Bioconductor's Trigger package

Lin Chen, Dipen Sangurdekar, and John D. Storey[‡]
[‡]Email: jstorey@princeton.edu

October 17, 2016

Contents

1 Overview

The `trigger` package guides an integrative genomic analysis. Integrative genomic data usually consists of genomic information from various sources, which includes genetic information (genotype), high-dimensional intermediate traits in the genome (e.g., gene expression, protein abundance) and/or higher-order traits (phenotypes) for an organism. In the following examples, we mainly discuss intermediate traits of gene expression. It should be noted that this package can also be applied to protein abundance and/or other continuous trait expression.

The package contains functions to: (1) construct global linkage map between genetic marker and gene expression; (2) analyze multiple-locus linkage (epistasis) for gene expression; (3) quantify the proportion of genome-wide variation explained by each locus and identify eQTL linkage hotspots; (4) estimate pair-wise causal gene regulatory probability and construct gene regulatory networks; and (5) identify causal genes for a quantitative trait of interest.

This document provides a tutorial for using the `trigger` package. The package contains the following functions:

- `trigger.build`: Format the input data
- `trigger.link`: Genome-wide eQTL analysis
- `trigger.mlink`: Multi-locus linkage (epistasis) analysis
- `trigger.eigenR2`: Estimate the proportion of genome-wide variation explained by each eQTL
- `trigger.loclink` and `trigger.net`: Network-Trigger analysis
- `trigger.netPlot2ps`: Write the network from a trigger probability matrix to a postscript file
- `trigger.trait`: Trait-Trigger analysis

To view the help file for the function `trigger.link` within R, type `?trigger.link`. If you identify bugs related to basic usage please contact the authors directly. Otherwise, any questions or problems rergarding `snm` should be sent to the Bioconductor mailing list. Please do not send requests for general usage to the authors.

2 A yeast data set

The basic input data of this package consists of (1) a $m_m \times n$ marker genotype matrix with m_m marker genotypes in rows and n samples/arrays in columns; (2) a $m_e \times n$ gene expression matrix (or intermediate trait expression matrix) with m_e genes in rows and$n$ samples/arrays in columns; (3) a $m_m \times 2$ marker position matrix, of which the first column is the chromosome name, and the second column is the position of each marker, with each row corresponding to one marker in the marker genotype matrix; and (4) a $m_e \times 3$ gene position matrix, of which the first column is the chromosome name, and the second/third column is the starting/ending coordinate of each gene, with each row corresponding to one gene in the expression matrix. Please code the names of autosomal chromosomes to be integers and the name of sex chromsome to be "X". Also note that the same unit (e.g., base pair, kb, or cM) should be used for marker positions and gene positions. As an illustration of input data format and various analysis offered in this package, we demonstrate the functionality of this package using a data set from a yeast eQTL study [**?**, **?**]. In the study, a genetic cross of *Saccharomyces cerevisiase* BY4716 and RM11-1a strains was utilized to generated 112 F1 recombinant segregants. Each individual strain was then genotyped and gene expression measurements were done in a controlled growth environment. The data set consists of a list of four matrices:

- `marker`: A 3244×112 genotype matrix
- `exp`: A 6216×112 gene expression matrix
- `marker.pos`: A 3244×2 matrix of marker position information
- `exp.pos`: A 6216×3 matrix of gene position information.

This yeast data set is included in the package as the dataset `yeast`. To load the data, type `data(yeast)`, and to view a description of this data type `?yeast`. Once the data is loaded, one can type `attach(yeast)` to attach the yeast data. After the analysis is done, type `detach(yeast)` to detach the data set. We use a randomly generated subset of the data for the purpose of this vignette.

```
> library(trigger)
>  data(yeast)
>  names(yeast)

[1] "marker"     "exp"        "marker.pos" "exp.pos"

> #reduce data size for vignette run time
> set.seed(123)
> #select subset of 400 traits
> gidx = sort(sample(1:6216, size = 400))
> yeast$exp = yeast$exp[gidx,]
> yeast$exp.pos = yeast$exp.pos[gidx,]
> #select subset of markers
> midx = sort(sample(1:3244, size = 500))
> yeast$marker = yeast$marker[midx,]
> yeast$marker.pos = yeast$marker.pos[midx,]
> attach(yeast)
> dim(exp)
```

```
[1] 400 112
```

The function `trigger.build` formats the input data and returns a `S4 class` object for the convenience of subsequent analyses. It will convert the marker genotype matrix to a matrix of integers starting from 1 (a matrix of 1 or 2 for haploid genotypes, or 1, 2, or 3 for diploid genotypes).

```
> trig.obj <- trigger.build(marker=marker, exp=exp,
+         marker.pos=marker.pos, exp.pos=exp.pos)
> trig.obj

*** TRIGGER object ***
Marker matrix with  500 rows and  112 columns
Expression matrix with  400 rows and  112 columns

> detach(yeast)
```

3 Genome-wide eQTL analysis

The function `trigger.link` computes pair-wise likelihood ratio statistic for linkage of each gene-marker pair in the genome. If there are markers on sex chromosome, `gender` of each sample should be specified and the gender-specific mean will be computed for each genotype to obtain a likelihood ratio statistic. When the option `norm` is `TRUE`, each gene (row) of expression data matrix will be normalized to standard $N(0,1)$ based on the rank of the expression values for each gene. Since the null likelihood ratio statistic follow a chi-square distribution, parametric p-values will be computed based on the observed statistics. The function updates `trig.obj` with a matrix `stat` of likelihood ratio statistics and a matrix of p-values `pvalue` corresponding to gene-marker pairs in the genome, with genes in rows and markers in columns.

The function `plot` with the argument `type = "link"` takes the matrix of p-values for linkage of each gene-marker pair, calls the measures below a `cutoff` to be significant and plots the significant gene-marker pairs in a genome-wide eQTL linkage map. Genes and markers are ordered according to their genome positions. Applying the functions to the yeast data set, in Figure **??** we plot the genome-wide linkage map of eQTL and gene expression at p-value cutoff 3.3×10^{-4}, which corresponds to 5% FDR[**?**]. Note that the function `plot` thresholds the significance measures `pvalue` below cutoff. If one would like to threshold the significance measures above a threshold, one can apply the function to the negative matrix of significance measures and choose a negative cutoff.

```
> trig.obj = trigger.link(trig.obj, norm = TRUE)

> plot(trig.obj, type = "link", cutoff = 1e-5)
```

4 Multi-locus linkage (epistasis) analysis

The function `mlink` performs multi-locus linkage (epistasis) analysis for each selected gene expression. The option `idx` in the function can be used to select a subset of genes at a time. The function identifies a major locus and a secondary locus for each selected gene and estimates the

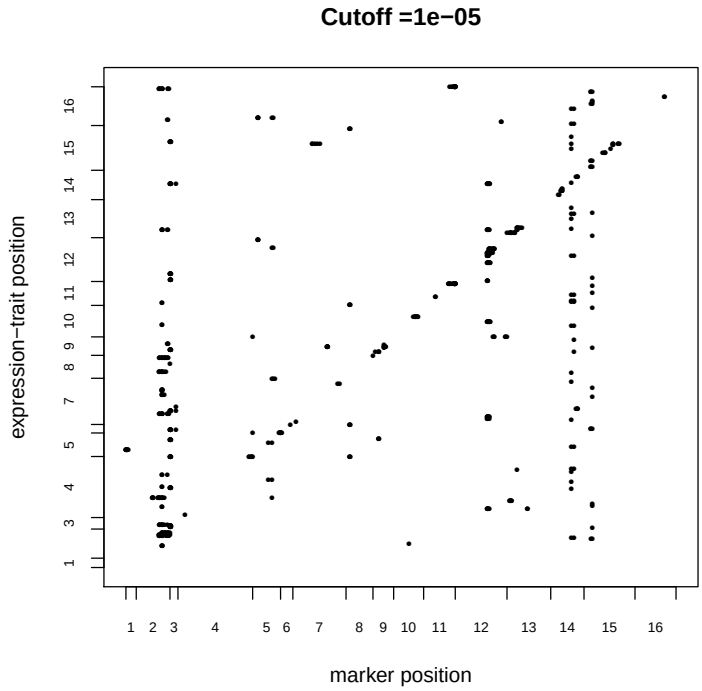

Figure 1: Genome-wide eQTL and gene expression linkage map.

likelihood ratio statistics of linkage. It then computes the posterior probabilities of major locus linkage, secondary locus linkage and joint linkage. Q-values (estimated false discovery rates) can be estimated for the joint linkage probabilities. A detailed description of the algorithm can be found in [**?**]. The function outputs a list of indices of major locus and secondary locus for each selected gene, a matrix of major locus linkage probability, secondary locus linkage probability and joint linkage probability, and a vector of q-values for joint linkage.

To visualize the significant epistasis loci in the genome, the function `plot` with argument `type = "mlink"` takes the output from `mlink` and plots the locus-pair showing significant epistasis effect at a q-value cutoff `qcut`. Each number in the output plot indicates the number of significant genes that are affected by the epistasis effect from a marker pair (if `bin.size = NULL`). If the option `bin.size` is specified, each chromosome will be divided into several bins, each with size `bin.size`. Markers within a bin will be considered as at a same position. Each number in the output plot indicates the number of significant genes that are affected by the epistasis effect from two markers, one from each bin. We plot the marker/bins with according to their genome positions. The x and y axis are the positions of the markers/bins.

We apply the function `mlink` to the yeast data set and use `plot` to plot the significant epistasis loci at a 10% FDR joint significant level [**?**] in Figure **??**.

```
> trig.obj = trigger.loclink(trig.obj)

[1] Computing local-linkages with a window size of 30 kb
[1] 10% completed
```

```
[1] 20% completed
[1] 30% completed
[1] 40% completed
[1] 50% completed
[1] 60% completed
[1] 70% completed
[1] 80% completed
[1] 90% completed
[1] 100% completed
```

```
> trig.obj = trigger.mlink(trig.obj, B = 10, idx = NULL)
```

```
[1] Start to calculate multi-locus linkage statistics ...
[1] 10% completed
[1] 20% completed
[1] 30% completed
[1] 40% completed
[1] 50% completed
[1] 60% completed
[1] 70% completed
[1] 80% completed
[1] 90% completed
```

```
> plot(trig.obj, type = "mlink", qcut = 0.2, bin.size = 50000)
```

5 Proportion of genome-wide variation captured by each eQTL

A classic R^2 measure can be used to estimate the proportion of one gene expression variation that captured by a locus of interest, and the R^2 in this setting is often called "heritability" in genetics. Recently, when genome-wide expression measurements are readily available, a high-dimensional version of R^2 is proposed to estimate the proportion of genome-wide expression variation explained by a locus [**?**]. Here we call it as $eigenR^2$. The function `trigger.eigenR2` estimates the $eigenR^2$ for each locus in the genome, and the `plot` function with argument `type = "eigenR2"` plots the estimated $eigenR^2$ values for each marker versus its genome position. A locus with high $eigenR^2$ is potentially a linkage hotspot that a lot of the genes are linked to. By chance, a random locus not capturing any genome-wide expression variation will have an expected $eigenR^2$ of $\frac{1}{n-1}$, where n is the sample size. If the logical option `adjust=TRUE`, the R^2 estimates will be adjusted for sample size effect and the expected $eigenR^2$ after adjustment is zero. One can also use the function `trigger.eigenR2` to compute the average of R^2 of a locus for every gene expression, by setting the logical option `meanR2=TRUE` (Fig **??**).

```
> trig.obj = trigger.eigenR2(trig.obj, adjust = FALSE)
```

```
> plot(trig.obj, type = "eigenR2")
```

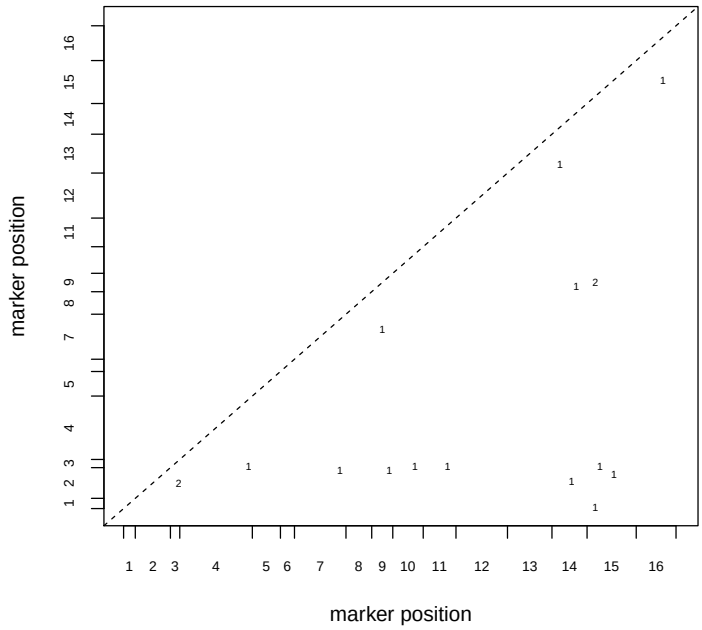

Figure 2: Plot of significant epistasis loci at 10% FDR level.

6 Network-Trigger analysis

The functions `trigger.loclink` and `trigger.net` provide an analysis based on an algorithm called "Trigger" to construct gene regulatory network [**?**]. The algorithm establishes the equivalence between a type of causal regulation of two genes to three testable conditions: 1) local linkage of a first gene, 2) secondary linkage of a second gene to the same locus, and 3) conditional independence between locus to second gene given the expression of the first one. The function estimates the posterior probability of each condition for every selected pair of genes in the genome and obtains the joint posterior probability of causal regulation for each gene pair as the product of the probabilities of three conditions. These regulatory probabilities can further be used to construct a gene regulatory network.

The function `trigger.loclink` identifies the best local linkage marker for every gene in the genome and estimates the local linkage probability for each by borrowing information across genes. One can use the option `window.size` to specify the size of a window that places a gene in the center. Every marker within the window is a candidate marker for local-linkage to the gene. If `window.size = NULL`, all the markers on a same chromosome as the gene will be used as candidate markers for local-linkage.

The function `trigger.net` takes the output of `trigger.loclink` and further estimates the secondary linkage and conditional independence probabilities. By specifying the probability threshold `prob.cut`, one can choose to only compute regulatory probabilities of a regulator to all the other genes, if the local-linkage probability of the regulator is above the threshold. The local linkage

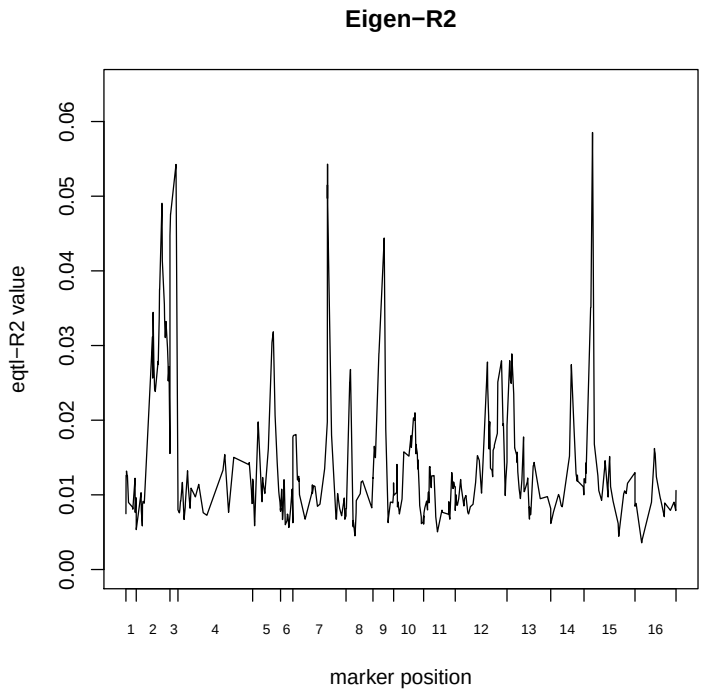

Figure 3: Plot of genome-wide $eigenR^2$

condition is not a necessary condition for establishing causality. The inclusion of this condition will increase the estimation efficiency and accuracy, and make the estimates conservative. An option of not including this condition by setting `include.loc = FALSE` is also provided. One can also select a subset of putative causal genes to construct the network, by specific the indices of the genes in the argument `idx`. In the following example, the network is constructed by analyzing by selecting all genes `idx = NULL` in the dataset. The function `trigger.net` outputs a matrix of genome-wide regulatory probabilities with putative regulators in rows and regulated genes in columns. The matrix is not symmetric. If gene i is estimated to be causal for gene j with high probability, then the probability of regulation from gene j to gene i should be low. Note that the function writes the data to a file `net_trigg_prob.txt` in the working directory and reads in the file at the end of computation. Before repeating the calculation, this file should be deleted since new results will be appended to the file.

```
> trig.obj = trigger.loclink(trig.obj, window.size = 10000)

[1] Computing local-linkages with a window size of 10 kb
[1] 10% completed
[1] 20% completed
[1] 30% completed
[1] 40% completed
[1] 50% completed
[1] 60% completed
[1] 70% completed
[1] 80% completed
```

```
[1] 90% completed
[1] 100% completed
```

```
> trig.prob = trigger.net(trig.obj, Bsec = 100, idx = NULL)
```

```
[1] Computing network-Trigger regulatory probabilities ...
[1] 10% completed
[1] 20% completed
[1] 30% completed
[1] 40% completed
[1] 50% completed
[1] 60% completed
[1] 70% completed
[1] 80% completed
[1] 90% completed
[1] 100% completed
```

```
>  dim(trig.prob)
```

```
[1] 400 400
```

7 Visualize directed network from estimated regulatory probability matrix

The package also offers a function `trigger.netPlot2ps` to visualize the structure of the causal regulatory network from network-Trigger probabilities. The function inputs a regulatory probability matrix `trig.prob`, constructs a directed network based on significant regulatory relationships above a threshold `pcut` and writes the network to a postscript file with name `filenam`. The function is dependent on the software `Graphviz` (available at http://www.graphviz.org/). If the total number of significant regulatory relationships (directed edges) of the network is below 1000, we plot each gene (node) with shape `node.shape` with its name labeled inside. The default shape is box. Otherwise, we plot each gene as a `dot` without name. The top `nreg` (by default `nreg`=20) regulators will be plotted in red ellipses with their names inside. One can also specified the color of nodes and edges in the plot with `node.color` and `edge.color`, respectively. The default color for genes (except for top regulators) is green, with blue edges connecting them. See manual of `Graphviz` for other available colors and shapes of nodes.

Our function will output a `filenam.dot` file, which is then written to a postscript file using `Graphviz` with the following four possible layouts: `radial` (default), `energy-minimized`, `circular` and `hierarchical` One can specify just the initial letter to choose one layout. For large networks, `radial` or `energy-minimized` is recommended. Once the layout is chosen, the function will run one of the following command to construct a network and write the network to a postscript file.

$ twopi -Tps filenam.dot -o filenam.ps # radial layout, the default layout
$ neato -Tps -Gmaxiter=1000 filenam.dot -o filenam.ps # energy-minimized layout
$ circo -Tps filenam.dot -o filenam.ps # circular layout
$ dot -Tps filenam.dot -o filenam.ps # hierarchical layout

The command `neato` plots the network with `energy-minimized` layout. To avoid long waiting for constructing a large network, we specify the maximum iteration to be 1000 (`-Gmaxiter=1000`).

In Figure **??**, we apply the function `trigger.netPlot2ps` to the estimated yeast regulatory probability matrix from previous section (output from `trigger.net` function). The resulting network is written to a postscript file `net95.ps` with `energy-minimized` layout.

```
trigger.netPlot2ps(trig.obj, trig.prob, pcut = 0.95, filenam = "net50", nreg = 20)

system("ps2pdf net50.ps")
```

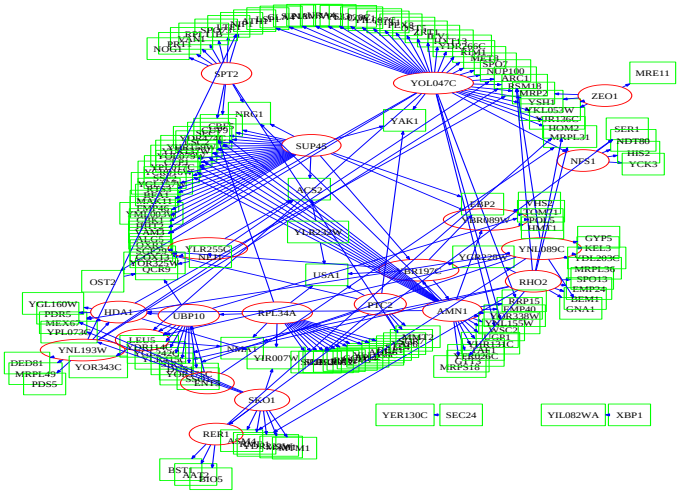

Figure 4: A yeast causal regulatory network at 95% trigger probability cutoff for the selected 400 genes.

8 Trait-Trigger analysis

Trait-Trigger is an algorithm that identifies the causal genes involved in the pathway from a fixed QTL to a quantitative trait of interest, where the trait can either be a gene transcript or a complex phenotypic trait. The function `trigger.trait` identifies putative causal genes for a given trait of interest, and then estimates the p-value for causal linkage for each regulator.

First, the genotype-expression data has to be exported to `cross` formation (See `qtl` package for details). This is done by the function `trigger.export2cross`, which formats the data, writes it to a `.csv` file in the working directory and then reads in the data and stores it in a `cross` file.

```
> # Re-attach dataset to re-include all traits for analysis
> data(yeast); attach(yeast)
> dim(exp)

[1] 6216  112

> trig.obj <- trigger.build(marker=marker, exp=exp,
+         marker.pos=marker.pos, exp.pos=exp.pos)

> cross = trigger.export2cross(trig.obj, plotarg = FALSE, verbose = FALSE)

 --Read the following data:
        112  individuals
        3244  markers
        6216  phenotypes
 --Cross type: bc
```

We apply the function to identify the causal genes for a *DSE1*, a daughter cell-specific protein, which was shown to be regulated by the regulator *AMN1* responsible for daughter cell separation[**?**]. If `addplot = TRUE`, the function plots a linkage map (Figure **??**) for the trait over the chromosome or chromosome where the LOD score for linkage crosses the threshold specified by `thr`. The function identifies *AMN1* as the putative causal regulator having the minimum p-value for causal linkage. Alternatively, the trait could be also entered as an expression vector `trait = exp[1727,]` or a phenotype vector having the same number of columns as the genotype data.

```
> causreg = trigger.trait(trig.obj, trait = "DSE1", cross = cross, addplot = TRUE, thr = 3)

Number of significant surrogate variables is:  14
Iteration (out of 5 ):1  2  3  4  5  Fitting 3 genes on chromosome 2

> causreg

      AMN1    YBR159W      CDC28
0.00000000 0.06312741 0.17088160
```

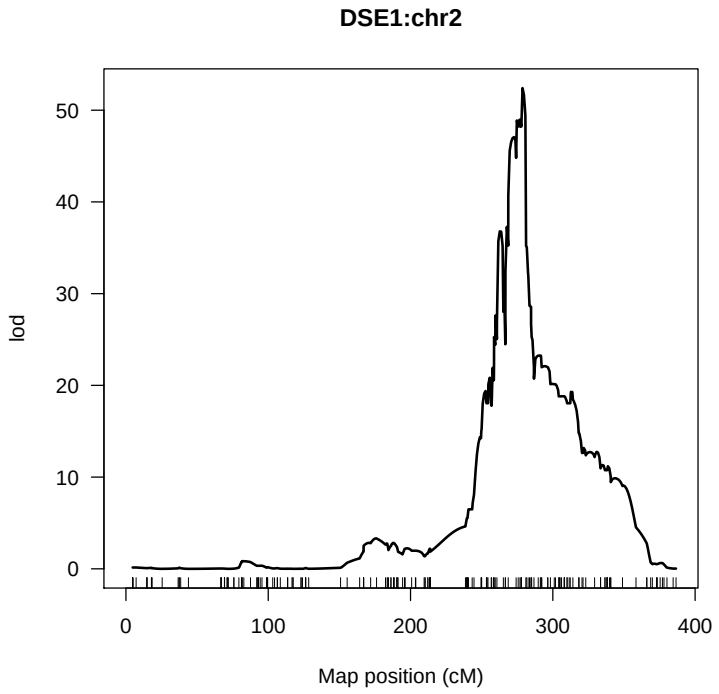

Figure 5: A linkage map for trait DSE1