

# Package ‘bqtl’

August 4, 2010

**Version** 1.0-26

**Date** 2010-08-04

**Title** Bayesian QTL mapping toolkit

**Author** Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**Maintainer** Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**Description** QTL mapping toolkit for inbred crosses and recombinant inbred lines. Includes maximum likelihood and Bayesian tools.

**Depends** R (>= 2.6.0)

**License** GPL (>= 2)

**URL** <http://famprevmed.ucsd.edu:16080/faculty/cberry/bqtl/>

**Repository** CRAN

**Date/Publication** 2010-08-04 21:39:17

## R topics documented:

A Starting Point . . . . .	2
adjust.linear.bayes . . . . .	3
bqtl . . . . .	4
bqtl-internal . . . . .	6
bqtl.fitter . . . . .	7
coef.bqtl . . . . .	8
configs . . . . .	9
covar . . . . .	10
formula.bqtl . . . . .	12
lapadj . . . . .	12
linear.bayes . . . . .	14
little.ana.bc . . . . .	16
little.ana.f2 . . . . .	17
little.bc.markers . . . . .	17

little.bc.pheno . . . . .	19
little.f2.markers . . . . .	19
little.f2.pheno . . . . .	21
little.map.dx . . . . .	21
little.map.frame . . . . .	22
little.mf.5 . . . . .	22
locus . . . . .	23
loglik . . . . .	25
make.analysis.obj . . . . .	26
make.loc.right . . . . .	28
make.location.prior . . . . .	29
make.map.frame . . . . .	30
make.marker.numeric . . . . .	31
make.regressor.matrix . . . . .	32
make.state.matrix . . . . .	33
make.varcov . . . . .	34
map.index . . . . .	35
map.location . . . . .	36
map.names . . . . .	37
marker.fill . . . . .	38
marker.levels . . . . .	39
plot.map.frame . . . . .	40
predict.bqtl . . . . .	42
predict.linear.bayes . . . . .	43
residuals.bqtl . . . . .	44
summary.adj . . . . .	45
summary.bqtl . . . . .	47
summary.map.frame . . . . .	48
summary.swap . . . . .	48
swap . . . . .	49
swapbc1 . . . . .	51
swapf2 . . . . .	53
twohk . . . . .	55
twohkbc1 . . . . .	56
varcov . . . . .	58

<b>Index</b>	<b>60</b>
--------------	-----------

## Description

Some pointers to a few key functions in *BQTL*

## New to R?

- Be sure to check out all of the free documentation that comes with R.
- The `example` function is very helpful in getting familiar with a new function. You type `example(fun)` and the examples in the documentation for `fun` are run, then you can read the documentation to get a better sense of what is really going on. My personal favorite is to type `par(ask=T)`, hit the 'enter' key, then `example(image)`, and 'enter' again; after each display you hit the 'enter' key to get to the next one.
- `library(bqtl)` is needed to load the *BQTL* functions and data sets.

## Key Functions

### Data Input \

`make.map.frame` defines the map,

`marker.levels` The help page describes several functions that define the coding scheme for marker levels,

`make.analysis.obj` combines marker data, phenotype data, and the `map.frame` to create an object that can be used by data analysis functions.

### Maximum Likelihood Methods \

`bqtl` does a host of things from marker regression and interval mapping to full maximum likelihood. The best way to get started is to run `example(bqtl)` and take a look at the resulting output.

`locus` is very helpful in specification of runs.

### Approximate Bayesian Analysis \

`linear.bayes` For a good starting point try `example(linear.bayes)`

## Author(s)

Charles C. Berry <cberry@ucsd.edu>

---

adjust.linear.bayes

*Use Laplace Approximations to improve linear approximations to the posterior*

---

## Description

The approximation provided by `linear.bayes` can be improved by performing Laplace approximations. This function is a development version of a wrapper to do that for all of the returned by `linear.bayes`.

## Usage

```
adjust.linear.bayes(lbo, ana.obj=lbo$call$ana.obj, ...)
```

**Arguments**

<code>lbo</code>	The object returned by <code>linear.bayes</code>
<code>ana.obj</code>	The <code>analysis.object</code> used to create <code>lbo</code> . This need not be given explicitly, iff the original version is in the search path.
<code>...</code>	currently unused

**Value**

A list of class "`adjust.linear.bayes`" containing:

<code>odds</code>	A vector, typically of length <code>k</code> giving the odds for models of size 1, 2, ..., <code>k</code> under a uniform posterior relative to a model with no genes.
<code>loc.posterior</code>	The marginal posterior probabilities by locus
<code>coefficients</code>	The marginal posterior means of the coefficients
<code>one.gene.adj</code>	Results of fits for one gene models
<code>n.gene.adj</code>	Results of fits for modles with more than one gene
<code>call</code>	the call to <code>adjust.linear.bayes</code>

**Note**

For large `linear.bayes` objects involving many gene models, this can require a very long time to run.

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**See Also**

[linear.bayes](#)

---

bqt1

*Bayesian QTL Model Fitting*

---

**Description**

Find maximum likelihood estimate(s) or posterior mode(s) for QTL model(s). Use Laplace approximation to determine the posterior mass associated with the model(s).

**Usage**

```
bqt1(reg.formula, ana.obj, ...)
```

**Usage**

```
bqtl(reg.formula, ana.obj, scope, expand.specials, ...)
```

**Arguments**

`reg.formula` A formula.object like `y ~ add.PVV4 * add.H15C12` . The names of the independent variables on the right hand side of the formula are the names of loci or the names of additive and dominance terms associated with loci. In addition, one can use `locus` or `configs` terms to specify one or a collection of terms in a shorthand notation. See [locus](#) for more details. The left hand side is the name of a trait variable stored in the search path, as a column of the data frame `data`, or `y` if the phenotype variable in `ana.obj` is used.

`ana.obj` The result of `make.analysis.obj` .

`...` Arguments to pass to `lapadj`, e.g. `rparm` and `return.hess`

**Details**

This function is a wrapper for `lapadj`. It does a lot of useful packaging through the `configs` terms. If there is no `configs` term, then the result is simply the output of `lapadj` with the `call` attribute replaced by the call to `bqtl`

**Value**

The result(s) of calling `lapadj`. If `configs` is used in the `reg.formula`, then the result is a list with one element for each formula. Each element is the value returned by `lapadj`

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**References**

Tierney L. and Kadane J.B. (1986) Accurate Approximations for Posterior Moments and Marginal Densities. *JASA*, **81**,82–86.

**See Also**

[locus](#), [configs](#), [lapadj](#)

**Examples**

```
data(little.ana.bc ) # load BC1 dataset

loglik( bqtl( bc.phenotype ~ 1, little.ana.bc ) ) #null loglikelihood
#on chr 1 near cM 25
loglik(bqtl(bc.phenotype~locus(chromo=1,cM=25),little.ana.bc))

little.bqtl <- # two genes with epistasis
  bqtl(bc.phenotype ~ m.12 * m.24, little.ana.bc)
```

```

summary(little.bqtl)

several.epi <-                                     # 20 epistatic models
  bqtl( bc.phenotype ~ m.12 * locus(31:50), little.ana.bc)
several.main <-                                    # main effects only
  bqtl( bc.phenotype ~ m.12 + locus(31:50), little.ana.bc)

max.loglik <- max( loglik(several.epi) - loglik(several.main) )

round(
  c( Chi.Square=2*max.loglik, df=1, p.value=1-pchisq(2*max.loglik,1))
  ,2)

five.gene <-                                       ## a five gene model
  bqtl( bc.phenotype ~ locus( 12, 32, 44, 22, 76 ), little.ana.bc , return.hess=TRUE )

regr.coef.table <- summary(five.gene)$coefficients

round( regr.coef.table[, "Value"] + # coefs inside 95% CI
  qnorm(0.025) * regr.coef.table[, "Std.Err"] %o%
  c("Lower CI"=1, "Estimate"=0, "Upper CI"=-1), 3)

```

---

bqtl-internal      *Internal BQTL functions*

---

## Description

Internal bqtl functions and objects

## Usage

```

x %equiv% y
map.dx(lambda, theta, min.lambda)
rhs.bqtl(reg.terms, ana.obj, bqtl.specials, local.covar, scope,
  expand.specials = NULL, method, ...)
zero.dup(x, dig=6)
uniq.config(swap.obj)

```

## Arguments

lambda	(2*(recomb fraction-1/2)
theta	recomb fraction
min.lambda	smallest map distance to use
reg.terms	a formula

ana.obj	an analysis.object
bqtl.specials	a vector of acceptable special names
local.covar	a function
scope	vector of strings
expand.specials	logical, whether to use expand.grid on the loci
method	e'g' "F2", "BC!", etc
...	not sure
swap.obj	result of swap
x	numeric vector or matrix
y	numeric vector or matrix
dig	how many significant digits to use

### Details

These are not to be called by the user.

---

bqtl.fitter	<i>Get loglikelihoods for many models of a common form</i>
-------------	--

---

### Description

For a single type of model, this function evaluates multiple models that differ only in terms of the loci involved. The looping is all done by internal C functions, so this is faster than simply using `bqtl` to do the same thing.

### Usage

```
bqtl.fitter(setup, loc.mat, ana.obj)
```

### Arguments

setup	The object returned by <code>bqtl( &lt;...&gt; , setup=TRUE )</code>
loc.mat	A matrix of locus numbers, s.t. <code>nrow(loc.mat)</code> equals the number of loci in <code>setup</code>
ana.obj	An <code>analysis.object</code> . Usually the one used in <code>setup</code>

### Details

In order to avoid the computational overhead of running large loops of very repetitive operations in R/S, `bqtl.fitter` used after the `setup=TRUE` option in `bqtl` will loop through the loci specified in `loc.mat` using internal C code. This is many times faster than running the same code via `bqtl`.

**Value**

For now it only returns the loglikelihood. But it would be trivial to build an option that would allow other quantities computed to be returned, and this should probably be done. However, some care is needed to keep objects from becoming unmanageably large.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[bqtl](#)

**Examples**

```
data( little.ana.bc )
little.setup <-
bqtl( bc.phenotype~locus(1)*locus(2), little.ana.bc, setup=TRUE )
combos <- t( as.matrix( expand.grid( 1:21, 44:64 ) ) )
little.update <- bqtl.fitter(little.setup, combos, little.ana.bc)
little.res <- matrix( little.update, nr=21 )
image( 1:21, 44:64, little.res )
rm(little.ana.bc, little.update, little.res )
```

---

coef.bqtl

*Extract Coefficients from fitted objects*

---

**Description**

Return a vector or matrix of coefficients as appropriate

**Usage**

```
coef.bqtl(object, ...)
```

**Usage**

```
## S3 method for class 'bqtl':
coef(object)
```

**Arguments**

object            The object returned by bqtl.

**Value**

A vector (if bqtl returned a single fit) or matrix (if bqtl returned a list with more than one fit)

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[bqt1](#)

---

configs

*Lookup loci or effects for genetic model formulas*

---

**Description**

Convert numeric indexes to names of regressors for a genetic model. One or many genetic models can be specified through the use of this function. It is used on the right hand side of a formula in the [bqt1](#) function.

**Usage**

```
configs(x, ..., scope, method=NULL)
```

**Usage**

```
configs(x, ..., scope=<<see below>>, method = NULL)
```

**Arguments**

x	Typically an integer, an integer vector, an array, or a list with a <code>configs</code> component such as returned by <a href="#">swapbc1</a> . However, it can also be a character string, vector, et cetera, in which case the elements must belong to <code>names(scope)</code>
...	Optional arguments to be used when <code>is.atomic(x)</code> is TRUE.
scope	(Optional and) Usually not supplied by the user. Rather <a href="#">bqt1</a> fills this in automatically. A vector of regressor names, like the <code>reg.names</code> component returned by <a href="#">make.analysis.obj</a> . When <code>mode(x)</code> is "character", then <code>names(scope)</code> <b>must be non-NULL</b>
method	(Optional and) Usually not supplied by the user. A method like "F2". Typically, this is determined by internal code.

## Details

`configs` is used in the model formula notation of `bqtl`, possibly more than once, and possibly with regressors named in the usual manner. `configs` is intended to speed up the specification and examination of genetic models by allowing many models to be specified in a shorthand notation in a single model formula. The names of genetic loci can consist of marker names, names that encode chromosome number and location, or other shorthand notations. The names of terms in genetic models will typically include the names of the locus and may prepend "add." or "dom." or similar abbreviations for the 'additive' and 'dominance' terms associated with the locus.

When used as in `bqtl(y ~ configs(34), my.analysis.obj)`, it will look up the term `my.analysis.obj$reg.names[34]`. When this is passed back to `bqtl`, it get pasted into the formula and is subsequently processed to yield the fit for a one gene model.

When used as in `bqtl(y ~ configs(34,75,172), my.analysis.obj)` it looks up each term and returns a result to `bqtl` that results in fitting a 3 gene model (without interaction terms).

When `x` is a vector, array, or list, the processing typically returns pieces of many model formulas. `bqtl(y ~ configs(26:75), ...)` results in a list of 50 different one gene model fits from `bqtl` for the terms corresponding to the 26th through the 75th variables. `bqtl(y ~ configs(cbind(c(15,45,192),c(16,46,193))), ...)` returns two four gene models. And more generally, whenever `is.array(x)` is TRUE, the columns (or slices) specify `dim(x)[1]/length(x)` different models. When `x$configs` is an array, this also happens. This turns out to be useful when the result of running `swapbc1` or `swapf2` is treated as an importance sample. In such a case, `bqtl(y ~ configs(my.swap), my.analysis.obj)` will return a list in which element `i` is the `i`th sample drawn when `my.swap <- swapbc1(...)` was run.

## Value

A character vector whose element(s) can be parsed as the right hand side of a model formula.

## Author(s)

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

## See Also

`bqtl` and the examples there for a sense of how to use `configs`, `make.analysis.obj` for the setup that encodes the marker map and the marker information, `swapbc1` and `swapf2` for generating samples to be screened by `bqtl`.

---

covar

*Treat locus as covariate*

---

## Description

Sometimes it helps speed computations to linearize the likelihood or at least a part of it w.r.t. the locus allele values. Both 'Haley-Knott regression' and 'composite interval mapping' use this approach. `covar` provides a mechanism for creating formula objects that specify such linearizations.

**Usage**

```
covar(x)
```

**Usage**

```
covar(x, ..., scope=<<see below>>, method=<<see below>>)
```

**Arguments**

<code>x</code>	The name of a locus (except for F2 designs, when it is the name of an effect like 'add.m.32') or any argument of the sort that <code>locus</code> allows.
<code>...</code>	If <code>x</code> evaluates to a single value, then additional atomic elements may be included as with <code>locus</code> .
<code>scope</code>	Not supplied by the user. see <code>locus</code>
<code>method</code>	Not supplied by the user. see <code>locus</code>

**Details**

The function `covar` actually only returns `x`. The real work is done by a `covar` function that is hidden inside of `bqtl`, where the arguments are parsed as for `locus`. Each of the return values from `locus` is prefixed by "covar(" and suffixed by ")". If `x` is a name of a locus or effect, then `paste("covar(", deparse(x), ")")` is returned. Later, when `bqtl` calls `lapadj`, terms like `covar(PVV4.1)` are recognized as requiring a linearization w.r.t. effect 'PVV4.1'.

**Value**

a character string or vector

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**References**

- HALEY, C. S. and S. A. KNOTT, 1992 A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity* 69:315-324.
- Knapp SJ, Bridges WC, and Birkes D. Mapping quantitative trait loci using molecular marker linkage maps. *Theoretical and Applied Genetics* 79: 583-592, 1990.
- ZENG, Z.-B., 1994 Precision mapping of quantitative trait loci. *Genetics* 136:1457-1468

**See Also**

[locus](#), [add](#), [dom](#), [configs](#)

---

<code>formula.bqt1</code>	<i>Extract formula from bqt1 object</i>
---------------------------	---

---

**Description**

formula method for class `bqt1`

**Usage**

```
## S3 method for class 'bqt1':
formula(x, ...)
```

**Arguments**

<code>x</code>	The object returned by <code>bqt1</code>
<code>...</code>	unused

**Value**

a formula object

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**See Also**

[bqt1](#)

---

<code>lapadj</code>	<i>Approximate marginal posterior for chosen model</i>
---------------------	--

---

**Description**

`lapadj` provides the Laplace approximation to the marginal posterior (over coefficients and dispersion parameter) for a given genetical model for a quantitative trait. A by-product is the parameter value corresponding to the maximum posterior or likelihood.

**Usage**

```
lapadj(reg.formula, ana.obj,
       rparm = NULL, tol = 1e-10,
return.hess = FALSE, mode.names = NULL, mode.mat = NULL,
       maxit = 100, nem = 1, setup.only=FALSE, subset=NULL, casewt=NULL,
       start.parm=NULL, ...)
```

**Usage**

```
lapadj(reg.formula, ana.obj,
       rparm = NULL, tol = 1e-10,
       return.hess = FALSE, mode.names = NULL, mode.mat = NULL,
       maxit = 100, nem = 1, setup.only=FALSE, subset=NULL, casewt=NULL,
       start.parm=NULL, ...)
```

**Arguments**

<code>reg.formula</code>	A formula, like <code>y~add.X.3+dom.X.3+add.x.45*add.x.72</code>
<code>ana.obj</code>	See <code>make.analysis.obj</code> , which returns objects like this
<code>rparm</code>	One of the following: A scalar that will be used as the ridge parameter for all design terms except for the intercept ridge parameter which is set to zero A vector who named elements can be matched by the design term names returned in <code>\$reg.vec</code> . If no term named "intercept" is provided, <code>rparm["intercept"]</code> will be set to zero. A vector with $(q-1) * k$ elements (this works when there are no interactions specified). If names are provided, these will be used for matching. Positive entries are 'ridge' parameters or variance ratios in a Bayesian prior for the regression coefficients. Larger values imply more shrinkage or a more concentrated prior for the regression coefficients.
<code>tol</code>	Iteration control parameter
<code>return.hess</code>	Logical, include the Hessian in the output?
<code>mode.names</code>	names to use as <code>dimnames(mode.mat)[[2]]</code>
<code>mode.mat</code>	Not usually set by the user. A matrix which indicates the values of regressor variables corresponding to the allele states. If <code>mode.mat</code> is not given by the user, <code>ana.obj\$mode.mat</code> is used.
<code>maxit</code>	Maximum Number of iterations to perform
<code>nem</code>	Number of EM iterations to use in reinitializing the pseudo-Hessian
<code>setup.only</code>	If TRUE, do not run. Return an object that can be use for a direct call to <code>.C</code>
<code>subset</code>	expression to evaluate using <code>ana.obj\$data</code> as the environment
<code>casewt</code>	a vector of non-negative weights
<code>start.parm</code>	Vector of starting values for the maximization
<code>...</code>	other objects needed in fitting

**Details**

The core of this function is a quasi-Newton optimizer due to Minami (1993) that has a computational burden that is only a bit more than the EM algorithm, but features fast convergence. This is used to find the mode of the posterior. Once this is in hand, one can find the Laplace approximation to the marginal likelihood. In addition, some useful quantities are provided that help in estimating the marginal posterior over groups of models.

**Value**

A list with components to be used in constructing approximations to the marginal posterior or a list that can be used to call the underlying C code directly. In the former case, these are:

adj	The ratio of the laplace approximation to the posterior for the correct likelihood to the laplace approximation to the posterior for the linearized likelihood
logpost	The logarithm of the posterior or likelihood at the mode
parm	the location of the mode
posterior	The laplace approximation of the marginal posterior for the exact likelihood
hk.approx	Laplace approximation to the linearized likelihood
hk.exact	Exact marginal posterior for the linearized likelihood
reg.vec	A vector of the variables used
rparm	Values of ridge parameters used in this problem.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**References**

Berry C.C.(1998) Computationally Efficient Bayesian QTL Mapping in Experimental Crosses. *ASA Proceedings of the Biometrics Section*. 164–169.

Minami M. (1993) Variance estimation for simultaneous response growth curve models. Thesis (Ph. D.)—University of California, San Diego, Department of Mathematics.

---

linear.bayes

*Bayesian QTL mapping via Linearized Likelihood*

---

**Description**

The Bayesian QTL models via a likelihood that is linearized w.r.t. a fixed genetic model. By default, all one and two gene models (without epistasis) are fitted and a MCMC sampler is used to fit 3,4, and 5 gene and (optionally) larger models.

**Usage**

```
linear.bayes(x, ana.obj, partial = NULL, rparm, specs, scope,
            subset, casewt, ...)
```

**Usage**

```
linear.bayes(x, ana.obj, partial=NULL, rparm=<<see below>>, specs=<<see below>>,
            scope=<<see below>>, subset=<<see below>>, casewt=<<see below>>, ...)
```

**Arguments**

<code>x</code>	a formula giving the QTL and the candidate loci or a <code>varcov</code> object
<code>ana.obj</code>	An analysis.object, see <a href="#">make.analysis.obj</a>
<code>partial</code>	a formula giving covariates to be controlled
<code>rparm</code>	A ridge parameter. A value of 1 is suggested, but the default is 0.
<code>specs</code>	An optional list with components <code>gene.number</code> (to indicate the model sizes), <code>burn.in</code> (to indicate the number of initial MCMC cycles to discard), and <code>n.cycles</code> (to indicate how many MCMC cycles to perform for each model size). If no values are supplied, <code>specs</code> defaults to <code>list(gene.number=c(1,2,3,4,5),burn.in=1,n.cycles=c(0,0,200,100,100))</code>
<code>scope</code>	Not generally used. If supplied this will be passed to <code>varcov</code> .
<code>subset</code>	Not generally used. If supplied this will be passed to <code>varcov</code> .
<code>casewt</code>	Not generally used. If supplied this will be passed to <code>varcov</code> .
<code>...</code>	optional arguments to pass to <code>twohk</code> and <code>swap</code>

**Details**

This function is a wrapper for [varcov](#), [twohk](#), [swap](#), and [summary.swap](#), and a better understanding of optional arguments and the object generated is gained from their documentation.

**Value**

<code>hk</code>	The object returned by <a href="#">twohk</a>
<code>swaps</code>	A list of objects returned by calls to <a href="#">swap</a> . Element <code>i</code> in <code>swaps</code> is for <code>i</code> gene models.
<code>smry</code>	A list of objects returned by calls to <a href="#">summary.swap</a> . Some elements may be <code>NULL</code> if no samples were requested or if the sampling process yielded degenerate results. Usually, this happens if no posterior is specified for the regression coefficients, i.e. if <code>rparm=0</code> was used or implied
<code>odds</code>	A Vector of odds (relative to a no gene setup) for each model size evaluated. The odds are computed under a prior that places equal weights on models of each size considered (and are, therefore, Bayes Factors). If models of size 1 and 2 are not evaluated or if some degenerate results were encountered, this will be <code>NULL</code>
<code>coefs</code>	A vector of posterior means of the regression coefficients. If models of size 1 and 2 are not evaluated or if some degenerate results were encountered, this will be <code>NULL</code>
<code>loc.posterior</code>	A vector of locus-wise posterior probabilities that the interval covered by this locus contains a gene.If models of size 1 and 2 are not evaluated or if some degenerate results were encountered, this will be <code>NULL</code>
<code>call</code>	The call that generated this object

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**References**

Berry C.C.(1998) Computationally Efficient Bayesian QTL Mapping in Experimental Crosses. *ASA Proceedings of the Biometrics Section*. 164–169.

**Examples**

```
data( little.ana.bc )
little.lin <- linear.bayes( bc.phenotype~locus(all), little.ana.bc, rparm=1 )
par(mfrow=c(2,3))
plot( little.ana.bc, little.lin$loc.posterior, type="h" )
little.lin$odds
par(mfrow=c(1,1))
plot(fitted(little.lin), residuals(little.lin))
```

---

little.ana.bc      *A simulated dataset*

---

**Description**

A simulation of a BC1 cross of 150 organisms with a genome of around 500 cM consisting of 5 chromosomes. The format is that created by [make.analysis.obj](#)

This dataset is built up from several others. The basic data are:

- little.bc.phenoA vector of phenotype data
- little.bc.markersA `map.frame` of marker data and
- little.dxA data frame with 50 rows and 2 columns that specify the map locations of a simulated set of markers

These are used to construct

- little.mf.5A `map.frame` with 'pseudo-markers' at least every 5 cM made from  

```
little.mf.5 <- make.map.frame(little.map.frame, nint=marker.fill(
  little.map.frame, reso=5, TRUE ))
```

Then phenotype, covariate, and marker data are combined with `little.mf.5`

- little.bc.pheno A data.frame with the variable `bc.phenotype`
- little.bc.markersA data.frame with marker state information

**See Also**

The examples in [make.analysis.obj](#)

---

little.ana.f2      *A simulated dataset*

---

**Description**

A simulation of an F2 cross of 150 organisms with a genome of around 500 cM consisting of 5 chromosomes. The format is that created by `make.analysis.obj`

**Usage**

```
data(little.ana.f2)
```

---

little.bc.markers      *Simulated Marker Data*

---

**Description**

The `little.bc.markers` data frame has 150 rows and 50 columns with the simulated marker data from a BC1 cross of 150 organisms with a genome of around 500 cM consisting of 5 chromosomes. Some NA's have been intentionally introduced.

**Usage**

```
data(little.bc.markers)
```

**Format**

This data frame contains the following columns:

- m.1** a factor with levels AA Aa
- m.2** a factor with levels AA Aa
- m.3** ditto
- m.4** ditto
- m.5** ditto
- m.6** ditto
- m.7** ditto
- m.8** ditto
- m.9** ditto
- m.10** ditto
- m.11** ditto
- m.12** ditto
- m.13** ditto

**m.14** ditto  
**m.15** ditto  
**m.16** ditto  
**m.17** ditto  
**m.18** ditto  
**m.19** ditto  
**m.20** ditto  
**m.21** ditto  
**m.22** ditto  
**m.23** ditto  
**m.24** ditto  
**m.25** ditto  
**m.26** ditto  
**m.27** ditto  
**m.28** ditto  
**m.29** ditto  
**m.30** ditto  
**m.31** ditto  
**m.32** ditto  
**m.33** ditto  
**m.34** ditto  
**m.35** ditto  
**m.36** ditto  
**m.37** ditto  
**m.38** ditto  
**m.39** ditto  
**m.40** ditto  
**m.41** ditto  
**m.42** ditto  
**m.43** ditto  
**m.44** ditto  
**m.45** ditto  
**m.46** ditto  
**m.47** ditto  
**m.48** ditto  
**m.49** ditto  
**m.50** ditto  
**row.names** row.names

---

little.bc.pheno      *Simulated Phenotype Data*

---

**Description**

The `little.bc.pheno` data frame has 150 rows and 1 columns.

**Usage**

```
data(little.bc.pheno)
```

**Format**

This data frame contains the following columns:

**bc.phenotype** a numeric vector of simulated phenotype data

---

little.f2.markers      *Simulated Marker Data*

---

**Description**

The `little.f2.markers` data frame has 150 rows and 50 columns with the simulated marker data from an F2 cross of 150 organisms with a genome of around 500 cM consisting of 5 chromosomes.

**Usage**

```
data(little.f2.markers)
```

**Format**

This data frame contains the following columns:

**m.1** a factor with levels AA Aa aa

**m.2** a factor with levels AA Aa aa

**m.3** ditto

**m.4** ditto

**m.5** ditto

**m.6** ditto

**m.7** ditto

**m.8** ditto

**m.9** ditto

**m.10** ditto

**m.11** ditto  
**m.12** ditto  
**m.13** ditto  
**m.14** ditto  
**m.15** ditto  
**m.16** ditto  
**m.17** ditto  
**m.18** ditto  
**m.19** ditto  
**m.20** ditto  
**m.21** ditto  
**m.22** ditto  
**m.23** ditto  
**m.24** ditto  
**m.25** a factor with levels A- aa  
**m.26** ditto  
**m.27** ditto  
**m.28** ditto  
**m.29** ditto  
**m.30** ditto  
**m.31** ditto  
**m.32** ditto  
**m.33** ditto  
**m.34** ditto  
**m.35** ditto  
**m.36** ditto  
**m.37** ditto  
**m.38** ditto  
**m.39** ditto  
**m.40** ditto  
**m.41** ditto  
**m.42** ditto  
**m.43** ditto  
**m.44** ditto  
**m.45** a factor with levels a-  
**m.46** ditto  
**m.47** ditto  
**m.48** ditto  
**m.49** a factor with levels AA Aa aa  
**m.50** a factor with levels AA Aa aa  
**row.names** row names

---

little.f2.pheno      *Simulated Phenotype Data*

---

### Description

The `little.f2.pheno` data frame has 150 rows and 1 columns.

### Usage

```
data(little.f2.pheno)
```

### Format

This data frame contains the following columns:

**f2.phenotype** a numeric vector of simulated phenotype data

---

little.map.dx      *Marker Map Description for Simulated Data*

---

### Description

The `little.map.dx` data frame has 50 rows and 2 columns that specify the map locations of a simulated set of markers

### Usage

```
data(little.map.dx)
```

### Format

This data frame contains the following columns:

**marker.names** a factor with levels `m.1 ... m.50`

**cM** a numeric vector of map locations in centimorgans

---

little.map.frame     *Package of Simulated Marker Map Information*

---

### Description

The `little.map.frame` data frame has 50 rows and 9 columns that describe the marker map of `little.map.dx` in the format produced by `make.map.frame`. `little.map.dx` has the minimal data needed to construct this.

### Usage

```
data(little.map.frame)
```

### Format

This data frame contains the following columns:

**marker.name** a factor with levels `m.1 m.2 ... m.50`

**cM** a vector of locations

**prior** weights to be used in sampling and Bayesian computations

**pos.type** a factor with levels `left right center`

**is.marker** always `TRUE` for these data

**pos.plot** a vector of plotting positions

**lambda** transformed recombination fractions

**locus** an abbreviated locus name

**chr.num** the chromosome number 1, 2, 3, 4, or 5.

---

little.mf.5     *Package of Simulated Marker Map Information*

---

### Description

The `little.mf.5` data frame has 114 rows and 9 columns consisting of `little.map.frame` plus 64 'virtual' marker loci

### Usage

```
data(little.mf.5)
```

**Format**

This data frame contains the following columns:

**marker.name** The marker names taken from `little.map.frame` and those created to fill virtual markers in between actual markers.

**cM** a vector of locations

**prior** weights to be used in sampling and Bayesian computations

**pos.type** a factor with levels `left right center`

**is.marker** TRUE for the 50 markers, FALSE for the 'virtual' markers

**pos.plot** a vector of plotting positions

**lambda** transformed recombination fractions

**locus** an abbreviated locus name

**chr.num** the chromosome number 1, 2, 3, 4, or 5.

---

locus

*Lookup loci or effects for genetic model formulas*


---

**Description**

Convert numeric indexes to names of regressors for a genetic model. One or many genetic models can be specified through the use of this function. It is used on the right hand side of a formula in the `bqtl` function.

**Usage**

```
locus(x, ..., scope, method, chromo, cM, ana.obj)
```

**Usage**

```
locus(x, ..., scope, method, chromo, cM, ana.obj)
add(x)
dom(x)
```

**Arguments**

<code>x</code>	Typically an integer, an integer vector, or an array whose elements are integers. These index loci described in a <code>map.frame</code> object. However, <code>x</code> can also be a character string, vector, et cetera, in which case the elements must belong to <code>names(scope)</code> .
<code>...</code>	Optional arguments (usually integers) to be used when <code>is.atomic(x)</code> is TRUE.
<code>chromo</code>	A chromosome number or 2 ordered numbers. The loci on the chromosome or in the range of chromosome numbers are used. If <code>chromo</code> is used, <code>x</code> must not be used.

<code>cM</code>	(Optional) map distance or two giving a location near a locus or range of locations from which loci will be included. If the one chromosome number is specified in <code>chromo</code> , <code>cM</code> must be ordered. If <code>cM</code> is omitted, all loci on the chromosome(s) will be included.
<code>scope</code>	(Optional and) Usually not supplied by the user. Rather <code>bqtl</code> fills this in automatically. A vector of regressor names, like the <code>reg.names</code> component returned by <code>make.analysis.obj</code> .
<code>method</code>	(Optional and) Usually not supplied by the user. Like <code>scope</code> , <code>bqtl</code> takes care of filling this in with "BC1", "F2", et cetera as appropriate.
<code>ana.obj</code>	Usually not specified by the user. This is the <code>analysis.object</code> to be used to lookup loci if a <code>chromo</code> argument is used.

### Details

`locus` is used in the model formula notation of `bqtl`, possibly more than once, and possibly with regressors named in the usual manner. `locus` is intended to speed up the specification and examination of genetic models by allowing many models to be specified in a shorthand notation in a single model formula. The names of genetic loci can consist of marker names, names that encode chromosome number and location, or other shorthand notations. The names of terms in genetic models will typically include the names of the locus and may prepend "add." or "dom." or similar abbreviations for the 'additive' and 'dominance' terms associated with the locus.

When used as in `bqtl( y ~ locus(34), my.analysis.obj )`, it will look up the term or terms corresponding to the 34th locus. When this is passed back to `bqtl`, it is pasted into a text string that will become a formula and is subsequently processed to yield the fit for a one gene model.

When used as in `bqtl( y ~ locus(34,75,172), my.analysis.obj )` it looks up each term and returns a result to `bqtl` that results in fitting a 3 gene model (without interaction terms).

When `x` is a vector or array, the processing typically returns pieces character strings for many model formulas. `bqtl(y ~ locus(26:75), ...)` results in a list of 50 different one gene model fits from `bqtl` for the terms corresponding to the 26th through the 75th variables. `bqtl(y ~ locus(cbind(c(15,45,192),c(16,46,193))), ...)` returns two three gene models. And more generally, whenever `is.array(x)` is TRUE, the columns (or slices) specify `dim(x)[1]/length(x)` different models.

The `chromo` argument performs a lookup of loci on the chromosome via the function `map.index`. If `cM` is also given, the locus nearest that location is used. If two values are given for `cM` all loci in the range are used.

`add(x)` and `dom(x)` are alternatives that specify that only the *additive* or *dominance* terms in an F2 intercross.

### Value

A character vector whose element(s) can be parsed as the right hand side of a model formula(s).

### Author(s)

Charles C. Berry <cberry@ucsd.edu>

**See Also**

`configs`, `bqt1`, and the examples there for a sense of how to use `locus`, `make.analysis.obj` for the setup that encodes the marker map and the marker information.

---

`loglik`*Extract loglikelihood, log posterior, or posterior from fitted models*

---

**Description**

A fitted model or a list of such generated by `bqt1` has a maximum log likelihood or log posterior and a posterior. These functions simply extract them.

**Usage**

```
loglik(x, ...)  
logpost(x, ...)  
posterior(x, ...)
```

**Arguments**

<code>x</code>	The object produced by <code>bqt1</code>
<code>...</code>	Currently unused

**Value**

A vector of numbers whose length equals the number of fitted models in `x`

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**See Also**

[bqt1](#)

---

make.analysis.obj *Set up data for QTL mapping*

---

### Description

Create commonly used objects for the analysis of a backcross or intercross experiment or of recombinant inbred lines.

### Usage

```
make.analysis.obj(data, map.frame, marker.frame, marker.levels=NULL,
method="F2", casewt=NULL, varcov=FALSE, mode.mat=NULL)
```

### Arguments

`data` A `data.frame` (or vector) of phenotype and (optionally) covariate information

`map.frame` A `map.frame.object` (see [make.map.frame](#)) encoding the map information and other details of the study

`marker.frame` A `marker.frame.object`. A matrix or `data.frame` of marker state information.

`marker.levels` A vector of length six or `NULL`. If `NULL` then the defaults for the elements are:

Element	F2.default	BC.default	RI.default
1	"AA"	"AA"	"AA"
2	"Aa"	"Aa"	"aa"
3	"aa"	"nil"	"nil"
4	"A-"	"nil"	"nil"
5	"a-"	"nil"	"nil"
6	"_"	"_"	"_"

NA's are allowed in `marker.frame` as well as the sixth element ("\_" by default) to denote missing data. To use other coding schemes replace "AA" and "aa" by codes for homozygous states, "Aa" by the code for heterozygotes, "A-" by the code for 'not aa', "a-" by the code for 'not AA', and "\_" by the missing code. Positions 3:5 are just placeholders if `method!="F2"`, but must be present.

`method` One of "F2", "BC1", "RI.self", or "RI.sib"

`casewt` If there are multiple observations on one genotype (such as in recombinant inbreds) this can be used to assign a weight to each observation. The wisdom of doing this is debatable.

`varcov` If `FALSE`, don't create a `varcov.object`. Otherwise give an index into `data` to select a dependent variable. See [varcov](#)

mode.mat If NULL use the default. For method=="F2" ( and the default marker.levels of AA, Aa, and aa ), this is a 3 by 2 matrix:

Genotype	add	dom
AA	1	-1
Aa	0	1
aa	-1	-1

For method=="BC1" ( and the default marker.levels of AA and Aa ),it is

Genotype	
AA	1
Aa	-1

and for RIL methods ( and the default marker.levels of AA and aa ),it is

Genotype	
AA	1
aa	-1

Other choices of marker.levels will relabel the corresponding rows.

## Details

A lot of stuff is bundled together in one object. The function is really just a wrapper calling other make.\* functions.

## Value

A list with components

data	data.frame of phenotype, covariate information, and regressors created by make.regressor.matrix
varcov	A varcov.object. See make.varcov
reg.names	The names of the regressors from make.regressor.matrix
method	The method argument in the call.
state.matrix	See make.state.matrix
loc.right	See make.loc.right
map.frame	See make.map.frame
casewt	The casewt argument
mode.mat	The mode.mat used
version	A string giving the version of BQTL from which the objects was created
call	The function call

**Note**

This can be quite a LARGE object. It might be better in crosses with lots (say, thousands) of markers, or in which many 'virtual' markers are used, or on computers with limited RAM to store each component separately. Not all components are used in every type of analysis.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[make.map.frame](#) for definition of the marker map, The internally used functions are: [make.loc.right](#), [make.state.matrix](#), [make.regressor.matrix](#), [make.varcov](#), and [make.marker.numeric](#)

**Examples**

```
data( little.bc.pheno )
data( little.mf.5 )
data( little.bc.markers )
names( little.bc.pheno )
little.ana.bc <- make.analysis.obj( little.bc.pheno$bc.phenotype,
                                  little.mf.5, little.bc.markers,
                                  method="BC1" )
summary( little.ana.bc )
```

---

make.loc.right	<i>Keep track of fully informative markers or states</i>
----------------	--

---

**Description**

Helps speed computations in multigene models by allowing a quick assessment of whether two loci are independent given the marker information for the individual.

**Usage**

```
make.loc.right( marker.frame, marker.distances )
```

**Arguments**

marker.frame A marker.frame.object

marker.distances

Actually a misnomer, this is a vector with a zero in the last position of each chromosome.

**Value**

A matrix of the same dimension as `marker.frame` whose elements index the column on the next (right) fully informative marker.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

---

```
make.location.prior
```

*Provide a default prior*

---

**Description**

Uses the map distances as a means of assigning a prior for chromosomal location. Basically, this function attempts to assign equal weight according to the spacing of markers and 'virtual' markers.

**Usage**

```
make.location.prior( x, add.2.end=0, normalize=TRUE )
```

**Arguments**

<code>x</code>	$x = e^{-mgd}$ , where <code>mgd</code> is the map distance in Morgans
<code>add.2.end</code>	How many Morgans to extend the first and last interval on each chromosome
<code>normalize</code>	If TRUE, let the result sum to 1.0

**Value**

A vector of `length(x)` whose sum is one, if `normalize==TRUE`

**Author(s)**

Charles C. Berry, <cberry@ucsd.edu>

---

make.map.frame      *Create marker map specifications*

---

### Description

A `map.frame` object describes a marker map and additional loci that may be used in a QTL study. Each row pertains to one locus. Names of markers, abbreviated names, distances, and other necessary and useful information are bundled.

### Usage

```
make.map.frame(dx, chr.num = NULL, prior = make.location.prior(lambda),
              morgan = 100, nint = NULL, reso = NULL)
```

### Arguments

<code>dx</code>	An object of class "map.frame" or class "data.frame" or a vector or a data.frame with a column named <code>cM</code> , <code>M</code> , or <code>dx</code> or whose first column gives location on each chromosome in centiMorgans (from start of chromosome or Morgans if <code>M</code> was the column name). It is best if <code>names(dx)</code> (for vector arguments) or <code>row.names(dx)</code> (for data.frame arguments) give names of markers for later reference, but this isn't really necessary.
<code>chr.num</code>	(Optional) Vector of chromosome numbers
<code>prior</code>	(Optional) Vector of Prior probabilities for the loci
<code>morgan</code>	100 if centiMorgans, 1 if Morgans
<code>nint</code>	(Optional) Vector of one plus number of 'virtual' markers to be inserted after each locus
<code>reso</code>	Maximum distance between loci. If necessary fill in with 'pseudo-markers'

### Details

The QTL analysis depends on information about the marker map and on specifications of the loci to be studied. The 'map.frame' contains this information.

### Value

A data frame with components:

<code>marker.name</code>	The full text identifier for this marker, e.g. "HH.360L.Col" is a marker on chromosome 1 of arabidopsis thaliana, and names like this can be used for reference purposes. 'Virtual' markers have a suffix appended to the name of the previous marker.
<code>cM</code>	Location on the chromosome. If this is a marker of a locus that was input via <code>dx</code> , then it is just the value of <code>dx</code> .

pos.type	"left" if it is the first locus on this chromosome,"right" if it is last, or "center" otherwise.
is.marker	TRUE if this was actually a marker, FALSE if it is a 'virtual' marker
pos.plot	Plotting position for this locus. Typically the same as dx.
lambda	Twice the recombination fraction minus one.
locus	An abbreviation for the locus of the form "C.<chr.num>.<cm>"
chr.num	The chromosome number.

**Note**

The idea in having all of this bundled together is to make it easier for plot and summary methods to be implemented and to allow convenient references in formula based methods.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**Examples**

```
data( little.map.dx )
little.map.frame <- make.map.frame( little.map.dx )
plot( little.map.frame ) # there is a plot method
# add 'virtual' markers to map
little.mf.5 <- make.map.frame(little.map.frame, reso=5)
print(little.mf.5[1:10,], digits=1) # show a few rows
plot( little.mf.5 ) # notice the 'virtual' markers added
```

---

```
make.marker.numeric
```

*Translate a marker.frame.object to numeric matrix*

---

**Description**

Not to be called directly by users. This utility function simply returns the coded numeric values corresponding to the allele states.

**Usage**

```
make.marker.numeric(marker.frame, level.names=NULL)
```

**Arguments**

`marker.frame` A data.frame.object consisting of factors or character vectors that encode the allele states.

`level.names` A vector of length 6 to translate the levels attribute or character codes into allele states that `make.state.matrix` understands. If necessary, dummy codes are used to fill the vector.

**Value**

A matrix, for which column *i* is `match(as.character(marker.frame[,i]), level.names)`

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

---

`make.regressor.matrix`

*Create regressors using expected marker values*

---

**Description**

Create regression variables for markers and loci between or near markers by imputation conditional on known marker states.

**Usage**

```
make.regressor.matrix(state.matrix, mode.mat=NULL)
```

**Arguments**

`state.matrix` A `state.matrix.object` - see [make.state.matrix](#) for more details

`mode.mat` A matrix which indicates the values of regressor variables corresponding to the allele states. If `mode.mat=NULL` (the default) a `mode.mat` is inferred from the dimensions of `state.matrix`. For the F2 intercross these are typically additive and dominance codes like (-1,0,1) and (1,-1,1). For BC1 backcross and RI lines, typically the values are (-1,1).

**Value**

A matrix with variables suitable for use as regressors.

**References**

Haley C.S. and Knott S.A. (1992) A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity* **69**,315-324.

**See Also**

[make.state.matrix](#)

---

```
make.state.matrix
```

*Create state.matrix.object*

---

## Description

Create a `state.matrix.object` to be used encode marker information in a form in which it can be used in subsequent calculations.

## Usage

```
make.state.matrix(marker.frame, marker.distances, method="F2")
```

## Arguments

`marker.frame` Actually, this is a misnomer. This is NOT a `marker.frame.object`. Rather it is obtained by by call like `make.marker.numeric(marker.frame.object)` (see `make.marker.numeric` ) and it is coerced to a matrix. It encodes marker allele states. One column is used for each marker or pseudo-marker (basically a placeholder with all missing values). The entries are in 1:6, if NA's are present, they are recoded to 6. The columns are arranged in linkage groups with presumed order reflected in the actual order of the columns.

`marker.distances`

Distances between the markers in the 'lambda' metric.  $-\log(\lambda)/2$  is the Haldance map distance. Linkage groups are separated by values of 0.0.

`method`

`method = "F2"` is the default, and "BC1", "RI.self", and "RI.sib" are other options. The assumed setup is as follows (strains are A and a):

marker state	F2.code	BC.code	RI.code
"AA"	1	1	1
"Aa"	2	2	
"aa"	3		2
"A-" (not aa)	4		
"a-" (not AA)	5		
"-" (unknown)	6	6	6

## Value

`n` by `k` by `q` array. `q` is 3 for `method="F2"` and 2 for others methods. Each element encodes the probability of the allele state conditional on the marker states.

## Note

It might have been better to design this array so that the third subscript moves fastest. In large problems, the current structure may involve excessive memory access.

## References

- Lander E.S. and Green P. (1987) Construction of multilocus genetic linkage maps in humans. *Proceedings of the National Academy of Sciences of the United States of America*, **84**(8), 2363–7.
- Jiang C. and Zeng Z-B. (1997) Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* **101**, 47-58.

---

make.varcov                      *Create moment matrices*

---

## Description

Create a moment matrix of the marker variables and of the regressors by the phenotype variable. For use in regression modelling on the markers.

## Usage

```
make.varcov(regressor.matrix, y, subset=is.finite(y), casewt=NULL)
```

## Arguments

regressor.matrix	The object produced by <code>make.regressor.matrix</code>
y	A vector of phenotype information with the same number of elements as there are rows in <code>regressor.matrix</code>
subset	Logical vector with the same number of elements as there are rows in <code>regressor.matrix</code> to indicate which rows to keep.
casewt	Optional vector of case weights.

## Value

A list with components

var.x	Moment matrix of the marker regressor variables
cov.xy	Moment matrix of the marker regressor variables versus the phenotype variable
var.y	The Second central moment of the phenotype variable
df	<code>sum(subset==TRUE) - 1</code>

## Note

It is generally NOT a good idea to do regressions on ill-conditioned designs using the moment matrices like this. The excuse for doing so here is twofold. First, calculations using this method are used to perform importance sampling, so minor numerical inaccuracies in computing the probabilities used in sampling get straightened out by the importance weights. Second, it will typically be the case that a prior is set on the regression coefficients and this results in a positive constant (aka a 'ridge' parameter) being added to diagonal of `varcov$var.x` and this reduces the ill-conditioning. Of course the rational for using the method is to speed the sampling, and it is very effective at doing so.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

---

 map.index

*Look up numerical index(es) of map locations*


---

**Description**

One way to index a locus (loci) in a genetic map is by the numerical index of its row (their rows). `map.index` performs a lookup in a specific `map.frame` given one (or two) chromosome number(s) and one (or two) map distance(s).

**Usage**

```
map.index(x, ... )
```

**Arguments**

<code>x</code>	A <code>map.frame</code> or <code>analysis.object</code>
<code>...</code>	For methods that look up a location in a <code>map.frame</code> the following named arguments may be used: <code>chromo</code> A chromosome number or 2 ordered numbers <code>cM</code> (Optional) map distance or two. If the same chromosome number is used twice in <code>chromo</code> , <code>cM</code> must be ordered. If <code>cM</code> is omitted, all loci on the chromosome will be included.

**Details**

It is often convenient to refer to genetic loci or regions by the numerical index(es) in a `map.frame`. `map.index` allows lookups according to the approximate map location.

**Value**

A numerical vector of one or more row numbers. If only `chromo` is specified, all row numbers on the specified chromosome are returned. If `chromo` has two elements, then all row numbers on those chromosomes with numbers in `range(chromo)` will be returned. If one of each of `chromo` and `cM` are specified, then the row number of the closest locus will be returned. For two of each, row numbers in the range of the closest matches will be returned.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[make.map.frame](#) for a description of how map information is organized.

**Examples**

```
data(little.ana.bc)
map.index(little.ana.bc,chromo=1,cM=25) # locus nearest 1,25
index.chr.1 <- map.index(little.ana.bc,chromo=1)
fit.on.1 <- bqt1(bc.phenotype~locus(index.chr.1),little.ana.bc)
summary(loglik(fit.on.1))
```

---

<code>map.location</code>	<i>Report map location</i>
---------------------------	----------------------------

---

**Description**

Report the chromosome number and location of loci in a genetic map.

**Usage**

```
map.location(x, ... )
map.loc(x, ... )
```

**Arguments**

<code>x</code>	A object of class <code>map.frame</code> , <code>analysis.object</code> , <code>bqt1</code> , or <code>bqt1.list</code>
<code>...</code>	Other arguments usage depend on the class of <code>x</code> : <code>y</code> A vector of row numbers or <code>map.names</code> specifying which subset of the <code>map.frame</code> of <code>x</code> is to be returned <code>chromo</code> : A vector of chromosome numbers <code>cM</code> (Optional) map distance vector. If the same chromosome number is used twice in <code>chromo</code> , <code>cM</code> must be ordered. If <code>cM</code> is omitted, all loci on each chromosome listed in <code>chromo</code> will be included. <code>map.names</code> A vector of <code>map.names</code>

**Details**

It is often helpful to refer to genentic loci by their locations. The methods of `map.location` (alias `map.loc`) will extract the row index, chromosome number and location, and the name for specified loci. For direct lookups of the loci in a `map.frame` or `analysis.object`, one must specify `y` or `chromo` or `map.names`. When `class(x)=="bqt1"` `map.locations` of terms used in a call to `bqt1` are returned. When `cM` is used, an attempt will be made to match the location; if the match fails, the nearest locus will be used. When there are two elements in `chromo` and two in `cM`, all the map locations in between the matching loci will be returned.

**Value**

An object of class `map.location` which inherits from `map.frame`. It has columns:

<code>chr.num</code>	The chromosome number
<code>cM</code>	The location in centiMorgans on that chromosome.

```
marker.name  The name by which that marker is known
attr(,"row.names")
             An index of the locations
```

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[make.map.frame](#)

**Examples**

```
data(little.ana.bc)

map.loc(little.ana.bc, c(1,15,45))
map.loc(little.ana.bc, chromo=3, cM=22)
map.loc(little.ana.bc, "m.12")
rm(little.ana.bc)
```

---

map.names

*Look up names of markers or loci*

---

**Description**

This is a generic helper function with methods that will return the names of markers or loci.

**Usage**

```
map.names(x, ...)
```

**Arguments**

**x** An object that has marker names in it. Methods for objects of the `map.frame`, `analysis.object`, `bqtl`, and `bqtl.list` class.

**...** For `class(x)=="analysis.object"` or `class(x)=="map.frame"`, arguments `chromo` and `cM` can be used as in `map.index`

**Details**

When applied to an object of class `bqtl`

```
map.names(x, ..., ana.obj )
```

can be used to specify where to find the data.

**Value**

A character vector

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[map.index](#), [map.location](#)

**Examples**

```
data(little.ana.bc)

map.names(little.ana.bc, chromo=1, cM=24)

map.names(little.ana.bc, chromo=c(1,1), cM=c(40,55))

fit <- bqt1( bc.phenotype ~ locus(23,42) , little.ana.bc )

map.names( fit )
```

---

marker.fill

*Map Positions Between Markers*

---

**Description**

Given a set of markers, one wants to create a finer map at a given resolution. `marker.fill` takes a collection of marker distances and a desired resolution and finds positions that are intermediate and at that resolution.

**Usage**

```
marker.fill(map.frame, reso, return.nint = FALSE)
```

**Arguments**

<code>map.frame</code>	A <code>map.frame</code> object.
<code>reso</code>	The desired interval between loci in the same metric as <code>map.frame\$cM</code>
<code>return.nint</code>	Whether to output a vector of number of intervals to produce in each existing interlocus interval

**Value**

If `return.nint` is `TRUE`, a vector of integers is returned. It indicates how many intervals to place between this marker and the next to achieve the desired minimum distance.

If `return.nint` is `FALSE`, a vector of distances is returned. The names attribute has suffixes added to indicate positions filled to the 'right' of existing markers. Thus if markers 'mark.01' and 'mark.02' are in succession at a distance of 3 and `reso==1`, then the value associated with 'mark.01' (which was 3) becomes 1, a value of 1 is associated with new loci called 'mark.01.1' and 'mark.01.2' in created with values of 1 each. The returned vector is ordered by chromosome, then marker or filled locus.

**See Also**

[make.map.frame](#)

**Examples**

```
data( little.map.frame )
little.nint <- marker.fill( little.map.frame, reso=5, TRUE )
cbind(nint=little.nint, cM=little.map.frame$cM) [1:10, ]
rm( little.map.frame, little.nint )
```

---

marker.levels      *Define marker level codes*

---

**Description**

The coding scheme used to define `marker.levels` is set up by these functions. *BQTL* has defaults that these functions can help the user to redefine.

**Usage**

```
bc1.levels( AA="AA", Aa="Aa", miss.val="--")
ri.levels(  AA="AA", aa="aa", miss.val="--")
f2.levels(  AA="AA", Aa="Aa", aa="aa", not.aa="A-", not.AA="a-",
miss.val="--")
```

**Arguments**

AA	Always used: the code for the homozygous state from one parent line
Aa	F2 and BC1 setups: the code for the heterozygous state
aa	F2 and RI setups: the code for the homozygous state for the other parent line
not.aa	F2 only: the code for a dominant marker that rules out aa
not.AA	F2 only: the code for a dominant marker that rules out AA
miss.val	The character string for a missing (unknown) allele state. NAs are automatically detected, so this is only needed if string values are used to denote missing values.

**Details**

It is essential that the codes intended by the user be clearly understood by *BQTL*. It is hoped that these functions provide a bridge between the internals of *BQTL* and the user's view of the marker codes. Numeric values can be used, but they will be coerced to character values.

**Value**

A vector with 6 elements corresponding to the values of AA, Aa, aa, not.aa, not.AA, and miss.val. For RI and BC1 setups, those that do not apply will be unnamed and set to "nil"

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[make.analysis.obj](#)

**Examples**

```
### show the defaults:

f2.levels()
bc1.levels()
ri.levels()

### suppose that 1,2,3 are codes used in F2:

f2.levels(1,2,3)

### show what would happen changing "Aa" to "H"

f2.levels(Aa="H")
bc1.levels(Aa="H")
```

---

plot.map.frame      *plots by chromosome location*

---

**Description**

Multiple x-y plots are formed using chromosome numbers (`chr.num`) and positions (`pos.plot`) specified in a object of the sort created by [make.map.frame](#)

**Usage**

```
## S3 method for class 'map.frame':
plot(x, y, ...)
```

**Usage**

```
plot.map.frame(x, y, fun, type, include.rug, rug.lwd,
              title.string, y.range, ylab, xlab, ...)
```

**Arguments**

**x** A `map.frame.object` or an `analysis.object`

**y** (optional) A vector with as many elements or a matrix with as many rows as `nrow(x)` . If omitted, a plot will be drawn in a single frame representing the chromosomes as horizontal bars and giving tic marks to show the locations markers and virtual markers (if any).

**...** more args: `fun` A plotting function to be used *after* the plot axes and labels have been drawn. The current default is `if (y.type == "matrix") matlines else lines` usually is good enough. But a fancier function could be used for a fancier plot. `type` "l" for lines, "p" for points, et cetera. see [par](#) `include.rug` if TRUE place a tick on the x-axis at each marker location `title.string` (optional) label to prepend to each title `ylab` plot label for y-axis, see [par](#) `xlab` plot label for x-axis, see [par](#)

**Details**

This function enables drawing graphs that depend on chromosome and chromosome location. Typically, one will use a command like `par(mfrow=c(nrows,ncols))` first to set up a page on which multiple plots will be drawn. However, one can draw one plot per page on postscript devices by leaving `par(mfrow=c(1,1))`

**Value**

NULL - this function is called only for its side effects

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**See Also**

[plot](#), [lines](#), and [matlines](#) for general information on plotting functions; [par](#) for optional arguments to add as arguments; and [make.map.frame](#) for the details on the object the drives this function.

**Examples**

```
data( little.ana.bc )
null.llk <- loglik(bqt1(bc.phenotype~1,little.ana.bc))
llk <- loglik( bqt1( bc.phenotype~locus(all), little.ana.bc) ) - null.llk
.old.par <- par(mfrow=c(2,3))
plot.map.frame(little.ana.bc$map.frame,llk)
```

```
par(.old.par)
```

---

```
predict.bqtl      fitted values from QTL models
```

---

### Description

The estimated coefficients and expected locus values are used to find fitted values for the QTL model

### Usage

```
predict.bqtl(object, newdata, ...)
fitted.bqtl(object, newdata, ...)
```

### Usage

```
## S3 method for class 'bqtl':
predict(object, newdata)
## S3 method for class 'bqtl':
fitted(object, newdata)
```

### Arguments

object	An object of class <code>bqtl</code>
newdata	An optional data.frame for which fitted values are to be found. If not specified, the a search for the original data frame for the fit will be made.

### Details

The estimated coefficients for a specific QTL model fit are used along with the expected locus values (conditionally on the marker values) are used to find fitted values for the QTL model. This is *not* the only way in which such fits could be obtained; one could condition the expect marker values on *both* the trait value and the marker values. One could also define fitted values for specific genotype combinations, e.g. for a backcross with k animals and a two gene model 4 fitted values could be determined for each animal leading to  $2*2*k$  values. In fact, using `newdata` one can do this.

### Value

A vector with as many elements as rows in `newdata` (after removing missing data) or in the original model.frame.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**See Also**

[bqt1](#)

**Examples**

```
data(little.ana.bc)

fit.pheno <- bqt1(bc.phenotype~locus(15)+locus(42), little.ana.bc)

summary(predict(fit.pheno))

genotype.grid <- expand.grid( c(-1,1), c(-1,1) )      # set up a grid
names(genotype.grid) <- map.names( fit.pheno )      # use matching names

fit.vals <- predict( fit.pheno, genotype.grid )      # make predictions
cbind( genotype.grid, fit.vals )                    # print them!
```

---

predict.linear.bayes

*Residuals or Predicted Values for linear.bayes objects*

---

**Description**

The `linear.bayes` object returns fitted coefficients. These are used to construct predicted values. Since the fitting process for `linear.bayes` objects is based on moments of centered variables, the 'intercept' is lost; see 'Details' below.

**Usage**

```
## S3 method for class 'linear.bayes':
residuals(object, newdata, return.resids)
## S3 method for class 'linear.bayes':
predict(object)
## S3 method for class 'linear.bayes':
fitted(object)
```

**Usage**

```
predict.linear.bayes(object, newdata=lb.call\$ana.obj, return.resids=FALSE, ...)
```

**Arguments**

`object`            An object returned by `linear.bayes`  
`newdata`            Optional `data.frame` in which to do the calculations  
`return.resids`        Not usually set by the user.

**Details**

Since the `linear.bayes` object is based on a moment matrix, some information is lost that must be reconstructed or assumed. The intercept and possibly the coefficients for control variates are among these. Also, when the call to `linear.bayes` supplied the moment matrix rather than formulae with which to create one, then it is unclear what variable was used as the regressand and hence which variable to use in forming residuals. So, in that case, `residuals` will report an error

**Value**

A vector of predicted values or residuals

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**See Also**

[linear.bayes](#)

---

`residuals.bqtl`        *Residuals from QTL models*

---

**Description**

The phenotype data, estimated coefficients, and expected locus values are used to find fitted values for the QTL model

**Usage**

```
residuals.bqtl(object, ...)
```

**Usage**

```
## S3 method for class 'bqtl':
residuals(object)
```

**Arguments**

`object`            An object of class `bqtl`

## Details

The estimated coefficients for a specific QTL model fit are used along with the expected locus values (conditionally on the marker values) are used to find fitted values for the QTL model; these are subtracted from the original trait values to get residuals. This is *not* the only way in which such fits could be obtained; one could condition the expected marker values on *both* the trait value and the marker values. One could also define fitted values for specific genotype combinations, e.g. for a backcross with  $k$  animals and a two gene model 4 fitted values could be determined for each animal leading to  $2 \times 2 \times k$  residuals.

## Value

A vector with as many elements trait values used in the original fitted model.

## Author(s)

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

## See Also

[bqt1](#)

## Examples

```
data(little.ana.bc)

fit.pheno <- bqt1(bc.phenotype~locus(15)+locus(42), little.ana.bc)

summary(residuals(fit.pheno))

plot(fitted(fit.pheno), residuals(fit.pheno))
```

---

summary.adj

*Summarize Laplace approximations*

---

## Description

The linear approximations of `swap` are much improved by the use a Laplace approximations for loci that are not markers. This function combines the results of a call like `bqt1(y~configs(swap.obj), ...)` with the data in `swap.obj` to provide improved posteriors, et cetera

## Usage

```
summary.adj(object, n.loc, coef.znames, mode.names=c("add", "dom"),
imp.denom=NULL, swap.obj=NULL, ...)
```

**Usage**

```
## S3 method for class 'adj':
summary(object, n.loc, coef.znames, mode.names=c("add",
"dom"), imp.denom=NULL, swap.obj=NULL)
```

**Arguments**

object	Typically, this is the result of a call like <code>bqt1(y~configs(swap.obj), ...)</code>
n.loc	The number of genes in this model
coef.znames	map.names for the sample space
mode.names	NULL except for "F2", in which case it is usually <code>c("add","dom")</code>
imp.denom	Optional, and only used when some sampling scheme other than the default MCMC generates object
swap.obj	The result of a call to <code>swap</code>

**Details**

There are a lot of details. This sections needs to be revised to reflect them.

**Value**

A list with components

adj	This multiplier adjusts the posterior odds for k vs k-1 gene models
var	An estimate of the variance of adj
coef	Posterior means of coefficients
loc	Marginal Posterior for location for k gene model
hk.ratio.mean	argh! I need to look this up

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**References**

Berry C.C. (1998) Computationally Efficient Bayesian QTL Mapping in Experimental Crosses. *ASA Proceedings of the Biometrics Section*, 164-169.

---

```
summary.bqtl      Summarize bqtl object
```

---

**Description**

Extract coefficients (and related stats), loglikelihood, and residual standard error of the trait.

**Usage**

```
## S3 method for class 'bqtl':
summary(object, ...)
```

**Arguments**

object	The result of <code>link{bqtl}</code>
...	Currently not used

**Value**

A list containing

`coefficients`

Either a vector of regression coefficients, or if `object` was created via `bqtl(..., return.hess=TRUE)` then a matrix with coefficients, standard errors, t-statistics, and p-values

`loglik` the loglikelihood or log posterior

`std.res` The residual standard deviation of the trait

`N` The counts of all observations, the number omitted, and the number used in the fit

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**Examples**

```
data(little.ana.bc)
fit <- bqtl( bc.phenotype~locus(4)*locus(45), little.ana.bc,
  return.hess=TRUE )
summary(fit)
```

---

summary.map.frame *Summary methods for basic data objects*

---

### Description

Provide a simple report on the data structure

### Usage

```
summary.map.frame(object, ...)
```

### Usage

```
## S3 method for class 'map.frame':
summary(object)
```

### Arguments

object            A map.frame or analysis.object

### Value

a list

### Author(s)

Charles C. Berry <cberry@ucsd.edu>

---

summary.swap            *Summarize Gibbs samples for a k-gene model*

---

### Description

Calculate marginal posteriors for location, posterior means for coefficients, and the Bayes Factor for k vs k-1 genes

### Usage

```
summary.swap(object, method, ncoef, nloc, ...)
```

### Usage

```
## S3 method for class 'swap':
summary(object, method=NULL, ncoef=length(sc.obj$alt.coef), nloc=sc.obj$nloc)
```

**Arguments**

<code>object</code>	The result of <code>swap</code>
<code>method</code>	Optional. One of the supported methods, see <code>make.analysis.obj</code>
<code>ncoef</code>	Optional. The number of coefficients in the class of models. Typically, $2 * nloc$ for <code>method=="F2"</code> and <code>nloc</code> for all other methods
<code>nloc</code>	Optional. The number of loci in the sample space.

**Value**

A list with components:

<code>loc.posterior</code>	A vector of (marginal) posterior odds for each locus compared to a no gene model
<code>coefs</code>	Posterior means of coefficients.
<code>ratio</code>	A list with components <code>mean</code> , an estimate of the Bayes Factor for k versus k-1 gene models, and <code>var</code> , an estimate of its variance.

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

---

swap

*MCMC sampling of multigene models*

---

**Description**

Given a k-gene model as a starting point, one gene is deleted and another is sampled in its place. This is done using an approximation to the posterior. Then another gene is deleted and another sampled,...

**Usage**

```
swap(varcov, invars, rparm, nreps, ana.obj, ...)
```

**Arguments**

<code>varcov</code>	The result of <code>make.varcov</code>
<code>invars</code>	Vector of numerical indexes of <code>ana.obj\$reg.names</code> telling which variables to start in the model. The first of these is immediately removed, so it is merely a placeholder. The number of genes in the model is therefore <code>k &lt;- length(invars)</code> (except when <code>ana.obj\$method=="F2"</code> when it is <code>k &lt;- length(unique(col(ana.obj\$reg.names)[invars]))</code> )
<code>rparm</code>	Scalar or vector with <code>nrow(varcov\$var.x)</code> elements; the 'ridge' parameters for the independent variables - larger values imply more shrinkage or a more concentrated prior for the regression coefficients.

<code>nreps</code>	How many cycles (of <code>k</code> samples each) to perform.
<code>ana.obj</code>	An <code>analysis.object</code> — see <a href="#">make.analysis.obj</a>
<code>...</code>	Additional arguments override the default choices of candidate loci ( <code>locs</code> ), prior for locus ( <code>locs.prior</code> ), or method specified by <code>ana.obj</code> . Also, the default prior for model ( <code>combo.prior</code> ) when <code>ana.obj\$method=="F2"</code> can be overridden. See <a href="#">swapbc1</a> and <a href="#">swapf2</a> for details.

### Details

An MCMC sampler for loci using the object of `make.varcov` is executed. This sampler uses the exact posterior probability under the assumed correctness of the regression model using expected genotypes given marker values. This amounts to linearizing the likelihood with respect to the (possibly unknown) locus states. For models where the loci are fully informative markers this is the true posterior.

The chain is implemented as follows: given a set of regressor variables to start, one variable is removed, all regressor variables not in the model are examined to determine the effect of each on the posterior. One variable is sampled. The process is repeated until each variable has been removed and a new one sampled in its place (possibly the same variable that was removed is sampled). And this whole cycle is repeated `nreps` times.

### Value

A list with components:

<code>config</code>	A <code>k</code> by <code>k</code> by <code>nreps</code> array (or, for <code>ana.obj\$method=="F2"</code> , a <code>2k</code> by <code>k</code> by <code>nreps</code> array) of the locations (variables) sampled in each iteration.
<code>posteriors</code>	A vector of length <code>k*nreps</code> with the posteriors of the models.
<code>coefs</code>	A <code>k</code> by <code>k</code> matrix of the regression coefficients (or, for <code>ana.obj\$method=="F2"</code> , a <code>2k</code> by <code>nreps</code> matrix).
<code>call</code>	The call to <code>swap</code>
<code>cond</code>	The <code>k*nreps</code> posterior probabilities of the <code>k-1</code> gene models.
<code>marg</code>	The <code>k*nreps</code> marginal posteriors for all <code>k</code> gene models that could be formed using the current <code>k-1</code> gene model
<code>alt.marginal</code>	A vector with <code>length(locs)</code> (or <code>2*length(locs)</code> ) elements. At each step, the posterior associated with each candidate locus is added to an element of this vector. After all steps are finished, the result is normalized to sum to one. This turns out to be a stable estimate of the marginal posterior.
<code>alt.coef</code>	A vector with <code>length(locs)</code> (or <code>2*length(locs)</code> ) elements. At each step, the product of each posterior times the coefficient(s) associated with a candidate locus is added to an element of this vector. After all steps are finished, the result is normalized by the total marginal posterior. This turns out to be a stable estimate of the marginal (over all models) posterior mean of the regression coefficients.

### Author(s)

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

## References

Berry C.C. (1998) Computationally Efficient Bayesian QTL Mapping in Experimental Crosses. *ASA Proceedings of the Biometrics Section*, 164-169.

## Examples

```
data( little.ana.bc )
little.vc <- varcov( bc.phenotype~locus(all), little.ana.bc)
little.4 <- swap( little.vc, c(1,15,55,75), rparm=1, 50, little.ana.bc )
little.4.smry <- summary( little.4 )
print(c("Bayes Factor (3 vs 4)"=little.4.smry$ratio$mean))
par(mfrow=c(3,2))
plot( little.ana.bc, little.4.smry$loc.posterior, type="h",
      ylab="E(genes)" )
rm(little.4,little.vc,little.ana.bc)
```

---

swapbc1

*Sample BC1 or Recombinant Inbred loci via approximate posterior.*

---

## Description

An MCMC sampler for loci using precomputed dispersion matrices, various priors, and a pre-selected set of variables. For use with BC1 (backcross) designs and recombinant inbred lines.

## Usage

```
swapbc1(varcov, invars, rparm, nreps, ana.obj, locs=NULL,
        locs.prior=NULL, tol=1e-10 )
```

## Arguments

varcov	The result of <code>make.varcov</code>
rparm	Scalar or vector with <code>nrow(varcov\$var.x)</code> elements; the 'ridge' parameters for the independent variables - larger values imply more shrinkage or a more concentrated prior for the regression coefficients.
nreps	How many cycles of MCMC to perform
ana.obj	A object produced by <code>make.analysis.obj</code>
invars	Which variables to start in the model. The first of these is immediately removed, so it is merely a placeholder. The number of genes in the model is therefore <code>k &lt;- length(invars)</code>
locs	The columns of <code>varcov\$var.x</code> to use. The default uses all of them.
locs.prior	The prior mass to associate with each variable. Typically, these sum to one, but sometimes they might each be set to one (as in computing lod scores).
tol	Used in forming QR decomposition. Let it be.

**Details**

An MCMC sampler for loci using the object of `make.varcov` is executed. This sampler uses the exact posterior probability under the assumed correctness of the regression model using expected genotypes given marker values. This amounts to linearizing the likelihood with respect to the (possibly unknown) locus states. For models where the loci are fully informative markers this is the true posterior.

The chain is implemented as follows: given a set of regressor variables to start, one variable is removed, all regressor variables not in the model are examined to determine the effect of each on the posterior. One variable is sampled. The process is repeated until each variable has been removed and a new one sampled in its place (possibly the same variable that was removed is sampled). And this whole cycle is repeated `nreps` times.

**Value**

A list with components:

<code>config</code>	A $k$ by $k$ by <code>nreps</code> array of the locations sampled in each iteration.
<code>posteriors</code>	A vector of length $k \times nreps$ with the posteriors of the models.
<code>coefs</code>	A $k$ by $k$ matrix of the regression coefficients.
<code>call</code>	The call to <code>swapbc1</code>
<code>cond</code>	The $k \times nreps$ posterior probabilities of the $k-1$ gene models.
<code>marg</code>	The $k \times nreps$ marginal posteriors for all $k$ gene models that could be formed using the current $k-1$ gene model
<code>alt.marginal</code>	A vector with <code>length(locs)</code> elements. At each step, the posterior associated with each candidate locus is added to an element of this vector. After all steps are finished, the result is normalized to sum to one. This turns out to be an exceedingly stable estimate of the marginal posterior.
<code>alt.coef</code>	A vector with <code>length(locs)</code> elements. At each step, the product of each posterior times the coefficient associated with a candidate locus is added to an element of this vector. After all steps are finished, the result is normalized by the total marginal posterior. This turns out to be an exceedingly stable estimate of the marginal (over all models) posterior mean of the regression coefficients.

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**References**

Berry C.C. (1998) Computationally Efficient Bayesian QTL Mapping in Experimental Crosses. *ASA Proceedings of the Biometrics Section*, 164-169.

**See Also**

[swapf2](#)

---

 swapf2

---

*Sample F2 loci via approximate posterior*


---

### Description

An MCMC sampler for loci using precomputed dispersion matrices, various priors, and a pre-selected set of variables. For use with F2 intercross design.

Using precomputed dispersion matrices, various priors, and a pre-selected set of variables, one locus is removed, all other loci are examined to determine the effect of each on the posterior. One locus is sampled. The process is repeated until each locus has been removed and a new one sampled in its place (possibly the same one that was removed is sampled).

### Usage

```
swapf2(varcov, invars, rparm, nreps, ana.obj, locs = NULL,
       locs.prior = NULL, combo.prior = rep(1, 3)/3, tol =
       1e-10)
```

### Usage

```
swapf2(varcov, invars, rparm, nreps, ana.obj, locs = <<see below>>,
       locs.prior = <<see below>>, combo.prior = <<see below>>, tol = 1e-10)
```

### Arguments

varcov	The result of <code>make.varcov</code> . The columns of <code>varcov\$var.x</code> must alternate 'additive' and 'dominance' terms.
rparm	The 'ridge' parameters for the independent variables - larger values imply more shrinkage or a more concentrated prior for the regression coefficients.
nreps	How many cycles of MCMC to perform
ana.obj	A object produced by <code>make.analysis.obj</code>
invars	A vector of variable indexes. This determines which variables to start in the model. If both additive and dominance terms are to be used, they should occupy adjacent locations in <code>invars</code> . The variable(s) associated with the first locus is (are) immediately removed, serving only as placeholder(s). If there are k loci associated with the variables, then all subsequent models have k loci, although the number of variables may vary according to the selection of one or both of the 'additive' or 'dominance' terms.
locs	The pairs of columns of <code>varcov\$var.x</code> to use. The default uses all of them.
locs.prior	Vector whose elements are the prior masses to associate with each locus. Typically, these sum to one, but sometimes they might each be set to one (as in computing lod scores). The default value sets them all to 1.0.

<code>combo.prior</code>	The prior probability for each term or combination of terms for the phenotypic effect at a locus. Typically, there will be three of these - one for the 'additive' term (linear in number of alleles from one parent strain), the 'dominance' term (quadratic in allele number), or both terms. The default sets them all to 1/3.
<code>tol</code>	Used in forming QR decomposition. Let it be.

### Details

A call to `swapf2` is used to obtain the results. This function is really just a wrapper.

### Value

A list with components:

<code>configs</code>	A $2k$ by $k$ by $nreps$ array of indexes of variables sampled in each of the $nreps$ iterations. Models using less than $2k$ variables <code>configs[,i,j]</code> will contain one or more zeroes in the last position(s)
<code>posteriors</code>	A vector of length $k*nreps$ with the posteriors of the models sampled.
<code>coefs</code>	A $2k$ by $k$ by $nreps$ matrix of the regression coefficients. Models using less than $2k$ variables <code>configs[,i,j]</code> will contain one or more zeroes in the last position(s)
<code>call</code>	The call to <code>swapf2</code>
<code>cond</code>	The $k*nreps$ posterior probabilities of the $k-1$ gene models.
<code>marg</code>	The $k*nreps$ marginal posteriors for all $k$ gene models that could be formed using the current $k-1$ gene model)
<code>alt.marginal</code>	A vector with <code>length(locs)</code> elements. At each step, the posterior associated with each candidate locus is added to an element of this vector. After all steps are finished, the result is normalized to sum to one. This turns out to be an exceedingly stable estimate of the relative marginal posterior.
<code>alt.coef</code>	A vector with $2*length(locs)$ elements. At each step, the product of each posterior times the coefficient associated with a candidate variable is added to an element of this vector. After all steps are finished, the result is normalized by the total marginal posterior. This turns out to be a rather stable estimate of the marginal (over all models) posterior mean of the regression coefficients.

### Author(s)

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

### References

Berry C.C. (1998) Computationally Efficient Bayesian QTL Mapping in Experimental Crosses. *ASA Proceedings of the Biometrics Section*, 164-169.

### See Also

[swapbc1](#)

**Description**

Fits all one and two gene models (without interactions aka 'epistasis') in an intercross, backcross, or recombinant inbred line. Uses a linear approximation to the likelihood, i.e. the expected allele states are used.

**Usage**

```
twohk( varcov, ana.obj, ...)
```

**Usage**

```
twohk( varcov, ana.obj, ...)
```

**Arguments**

<code>varcov</code>	An object produced by <code>make.varcov</code>
<code>ana.obj</code>	An <code>analysis.object</code> — see <code>make.analysis.obj</code>
<code>...</code>	Additional arguments override the default choices of candidate loci ( <code>locs</code> ), prior for locus ( <code>locs.prior</code> ), or method specified by <code>ana.obj</code> : <code>locs</code> A vector indexing the loci to use. <code>locs.prior</code> The prior mass to associate with each locus. Typically, these sum to one, but sometimes they might each be set to one (as in computing lod scores). <code>combo.prior</code> Only valid for <code>ana.obj\$method=="F2"</code> . The prior probability for each term or combination of terms for the phenotypic effect at a locus. Typically, there will be three of these - one for the 'additive' term (linear in number of alleles from one parent strain), the 'dominance' term (quadratic in allele number), or both terms. The default sets them all to 1/3.

**Details**

The marginal posterior (integrating over regression parameters and dispersion) is calculated for each one and two gene model under the assumed correctness of the regression model using expected genotypes given marker values. This amounts to linearizing the likelihood with respect to the (possibly unknown) locus states. For models where the loci are fully informative markers this is the true posterior.

**Value**

A list with components:

loc.1	The marginal posterior for each one gene model relative to a no gene model. For <code>twohkf2</code> this is a matrix of 3 columns; the first for models with additive terms, the second for dominance terms, and the third for both. The sum over all three columns yields the marginal posterior for the locus.
loc.2	The marginal posterior for each locus — obtained by summing over all two gene models that include that locus— relative to a no gene model. For <code>twohkf2</code> this is a matrix of 3 columns; the first for models with additive terms, the second for dominance terms, and the third for both.
coefs.1	The regression coefficients for the genetic effect for each locus. For <code>twohkf2</code> , this is a matrix with two rows; the first is for the 'additive effect' and the second is for the 'dominance' effect.
coefs.2	The marginal posterior mean of regression coefficients for the genetic effect for each locus - obtained by averaging over all two gene models that include that locus according to the posterior masses. For <code>twohkf2</code> , this is a matrix with two rows; the first is for the 'additive effect' and the second is for the 'dominance' effect.

**Author(s)**

Charles C. Berry <cberry@ucsd.edu>

**References**

Haley C.S. and Knott S.A. (1992) A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity* **69**,315-324.

**Examples**

```
data(little.ana.bc)
little.vc<-make.varcov(little.ana.bc$data[,little.ana.bc$reg.names],
  little.ana.bc$data$bc.phenotype)
little.2<- twohk(little.vc,little.ana.bc,rparm=1)
print( c(odds.1=sum(little.2$loc.1),odds.2=sum(little.2$loc.2)) )
par(mfrow=c(3,2))
little.pe <- 2 * little.2$loc.2 / sum(little.2$loc.2) #locus-wise posterior expectation
plot(little.ana.bc,little.pe,type="h",ylab="E(genes)")
rm(little.2,little.vc,little.pe,little.ana.bc)
```

---

twohkbcl

*One and Two Gene Models Using Linearized Posterior*

---

**Description**

Fits all one and two gene models (without interactions aka 'epistasis') in an intercross, backcross, or recombinant inbred line. Uses a linear approximation to the likelihood, i.e. the expected allele states are used.

**Usage**

```
twohkbcl(varcov, ana.obj, rparm = 0, locs = NULL, locs.prior =
        NULL)
```

**Usage**

```
twohkbcl(varcov, rparm, locs=<<see below>> , locs.prior=<<see below>> )
```

```
twohkf2(varcov, ana.obj, rparm, locs=<<see below>>, locs.prior=<<see below>>,
        combo.prior=<<see below>>)
```

**Arguments**

varcov	An object produced by <code>make.varcov</code>
ana.obj	An object produced by <code>make.analysis.obj</code>
rparm	The 'ridge' parameters for the independent variables - larger values imply more shrinkage or a more concentrated prior for the regression coefficients.
locs	The columns (or pairs of columns for <code>twohkf2</code> ) of <code>varcov\$var.x</code> to use. The default uses all of them.
locs.prior	The prior mass to associate with each locus. Typically, these sum to one, but sometimes they might each be set to one (as in computing lod scores).
combo.prior	Only valid for <code>twohkf2</code> . The prior probability for each term or combination of terms for the phenotypic effect at a locus. Typically, there will be three of these - one for the 'additive' term (linear in number of alleles from one parent strain), the 'dominance' term (quadratic in allele number), or both terms. The default sets them all to 1/3.

**Details**

The marginal posterior (integrating over regression parameters and dispersion) is calculated for each one and two gene model under the assumed correctness of the regression model using expected genotypes given marker values. This amounts to linearizing the likelihood with respect to the (possibly unknown) locus states. For models where the loci are fully informative markers this is the true posterior.

**Value**

A list with components:

loc.1	The marginal posterior for each one gene model. For <code>twohkf2</code> this is a matrix of 3 columns; the first for models with additive terms, the second for dominance terms, and the third for both. The sum over all three columns yields the marginal posterior for the locus.
loc.2	The marginal posterior for each locus - obtained by summing over all two gene models that include that locus. For <code>twohkf2</code> this is a matrix of 3 columns; the first for models with additive terms, the second for dominance terms, and the third for both.

- `coefs.1` The regression coefficients for the genetic effect for each locus. For `twohkf2`, this is a matrix with two rows; the first is for the 'additive effect' and the second is for the 'dominance' effect.
- `coefs.2` The marginal posterior mean of regression coefficients for the genetic effect for each locus - obtained by averaging over all two gene models that include that locus according to the posterior masses. For `twohkf2`, this is a matrix with two rows; the first is for the 'additive effect' and the second is for the 'dominance' effect.

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**References**

Haley C.S. and Knott S.A. (1992) A simple regression method for mapping quantitative trait loci in line crosses using flanking markers. *Heredity* **69**,315-324.

---

`varcov`

*Create moment matrices*

---

**Description**

Create a moment matrix of the marker variables and of the regressors by the phenotype variable. For use in regression modelling on the markers.

**Usage**

```
varcov(x, ana.obj, partial=NULL, scope=ana.obj$reg.names, ... )
```

**Usage**

```
varcov(x, ana.obj, partial=NULL, scope=<<see below>>)
```

**Arguments**

- `x` A formula to specify the dependent and independent variables to be used in subsequent calculations e.g `trait ~ locus(.)`
- `ana.obj` An `analysis.object`, see [make.analysis.obj](#)
- `partial` A formula whose right hand side specifies variables to be treated as covariates.
- `scope` Usually not explicitly used. Optional vector of variable names.

**Details**

This is just a wrapper for [make.varcov](#).

**Value**

A list with components

<code>var.x</code>	Moment matrix of the marker regressor variables
<code>cov.xy</code>	Moment matrix of the marker regressor variables versus the phenotype variable
<code>var.y</code>	The Second central moment of the phenotype variable
<code>df</code>	The degrees of freedom, when no variables are specified in <code>partial</code> it is <code>sum(subset==TRUE) - 1</code>

**Note**

It is generally NOT a good idea to do regressions on ill-conditioned designs using the moment matrices. The excuse for doing so here is twofold. First, calculations using this method are used to perform importance sampling, so minor numerical inaccuracies in computing the probabilities used in sampling get straightened out by the importance weights. Second, it will typically be the case that a prior is set on the regression coefficients and this results in a positive constant (aka a 'ridge' parameter) being added to diagonal of `varcov()$var.x` and this reduces the ill-conditioning. Of course the rational for using the method is to speed the sampling, and it is very effective at doing so.

**Author(s)**

Charles C. Berry <[cberry@ucsd.edu](mailto:cberry@ucsd.edu)>

**See Also**

The examples in [swap](#) and [twohk](#).

# Index

## \*Topic **datasets**

- little.ana.bc, 15
- little.ana.f2, 16
- little.bc.markers, 16
- little.bc.pheno, 18
- little.f2.markers, 18
- little.f2.pheno, 20
- little.map.dx, 20
- little.map.frame, 21
- little.mf.5, 21

## \*Topic **hplot**

- plot.map.frame, 39

## \*Topic **manip**

- make.analysis.obj, 25
- make.loc.right, 27
- make.map.frame, 29
- marker.levels, 38

## \*Topic **methods**

- coef.bqtl, 7
- formula.bqtl, 11
- map.index, 34
- predict.bqtl, 41
- predict.linear.bayes, 42
- residuals.bqtl, 43
- summary.adj, 44
- summary.bqtl, 46
- summary.map.frame, 47
- summary.swap, 47

## \*Topic **models**

- bqtl.fitter, 6
- loglik, 24
- twohk, 54

## \*Topic **regression**

- A Starting Point, 1
- adjust.linear.bayes, 2
- bqtl, 3
- covar, 9
- lapadj, 11
- linear.bayes, 13

- locus, 22

- map.location, 35
- swap, 48

## \*Topic **utilities**

- bqtl-internal, 5
- configs, 8
- make.location.prior, 28
- make.marker.numeric, 30
- make.regressor.matrix, 31
- make.state.matrix, 32
- make.varcov, 33
- map.names, 36
- marker.fill, 37
- swapbc1, 50
- swapf2, 52
- twohkbc1, 55
- varcov, 57

- %equiv%(bqtl-internal), 5

- A Starting Point, 1

- add, 10

- add(*locus*), 22

- adjust.linear.bayes, 2

- bc1.levels(*marker.levels*), 38

- bqtl, 2, 3, 6–9, 11, 22, 24, 41–44

- bqtl-internal, 5

- bqtl-package(*A Starting Point*), 1

- bqtl.fitter, 6

- coef.bqtl, 7

- configs, 4, 8, 10

- covar, 9

- dom, 10

- dom(*locus*), 22

- f2.levels(*marker.levels*), 38

- fitted.bqtl(*predict.bqtl*), 41

- fitted.linear.bayes

- (*predict.linear.bayes*), 42

`formula.bqtl`, 11

`lapadj`, 4, 11

`linear.bayes`, 2, 3, 13, 43

`lines`, 40

`little.ana.bc`, 15

`little.ana.f2`, 16

`little.bc.markers`, 16

`little.bc.pheno`, 18

`little.f2.markers`, 18

`little.f2.pheno`, 20

`little.map.dx`, 20, 21

`little.map.frame`, 21

`little.mf.5`, 21

`locus`, 2, 4, 10, 22

`loglik`, 24

`logpost` (*loglik*), 24

`make.analysis.obj`, 2, 4, 8, 9, 12, 14–16, 23, 24, 25, 39, 48–50, 52, 56, 57

`make.loc.right`, 27, 27

`make.location.prior`, 28

`make.map.frame`, 2, 21, 25, 27, 29, 34, 36, 38–40

`make.marker.numeric`, 27, 30, 32

`make.regressor.matrix`, 27, 31, 33

`make.state.matrix`, 27, 31, 32

`make.varcov`, 27, 33, 48, 50, 54, 56, 57

`map.dx` (*bqtl-internal*), 5

`map.index`, 23, 34, 37

`map.loc` (*map.location*), 35

`map.location`, 35, 37

`map.names`, 36

`marker.fill`, 37

`marker.levels`, 2, 38

`matlines`, 40

`par`, 40

`plot`, 40

`plot.analysis.object`  
(*plot.map.frame*), 39

`plot.map.frame`, 39

`posterior` (*loglik*), 24

`predict.bqtl`, 41

`predict.linear.bayes`, 42

`residuals.bqtl`, 43

`residuals.linear.bayes`  
(*predict.linear.bayes*), 42

`rhs.bqtl` (*bqtl-internal*), 5

`ri.levels` (*marker.levels*), 38

`summary.adj`, 44

`summary.analysis.object`  
(*summary.map.frame*), 47

`summary.bqtl`, 46

`summary.map.frame`, 47

`summary.swap`, 14, 47

`swap`, 14, 48, 48, 58

`swapbc1`, 8, 9, 49, 50, 53

`swapf2`, 9, 49, 51, 52

`twohk`, 14, 54, 58

`twohkbcl`, 55

`twohkf2` (*twohkbcl*), 55

`uniq.config` (*bqtl-internal*), 5

`varcov`, 14, 25, 57

`version.bqtl` (*bqtl-internal*), 5

`zero.dup` (*bqtl-internal*), 5