Package 'eqtl'

August 22, 2008

R topics documented:

`The 'eqtl' package` *Tools for analyzing eQTL experiments: A complementary package to Karl Broman's R/qtl package for genome-wide analysis*

Description

Analyzis of experimental crosses to identify loci contributing to variation in quantitative traits (called quantitative trait loci, QTLs)

Version:	1.0
Date:	2008-03-05
Title:	Tools for analyzing eQTL experiments: A complementary package to Karl Broman's R/qtl package.
Author:	Hamid A Khalili ⟨hamid.khalili@gmail.com⟩and Olivier Loudet, INRA - Versailles
Maintainer:	Hamid A Khalili ⟨hamid.khalili@gmail.com⟩
Description:	Analysis of experimental crosses to identify loci (called quantitative trait loci, QTLs) contributing to variation in quantitative traits
License:	GPL version 2 or later

Author(s)

Hamid A Khalili, ⟨hamid.khalili@gmail.com⟩

References

Broman KW, Wu H, Sen S, Churchill GA (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

`A new starting point` *Introductory comments on R/eqtl*

Description

A brief introduction to the R/eqtl package, with a walk-through of a typical analysis.

Preliminaries to R/eqtl

- In order to use the R/eqtl package, you must type (within R) `library(qtl)` and `library(eqtl)`. You may want to include this in a `.Rprofile` file.
- Documentation and several tutorials are available from the R archive (`http://cran.r-project.org`).
- Use the `help.start` function to start the html version of the R help.
- Type `library(help=qtl)` to get a list of the functions in R/qtl.
- Type `library(help=eqtl)` to get a list of the functions in R/eqtl.
- Download the latest version of R/qtl and R/eqtl.

Walk-through of a typical analysis

Here we briefly describe the use of R/eqtl and R/qtl to analyze an experimental cross. R/eqtl is an add-on package to Karl Broman's R/qtl. It requires the 'qtl' package and uses some of its functions. Therefore this tutorial takes in consideration prior knowledge of R/qtl. You must read the R/qtl documentation and tutorial before to perform any analysis with the 'eqtl' add-on.

A difficult first step in the use of most data-analysis software is to import the data in adequate format. This step is perfectly described in R/qtl tutorial. With R/eqtl you should import some extra data in addition to the data needed for R/qtl. We won't discuss about data import at this point. This step is described in the next chapter 'First step'.

We consider the example data `seed10`, an experiment on gene expression in *Arabidopsis thaliana*. Use the `data` function to load the data.

```
data(seed10)
```

`seed10` data have class `cross` and `riself`. It describes an experiment on a RIL population obtained by single seed descent. The function `summary.cross` gives summary information on the data, and checks the data for internal consistency. A lot of utility functions are available in 'qtl' and are widely described in Karl's tutorial.

To project our results on the physical map, we also need to load the physical position of the genetic markers and the genomic physical coordinates of the probes used to estimate expression traits described in `seed10`. For information, `BSpgmap` and `ATH.coord` are simple data frame with specific column names.

```
data(BSpgmap)
names(BSpgmap)
data(ATH.coord)
names(ATH.coord)
```

Before running the QTL analysis, intermediate calculations need to be performed. The function `calc.genoprob` is used to compute the conditional probabilities at each pseudo-marker. `sim.geno` simulates sequences of genotypes from their joint probabilities. See 'qtl' manual for details. These steps have been already performed on seed10 and you may not need to run them again. Here, pseudo-markers have been defined every 0.5 centimorgan (`step=0.5`).

```
seed10 <- calc.genoprob(seed10, step=0.5, off.end=0, error.prob=0,
map.function='kosambi', stepwidth='fixed')
seed10 <- sim.geno(seed10, step=0.5, off.end=0, error.prob=0,
map.function='kosambi', stepwidth='fixed')
```

Use the `scanone` function to perform an interval mapping. `BaySha.em <- scanone(seed10,method='em',pheno.c`

The microarray probes usually contains data for which we don't want to perform any QTL analysis like the buffers, the controls or some missed probes. The function `clean.phe` cleans the `seed10` and/or the `BaySha.em` data for undesired phenotypes.

```
seed10.cleaned <- clean.phe(seed10,"Buffer")
seed10.cleaned <- clean.phe(seed10,"Ctrl")
BaySha.em <- clean.phe(BaySha.em,"Buffer")
BaySha.em <- clean.phe(BaySha.em,"Ctrl")
```

In this example, dropped data comes from probes named `"Buffer"` and `"Ctrl"` found within CATMA data. Note that one could a priori clean the `seed10` data before computing the interval mapping. The scanone object will be directly generated clean.

One of the major problematic step for genome-wide expression QTL analysis, is to read all the LOD curves and sytematically define the QTLs. Because of the amount of results, it is not feasible to read by eyes all the LOD curves. Use `define.peak` function to define QTL with drop LOD support interval from the scanone results, here the interval mapping results `BaySha.em`.

```
BaySha.peak <- define.peak(BaySha.em,locdolumn='all')
class(BaySha.peak)
```

The parameter `lodcolumn='all'` specifies to analyse all LOD columns (all the traits) of the scanone object `BaySha.em`. `lodcolumn='CATrck'` specifies to analyse the scanone LOD column `CATrck` only, which is supposed to be the interval mapping result of the trait `CATrck`.

We call `peak` object, the results of the `define.peak` function. The `peak` object is used to store the QTL definition. The QTL are defined by several features decribed in the `peak` objects attributes. At this step, a QTL is only defined by its LOD score, its location, the subjective quality of the LOD peak. See `define.peak` function for details.

```
attributes(BaySha.peak)
```

Back to the `define.peak` parameters. `graph=TRUE` specifies to draw the LOD curves with LOD support interval. The curves showing a QTL detected will be drawn on different charts for each chromosome. Note that, no graphical setup has been defined and therefore all graphs generated will appear one above the others. You should specify the graphical parameter `mfrow` of the R function `par()` before running `define.peak` to draw all charts in the same window. You may not want to set the parameters `graph=TRUE` and `lodcolumn='all'` at the same time, depending on the amount of traits analyzed.

The following command lines gives an example to define QTL and draw chart for a unique trait `CATrck`.

```
png(filename='CATrck.png',width=800,height=600)
par(mfrow=c(1,5))
define.peak(BaySha.im, lodcolumn='CATrck', graph=TRUE, chr=c(1,5))
par(mfrow=c(1,1))
dev.off();
```

`png()` and `dev.off()` are classical R functions which indicates here to print the graph generated as a png file `'CATrck.png'`. By using these functions, you can page set the graph as you wanted. Differently, the `define.peak` function parameter's, `save.pict=TRUE`, will systematically save all single LOD curves generated for each chromosome as png files. The files generated will be named with the names of the trait and the chromosomes where the QTLs are located. So beware to the amount of data you're analysing before setting the parameters `save.pict=TRUE`.

The way to access QTL results within `peak` object is quite simple:

```
BaySha.peak
BaySha.peak$CATrck
```

`BaySha.peak` will give you the `define.peak` results ordered by trait and chromosomes, respectively. `BaySha.peak$CATrck` will give you the results for the trait `'CATrck'` and so on for other trait names. If no QTL had been detected for a trait, the result will be the value `NA`.

To complete the QTL analysis, use the functions `calc.adef`, `localize.qtl` and `classify.qtl` to compute, for each QTL previously detected in `peak` object, the additive effect, the estimated physical location and the estimated acting-type in case of eQTL, respectively. All of these functions will add peak features to the `peak` object.

```
BaySha.peak <- localize.qtl(cross=seed10.cleaned, peak=BaySha.peak,
data.gmap=BSpgmap)
BaySha.peak <- calc.adef(cross=seed10.cleaned, scanone=BaySha.em,
peak=BaySha.peak)
BaySha.peak <- classify.qtl(cross=seed10.cleaned, peak=BaySha.peak,
etrait.coord=ATH.coord, data.gmap=BSpgmap)
attributes(BaySha?peak)
```

For each of these functions you have to specify the `peak` object. You also need to specify the related `cross` object and `scanone` results, the related genetic map physical data `BSpgmap` and the expression traits physical data `ATH.coord`. Note that, the expression trait physical data (here `ATH.coord`) may contain more traits than those studied. Conversely, all traits studied within the `peak`, the `scanone` or the `cross` objects must be described in `ATH.coord`.

Use `calc.Rsq` function to compute, from a `peak` object, the contribution of the individual QTLs to the phenotypic variation. At the same time this function tests and computes the contribution of significant epistatic interactions between QTLs. By default the significant threshold is set to `th=0.001`. In case you wanted to take all QTL interactions whatever the significance, you must set `th=1`.

```
BaySha.Rsq <- calc.Rsq(cross=seed10.cleaned,peak=BaySha.peak)
BaySha.Rsq
plot.Rsq(rsq=BaySha.Rsq)
```

The function `peak.2.array` will format all QTL results in a simple array. The column names are the names of the peak features described in `peak` object. This array have class `peak.array`. `Rsq.2.array` add the R square column to the QTL array. Formating the results as a simple array allows to use all basic and complex R functions (statistical, summary, graphical, histograms...) to study the results customly and in the simplest way. This format also allows to write the results in a file (like text or CSV) to save out the data.

```
BaySha.array <- peak.2.array(BaySha.peak);
BaySha.array <- Rsq.2.array(rsq=BaySha.Rsq,BaySha.array);
```

'eqtl' provides also useful functions to get an overview of the QTLs results stored in `peak.array`: The `summary.peak` function gives a variety of summary information and an overview of peak distribution. Summary graphs are available by setting `graph=TRUE`. Like `define.peak`, no graphical parameters had been setted and therefore all graphs generated will appear one above the others in the same R graph window. You may define `mfrow` before running `summary.peak` to draw all charts in the same R window.

Whole QTL summary with graphs:

```
par(mfrow=c(2,4))
BaySha.summary <- summary.peak(peak.2.array,seed10.cleaned,graph=TRUE)
par(mfrow=c(1,1))
names(BaySha.summary)
BaySha.summary
```

QTL summary with graphs excluding QTL localized on the chromosome 3 between 5000 and 6000 bp:

```
par(mfrow=c(2,4))
BaySha.sum_exc <- summary.peak( BaySha.array, seed10.cleaned,
exc=data.frame(inf=5000, sup=6000, chr=3), graph=TRUE)
par(mfrow=c(1,1))
names(BaySha.sum_exc)
BaySha.sum_exc
```

The function `plot.genome` provides basic informations and an overview about genome-wide eQTL parameters.

```
plot.genome(seed10.cleaned, BaySha.array, ATH.coord, BSpgmap, chr.size=c(30432457,
19704536, 23470536, 18584924, 26991304), save.pict=TRUE);
```

The parameter `chr.size` is the size of the chromosomes in base pair (here A. thaliana). These sizes are used to delimit the chromosomes for genome-wide graphs. For this function, the page setting has already been specified. `save.pict=TRUE` will save all graphs in different files within the current folder.

Use the function `cim.peak` to systematically perform a composite interval mapping by running a single genome scan `scanone` with previously defined QTL as additives covariates.

The additive covariates are defined from a `peak` object as the closest flanking marker of LOD peaks with the function `map.peak`. `cim.peak` returns an object of class `scanone` and therefore could be analyzed by the `define.peak` function. Then, the results can be analyzed by `calc.adef`, `localize.qtl`, `calc.Rsq`, etc... Due to the model, the LOD curve present a high (artefactual) LOD peak at the additive covariates locations which will be wrongly detected as a strong QTL by the function `define.peak`. To avoid that, use `wash.covar` function which will set the LOD score at the covariates location to 0 LOD. This function take care of a genetic window size which specifies the size of the region to ”wash”.

```
BaySha.cem <- cim.peak(seed10.cleaned,BaySha.peak)

covar <- map.peak(BaySha.peak)
covar

my_washed_BaySha.cem <- wash.covar(BaySha.cem, covar, window.size=20)
BayShacim.peak <- define.peak(BaySha.em, lodcolumn='all')
BayShacim.peak <- calc.adef(cross=seed10.cleaned, scanone=my_washed_BaySha.cem,
peak=BayShacim.peak)
BayShacim.peak <- localize(cross=seed10.cleaned, peak=BayShacim.peak,
data.gmap=BSpgmap)
BayShacim.peak <- classify(cross=seed10.cleaned, peak=BayShacim.peak,
etrait.coord=ATH.coord,data.gmap=BSpgmap)
BayShacim.Rsq <- calc.Rsq(cross=seed10.cleaned, peak=BayShacim.peak)
plot.Rsq(BayShacim.Rsq)
BaySha.cim.array <- peak.2.array(BayShacim.peak)
BaySha.cim.array <- Rsq.2.array(BayShacim.Rsq,BayShacim.array)
```

enjoy ;o)

Author(s)

Hamid A Khalili, ⟨hamid.khalili@gmail.com⟩

`First step` *Importing the data*

Description

R/eqtl needs to import some data in addition to those necessary for R/qtl package: the physical data of the genetic markers and the phycical coordinates of the probes used to measure the expression traits.

The physical data of the genetic map

This is a simple data frame with columns `"Marker"`,`"chr"` and `"PP"` as described for `BSpgmap` dataset. You can import these data by the way you wanted to obtain this data frame. I describes here one simple way to import it from a file:

- **The file format:**
  The first line contains the names of the columns coma separated. The following lines contains the informations needed (coma separated). This file could be created as CSV file from Excel or a simple text editor. Of course, these informations describes the map of the experiment stored as `cross` object (here `seed10`). The markers must appear in

the same order as the markers and chromosomes in the `cross` object (in the order of the map !).

Take a look on the sample file:

```
"Marker","chr","PP"
"MSAT100008",1,8639
"T1G11",1,1243250
"F21M12",1,3212191
"IND4992",1,4992444
"IND6375",1,6375557
"MSAT1.10",1,7296649
"MSAT108193",1,8192951
```

etc...

Take a look whitin R to the `cross` object:

```
seed10.cleaned$geno$'1'$map
seed10.cleaned$geno$'2'$map
seed10.cleaned$geno$'3'$map
```

etc...

- **The R command to import the data within R:**
  ```
  a_new_BSpgmap <- read.table("path to the file",header=TRUE,sep=",")
  ```

The coordinates of the expression traits

This is a simple data frame with columns `"etrait.name"`,`"chr"`, `"start"` and `"stop"` as described for `CATMA.coord`. By the same way than the map data importation, you can do by the way you wanted. The importation process is quite similar. Here the file can describe more expression trait than the phenotypes described in `cross` object. Of course all etraits described in `cross` object must have coordinates in the file.

- **The file format:**
  ```
  "etrait.name","chr","start","stop"
  "CATMA1A00010",1,4707,4972
  "CATMA1A00020",1,6442,6653
  "CATMA1C71002",1,7579,7791
  "CATMA1A00030",1,12268,12486
  "CATMA1A00035",1,30923,31142
  "CATMA1A00040",1,31232,31381
  "CATMA1A00045",1,33814,34211
  "CATMA1A00050",1,38785,38971
  ```
  etc...

  Take a look within R to the `cross` object:
  ```
  names(seed10.cleaned$phe)[1:10]
  ```

- **The R command to import the data within R:**
  ```
  new_CATMA.coord <- read.table('./data_Files/listGST-coord.tab', sep="\t",
  header= TRUE)
  ```

Author(s)

Hamid A khalili

`ATH.coord` *Data on probes coordinates*

Description

Data for the physical coordinates of A. thaliana GST (probes).

Usage

```
data(ATH.coord)
```

Format

A data frame with 30334 observations on the following 4 variables representing GST genomic coordinates:

`etrait.name` a factor with GST names as levels.

`chr` an integer vector determining the chromosomes.

`start` an integer vector determining the GST start location in base pair.

`stop` an integer vector determining the GST stop location in base pair.

Details

These Gene Sequence Tags are used on the CATMA microarray. See references below. They link to the expression phenotypes measured on their related cross object (here, they decribes the `seed10` object). Usually, all expression traits are not taken into account within a QTL analysis, therefore this list could contain more traits than the ones phenotyped within the cross object. On the other hand, every phenotype analysed must be found within `ATH.coord` data frame.

Source

Jean-Pierre Renou and Alain Lecharny (CATdb a Complete Arabidopsis Transcriptome database)(http://urgv.evry.inra.fr/CATdb)

References

Crowe M.L. et al., (2003) CATMA A complete Arabidopsis GST database. Nucleic Acids Res 31:156-158.

Examples

```
data(ATH.coord);
```

`BSpgmap` *Genetic map data of a RIL population*

Description

Genetic map physical data of an *Arabidopsis thaliana* Recombinant Inbred Line population

Usage

```
data(BSpgmap)
```

Format

A data frame with 69 observations on the following 3 variables representing the physical location of genetic markers:

`Marker` a factor with genetic marker names as levels.

`chr` a numeric vector determining the chromosomes.

`PP` a numeric vector determining the markers physical position on the chromosome in base pair (bp).

Details

Physical data of the 33RV population genetic map. This population was created from a Bay0 x Sha cross by Olivier Loudet and Sylvain Chaillou between 1997 and 2000 at INRA Versailles. For complete description of the population see reference below.

Source

Loudet (Genetics and Plant breeding, the VAST lab, INRA VERSAILLES) `http://www.inra.fr/vast/`

References

Loudet et al.(2002) Theoretical and Applied Genetics, vol 104, pp 1173-1184

Examples

```
data(BSpgmap);
```

`calc.adef` *Compute the additive effect at each QTL marker*

Description

Compute the additive effect at QTL marker by meaning the phenotypic value for each genotypic group.

Usage

```
calc.adef(cross, scanone, peak, round, ...)
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.
`scanone`	An object of class `scanone`. See 'qtl' package manual for `read.cross` function details.
`peak`	An object of class `peak`. See `define.peak` function for details.
`round`	An optional integer indicating the precision to be used for the additive effect value. See `round` function for details.
`...`	Additional arguments passed to the functions `plot` and `effectplot` when it is called.

Details

Use Karl Broman's `effectplot` function to mean the phenotype for each genotypic group defined at the QTL marker. The additive effect is computed as the difference between the phenotypical means of the two genotypic groups (homozygous). The parental reference allele is allele 2. By default, allele 1 is encoded as A and allele 2 as B, therefore the additive effect is **mean(B)-mean(A)** where **mean(A)** is the phenotypical mean of genotypic group **A** and **mean(B)** is the phenotypical mean of the genotypic group **B**.

Value

The input `peak` object is returned with component, `adef`, added to components of `peak$trait$chromosome` for each previously detected QTLs.

`additive.effect`	The additive effect value at the QTL marker

Note

It is necessary to previously perform the `sim.geno` function. It is not recommended to plot the allelic contribution by using the function `calc.adef`. It is preferable to use directly Karl Broman's `effect.plot` function (using the parameter `draw=TRUE`). See 'qtl' package manual for `effectplot` function details.

Author(s)

Hamid A. Khalili

References

Broman KW, Wu H, Sen S, Churchill GA (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

See Also

`effectplot`,`define.peak`,`read.cross`,`plot`

Examples

```
data(seed10);

# Genotype probabilities
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

# Genome scan and QTL detection
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak(out.em, 'all');

# Additive effect computing
out.peak <- calc.adef(seed10,out.em,out.peak,round=3);

# Additive effect of the QTLs affecting the 100th trait and localized on chromosome 1
out.peak[[26]]$'4'$additive.effect;

# Peak's features describing the QTLs affecting the 100th trait and localized on chromosome 1
out.peak[[26]]$'4';

# idem for the trait 'CATrck'
out.peak$CATrck
out.peak$CATrck$'4'
out.peak$CATrck$'4'$additive.effect
```

`calc.Rsq` *Estimate R square of individual QTLs and QTL interactions*

Description

Estimates the R square (phenotypic contribution) for individual QTLs and their significant interactions for each trait from cross and peak objects.

Usage

```
calc.Rsq(cross,peak,th=0.001,round)
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.
`peak`	An object of class `peak`. See `define.peak` for details.
`th`	A numeric vector of length 1 with value between 0 and 1. The R square significance threshold to keep the R square values.
`round`	An optional integer indicating the precision to be used for the R square value and it significance. The function 'round' is used for R square value `rsq`. The function 'signif' is used for the significance `pF` (i.e. p(F)). See `round` and `signif` functions for details.

Details

Estimate the proportion of the phenotypic variation explained by the segregation of an individual QTL or a significant QTL interactions (also called R square). Compute R square i.e. to compare the phenotypic variation explained by the presence of a genetic determinant (an individual QTL or a QTLs interaction) with the total phenotypic variation. Here we use an AnOVa with a linear model including all possible epistatic interactions.

Let a trait be affected by 3 QTLs localised at 3 markers **M1**, **M2** and **M3** respectively. The AnOVa is computed for the linear model:

$$M1 + M2 + M3 + M1 : M2 + M1 : M3 + M2 : M3 + M1 : M2 : M3$$

The R square for each genotypic group is the comparison of the variance due to the between-groups variability (called Mean Square Effect, or MSeffect) with the within- group variability (called Mean Square Error, or Mserror).

The significance of an epistatic interaction is the significance of the group effect of each interaction computed by the AnOVa. If one wants to store the results within a QTL database, it might be useful to compute all genetic determinants by setting `th=1` and then to extract the significant results by SQL queries.

Value

Return an object of class `rsq` which is a simple data frame with columns:

`qtl`	The name of the genetic determinant. If the genetic determinant is an individual QTL, the name is formated as `"trait_name.chr_number.order_number"`. In the case of epistasis, the genetic determinant name is formated as the list of individual genetic determinant (QTL) names separated by `":"`.
`rsq`	The R square value (set to NA if not significant: `pF < th`).
`pF`	The significance (set to NA if not significant: `pF < th`).

Author(s)

Hamid A. Khalili

See Also

`read.cross`, `define.peak`

Examples

```
data(seed10);

# Genotype probabilities
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

# Genome scan and QTL detection
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak(out.em, 'all');

# R square computing
out.rsq <- calc.Rsq(seed10,out.peak);
```

```
# R square computing without taking account of any significance
out.rsq <- calc.Rsq(seed10,out.peak,th=1);
```

`cim.peak` *Genome scan using previously detected QTLs as covariates*

Description

Use the LOD peaks previously detected in a `peak` object to define the additive covariates and perform a single genome scan taking cofactors into account.

Usage

```
cim.peak(cross,peak)
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.
`peak`	An object of class `peak`. See `define.peak` function for details.

Details

Perform a composite interval mapping using the `scanone` function with additive covariates previously defined in the related peak object. A scan is performed for traits which are affected by at least one QTL. The additive covariates for each trait are defined as the closest flanking marker to each significant LOD peak (defined in the peak feature `peak_cM`). Each trait scan generates a `scanone` object which is concatenated to the others `scanone` objects.

Value

Return an object of class `scanone`.

Author(s)

Hamid A. Khalili

References

Broman KW, Wu H, Sen S, Churchill GA (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

See Also

`define.peak`,`c.scanone`,`scanone`,`find.flanking`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak(out.em, 'all');
out.cem <- cim.peak(seed10,out.peak);
```

`classify.qtl` *Estimate the acting type of expression QTL*

Description

Estimate wether an eQTL is cis- or trans- acting.

Usage

```
classify.qtl(cross, peak, etrait.coord, data.gmap)
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.
`peak`	An object of class `peak`. See `define.peak` for details.
`etrait.coord`	A `data.frame` with column names `"etrait.name","chr","start","stop"` specifying the etrait (expression trait) location on the genome: `etrait.coord$array_element_name` is character strings vector specifying the name of the etraits. `etrait.coord$chr` is a vector of integers specifying the chromosome on which the markers are localized. `etrait.coord$start` is a vector of integers specifying the start of the etrait's sequence in base pair. `etrait.coord$stop` is a vector of integers specifying the stopo of the etrait's sequence in base pair.
`data.gmap`	A `data.frame` with column names `"Marker"`, `"chr"` and `"PP"` specifying the marker physical location. Those ones must be the same markers defined in the related `cross` object. `data.gmap$Marker` is a vector character strings specifying the names of markers. `data.gmap$chr` is a vector of integers specifying the chromosomes on which the markers are localized. `data.gmap$PP` is a vector of integers specifying the physical marker locations on the chromosomes in base pair.

Details

Useful in case of genome-wide expression QTL mapping. Determining cis-acting and trans-acting eQTL (or cis- and trans- eQTL) gives a basic overview about the global eQTL network. The (potential) cis-eQTL are those which colocalize with the controlled gene. These could be typically explained by a modification within gene promoter and therefore actually correspond to a cis-regulation (note that it would remain to be confirmed on a case by case basis: due to the lack of precision in QTLs localization for all analysis methods, a cis-acting is still biologically hypothetical; plus it could also correspond to a trans-acting eQTL localised close to its target gene). eQTLs which contains the regulated gene within their LOD support interval are classified in this category as `cis`. The trans-acting eQTLs are defined as those which do not colocalize with the affected gene. These could typically correspond to the mode of action of a transcription factor on the regulation of another gene's expression. eQTL which do not contain the regulated gene within their LOD support interval are classified as `trans`.

Value

The input `peak` object is returned with a component `type` added to the components of `names(peak$trait$chromosome)` for each previously detected QTL:

`type`	`cis` or `trans` for cis- and trans- eQTL respectively. `<NA>` if the etrait location is unknown or not nuclear.

Note

The QTL support interval locations are defined within a `peak` object. This classification (performed by `classify.qtl`) depends entirely on the support interval definition computed by the function `define.peak`. This function tend to underestimate cis-eQTL number as LOD-drop value are more conservative. It does not replace the scientist's eye on the LOD curve.

Author(s)

Hamid A. Khalili

See Also

`read.cross`,`define.peak`,`calc.adef`

Examples

```
data(seed10);

# Genotype probabilities
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');

# Genome scan and QTL detection
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak( out.em, 'all');

# Additive effect computing and peaks localization
out.peak <- calc.adef(seed10,out.em,out.peak);
```

```
data(BSpgmap);
out.peak <- localize.qtl(seed10,out.peak,BSpgmap);

# Estimated actind-type of the expression QTL affecting the 100th expression trait and
# localized on chromosome 1
data(ATH.coord)
out.peak <- classify.qtl(seed10,out.peak,ATH.coord,BSpgmap);
out.peak[[26]]$'4'$type;

# idem for the trait 'CATrck'
out.peak$CATrck$'4'$type;
```

cleanphe — *Remove undesired phenotypes and LOD results from cross and scanone object respectively*

Description

Drop the phenotypes or the LOD results within an object of class `cross` or `scanone` respectively. The names of the phenotypes and the lodcolumns to remove are defined by a character string or a regular expression.

Usage

```
cleanphe(x, string = "Buffer")
```

Arguments

`x`	An object of class `cross` or class `scanone` containing at least two phenotypes. See 'qtl' package manual for `read.cross` and `scanone` functions details.
`string`	The string which describes the names of the phenotypes or the results to remove. It can be defined as a regular expression or just the name of a column. See `grep` for details.

Details

This function is useful to systematically drop phenotypes like buffers or controls existing in microarray data or clean out the scanone results in context of expression QTL mapping. The names of the phenotypes and the results from objects of class `cross` and class `scanone` are column names which are defined by a single string or a regular expression specified by the argument `string`. The data to remove are searched by the `grep` function as follow:

`grep(string,names(x))` when x have class `scanone`.
`grep(string,names(x$pheno))` when x have class `cross`.

Value

Return the input cross or scanone object.

Author(s)

Hamid A. Khalili

See Also

`grep`,`scanone`,`read.cross`

Examples

```
data(seed10);

# Genotype probabilities and genome scan
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');

# Clean cross object and genome scan
seed10 <- cleanphe(seed10,'Buffer');
seed10 <- cleanphe(seed10,'ctrl');

out.em <- cleanphe(out.em,'Buffer');
out.em <- cleanphe(out.em,'ctrl');
```

`cover.peak` *List QTLs within a genetical region from a peak object*

Description

List QTLs which cover a given genetical region from peak object data.

Usage

```
cover.peak(peak,pos,chr,pre=0)
```

Arguments

`peak`	An object of class `peak`. See `define.peak` function for details.
`pos`	A single numeric value : the genetic position.
`chr`	A single integer value : the chromosome.
`pre`	A single numeric value : the precision of the targeted genetic position.

Details

This function searches QTL from peak object which cover a genetical region. The targeted genetic region is defined as a single genetic position `pos` around which the QTLs are searched; the size of this region is defined by `pre` which is the max distance from `pos` on which the QTLs are searched. `pre=0` will set to search QTLs which cover only the single genetic position `pos`. The QTLs are defined by LOD peaks with support interval in a peak object.

Value

return a data frame of class `peak.array`

Author(s)

Hamid A. Khalili

See Also

`grep`,`scanone`,`read.cross`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');

out.peak <- define.peak( out.em, 'all');

# return the list of QTL which colocalize at 4 cM on chromosome 3
my_peak <- cover.peak(out.peak,2,4,pre=0);
my_peak;

# return the list of QTL which colocalize on the genetic region 400-406 bp
# on chromosome 4
my_peak <- cover.peak(out.peak,pos=5,chr=4,pre=1);
my_peak;
```

`define.peak` *Define QTL with support interval and exclusionary window*

Description

Define QTL with LOD-drop support interval by using the results of a single QTL genome scan `scanone` and using a genetical exclusionary window.

Usage

```
define.peak(scanone,lodcolumn=1,chr=,th=2.3,si=1.5,graph=FALSE,window.size=20,round,save.pict=F
```

Arguments

`scanone`	An object of class `scanone`. See "qtl" package manual for `scanone` function details.
`lodcolumn`	Indicates on which of the LOD score columns (phenotypes) should QTLs be defined. This can be "`all`" which indicates all LOD score columns. This can also be a vector of integers indicating which of the columns should be used or a strings vector matching the names of the LOD columns, the phenotypes' name, to analyse. See "qtl" package manual for `scanone` function details.
`chr`	An optional vector indicating the chromosomes for which QTLs should be defined.
`th`	A single numeric value which sets the LOD score significance threshold. Only peaks with LOD score above this value will be analysed.

`si`	A single numeric value which sets the QTL's Support Interval. `si` is the value of the accepted drop of LOD score to estimate the likely region on which the QTL is localized.
`graph`	If `TRUE`, draws the LOD curve with LOD peaks and support interval for the detected QTLs.
`window.size`	The exclusionary window size: A single numeric value setting the minimum genetic distance between two distinct QTLs to be considered.
`save.pict`	If `TRUE`, save the LOD curves drawn with support interval as png files named like `"trait name"_"chromosome"_"a number".png` in the current folder.
`round`	An optional integer indicating the precision to be used for the LOD score. See `round` function for details.
`phe.name`	An optional character string specifying the name of the analysed trait. When performing `scanone` on a single trait, the lodcolumn is named 'lod' and as the analysed trait.
`...`	Passed to the functions `plot` and `plot.scanone` when they are called (if `graph=TRUE`). Passed the maximum size of the genomic region parameter: `m=10` should set 2*10cM for the inferior and the superior SI bounds from the position of the peak

Details

This function is used to detect and report QTL regions from a one-QTL genome scan performed by `scanone` function. A QTL is considered as a genomic region defined by a maximum LOD score peak value, its position and the position of its support interval (here called "SI"). The SI is estimated by the accepted drop of LOD score from the maximum LOD value defining the QTL region (the LOD peak). The FDR fells as the QTL SI size increases with lower LOD scores away from the peak. Usually we use `si=1.5` or `si=2`. A genetic exclusionary window sets the minimum distance between two distinct QTLs we consider being able to detect and depends directly on the size of the population. Due to the shape of the LOD curve, the drop of LOD score cannot be reached in some cases. Therefore a maximum SI size is set at 20cM by default. `m=10` will set 2*10cM for the inferior and the superior SI bounds. `graph=TRUE` specify to draw the LOD curves and the LOD SI on different chart for each QTL on their chromosome. No graphical setup has been defined and therefore they will be drawn one above the others in the same R graphical window. To setup the graph page and print all chart in same window, one may use the graphical parameter `mfrow` of the R function `par()` according the needs before launching `define.peak`. You may not want to set `graph=TRUE` and `lodcolumn="all"` at the same time depending on the amount of data. The parameter `save.pict` is useful to save systematically all charts generated by `define.peak`. These graphs are already page setted by the usual graphical functions (like `mfrow`).

Value

Return an object of class `peak` which is a list of components corresponding to traits. `names(peak)` contains the names of the traits. Each trait is itself a list with elements corresponding to chromosomes. For chromosomes on which no QTL have been detected, `peak$trait$chromosome` contains the `NA` value (where `chromosome` is the number of the chromosome). For those on which a QTL has been detected `peak$trait$chromosome` contains a data frame where rows are detected QTLs and columns are peak features (which describe QTLs). `names(peak$trait$chromosome)` contains the peak features:

`lod`	The peak's LOD score.
`mname.peak`	The maximum LOD peak's (pseudo-)marker name.
`peak.cM`	The maximum LOD peak's genetic position in centiMorgan (cM).
`mname.inf`	The (pseudo-)marker's name corresponding to the inferior si bound.
`inf.cM`	The genetic position of the inferior SI bound in centiMorgan (cM).
`mname.sup`	The (pseudo-)marker's name corresponding to the superior SI bound.
`sup.cM`	The genetic position of the superior SI bound in centiMorgan (cM).
`si.quality`	The subjective quality if the support interval. Due to the shape of the LOD curves and the methods used to define the LOD peaks, the subjective quality of the QTLs are various.

The subjective quality of the support interval

A QTL whose support interval could have been reached and defined, has more weight than a QTL whose support interval could not and has been defined by his maximum size (argument `m`). This quality information just correspond to symbols indicating, how were defined each bounds of the QTL support interval. The right symbols gives the information for the superior SI bounds and so on for the right bounds. '+' indicate that the LOD-drop support interval has been reached. '<-' and '->' indicates that the LOD-drop SI hasn't been reached before the maximum SI size (defined by `m` argument) for the inferior and the inferior bounds respectively. '|' indicates that the LOD-drop SI has been delimited by the beginning or the end of the LOD curve for the inferior or the superior bounds respectively. Therefore the quality symbols '|->' indicates that the SI has been delimited on the left by the beginning of the LOD curve and on the right by the maximum SI size. Therefore, the drop of LOD score is not reached neither on the left or on the right. '+|' indicates that the SI has been reached on the left but has been delimited on its right by the end of the LOD curve.

Author(s)

Hamid A. Khalili

References

Broman KW, Wu H, Sen S, Churchill GA (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890

See Also

`scanone`,`read.cross`

Examples

```
data(seed10);

# Genotype probabilities and genome scan
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');

################################################
```

```
# Detecting QTL with LOD drop support interval #
################################################

# Defining QTLs for all traits and saving the curves in png files.
out.peak <- define.peak(out.em, 'all',graph=TRUE,save.pict=TRUE,round=3);

# Defining QTLs for few traits and drawing the curves.
par(mfrow=c(1,5));
out.peak <- define.peak(out.em,lodcolumn=c(3,4,40,49),graph=TRUE,round=3);
par(mfrow=c(1,1));

# Defining QTLs for one trait and drawing the curves.
out.peak <- define.peak(out.em,lodcolumn='CATrck',graph=TRUE,round=3);
```

`drop.peakfeat` *Erase peak features in peak object*

Description

Erase chosen peak features informations from a `peak` object.

Usage

```
drop.peakfeat(peak, feat)
```

Arguments

`peak`	An object of class `peak`. See `define.peak` function for details.
`feat`	A character string vector containing the names of the features to delete. Features could be: `"additive.effect"`,`"peak.bp"`,`"inf.bp"`,`"sup.bp"` or `"type"`. See `calc.adef`, `localize.qtl`,`classify.qtl` functions for details.

Details

In `peak` object, QTL is defined by peak features. This function is useful to erase some peak features by avoiding to re-perform all the analyses (mainly the `define.peak` function). Only the peak features generated by the functions `calc.adef`, `localize.qtl` and `classify.qtl` should be removed. This function is used by the functions 'calc.adef', 'localize.qtl' and 'classify.qtl'.

Value

An object of class `peak`

Author(s)

Hamid A. Khalili

See Also

`define.peak`,`localize.qtl`,`calc.adef`,`classify.qtl`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em')
out.peak <- define.peak(out.em,lodcolumn='CATrck');
out.peak <- calc.adef(seed10,out.em,out.peak)

out.peak;

data(BSpgmap);
out.peak <- localize.qtl(seed10,out.peak,BSpgmap);

out.peak;

out.peak <- drop.peakfeat(out.peak,'additive.effect');
out.peak <- drop.peakfeat(out.peak,c('inf.bp','sup.bp'));

out.peak;
```

`eqtlversion` *Installed version of R/eqtl*

Description

Print the version number of the currently intalled version of R/eqtl

Usage

```
eqtlversion()
```

Value

A character string with the version number of the currently installed version of R/eqtl.

Author(s)

Hamid A Khalili, ⟨hamid.khalili@gmail.com⟩

See Also

`version`

genoplot — *Genome plot of the eQTL data on the expression traits locations*

Description

Plot the estimated eQTL positions with the genomic positions of the controlled gene.

Usage

```
genoplot( peak.array, cross, etrait.coord, data.gmap, chr.size, save.pict=FALSE, ...)
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.
`peak.array`	An object of class `peak.array`. See `peak.2.array` function for details.
`etrait.coord`	A data frame specifying the etrait genomic locations with colums: `etrait.name` a factor with array element or gene name as levels. `chr` an integer vector determining the chromosome. `start` an integer vector determining the GST start location in base pair. `stop` an integer vector determining the GST stop location in base pair.
`data.gmap`	A data frame with column names `"Marker"`, `"chr"` and `"PP"` specifying the marker physical locations. Those one must be the same markers defined in the related `cross` object. `data.gmap$Marker` is a vector character strings specifying the names of markers. `data.gmap$chr` is a vector of integers specifying the chromosome on which the markers are localized. `data.gmap$PP` is a vector of integers specifying the physical marker location on the chromosome in base pair.
`chr.size`	A vector of integer specifying the size of the chromosomes in base pair in order of the chromosomes.
`save.pict`	If TRUE, save each charts generated by `genoplot` as png files in the current folder.
`...`	Ignored at this step.

Details

Useful for genetical genomics studies. This function gives a graphical overview of the global eQTL network by plotting the estimated eQTL positions with the genomic positions of the affected traits. Six charts are generated and all locations data are represented on a physical scale. The genomic ditribution of the affected traits and the QTLs genomic distribution are described by two histograms. If `save.pict=TRUE`, these histograms are saved as `./histogram_controled_gst.png` and `./histogram_qtl.png` files, respectively. The etrait eQTL plot are represented with LOD color scale (from green to red in order of increasing LOD score) and with additive effect color scale (from green to red in order of increasing additive effect, yellow representing the null additive effect). Four etrait eQTL plot are generated representing the eQTL locations as single LOD peaks or support interval regions, both with LOD and additive effect color scales. If `save.pict=TRUE`, these plot are saved as `"lod_dotplot_traitxqtl.png"`, `"ae_dotplot_traitxqtl.png"`, `"lod_siplot_traitxqtl.png"` and `"ae_siplot_traitxqtl.png"` files.

Value

return a list with elements:

`coord_etrait`	the etrait coordinates.
`coord_qtl`	the QTL coordinates.
`limit`	the chromosomes limits.
`add_etrait`	the cumulates size of the chromosomes in bp for the etrait.
`add_qtl`	the cumulates size of of the chromosomes in bp for the QTL.

Author(s)

Hamid A. Khalili

See Also

`define.peak`,`read.cross`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak( out.em, 'all');
out.peak <- calc.adef(seed10,out.em,out.peak);

data(BSpgmap);
data(ATH.coord);

out.peak <- localize.qtl(seed10, out.peak, BSpgmap);
out.array <- peak.2.array(out.peak)

genoplot(out.array, seed10, ATH.coord, BSpgmap,
         chr.size=c(30432457,19704536,23470536,18584924,26991304), save.pict=TRUE);
# NB: the size of the Arabidopsis thaliana chromosomes are
# 30432457, 19704536, 23470536, 18584924 and 26991304 total base pairs
# for chromosomes 1 to 5 respectively
```

---

gpt *Global Permutation Threshold*

Description

Computes a Global Permutation Threshold to estimate a LOD score significance threshold.

Usage

```
gpt(cross, n_etrait=100, n_perm=1000)
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.
`n_etrait`	An integer which specifies the number of individuals on which the permutation test are performed.
`n_perm`	An integer. This argument defines the number of permutation replicates.

Details

Computes a Global Permutation Threshold which fits to a single-QTL scan (using `scanone` function) by permuting the phenotypes while maintaining the genotype for a sample of individuals randomly chosen within an object of class `cross`. The GPT estimates the LOD score significance threshold if 1000 permutations at least are computed on 100 individuals at least (i.e. 100,000 permutations).

Value

An object of class `scanoneperm`

Author(s)

Hamid A. Khalili

References

Churchill and Doerge (1994) Empirical threshold values for quantitative trait mapping. Genetics 138:963-971

See Also

`read.cross`,`scanone`,`add.threshold`

Examples

```
data(seed10);

# Genotype probabilities
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');

# Compute the global permutation test with 1000 permutations on 100 individuals
# out_1000.gpt <- gpt(seed10,100,1000);

# Compute the global permutation threshold with 100 permutations on 100 individuals
out_100.gpt <- gpt(seed10, 10, 10)

# Significance LOD threshold value with alpha at 0.05 (5
# th_1000 <- out_1000.gpt[order(out.gpt,decreasing=TRUE)][5000];
th_100 <- out_100.gpt[order(out_100.gpt,decreasing=TRUE)][50];

th_100;
mean(summary(out_100.gpt, alpha=0.05));
```

```
hist(out_100.gpt,nclass=50,col='gray')
abline(v=th_100,col='red')

# out.em <- scanone(seed10, method='em', chr=c(1:5));
# plot(out.em, chr=c(1:5));
# add.threshold(out.em, chr=c(1:5), perms=out_1000.gpt, alpha=0.05);
# add.threshold(out.em, chr=c(1:5), perms=out_1000.gpt, alpha=0.1, col="green");
```

`localize.qtl` *Compute QTL physical positions from QTL genetic positions*

Description

Compute QTL physical positions from QTL genetic positions from an object of class codepeak and the marker physical positions.

Usage

```
localize.qtl( cross, peak, data.gmap, round )
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.
`peak`	An object of class `peak`. See `define.peak` for details.
`data.gmap`	A `data.frame` with column names `"Marker"`, `"chr"` and `"PP"` specifying the marker physical locations. Those one must be the same markers described in the related `cross` object. `data.gmap$Marker` is a vector of character strings specifying the names of the markers. `data.gmap$chr` is a vector of integers specifying the chromosomes on which the markers are localized. `data.gmap$PP` is a vector of integers specifying the physical marker locations on the chromosomes in base pair.
`round`	An optional integer indicating the precision to be used for the physical position. The physical position being estimated, non integer nucleotidic position values could be obtained. See `round` function for details.

Details

Linearly compute the physical position from `peak$peak_cM` and the flanking marker locations:
A + B/C*D
A is the physical position of the first flanking marker. **B** and **C** are the genetic and the physical distances between the two flanking markers respectively. **D** is the genetic position of the qtl peak.

Value

The input `peak` object is returned with components added to components of `names(peak$trait$chromosome)` for each previously detected QTL:

`peak.bp`	is the physical location of the maximum LOD peak.
`inf.bp`	is the physical location of the SI lower bound.
`sup.bp`	is the physical location of the SI upper bound.

Author(s)

Hamid A. Khalili

See Also

`read.cross`,`define.peak`,`calc.adef`

Examples

```
data(seed10);

# Genotype probabilities
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

# Genome scan and QTL detection
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak(out.em, 'all');

# Additive effect computing
out.peak <- calc.adef(seed10,out.em,out.peak,round=3);

# Localizing peaks
data(BSpgmap);
out.peak <- localize.qtl( seed10, out.peak, BSpgmap, round=0);

# Peak features describing the QTLs affecting the 100th trait and
# localized on the chromosome 1
out.peak[[26]]$'4';

# Genetic and physical position of maximum LOD peaks affecting the 100th trait and
# localized on chromosome 1
out.peak[[26]]$'4'$peak.cM;
out.peak[[26]]$'4'$peak.bp;

# Genetic and physical position of QTLs' SI inferior bounds of the 100th trait and
# localized on chromosome 1
out.peak[[26]]$'4'$inf.cM;
out.peak[[26]]$'4'$inf.bp;

# Genetic and physical position of QTLs' SI superior bounds of the 100th trait and
# localized on chromosome 1
out.peak[[26]]$'4'$sup.cM;
out.peak[[26]]$'4'$sup.bp;
```

```
# idem for trait 'CATrck'
out.peak$CATrck$'4'$peak.cM;
out.peak$CATrck$'4'$peak.bp;
out.peak$CATrck$'4'$inf.cM;
out.peak$CATrck$'4'$inf.bp;
out.peak$CATrck$'4'$sup.cM;
out.peak$CATrck$'4'$sup.bp;
```

`map.peak` *Resume maximum LOD peak position from peak object*

Description

Resume all maximum LOD peaks position from `peak` object as a data frame. This function is useful for Composite Interval Mapping to define as co-factor previoulsy detected QTLs.

Usage

```
map.peak(peak)
```

Arguments

`peak`	An object of class `peak`. See `define.peak` function for details.

Details

Resume all detected QTLs location from `peak` object as a data frame. This function could be used by the function `wash.covar` and gives an overview of the covariates which can be used for a Composite Interval Mapping.

Value

Returns a data frame with columns:

`trait`	The names of the affected traits.
`chr`	The names of the chromosomes on which the QTL has been detected.
`cM`	The genetic position of the detected QTL.

Author(s)

Hamid A. Khalili

See Also

`define.peak`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
 map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
 map.function='kosambi', stepwidth='fixed');

out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak(out.em, 'all');

covar <- map.peak(out.peak);

covar;
```

`mnames.map` *List all markers from a cross object*

Description

Return the list of all markers for all of the chromosomes.

Usage

```
mnames.map(cross)
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.

Details

Return the list of all markers of the cross object sort by chromosome appearance and the marker relatives position. This function is used by the `cim.peak` function.

Value

A vector containing all marker names sort by the marker relatives position and chromosomes appearance.

Author(s)

Hamid A. Khalili

See Also

`cim.peak`,`read.cross`

Examples

```
data(seed10);
mnames.map(seed10);
```

`peak.2.array` *Build a simple array from a peak object*

Description

Build a simple array from a peak object.

Usage

```
peak.2.array(peak)
```

Arguments

`peak`	An object of class `peak`. See `define.peak` for function details.

Details

Useful for a genome-wide eQTL mapping. Formating the results as a simple array allow to use all classical R functions (graphical, statistical, summaries, ...) and permits all kind of manipulation for the results by the simplest way. All expression traits are represented and those which are not affected by any QTL, contain the empty data `<NA>` in each column.

Value

Return an object of class `array.peak` which is a data frame whith columns:

`trait`	The name of the studied traits.
`chr`	The number of the chromosome.
`mname.peak`	The peak (pseudo-)marker name when a QTL was detected. `<NA>` if no QTL was detected.
`lod`	The peak LOD score when a QTL was detected. `<NA>` if no QTL was detected.
`peak.cM`	The genetic position of the peak in centiMorgan(cM) when QTL was detected. `<NA>` if no QTL was detected.
`mname.inf`	The (pseudo-)marker name corresponding to the inferior SI bound when a QTL was detected. `<NA>` if no QTL was detected.
`inf.cM`	The genetic position of the inferior SI bound in centiMorgan(cM) when a QTL was detected. `<NA>` if no QTL was detected.
`mname.sup`	The (pseudo-)marker names corresponding to the superior SI bound when a QTL was detected. `<NA>` if no QTL was detected.
`sup.cM`	The genetic position of the superior SI bound in centiMorgan(cM) if a QTL was detected. `<NA>` if no QTL was detected.
`si.quality`	The subjective quality if the support interval. See '`define.peak`' for details.
`additive.effect`	The additive effects of the QTL. `<NA>` if no QTL has been detected.
`peak.bp`	The physical position of the maximum LOD peak. `<NA>` if no QTL was detected.
`inf.bp`	The physical position of the SI lower bound. `<NA>` if no QTL was detected.

`sup.bp`	The physical position of the SI upper bound. `<NA>` if no QTL was detected.
`type`	The estimated type of the eQTLs (trans or cis for cis- and trans- eQTL respectively). `<NA>` if no QTL was detected or in case of non nuclear expression trait.

Author(s)

Hamid A. Khalili

See Also

`scanone`,`read.cross`

Examples

```
data(seed10);

# Genotype probabilities and genome scan
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');

# Defining QTLs for all traits
out.peak <- define.peak( out.em, 'all',graph=TRUE,save.pict=TRUE);

# do not run
# out.array <- peak.2.array(out.peak);

# Computing additive effect
out.peak <- calc.adef(seed10,out.em,out.peak);

# Localizing peak
data(BSpgmap);
out.peak <- localize.qtl(seed10,out.peak,BSpgmap);

out.array <- peak.2.array(out.peak);
```

`peaksummary` *Print summary of QTL definition*

Description

Print summary information about QTL contained in a `peak` object.

Usage

```
peaksummary(peak.array,cross,exc=data.frame(inf=0,sup=0,chr=NA),graph=FALSE,...)
```

Arguments

`peak.array`	An object of class `peak.array`. See `peak.2.array` and `Rsq.2.array` functions for details.
`cross`	An object of class `cross`. See ”qtl” package manual for `read.cross` function details.
`exc`	A data frame with columns `inf`, `sup` and `chr` which represent a genomic region to exclude from the summary. `inf`, `sup`,`chr` represents the genomic location in base pair (start and stop of the sequence to exclude respectively), `chr` specify the chromosome. They are single numeric values.
`graph`	If TRUE, print summary graphs.
`...`	Ignored at this point.

Value

Return a list containing a variety of summary information about QTL distribution according to the `peak` features.

Note

No page setting has been specified in the `peaksummary` function therefore if `graph=TRUE` all graphs will appear one above the others within the same R graphical window. You should specified use the parameter `mfrow` of the R function `par()` to setup the graph page.

Author(s)

Hamid A. Khalili

See Also

`define.peak`,`read.cross`,`peak.2.array`,`Rsq.2.array`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');

out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak(out.em,'all');
out.peak <- calc.adef(seed10,out.em,out.peak);

data(BSpgmap);
out.peak <- localize.qtl(seed10,out.peak,BSpgmap);
out.array <- peak.2.array(out.peak);

# Whole QTL summary woth graph
par(mfrow=c(2,4));
peaksummary( out.array, seed10, graph=TRUE);
par(mfrow=c(1,1));

# QTL summary with graphs excluding the QTLs localized
# on chromosome 3 between 5000 and 6000 bp.
par(mfrow=c(2,4));
peaksummary( out.array, seed10, exc=data.frame(inf=5000,sup=6000,chr=3), graph=TRUE);
par(mfrow=c(1,1));
```

plotRsq *Plot R square data*

Description

Draw histograms of R square value distribution for `rsq` object.

Usage

```
plotRsq( rsq, par=c(2,2), ...)
```

Arguments

`rsq`	An object of class `rsq`. See `calc.Rsq` for function details.
`par`	A vector of two integers corresponding to the `mfrow` parameter of the `par()` function.
`...`	Passed to the function `hist` and `par` when they are called.

Details

Draw histograms of R square value distribution from an object of class `rsq`. Three histograms are drawn: the first one shows the R square value distribution of single QTLs. The second shows the distribution for QTL interactions. The last one shows all R square values distribution.

Value

none

Author(s)

Hamid A. Khalili

See Also

`calc.Rsq`,`peak.2.array`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
        map.function='kosambi', stepwidth='fixed');

out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak(out.em,'all');
out.rsq <- calc.Rsq(seed10,out.peak);

# plotting R quare data
plotRsq(out.rsq);
plotRsq(out.rsq,par=c(1,3));
```

`pseudo.map` *The makers and pseudo-markers genetic map*

Description

List the markers and pseudo-markers genetic positions for all of the chromosomes.

Usage

```
pseudo.map( cross )
```

Arguments

`cross`	An object of class `cross`. See 'qtl' package manual for `read.cross` function details.

Details

This function list the markers and pseudo-marker genetic positions for all of the chromosomes. This function is used by others functions.

Value

A vector containing containing the genetic position of markers and pseudo-marker for all of the chromosomes sort by positions and chromosomes appearance.

Note

It is necessary to previously perform the `calc.genoprob` function.

Author(s)

Hamid A. Khalili

See Also

`calc.genoprob`

Examples

```
data(seed10);

# Genotype probabilities
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

pseudo.map(seed10);
```

`Rsq.2.array` *Add R square data to peak.array data frame*

Description

Add the single QTL R square data to the related general QTL description contained within `peak.array` data frame.

Usage

```
Rsq.2.array(rsq,peak.array)
```

Arguments

`rsq`	An object of class`rsq`. See `calc.Rsq` for function details.
`peak.array`	An object of class `peak.array`. See `peak.2.array` for function details.

Details

Useful to store whole single QTL description within a simple array by adding the single QTL R square data. Add two columns containing the R square data from `rsq` object to the related `peak.array` data frame. Column `Rsq` contains the R square values and column `RpF` contains the R square significance. The R square data are computed by the function `calc.Rsq`.

Value

Return an object of class `rsq` which is a simple data frame with columns:

`qtl`	The name of the genetic determinant. If the genetic determinant is an individual QTL, the name is formated as `'trait_name'.'chr_number'.'a number'` . In the case of interactives QTL, the genetic determinant name is formated as the list of individual genetic determinant names separated by '`:`'.
`rsq`	The Fisher value (set to NA if not significant: `pF < th`).
`pF`	The significance (set to NA if not significant: `pF < th`).

Author(s)

Hamid A. Khalili

See Also

`calc.Rsq`,`peak.2.array`

Examples

```
data(seed10);

# Genotype probabilities
seed10 <- calc.genoprob( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=2, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

# Genome scan and QTL detection
out.em <- scanone( seed10, pheno.col=1:50, model='normal', method='em');
out.peak <- define.peak( out.em, 'all');

# Computing additive effect
out.peak <- calc.adef(seed10,out.em,out.peak);

# Localizing peak
data(BSpgmap);
out.peak <- localize.qtl(seed10,out.peak,BSpgmap);
out.array <- peak.2.array(out.peak);

# R square computing
out.rsq <- calc.Rsq(seed10,out.peak);

# Adding R square data
out.array <- Rsq.2.array(out.rsq,out.array);
```

`seed10` *Data on gene expression level variation*

Description

Data from an experiment on the expression level variation in the *A.thaliana* seed at 10 day after pollination.

Usage

```
data(seed10)
```

Format

An object of class `cross` and `riself`.

Details

There are 420 RIL individuals typed at 69 markers and 160 individuals have been retained for phenotyping. The population is the 33RV Versailles RIL population Bay x Sha. See references below. The phenotype is a sample of 500 Gene Sequence Tags hybridization signals measured on the CATMA microarray. See references below.

Source

Jean-Pierre Renou and Alain Lecharny (CATdb a Complete Arabidopsis Transcriptome database) `http://urgv.evry.inra.fr/CATdb`
Loudet (Genetic and Plant breeding, the VAST lab, INRA VERSAILLES)
`http://dbsgap.versailles.inra.fr/vnat/Documentation/33/DOC.html`

References

Crowe M.L. et al., (2003) CATMA, A complete Arabidopsis GST database. Nucleic Acids Res 31:156-158.
Loudet et al.(2002) Theoretical and Applied Genetics, vol 104, pp 1173-1184

Examples

```
data(seed10)
```

`wash.covar` *Erase additive covariates LOD peaks on the LOD curve*

Description

Sets LOD curve to 0 for a given region size around cofactors included in CIM.

Usage

```
wash.covar(scanone,covar,window.size=20)
```

Arguments

`scanone`	An object of class `scanone`. See 'qtl' package manual for `scanone` function details.
`covar`	A data frame with columns `'trait'`, `'chr'` and `'cM'`. `covar$trait` is a character strings vector which specifies the names of the traits. `covar$chr` is an integer vector which specifies the number of the chromosome. `covar$cM` is a numeric vector which specifies the cofactor position in cM.
`window.size`	a single numeric value which specifies the size of the region to set at zero LOD.

Details

This function is useful to extract the new QTLs from composite interval mapping results. The artifactual LOD peak value obtained from the cofactors are set at zero LOD. Then the QTLs are defined by using the function `define.peak` . The cofactors loci are defined in a data frame which can be performed by the function `map.peak`. In this case, the cofactors will be at the maximum LOD peak location defined within the related peak object.

Value

The input `scanone` object is returned.

Author(s)

Hamid A. Khalili

See Also

`scanone`, `cim.peak`, `map.peak`

Examples

```
data(seed10);

seed10 <- calc.genoprob( cross=seed10, step=3, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');
seed10 <- sim.geno( cross=seed10, step=3, off.end=0, error.prob=0,
         map.function='kosambi', stepwidth='fixed');

out.em <- scanone( seed10, pheno.col=1:5, model='normal', method='em');
out.peak <- define.peak( out.em, 'all');

covar <- map.peak(out.peak)

out.cem <- cim.peak(seed10,out.peak);
out.cem <- wash.covar(out.cem,covar);

out_composite.peak <- define.peak(out.cem,'all');
```

Index