Using arghqtl: examples from *Arabidopsis thaliana*

Tom Ellis

2019-01-03

Contents

1 Background

Often one has used the same linkage-mapping population to look for QTL for multiple traits, or perform replicate experiments in the same or different environments. This is espeically true for plants, because once a mapping population has been generated it is straightforward to maintain indefinitely as seeds. However, the abundance of QTL models available can make it tricky to plot and interpret the results.

For example, Ågren et al. (2013) generated a mapping population derived from locally adpated lines of *Arabidopsis thaliana* from sites in Italy and Sweden, and tested them in reciprocal transplant experiments at both source sites over several years. Further work used the same mapping populations to delve into the genetic basis of relevant biolgical functions, such flowering time (Ågren et al. 2017), freezing tolerance (Oakley et al. 2014), and seed dormancy(Postma and Ågren 2016, 2015).

With so many traits over multiple years, this means a lot of QTL models to deal with. R/QTL has excellent facilities for fitting QTL models, but plotting tools are basic, and fairly ugly. `arghqtl` was created to make plotting and handling them easier - specifically cases where you think about how to plot a large QTL experiment and think 'aaaargh'.

2 Data

This vignette will use data from Ågren et al. (2013) to illustrate how to use `arghqtl`. Ågren et al. (2013) transplanted ~400 recombinant inbred lines in Autumn of 2009, 2010 and 2011, and quantified QTL survival to reproduction, fecundity (fruit number per adult plant) and overall fitness (fruit number per plant seedling). They mapped QTL for these traits using R/QTL (Broman et al. 2003), and we will begin by importing the `qtl` objects generated by the `stepwiseqtl()` function for these data, and the linkage map data used to

create them. Here, we only consider data for 2010 and 2011. We will also need data on QTL model fits (the output of `fitqtl`), which we will also take as given.

```
data("qtl_models") # qtl models.
data("qtl_fits") # outputs of fitqtl
data("mloc") # linkage map:
```

For each trait there are data for Italy (it) and Sweden (sw) in each year. Each element is a `qtl` object describing a QTL model.

3 Fitness in a single year

We will begin by plotting a simple example: QTL for fitness at two sites in 2011; see figure 1 for what we are aiming for. More specifically the aim is to create a plot showing each of the 5 chromosomes, each with two lanes - one for Italy and one for Sweden. To create a QTL plot we first need to define a set of coordinates, onto which further information will be added.

3.1 `plotQTL` objects

```
library('arghqtl')

pos_plot <- plotQTL(1:5, mloc, nlanes = 2)
```

`pos_plot` is a list of different kinds of information about the plot; see `?plotQTL` for the full description. The really crucial information is that it describes a set of 5 tracks (chromosomes), each with 2 lanes.

```
pos_plot$ntracks
#> [1] 5
pos_plot$nlanes
#> [1] 2
```

Lanes can be defined by the positions along the x-axis of their edges, and their centres. This is important because we will plot QTL along lane centres, and some kind of divider to separate lanes along their edges. Calling `pos_plot$lane_centres` shows a table of positions for both lanes on each track.

```
pos_plot$lane_centres
#>          1    2    3    4    5
#> lane_1 2.5 16.5 30.5 44.5 58.5
#> lane_2 7.5 21.5 35.5 49.5 63.5
```

`pos_plot$lane_margins` shows the same information for lane margins. There are three margins rather than two, because every lane must be surrounded by a margin on either side.

```
pos_plot$lane_margins
#>           1  2  3  4  5
#> margin_1  0 14 28 42 56
#> margin_2  5 19 33 47 61
#> margin_3 10 24 38 52 66
```

3.2 Setting up the plot

This code creates a blank (base) plot, and adds marker positions.

```
plotQTL_base(pos_plot, by=20, maxy = 20, track_labels = paste("Chr.", 1:5))
plotQTL_ladder(pos_plot)
```

We can add labels using the base R `text` function, and telling it where the lanes are.

```
# lane labels
text(pos_plot$lane_centres, 10, c("It", "Sw"))
```

3.3 Adding QTL positions.

To plot a QTL model, use `plotQTL_qtl` and tell it which lane to plot to. QTL will be plotted to the correct chromosome automatically, along with their 95% Bayesian credible intervals, and the direction of effects. To do this we also need to supply a `fitqtl` object.

```
# Plot QTL in Italy
plotQTL_qtl(
  plotQTL = pos_plot,
  qtl_object = qtl_models$ffit$it2011,
  model_fit = qtl_fits$ffit$it2011,
  lane = 1,
  col = 'red3'
  )
# Plot QTL in Sweden
plotQTL_qtl(
  plotQTL = pos_plot,
  qtl_object = qtl_models$ffit$sw2011,
  model_fit = qtl_fits$ffit$sw2011,
  lane = 2,
  col = 'blue4'
  )
```

Ågren et al. (2013) distinguished between 'well-defined' and 'poorly-defined' QTL depending on whether or not their credible intervals were shorter or longer than 1/4 the length of the shortest chromosome (15.2cM). You can supply a threshold length to define poorly-defined QTL using `fill_cutoff`, which will render poorly defined QTL with open arrows.

```
plotQTL_qtl(
  plotQTL = pos_plot,
  qtl_object = qtl_models$ffit$it2011,
  model_fit = qtl_fits$ffit$it2011,
  lane = 1,
  fill_cutoff = 15.2,
  col = 'red3'
  )
```

Here is the full code to create 1:

```
par(mar = c(0,3,0,0))

# Plot skeleton
pos_plot <- plotQTL(
  chr = 1:5,
  marker_locations = mloc,
  nlanes = 2,
  left_gap = 2,
```

```
  right_gap = 4)

# Plot empty QTL map.
plotQTL_base(pos_plot, by=20, maxy = 20, track_labels = paste("Chr.", 1:5))
plotQTL_ladder(pos_plot, lty=2)
text(pos_plot$lane_centres, 10, c("It", "Sw"))

# Plot QTL in Italy
plotQTL_qtl(
  plotQTL = pos_plot,
  qtl_object = qtl_models$ffit$it2011,
  qtl_fits$ffit$it2011,
  lane = 1,
  col = 'red3'
  )
# Plot QTL in Sweden
plotQTL_qtl(
  plotQTL = pos_plot,
  qtl_object = qtl_models$ffit$sw2011,
  model_fit = qtl_fits$ffit$sw2011,
  lane = 2,
  col = 'blue4'
  )
```

4 Colocalisation of QTL

Figure 2 shows a more complicated example. There are three traits, assess in two years at two sites. We can't really know exactly where a QTL 'is', but we can group QTL that map to roughly the same place, as indicate with grey boxes.

4.1 Group QTL into clusters

`cluster_qtl` takes a **list** of multiple QTL models and their corresponding `fitqtl` objects, and clusters them into groups. Loci from two QTL models are considered to colocalise if:

1. Their 95% Bayesian credible intervals overlap, and/or
2. Maximum-likelihood estimates for QTL positions are within 2cM of each other.

This procedure is rough, and far from ideal, but we lack any better criteria for deciding whether two loci are the same or different (Ågren et al. 2013, 2017; Dittmar et al. 2014; Oakley et al. 2014). It will be immeidately obvious that the longer the credible intervals are for a locus, the more likely it is to overlap with something, so we remove QTL with CIs longer than some threshold; as previously mentioned, following we will use 15.2cM.

```
cluster_ffit <-
  cluster_qtl(
  chr = 1:5,
  qtl_list = qtl_models$ffit,
  model_fit_list = qtl_fits$ffit,
  threshold = 15.2
  )
```

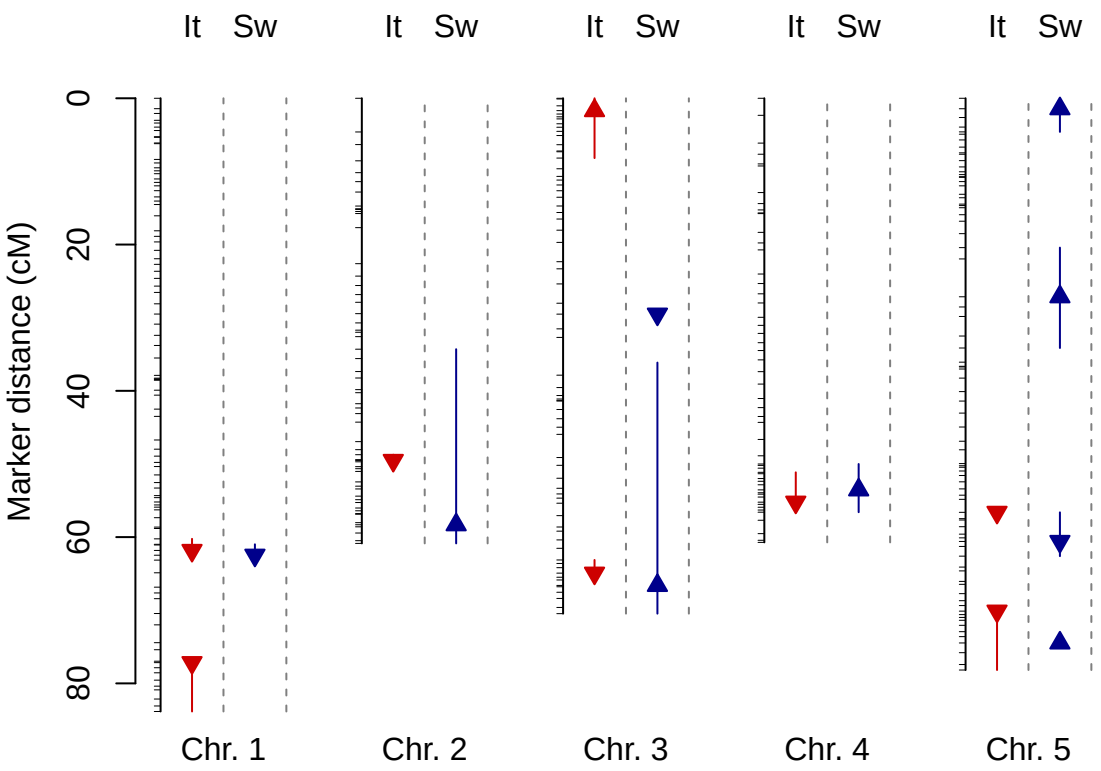

Figure 1: QTL for fruit number per planted seedling in Sweden and Italy in 2011.

`cluster_qtl` returns three dataframes. Firstly, `cluster_ffit$summary` returns information on each cluster identified, and the positions, effect sizes and percentage of exlpained phenotypic variance explained by all constituent loci. Note that mean values are weighted by the LOD score of each constituent QTL. `cluster_ffit$full.list` shows the details for each constituent QTL and which cluster it belongs to. Finally, `boxes` summarises upper and lower bounds of each cluster to facilitate plotting.

There is a slight complication for in that we need to exclude Sweden in 2010 from the objects related to survival, because no QTL were identified there:

```
clusters <- list(
  frut = cluster_qtl(chr=1:5, qtl_models$frut, qtl_fits$frut, 15.2),
  surv = cluster_qtl(chr=1:5, qtl_models$surv[c(1,2,4)], qtl_fits$surv[c(1,2,4)], 15.2),
  ffit = cluster_qtl(chr=1:5, qtl_models$ffit, qtl_fits$ffit, 15.2))
```

4.2 Plot clusters

First create the base plot. We want to plot four lanes for each trait for a total of twelve lanes.

```
pos_plot <- plotQTL(1:5, mloc, nlanes = 12)
plotQTL_base(pos_plot, by=20, maxy = 20)
text(35, -95, "Chromosome")
```

The `boxes` data generated by `cluster_qtl` show chromosome number, and upper and lower bounds for cluster position. It is straight forward to plot these. We do this first so that they do not obscure lane dividers and QTL.

```
plotQTL_boxes(pos_plot, clusters$frut$boxes, margins = c(1,5), col='gray75', border='gray50')
plotQTL_boxes(pos_plot, clusters$surv$boxes, margins = c(5,9), col='gray75', border='gray50')
plotQTL_boxes(pos_plot, clusters$ffit$boxes, margins = c(9,13), col='gray75', border='gray50')
```

4.3 Custom labels

Lane dividers can be customised based on lane position. Calling `plotQTL_ladder` but setting `lty=0` plot marker positions, but leaves lane dividers blank. You can call the base R function `segments` to plot lane dividers independently. In this case, dashed lines are drawn between each lane, and traits are separated by solid line. Notice that no line is plotted on the left most margin (the `-1` index to `pos_plot$lane_margins`), because we have already added marker positions.

```
plotQTL_ladder(pos_plot, lty=0)

# Lanes dividers
segments(pos_plot$lane_margins[-1,], 0,
         pos_plot$lane_margins[-1,], rep(-pos_plot$track_lengths, each=pos_plot$nlanes),
         lty=c(2,2,2,1), col=c('gray50','gray50','gray50',1))
```

QTL models follow a hierarchy of year within site within trait. We can add custom lane labels to reflect that by specifying where we would like things plotted. Notice that these examples specify either the centres of lanes or the lane margins, dependning where things need centering.

```
# Trait labels
text(pos_plot$lane_margins[c(3,7,11),], 15, c("Fruits\nper plant", "Survival", "Fruits\nper seedling"),
     col=1, cex=0.6, srt=-90, adj=c(0.5,0.5))
# Site labels
text(pos_plot$lane_margins[seq(2,13,2),], 6, c("IT", "SW"),
     cex=0.6, adj=c(0.5,0.5), srt=0, col=c('red3','blue4'))
```

```
# Year labels
text(pos_plot$lane_centres, 2, c("10", "11"),
     cex=0.5, adj=c(0.5,0.5), srt=0, col='gray30')
```

4.4 Plot multiple QTL objects

There are twelve QTL models to plot, and it would be confusing to copy-paste the command for `plotQTL_qtl` each time. Instead, you can just use the same call multiple times with a loop. Of course, it would be even better to loop over traits, but in this example this is difficult because of the Null QTL object for survival in Sweden in 2010.

```
for (i in 1:4) {
  plotQTL_qtl(
    pos_plot,
    qtl_models$frut[[i]],
    qtl_fits$frut[[i]],
    fill_cutoff = 15.2,
    lane = i,
    col = colvec[i]
  )
}
for (i in 1:4) {
  plotQTL_qtl(
    pos_plot,
    qtl_models$surv[[i]],
    qtl_fits$surv[[i]],
    fill_cutoff = 15.2,
    lane = i + 4,
    col = colvec[i]
  )
}
for (i in 1:4) {
  plotQTL_qtl(
    pos_plot,
    qtl_models$ffit[[i]],
    qtl_fits$ffit[[i]],
    fill_cutoff = 15.2,
    lane = i+8,
    col = colvec[i]
  )
}
```

4.5 Full plot code

Here is the full code to create figure 2.

```
pos_plot <- plotQTL(1:5, mloc, nlanes = 12)
plotQTL_base(pos_plot, by=20, maxy = 20)
text(35, -95, "Chromosome")

# Plot boxes
plotQTL_boxes(pos_plot, clusters$frut$boxes, margins = c(1,5), col='gray75', border='gray50')
```

```r
#> Warning in if (margins == "full") {: the condition has length > 1 and only
#> the first element will be used
plotQTL_boxes(pos_plot, clusters$surv$boxes, margins = c(5,9), col='gray75', border='gray50')
#> Warning in if (margins == "full") {: the condition has length > 1 and only
#> the first element will be used
plotQTL_boxes(pos_plot, clusters$ffit$boxes, margins = c(9,13), col='gray75', border='gray50')
#> Warning in if (margins == "full") {: the condition has length > 1 and only
#> the first element will be used

# Lanes dividers
segments(pos_plot$lane_margins[-1,], 0,
         pos_plot$lane_margins[-1,], rep(-pos_plot$track_lengths, each=pos_plot$nlanes),
         lty=c(2,2,2,1), col=c('gray50','gray50','gray50',1))
plotQTL_ladder(pos_plot, lty=0)

# Trait labels
text(pos_plot$lane_margins[c(3,7,11),], 15, c("Fruits\nper plant", "Survival", "Fruits\nper seedling"),
     col=1, cex=0.6, srt=-90, adj=c(0.5,0.5))
# Site labels
text(pos_plot$lane_margins[seq(2,13,2),], 6, c("IT", "SW"),
     cex=0.6, adj=c(0.5,0.5), srt=0, col=c('red3','blue4'))
# Year labels
text(pos_plot$lane_centres, 2, c("10", "11"),
     cex=0.5, adj=c(0.5,0.5), srt=0, col='gray30')

colvec = c("red3", "red3", "blue4", "blue4")
for (i in 1:4) {
  plotQTL_qtl(
    pos_plot,
    qtl_models$frut[[i]],
    qtl_fits$frut[[i]],
    fill_cutoff = 15.2,
    lane = i,
    col = colvec[i]
  )
}
for (i in 1:4) {
  plotQTL_qtl(
    pos_plot,
    qtl_models$surv[[i]],
    qtl_fits$surv[[i]],
    fill_cutoff = 15.2,
    lane = i + 4,
    col = colvec[i]
  )
}
#> Warning in plotQTL_qtl(pos_plot, qtl_models$surv[[i]],
#> qtl_fits$surv[[i]], : Supply either a qtl object with a fitqtl object, or
#> else a QTL cluster object from cluster_qtl.
for (i in 1:4) {
  plotQTL_qtl(
    pos_plot,
    qtl_models$ffit[[i]],
```

```
    qtl_fits$ffit[[i]],
    fill_cutoff = 15.2,
    lane = i+8,
    col = colvec[i]
  )
}
```

Literature cited

Ågren, Jon, Christopher G Oakley, Sverre Lundemo, and Douglas W Schemske. 2017. "Adaptive Divergence in Flowering Time Among Natural Populations of *Arabidopsis Thaliana*: Estimates of Selection and QTL Mapping." *Evolution* 71 (3): 550–64.

Ågren, Jon, Christopher G. Oakley, John K. McKay, John T. Lovell, and Douglas W. Schemske. 2013. "Genetic Mapping of Adaptation Reveals Fitness Tradeoffs in *Arabidopsis Thaliana*." *Proceedings of the National Academy of Sciences* 110 (52): 21077–82. https://doi.org/10.1073/pnas.1316773110.

Broman, Karl W., Hao Wu, Saunak Sen, and Gary A. Churchill. 2003. "R/Qtl: QTL Mapping in Experimental Crosses." *Bioinformatics* 19: 889–90.

Dittmar, Emily L, Christopher G Oakley, Jon Ågren, and Douglas W Schemske. 2014. "Flowering Time QTL in Natural Populations of *Arabidopsis Thaliana* and Implications for Their Adaptive Value." *Molecular Ecology* 23 (17): 4291–4303.

Oakley, Christopher G, Jon Ågren, Rachel A Atchison, and Douglas W Schemske. 2014. "QTL Mapping of Freezing Tolerance: Links to Fitness and Adaptive Trade-Offs." *Molecular Ecology* 23 (17): 4304–15.

Postma, Froukje M., and Jon Ågren. 2015. "Maternal Environment Affects the Genetic Basis of Seed Dormancy in *Arabidopsis Thaliana*." *Molecular Ecology* 24 (4): 785–97. https://doi.org/10.1111/mec.13061.

———. 2016. "Early Life Stages Contribute Strongly to Local Adaptation in *Arabidopsis Thaliana*." *Proceedings of the National Academy of Sciences* 113 (27): 7590–5. https://doi.org/10.1073/pnas.1606303113.

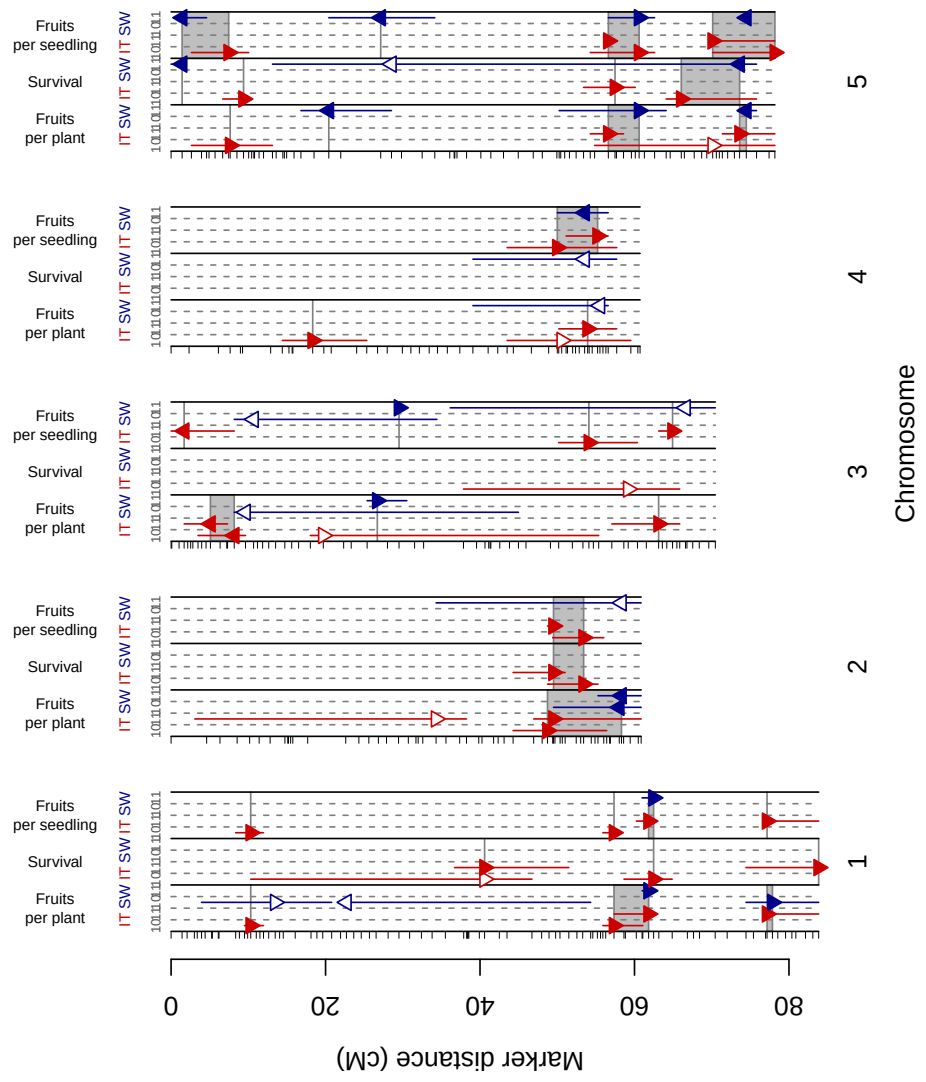

Figure 2: QTL for fecundity, survival and fitness in 2010 and 2011. Arrows indicate most-likely QTL position and the effect of the Swedish genotype (upward: increased phenotype; downward: decreased phenotype) in Italy (red) and Sweden (blue). Vertical bars show the 95% Bayesian credible intervals for QTL position. Open arrows show QTL with credible intervals >15.2cM. Grey boxes indicate the range of colocalising QTL for a single trait across site-year combinations.