Package 'cape'

June 9, 2016

Type Package

Title Combined Analysis of Pleiotropy and Epistasis

Version 2.0.2

Date 2016-03-09

Author Anna L. Tyler, Wei Lu, Justin J. Hendrick, Vivek M. Philip, and Greg W. Carter

Maintainer Anna L. Tyler <Anna.Tyler@jax.org>

Description Combines complementary information across multiple related phenotypes to infer directed epistatic interactions between genetic markers. This analysis can be applied to a variety of engineered and natural populations.

Depends R (>= 3.2.2)

Imports grDevices, graphics, stats, utils, corpcor, evd, qpcR, Matrix, igraph, fdrtool, shape, parallel, doParallel, RColorBrewer, foreach, HardyWeinberg, regress

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-09 19:59:15

R topics documented:

cape-package *Combinatorial Analysis of Epistasis and Pleiotropy*

Description

This package infers predictive networks between genetic variants and between genetic variants and phenotypes. It uses complementary information of pleiotropic gene variants across different phenotypes to resolve models of epistatic interactions between genetic variants. To do this, cape reparameterizes main effect and interaction coefficients from a pairwise variant regressions into directed influence parameters. These parameters describe how gene variants influence each other, in terms of suppression and enhancement, as well as how gene variants influence phenotypes.

Details

Package:	cape
Type:	Package
Version:	1.3
Date:	2013-08-12
License:	GPL-3

The cape analysis begins by reading in a genetic data set with `read.population`. The data are converted into a data object, to which results are added throughout the analysis. This data object is an argument in most functions, and is referred to as `data.obj`. Because this package uses pleiotropy to resolve models of epistasis, the phenotypes used should have common underlying molecular players, but not be perfectly correlated. In general phenotypes should be correlated with a Pearson r between 0.4 and 0.8. The phenotypes of interest are then decomposed into eigentraits using `get.eigentraits`. Any number of phenotypes can be decomposed, but cape requires between two and 12 eigentraits for the analysis. The phenotype decomposition into eigentraits maximizes the complementary information between the phenotypes. Before investigating epistatic interactions through a pair-wise scan of the genetic variants, a single-variant scan (`singlescan`) is run. This allows for thresholding of markers for the pair scan if the cross is prohibitively large to test all pairs of variants. The single-variant scan also allows selection of variants with very large main effects to be used as covariates in the pair scan. The pair scan (`pairscan`) performs a regression on each variant pair. Finally, the coefficients from the pairwise scan are reparameterized to yield directional influences between variants and from variants to phenotypes. The final result is an asymmetric adjacency matrix describing these variant influences. The p values of these influences are corrected for multiple tests.

Author(s)

Anna L. Tyler, Wei Lu, Justin J. Hendrick, Vivek M. Philip, and Gregory W. Carter Maintainer: Anna L. Tyler <anna.tyler@jax.org>

References

Carter, G. W., Hays, M., Sherman, A., & Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. PLoS genetics, 8(10), e1003010. doi:10.1371/journal.pgen.1003010

Examples

```
data(obesity.cross)
str(obesity.cross)
```

`calc.p` *Calculate P Values for Interactions Based on Permutations*

Description

This function uses the perumtation results to calculate empirical p values for the variant-to-variant influences calculated by `error.prop`. It can also optionally adjust these p values using Holm's step-down procedure, false discovery rate (fdr), or local false discovery rate (lfdr).

Usage

```
calc.p(data.obj, pairscan.obj,
pval.correction = c("holm", "fdr", "lfdr", "none"), n.cores = 2)
```

Arguments

`data.obj`	The object in which all results are stored. See `read.population`.
`pairscan.obj`	The object in which the results from the pairscan are stored. See `pairscan`.
`pval.correction`	One of "holm", "fdr", "lfdr" or "none", indicating whether the p value correction method used should be the Holm step-down procedure, false discovery rate, local false discovery, or no correction rate respectively.
`n.cores`	An integer specifying the number of cores to be used in parallel processing.

Value

The data object is returned with a new list with two elements. The elements correspond to the two directions of influence: marker1 to marker2 and marker2 to marker1. Each element contains a table with the source and target variants, the empirical p values, and the adjusted p values, along with the effect size, standard error and t statistic for each interaction.

References

Holm, S. (1979). A simple sequentially rejective multiple test procedure. Scandinavian journal of statistics, pages 65-70. Benjamini, Y., & Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. Journal of the Royal Statistical Societ. Series B (Methodological), 289-300. Liao, J.G., Lin, Y., Selvanayagam, Z.E., & Shih, W.J. (2004). A mixture model for estimating the local false discovery rate in DNA microarray analysis. Bioinformatics, 20(16), 2694-2701. doi:10.1093/bioinformatics/bth310

delete.pheno — *Remove phenotypes from the phenotype matrix*

Description

This function deletes an unwanted phenotype or phenotypes from the phenotype matrix.

Usage

```
delete.pheno(data.obj, phenotypes)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
phenotypes	A vector of either column numbers or column names designating which phenotype or phenotypes should be deleted.

Value

This function returns the data object with the specified phenotypes removed.

Examples

```
data(obesity.cross)
str(obesity.cross)
obesity.cross <- delete.pheno(obesity.cross, "insulin")
str(obesity.cross)
```

direct.influence — *Calculate the significance of direct influences of variant pairs on phenotypes*

Description

This function recasts the variant-to-eigentrait effects in terms of variant-to-phenotype effects. It multiplies the β-coefficient matrices of each variant (i) and each phenotype (j) (β_i^j) by the singular value matrices ($V \cdot W^T$) obtained from the singular value decomposition performed in get.eigentraits. $\beta_i^j = V \cdot W^T$. It also uses the permutation data from the pairwise scan (pairscan) to calculate an empirical p value for the influence of each marker pair on each phenotype. The empirical p values are then adjusted for multiple testing using Holm's step-down procedure.

Usage

```
direct.influence(data.obj, pairscan.obj, transform.to.phenospace = TRUE,
pval.correction = c("holm", "fdr", "lfdr"), verbose = FALSE,
save.permutations = FALSE)
```

Arguments

`data.obj`	The object in which all results are stored. See `read.population`.
`pairscan.obj`	The object in which the results from the pairscan are stored. See `pairscan`.
`transform.to.phenospace`	A logical value. If TRUE, the influence of each marker on each eigentrait is transformed to the influence of each marker on each of the original phenotypes. If FALSE, no transformation is made. If the pair scan was done on eigentraits, the influence of each marker on each eigentrait is calculated. If the pair scan was done on raw phenotypes, the influence of each marker on each phenotype is calculated. The default behavior is to transform variant influences on eigentraits to variant influences on phenotypes.
`pval.correction`	One of "holm", "fdr", or "lfdr", indicating whether the p value correction method used should be the Holm step-down procedure, false discovery rate or local false discovery rate respectively.
`verbose`	A logical value. If TRUE, the progress of the function is printed to the screen. If FALSE, the default, nothing is printed.
`save.permutations`	A logical value indicating whether the data from permutations should be saved. Saving the permutations requires more memory but can be helpful in diagnostics. If save.permutations is TRUE all permutation data are saved in an object called "permutation.data.RData".

Value

This function adds a total of six objects to the data object. First, a flag (`transform.to.phenospace`) is added to the data object to indicate whether variant influences were transformed to phenotype space.

The results from the pairwise scan and the permutations of the pairwise scan are converted to phenospace if specified. These actions each add one object each to the data object (`var.to.pheno.influence` and `var.to.pheno.influence.perm`). Each element is itself a list of matrices corresponding to the original phenotypes. Each matrix contains one row per marker pair (or permutation of a marker pair) and contains the influence coefficient and standard error of the influence coefficient for each pair.

After the coefficients have been transformed to phenotype space, each marker is considered individually and its influence on each phenotype across all marker pair contexts is tabulated. This is done for both the pairwise scan and the permutations of the pairwise scans and adds two new objects (`var.to.pheno.test.stat` and `var.to.pheno.test.stat.perm`) to the data object. Each object is a list containing one element for each of the original phenotypes. Each element contains a table in which all instances of each marker are listed along with that marker's direct phenotypic influence, the standard error of the influence, and the t statistic (β/σ) of the influence.

Finally, because each marker can only have one influence on each phenotype, the influences from each marker pair context are filtered to report only the maximum influence of each marker on each phenotype across all marker pair contexts. This process adds an object to the data object called `max.var.to.pheno.influence`. This object is a list containing one element per phenotype. It tabulates the maximum influence of each marker on each phenotype, as well as the empirical and Holm's corrected p values associated with each influence.

References

Carter, G. W., Hays, M., Sherman, A., & Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. PLoS genetics, 8(10), e1003010. doi:10.1371/journal.pgen.1003010
Holm, S. (1979). A simple sequentially rejective multiple test procedure. Scandinavian journal of statistics, pages 65-70.

`error.prop` *Estimate Errors of Regression Coefficients*

Description

This function uses error propagation formulas for quantities computed from regression coefficients to estimate the error for all regression coefficients.

Usage

```
error.prop(data.obj, pairscan.obj, perm = FALSE, verbose = FALSE, n.cores = 2)
```

Arguments

`data.obj`	The object in which all results are stored. See `read.population`.
`pairscan.obj`	The object in which the results from the pairscan are stored. See `pairscan`.
`perm`	A logical value to indicate whether error propagation should be performed on the test statistics (FALSE) or the permuted test statistics (TRUE).
`verbose`	A logical value to indicate whether the progress of the function should be printed to the screen.
`n.cores`	An integer specifying the number of cores to be used in parallel processing.

Value

This function returns the data object with a new list element: var.to.var.influences if perm is set to FALSE and var.to.var.influences.perm if perm is set to TRUE. These tables include the errors calculated for the marker1 to marker2 influences as well as the marker2 to marker1 influences. These results are used by `calc.p` to calculate empirical p values.

References

Carter, G. W., Hays, M., Sherman, A., & Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. PLoS genetics, 8(10), e1003010. doi:10.1371/journal.pgen.1003010

Bevington PR (1969) Data Reduction and Error Analysis for the Physical Sciences. New York: McGraw-Hill.

filter.hwe *Filter markers by Hardy-Weinberg equilibrium*

Description

This function tests markers for Hardy-Weinberg equilibrium (HWE) and removes markers that are significantly out of HWE.

Usage

```
filter.hwe(data.obj, geno.obj, p.thresh = 1e-6, run.parallel = TRUE, n.cores = 2)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which the genotype object is stored.
p.thresh	The p value below which markers are considered out of HWE.
run.parallel	A logical value to indicate whether to run the process in parallel.
n.cores	An integer specifying the number of cores to be used in parallel processing.

Value

This function returns the data object with the marker information pared down to only those markers in HWE.

References

Add reference for HWE. Hartl and Clark?

filter.maf *Filter markers by minor allele frequency.*

Description

This function tests markers for minor allele frequency and removes markers below the user-set threshold.

Usage

```
filter.maf(data.obj, geno.obj, maf = 0.05)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which the genotype object is stored.
maf	The minor allele frequency below which markers are removed.

Value

This function returns the data object with the marker information pared down to only those markers with minor allele frequencies greater than that specified by the user.

---

get.covar *Get information about covariates*

Description

This function returns information about all covariates that originated from either genotype or phenotype.

Usage

```
get.covar(data.obj, covar = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
covar	A character string indicating which covariates to return information about.

Value

This function returns a list with the following elements:

- covar.names: A vector of covariate names
- covar.type: A vector containing "p" and "g" values to indicate whether each covariate originate as a genetic marker or a phenotype.
- covar.loc: A numeric vector indicating the chromosomal coordinate of each covariate. Phenotypic covariates are given dummy coordinates of 1:n.
- covar.table: A matrix giving the values of each covariate for each individual.

---

get.eigentraits *Calculate eigentraits from phenotype matrix*

Description

This function performs the singular value decomposition (SVD) on the phenotype matrix after first removing individuals with missing data. The eigentraits are the left singular vectors of the decomposition. This function optionally mean centers and normalizes the phenotype matrix before performing the SVD.

Usage

```
get.eigentraits(data.obj, scale.pheno = TRUE, normalize.pheno = TRUE)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
scale.pheno	A logical value specifying whether the phenotypes should be mean centered before the SVD is performed. The default, and recommended, value is TRUE.
normalize.pheno	A logical value specifying whether the phenotypes should be quantile normalized before the SVD is performed.

Value

This function adds three new elements to the data.obj list.

ET	The left singular vectors from the SVD. These are the eigentraits.
singular.values	The singular values from the SVD. These are used later internally to convert variant effects from eigentrait space to phenotype space.
right.singular.vectors	The right singular vectors from the SVD. These are used later internally to convert variant effects from eigentrait space to phenotype space.

Note

There must be more individuals than phenotypes to perform this calculation. An error results if there are more phenotypes than individuals.

References

Carter, G. W., Hays, M., Sherman, A., & Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. PLoS genetics, 8(10), e1003010. doi:10.1371/journal.pgen.1003010

See Also

norm.pheno, plotSVD

---

get.geno *Retrieve the genotype matrix.*

Description

This function retrieves the genotype matrix either from the data object or the genotype object. If the genotype matrix is held in a separate genotype object, the marker and individual information in the data object is used to filter the genotype matrix to the appropriate markers and individuals.

Usage

```
get.geno(data.obj, geno.obj)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which genotype data are stored. See read.geno.

Value

Returns a matrix of genotype values for the individuals and markers specified in the data object.

See Also

read.geno read.pheno make.data.obj read.population get.pheno

---

get.marker.chr *Get chromosome assignments for a vector of markers.*

Description

This function returns the chromosome that each marker in the user-supplied vector is on.

Usage

```
get.marker.chr(data.obj, markers)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
markers	A vector of either marker names or marker numbers for which chromosome assignments are desired.

Value

Returns a vector of numeric values specifying which chromosome each marker in the input vector resides on.

See Also

get.marker.idx get.marker.location get.marker.name get.marker.num

get.marker.idx *Get the column index of markers in the genotype matrix*

Description

This function returns the column index in the genotype matrix for each marker specified by the user.

Usage

```
get.marker.idx(data.obj, markers)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
markers	A vector of either marker names or marker numbers for which the column indices are desired

Value

Returns a vector of numeric values specifying the index of each marker in the genotype matrix.

See Also

get.marker.chr get.marker.location get.marker.name get.marker.num

get.marker.location *Get the chromosomal coordinate of markers*

Description

This function returns the chromosomal coordinate of markers specified by the user.

Usage

```
get.marker.location(data.obj, markers)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
markers	A vector of either marker names or marker numbers for which the chromosomal coordinates are desired.

Value

Returns a vector of numeric values specifying the chromosomal coordinate of each marker in the input vector.

See Also

get.marker.chr get.marker.idx get.marker.name get.marker.num

---

get.marker.name *Get marker names from marker numbers*

Description

This function returns the name of markers specified by number

Usage

```
get.marker.name(data.obj, markers)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
markers	A vector of marker numbers or names for which names are desired. If the vector contains marker names, the marker names will be returned unchanged.

Value

Returns a vector of character strings specifying the name of each marker designated by a number or name in the input vector. Markers can either be named with a character string or a number in the genotype matrix. This function allows conversion between them.

See Also

get.marker.chr get.marker.idx get.marker.location get.marker.num

---

get.marker.num *Get marker numbers from marker names*

Description

This function returns the marker number of each marker specified by name

Usage

```
get.marker.num(data.obj, markers)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
markers	A vector of marker numbers or names for which names are desired. If the vector contains marker numbers, the marker numbers will be returned unchanged.

Value

Returns a numeric vector of specifying the number of each specified marker. Markers can either be named with a character string or a number in the genotype matrix. This function allows conversion between them.

See Also

get.marker.chr get.marker.idx get.marker.location get.marker.name

---

`get.marker.val` *Get marker values*

Description

This function returns genotypes for each individual at a given marker. This function is relatively time-consumig and is not recommended for high-throughput use.

Usage

```
get.marker.val(data.obj, geno.obj = NULL, markers)
```

Arguments

`data.obj`	The object in which all results are stored. See `read.population`.
`geno.obj`	The object in which the genotype matrix and marker information are stored. See `read.geno`.
`markers`	A vector of marker numbers or names for which names are desired.

Value

Returns a matrix listing the individual genotypes of the markers given in the input. Markers can be names either with a character string or a number.

See Also

get.marker.chr get.marker.idx get.marker.location get.marker.name get.marker.num

get.network *Convert the final results to a form plotted by* plotNetwork *and* plotCollapsedVarInf

Description

This function converts the significant epistatic interactions to a form that can be plotted as a network. This conversion also optionally condenses the network based on linkage between markers. The degree to which the network is condensed is determined by the argument r2.thresh. This value sets the correlation at which two markers are considered linked.

Usage

```
get.network(data.obj, p.or.q = 0.05,
standardized = TRUE, min.std.effect = 0,
collapse.linked.markers = TRUE, verbose = FALSE,
plot.linkage.blocks = FALSE, threshold.power = 1)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
p.or.q	A numerical threshold indicating the maximum adjusted p value considered significant. If an fdr method has been used to correct for multiple testing, this value specifies the maximum q value considered significant.
standardized	A logical value indicating whether values placed in weighted adjacency matrix are standardized values or unstandardized values.
min.std.effect	A numerical threshold indicating the absolute value of the minimum standard effect size to be shown in the plot. The default value of 0 performs no thresholding.
collapse.linked.markers	A logical value. If TRUE markers are combined into linkage blocks based on correlation. If FALSE, each marker is treated as an independent observation.
verbose	A logical value indicating whether the function progress should be printed to the screen.
plot.linkage.blocks	A logical value indicating whether the chromosomes should be plotted with their linkage blocks delineated. The type of plot produced differs depending on which choice is specified by linkage.method.
threshold.power	The linkage blocks are calculated by finding communities in a correlation matrix of genetic markers. This power indicates a soft-thresholding power for the correlation matrix. The higher the power, the more blocks will be generated.

Details

This function delineates linkage blocks based on the correlation between adjacent markers. A correlation matrix is first calculated between all marker pairs on a single chromosome. This similarity matrix is used to construct a weighted network, and the fastgreedy community detection algorithm from the R package igraph is used to detect individual communities within the network. Each community of contiguous markers is designated as a distinct linkage block.

References

Csardi G, Nepusz T: The igraph software package for complex network research, InterJournal, Complex Systems 1695. 2006. http://igraph.org

See Also

plotNetwork, plotCollapsedVarInf

`get.pheno` *Retrieve the genotype matrix.*

Description

This function retrieves the phenotype matrix either from the data object.

Usage

```
get.pheno(data.obj, scan.what = c("eigentraits", "raw.traits"))
```

Arguments

`data.obj`	The object in which all results are stored. See `read.population`.
`scan.what`	A character string defining whether the eigentraits or phenotypes are desired.

Value

Returns a matrix of phenotype values.

See Also

`read.geno` `read.pheno` `make.data.obj` `read.population` `get.geno`

histPheno *Plot histograms of phenotypes.*

Description

This function plots histograms of phenotypes.

Usage

```
histPheno(data.obj, pheno.which = NULL,
pheno.labels = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
pheno.which	A vector of character strings indicating which phenotypes should be plotted. If pheno.which is NULL, all phenotypes will be plotted.
pheno.labels	An optional vector of character strings giving alternate names for the phenotypes being plotted.

See Also

plotPhenoCor, plotPheno, qqPheno

impute.missing.geno *Impute missing genotypes in measured markers.*

Description

This function imputes missing genotype data using weighted k nearest neighbors imputation.

Usage

```
impute.missing.geno(data.obj, geno.obj = NULL,
impute.full.genome = FALSE, k = 10, ind.missing.thresh = 0,
marker.missing.thresh = 0,
prioritize = c("ind", "marker", "fewer"),
max.region.size = NULL, min.region.size = NULL,
run.parallel = TRUE, verbose = FALSE, n.cores = 2)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which genotype data are stored. See read.geno.
impute.full.genome	In CAPE it is possible to scan a subset of the full genotype matrix. This argument indicates whether imputation should be done only on the subset of markers being scanned or on the entire genotype matrix.
k	The number of neighbors to use in k-nearest neighbors imputation.
ind.missing.thresh	A percentage. After imputation the number of data points still missing is assessed. Markers for which large numbers of individuals still have missing data will be removed. This arguement defines the percentage of missing individuals above which a marker will be removed.
marker.missing.thresh	A percentage. After imputation the number of data points still missing is assessed. Individuals for which large numbers of markers still have missing data will be removed. This arguement defines the percentage of missing markers above which an individual will be removed.
prioritize	Markers and individuals with excessive missing data after imputation will be removed. If there is one marker with very low genotyping coverage, it is preferrable to remove that marker rather than all the individuals who are missing a genotype for that marker. This argument allows prioritization of the removal of markers or individuals. The default value is "fewer," meaning that priority goes to whichever dimension (individuals or markers) results in removal of fewer elements.
max.region.size	A number indicating the maximum number of markers to consider in each imputation step. This value defaults to the maximum number of markers on one chromosome.
min.region.size	A number indicating the minimum number of markers to consider in each imputation step. This value defaults to the minimum number of markers on one chromosome.
run.parallel	A logical value indicating whether this process should be run in parallel.
verbose	A logical value indicating whether the progress of the process should be printed to the screen
n.cores	An integer specifying the number of cores to be used in parallel processing.

Value

Because both the data.obj and the geno.obj are manipulated by this function, it returns a list of two elements in which the first is the data.obj and the second is the geno.obj. These objects need to be separated to continue with subsequent functions. If no geno.obj was provided to the function, the result it just the data.obj with the genotype matrix embedded.

See Also

read.geno read.pheno make.data.obj read.population

`make.data.obj` *Generate data.obj from pheno.obj and geno.obj*

Description

This function combines information from a phenotype matrix and a genotype object to create a data object in which the results of the analysis will be stored.

Usage

```
make.data.obj(pheno.obj, geno.obj)
```

Arguments

`pheno.obj`	The object in which all results are stored. See `read.population`.
`geno.obj`	The object in which the genotype matrix and marker information are stored. See `read.geno`.

Value

Returns the data object. Marker information, but not the genotype matrix itself are transferred to the pheno.obj to generate a data.obj. The geno.obj is unchanged and is not returned.

See Also

read.geno read.pheno read.population

`marker2covar` *Create a covariate from a genetic marker.*

Description

Covariates can be derived from either genetic markers or phenotypic values. This function generates covariates from genetic markers. These covariates are treated distinctly from those generated from phenotypic values. For example, they are used in kinship calculations. The are also plotted in their original positions, rather than placed at the end as the phenotypic covariates are.

Usage

```
marker2covar(data.obj, geno.obj = NULL,
singlescan.obj = NULL, covar.thresh = NULL,
markers = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which the genotype matrix and marker information are stored. See read.geno.
singlescan.obj	The results of the singlescan generated by singlescan
covar.thresh	Genetic markers can be selected as covariates based on their standardized effect sizes in the singlescan. covar.thresh indicates the standardized effect size threshold for generating covariates. All markers with standardized effect sizes above this value will be converted to covariates.
markers	A vector of names or numbers indicating which genetic markers should be converted to covariates.

Value

Returns the data object with two additional elements.

g.covar.table	A table of the covariates and their values.
g.covar	A table with positional information for each covariate. The table contains three rows indicating each covariate's name, chromosome, and chromosomal position.

See Also

pheno2covar

---

norm.pheno *Normalize and mean center phenotypes*

Description

This function performs quantile normalization on phenotypes and optionally mean centers them.

Usage

```
norm.pheno(data.obj, mean.center = TRUE)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
mean.center	A logical value. If TRUE the phenotypes are mean centered in addition to being normalized. if FALSE, the phenotypes are not mean centered.

Details

In quantile normalization the values of the phenotype are sorted and replaced 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.

Value

This function returns data.obj with the normalized phenotypes in place of the original phenotypes.

Note

Both normalization and mean centering are highly recommended before obtaining the eigentraits with singular value decomposition (SVD) (see `get.eigentraits`). This normalization procedure can also be performed by `get.eigentraits`

See Also

`get.eigentraits`

---

obesity.cross *Mouse cross data from Reifsnyder et al. (2000)*

Description

Data from a cross between non-obese, non-diabetic (NON) mice, and diabetes-prone New Zealand obese (NZO) mice. The experiment is described in Reifsnyder et al. (2000). The data object is a list. The first element is a matrix of phenotype data. In includes insulin levels (ng/mL), plasma glucose levels (mg/dL), total body weight (g), all measured at age 24 weeks. The second element is the genotype matrix containing the genotype of each mouse at each of 85 markers across the genome. The final elements are the chromosome vector, which indicates which chromosome on which each marker is found, and a vector of chromosomal positions indicating the location of each marker on the chromosome in centimorgans (cM).

Usage

```
data(obesity.cross)
```

Format

The format is: List of 4 $ pheno : num [1:204, 1:3] 63.3 31 43.3 33.3 35.3 15.9 23.6 NA 16 22 ... ..- attr(*, dimnames)=List of 2 .. ..$: NULL .. ..$: chr [1:3] body_weight sex $ geno : num [1:204, 1:85] 0.5 0.5 0.5 0 0.5 0 0.5 NA 0 0 ... ..- attr(*, dimnames)=List of 2 .. ..$: NULL .. ..$: chr [1:85] D1Mit296 D1Mit211 D1Mit411 D1Mit123 ... $ chromosome : chr [1:85] 1 1 1 1 ... $ marker.location: num [1:85] 2.08 10.59 12.62 17.67 22.88 ...

Source

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

pairscan *Perform regressions for all pairs of markers and all phenotypes.*

Description

This function performs the pairwise regression on all selected marker pairs. The phenotypes used can be either eigentraits or raw phenotypes. Permutation testing is also performed, and kinship corrections are implemented if requested.

Usage

```
pairscan(data.obj, geno.obj = NULL, covar = NULL,
scan.what = c("eigentraits", "raw.traits"), total.perm = NULL,
min.per.genotype = NULL, max.pair.cor = NULL,
n.top.markers = NULL, use.kinship = FALSE, kin.full.geno = TRUE,
sample.kinship = TRUE, num.kin.samples = 100, n.per.sample = 100,
verbose = FALSE, num.pairs.limit = 1e6, num.perm.limit = 1e7,
overwrite.alert = TRUE, n.cores = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which the genotype matrix and marker information are stored. See read.geno.
covar	A vector of character strings indicating which covariates should be used in the regressions.
scan.what	A character string uniquely identifying whether eigentraits or raw traits should be scanned.
total.perm	The total number of permutations to be performed.
min.per.genotype	The minimum number of individuals allowable per genotype. If for a given marker pair, one of the genotypes is underrepresented, the marker pair is not tested. If this value is NULL, max.pair.cor must have a numeric value.
max.pair.cor	A numeric value between 0 and 1 indicating the maximum Pearson correlation that two markers are allowed. If the correlation between a pair of markers exceeds this threshold, the pair is not tested. If this value is set to NULL, min.per.genotype must have a numeric value.
n.top.markers	This optional integer is used in the generation of a null distribution. To generate the null, a singlescan is performed on permuted data and the top ranking markers are used in a pairscan. n.top.markers specifies how many of these markers should be chosen. If NULL, n.top.markers is the number of markers being scanned in the pairscan.
use.kinship	A logical value indicating whether a kinship correction should be implemented.

`kin.full.geno` A logical value indicating whether the entire genotype matrix should be used to calculate kinship corrections. If FALSE, only the subset of genetic markers being scanned will be used to calculate kinship corrections.

`sample.kinship` A logical value indicating whether kinship matrices should be sampled (TRUE) or calculated directly (FALSE). Sampling kinshp matrices can be faster in the case of very large genotype matrices.

`num.kin.samples`
If sample.kinship is TRUE, this integer indicates how many samples should be used to calculate kinship matrices.

`n.per.sample` If sample.kinship is TRUE, this integer indicates how many markers should be used in each sample of the genotype matrix for calculating kinship matrices.

`verbose` A logical value indicating whether the progress of the scan should be printed to the screen.

`num.pairs.limit`
A number indicating the maximum number of pairs to scan. If the number of pairs exceeds this threshold, the function asks for confirmation before proceeding with the pairwise scan.

`num.perm.limit` A number indicating the maximum number of total permutations that will be performed. If the number of total permutations exceeds this threshold, the function asks for confirmation before proceeding with the pairwise scan.

`overwrite.alert`
If TRUE, this triggers an alert warning the user that the output of this function should be saved separately from the data.obj.

`n.cores` An integer specifying the number of cores to be used in parallel processing. If NULL, the choice is made automatically.

Details

Not all marker pairs are necessarily tested. Before markers are tested for interaction, they are checked for several conditions. Pairs are discarded if (1) at least one of the markers is on the X chromosome, or (2) there are fewer than min.per.genotype individuals in any of the genotype combinations.

Value

IMPORTANT NOTE: Prior versions of this function modified the data.obj. The new version creates a separate pairscan.obj, and the output of this function should NOT be assigned to the data.obj. The pairscan.obj contains the following elements:

`pairscan.results`
The results of the pairwise scan on the provided phenotype and genotypes.

If permutations have been performed (n.perm > 0), an additional element is added to the object reporting the results of the permutation tests:

`pairscan.perm` The results of the permutations of the pairwise scan on the provided phenotype and genotypes.

Each of these results elements is itself a list of 3 elements:

`pairscan.effects`
: A table of effects of each marker pair. The columns list the effects in the following order: marker1, marker2, the variance of marker1, the covariance of marker1 and marker2, the variance of marker2, the covariance of marker1 and the interaction effect, the covariance between marker2 and ther interaction effect, and the variance of the interaction.

`pairscan.se`
: A table of the standard errors from the test on each marker pair. The columns are identical to those described for pairscan.effects

`model.covariance`
: This is a table in which each row is the linearized matrix of the variance-covariance matrix of each pairwise regression.

`pairs.tested`
: A two-column matrix identifying the marker pairs tested for each permutation.

The results element for the permutation tests has the same structure as for the pairwise scan except that each row represents the results of one permutation.

References

Carter, G. W., Hays, M., Sherman, A., & Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. PLoS genetics, 8(10), e1003010. doi:10.1371/journal.pgen.1003010

See Also

`select.markers.for.pairscan`, `plotPairscan`

---

pheno2covar *Create a covariate from a phenotype.*

Description

Covariates can be derived from either genetic markers or phenotypic values. This function generates covariates from phenotypes. These covariates are treated distinctly from those generated from genetic markers. For example, covariates derived from genetic markers are used in kinship calculations, whereas covariates derived from phenotypes are not. Covariates derived from phenotypes are also plotted after all genetic markers.

Usage

```
pheno2covar(data.obj, pheno.which)
```

Arguments

`data.obj`
: The object in which all results are stored. See `read.population`.

`pheno.which`
: A vector of character strings indicating which phenotypes should be converted to covariates.

Value

Returns the data object with two additional elements.

g.covar.table	A table of the covariates and their values.
p.covar	A vector of the names of the phenotypic covariates

See Also

marker2covar

---

plotCollapsedVarInf *Plot variant-to-variant influences*

Description

This function plots the final network as does plotVariantInfluences, but it plots the network condensed by linkage blocks as performed by get.network. This function can only be run after running get.network

Usage

```
plotCollapsedVarInf(data.obj, expand.labels = FALSE,
all.markers = FALSE, scale.effects = c("none", "log10", "sqrt"))
```

Arguments

data.obj	The object in which all results are stored. See read.population.
expand.labels	A logical value. If FALSE, the markers are labeled as linkage blocks ("block1", "block2" and so forth). If TRUE, the block labels are expanded to show all-marker names included in each block.
all.markers	A logical value. If TRUE all markers are plotted. If FALSE only markers tested in the pair scan are plotted.
scale.effects	A string indicating a scaling function by which the effects should be scaled. This is useful in increasing contrast between effects with large variance.

See Also

pairscan

plotNetwork *Plot the final epistatic network*

Description

This function plots the final results with a using a different layout than plotVariantInfluences. Instead of the adjacency matrix, this function plots interactions between markers with arrows indicating the direction of influence. The function get.network must be run before plotting the network.

Usage

```
plotNetwork(data.obj, collapsed.net = TRUE,
trait = NULL, phenotype.labels = NULL,
main.lwd = 4, inter.lwd = 3, label.cex = 1.5,
percent.bend = 15, chr.gap = 1, label.gap = 5,
positive.col = "brown", negative.col = "blue")
```

Arguments

data.obj	The object in which all results are stored. See read.population.
collapsed.net	A logical value. If TRUE, a network condensed by linkage (see get.network) is plotted. If FALSE, the full network is plotted.
trait	A character vector indicating which traits should be plotted. If NULL, all traits are plotted.
phenotype.labels	A vector of strings listing alternate names for the phenotypes.
main.lwd	A numeric value specifying the line width of the main effects bars surrounding the chromosome bars.
inter.lwd	A numeric value specifying the line width of the interaction arrows drawn inside the chromosome circle
label.cex	A numberic value specifying the size of the chromomsome labels.
percent.bend	A numeric value between 0 and 100 specifying how much curvature should be added to the arrows within the circle.
chr.gap	A numeric value between 0 and 100 indicating what percentage of the circle should be devoted to individual gaps between chromosomes.
label.gap	A numeric value between 0 and 100 indicating what percentage of the circle should be devoted to the phenotype labels.
positive.col	The color in which to plot the postive values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."
negative.col	The color in which to plot the negative values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."

See Also

get.network, plotVariantInfluences

Examples

```
## Not run:
plotNetwork(obesity.cross, collapsed.net = TRUE)
plotNetwork(obesity.cross, collapsed.net = FALSE)
plotNetwork(obesity.cross, collapsed.net = TRUE, trait = "glucose")

## End(Not run)
```

---

plotPairscan *plot the results from* pairscan

Description

This function plots the results from the pairwise regressions. The matrices are plotted with results coded on a yellow to blue scale. Results from all phenotypes are plotted on the same scale.

Usage

```
plotPairscan(data.obj, pairscan.obj,
phenotype = NULL, standardized = TRUE,
show.marker.labels = FALSE, show.chr = TRUE,
label.chr = TRUE, pdf.label = "Pairscan.Regression.pdf",
neg.col = "blue", pos.col = "brown",
col.pal = "dark", verbose = FALSE)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
pairscan.obj	The object in which the results from pairscan are stored.
phenotype	A character vector indicating which phenotypes should be plotted. All phenotypes are plotted if phenotype is set to NULL.
standardized	A logical value. If FALSE, the plotted values are the beta coefficients of the interaction in the pairwise model. If TRUE, the t statistics (β/σ) are plotted.
show.marker.labels	A logical value indicating whether marker labels should appear on the axes.
show.chr	A logical value. If TRUE, chromosomes are indicated by alternating gray and white blocks along plot axes.
label.chr	A logical value. If TRUE and if show.chr is TRUE, chromosome numbers are printed inside each gray and white block.
pdf.label	A character string indicating the name of the file that the plots should be printed to. When multiple phenotypes are plotted, all penotypes are plotted on the same scale and each is plotted in a different page of the pdf.

neg.col	The color in which to plot the negative values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."
pos.col	The color in which to plot the positive values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."
col.pal	Either "light" or "dark" depending on the desired range of colors used.
verbose	A logical value indicating whether the progress of the plotting function should be reported. Large data sets can take a long time to process and plot.

Value

No values are returned.

See Also

pairscan

---

plotPheno *Plot phenotype values by individual.*

Description

This function plots phenotype values by individual, and can color points based on covariates if desired.

Usage

```
plotPheno(data.obj, pheno.which = NULL,
color.by = NULL, group.labels = NULL,
pheno.labels = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
pheno.which	A vector of character strings indicating which phenotypes should be plotted. If pheno.which is NULL, all phenotypes will be plotted.
color.by	A vector of character strings indicating how the points should be colored. Points are colored based on covariates. These covariates must be specified using pheno2covar or marker2covar before they can be recognized by this function.
group.labels	A vector of character strings naming the groups that are specified by the covariates. For example, if coloring by the covariate "sex," coded as 0 for females and 1 for males, the group labels would be c("female", "male"). The vector must be the same length as the number of groups, and must match the numerical order of the groups.
pheno.labels	An optional vector of character strings giving alternate names for the phenotypes being plotted.

See Also

pheno2covar, marker2covar, plotPhenoCor

---

plotPhenoCor *Plot correlations between phenotype pairs.*

Description

This function plots pairs of phenotypes against each other to examine phenotype correlations. CAPE performs best when phenotypes are moderately correlated ($0.4 < r < 0.8$). This script can help prioritize phenotypes for use in CAPE. Phenotype pairs are plotted in a panel in which the lower triangle shows points colored by the indicated covariate, and the upper triangle shows correlation statistics.

Usage

```
plotPhenoCor(data.obj, pheno.which = NULL,
color.by = NULL, group.labels = NULL,
text.cex = 1, pheno.labels = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
pheno.which	A vector of character strings indicating which phenotypes should be plotted. If pheno.which is NULL, all phenotypes will be plotted.
color.by	A vector of character strings indicating how the points should be colored. Points are colored based on covariates. These covariates must be specified using pheno2covar or marker2covar before they can be recognized by this function.
group.labels	A vector of character strings naming the groups that are specified by the covariates. For example, if coloring by the covariate "sex," coded as 0 for females and 1 for males, the group labels would be c("female", "male"). The vector must be the same length as the number of groups, and must match the numerical order of the groups.
text.cex	A numerical value indicating the size of the text showing the correlation values between phenotypes.
pheno.labels	An optional vector of character strings giving alternate names for the phenotypes being plotted.

See Also

pheno2covar, marker2covar, plotPheno

plotSinglescan *Plot the results of singlescan*

Description

This function plots the results obtained from the single-marker regression performed by singlescan. The effects (β) of each regression on each phenotype or eigentrait are plotted as a vertical line. Chromosomes and traits for plotting can be specified.

Usage

```
plotSinglescan(data.obj, singlescan.obj,
chr = NULL, traits = NULL, show.alpha.values = NULL,
standardized = TRUE, show.marker.labels = FALSE,
mark.covar = FALSE, mark.chr = TRUE, plot.type = "h",
overlay = FALSE, trait.colors = NULL,
show.rejected.markers = FALSE, show.selected.markers = FALSE,
relative.spacing = FALSE, gap.grades = 10,
min.gap = 1, max.gap = 10, min.gap.genome = 1000,
max.gap.genome = 1000000, fixed.scale = FALSE,
ymax = NULL, cex.axis = 0.5, cex.points = 1,
cex.alpha = 0.5, cex.labels = 2, cex.legend = 0.7)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
singlescan.obj	The object in which the results from singlescan are stored.
chr	An optional vector indicating which chromosomes should be plotted. If NULL, the default, all chromosomes are plotted.
traits	An optional vector indicating which traits should be plotted. If NULL, the default, all traits are plotted.
show.alpha.values	A numeric vector indicating which alpha values specified in singlescan should be plotted. If NULL, all alpha values calculated in singlescan are plotted.
standardized	A logical value. If TRUE, the absolute value of the regression t statistics (β/σ) are plotted. If FALSE, the raw regression coefficients (β) are plotted.
show.marker.labels	A logical value indicating whether marker names should be printed along the plot axes.
mark.covar	A logical value. If TRUE, the covarates for the pair scan are marked in red. If FALSE, all markers are plotted in black.
mark.chr	A logical value. If TRUE, alternating chromosomes are shaded in gray to aid in visualizing chromosome boundaries.
plot.type	A character indicating whether the marker effects should be plotted as vertical lines ("h"), points ("p"), lines ("l"), or both lines and points ("b").

overlay	A logical value indicating whether plots of scans should be overlayed on top of each other or plotted in separate panels.
trait.colors	If overlay is TRUE, this argument specifies different colors for the ovelayed scans. There are four default colors (black, blue, purple, darkgreen). Alternate or additional colors can be specified manally.
show.rejected.markers	A logical value. If select.markers.for.pairscan has been run, setting this value to TRUE places indicators above markers that have been rejected from being included in the pair scan. show.selected.markers and show.rejected markers cannot be set to TRUE simultaneously.
show.selected.markers	A logical value. If select.markers.for.pairscan has been run, setting this value to TRUE places indicators above markers that have been selected for the pair scan.
relative.spacing	A logical value indicating whether markers should be plotted at even intervals (FALSE) or spaced relative to their actual genomic locations (TRUE).
min.gap	The minimum plotted gap between markers.
max.gap	The maximum plotted gap between markers.
min.gap.genome	The minimum gap between markers in genomic distance.
max.gap.genome	The maximum gap between markers in genomic distance.
gap.grades	An integer indicating how finely to bin between minimum gap distance and maximum gap distance.
fixed.scale	A logical value to indicate whether the y axes of multiple plots should have the same minimum and maximum.
ymax	An optional numeric value indicating a fixed maximum for the y axes.
cex.axis	A numeric value indicating the size of the font for the axes.
cex.points	A numeric value indicating the pointsize or line width for data plotting.
cex.alpha	A numeric value indicating the font size for writing alpha values.
cex.labels	A numeric value indicating the font size for labels.
cex.legend	A numeric value indicating the size of the legend.

Value

Nothing is returned from this function. It produces a plot of the effects

See Also

singlescan, select.markers.for.pairscan

Examples

```
# plot all markers and both eigentraits
## Not run: plotSinglescan(obesity.cross)
# plot only results from chromosomes 1 through 4
## Not run: plotSinglescan(obesity.cross, chr = c(1:4))
```

plotSinglescan.heat *Plot the results of singlescan as a heatmap*

Description

This function plots the results obtained from the single-marker regression performed by singlescan. The effects (β) of each regression on each phenotype or eigentrait are plotted as heatmap. This method allows better comparison of effects between phenotypes than plotSinglescan.

Usage

```
plotSinglescan.heat(data.obj, singlescan.obj,
standardized = TRUE, show.marker.labels = FALSE,
show.chr.boundaries = TRUE, label.chr = TRUE,
threshold.above = NULL, color = "blue",
light.dark = "light", scale.fun = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
singlescan.obj	The object in which the results of singlescan are stored.
standardized	A logical value. If TRUE, the absolute value of the regression t statistics (β/σ) are plotted. If FALSE, the raw regression coefficients (β) are plotted.
show.marker.labels	A logical value indicating whether marker names should be printed along the plot axes.
show.chr.boundaries	A logical value. If TRUE the boundaries of the chromosomes are marked with vertical lines.
label.chr	A logical value. If TRUE, the chromosomes are labeled with their numbers
threshold.above	A numeric value. Effects above this value are colored white. This is useful in preventing large effects from swamping out differences between smaller effects.
color	The color to be used in the raster image.
light.dark	Either "light", or "dark" to indicate the range of colors that should be plotted.
scale.fun	The name of function, such as sqrt or log10, that will scale the values in the matrix to increase contrast in the image.

Value

Nothing is returned from this function. It produces a heatmap of the effects

See Also

singlescan, plotSinglescan

plotSVD *Plot the results of the singular value decomposition of the phenotype matrix*

Description

This function plots the results of the singular value decomposition (SVD) of the phenotype matrix. This plot is useful for selecting eigentraits for further analysis.

Usage

```
plotSVD(data.obj, orientation = c("vertical", "horizontal"),
neg.col = "blue", pos.col =  "brown", light.dark = "dark")
```

Arguments

data.obj	The object in which all results are stored. See read.population.
orientation	A character string ("vertical", or "horizontal") indicating whether the plot should be vertically or horizontally oriented.
neg.col	The color in which to plot the negative values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."
pos.col	The color in which to plot the positive values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."
light.dark	Either "light" or "dark" depending on the desired range of colors used.

Value

Invisibly returns the fraction of variance accounted for by each ET.

See Also

get.eigentraits

plotVariantInfluences *Plot variant-to-variant influences*

Description

This function plots the reparameterized influences of variants on each other. The epistatic interactions from the pairwise scan are reparameteriezed to the terms m_{12} and m_{21}, where the subscripts indicate the source and target variants respectively. These terms are interpreted as the effect that the source variant exerts on the target variant when both are present. Negative influences represent suppression while positive influences represent enhancement.

Usage

```
plotVariantInfluences(data.obj, p.or.q = 0.05,
min.std.effect = 0, plot.all.vals = FALSE,
standardize = TRUE, pos.col = "brown",
neg.col = "blue", not.tested.col = "lightgray",
light.dark = "light", show.marker.labels = FALSE,
show.chr = TRUE, label.chr = TRUE,
scale.effects = c("log10", "sqrt", "none"),
pheno.width = 11, covar.width = 11,
covar.labels = NULL, phenotype.labels = NULL)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
p.or.q	A threshold indicating the maximum adjusted p value considered significant. If an fdr method has been used to correct for multiple testing, this value specifies the maximum q value considered significant. Only marker pairs with p or q values below this threshold will be plotted.
min.std.effect	A numerical threshold indicating the absolute value of the minimum standard effect size to be shown in the plot. The default value of 0 performs no thresholding.
plot.all.vals	A logical value indicating whether all values should be plotted regardless of significance. If TRUE, the significant values are highlighted in bright colors. If FALSE, non-significant values are not plotted.
neg.col	The color in which to plot the negative values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."
pos.col	The color in which to plot the positive values. Colors can be one of the following: "green", "purple", "orange", "blue", "brown", "yellow", or "gray."
standardize	A logical value. If TRUE the values in each entry of the plotted matrix are the standardized effect sizes β/σ. If FALSE, the raw β values are plotted.
not.tested.col	A color name used to mark variant pairs that were not tested due to linkage. The color distinguishes these pairs from those that were tested, but do not have significant interactions. The default color is light "gray". This can be changed to "white" or FALSE if no marking is desired.
light.dark	Either "light" or "dark" to indicate what color scheme should be used for plotting.
show.marker.labels	A logical value indicating whether the marker labels should be printed along the plot axes.
show.chr	A logical value. If TRUE, chromosomes are indicated by alternating gray and white blocks along plot axes.
label.chr	A logical value. If TRUE and if show.chr is TRUE, chromosome numbers are printed inside each gray and white block.
scale.effects	A string indicating a scaling function by which the effects should be scaled. This is useful in increasing contrast between effects with large variance.

pheno.width — A numeric value indicating the width of the phenotypes relative to the width of each cell in the interaction matrix. Each cell in the interaction matrix has a width of 1, so a pheno.width of 10 makes the phenotypes 10 times wider for ease of viewing.

covar.width — A numeric value indicating the width of the covariates relative to the width of each cell in the interaction matrix. Each cell in the interaction matrix has a width of 1, so a covar.width of 10 makes the covariates 10 times wider for ease of viewing.

covar.labels — An optional vector of strings specifying a label for each covariate. If this argument is NULL, the covariate names from the data object will be used.

phenotype.labels — An optional vector of strings specifying a label for each phenotype. If this argument is NULL, the phenotype names from the data object will be used.

Value

This function invisibly returns the plotted influence matrix.

See Also

pairscan

---

qqPheno *Plot qq plots of phenotype pairs.*

Description

This function plots qq plots of pairs phenotypes.

Usage

```
qqPheno(data.obj, pheno.which = NULL,
pheno.labels = NULL)
```

Arguments

data.obj — The object in which all results are stored. See read.population.

pheno.which — A vector of character strings indicating which phenotypes should be plotted. If pheno.which is NULL, all phenotypes will be plotted.

pheno.labels — An optional vector of character strings giving alternate names for the phenotypes being plotted.

See Also

plotPhenoCor, plotPheno, histPheno

read.geno *Read in and format data for analysis by* cape

Description

This function reads in genotypd data for cape analysis and formats it into a genotype object used by other functions in cape. The file can be in cape format (See read.population), a csv file, or a compressed RData file generated by saveRDS. See Details for further descriptions of the files.

Usage

```
read.geno(file.format = c("cape", "csv", "rdata"),
filename = NULL, geno.col = NULL, delim = ",",
na.strings = "-", check.chr.order = TRUE)
```

Arguments

file.format	A character string indicating which of the accepted formats describes the file to be read in. See Details for specifics.
filename	An optional character string with path name specifying the file to be read in. Omission of this argument will prompt a dialog box for selecting a file.
geno.col	An optional numeric vector specifying which columns the genotypes of interest are in. If omitted, all genotypes are read in.
delim	A character string indicating the delimeter in the data file. The default indicates a comma-separated file (",").
na.strings	The symbol used to denote missing data in the file. Misspecifying this character can lead to errors in processing the file in which cape misstakenly thinks some phenotypes have character values in them.
check.chr.order	A logical value indicating whether the order of the chromosomes should be checked. In general, chromosomes should be entered in increasing numerical value. CAPE does not sort chromosomes, and they will be plotted in the order in which they are entered. If the chromosomes have non-numeric and non-X or Y names, and cannot be checked appropriately, or an alternate order is desired, set check.chr.order to FALSE.

Details

Genotype data can be contained in one of three file formats: cape, csv, or rdata. For a description of the cape format, see read.population. The csv format must contain the following:

- header: A header labeling each column is required. The headers typically contain a name for each marker, for example "D15MIT80."
- chromosomes: The second line of the file must contain the chromosome on which each marker is found.

- `marker location`: The third line of the file must contain the chromosomal locations of the markers.
- `genotypes`: Genotypes may be coded in one of three different formats: (1) As letters, for example A,H,B, indicating homozygous for allele 1, heterozygous, and homozygous for allele 2 respectively. "H" must be used for heterozygotes, but the other genotypes may be coded with any other letters. (2) As the numbers 0,1,2 indicating homozygous for allele 1, heterozygous, and homozygous for allele 2 respectively. (3) As continuous probabilities of the presence of the reference allele. An individual homozygous for allele 1 would be coded as 0, a heterozygous individual as 0.5, and an individual homozygous for allele 2 as 1. The continuous probabilities allow for uncertainty in genotyping that is not automatically available in the A,H,B or 0,1,2 encodings.

The rdata format follows the same format as the csv file, but is used for large data that cannot be reasonably stored in csv format. The file should be saved using the function `saveRDS`

Value

This function is used primarily when genotype data are too large to include in the data.obj, but can be used for small data as well. This function converts genotype data into a list object called the geno.obj. Upon creation the geno.obj contains five elements: "geno", "marker.names", "chromosome", "marker.location", "marker.num"

- genoA matrix containing the genotype data for the population. Each genotype is stored in a column, and individuals are stored in rows. Regardless of original format, the genotypes are converted to probabilities for in the data object. Genotypes originally coded as A,H,B for example, will be encoded as 0,0.5,1 respectively.
- marker.names A vector containing the names of all markers.
- chromosomeA vector containing the chromosome on which each marker is found.
- marker.locationA vector containing the chromosomal position of each marker.
- marker.numA vector containing a numerical identifier for each marker.

These elements are used by future functions to identify markers and should not be changed. After running both `read.population` and `read.geno`, the function `make.data.obj` should be run to transfer marker sinformation from the geno.obj to the data.obj.

See Also

`read.population`, `read.pheno`, `make.data.obj`

`read.pheno` *Read in and format data for analysis by* `cape`

Description

This function reads in data for `cape` analysis and formats it into an object used by other functions in `cape`. A single comma-separated file containing both phenotype and genotype data is required. Chromosome and marker locations are required for each marker, and markers are assumed to be in order.

Usage

```
read.pheno(file.format = c("cape", "csv"), filename = NULL,
pheno.col = NULL, id.col = 1, delim = ",", na.strings = "-")
```

Arguments

file.format	A character string indicating which of the accepted formats describes the file to be read in. See Details for specifics.
filename	An optional character string with path name specifying the file to be read in. Omission of this argument will prompt a dialog box for selecting a file.
pheno.col	An optional numeric vector specifying which columns the phenotypes of interest are in. If omitted, all phenotypes are read in.
id.col	An integer indicating in which column the individual IDs are stored.
delim	A character string indicating the delimeter in the data file. The default indicates a comma-separated file (",").
na.strings	The symbol used to denote missing data in the file. Misspecifying this character can lead to errors in processing the file in which cape misstakenly thinks some phenotypes have character values in them.

Details

Phenotype data can be contained in one of two file formats: cape, or csv. For a description of the cape format, see read.population. The csv format must contain the following:

- header: A header labeling each column is required
- phenotypes: All phenotypes are required to be numeric. Phenotypes that are not numeric must be coded numerically. For example sex can be coded as [0,1]. Missing values are indicated with the symbol specified by na.strings. The default symbol for na.strings is '-'

Value

This function is typically used in conjunction with read.geno and make.data.obj when genotype data are too large to include in the data.obj, but can be used for small data as well. read.pheno initializes the data.obj with the phenotype matrix element "pheno" (see below). After read.geno is run, to read in the genotype data separately, make.data.obj is run to transfer information from the geno.obj to the initialized data.obj.

pheno	A matrix containing the phenotype data for the population. Each phenotype is stored in a column, and individuals are stored in rows.

See Also

read.population, read.geno, make.data.obj

read.population *Read in and format data for analysis by* cape

Description

This function reads in data for cape analysis and formats it into an object used by other functions in cape. A single comma-separated file containing both phenotype and genotype data is required. Chromosome and marker locations are required for each marker, and markers are assumed to be in order.

Usage

```
read.population(filename = NULL, pheno.col = NULL,
geno.col = NULL, delim = ",", na.strings = "-",
check.chr.order = TRUE)
```

Arguments

filename	An optional character string with path name specifying the file to be read in. Omission of this argument will prompt a dialog box for selecting a file.
pheno.col	An optional numeric vector specifying which columns the phenotypes of interest are in. If omitted, all phenotypes are read in.
geno.col	An optional numeric vector specifying which columns the genotypes of interest are in. If omitted, all genotypes are read in.
delim	A character string indicating the delimeter in the data file. The default indicates a comma-separated file (",").
na.strings	The symbol used to denote missing data in the file. Misspecifying this character can lead to errors in processing the file in which cape misstakenly thinks some phenotypes have character values in them.
check.chr.order	A logical value indicating whether the order of the chromosomes should be checked. In general, chromosomes should be entered in increasing numerical value. CAPE does not sort chromosomes, and they will be plotted in the order in which they are entered. If the chromosomes have non-numeric and non-X or Y names, and cannot be checked appropriately, or an alternate order is desired, set check.chr.order to FALSE.

Details

All phenotype and genotype data must be contained in a single comma-separated file. The phenotypes should be listed in columns at the beginning of the file, followed by the genotype data. Each row of the file corresponds to one individual. The file must contain the following attributes:

- header: A header labeling each column is required

- `chromosomes`: The second line of the file must contain the chromosome on which each marker is found. This line should begin with empty spaces in the phenotype columns followed by a chromosome label for each marker.
- `marker location`: The third line of the file must contain the chromosomal locations of the markers. Like the line of chromosome labels, this line should begin with empty spaces in the phenotype columns followed by a chromosomal position for each marker.
- `phenotypes`: The phenotypes must be listed in the first columns of the file. All phenotypes are required to be numeric. Phenotypes that are not numeric must be coded numerically. For example sex can be coded as [0,1]. Missing values are indicated with the symbol specified by na.strings. The default symbol for na.strings is '-'
- `genotypes`: Genotypes may be coded in one of three different formats: (1) As letters, for example A,H,B, indicating homozygous for allele 1, heterozygous, and homozygous for allele 2 respectively. "H" must be used for heterozygotes, but the other genotypes may be coded with any other letters. (2) As the numbers 0,1,2 indicating homozygous for allele 1, heterozygous, and homozygous for allele 2 respectively. (3) As continuous probabilities of the presence of the reference allele. An individual homozygous for allele 1 would be coded as 0, a heterozygous individual as 0.5, and an individual homozygous for allele 2 as 1. The continuous probabilities allow for uncertainty in genotyping that is not automatically available in the A,H,B or 0,1,2 encodings.

Value

The file is converted to a list object that is used as the main argument in most functions. This object is always referred to as data.obj. All data and analysis results will eventually be stored in this object. Upon creation the data.obj contains four elements:

`pheno`	A matrix containing the phenotype data for the population. Each phenotype is stored in a column, and individuals are stored in rows.
`geno`	A matrix containing the genotype data for the population. Each genotype is stored in a column, and individuals are stored in rows. Regardless of original format, the genotypes are converted to probabilities for in the data object. Genotypes originally coded as A,H,B for example, will be encoded as 0,0.5,1 respectively.
`chromosome`	A vector containing the chromosome on which each marker is found.
`marker.location`	A vector containing the chromosomal position of each marker.

`remove.ind` *Remove individuals from the data.obj*

Description

This function removes individuals from a data set.

Usage

```
remove.ind(data.obj, ind.which)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
ind.which	A numerical vector indicating which individuals should be removed.

Value

The data object is returned with the specified individuals removed from the phenotype matrix.

See Also

remove.markers

---

remove.markers *Remove markers from the data.obj*

Description

This function removes genetic markers from a data set.

Usage

```
remove.markers(data.obj, markers.which)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
markers.which	A vector indicating the names or ID numbers of which markers should be removed.

Value

The data object is returned with the specified markers removed.

See Also

remove.ind

select.by.chr *Subset a cross object to include only specified chromosomes.*

Description

This function subsets a data object to include only specified chromosomes.

Usage

```
select.by.chr(data.obj, chr)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
chr	A vector of desired chromosomes

Value

This function returns the data object with the genotype matrix pared down to only the desired chromosomes.

See Also

select.by.ind

select.by.ind *Subset a cross object to include specific individuals*

Description

This function subsets a cross to include individuals based on either phenotypic or genotypic values. For example, this function can subset a cross to include all individuals with a phenotype value greater than x or a genotype value equal to y.

Usage

```
select.by.ind(data.obj, geno.or.pheno = pheno, expr)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.or.pheno	A character value, either "geno", or "pheno" to specify which matrix should be used to subset individuals.
expr	An quoted expression used to subset individuals.

Value

The cross object is returned including only the individuals meeting the criteria in the provided expression.

See Also

select.by.chr

Examples

```
data(obesity.cross)
hist(obesity.cross$pheno[,"insulin"], main = "original insulin distribution",
xlab = "insulin (ng/mL)", xlim = c(0, 30))
obesity.cross <- select.by.ind(obesity.cross, "pheno", "insulin < 25")
hist(obesity.cross$pheno[,"insulin"], main = "subset insulin distribution",
xlab = "insulin (ng/mL)", xlim = c(0, 30))
```

---

select.eigentraits *Select a subset of the eigentraits for further analysis*

Description

This function selects the specified eigentraits for further analysis. After the singular value decomposition (SVD) of multiple traits by get.eigentraits, a subset of the eigentraits, for example the first two, may carry useful information, while others may be dominated by noise. In this case, this function can be used to select the first two eigentraits for use in the analysis.

Usage

```
select.eigentraits(data.obj, traits.which = c(1, 2))
```

Arguments

data.obj	The object in which all results are stored. See read.population.
traits.which	A numeric vector indicating which eigentraits should be retained for further analysis.

Details

Before the selection of eigentraits, the eigentraits should be examined using plotSVD

Value

This function returns the data object with only the selected eigentraits.

References

Carter, G. W., Hays, M., Sherman, A., & Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. PLoS genetics, 8(10), e1003010. doi:10.1371/journal.pgen.1003010

See Also

get.eigentraits, plotSVD

---

select.markers.for.pairscan *A required step that filters variable and non-redundant markers for the pairscan*

Description

This function selects markers for the pair scan. If the markers are to be thresholded by a significance cutoff, this function filters them. It then checks to make sure all pairs of markers have had at least one recombination between them, and that all markers are variable across individuals. Mono-allelic markers are removed. If any pair of markers carry identical genotype information, the first marker of the pair is discarded.

Usage

```
select.markers.for.pairscan(data.obj, geno.obj = NULL,
singlescan.obj, alpha.thresh = NULL,
t.thresh = NULL, specific.markers = NULL,
num.markers = NULL, step.size = NULL,
tolerance = 10, verbose = TRUE)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which the genotype data is stored. See read.geno.
singlescan.obj	The object in which results from singlescan are stored.
alpha.thresh	A numeric threshold for selecting markers based on significance at the given alpha value.
t.thresh	A numerical value indicating a standardized effect size below which markers are rejected for the pair scan.
specific.markers	An optional vector of column numbers or names specifying specific markers to include in the pairscan.
num.markers	An optional number. If specified, the algorithm attempts to find a standardized effect threshold that will give the number of markers specified. This can take a long time, since linear non-independence must be calculated for each thresholded set of markers.
step.size	If num.markers is specified, this argument determines how the t.thresh is changed when searching for markers to include in the pairscan.

tolerance	If num.markers is specified, tolerance determines how close the algorithm will get to the number of desired markers. If 100 markers are desired, and tolerance is set to 10, the algorithm will stop if it finds a threshold giving between 90 and 110 markers.
verbose	A logical value indicating whether the progress of the threshold finding algorithm should be reported.

Details

The markers that were selected for independence can be visualized with the function plotSinglescan with either show.selected.markers or show.rejected.markers set to TRUE.

Value

This function returns the data object with an added genotype matrix to be used in the pair scan. This matrix has the same structure as the original genotype matrix, but contains only the filtered markers.

See Also

singlescan, plotSinglescan

---

select.pheno *Select phenotypes for analysis*

Description

This function is used to pare down a matrix of phenotypes to only those that will be included in the analysis.

Usage

```
select.pheno(data.obj, phenotypes)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
phenotypes	A vector of either column numbers of the desired phenotypes, or character strings that uniquely identify the columns containing the desired phenotypes.

Value

This function returns the data object with a phenotype matrix that contains only the specified phenotypes.

Examples

```
data(obesity.cross)
obesity.cross <- select.pheno(obesity.cross, c("glucose", "body_weight"))
```

singlescan *Run the single-variant regression for all phenotypes*

Description

This function runs the single-variant regression for either raw phenotypes or eigentraits using a kinship correction if desired. It also performs permutation testing and calculates significance thresholds.

Usage

```
singlescan(data.obj, geno.obj = NULL, n.perm = NULL,
covar = NULL, alpha = c(0.01, 0.05),
scan.what = c("eigentraits", "raw.traits"),
use.kinship = FALSE, kin.full.geno = FALSE,
run.parallel = TRUE, sample.kinship = FALSE,
num.kin.samples = 1000, n.per.sample = 10,
verbose = FALSE, overwrite.alert = TRUE, n.cores = 2)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
geno.obj	The object in which the genotype matrix and marker information are stored. See read.geno.
n.perm	The number of permutations to be performed.
covar	A vector of marker names to be used as covariates. See pheno2covar and marker2covar.
alpha	A vector of alpha values for which significant standardized effect values will be calculated.
scan.what	A character string indicating uniquely whether raw traits or eigentraits should be tested.
use.kinship	A logical value indicating whether a kinship correction should be implemented.
kin.full.geno	A logical value indicating whether the entire genotype matrix should be used to calculate kinship corrections. If FALSE, only the subset of genetic markers being scanned will be used to calculate kinship corrections.
run.parallel	A logial value indicating whether to run calculations in parallel.
sample.kinship	A logical value indicating whether kinship matrices should be sampled (TRUE) or calculated directly (FALSE). Sampling kinshp matrices can be faster in the case of very large genotype matrices.
num.kin.samples	If sample.kinship is TRUE, this integer indicates how many samples should be used to calculate kinship matrices.
n.per.sample	If sample.kinship is TRUE, this integer indicates how many markers should be used in each sample of the genotype matrix for calculating kinship matrices.

`verbose` A logical value indicating whether the progress of the scan should be printed to the screen.

`overwrite.alert` If TRUE, this triggers an alert warning the user that the output of this function should be saved separately from the data.obj.

`n.cores` An integer specifying the number of cores to be used in parallel processing.

Value

IMPORTANT NOTE: Prior versions of this function modified the data.obj. The new version creates a separate singlescan.obj, and the output of this function should NOT be assigned to the data.obj. The singlescan.obj is a list containing the following four elements:

`alpha` A vector containing the alpha values requested by the user.

`alpha.thresh` A list containing the same number of elements as alpha values requested. For each value, a significance threshold in terms of t statistic is given.

`singlescan.results` A list in which each element corresponds to one trait being scanned. Each element contains a table with the results from the single-marker scan. The table includes, in columns, the effect size marker, the standard error of the effect size, the t statistic from the regression, and the p value. There is one row for each marker.

References

Carter, G. W., Hays, M., Sherman, A., & Galitski, T. (2012). Use of pleiotropy to model genetic interactions in a population. PLoS genetics, 8(10), e1003010. doi:10.1371/journal.pgen.1003010

See Also

`plotSinglescan`

---

sortCross *Sort the genetic markers in the data.obj.*

Description

This function sorts genetic markers based on chromosome and chromosomal position.

Usage

```
sortCross(data.obj, geno.obj)
```

Arguments

`data.obj` The object in which all results are stored. See `read.population`.

`geno.obj` The object in which the genotype matrix and marker information are stored. See `read.geno`.

Value

This function returns the data.obj with the genetic markers sorted by chromosome and position.

`writePopulation` *Write out a cape data object to .csv format.*

Description

This function takes in a data object and writes it to a .csv file. This file can be read in by `read.population`.

Usage

```
writePopulation(data.obj, filename,
geno.to.convert = NULL, conversion = NULL)
```

Arguments

`data.obj`	The object in which all results are stored. See `read.population`.
`filename`	A string specifying the name of the file to be written.
`geno.to.convert`	A vector of genotypes in the data object that are to be converted to another form. For example, to convert genotypes coded as (0, 0.5, 1), to ("A", "H", "B"), set geno.to.convert to c(0, 0.5, 1) and conversion to c("A", "H", "B").
`conversion`	A vector the same length as geno.to.convert. The values in this vector will be written in place of the values in geno.to.convert in the final file.

Value

This function writes a data object to a .csv file that can be read by `read.population`. Nothing is returned.

See Also

`read.population`

writeVariantInfluences *Write the final results to a file*

Description

This function takes in the final data object and writes the variant influences that are at or below the specified significance level to a file in the current working directory.

Usage

```
writeVariantInfluences(data.obj, p.or.q = 0.05,
filename = "Variant.Influences.csv", delim = ",",
mark.covar = FALSE, write.file = TRUE)
```

Arguments

data.obj	The object in which all results are stored. See read.population.
p.or.q	A threshold indicating the maximum adjusted p value considered significant. If an fdr method has been used to correct for multiple testing, this value specifies the maximum q value considered significant. Only marker pairs with p or q values below this threshold will be plotted.
filename	A character vector specifying the name of the file.
delim	A character string indicating the delimeter in the data file. The default indicates a comma-separated file (",").
mark.covar	A logical value. If TRUE, an asterisk is appended the names of markers used as covariates in the pair scan.
write.file	A logical value indicating whether the table should be written to a file or simply returned.

Value

This function writes a table of direct influences to a file. It also returns the table invisibly, i.e. if the output of the function is assigned to an object, the object will contain the table of influences. Otherwise, nothing is returned.

Examples

```
# here the table is written to a file, but nothing is returned.
## Not run: writeVariantInfluences(obesity.cross)

# here the table is written to a file, and returned to the
# object sig.table
## Not run:
sig.table <- writeVariantInfluences(obesity.cross)
print(sig.table)
## End(Not run)
```

Index