CTL mapping in R

Danny Arends, and Ritsert C. Jansen

University of Groningen
Groningen Bioinformatics Centre & GCC

Revision # 1

XX Okt 2011

Abstract: Tutorial for the Correlated Trait Loci (CTL) mapping package to reconstruct genetic regulatory networks using the R package 'ctl' and the mapctl commandline tool. This tutorial is targetted at people with a basic understanding of genetics and would like to analyse there inbred cross (RIL, DH, BC) or any population using SNP markers. Main focus will be explaining the functionality of the software, and a how-to on using datasets currently in Excel, tab delimited files (.csv) or use *read.cross* from R/qtl. In the methodology and background only a short introduction in CTL mapping theory is given.

Software:	http://www.mapctl.org/download.html
Manual:	http://www.mapctl.org/manual.html
Source-code:	http://www.github.com/DannyArends/ctl
E-mail:	Danny.Arends@Gmail.com

Some words in advance

Welcome reader to this tutorial, you'll be thrown into the world of network reconstruction using differential correlations. This methodology CTL mapping is not the easiest but the results can be of great help unraveling trait to trait interactions in all types of eukaryotes. In this tutorial I'll try to use as few difficult concepts as possible. But some always *sneak* in.

Enjoy this tutorial, and good luck hunting that heritability.
Danny Arends

PS. Use a good text editor
PPS. Use a repository / backup system

Background & Methodology

Some introduction on the methodology, skip this chapter first (DO come back later) if you have a dataset and just want to use the software.

Format and load your data

Loading your Excel / CSV data
Use the example excel file and simply replace the example data by your own then:

$$File- > SaveAs- > TODO$$

After you have stored the data as plain text open the files in a text editor (NOT WORD, for windows Notepad++ is a good option) and check to see if your data is stored correctly, if not reformat your data in both the genotype.csv and the phenotype.csv files to match the format below.
NOTE: Individual order must match between the genotype and the phenotype file...

The genotypes.csv file containing the genotype matrix is stored individuals x genetic marker:

	Chromosome	Loc
Marker1	1	
marker2	1	
marker3	1	
...	...	
MarkerN	7	

The phenotypes.csv file containing individuals x trait measurements:

	Chromosome	Location	Ind1	Ind2	...	IndN	
Trait1	...	...	5.8	11.0	9.0	...	1.6
Trait2	...	...	6.3	12.2	NA	...	1.3
Trait3	...	...	5.1	11.1	12.3	...	2.0
...	...	...	...	...	...	...	...
Trait	...	...	9.8	15.8	23.0	...	3.4

After verifying your files are formatted correctly (this is the source of most errors). Start R and load in the data, check that the output looks like below:

```
>   setwd("/Path/To/Data/")
>   genotypes <- read.csv("genotypes.csv",row.names=1, header=FALSE,sep="\t")
>   traits <- read.csv("phenotypes.csv",row.names=1, header=FALSE,sep="\t")
>   genotypes[1:5,1:10]     #Show 5 individuals, 8 markers, and genetic locations
>   traits[1:5,1:10]        #Show 5 individuals, 8 traits, and genetic locations
>   mapinfo <- genotypes[,1:2]
>   genotypes <- genotypes[,-c(1,2)]  # Remove the genetic locations
>   traits <- traits[,-c(1,2)]        # Remove the genetic locations
```

Use an R/qtl formatted dataset
Provided is the main interface functions to R/qtl: $CTLscan.cross()$ this functions accepts and R/qtl formatted cross object as input and will scan CTL, perform permutations and transform the detected differential

correlation to LOD score matrices. As an example we show the code to load the internal multitrait dataset provided in R/qtl, load your own by using the *read.cross*() function:

```
> require(qtl)          #Loads the R/qtl package
> data(multitrait)      #Loads the dataset
> multitrait            #Print basic dataset information
> ?read.cross           #List of formats supported by R/qtl
```

Adding genetic map information
This step is optional, but plots and generated networks look much nicer when chromosome locations of genetic markers are supplied. The 'mapinfo' object is matrix with 3 columns: "Chr" - the chromosome number, "cM" - the location of the marker in centiMorgans and the 3rd column "Mbp" - The location of the marker in Mega basepairs.

NOTE: Marker order must match with the genotype file...

The structure of the *mapinfo.csv* file:

	Chr	cM	Mbp
Ind1	1	1.0	0.01
Ind2	1	1.2	0.34
Ind3	1	3.1	1.3
...	...	...	...
IndN	24	120.8	20.3

We load the genetic map data into R by using the following command:

```
> mapinfo <- read.csv("mapinfo.csv",row.names=1,col.names=TRUE)
> mapinfo[1:5,1:3]      #Show the first 5 marker records
```

Finally... after all this we can start scanning QTL and CTL...

Scanning for CTL / *CTLscan*

We start explaining how to map CTL using the basic options of the *CTLscan* function. This function scan CTL and produces an output "CTLobject".

```
> require(ctl)                       #Loads the R/ctl package
> data(ath.metab)                    #Loads the example dataset
> geno   <- ath.metab$genotypes      #Short name
> traits <- ath.metab$phenotypes     #Short name
> #Scan all phenotypes for CTLs, using the default options (Could take some time)
> ctls   <- CTLscan(geno,traits)
```

If you have loaded an R/qtl "cross" object, use the *CTLscan.cross* interface, this will automagically extract the cross, check if the type of "cross" is supported, and deal with the *geno.enc* parameter:

```
> library(qtl)
> data(multitrait)
> ctls   <- CTLscan.cross(multitrait)
```

Now you might feel tempted to just dive into the results, But... first lets take a look at some options that might apply to your experimental setup like treatments applied or growth conditions, and which (if set incorrectly) could seriously reduce mapping power. Also you might want to add your own/previous QTL results.

CTLscan options
The genotype matrix and the *geno.enc* parameter, if data is of a different encoding then the standard 1 / 2 it can be specified by using the *geno.enc* parameter.

```
> #Load a dataset with "AA" and "AB" genotypes
> ctls <- CTLscan(geno,traits,geno.enc=c("AA","AB"))
```

The important *n.perms* parameter, which should be set to as many as possible, especially when publishing any results, but for a quick the default of 100 is sufficient:

```
> ctls_quick_scan      <- CTLscan(geno,traits,n.perm=100)
> ctls_for_publication <- CTLscan(geno,traits,n.perm=15000)
```

CTLs could be condition dependant just like QTL, so if one has multiple conditions or treatments. we need to accomodate that in the underlying models.

NOTE: Current version doesn't handle conditions (only the QTL are compensated).

If no QTLs are supplied the internal QTL mapping routine is used, which can deal with just a single condition per individual. An environmental condition can be specified by the *conditions* parameter:

```
> #Specify that individuals 1:51 were in condition A / e.g. treatment A
> #Specify that individuals 52:100 were in condition B / e.g. treatment B
> #Specify that individuals 101:162 were in condition C / e.g. control
> conditions <- c(rep(1,51),rep(2,49),rep(3,62))
> ctls <- CTLscan(geno, traits, conditions=conditions)
```

When experimental setups are more complex a user can supply QTL results via the *have.qtl* option. An example is shown below in which we use QTL results obtained by using single marker mapping via *scanone* and the new Multiple QTL mapping routine *mqmscan* in R/qtl:

```
> require(qtl)        #load the R/qtl package
> data(multitrait)    #load the dataset
> oneqtls   <- scanone(multitrait, pheno.col=1:24)[,-c(1,2)]
> ctls_one  <- CTLscan.cross(multitrait, pheno.col=1:2, have.qtl=qtls)
```

And a more advanced example where the MQM scan uses 30 genetic markers as cofactors to first build up a genetic model underlying the QTL and locus to locus interactions:

```
> cofactors <- mqmautocofactors(multitrait, 30)
> mqm_qtls  <- mqmscan(multitrait, cofactors=cofactors, pheno.col=1:24)[,-c(1,2)]
> ctls_mqm  <- CTLscan.cross(multitrait, pheno.col=1:2, have.qtl=qtls)
```

RESULTS - Plots and CTL networks

So now that we have our ctls calculated we can visualize the results in multiple ways. The easiest way is first to plot the two overview CTL heatmaps. The first heatmap shows the summarized LOD scores for the Trait X Marker CTLs. This plot normally looks very similar to a QTL heatmap:

```
> image(ctls,against="markers")
```

The second heatmap shows the summarized LOD scores for the Trait * Trait CTLs, this shows the summarized amount of evidence we find for each Trait * Trait interaction:

```
> image(ctls,against="phenotypes")
```

After the overview plots it is time to go more in depth, and look at some individual CTL profiles the CTLobject 'ctls' contains multiple CTLscans. To print a summary of the first scan use:

```
> ctls[[1]]
```

To plot other summaries just replace the 1 by the trait number. We can also use the plot function on individual CTLscans and shows a colorful and detailed view of the (selected) Trait x Marker X Traits interactions detected in the dataset.

```
> plot(ctls[[1]])
```

Get a list of significant CTLs

Get significant CTL interactions by using CTLsignificant.

```
> sign <- CTLsignificant(ctl_scan)
```

CTLnetwork and cytoscape
The CTLnetwork function outputs CTL and (optional) QTL data from the CTLobject into a network format, if mapinfo is available it is used. The network format is relatively easy and can also be parsed by many other programs. As an example we use cytoscape, but first lets generate some network input files:

```
> ctl_network <- CTLnetwork(ctls, lod.threshold = 5, mapinfo)
```

The CTL edges in the network_full.sif file are annotated as:

Trait	CTL	Trait	LOD Score

The CTL edges in the network_summary.sif file look like this:

Trait	CTL	Trait	avgLOD Score	sumLOD Score	nTraits

Additional QTLs are added to both files in the folowing structure:

Trait	QTL	Marker	LOD Score

This allows the first additional infromation column (Nr 4) to act as an distance measurement, the closer two traits or two markers are together in the network the stronger a detected QTL or a CTL.

Big data and Dctl

Big datsets require special care, R isn't the best platform to deal with large scale genomics data from micro- or tilling arrays (Just to mention RNAseq and full DNA strand sequencing). Thus we also provide a commandline version of our CTL mapping routine: mapctl.exe, for more information on all the command line parameters can be found at http://www.mapctl.org/mapctl.html The mapctl commandline tool uses the same input files as the R version, and supports qtab the new genotype and phenotype encoding scheme used by qtlHD. It provides only QTL and CTL scanning and permutation features, the output is analyzed in R as outlined above. To load the results from a mapctl scan into R by specifying the input files and the output directory created by the tool using:

```
> ctls <- read.dctl("genotypes.csv","phenotypes.csv",results="<PATH>/output/")
```

Some famous last words

Package CTL function overview

CTLscan	Main function for Genome wide scan for Correlated Trait Loci
CTLscan.network	Scan only a subset of traits for CTL using start gene(s)
CTLpermute	Significance thresholds for CTL by permutation
plot	Plot CTL profiles at increasing cut-offs
image	Image a multiple phenotype scan