CTL mapping in R

Danny Arends, Pjotr Prins, and Ritsert C. Jansen

University of Groningen
Groningen Bioinformatics Centre & GCC
Revision # 1

First written: Oct 2011
Last modified: Jan 2018

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, F2, DH, BC) or any outbred population (e.g. Human GWAS) using SNP markers. The main focus will be explaining the functionality of the software, and a how-to format your data for R when using datasets currently stored in Excel format, 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/CTLmapping
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. The CTL mapping methodology is not the easiest but the results can be of great help unraveling trait to trait interactions in all types of crosses. 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 missing heritability.
Danny Arends

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

Background & Methodology

Differences in correlations between traits within an inbred population are determined at each genetic marker. Phenotypes are assigned to genotype groups and a single phenotype is used to scan all other phenotypes for a loss or gain of correlation. The likelihood profiles (similar to QTL profiles) of this 'loss of correlation' measurement shows a very high degree of overlap with classical QTL profiles. However additional information is available from the phenotype x phenotype interactions. With the right dataset (ideally a combination of: classical phenotypes, protein abundance and gene expression) CTL shows the genetic wiring of the classical phenotypes and identify key players in the genetic / protein network underlying classical phenotypes using QTL and CTL information.

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 (do not format data using MS WORD, for windows Notepad++ is a good option) and check to see if your data is formatted correctly. If this is not the case, please reformat your data to match the example format shown below.

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

	Marker1	Marker2	Marker3	...	MarkerN
Ind1	1	1	1	...	2
Ind2	1	2	2	...	1
Ind3	2	2	1	...	1
...	...	...	...	...	...
IndN	1	1	1	...	2

The phenotypes.csv file containing individuals (rows) x traits (columns) measurements:

	Trait1	Trait2	Trait3	...	TraitN
Ind1	5.8	11.0	9.0	...	1.6
Ind2	6.3	12.2	NA	...	1.3
Ind3	5.1	11.1	12.3	...	2.0
...	...	...	...	...	...
IndN	9.8	15.8	23.0	...	3.4

NOTE: Individual order must match between the genotype and the phenotype file...

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, 10 markers
>   traits[1:5,1:10]       #Show 5 individuals, 10 traits
```

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.metabolites)           #Loads the example dataset
> geno   <- ath.metab$genotypes   #Short name
> traits <- ath.metab$phenotypes  #Short name
> # Scan all phenotypes against each other for CTLs, using the default options
> 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
CTL mapping can be performed using three different strategies ("Exact", "Full", and "Pairwise") to calculate significance. The default strategy "Exact" is the fastest one, but the statistical power of this stategy is poor and it doesn't handle data distributions with a lot of outliers properly. For publication it is better to use one of the other strategies "Full" or "Pairwise" since these strategies perform permutations, which deals with outliers and non-normal distributions in a correct way.

When the startegy is not set to the default "Exact", the important parameter to consider is the *n.perms* parameter, which should be set to as many as possible. When you just want to have a quick estimation if there are any effects (1) use the default strategy, or (2) use only a limited a mount of permutations. However, when publishing any results, make sure to increase the number of permutations to e.g. 15000, for a quick look at any effects the default of setting of doing 100 permutations should be sufficient:

```
> ctls_quick_scan1      <- CTLscan(geno, traits)
> ctls_quick_scan2      <- CTLscan(geno, traits, strategy = "Full", n.perm=100)
> ctls_for_publication  <- CTLscan(geno, traits, strategy = "Full", n.perm=15000)
```

To make the analysis fast, use the *nthreads* parameter, this will use multiple logical cores when the pc has a multi-core CPU, and can reduce the required total analysis by quite a bit. The algorithm will not allow you to specify more cores then logical processors, so when setting a value too large the algorithm will use only the maximum amount of logical cores detected.

```
> ctls_quicker_scan <- CTLscan(geno, traits, nthreads = 4)
```

Non-parametric analysis (using ranked data) is done by default, to minimize the effect of outliers on the analysis. However if your trait distributions are normally distributed you should / can enable paramteric analysis, to obtain more statistical power. To perform parametric analysis set the parameter *parametric* to TRUE, this use the original phenotype values for the analysis.

```
> ctls_parametric_scan <- CTLscan(geno, traits, parametric = TRUE)
```

A new feature added to allow computation of CTLs on a trait by trait basis is to disable multiple testing correction. By default it is enable to prevent users from overestimating CTLs in their data. Since CTL mapping is doing a lot of statistical testing during a single analysis run, proper multiple testing should always be performed. However when running CTL mapping on a cluster (e.g. on a trait by trait basis) it might be needed to disable the automated multiple testing correction, since each node of the cluster will only see a subset of the data. Disable multiple testing adjustment, byu setting the parameter *adjust* to FALSE.

```
> ctls_uncorrected_scan <- CTLscan(geno, traits, adjust = FALSE)
```

When the analysis finishes with an error, you can turn on verbose output, but setting the parameter *verbose* to TRUE. This will produce a lot of textual output in the R window, which can sometimes help to identify an issue with the dataset. turning on verbose output has no influence on the analysis, and will only make the algorithm produce more text output during the analysis.

```
> ctls_scan <- CTLscan(geno, traits, verbose = TRUE)
```

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 the individual trait CTL scans and show a more 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