 1, mean = y, sd = mult.fact[x])))
})
plot(sapply(1:50, function(x) cor(phe, facs[,x])),
     ylab = "cor. coef. (expression ~ chr5 QTL genotype)",
     xlab = "gene id")
```

In this simplistic example, we are simulating a case where expression drives a linear, additive QTL effect. However, this approach is extensible and permits inference of QTL*Treatment, epistasis, multiple QTL models and other cases.

So, lets run the covariate scan, testing how the QTL profile is affected by the presence of gene expression in the model.

```
expression.covariates = facs
colnames(expression.covariates)<-paste0("gene",1:ncol(expression.covariates))
pheno.col = 1

qtl = makeqtl(cross, chr = max(s1)$chr, pos = max(s1)$pos, what = "prob")

test_additive<-covScanQTL(cross = cross,
                  pheno.col = 1,
                  qtl = qtl,
                  expression.covariates = expression.covariates,
                  qtl.method = "hk",
                  nperm = 20)
## creating the marker covariate dataset
## running covariate scan
```

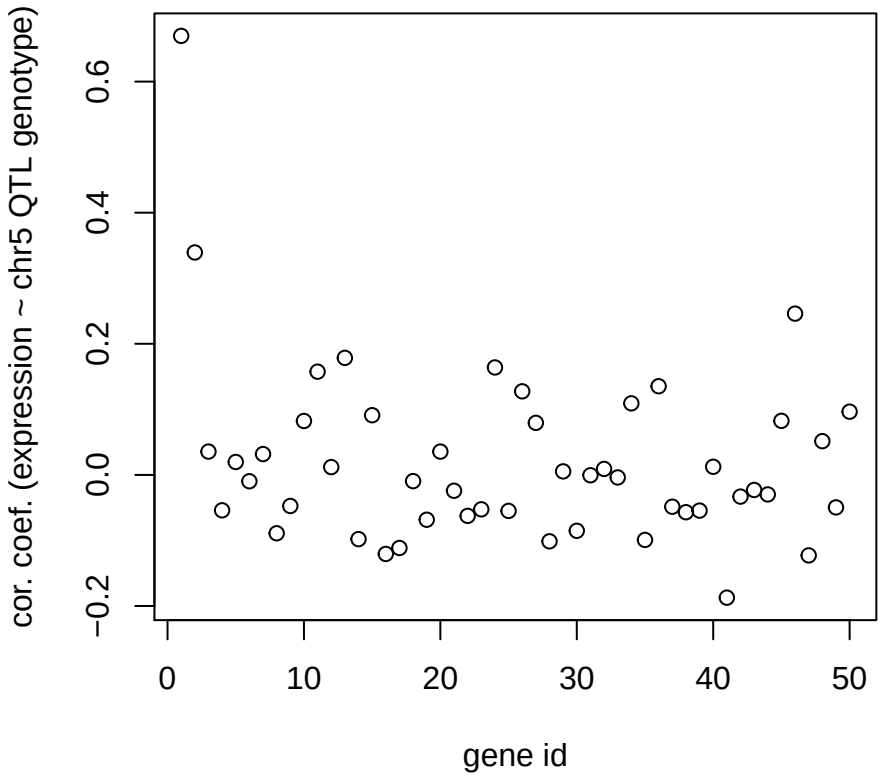
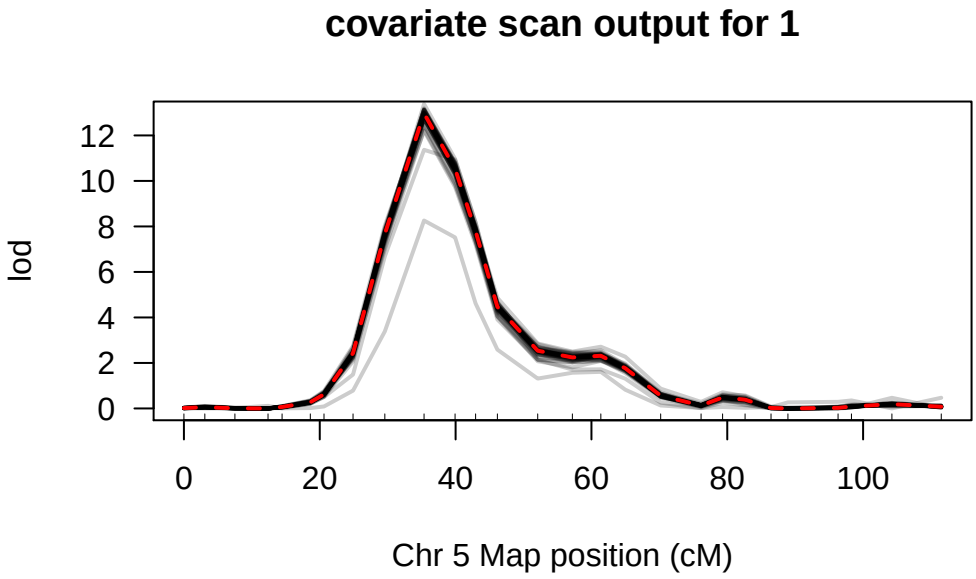


Figure 3: correlation matrix of simulated gene expression data.



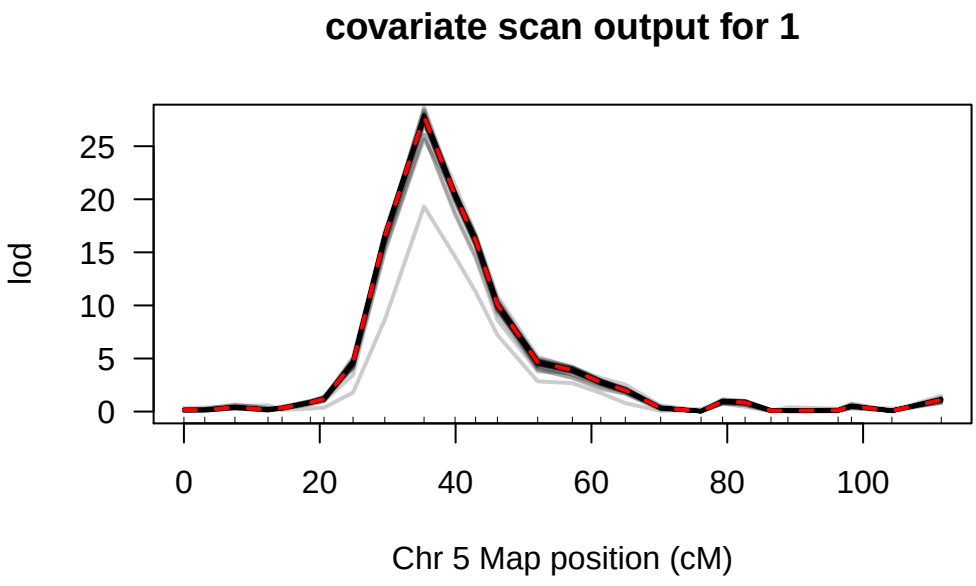
```
## running permutation: 10 20
## done
kable(head(test_additive), caption = "top candidate genes using a simple, additive covariate scan approach
```

Table 1: top candidate genes using a simple, additive covariate scan approach.

phenotype	candidateID	lodAtPeak	diffAtPeak	rank	perm.p
1	gene1	8.260024	4.7107869	1	0.00
1	gene2	11.368052	1.6027586	2	0.00
1	gene41	12.199803	0.7710079	3	0.00
1	gene46	12.256283	0.7145280	4	0.00
1	gene24	12.463861	0.5069500	5	0.05
1	gene16	12.468903	0.5019071	6	0.05

Now include another QTL in the model and enforce an epistatic interaction between the 2nd (focal) and 1st QTL.

```
qtl2 = makeqtl(cross, chr = summary(s1)$chr[4:5],
               pos = summary(s1)$pos[4:5], what = "prob")
test_epistasis<-covScanQTL(cross = cross,
                 pheno.col = 1,
                 qtl = qtl2,
                 which.epiqtl = 1,
                 focalqtl.index = 2,
                 expression.covariates = expression.covariates,
                 qtl.method = "hk",
                 nperm = 20)
## creating the marker covariate dataset
## running covariate scan
```



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

kable(head(test_epistasis), caption = "top candidate genes using a covariate scan that incorporates epista
```

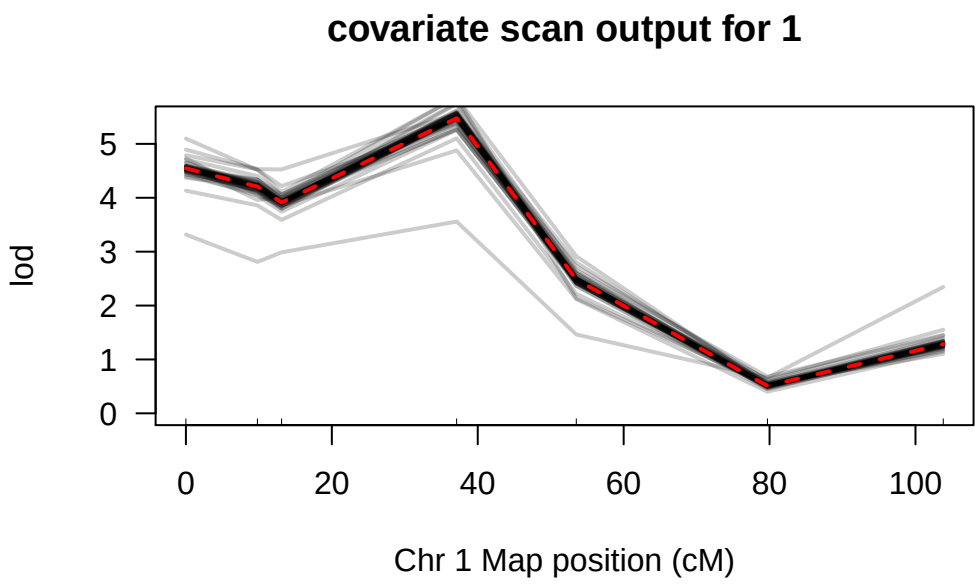
Table 2: top candidate genes using a covariate scan that incorporates epistasis

phenotype	candidateID	lodAtPeak	diffAtPeak	rank	perm.p
1	gene1	19.31651	8.4848130	1	0
1	gene2	25.62630	2.1750254	2	0
1	gene46	26.06901	1.7323192	3	0
1	gene41	26.08722	1.7141057	4	0
1	gene17	27.03513	0.7661928	5	0
1	gene10	27.14557	0.6557532	6	0

Now a little more complexity with an F2 cross and experimental covariates

```
data(fake.f2)
cross<-fake.f2
phe<-pull.pheno(cross, 1)
mult.fact<-exp(seq(from = 0, to = 50, length.out = 50))
facs<-sapply(1:50, function(x){
  scale(sapply(scale(phe), function(y) rnorm(n = 1, mean = y, sd = mult.fact[x])))
})
facs<-data.frame(facs)
colnames(facs)<-paste0("gene",1:ncol(facs))
cross<-calc.genoprob(cross)
sex = data.frame(sex = pull.pheno(cross, pheno.col = "sex"))
s1<-scanone(cross, addcovar = sex)
qtl = makeqtl(cross, chr = summary(s1)$chr[1], pos = summary(s1)$pos[1], what = "prob")

test_QTLxE<-covScanQTL(cross = cross,
                  pheno.col = 1,
                  qtl = qtl,
                  addcovar = sex,
                  intcovar = sex,
                  expression.covariates = facs,
                  qtl.method = "hk",
                  nperm = 20)
## creating the marker covariate dataset
## running covariate scan
```




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

kable(head(test_QTLxE), caption = "top candidate genes using a covariate scan that incorporates GxE")
```

Table 3: top candidate genes using a covariate scan that incorporates GxE

phenotype	candidateID	lodAtPeak	diffAtPeak	rank	perm.p
1	gene1	3.558792	1.9176026	1	0.00
1	gene48	4.878669	0.5977258	2	0.00
1	gene31	5.102574	0.3738204	3	0.00
1	gene26	5.257331	0.2190640	4	0.10
1	gene20	5.267516	0.2088787	5	0.00
1	gene22	5.274033	0.2023616	6	0.05