cape: A package for the combined analysis of epistasis and pleiotropy

Anna L. Tyler[1], Wei Lu[1,2], Justin J. Hendrick[1], Vivek M. Philip[1], and Gregory W. Carter[1]

June 9, 2016

[1]The Jackson Laboratory, Bar Harbor, ME, 04609 [2]Department of Electrical and Computer Engineering, Duke University, Durham, NC 27708

Contents

Abstract **1**

Loading Data **2**

Manipulating the Data Object **4**

Looking at the Data **5**

Decomposing the Phenotypes **9**

Kinship Corrections **11**

Single-Variant Scan **12**

Covariate Selection **14**

Pairwise Scan **15**

Combined Analysis for Detection of Interactions **18**

Interpretation of Results **23**

Tables of Functions **27**

Abstract

Here we present an R package for the Combined Analysis of Epistasis and Pleiotropy, or `cape`. This package implements a method, originally described in Carter et al. (2012), that infers directed interaction networks between genetic variants for predicting the influence

of genetic perturbations on phenotypes. This method takes advantage of complementary information in partially pleiotropic genetic variants to resolve directional influences between variants that interact epistatically. `cape` can be applied to a variety of genetic variants, such as single nucleotide polymorphisms (SNPs), copy number variations (CNVs) or structural variations (SVs). Here we demonstrate the functionality of `cape` by inferring a predictive network between quantitative trait loci (QTL) in a cross between the non-obese, non-diabetic (NON) mouse and the New Zealand obese (NZO) mouse (Reifsnyder, 2000).

Loading Data

For the purposes of demonstration, we will reanalyze a data set described in Reifsnyder (2000). This data set was established to find quantitative trait loci (QTL) for obesity and other risk factors of type II diabetes in a reciprocal back-cross of non-obese non-diabetic NON/Lt mice and diabetes-prone, New Zealand obese (NZO/HILt) mice. The study found multiple main-effect QTL influencing phenotypes associated with diabetes and obesity as well as multiple epistatic interactions. In addition, maternal environment (i.e. whether the mother was obese) was found to interact with several markers and epistatic pairs to influence the risk of obesity and diabetes of the offspring. The complex nature of diabetes and obesity, along with their complex and polygenic inheritance patterns, make this data set ideal for an analysis of epistasis and pleiotropy.

Included in this dataset are 204 male mice genotyped at 85 markers across the genome. The phenotypes included are the body weight (g), insulin levels (ng/mL), and plasma glucose levels (mg/dL), all measured at age 24 weeks. In addition, there is a variable called "mom" indicating whether the mother of each mouse was normal weight (0) or obese (1). After installing `cape`, load the package and the dataset type the following in the R command line.

```
> library(cape)
> data(obesity.cross)
```

There are multiple functions that can be used to load your own data set. The primary function is called `read.population()`. This function is similar to the function `read.cross()` used in the R package `qtl` (Broman et al., 2003), and accepts the basic `R/qtl` CSV format. For more information about this function type `?read.population`. In this function the data file can be specified as an argument.

```
> obesity.cross <- read.population("Obesity.Cross.csv")
```

Alternatively, the filename can be left blank to choose a file through the file system. The following code will bring up a window for choosing the desired file:

```
> obesity.cross <- read.population()
```

If the phenotypes are not specified in `read.population()` the default behavior is to read in all phenotypes. The functions `delete.pheno()` and `select.pheno()` can be used after reading in the data to narrow down the phenotypes for analysis. For `cape` analysis there

must be at least two phenotypes. There must be at least two because the fundamental concept of `cape` uses partial pleiotropy to refine models of epistasis. However, too many phenotypes may reduce power to detect interactions. cape handles up to 12 phenotypes, and if you have more in your data set, such as in the case of gene expression data, we suggest some form of dimension reduction prior to analysis. We offer dimension reduction through singular value decomposition (SVD), which is discussed below.

Upon being read in, the data are formatted and stored in a data object. This object is referred to as `data.obj` in the argument lists of most functions in `cape`. The main functions in this analysis return `data.obj` with any results appended to it. The structure of this object can be viewed at any time by using the core R function `str()`. For example, the data that we have just loaded is named `obesity.cross`, and its structure can be seen here:

```
> str(obesity.cross)

List of 6
 $ pheno          : num [1:204, 1:4] 58.6 49.9 56 53.7 48.7 41.1 45.4 44.1 40.4 40.1 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:204] "1" "2" "3" "4" ...
  .. ..$ : chr [1:4] "body_weight" "glucose" "insulin" "mom"
 $ geno           : num [1:204, 1:84] 1 1 1 0 1 0 1 NA 0 0 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:204] "1" "2" "3" "4" ...
  .. ..$ : chr [1:84] "1" "2" "3" "4" ...
 $ chromosome     : chr [1:84] "1" "1" "1" "1" ...
 $ marker.names   : chr [1:84] "D1Mit296" "D1Mit211" "D1Mit411" "D1Mit123" ...
 $ marker.num     : int [1:84] 1 2 3 4 5 6 7 8 9 10 ...
 $ marker.location: num [1:84] 2.08 10.59 12.62 17.67 22.88 ...
```

Individual elements of this list can be accessed by name by using the following syntax:

```
data.obj$name
```

When the cross is first loaded into `cape` it contains the following elements:

- **$pheno** - The phenotype matrix in which individuals are in rows and phenotype values are in columns.
- **$geno** - The genotype matrix in which individuals are in rows and genetic loci are in columns. Each cell of the genotype matrix contains the probability that individual i has the reference allele at that position. By default, the reference allele is the allele that is alphabetically first if the original data file used letter designations for genotypes, or "0" if the original data file used numeric genotype designations. In the example backcross, NON/Lt is the reference allele and the heterozygote state is coded as the perturbation. Thus `cape` will model the effects of NZO/HILt variants.
- **$chromosome** - The chromosome on which each genetic locus is situated

- **$marker.names** - The alphanumeric names of markers.
- **$marker.location** - The position of each locus on each chromosome
- **$marker.num** - Numeric IDs for each marker

Most functions performed on `data.obj` will add elements to this list containing the results of the analyses. Major exceptions to this are `singlescan()`, and `pairscan()`, which generate their own singlescan and pairscan objects.
In the latest version of cape there are three additional functions for reading in data set. These are `read.pheno()`, `read.geno()`, and `make.data.obj()`. These are used when the genotype data are very large and are more practically stored in a separate object called the geno.obj. File formats for these functions can be found in their respective help files.
The function `read.pheno()` initializes the data.obj. The function `read.geno()` initializes a separate geno.obj, which contains information about the genetic markers. This will be passed to cape functions separately from the data.obj. To complete the initialization of the data.obj, marker information must be transferred to the object generated by `read.pheno()`. This is done using `make.data.obj()`. After using these three functions, you will have a data.obj with phenotype data and genetic marker information, and a geno.obj with the genetic data.

```
 > obesity.cross <- read.pheno()
> geno.data <- read.geno()
> obesity.cross <- make.data.obj(obesity.cross, geno.data)
```

Manipulating the Data Object

There are a number of functions available for simple manipulation of the data object. For example, after the data have been read in, phenotypes may be removed from the phenotype matrix using `delete.pheno()`. To select specific phenotypes, use the function `select.pheno()`.

```
> obesity.cross <- select.pheno(obesity.cross,
+ phenotypes = c("body_weight", "glucose", "insulin", "mom"))
```

In `cape` covariates are imported as phenotypes and must be reassigned covariates. To code a phenotypic variable, such as sex, experimental treatment, or another environmental variable, as a covariate, use the function `pheno2covar()`. Here we convert the factor maternal environment ("mom") to a covariate.

```
> obesity.cross <- pheno2covar(obesity.cross, "mom")
```

Genetic markers can also be used as covariates using the function `geno2covar()`.
The data object can also be subset by chromosome (`select.by.chr()`) or by individual (`select.by.ind()`) before the analysis. Individuals can be selected based either on phenotype

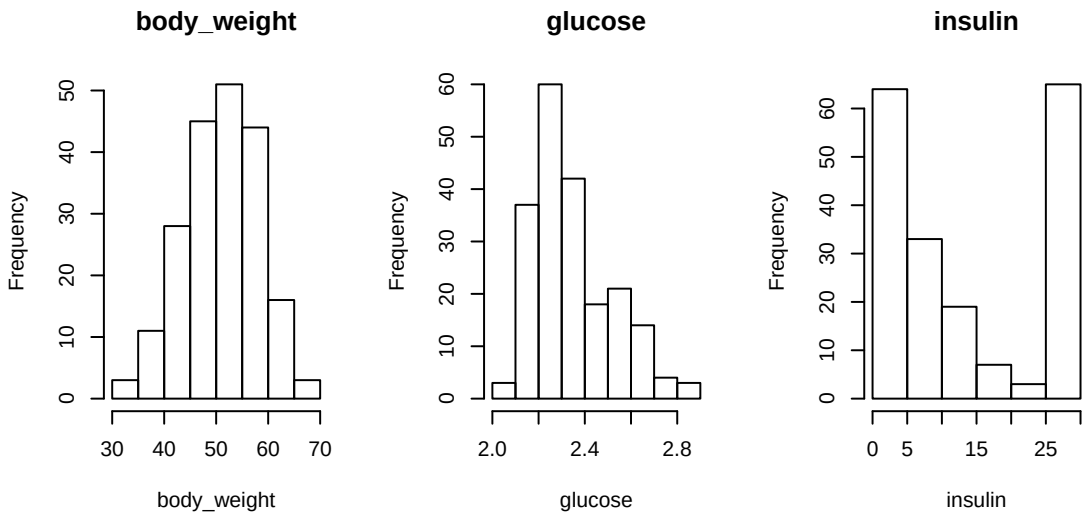
or genotype value. For example, to use only individuals with a plasma insulin level less than than 25 ng/mL:

```
> obesity.cross <- select.by.ind(obesity.cross, "pheno", "insulin < 25")
```

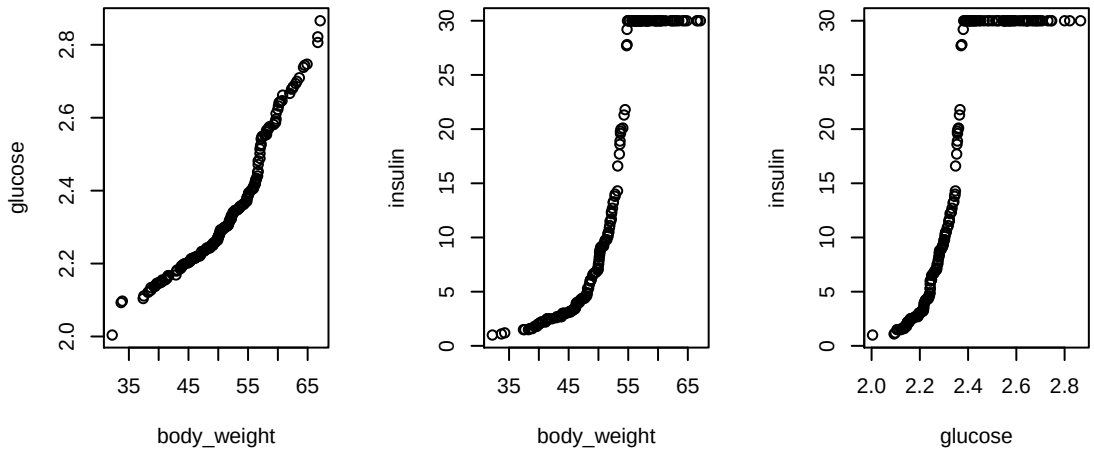
In general we recommend saving the data object after each major computation, as well as the singlescan and pairscan objects. This allows for easy re-analysis, and prevents having to repeat lengthy calculations if there is an unexpected problem. Saving can be done with either `save()` or `saveRDS()`.

Examining the Data

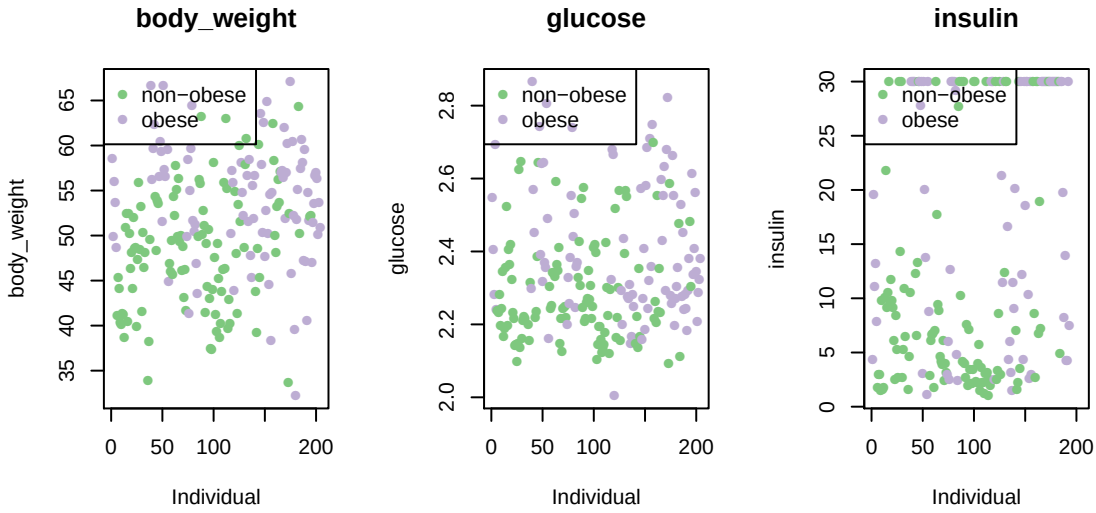
Before proceeding with an analysis it is recommended that the data be examined by eye. The R package `qtl` has sophisticated plotting tools for examining genetic cross data,which we do not try to duplicate here. It is straightforward enough, however, to examine phenotype distributions for normality, batch effects and other quality control issues. Using the function `histPheno()`, we can look as the distribution of each phenotype:



While body weight looks relatively normally distributed, glucose and insulin have obviously non-normal distributions.
Examination of the Q-Q plots of pairs of phenotypes, using the function `qqPheno()`, can reveal phenotyping errors and other pathologies. Here we see a threshold and ceiling effect in the relationship between the distributions of insulin and the other two phenotypes.



We can also examine the phenotypes by individual using `plotPheno()` to get an idea of systematic bases in the data.

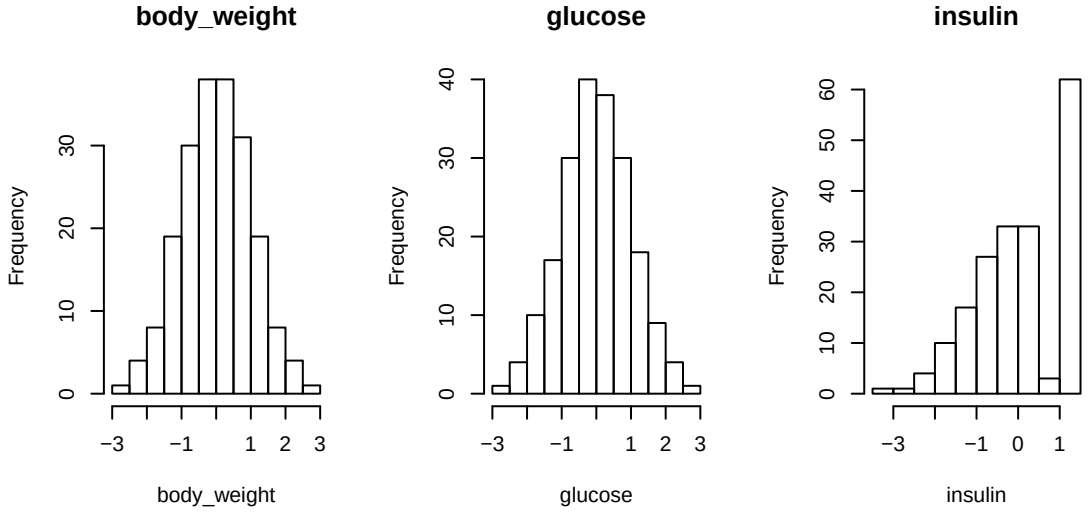


In general we recommend mean centering and normalizing all phenotypes before proceeding with the analysis. Phenotype normalization can be achieved through log transformation, quantile normalization, or another method before the analysis. The function `norm.pheno()` uses quantile normalization to fit the phenotypes to a normal distribution. Briefly, this process sorts the values of the phenotype and replaces each with a corresponding value drawn from a normal distribution with the same standard deviation and mean as the original distribution. Mean centering subtracts the mean phenotype value from each phenotype value yielding a distribution centered around 0.

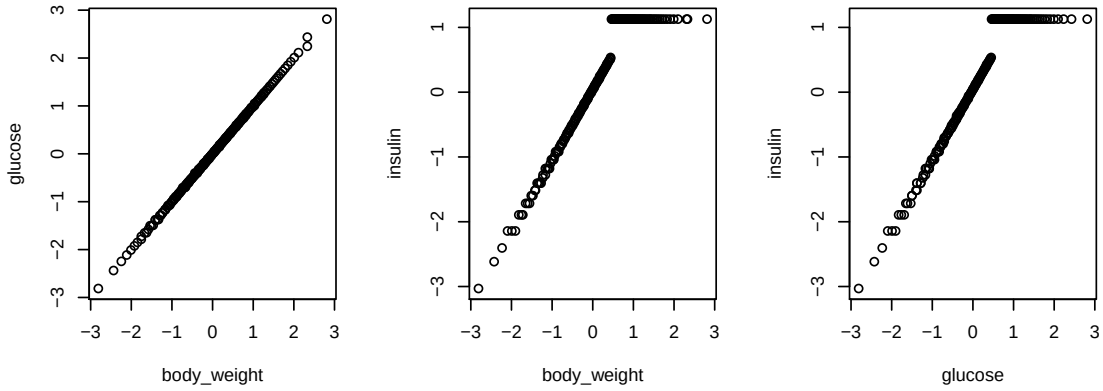
```
> obesity.cross <- norm.pheno(obesity.cross, mean.center = TRUE)
```

Plotting the histograms of the normalized data confirms the normalization. Only insulin

still has a ceiling effect, which cannot be removed by normalization because rank cannot be determined among equal values.



We can also see that the Q-Q plots from before show the example phenotypes have been converted to the same distribution. The ceiling effect is still visible in the insulin measurement, but this cannot be removed through normalization, and overall, the distribution is more similar to those of the other phenotypes than before normalization. Knowing that this ceiling effect is present will be important in interpreting the results of the analysis.



At this point, if we decide to exclude insulin from the analysis because its distribution cannot be normalized, we can simply remove it from the data object using `delete.pheno()`.

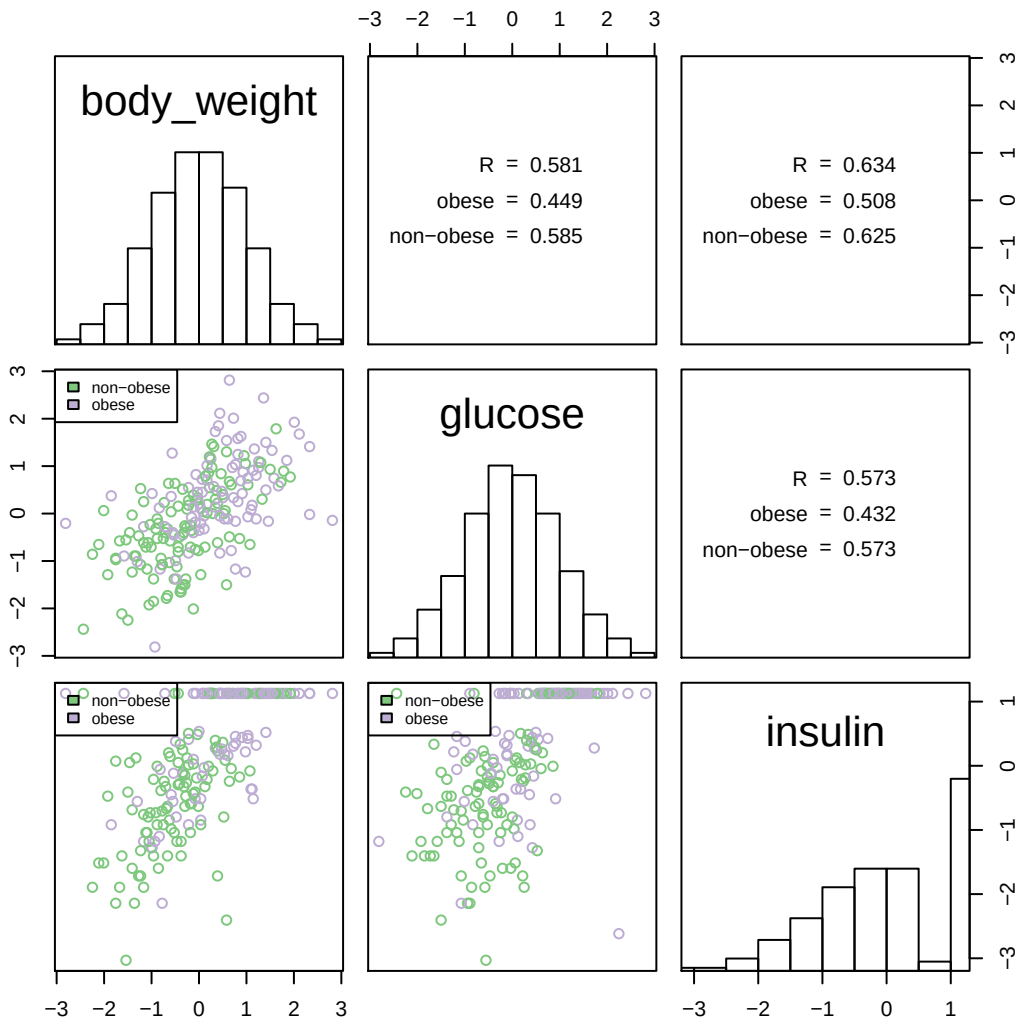
```
> obesity.cross <- delete.pheno(obesity.cross, phenotypes = "insulin")
```

For the example here, however, we will include it in the analysis.

A Note on Phenotype Selection

Phenotype selection is an important component of the `cape` analysis and should be given considerable thought. This method relies on the selection of two or more phenotypes that have common genetic factors but are not identical across all individuals. Such phenotypes may describe multiple aspects of a single complex trait, such as obesity or diabetes, and may encompass a combination of molecular phenotypes, such as plasma glucose levels, and phenotypes, such as body weight, that are measured at the organismal level. The central assumption of this method is that different genetic interactions found for a single gene pair in the context of different phenotypes represent multiple manifestations of a single underlying gene network. By measuring the interactions between genetic variants in different contexts we can gain a clearer picture of the network underlying statistical epistasis (Carter et al., 2012).

The phenotypes in the Reifsnyder (2000) data set are ideal for the `cape` analysis. They measure different aspects of the diabetes and obesity, two complex traits that are known to be related biologically and are highly correlated. The phenotypes themselves range from molecular phenotypes to organismal phenotypes. By examining the correlations between phenotypes, we can see that the phenotypes measured in this experiment are correlated, but not identical across all individuals. Ideally, phenotypes used in cape should have a Pearson correlation coefficient r between 0.4 and 0.8.



Note that all mice in the example backcross are male. For multisex populations, sex is usually a covariate and correlation should be assessed for each sex separately.

Decomposing the Phenotypes

Although `cape` can find genetic variants associated with raw phenotypes, the analysis was designed to work on composite traits called "eigentraits." Eigentraits are calculated by factoring the matrix of phenotypes by singular value decomposition (SVD):

$$Y = U \cdot V \cdot W^T$$

Where Y is a matrix containing one column for each mean-centered, normalized phenotype and one row for each individual. If Y contains more individuals than phenotypes, the U

matrix has the same dimensions as Y with each column containing one eigentrait. V contains the singular values, and W^T contains the right singular vectors in rows. See Carter et al. (2012) for more details.

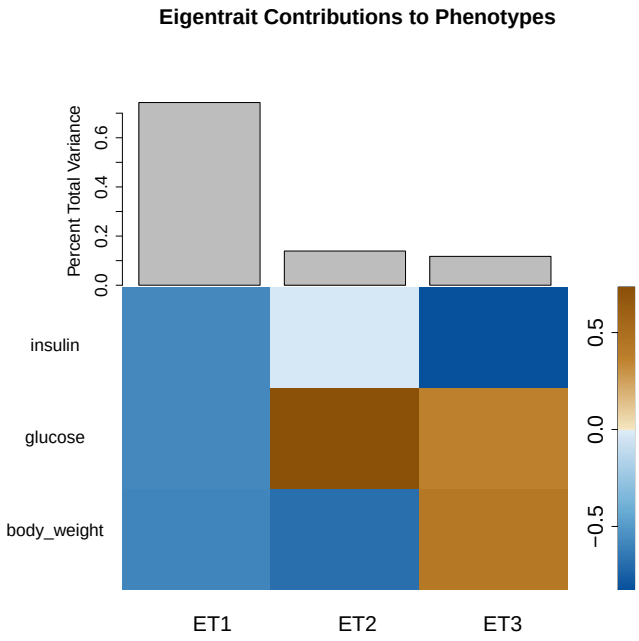
The SVD de-correlates the phenotypes concentrating phenotypic features into individual eigentraits. One benefit of this process is that variants that are weakly correlated to several phenotypes due to common underlying processes may be strongly correlated to one of the eigentraits. This eigentrait captures the information of the underlying process, making strong main effects distributed between phenotypes easier to detect and identify as potential interaction loci and/or covariates. Thus, analysis of eigentraits is recommended over the analysis of raw traits.

To decompose the phenotypes to eigentraits use the function `get.eigentraits()` If the phenotypes have not already been mean centered and normalized, this step can be performed here. In this example, we do not carry out these steps, as they have already been performed.

```
> obesity.cross <- get.eigentraits(obesity.cross, scale.pheno = FALSE,
+ normalize.pheno = FALSE)
```

This function performs the SVD and returns the matrix of eigentraits in a new element of the list called `ET`. The result of the decomposition can be viewed with the function `plotSVD()`.

```
plotSVD(obesity.cross, orientation = "vertical")
```



In the example illustrated here, the first eigentrait captures more than 70% of the variance in the three phenotypes. This eigentrait describes the processes by which body weight, glucose

levels, and insulin levels all vary together. The correlations between obesity and risk factors for obesity, such as elevated insulin and fasting glucose levels are well known (Permutt et al., 2005; Das and Elbein, 2006; Haffner, 2003). The second eigentrait captures nearly 20% of the variance in the phenotypes. It captures the processes through which glucose and body weight vary in opposite directions. This eigentrait may be important in distinguishing the genetic discordance between obesity and diabetes. While obesity is a strong risk factor for diabetes, not all those who are obese have diabetes, and not all those with diabetes are obese (Permutt et al., 2005; Burcelin et al., 2002).

The third eigentrait is less interpretable biologically, as it describes the divergence of blood glucose and insulin levels. It may represent a genetic link between glucose and body weight that is non-insulin dependent. Because we are primarily interested in the connection between diabetes and insulin, we will use only the first two eigentraits for the analysis. In many cases in which more than two phenotypes are being analyzed, the first two or three eigentraits will capture the majority of the variance in the data and capture obvious features. Other eigentraits may capture noise or systematic bias in the data. Often the amount of total variance captured by such eigentraits is small, and they can be removed from the analysis.

Ultimately, there is no universal recipe for selecting which eigentraits should be included in the analysis, and the decision will be based on how the eigentraits contribute to the original phenotypes and how much variance in the data they capture.

To select eigentraits for the analysis, use the function `select.eigentraits()`. Here we select the first two eigentraits.

```
> obesity.cross <- select.eigentraits(obesity.cross, traits.which = c(1,2))
```

Kinship Corrections

`cape` includes a correction for relatedness among individuals called a kinship correction Kang et al. (2008), which can be applied in the single-variant scan and the pairwise scan (described below). In the singlescan, relatedness matrices among individuals are calculated using a leave-one-chromosome-out (LOCO) method. For each chromosome (C) being tested, the relatedness matrix (K_C) is calculated using the markers on the remaining chromosomes as follows:

$$K_C = \frac{G_C \times G_C^T}{n},$$

where G_C is the genotype matrix with markers on chromosome C removed and n is the number of markers in G_C. This matrix is then used to correct the genotype and phenotype matrices for kinship using hierarchical linear models Kang et al. (2008); Gelman and Hill (2006).
In the pairwise scan, the same methods are performed except that relatedness matrices are calculated leaving two chromosomes out depending on where the two markers in the tested pair reside.

Relatedness matrices can be calculated using all markers or only the markers being tested. For very large genotype matrices, there is also the option of sampling the relatedness matrix. This is less accurate than calculating the relatedness matrix directly, but can save large amounts of computation time.

Single-Variant Scan

Once the eigentraits for the analysis have been selected, the single-locus scan is run to investigate how individual markers are associated with each eigentrait. Note that this scan performs a linear regression at each marker. A more sophisticated single-locus analysis can be performed by `R/qtl` (Broman et al., 2003). This single-marker scan in `cape` performs the following regression for each locus on each eigentrait:

$$U_i^j = \beta_0^j + x_i \beta^j + \epsilon_i^j$$

The index i runs from 1 to the number of individuals, and j runs from 1 to the number of eigentraits or phenotypes. x_i is the probability of the presence of the reference allele for individual i at locus j. The purpose of this scan is two-fold:

- In large data sets the number of possible variant pairs may be too large to test exhaustively. The single-variant scan can be used as a filtering step to choose variants that will be included in the pair scan.
- Large main effects can obscure interactions. In sufficiently powered studies, conditioning on the large QTL can aid in the discovery of interactions, variants with large main-effects can be used as covariates in the pair scan.

The function `singlescan()` runs the single-marker scan.

```
> obesity.singlescan <- singlescan(obesity.cross, n.perm = 100,
+ covar = "mom", scan.what = "eigentraits", alpha = c(0.01, 0.05),
+ verbose = FALSE, use.kinship = FALSE, overwrite.alert = FALSE,
+ run.parallel = FALSE, n.cores = 2)
```

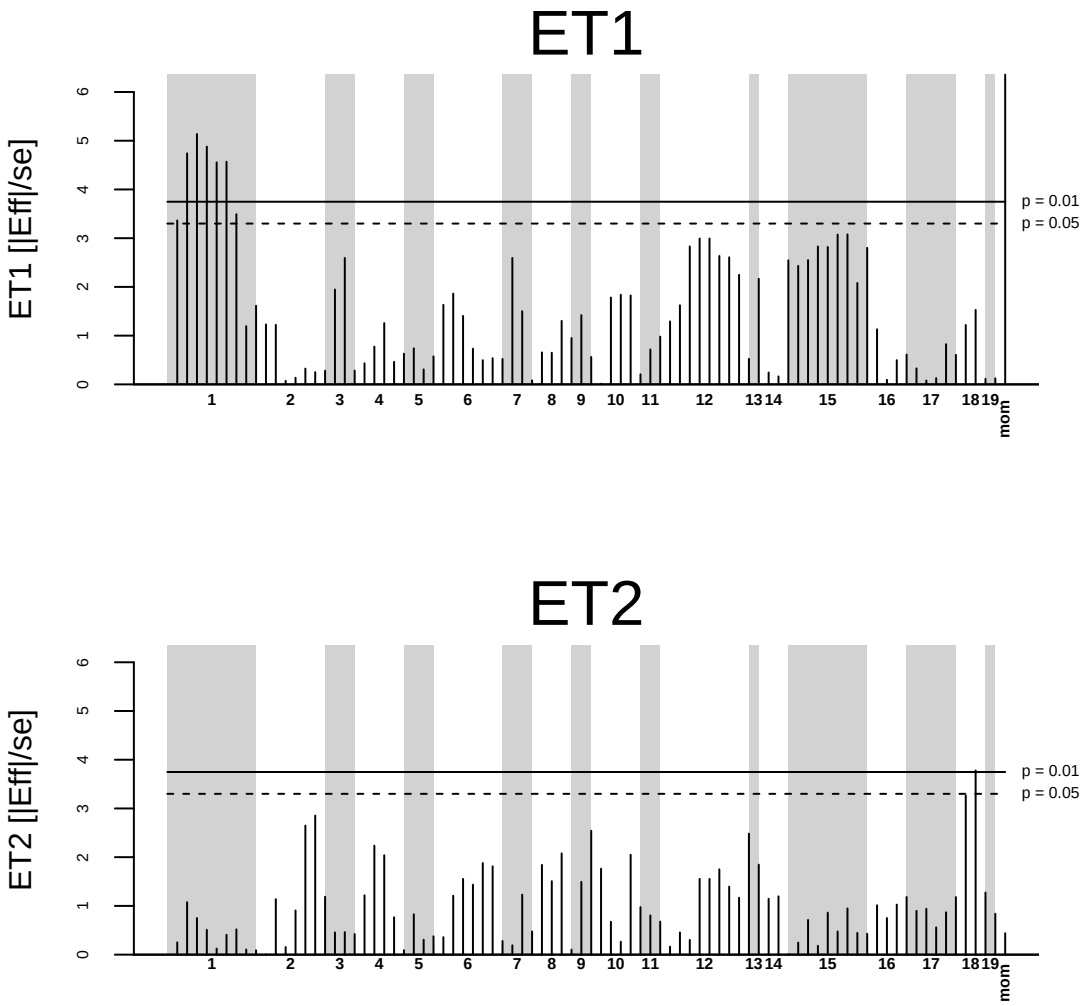
```
10%.20%.30%.40%.50%.60%.70%.80%.100%.10%.30%.50%.60%.80%.100%.20%.50%.70%.100%.20%.50%.70%.
```

The argument `alpha` takes a vector of alpha values for which significance thresholds will be calculated. These values are used primarily for plotting purposes.
Covariates to be used in the single-marker scan should be specified through the argument `covar`. These covariates should have already been moved to the marker matrix by `pheno2covar()` or `marker2covar()`. An additional argument `scan.what` determines whether "eigentraits" or "raw.traits" will be used as the dependent variable in the regression.

The single-marker scan currently does not support markers on sex chromosomes. Because the X chromosome is hemizygous in males, sex differences in phenotype can lead to false associations, and markers on this chromosome require special consideration (Broman et al., 2006). Before the single-marker scan is performed, any markers on the X and Y chromosomes

are removed from the data object. The results of the single variant scan can be visualized with the function `plotSinglescan()`.

```
> plotSinglescan(data.obj = obesity.cross, singlescan.obj = obesity.singlescan,
+ mark.chr = TRUE, mark.covar = FALSE)
```



In this figure the t-statistic (β/σ) of each marker is plotted as a vertical line. Results for both eigentraits are shown here as ET1 and ET2, and chromosome numbers are written along the x axis. If filtering of markers for the pair-wise scan is desired, an effect size cutoff can be specified using `select.markers.for.pairscan()`. If a specific number of markers is desired, `select.markers.for.pairscan()` can determine the threshold at which this number of markers is included in the pair scan. Setting a threshold for inclusion is useful for large data sets for which it is impractical to test all marker pairs. Here we use all markers

in the pairwise scan. In this example we will not filter the markers.

A kinship correction can be implemented in the singlescan simply by setting the argument `use.kinship` to TRUE.

Covariate Selection

As mentioned previously, conditioning on large main effects may aid in the discovery of interactions. Therefore `cape` allows specification of genetic markers as covariates. This specification is recommended if there are markers with particularly large effect sizes that may obfuscate weak interactions. In an organism in which variants segregate relatively independently, such as in *Drosophila* (Mackay et al., 2012), it may be useful to select individual markers with very high effect sizes as covariates. In a population with more correlation between adjacent markers, more care should be given to this process. In a mouse F2 intercross or backcross, on the other hand, markers on a single chromosome tend to be linked, and groups of markers may be significantly associated with a phenotype due to linkage with a causative locus. In this case, markers from a single linkage block carry partially redundant information and, if treated as independent covariates, will confound interactions between makers in the linkage block and other truly independent markers. Thus, if markers in the study population are linked, it is recommended that any marker covariates be set manually and that only one covariate should be selected from each linkage block.

Setting covariates, as well as the filtering of markers for the pair-wise scan can be done with the function `select.markers.for.pairscan()`. When calling this function, if thresholding the markers based on effect size, set `use.pairs.threshold` to TRUE, and `pairscan.thresh` to the desired standardized effect size threshold. Alternatively, this function can be used to select a threshold based on a desired number of markers to test in the pair-wise scan. The target number of markers desired is set in `num.markers`. The function then searches through multiple thresholds to find a linearly independent matrix holding the number of desired markers. It starts the search at the threshold set in `start.thresh` and stops when it finds a threshold resulting in `num.markers` plus or minus the `tolerance`. This process is slow, and it is recommended that the starting threshold be set as close to the estimated correct threshold as possible and run only once to find the desired threshold. After the first run, the threshold can be set manually.
In addition to selecting markers based on an effect size threshold, `select.markers.for.pairscan()` can also set specific markers as covariates through the argument `specific.markers`. This argument takes a vector of strings and sets each one as a covariate.

```
> obesity.cross <- select.markers.for.pairscan(data.obj = obesity.cross,
+ singlescan.obj = obesity.singlescan)

82 markers were selected for the pairscan.
This makes 3321 possible pairs.
```

Pairwise Scan

The purpose of the pairwise scan is to find interactions, or epistasis, between variants. The epistatic models are then combined across phenotypes or eigentraits to infer a parsimonious network that takes data from all eigentraits into account.

To find epistatic interactions `pairscan()` tests the following model for each variant 1 and 2:

$$U_i^j = \beta_0^j + \underbrace{\sum_{c=1}^{2} x_{c,i}\beta_c^j}_{\text{covariates}} + \underbrace{x_{1,i}\beta_1^j + x_{2,i}\beta_2^j}_{\text{main effects}} + \underbrace{x_{1,i}x_{2,i}\beta_{12}^j}_{\text{interaction}} + \epsilon_i^j$$

The terms in this equation are the same as those in the equation for the single-variant scan except for the addition of the term for the interaction between the two variants being tested. This additional term brings further complications to the model, and restricts which markers can be tested. Because many markers, including covariates, may be included in the model, we need to be careful about including only markers that are linearly independent of each other. Linear dependence between markers occurs when two markers are fully linked and therefore perfectly correlated. The two markers provide the same genetic information, and one can be discarded without loss of information. Before running the pair scan, it is important that we reduce the genetic matrix to only markers that are linearly independent of one another. This step is performed by the function `select.markers.for.pairscan()`. It calculates the correlation between all pairs of markers. If any are found to be perfectly correlated, the first marker is removed.

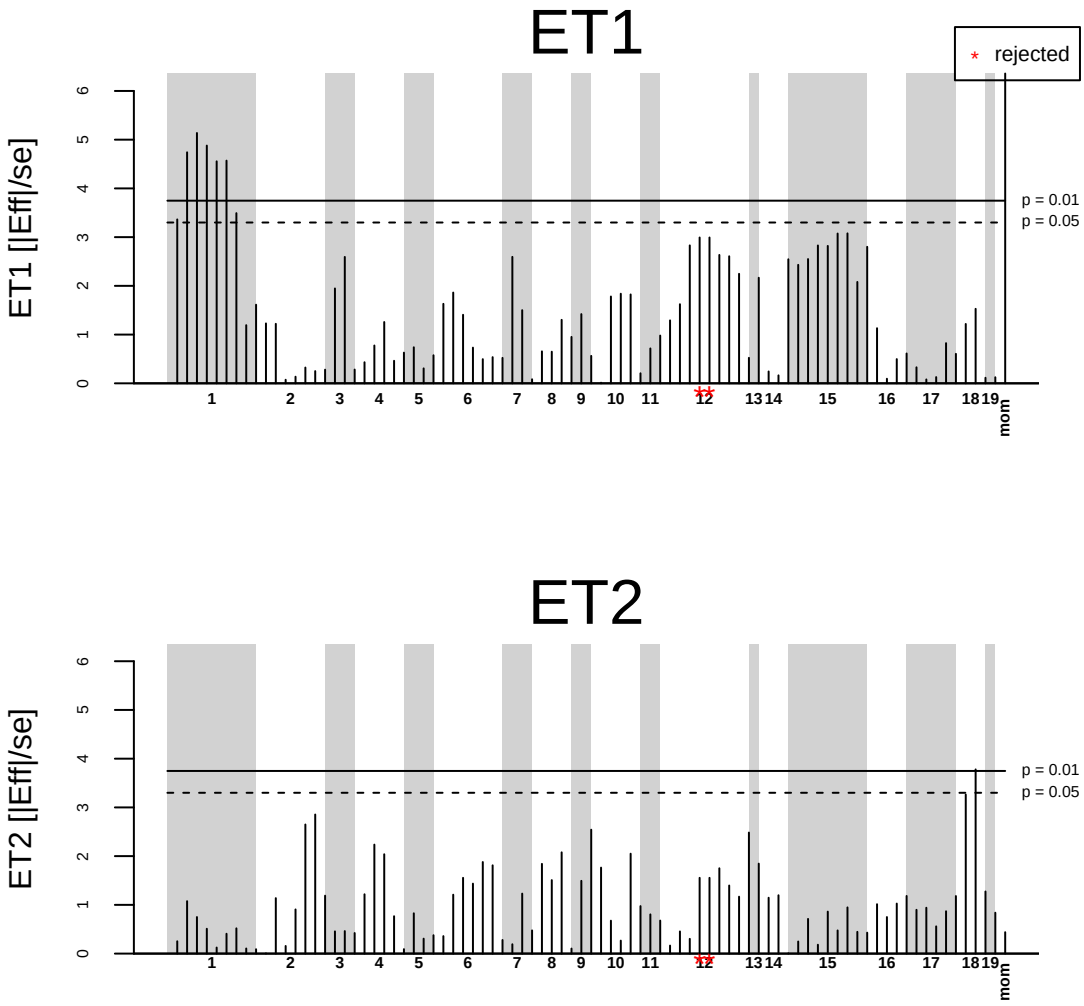
As mentioned above, `select.markers.for.pairscan()` also optionally filters markers for inclusion in the pair scan by standardized effect size. This is useful in large crosses in which it may be impossible to test all possible pairs of markers in a reasonable amount of time.

```
obesity.cross <- select.markers.for.pairscan(data.obj = obesity.cross,
singlescan.obj = obesity.singlescan)
```

The number of markers removed is printed to the screen, and the names of these markers are written to the file `markers.removed.txt` in the current working directory. This filtering step adds one element to the data object: a filtered genotype matrix called `geno.for.pairscan` which holds the markers that will be used in the pairscan.

The results of the filtering can be visualized with `plotSinglscan()`. This visualization can be helpful in seeing the locations of discarded markers, especially if many have been removed. To indicate which markers have been selected for the pair scan, set `show.selected.markers` to `TRUE`. To indicate which markers have been rejected from the pair scan, set `show.rejected.markers` to `TRUE`.

```
> plotSinglescan(data.obj = obesity.cross, singlescan.obj = obesity.singlescan,
+ mark.chr = TRUE, show.rejected.markers = TRUE, standardized = TRUE)
```



After ensuring that all markers are linearly independent and thresholded satisfactorily, the pairscan can be run using `pairscan()`.

```
> obesity.pairscan <- pairscan(data.obj = obesity.cross, covar = "mom",
+ scan.what = "eigentraits", total.perm = 1000, min.per.genotype = 6,
+ verbose = FALSE, overwrite.alert = FALSE, n.cores = 2)
```

The arguments here are familiar from `singlescan()`, with the exception of two additional arguments. `min.per.genotype` and `max.pair.cor` are thresholds that prevent highly correlated markers from being tested in pairs. Only one of these arguments can be set depending on whether the genotypes in the data set are continuous or discrete. If the genotypes are discrete, `min.per.genotype` can be set to prevent empty cells in genotype matrices. In an intercross, there are three possible genotypes at each marker, giving a total of nine possible genotypes for the pair. For example, for two markers each with the genotypes

AA, AB, and BB, the pairwise genotypes are the following:

	AA	AB	BB
AA			
AB			
BB			

In a backcross each marker only has two possible genotypes, AA and AB, yielding four possible pairwise genotypes. In both cases, insufficient representative individuals of single genotypes indicates that the two markers in question are linked. This linkage leads to false associations in both the effects and permutations. False associations in permutations results in a heavy-tailed null distribution, which artificially inflates the threshold of significance and reduces power to find true interactions. Because of this effect, linked marker pairs are not tested. The threshold used to determine insufficient recombination between markers is given by `min.per.genotype`. The default behavior is to reject any marker pair for which there are fewer than six individuals in any of the genotype cells. This value can be adjusted, but caution should be used in interpreting the results if `min.per.genotype` is very low or 0.

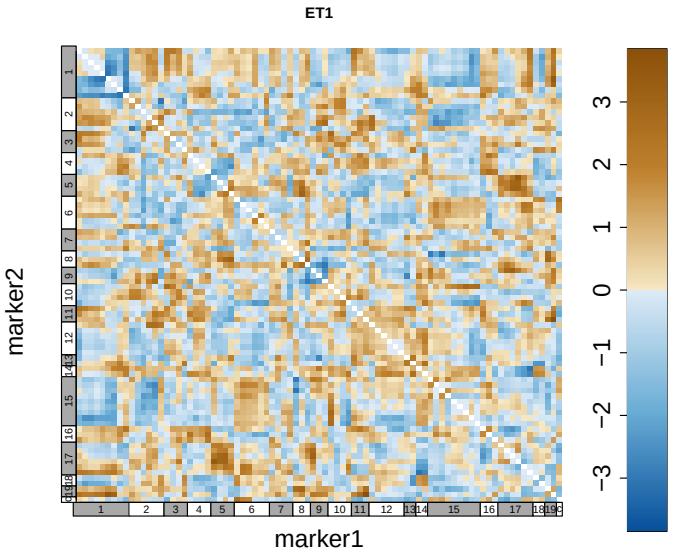
Alternatively, if genotypes are coded continuously, the argument `max.pair.cor` can be set to establish an upper threshold on the Pearson correlation between marker pairs. As in the case with discrete genotypes, testing highly correlated markers against each other can lead to false associations. Either `max.pair.cor` or `min.per.genotype`, but not both, must be set.

The `total.perm` argument in `pairscan()` sets the total number of permutations performed over all marker pairs. To generate a null distribution for the pair-wise tests, CAPE performs permutation tests in the following way: First singlescan is run on the permuted phenotypes. Then the N markers with the largest effect sizes are selected, where N is the same number of markers being tested in the pairwise scan. This ensures that if there are very few markers being tested in the pairwise scan, we do not generate a biased null distribution by only performing permutation tests repeatedly on these markers. A pairwise scan is then performed on these top N markers using the permuted phenotype. To calculate empirical p values, the permutations are combined across all markers to result in one large null distribution. Thus, large null distributions can be achieved with relatively few permutations per pair. This is useful, as even with relatively sparse genotyping the number of tests performed can be large and take many hours to perform. In this demonstration we only run 1000 permutations, but in actual use, we recommend at least 500,000 permutations. Low numbers of permutations tend to result in many false positives.

The results of the pair scan can be plotted with `plotPairscan()`. This plot shows the resulting β/σ for each pair of markers for ET1. Gray and white bars show the boundaries of the chromosomes.

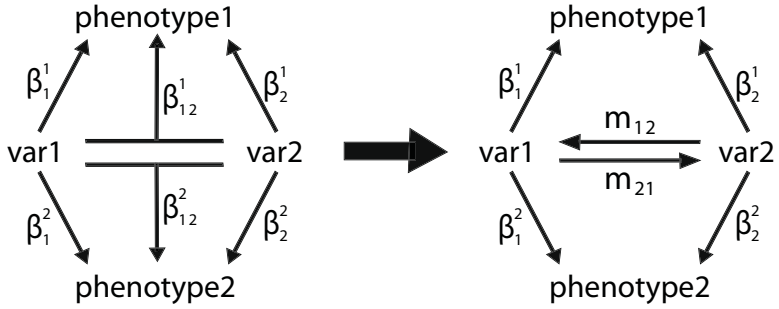
```
> plotPairscan(data.obj = obesity.cross, pairscan.obj = obesity.pairscan,
+ phenotype = "ET1", pdf.label = "Pairscan_Regression.pdf")

null device
          1
```



Combined Analysis for Detection of Interactions

From the pair scan, each pair of markers 1 and 2 receives a set of β coefficients describing the main effect of each marker on each eigentrait j (β_1^j and β_2^j) as well as the interaction effect of both markers on each eigentrait (β_{12}^j) (See figure below). The central idea of `cape` is that these coefficients can be combined across eigentraits and reparameterized to calculate how each pair of markers influences each other directly and independently of eigentrait.



The first step in this reparameterization is to define two new parameters (δ_1 and δ_2) in terms of the interaction coefficients. δ_1 can be thought of as the additional genetic activity

of marker 1 when marker 2 is present. Together the δ terms capture the interaction term, and are interpreted as the extent to which each marker influences the effect of the other on downstream phenotypes. For example, a negative δ_2 indicates that the presence of marker 2 represses the effect of marker 1 on the phenotypes or eigentraits. The δ terms are related to the main effects and interaction effects as follows:

$$\begin{bmatrix} \beta_1^1 & \beta_2^1 \\ \beta_1^2 & \beta_2^2 \\ \vdots & \vdots \end{bmatrix} \cdot \begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} \beta_{12}^1 \\ \beta_{12}^2 \\ \vdots \end{bmatrix} \tag{1}$$

In multiplying out this equation, it can be seen how the δ terms influence each main effect term to give rise to the interaction terms independent of phenotype.

$$\beta_1^j \delta_1 + \beta_2^j \delta_2 = \beta_{12}^j \tag{2}$$

If $\delta_1 = 0$ and $\delta_2 = 0$, there are no addition effects exerted by the markers when both are present. Substitution into the equations above shows that the interaction terms β_{12}^j are 0 and thus the interaction terms have no effect on the phenotype.

Alternatively, consider the situation when $\delta_1 = 1$ and $\delta_2 = 0$. The positive δ_1 indicates that marker 1 should exert an additional effect when marker 2 is present. This can be seen again through substitution into equation 2:

$$\beta_j^1 = \beta_{12}^j$$

These non-zero terms show that there is an interaction effect between marker 1 and marker 2. The positive δ_1 indicates that this interaction is driven through an enhanced effect of marker 1 in the presence of marker 2.

The δs are calculated by solving for equation 1 using matrix inversion:

$$\begin{bmatrix} \delta_1 \\ \delta_2 \end{bmatrix} = \begin{bmatrix} \beta_1^1 & \beta_2^1 \\ \beta_1^2 & \beta_2^2 \\ \vdots & \vdots \end{bmatrix}^{-1} \cdot \begin{bmatrix} \beta_{12}^1 \\ \beta_{12}^2 \\ \vdots \end{bmatrix}$$

This inversion is exact for two eigentraits, and `cape` implements pseudo-inversion for up to 12 eigentraits.

The δ terms are then translated into directed variables defining the marker-to-marker influences m_{12} and m_{21}. Whereas δ_2 described the change in activity of marker 2 in the presence of marker 1, m_{12} can be thought of as the direct influence of marker 2 on marker 1, with negative values indicating repression and positive values indicating enhancement. The terms m_{12} and m_{21} are self-consistent and defined in terms of δ_1 and δ_2:

$$\delta_1 = m_{12}(1 + \delta_2),\ \delta_2 = m_{21}(1 + \delta_1)$$

Rearranging these equations yields the solutions:

$$m_{12} = \frac{\delta_1}{1+\delta_2},\ m_{21} = \frac{\delta_2}{1+\delta_1}.$$

These directed influence variables provide a map of how each marker influences each other marker independent of phenotype. The significance of these influences is determined through standard error analysis on the regression parameters (Bevington, 1994; Carter et al., 2012). This step is particularly important as matrix inversion can lead to large values but larger standard errors, yielding insignificant results. As an example, the variance of m_{12} is calculated by differentiating with respect to all model parameters:

$$\sigma^2_{m_{12}} \cong \sum_{ij} \sigma^2_{\beta_i^j} \left(\frac{\partial m_{12}}{\partial \beta_i^j} \right)^2 + 2 \sum_{i<k, j<l} \sigma^2_{\beta_i^j \beta_k^l} \left(\frac{\partial m_{12}}{\partial \beta_i^j} \right) \left(\frac{\partial m_{12}}{\partial \beta_k^l} \right)$$

In this equation, the indices i and k run over regression parameters and j and l run from 1 to the number of traits. The calculations of the δ and the m terms, as well as the error propagation are performed by the function `error.prop()`.

```
> obesity.cross <- error.prop(data.obj = obesity.cross,
+ pairscan.obj = obesity.pairscan,
+ perm = FALSE, verbose = FALSE, n.cores = 2)
```

This function is applied to both the results from the pairwise scan, as well as the permutations of the pairwise scan for later calculation of p values. To apply the calculations to the pairwise scan permutations, set `perm = TRUE`.

```
> obesity.cross <- error.prop(data.obj = obesity.cross,
+ pairscan.obj = obesity.pairscan,
+ perm = TRUE, verbose = FALSE, n.cores = 2)
```

For large scans it might be desirable to observe the progress of the calculations. This can be done by setting `verbose = TRUE`. After these calculations have been performed the results of the permutation testing can be used to calculate empirical p values for each of the variant-to-variant effects.

```
> obesity.cross <- calc.p(data.obj = obesity.cross,
+ pairscan.obj = obesity.pairscan, pval.correction = "fdr",
+ n.cores = 2)
```

This function also adjusts the empirical p values for multiple testing. The default correction is Holm's stepdown procedure (Holm, 1979). Two other methods, false discovery rate (FDR) (Benjamini and Hochberg, 1995) and local false discovery rate (lFDR) (Liao et al., 2004) are also available. The latter methods of correction for multiple testing are less stringent than the Holm's step-down procedure. The function `calc.p()` adds two elements to the data object. The purpose and returned results of each function is summarized below. For tables of all functions, see the Tables of Functions section at end of this document.

Functions for Parsing Pair Scan Results

Function	Behavior	Result Label	Structure	Columns
error.prop	propagates errors of coefficients calculated in the pair scan	var.to.var.influences or var.to.var.influences.perm	matrix	marker1, marker2, m12, m21, m12 σ, m21 σ
calc.p	calculates empirical and corrected p values or fdr of interactions using permutations	empirical.p	list of 2 matrices (m12, m21)	marker1, marker2, source, target, influence coefficient, influence σ, empirical p value, adjusted p value or fdr

The final step in calculating the network of directed influences is to translate the effect of each marker on the eigentraits to effects on the original phenotypes. The effects of variants on eigentraits are defined in terms of the interaction coefficients m_{12} and m_{21} as well as the main effect of each variant β_i^j. To translate these effects to be in terms of the original phenotypes, the coefficient matrices are multiplied by the singular value matrices $V \cdot W^T$. With two phenotypes and two eigentraits this conversion results in no loss of information. It should also be noted that the translation back to phenotype space does not affect the variant-to-variant influences.

The translation to phenotype space is performed by the function `direct.influence()`.

```
> obesity.cross <- direct.influence(data.obj = obesity.cross,
+ pairscan.obj = obesity.pairscan, pval.correction = "fdr")
```

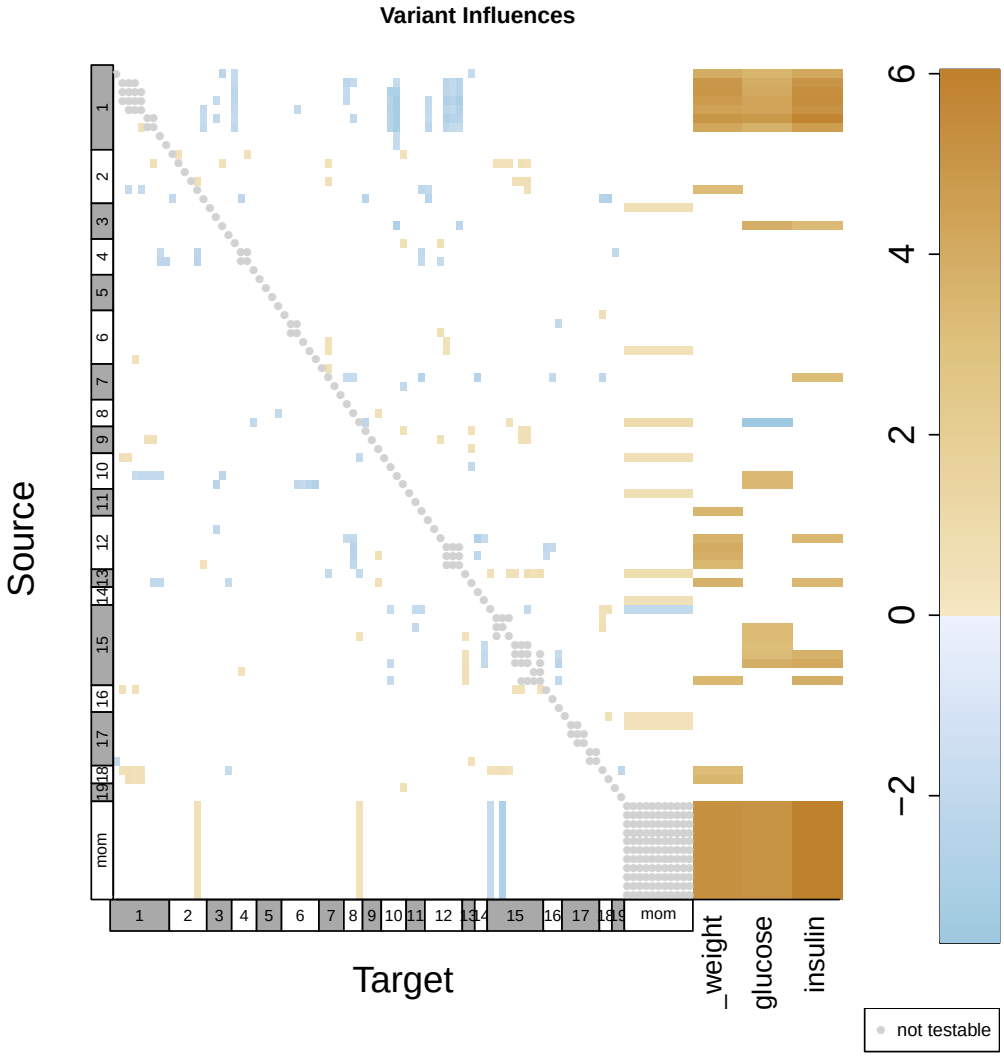
In these calculations each marker is assigned multiple direct influences on each phenotype dependent on the marker it was paired with. To reduce the results to a single direct influence of each marker on each phenotype, `direct.influence()` selects the maximum influence observed for each marker. These effects are stored in `max.var.to.pheno.influence`. All effects are accessible in the element `variant.to.phenotype.influences`. To save memory, data from the permutations are not saved by default; however, if `save.permutations` is set to `TRUE` all data are saved in the object `permutation.data.RData`. For details on the elements added to `obesity.cross` and those saved in `permutation.data.RData`, see the Tables of Functions section at end of this document.

The function `direct.influence()` also performs correction for multiple testing for each of the influences. Holm step-down correction (Holm, 1979), FDR Benjamini and Hochberg (1995), and local FDR (Liao et al., 2004) are all available as correction methods.

The marker-to-marker and marker-to-phenotype influences can be plotted with the function `plotVariantInfluences()`. This function plots the adjacency matrix of the final network. It shows all significant influences between variants and between variants and phenotypes.

```
plotVariantInfluences(obesity.cross, p.or.q = 0.05,
```

```
standardize = TRUE, not.tested.col = "lightgray",
pheno.width = 8)
```



There are several arguments in this function to note. The argument `p.or.q` takes in the p, q, or local fdr value at which interactions are considered significant. Only significant interactions are plotted. Non-significant interactions are colored white. It should be noted that some of the white boxes have gray dots in them. These dots indicate that these pairs were not tested for interactions because they were filtered out due to insufficient representation of individual phenotypes. This usually occurs because the markers are linked and there is low recombination between them. These pairs are marked with dots to distinguish them from pairs that were tested but did not have significant interactions between them. The color of the dots can be changed with the argument `not.tested.col`. The default is `lightgray`. This can be changed to `FALSE` or `NA` if no distinction between not-tested and not-significant

is desired. Gray and white bars along the margins of the plot indicate the boundaries of the chromosomes.

It should also be noted that not all of the original markers from the single-locus scan are represented in the final figure. This figure is showing only those markers that were included in the pair scan.

The argument `scale.effects` allows application of a function, either log_{10} or square root to the values in the adjacency matrix. These transformations can help increase the contrast between markers. And finally, the argument `pheno.width` specifies how wide the main effects should be relative to the interaction effects for better visualization.

A table of significant influences can be written to a file using the function `writeVariantInfluences()`.

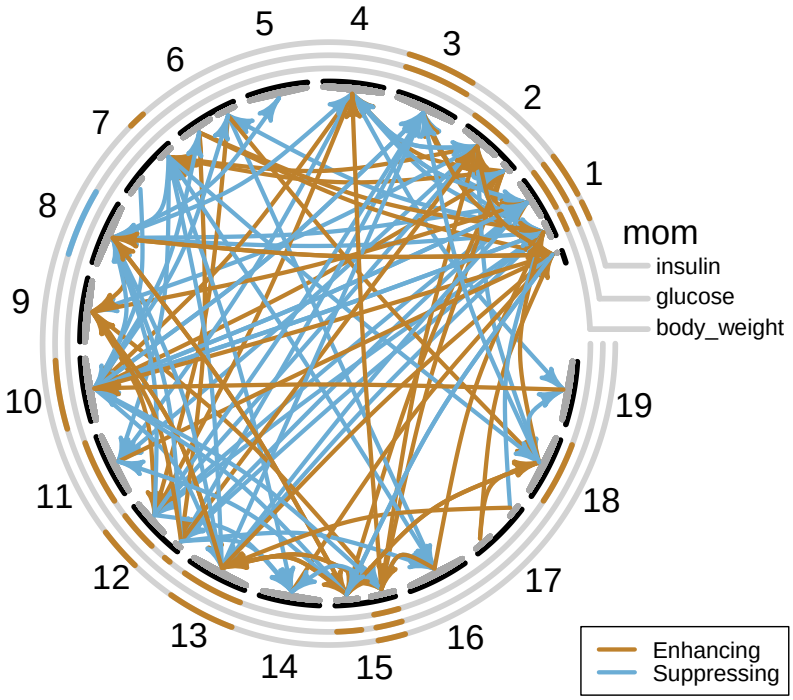
```
writeVariantInfluences(obesity.cross, p.or.q = 0.05,
filename =  "Significant.Influences.txt")
```

Interpretation of Results

The table of variant-to-variant and variant-to-phenotype influences is the primary output of `cape`. The function for plotting results, `plotVariantInfluences()`, plots an asymmetric adjacency matrix, which is useful for identifying patterns in variant interactions.

Another useful visualization of the network can be plotted using `plotNetwork()`. This function plots the chromosomes in a circle and shows interactions as arrows between regions on chromosomes (see figures below). Main effects are plotted in the circles around the chromosomes. Before plotting the network in this format, the function `get.network()` must be be run to create a network object. The function `get.network()` network offers the option of condensing markers based on their linkage. To group markers into linkage blocks `cape` calculates a correlation matrix for all markers on each chromosome. This similarity matrix is used as an adjacency matrix to construct a weighted network depicting the similarity between all pairs of markers on a single chromosome. Using the fastgreedy community detection algorithm (Clauset et al., 2004) in R/igraph (Csardi and Nepusz, 2006), `cape` then calculates the community membership of the vertices in the network. Adjacent markers in the same community are assigned to a single QTL region.
Finally, the argument `p.or.q` determines the p, q, or local FDR value at which marker influences are determined significant.

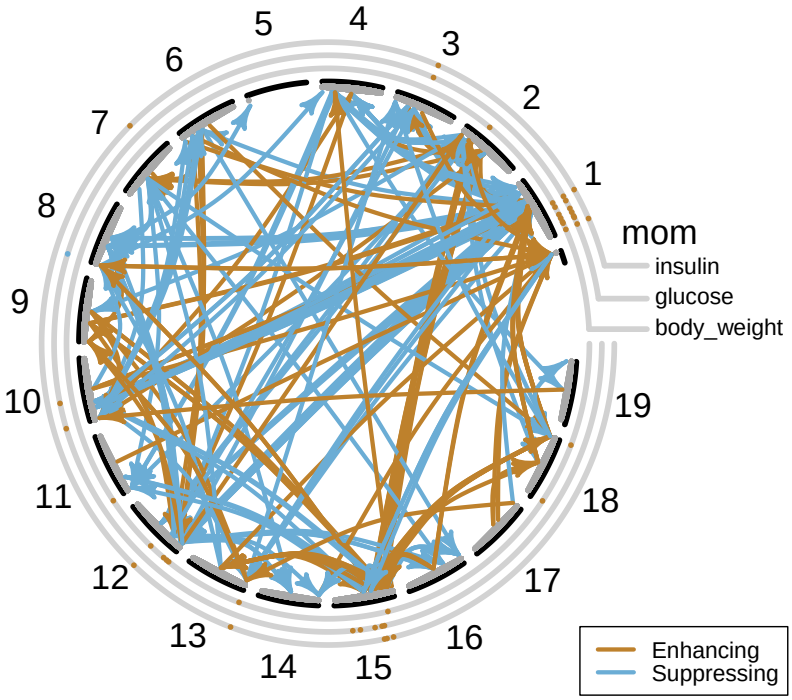
```
obesity.cross <- get.network(obesity.cross, p.or.q = 0.05,
collapse.linked.markers = TRUE)
plotNetwork(obesity.cross, collapsed.net = TRUE)
```

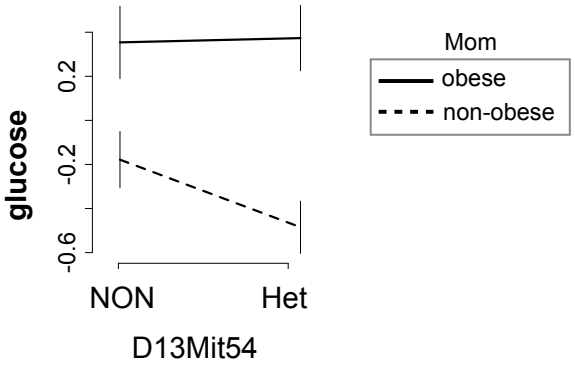


If no collapse by linkage is desired, the argument `collapse.linked.markers` can be set to `FALSE`.

```
obesity.cross <- get.network(obesity.cross, p.or.q = 0.05,
collapse.linked.markers = TRUE)
plotNetwork(obesity.cross, collapsed.net = FALSE)
```

Both the network figure and the adjacency matrix show direct influences of markers on the phenotypes as well as interactions between markers. As expected, many NZO variants on multiple chromosomes show positive effects on plasma insulin and glucose levels as well as on body weight. Two loci had no main effects themselves, but enhanced the effects of maternal obesity on glucose levels. In an interaction plot (Figure **??**), we can see that animals with obese mothers had higher blood glucose levels than those with non-obese mothers. However, those individuals with the heterozygous genotype at D13MIT54 showed a greater increase in glucose with obese mothers than did the individuals with the NON genotype at this locus. Thus the NZO allele at this locus enhances the effect of maternal obesity on blood glucose levels.





It can also be seen in this figure why the direction of action from the genetic locus to the maternal factor was inferred. The genetic locus enhances the phenotypic effect of maternal obesity, but the genetic marker itself does not have a direct effect on glucose that can be enhanced or suppressed by maternal obesity. Furthermore, the phenotypic effects associated

with this locus on insulin and body weight are not influenced by maternal obesity.
These findings illustrate how `cape` is designed to find interactions that simultaneously model all phenotypes under the assumption that interactions between variants across multiple contexts represent a single underlying interaction network. Thus we recommend users assess single-phenotype epistasis using functions in `cape` or in parallel analyses using tools such as `R/qtl` and `R/qtlbim` (Yandell et al., 2007).

Tables of Functions

Basic `cape` Functions

Reading and Writing Data	
read.population	read in population data
writeVariantInfluences	write out a table of the variant influences to each other and the variant influences on traits.
Data Manipulation	
delete.pheno	delete the specified phenotypes from the phenotype matrix
get.covar	retrieve information about set covariates
marker2covar	designate a genetic marker as a covariate
norm.pheno	use rank Z normalization to normalize the phenotypes
pheno2covar	designate a phenotypic value as a covariate
select.eigentraits	select a subset of eigentraits to use in the analysis
select.pheno	select a subset of phenotypes to use in the analysis
select.by.chr	select a subset of chromosomes to use in the analysis
select.by.ind	subset the individuals in the population by either phenotypic or genotypic values
Plotting Functions	
histPheno	plot histograms of phenotypes
plotNetwork	plot a network view of the significant influences of variants on each other and variants on traits
plotPheno	plot phenotypes by individual with the option of coloring points according to a covariate
plotPhenoCor	plot correlations between phenotypes with the option of coloring points according to a covariate
plotSinglescan	plot the results of the single-locus scan or markers selected for pairscan
plotSinglescan.heat	plot the results of the single-locus scan as a heatmap.
plotPairscan	plot the results of the pair scan
plotSVD	plot the results of the singular value decomposition
plotVariantInfluences	plot the adjacency matrix showing the significant influences of variants on each other and variants on traits
qqPheno	plot qq plots for pairs of phenotypes.

Functions for cape Analysis

Function	Behavior	Result Label	Structure	Internal elements	Each Row	Columns
calc.p	calculates empirical and corrected p values of interactions using permutations	empirical.p	list of 2 matrices	m12, m21	marker pair	marker1, marker2, source, target, influence coefficient, influence σ, empirical p value, adjusted p value
direct.influence	calculates the direct influence of each marker pair on each trait and the p value of each influence	var.to.pheno.influence	list of N_t* matrices	each matrix is named with its respective trait	marker pair	marker1, marker2, influence coefficients and standard errors for each marker on each trait
		var.to.pheno.influence.perm†	list of N_t* matrices	each matrix is named with its respective trait	permutation of each marker pair	marker1, marker2, permuted influence coefficients and standard errors for each marker on each trait
		var.to.pheno.test.stat†	list of N_t* matrices	each matrix is named with its respective trait	one marker in one marker pair context	marker, influence coefficient, σ, t statistic, |t statistic|
		var.to.pheno.test.stat.perm†	list of N_t* matrices	each matrix is named with its respective trait	one permutation for each marker in one marker pair context	marker, influence coefficient, σ, t statistic, |t statistic|
		max.var.to.pheno.influence	list of N_t* matrices	each matrix is named with its respective trait	marker	marker, influence coefficient, σ, t statistic, |t statistic|, empirical p value, adjusted p value
error.prop	propagates errors of coefficients calculated in the pair scan	var.to.var.influences or var.to.var.influences.perm	matrix		marker pair	marker1, marker2, m12, m21, m12 σ, m21 σ
get.eigentraits	performs singular value decomposition on phenotype matrix	ET	matrix		individual	one column for each eigentrait
maximum.influence	calculates the maximum influence that each marker has on each trait across all pair contexts	max.var.to.pheno.influence	list of N_t* matrices	each matrix is named with its respective trait	marker pair	marker, influence coefficient, σ, t statistic, |t statistic|, empirical p value, adjusted p value
singlescan	performs all single-locus regressions for each marker on each trait	singlescan.results	list of N_t* matrices	each matrix is named with its respective trait	marker	marker coefficient, σ, t statistic, p value
pairscan(n.perm $\geq$ 0)	performs pair-wise marker regression for each trait	pairscan.results	list of N_t* lists, each with 3 elements	twoD.effects	marker pair	marker labels and coefficients for each marker tested and each marker used as a covariate, including the interaction term
				twoD.se	marker pair	marker labels and σ for each coefficient in twoD.effects
				model.covariance	marker pair	the model covariance matrix for each regression
pairscan(n.perm $\geq$ 1)	performs pair-wise marker regression and permutations for each trait	pairscan.perm	list of N_t* lists, each with 3 elements	twoD.effects.perm	one permutation of marker pair	marker labels and coefficients for each marker tested and each marker used as a covariate, including the interaction term for all permutations
				twoD.se.perm	one permutation of marker pair	marker labels and σ for each coefficient in twoD.effects.perm
				model.covariance.perm	one permutation of marker pair	the model covariance matrix for each permuted regression
select.markers.for.pairscan	selects markers for the pair scan based on which markers are linearly independent of the other markers in the data set	geno.for.pairscan	matrix		individual	one column for each independent marker selected for the pair scan
		covar.for.pairscan	matrix		marker	each of the N_t* columns indicates which markers are to be used as covariates in the pair scan of each trait

*N_t = Number of traits (phenotypes or eigentraits) scanned

†These objects are not added to the data object, but saved in `permutation.data.RData` if specified.

References

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 289–300.

Bevington, P. (1994). *Data reduction and error analysis for the physical sciences, ise*. McGraw-Hill, New York.

Broman, K. W., Sen, S., Owens, S. E., Manichaikul, A., Southard-Smith, E. M., and Churchill, G. A. (2006). The X chromosome in quantitative trait locus mapping. *Genetics*, 174(4):2151–2158.

Broman, K. W., Wu, H., Sen, S., and Churchill, G. A. (2003). R/qtl: QTL mapping in experimental crosses. *Bioinformatics*, 19:889–890.

Burcelin, R., Crivelli, V., Dacosta, A., Roy-Tirelli, A., and Thorens, B. (2002). Heterogeneous metabolic adaptation of C57BL/6J mice to high-fat diet. *American journal of physiology. Endocrinology and metabolism*, 282(4):E834–42.

Carter, G. W., Hays, M., Sherman, A., and Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. *PLoS genetics*, 8(10):e1003010.

Clauset, A., Newman, M. E. J., and Moore, C. (2004). Finding community structure in very large networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 70(6 Pt 2):066111.

Csardi, G. and Nepusz, T. (2006). The igraph software package for complex network research. *InterJournal*, Complex Systems:1695.

Das, S. K. and Elbein, S. C. (2006). The Genetic Basis of Type 2 Diabetes. *Cellscience*, 2(4):100–131.

Gelman, A. and Hill, J. (2006). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Haffner, S. (2003). Epidemic Obesity and the Metabolic Syndrome. *Circulation*, 108(13):1541–1545.

Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, pages 65–70.

Kang, H. M., Zaitlen, N. A., Wade, C. M., Kirby, A., Heckerman, D., Daly, M. J., and Eskin, E. (2008). Efficient control of population structure in model organism association mapping. *Genetics*, 178(3):1709–1723.

Liao, J. G., Lin, Y., Selvanayagam, Z. E., and Shih, W. J. (2004). A mixture model for estimating the local false discovery rate in DNA microarray analysis. *Bioinformatics*, 20(16):2694–2701.

Mackay, T. F. C., Richards, S., Stone, E. A., Barbadilla, A., Ayroles, J. F., Zhu, D., Casillas, S., Han, Y., Magwire, M. M., Cridland, J. M., Richardson, M. F., Anholt, R. R. H., Barrón, M., Bess, C., Blankenburg, K. P., Carbone, M. A., Castellano, D., Chaboub, L., Duncan, L., Harris, Z., Javaid, M., Jayaseelan, J. C., Jhangiani, S. N., Jordan, K. W., Lara, F., Lawrence, F., Lee, S. L., Librado, P., Linheiro, R. S., Lyman, R. F., Mackey, A. J., Munidasa, M., Muzny, D. M., Nazareth, L., Newsham, I., Perales, L., Pu, L.-L., Qu, C., Ràmia, M., Reid, J. G., Rollmann, S. M., Rozas, J., Saada, N., Turlapati, L., Worley, K. C., Wu, Y.-Q., Yamamoto, A., Zhu, Y., Bergman, C. M., Thornton, K. R., Mittelman, D., and Gibbs, R. A. (2012). The Drosophila melanogaster Genetic Reference Panel. *Nature*, 482(7384):173–178.

Permutt, M. A., Wasson, J., and Cox, N. (2005). Genetic epidemiology of diabetes. *The Journal of clinical investigation*, 115(6):1431–1439.

Reifsnyder, P. C. (2000). Maternal Environment and Genotype Interact to Establish Diabesity in Mice. *Genome Research*, 10(10):1568–1578.

Yandell, B. S., Mehta, T., Banerjee, S., Shriner, D., Venkataraman, R., Moon, J. Y., Neely, W. W., Wu, H., von Smith, R., and Yi, N. (2007). R/qtlbim: QTL with Bayesian Interval Mapping in experimental crosses. *Bioinformatics*, 23(5):641–643.