Package 'cape'

May 19, 2022

Title Combined Analysis of Pleiotropy and Epistasis for Diversity Outbred Mice

Version 3.1.1

Description Combined Analysis of Pleiotropy and Epistasis infers predictive networks between genetic variants and phenotypes. It can be used with standard two-parent populations as well as multi-parent populations, such as the Diversity Outbred (DO) mice, Collaborative Cross (CC) mice, or the multi-parent advanced generation intercross (MAGIC) population of Arabidopsis thaliana. It uses complementary information of pleiotropic gene variants across different phenotypes to resolve models of epistatic interactions between alleles. To do this, cape reparametrizes main effect and interaction coefficients from pairwise variant regressions into directed influence parameters. These parameters describe how alleles influence each other, in terms of suppression and enhancement, as well as how gene variants influence phenotypes. All of the final interactions are reported as directed interactions between pairs of parental alleles. For detailed descriptions of the methods used in this package please see the following references.
Carter, G. W., Hays, M., Sherman, A. & Galitski, T. (2012) <doi:10.1371/journal.pgen.1003010>.
Tyler, A. L., Lu, W., Hendrick, J. J., Philip, V. M. & Carter, G. W. (2013) <doi:10.1371/journal.pcbi.1003270>.

License GPL-3

Encoding UTF-8

LazyData false

Depends R (>= 3.6)

Suggests testthat (>= 2.3.2), knitr (>= 1.29), rmarkdown, parallel

Imports abind, caTools, corpcor, doParallel, evd, foreach, here, igraph, Matrix, pheatmap, pracma, propagate, qtl, qtl2, qtl2convert, R6 (>= 2.4.1), RColorBrewer (>= 1.1-2), regress (>= 1.3-21), shape (>= 1.4.5), stats, tools, utils, yaml (>= 2.2.1)

VignetteBuilder knitr

RoxygenNote 7.1.1

NeedsCompilation no

Author Anna Tyler [aut, cre],
Jake Emerson [aut],
Baha El Kassaby [aut],
Ann Wells [aut],
Georgi Kolishovski [aut],
Vivek Philip [aut],
Gregory Carter [aut]

Maintainer Anna Tyler <anna.tyler@jax.org>

Repository CRAN

Date/Publication 2022-05-19 13:40:02 UTC

R topics documented:

`calc_delta_errors` *Error propagation*

Description

This function performs error propagation on coefficients and standard errors.

Usage

```
calc_delta_errors(markers, beta_m, se, beta_cov)
```

Arguments

`markers`	The marker names being tested
`beta_m`	The main-effects coefficient matrix for the pairwise regression of the given pair.
`se`	The standard errors for the marker pair.
`beta_cov`	The model covariance matrix from the pairwise regression

Value

Returns the error propagated coefficients and standard errors for m12 and m21

calc_emp_p *Calculate empirical p-values*

Description

This function uses ecdf to calculate empirical p values given a null distribution and an observed distribution

Usage

```
calc_emp_p(obs_dist, null_dist)
```

Arguments

obs_dist	The observed distribution
null_dist	The null distribution

Value

An empirical p value for each observed value

calc_p *Calculate P Values for Interactions Based on Permutations*

Description

Calculate P Values for Interactions Based on Permutations

Usage

```
calc_p(
  data_obj,
 pval_correction = c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
    "none")
)
```

Arguments

data_obj	A Cape data object
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.

Value

The data object is returned with a new table called var_to_var_p_val. This table is the same as var_to_var_influences, but with p value and adjusted p value columns appended.

Examples

```
## Not run:
data_obj <- calc_p(data_obj, "fdr")

## End(Not run)
```

`Cape-class` *The CAPE data object*

Description

The CAPE data object

The CAPE data object

Details

Class `Cape` defines a CAPE analysis object.

Slots

`parameter_file` string, full path to YAML file with initialization parameters

`yaml_parameters` string representing YAML CAPE parameters. See the vignette for more descriptions of individual parameter settings.

`results_path` string, full path to directory for storing results (optional, a directory will be created if one is not specified)

`save_results` Whether to save cape results. Defaults to FALSE.

`use_saved_results` Whether to use existing results from a previous run. This can save time if re-running an analysis, but can lead to problems if the old run and new run have competing settings. If errors arise, and use_saved_results is set to TRUE, try setting it to FALSE, or deleting previous results.

`pheno` A matrix containing the traits to be analyzed. Traits are in columns and individuals are in rows.

`chromosome` A vector the same length as the number of markers indicating which chromosome each marker lives on.

`marker_num` A vector the same length as the number of markers indicating the index of each marker

`marker_location` A vector the same length as the number of markers indicating the genomic position of each marker. The positions are primarily used for plotting and can be in base pairs, centiMorgans, or dummy variables.

`marker_selection_method` A string indicating how markers should be selected for the pairscan. Options are "top_effects" or "from_list." If "top_effects," markers are selected using main effect sizes. If "from_list" markers are specified using a vector of marker names. See `select_markers_for_pairscan`.

`geno_names` The dimnames of the genotype array. The genotype array is a three-dimensional array in which rows are individuals, columns are alleles, and the third dimension houses the markers. Genotypes are pulled for analysis using `get_geno` based on geno_names. Only the individuals, alleles, and markers listed in geno_names are taken from the genotype matrix. Functions that remove markers and individuals from analysis always operate on geno_names in addition to other relevant slots. The names of geno_names must be "mouse", "allele", "locus."

`geno` A three dimensional array holding genotypes for each animal for each allele at each marker. The genotypes are continuously valued probabilities ranging from 0 to 1. The dimnames of geno must be "mouse", "allele", and "locus," even if the individuals are not mice.

`geno_for_pairscan` A two-dimensional matrix holding the genotypes that will be analyzed in the pairscan. Alleles are in columns and individuals are in rows. As in the geno array, values are continuous probabilities ranging from 0 to 1.

`peak_density` The density parameter for `select_markers_for_pairscan`. Determines how densely markers under an individual effect size peak are selected for the pairscan if marker_selection_method is TRUE. Defaults to 0.5.

`window_size` The window size used by `select_markers_for_pairscan`. It specifies how many markers are used to smooth effect size curves for automatic peak identification. If set to NULL, window_size is determined automatically. Used when marker_selection_method is TRUE.

`tolerance` The wiggle room afforded to `select_markers_for_pairscan` in finding a target number of markers. If num_alleles_in_pairscan is 100 and the tolerance is 5, the algorithm will stop when it identifies anywhere between 95 and 105 markers for the pairscan.

`ref_allele` A string of length 1 indicating which allele to use as the reference allele. In two-parent crosses, this is usually allele A. In DO/CC populations, we recommend using B as the reference allele. B is the allele from the C57Bl6/J mouse, which is often used as a reference strain.

`alpha` The significance level for calculating effect size thresholds in the `singlescan`. If singlescan_perm is 0, this parameter is ignored.

`covar_table` A matrix of covariates with covariates in columns and individuals in rows. Must be numeric.

`num_alleles_in_pairscan` The number of alleles to test in the pairwise scan. Because Cape is computationally intensive, we usually need to test only a subset of available markers in the pairscan, particularly if the kinship correction is being used.

`max_pair_cor` the maximum Pearson correlation between two markers. If their correlation exceeds this value, they will not be tested against each other in the pairscan. This threshold is set to prevent false positive arising from testing highly correlated markers. If this value is set to NULL, min_per_genotype must be specified.

`min_per_genotype` minimum The minimum number of individuals allowable per genotype combination in the pair scan. If for a given marker pair, one of the genotype combinations is underrepresented, the marker pair is not tested. If this value is NULL, max_pair_cor must be specified.

`pairscan_null_size` The total size of the null distribution. This is DIFFERENT than the number of permutations to run. Each permutation generates n choose 2 elements for the pairscan. So for example, a permutation that tests 100 pairs of markers will generate a null distribution of size 4950. This process is repeated until the total null size is reached. If the null size is set to 5000, two permutations of 100 markers would be done to get to a null distribution size of 5000.

`p_covar` A vector of strings specifying the names of covariates derived from traits. See `pheno2covar`.

`g_covar` A vector of strings specifying the names of covariates derived from genetic markers. See `marker2covar`.

`p_covar_table` A matrix holding the individual values for each trait-derived covariate. See `pheno2covar`.

`g_covar_table` A matrix holding the individual values for each marker-derived covariate. See `marker2covar`.

`model_family` Indicates the model family of the phenotypes This can be either "gaussian" or "binomial". If this argument is length 1, all phenotypes will be assigned to the same family. Phenotypes can be assigned different model families by providing a vector of the same length as the number of phenotypes, indicating how each phenotype should be modeled. See `singlescan`.

`scan_what` A string indicating whether "eigentraits", "normalized_traits", or "raw_traits" should be analyzed. See `get_pheno`.

`ET` A matrix holding the eigentraits to be analyzed.

`singular_values` Added by `get_eigentraits`. A vector holding the singular values from the singular value decomposition of the trait matrix. They are used in rotating the final direct influences back to trait space from eigentrait space. See `get_eigentraits` and `direct_influence`.

`right_singular_vectors` Added by `get_eigentraits`. A matrix containing the right singular vectors from the singular value decomposition of the trait matrix. They are used in rotating the final direct influences back to trait space from eigentrait space. See `get_eigentraits` and `direct_influence`.

`traits_scaled` Whether the traits should be mean-centered and standardized before analyzing.

`traits_normalized` Whether the traits should be rank Z normalized before analyzing.

`var_to_var_influences_perm` added in `error_prop` The list of results from the error propagation of permuted coefficients.

`var_to_var_influences` added in `error_prop` The list of results from the error propagation of coefficients.

`pval_correction` Options are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY","fdr", "none"

`linkage_blocks_collapsed` A list containing assignments of markers to linkage blocks calculated by `linkage_blocks_network` and `plot_network`. In this list there can be multiple markers assigned to a single linkage block.

`linkage_blocks_full` A list containing assignments of markers to linkage blocks when no linkage blocks are calculated. In this list there can only be one marker per "linkage block". See `linkage_blocks_network` and `plot_network`.

`var_to_var_p_val` The final table of cape interaction results calculated by `error_prop`.

`max_var_to_pheno_influence` The final table of cape direct influences of markers to traits calculated by `direct_influence`.

collapsed_net An adjacency matrix holding significant cape interactions between linkage blocks. See plot_network and get_network.

full_net An adjacency matrix holding significant cape interactions between individual markers. See plot_network and get_network.

use_kinship Whether to use a kinship correction in the analysis.

kinship_type Which type of kinship matrix to use. Either "overall" for the overall kinship matrix or "ltco" for leave-two-chromosomes-out.

transform_to_phenospace whether to transform to phenospace or not.

Public fields

parameter_file full path to YAML file with initialization parameters.

yaml_parameters string representing YAML CAPE parameters. See the vignette for more descriptions of individual parameter settings.

results_path string, full path to directory for storing results (optional, a directory will be created if one is not specified).

save_results Whether to save cape results. Defaults to FALSE.

use_saved_results Whether to use existing results from a previous run. This can save time if re-running an analysis, but can lead to problems if the old run and new run have competing settings. If errors arise, and use_saved_results is set to TRUE, try setting it to FALSE, or deleting previous results.

pheno A matrix containing the traits to be analyzed. Traits are in columns and individuals are in rows.

chromosome A vector the same length as the number of markers indicating which chromosome each marker lives on.

marker_num A vector the same length as the number of markers indicating the index of each marker.

marker_location A vector the same length as the number of markers indicating the genomic position of each marker. The positions are primarily used for plotting and can be in base pairs, centiMorgans, or dummy variables.

geno_names The dimnames of the genotype array. The genotype array is a three-dimensional array in which rows are individuals, columns are alleles, and the third dimension houses the markers. Genotypes are pulled for analysis using get_geno based on geno_names. Only the individuals, alleles, and markers listed in geno_names are taken from the genotype matrix. Functions that remove markers and individuals from analysis always operate on geno_names in addition to other relevant slots. The names of geno_names must be "mouse", "allele", "locus."

geno A three dimensional array holding genotypes for each animal for each allele at each marker. The genotypes are continuously valued probabilities ranging from 0 to 1. The dimnames of geno must be "mouse", "allele", and "locus," even if the individuals are not mice.

peak_density The density parameter for select_markers_for_pairscan. Determines how densely markers under an individual effect size peak are selected for the pairscan if marker_selection_method is TRUE. Defaults to 0.5.

`window_size` The window size used by `select_markers_for_pairscan`. It specifies how many markers are used to smooth effect size curves for automatic peak identification. If set to NULL, window_size is determined automatically. Used when marker_selection_method is TRUE.

`tolerance` The wiggle room afforded to `select_markers_for_pairscan` in finding a target number of markers. If num_alleles_in_pairscan is 100 and the tolerance is 5, the algorithm will stop when it identifies anywhere between 95 and 105 markers for the pairscan.

`ref_allele` A string of length 1 indicating which allele to use as the reference allele. In two-parent crosses, this is usually allele A. In DO/CC populations, we recommend using B as the reference allele. B is the allele from the C57Bl6/J mouse, which is often used as a reference strain.

`alpha` The significance level for calculating effect size thresholds in the `singlescan`. If singlescan_perm is 0, this parameter is ignored.

`covar_table` A matrix of covariates with covariates in columns and individuals in rows. Must be numeric.

`num_alleles_in_pairscan` The number of alleles to test in the pairwise scan. Because Cape is computationally intensive, we usually need to test only a subset of available markers in the pairscan, particularly if the kinship correction is being used.

`max_pair_cor` The maximum Pearson correlation between two markers. If their correlation exceeds this value, they will not be tested against each other in the pairscan. This threshold is set to prevent false positive arising from testing highly correlated markers. If this value is set to NULL, min_per_genotype must be specified.

`min_per_genotype` minimum The minimum number of individuals allowable per genotype combination in the pair scan. If for a given marker pair, one of the genotype combinations is underrepresented, the marker pair is not tested. If this value is NULL, max_pair_cor must be specified.

`pairscan_null_size` The total size of the null distribution. This is DIFFERENT than the number of permutations to run. Each permutation generates n choose 2 elements for the pairscan. So for example, a permutation that tests 100 pairs of markers will generate a null distribution of size 4950. This process is repeated until the total null size is reached. If the null size is set to 5000, two permutations of 100 markers would be done to get to a null distribution size of 5000.

`p_covar` A vector of strings specifying the names of covariates derived from traits. See `pheno2covar`.

`g_covar` A vector of strings specifying the names of covariates derived from genetic markers. See `marker2covar`.

`p_covar_table` A matrix holding the individual values for each trait-derived covariate. See `pheno2covar`.

`g_covar_table` A matrix holding the individual values for each marker-derived covariate. See `marker2covar`.

`model_family` Indicates the model family of the phenotypes. This can be either "gaussian" or "binomial". If this argument is length 1, all phenotypes will be assigned to the same family. Phenotypes can be assigned different model families by providing a vector of the same length as the number of phenotypes, indicating how each phenotype should be modeled. See `singlescan`.

`scan_what` A string indicating whether "eigentraits", "normalized_traits", or "raw_traits" should be analyzed. See `get_pheno`.

ET A matrix holding the eigentraits to be analyzed.

singular_values Added by get_eigentraits. A vector holding the singular values from the singular value decomposition of the trait matrix. They are used in rotating the final direct influences back to trait space from eigentrait space. See get_eigentraits and direct_influence.

right_singular_vectors Added by get_eigentraits. A matrix containing the right singular vectors from the singular value decomposition of the trait matrix. They are used in rotating the final direct influences back to trait space from eigentrait space. See get_eigentraits and direct_influence.

traits_scaled Whether the traits should be mean-centered and standardized before analyzing.

traits_normalized Whether the traits should be rank Z normalized before analyzing.

var_to_var_influences_perm added in error_prop. The list of results from the error propagation of permuted coefficients.

var_to_var_influences added in error_prop. The list of results from the error propagation of coefficients.

pval_correction Options are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY","fdr", "none".

var_to_var_p_val The final table of cape interaction results calculated by error_prop.

max_var_to_pheno_influence The final table of cape direct influences of markers to traits calculated by direct_influence.

full_net An adjacency matrix holding significant cape interactions between individual markers. See plot_network and get_network.

use_kinship Whether to use a kinship correction in the analysis.

kinship_type which type of kinship matrix to use

transform_to_phenospace whether to transform to phenospace or not.

Active bindings

geno_for_pairscan geno for pairscan

marker_selection_method marker selection method

linkage_blocks_collapsed linkage blocks collapsed

linkage_blocks_full linkage blocks full

collapsed_net collapsed net

Methods

Public methods:

- Cape$assign_parameters()
- Cape$check_inputs()
- Cape$check_geno_names()
- Cape$new()
- Cape$plotSVD()
- Cape$plotSinglescan()

- Cape$plotPairscan()
- Cape$plotVariantInfluences()
- Cape$plotNetwork()
- Cape$plotFullNetwork()
- Cape$writeVariantInfluences()
- Cape$set_pheno()
- Cape$set_geno()
- Cape$create_covar_table()
- Cape$save_rds()
- Cape$read_rds()

Method `assign_parameters()`**:** Assigns variables from the parameter file to attributes in the Cape object.

Usage:

```
Cape$assign_parameters()
```

Method `check_inputs()`**:** Checks the dimensionality of inputs and its consistency.

Usage:

```
Cape$check_inputs()
```

Method `check_geno_names()`**:** Checks genotype names.

Usage:

```
Cape$check_geno_names()
```

Method `new()`**:** Initialization method.

Usage:

```
Cape$new(
  parameter_file = NULL,
  yaml_parameters = NULL,
  results_path = NULL,
  save_results = FALSE,
  use_saved_results = TRUE,
  pheno = NULL,
  chromosome = NULL,
  marker_num = NULL,
  marker_location = NULL,
  geno_names = NULL,
  geno = NULL,
  .geno_for_pairscan = NULL,
  peak_density = NULL,
  window_size = NULL,
  tolerance = NULL,
  ref_allele = NULL,
  alpha = NULL,
  covar_table = NULL,
  num_alleles_in_pairscan = NULL,
```

```
  max_pair_cor = NULL,
  min_per_genotype = NULL,
  pairscan_null_size = NULL,
  p_covar = NULL,
  g_covar = NULL,
  p_covar_table = NULL,
  g_covar_table = NULL,
  model_family = NULL,
  scan_what = NULL,
  ET = NULL,
  singular_values = NULL,
  right_singular_vectors = NULL,
  traits_scaled = NULL,
  traits_normalized = NULL,
  var_to_var_influences_perm = NULL,
  var_to_var_influences = NULL,
  pval_correction = NULL,
  var_to_var_p_val = NULL,
  max_var_to_pheno_influence = NULL,
  full_net = NULL,
  use_kinship = NULL,
  kinship_type = NULL,
  transform_to_phenospace = NULL,
  plot_pdf = NULL
)
```

Arguments:

`parameter_file` string, full path to YAML file with initialization parameters

`yaml_parameters` string representing YAML CAPE parameters. See the vignette for more descriptions of individual parameter settings.

`results_path` string, full path to directory for storing results (optional, a directory will be created if one is not specified)

`save_results` Whether to save cape results. Defaults to TRUE.

`use_saved_results` Whether to use existing results from a previous run. This can save time if re-running an analysis, but can lead to problems if the old run and new run have competing settings. If errors arise, and use_saved_results is set to TRUE, try setting it to FALSE, or deleting previous results.

`pheno` A matrix containing the traits to be analyzed. Traits are in columns and individuals are in rows.

`chromosome` A vector the same length as the number of markers indicating which chromosome each marker lives on.

`marker_num` A vector the same length as the number of markers indicating the index of each marker

`marker_location` A vector the same length as the number of markers indicating the genomic position of each marker. The positions are primarily used for plotting and can be in base pairs, centiMorgans, or dummy variables.

`geno_names` The dimnames of the genotype array. The genotype array is a three-dimensional array in which rows are individuals, columns are alleles, and the third dimension houses

the markers. Genotypes are pulled for analysis using get_geno based on geno_names. Only the individuals, alleles, and markers listed in geno_names are taken from the genotype matrix. Functions that remove markers and individuals from analysis always operate on geno_names in addition to other relevant slots. The names of geno_names must be "mouse", "allele", "locus."

geno A three dimensional array holding genotypes for each animal for each allele at each marker. The genotypes are continuously valued probabilities ranging from 0 to 1. The dimnames of geno must be "mouse", "allele", and "locus," even if the individuals are not mice.

.geno_for_pairscan A two-dimensional matrix holding the genotypes that will be analyzed in the pairscan. Alleles are in columns and individuals are in rows. As in the geno array, values are continuous probabilities ranging from 0 to 1.

peak_density The density parameter for select_markers_for_pairscan. Determines how densely markers under an individual effect size peak are selected for the pairscan if marker_selection_method is TRUE. Defaults to 0.5.

window_size The window size used by select_markers_for_pairscan. It specifies how many markers are used to smooth effect size curves for automatic peak identification. If set to NULL, window_size is determined automatically. Used when marker_selection_method is TRUE.

tolerance The wiggle room afforded to select_markers_for_pairscan in finding a target number of markers. If num_alleles_in_pairscan is 100 and the tolerance is 5, the algorithm will stop when it identifies anywhere between 95 and 105 markers for the pairscan.

ref_allele A string of length 1 indicating which allele to use as the reference allele. In two-parent crosses, this is usually allele A. In DO/CC populations, we recommend using B as the reference allele. B is the allele from the C57Bl6/J mouse, which is often used as a reference strain.

alpha The significance level for calculating effect size thresholds in the singlescan. If singlescan_perm is 0, this parameter is ignored.

covar_table A matrix of covariates with covariates in columns and individuals in rows. Must be numeric.

num_alleles_in_pairscan The number of alleles to test in the pairwise scan. Because Cape is computationally intensive, we usually need to test only a subset of available markers in the pairscan, particularly if the kinship correction is being used.

max_pair_cor the maximum Pearson correlation between two markers. If their correlation exceeds this value, they will not be tested against each other in the pairscan. This threshold is set to prevent false positive arising from testing highly correlated markers. If this value is set to NULL, min_per_genotype must be specified.

min_per_genotype minimum The minimum number of individuals allowable per genotype combination in the pair scan. If for a given marker pair, one of the genotype combinations is underrepresented, the marker pair is not tested. If this value is NULL, max_pair_cor must be specified.

pairscan_null_size The total size of the null distribution. This is DIFFERENT than the number of permutations to run. Each permutation generates n choose 2 elements for the pairscan. So for example, a permutation that tests 100 pairs of markers will generate a null distribution of size 4950. This process is repeated until the total null size is reached. If the null size is set to 5000, two permutations of 100 markers would be done to get to a null distribution size of 5000.

p_covar A vector of strings specifying the names of covariates derived from traits. See pheno2covar.

g_covar A vector of strings specifying the names of covariates derived from genetic markers. See marker2covar.

p_covar_table A matrix holding the individual values for each trait-derived covariate. See pheno2covar.

g_covar_table A matrix holding the individual values for each marker-derived covariate. See marker2covar.

model_family Indicates the model family of the phenotypes This can be either "gaussian" or "binomial". If this argument is length 1, all phenotypes will be assigned to the same family. Phenotypes can be assigned different model families by providing a vector of the same length as the number of phenotypes, indicating how each phenotype should be modeled. See singlescan.

scan_what A string indicating whether "eigentraits", "normalized_traits", or "raw_traits" should be analyzed. See get_pheno.

ET A matrix holding the eigentraits to be analyzed.

singular_values Added by get_eigentraits. A vector holding the singular values from the singular value decomposition of the trait matrix. They are used in rotating the final direct influences back to trait space from eigentrait space. See get_eigentraits and direct_influence.

right_singular_vectors Added by get_eigentraits. A matrix containing the right singular vectors from the singular value decomposition of the trait matrix. They are used in rotating the final direct influences back to trait space from eigentrait space. See get_eigentraits and direct_influence.

traits_scaled Whether the traits should be mean-centered and standardized before analyzing.

traits_normalized Whether the traits should be rank Z normalized before analyzing.

var_to_var_influences_perm added in error_prop The list of results from the error propagation of permuted coefficients.

var_to_var_influences added in error_prop The list of results from the error propagation of coefficients.

pval_correction Options are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY","fdr", "none"

var_to_var_p_val The final table of cape interaction results calculated by error_prop.

max_var_to_pheno_influence The final table of cape direct influences of markers to traits calculated by direct_influence.

full_net An adjacency matrix holding significant cape interactions between individual markers. See plot_network and get_network.

use_kinship Whether to use a kinship correction in the analysis.

kinship_type Which type of kinship matrix to use. Either "overall" or "ltco."

transform_to_phenospace whether to transform to phenospace or not.

plot_pdf logical. If TRUE, results are generated as pdf

Method `plotSVD()`**:** Plot Eigentraits

Usage:

```
Cape$plotSVD(filename)
```

Arguments:

`filename` filename of result plot

Method `plotSinglescan()`**:** Plot results of single-locus scans

Usage:

```
Cape$plotSinglescan(
  filename,
  singlescan_obj,
  width = 20,
  height = 6,
  units = "in",
  res = 300,
  standardized = TRUE,
  allele_labels = NULL,
  alpha = alpha,
  include_covars = TRUE,
  line_type = "l",
  pch = 16,
  cex = 0.5,
  lwd = 3,
  traits = NULL
)
```

Arguments:

`filename` filename of result plot.

`singlescan_obj` a singlescan object from `singlescan`

`width` width of result plot, default is 20.

`height` height of result plot, default is 6.

`units` units of result plot, default is "in".

`res` resolution of result plot, default is 300.

`standardized` If TRUE t statistics are plotted. If FALSE, effect sizes are plotted, default is TRUE

`allele_labels` A vector of labels for the alleles if different that those stored in the data_object.

`alpha` the alpha significance level. Lines for significance values will only be plotted if n_perm > 0 when `singlescan` was run. And only alpha values specified in `singlescan` can be plotted.

`include_covars` Whether to include covariates in the plot.

`line_type` as defined in plot

`pch` see the "points()" R function. Default is 16 (a point).

`cex` see the "points()" R function. Default is 0.5.

`lwd` line width, default is 3.

`traits` a vector of trait names to plot. Defaults to all traits.

Method `plotPairscan()`**:** Plot the result of the pairwise scan

Usage:

```
Cape$plotPairscan(
  filename,
```

```
  pairscan_obj,
  phenotype = NULL,
  show_marker_labels = TRUE,
  show_alleles = FALSE
)
```

Arguments:

`filename` filename of result plot.

`pairscan_obj` a pairscan object from `pairscan`

`phenotype` The names of the phenotypes to be plotted. If NULL, all phenotypes are plotted.

`show_marker_labels` If TRUE marker labels are plotted along the axes. If FALSE, they are omitted.

`show_alleles` If TRUE, the allele of each marker is indicated by color.

Method `plotVariantInfluences()`**:** Plot cape coefficients

Usage:

```
Cape$plotVariantInfluences(
  filename,
  width = 10,
  height = 7,
  p_or_q = p_or_q,
  standardize = FALSE,
  not_tested_col = "lightgray",
  covar_width = NULL,
  pheno_width = NULL
)
```

Arguments:

`filename` filename of result plot.

`width` width of result plot, default is 10.

`height` height of result plot, default is 7.

`p_or_q` A threshold indicating the maximum p value (or q value if FDR was used) of significant interactions and main effects.

`standardize` Whether to plot effect sizes (FALSE) or standardized effect sizes (TRUE), default is TRUE.

`not_tested_col` The color to use for marker pairs not tested. Takes the same values as pos_col and neg_col, default is "lightgray".

`covar_width` See pheno_width. This is the same effect for covariates.

`pheno_width` Each marker and trait gets one column in the matrix. If there are many markers, this makes the effects on the traits difficult to see. pheno_width increases the number of columns given to the phenotypes. For example, if pheno_width = 11, the phenotypes will be shown 11 times wider than individual markers.

Method `plotNetwork()`**:** Plots cape results as a circular network

Usage:

```
Cape$plotNetwork(
  filename,
```

```
  label_gap = 10,
  label_cex = 1.5,
  show_alleles = FALSE
)
```

Arguments:

`filename` filename of result plot.

`label_gap` A numeric value indicating the size of the gap the chromosomes and their labels, default is 10.

`label_cex` A numeric value indicating the size of the labels, default is 1.5.

`show_alleles` TRUE show the alleles, FALSE does not show alleles. Default is FALSE.

Method `plotFullNetwork()`**:** Plot the final epistatic network in a traditional network view.

Usage:

```
Cape$plotFullNetwork(
  filename,
  zoom = 1.2,
  node_radius = 0.3,
  label_nodes = TRUE,
  label_offset = 0.4,
  label_cex = 0.5,
  bg_col = "lightgray",
  arrow_length = 0.1,
  layout_matrix = "layout_with_kk",
  legend_position = "topright",
  edge_lwd = 1,
  legend_radius = 2,
  legend_cex = 0.7,
  xshift = -1
)
```

Arguments:

`filename` filename of result plot.

`zoom` Allows the user to zoom in and out on the image if the network is either running off the edges of the plot or too small in the middle of the plot, default is 1.2.

`node_radius` The size of the pie chart for each node, default is 0.3.

`label_nodes` A logical value indicating whether the nodes should be labeled. Users may want to remove labels for large networks, default is TRUE.

`label_offset` The amount by which to offset the node labels from the center of the nodes, default is 0.4.

`label_cex` The size of the node labels, default is 0.5.

`bg_col` The color to be used in pie charts for non-significant main effects. Takes the same values as pos_col, default is "lightgray".

`arrow_length` The length of the head of the arrow, default is 0.1.

`layout_matrix` Users have the option of providing their own layout matrix for the network. This should be a two column matrix indicating the x and y coordinates of each node in the network, default is "layout_with_kk".

`legend_position` The position of the legend on the plot, default is "topright".

`edge_lwd` The thickness of the arrows showing the interactions, default is 1.

`legend_radius` The size of the legend indicating which pie piece corresponds to which traits, default is 2.

`legend_cex` The size of the labels in the legend, default is 0.7.

`xshift` A constant by which to shift the x values of all nodes in the network, default is -1.

Method `writeVariantInfluences()`**:** Write significant cape interactions to a csv file.

Usage:

```
Cape$writeVariantInfluences(
  filename,
  p_or_q = 0.05,
  include_main_effects = TRUE
)
```

Arguments:

`filename` filename of csv file

`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, default is 0.05.

`include_main_effects` Whether to include main effects (TRUE) or only interaction effects (FALSE) in the output table, default is TRUE.

Method `set_pheno()`**:** Set phenotype

Usage:

```
Cape$set_pheno(val)
```

Arguments:

`val` phenotype value.

Method `set_geno()`**:** Set genotype

Usage:

```
Cape$set_geno(val)
```

Arguments:

`val` genotype value.

Method `create_covar_table()`**:** Create covariate table

Usage:

```
Cape$create_covar_table(value)
```

Arguments:

`value` covariate values

Method `save_rds()`**:** Save to RDS file

Usage:

```
Cape$save_rds(object, filename)
```

Arguments:

`object` data to be saved.

`filename` filename of result RDS file.

Method `read_rds()`**:** Read RDS file

Usage:

```
Cape$read_rds(filename)
```

Arguments:

`filename` RDS filename to be read.

Examples

```
## Not run:
param_file <- "cape_parameters.yml"
results_path = "."
cape_obj <- read_population("cross.csv")
combined_obj <- cape2mpp(cape_obj)
pheno_obj <- combined_obj$data_obj
geno_obj <- combined_obj$geno_obj

data_obj <- Cape$new(parameter_file = param_file,
results_path = results_path, pheno = pheno_obj$pheno, chromosome = pheno_obj$chromosome,
marker_num = pheno_obj$marker_num, marker_location = pheno_obj$marker_location,
geno_names = pheno_obj$geno_names, geno = geno_obj)

## End(Not run)
```

---

cape2mpp *Converts a* `read_population` *object to a multi-parent object*

Description

This function converts an object formatted for cape 1.0 to an object formatted for cape 2.0

Usage

```
cape2mpp(data_obj, geno_obj = NULL)
```

Arguments

`data_obj`	a data_obj formatted for cape 1.0
`geno_obj`	a genotype object. If geno_obj is NULL the genotype object is generated from data_obj$geno.

Value

This function returns a list with two objects: `list("data_obj" = data_obj, "geno_obj" = geno_obj)` These two objects must be separated again to run through cape.

Examples

```
## Not run:
new_data_obj <- cape2mpp(old_data_obj)

## End(Not run)
```

`direct_influence` *Calculate the significance of direct influences of variant pairs on phenotypes*

Description

This function rotates the variant-to-eigentrait effects back to 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", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr",
    "none"),
  perm_data = NULL,
  save_permutations = FALSE,
  n_cores = 4,
  path = ".",
  verbose = FALSE
)
```

Arguments

`data_obj`	a `Cape` object
`pairscan_obj`	a pairscan object
`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", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none", indicating whether the p value correction method used should be the Holm step-down procedure, false discovery rate or local false discovery rate respectively.

perm_data
: The permutation data generated by pairscan.

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.RDS".

n_cores
: The number of cores to use if using parallel computing

path
: The path in which to write output data

verbose
: A logical value indicating whether to write progress to standard out.

Value

This function returns data_obj with an additional list called max_var_to_pheno_influence. This list has one element for each trait. Each element is a table with eight columns: marker: the marker name conditioning_marker: the marker whose effect was conditioned on to achieve the maximum main effect of marker. coef: the direct influence coefficient. se: the standard error of the direct influence coefficient t_stat: the t statistic for the direct influence coefficient |t_stat|: the absolute value of the t statistic emp_p: the empirical p value of the direct influence coefficient p_adjusted: the adjusted p value of the direct influence coefficient.

Examples

```
## Not run:
data_obj <- direct_influence(data_obj, pairscan_obj)

## End(Not run)
```

---

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,
```

```
  run_parallel = FALSE,
  n_cores = 4,
  just_m = FALSE
)
```

Arguments

data_obj	a Cape object
pairscan_obj	a pairscan object from 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.
run_parallel	boolean, default = FALSE
n_cores	The number of cores to use if run_parallel is TRUE, default = 4
just_m	If TRUE only the m12 and m21 values are calculated. If FALSE, the default, the standard deviations are also calculated.

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 (m21) influences as well as the marker2 to marker1 (m12) influences. These results are used by calc_p to calculate empirical p values.

Examples

```
## Not run:
#run error propagateion on test statistics and
#permuted test statistics
data_obj <- error_prop(data_obj, pairscan_obj, perm = TRUE)
data_obj <- error_prop(data_obj, pairscan_obj, perm = FALSE)

## End(Not run)
```

---

get_covar *Get covariate information*

Description

This function returns information about the covariates specified for the cape run.

Usage

```
get_covar(data_obj)
```

Arguments

data_obj	a Cape object

Value

Returns a list with the following elements: covar_names: a character vector holding the names of the covariates covar_type: a character vector indicating whether each covariate derived from the phenotype matrix ("p") or the genotype matrix ("g") covar_loc: A numeric vector indicating the locations of each covariate covar_table: A matrix holding the individual values for each covariate.

Examples

```
## Not run:
covar_info <- get_covar(data_obj)

## End(Not run)
```

---

get_eigentraits *Calculate eigentraits*

Description

This function uses singular value decomposition (SVD) to calculate eigentraits from the phenotype matrix in the cape data object. It adds the eigentrait matrix to the data object along with the singular values and the right singular vectors.

Usage

```
get_eigentraits(data_obj, scale_pheno = TRUE, normalize_pheno = TRUE)
```

Arguments

data_obj	a Cape object
scale_pheno	A logical value indicating whether to mean-center and standardize the traits.
normalize_pheno	A logical value indicating whether to rankZ normalize the phenotypes.

Details

If scale_pheno is TRUE, the phenotypes are mean-centered and standardized before running the svd.

Because we use SVD in this step, there can be no missing values in the phenotype matrix. Any individuals with missing values are removed with a message.

Value

Returns the data object with the eigentraits, singular values, and right singular vectors added.

Examples

```
## Not run:
data_obj <- get_eigentraits(data_obj)

## End(Not run)
```

`get_geno` *Gets the geno object*

Description

This is an internal function returns the genotype matrix for scanning as defined by the markers and individuals specified in

Usage

```
get_geno(data_obj, geno_obj)
```

Arguments

`data_obj`	a `Cape` object
`geno_obj`	a genotype object.

Value

Returns the genotype array matching the markers and individuals specified in data_obj$geno_names

Examples

```
## Not run:
geno <- get_geno(data_obj, geno_obj)

## End(Not run)
```

`get_marker_location` *Get marker genomic position*

Description

Given a vector of marker names or numbers, this function returns the genomic coordinates for each marker, not including the chromosome number, which is retrieved using `get_marker_chr`.

Usage

```
get_marker_location(data_obj, markers)
```

Arguments

data_obj	a Cape object
markers	A vector of marker names

Value

A vector the same length as the input markers vector indicating the genomic coordinate of each marker.

Examples

```
## Not run:
marker_names <- dimnames(geno_obj)[[3]]
marker_loc <- get_marker_location(data_obj, marker_names)

## End(Not run)
```

---

get_marker_name *Get marker names*

Description

Given a vector of marker numbers, this function returns the name of each marker.

Usage

```
get_marker_name(data_obj, markers)
```

Arguments

data_obj	a Cape object
markers	A vector of marker numbers

Value

A vector the same length as the input markers vector indicating the name of each marker

Examples

```
## Not run:
marker_names <- get_marker_name(data_obj, 1:10)

## End(Not run)
```

get_network *Convert the final results to an adjacency matrix.*

Description

This function converts the significant cape interactions to an adjacency matrix, which is then used by plot_network

Usage

```
get_network(
  data_obj,
  geno_obj,
  p_or_q = 0.05,
  min_std_effect = 0,
  standardize = FALSE,
  collapse_linked_markers = TRUE,
  threshold_power = 1,
  verbose = FALSE,
  plot_linkage_blocks = FALSE
)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object
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.
min_std_effect	This numerical value offers an additional filtering method. If specified, only interactions with standardized effect sizes greater then the min_std_effect will be returned.
standardize	A logical value indicating whether the values returned in the adjacency matrix should be effect sizes (FALSE) or standardized effect sizes (TRUE). Defaults to FALSE.
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.
threshold_power	A numerical value indicating the power to which to raise the marker correlation matrix. This parameter is used in linkage_blocks_network to determine soft thresholding in determining linkage block structure. Larger values result in more splitting of linkage blocks. Smaller values result in less splitting. The default value of 1 uses the unmodified correlation matrix to determine linkage block structure.

verbose A logical value indicating whether to print algorithm progress to standard out.

plot_linkage_blocks
A logical value indicating whether to plot heatmaps showing the marker correlation structure and where the linkage block boundaries were drawn.

Value

This function returns the data object with an adjacency matrix defining the final cape network based on the above parameters. The network is put into the slot collapsed_net if collapse_linked_markers is set to TRUE, and full_net if collapse_linked_markers is set to FALSE. run_cape automatically requests both networks be generated.

Examples

```
## Not run:
data_obj <- get_network(data_obj, geno_obj)

## End(Not run)
```

---

get_pairs_for_pairscan

Select marker pairs for pairscan

Description

This function selects which marker pairs can be tested in the pair scan. Even if all markers are linearly independent, some marker pairs may have insufficient recombination between them to populate all genotype combinations. Marker pairs for which genotype combinations have insufficient numbers of individuals are not tested. This function determines which marker pairs have sufficient representation in all genotype combinations.

Usage

```
get_pairs_for_pairscan(
  gene,
  covar_names = NULL,
  max_pair_cor = NULL,
  min_per_genotype = NULL,
  run_parallel = FALSE,
  n_cores = 4,
  verbose = FALSE
)
```

Arguments

gene	A two dimensional genotype matrix with rows containing individuals and columns containing markers. Each entry is a value between 0 and 1 indicating the genotype of each individual at each marker.
covar_names	A character vector indicating which covariates should be tested.
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.
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.
run_parallel	A logical value indicating whether multiple processors should be used.
n_cores	The number of cores to be used if run_parallel is TRUE
verbose	A logical value. If TRUE, the script prints a message to the screen to indicate that it is running. If FALSE, no message is printed.

Details

One and only one of min_per_genotype or max_pair_cor should be specified. We recommend that if you have continuous genotype probabilities, you use max_pair_cor. If both values are specified, this function will preferentially use max_pair_cor.

Value

This function returns a two-column matrix of marker pairs. This matrix is then used as an argument in one_pairscan_parallel, pairscan_null_kin, pairscan_null and pairscan to specify which marker pairs should be tested.

Examples

```
## Not run:
gene <- data_obj$geno_for_pairscan
data_obj <- get_pairs_for_pairscan(gene)

## End(Not run)
```

---

get_pheno *Get the phenotype matrix*

Description

This function can return a number of different trait matrices depending on the arguments.

Usage

```
get_pheno(
  data_obj,
  scan_what = c("eigentraits", "normalized_traits", "raw_traits"),
  covar = NULL
)
```

Arguments

data_obj	a Cape object
scan_what	A character string. One of "eigentraits", "normalized.trait", or "raw_traits." If "eigentraits" the function returns the eigentraits matrix. If "normalized_traits" the function returns the trait matrix after mean-centering and normalizing. If "raw.trait" the function returns the trait matrix before mean-centering and normalization were applied.
covar	A character value indicating which, if any, covariates the traits should be adjusted for. If covariates are specified, the function fits a linear model to specify the traits with the covariates and returns the matrix of residuals (i.e. the traits after adjusting for the covariates).

Value

A matrix in which each column is a trait, and each row is an individual. The values correspond to the argument settings described above.

Examples

```
## Not run:
#get eigentrait matrix
eigenT <- get_pheno(data_obj, "eigentraits")

#get normalized trait matrix
pheno <- get_pheno(data_obj, "normalized_traits")

#get normalized traits adjusted for sex
pheno <- get_pheno(data_obj, "normalized_traits", covar = "sex")

#get raw trait matrix
pheno <- get_pheno(data_obj, "raw_traits")

## End(Not run)
```

hist_pheno *Plot trait histograms*

Description

This function plots histograms of the traits held in the pheno slot of the data object.

Usage

```
hist_pheno(data_obj, pheno_which = NULL, pheno_labels = NULL)
```

Arguments

data_obj	A Cape object
pheno_which	A vector of strings indicating which traits to plot. Defaults to all traits.
pheno_labels	A vector of strings providing alternate names for the traits in the plot if the names in the data object are not good for plotting

Examples

```
## Not run:
hist_pheno(data_obj)

## End(Not run)
```

impute_missing_geno *Impute missing genotype data using k nearest neighbors*

Description

This function uses k nearest neighbors to impute missing genotype data on a per chromosome basis. If missing genotypes remain after imputations the user can prioritize whether to remove individuals, markers, or whichever has fewer missing values.

Usage

```
impute_missing_geno(
  data_obj,
  geno_obj = NULL,
  k = 10,
  ind_missing_thresh = 0,
  marker_missing_thresh = 0,
  prioritize = c("fewer", "ind", "marker"),
  max_region_size = NULL,
```

```
  min_region_size = NULL,
  run_parallel = FALSE,
  verbose = FALSE,
  n_cores = 2
)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object
k	The number of nearest neighbors to use to impute missing data. Defaults to 10.
ind_missing_thresh	percent A percentage of acceptable missing data. After imputation if an individual is missing more data than the percent specified, it will be removed.
marker_missing_thresh	A percentage of acceptable missing data. After imputation if a marker is missing more data than the percent specified, it will be removed.
prioritize	How to prioritize removal of rows and columns with missing data. "ind" = remove individuals with missing data exceeding the threshold before considering markers to remove. "marker" = remove markers with missing data exceeding the threshold before considering individuals to remove. "fewer" = Determine how much data will be removed by prioritizing individuals or markers. Remove data in whichever order removes the least amount of data.
max_region_size	maximum number of markers to be used in calculating individual similarity. Defaults to the minimum chromosome size.
min_region_size	minimum number of markers to be used in calculating individual similarity Defaults to the maximum chromosome size.
run_parallel	A logical value indicating whether to run the process in parallel
verbose	A logical value indicating whether to print progress to the screen.
n_cores	integer number of available CPU cores to use for parallel processing

Details

This function is run by run_cape and runs automatically if a kinship correction is specified and there are missing values in the genotype object.

The prioritize parameter can be a bit confusing. If after imputation, there is one marker for which all data are missing, it makes sense to remove that one marker rather than all individuals with missing data, since all individuals would be removed. Similarly, if there is one individual with massive amounts of missing data, it makes sense to remove that individual, rather than all markers that individual is missing. We recommend always using the default "fewer" option here unless you know for certain that you want to prioritize individuals or markers for removal. There is no need to specify max_region_size or min_region_size, but advanced users may want to specify them. There is a trade-off between the time it takes to calculate a distance matrix for a large matrix and the time it takes to slide through the genome imputing markers. This function does not yet

support imputation of covariates. If individuals are genotyped very densely, the user may want to specify max_region_size to be smaller than the maximum chromosome size to speed calculation of similarity matrices.

Value

This function returns a list that includes both the data_obj and geno_obj These objects must then be separated again to continue through the cape analysis.

Examples

```
## Not run:
combined_obj <- impute_missing_geno(data_obj, geno_obj)
new_data_obj <- combined_obj$data_obj
noew_geno_obj <- combined_obj$geno_obj

## End(Not run)
```

---

kinship *Calculate the kinship matrix*

Description

This function produces a realized relationship matrix (kinship matrix) for use in adjusting for the effect of inbred relatedness. We use the R/qtl2 function calc_kinship.

Usage

```
kinship(
  data_obj,
  geno_obj,
  type = c("overall"),
  n_cores = 4,
  pop = c("MPP", "2PP", "RIL")
)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object
type	type of kinship correction. Default is overall.
n_cores	The number of cores. Defaults to 4.
pop	population type, "MPP" (multi-parental population), "2PP" (2 parents), "RIL" (recombinant inbred line)

Details

Broman KW, Gatti DM, Simecek P, Furlotte NA, Prins P, Sen Ś, Yandell BS, Churchill GA (2018) R/qtl2: software for mapping quantitative trait loci with high-dimensional data and multi-parent populations. Genetics 211:495-502 doi:10.1534/genetics.118.301595

This uses the function probs_doqtl_to_qtl2 from qtl2convert: Karl W Broman (2019). qtl2convert: Convert Data among R/qtl2, R/qtl, and DOQTL. https://kbroman.org/qtl2/, https://github.com/rqtl/qtl2convert/. And genoprob_to_alleleprob from qtl2.

Value

This function returns an n by n matrix, where n is the number of individuals in the test population. The entries of the matrix represent the level of relatedness between pairs of individuals. For more information see Kang, H. M. et al. Efficient control of population structure in model organism association mapping. Genetics 178, 1709–1723 (2008).

Examples

```
## Not run:
K <- kinship(data_obj, geno_obj)

## End(Not run)
```

---

load_input_and_run_cape

Loads input and run CAPE

Description

This function loads the input file path and runs cape It is used to run CAPE from a non R script (python)

Usage

```
load_input_and_run_cape(
  input_file = NULL,
  yaml_params = NULL,
  results_path = NULL,
  run_parallel = FALSE,
  results_file = "cross.RDS",
  p_or_q = 0.05,
  n_cores = 4,
  initialize_only = FALSE,
  verbose = TRUE,
  param_file = NULL,
  create_report = FALSE,
  qtl_id_col = NULL,
```

```
  qtl_na_strings = "-"
)
```

Arguments

input_file	data input to be loaded
yaml_params	a parameter set up in the form of a YAML string
results_path	path to the results
run_parallel	boolean, if TRUE runs certain parts of the code as parallel blocks
results_file	the name of the saved data_obj RDS file. The base name is used as the base name for all saved RDS files.
p_or_q	A threshold indicating the maximum adjusted p value considered
n_cores	integer, default is 4
initialize_only	boolean, default: FALSE
verbose	boolean, output goes to stdout
param_file	path to yml parameter file for running cape
create_report	boolean, if true we create the corresponding HTML report page
qtl_id_col	argument for read_population, an optional column number for individual IDs
qtl_na_strings	argument for read_population, an optional string for missing values

Examples

```
## Not run:
#load input in qtl2 zip format
load_input_and_run_cape("cross_file.zip")

#load input in qtl csv format
load_input_and_run_cape("cross_file.csv")

## End(Not run)
```

---

marker2covar *Creates a covariate from a genetic marker*

Description

Occasionally, researchers may want to condition marker effects on another genetic marker. For example, the HLA locus in humans has very strong effects on immune phenotypes, and can swamp smaller effects from other markers. It can be helpful to condition on markers in the HLA region to find genetic modifiers of these markers.

Usage

```
marker2covar(
  data_obj,
  geno_obj,
  singlescan_obj = NULL,
  covar_thresh = NULL,
  markers = NULL
)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object
singlescan_obj	It is possible to automatically identify markers to use as covariates based on their large main effects. If this is desired, a singlescan object is required.
covar_thresh	If designating markers as covariates based on their main effect size is desired, the covar_thresh indicates the main effect size above which a marker is designated as a covariate.
markers	Marker covariates can also be designated manually. markers takes in a vector of marker names or numbers and assigns the designated markers as covariates.

Value

This function returns the data object with additional information specifying which markers are to be used as covariates. this information can be retrieved with get_covar.

See Also

get_covar

Examples

```
## Not run:
#convert markers with effect sizes greater than 6 to covariates.
#this requires a singlescan_obj
data_obj <- marker2covar(data_obj, geno_obj, singlescan_obj, covar_thresh = 6)

#convert the first marker to a covariate
#this does not require a singlescan_obj
marker_name <- dimnames(geno_obj)[[3]][1]
data_obj <- marker2covar(data_obj, geno_obj, markers = marker_name)

## End(Not run)
```

norm_pheno *Mean-center and normalize phenotypes*

Description

This function is a wrapper for mean-centering normalizing and standardizing the trait matrix. in a data_obj.

Usage

```
norm_pheno(data_obj, mean_center = TRUE)
```

Arguments

data_obj	a Cape object mean_center a logical value indicating whether the traits should be mean centered. If FALSE, the traits are only normalized.
mean_center	mean center

Value

the data object is returned. The pheno slot of the data object will have normalized and/or mean-centered traits. The function also preserves the original trait matrix in a slot called raw_pheno.

Examples

```
## Not run:
data_obj <- norm_pheno(data_obj)

## End(Not run)
```

pairscan *This function performs the pairwise scan on all markers.*

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.

Usage

```
pairscan(
  data_obj,
  geno_obj = NULL,
  scan_what = c("eigentraits", "raw_traits"),
  pairscan_null_size = NULL,
  max_pair_cor = NULL,
  min_per_genotype = NULL,
  kin_obj = NULL,
  num_pairs_limit = 1e+06,
  num_perm_limit = 1e+07,
  overwrite_alert = TRUE,
  run_parallel = FALSE,
  n_cores = 4,
  verbose = FALSE
)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object
scan_what	A character string uniquely identifying whether eigentraits or raw traits should be scanned. Options are "eigentraits", "raw_traits"
pairscan_null_size	The total size of the null distribution. This is DIFFERENT than the number of permutations to run. Each permutation generates n choose 2 elements for the pairscan. So for example, a permutation that tests 100 pairs of markers will generate a null distribution of size 4950. This process is repeated until the total null size is reached. If the null size is set to 5000, two permutations of 100 markers would be done to get to a null distribution size of 5000.
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.
min_per_genotype	The minimum number of individuals allowable per genotype combination. If for a given marker pair, one of the genotype combinations is underrepresented, the marker pair is not tested. If this value is NULL, max_pair_cor must have a numeric value.
kin_obj	a kinship object calculated by kinship.
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 raises a warning to users not to overwrite their data object with a singlescan object. A warning necessary after a new version of cape began separating results from different functions into different results objects
run_parallel	Whether to run the analysis on parallel CPUs
n_cores	The number of CPUs to use if run_parallel is TRUE
verbose	Whether to write progress to the screen

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

This function returns an object assigned to pairscan_obj in run_cape.

The results object is a list of five elements: ref_allele: The allele used as the reference for the tests. max_pair_cor: The maximum pairwise correlation between marker pairs pairscan_results: A list with one element per trait. The element for each trait is a list of the following three elements: pairscan_effects: the effect sizes from the linear models pairscan_se: the standard errors from the linear models model_covariance: the model covariance from the linear models. pairscan_perm: The same structure as pairscan_results, but for the permuted data. pairs_tested_perm: A matrix of the marker pairs used in the permutation tests.

See Also

select_markers_for_pairscan, plot_pairscan

Examples

```
## Not run:
pairscan_obj <- pairscan(data_obj, geno_obj, pairscan_null_size = 10000)

## End(Not run)
```

---

pheno2covar *Create a covariate from a trait*

Description

This function takes a variable from the phenotype matrix for example, diet treatment or sex and converts it to a covariate.

Usage

```
pheno2covar(data_obj, pheno_which)
```

Arguments

data_obj	a Cape object
pheno_which	vector of trait names to be used as covariates

Value

Returns the data object with the specified traits removed from the phenotype matrix and transferred where they will be used as covariates. Information about assigned covariates can be retrieved with get_covar.

Examples

```
## Not run:
#convert weight to a covariate
data_obj <- pheno2covar(data_obj, "weight")

## End(Not run)
```

---

plink2cape *Convert plink2 files to cape format*

Description

Convert plink2 files to cape format

Usage

```
plink2cape(
  ped = "test.ped",
  map = "test.map",
  pheno = "test.pheno",
  out = "out.csv",
  missing_genotype = "0",
  no_fid = FALSE,
  no_parents = FALSE,
  no_sex = FALSE,
  no_pheno = FALSE,
  verbose = FALSE,
  overwrite = FALSE
)
```

Arguments

ped	full path to the ped file
map	full path to the map file
pheno	full path to the pheno file
out	full path to the output file
missing_genotype	default is "0"
no_fid	boolean, default is FALSE
no_parents	boolean, default is FALSE
no_sex	boolean, default is FALSE
no_pheno	boolean, default is FALSE
verbose	boolean, default is FALSE, gives some happy little progress messages
overwrite	boolean, default is FALSE, will only remove the existing file if this is set to TRUE

Details

For further information about PLINK and its file formats, see https://zzz.bwh.harvard.edu/plink/

Value

A list with two elements: data_obj and geno_obj These objects are formatted for use in cape and must then be separated to use in run_cape.

References

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MAR, Bender D, Maller J, Sklar P, de Bakker PIW, Daly MJ & Sham PC (2007) PLINK: a toolset for whole-genome association and population-based linkage analysis. American Journal of Human Genetics, 81.

Examples

```
## Not run:
#convert files with default names to a data_obj
data_obj <- plink2cape()

## End(Not run)
```

plot_effects *Plot Interaction Effects*

Description

This function plots phenotypic effects of individual cape interactions. It serves as a wrapper for the functions plot_lines plot_bars plot_points, and plot_int_heat. Each of those functions plots individual cape interactions in different forms.

Usage

```
plot_effects(
  data_obj,
  geno_obj,
  marker1,
  marker2 = NULL,
  pheno_type = "normalized",
  plot_type = c("l", "p", "b", "h"),
  error_bars = "none",
  ymin = NULL,
  ymax = NULL,
  covar = NULL,
  marker1_label = NULL,
  marker2_label = NULL,
  bin_continuous_genotypes = TRUE,
  ref_centered = TRUE,
  gen_model1 = "Additive",
  gen_model2 = "Additive",
  bins_marker1 = 50,
  bins_marker2 = 50
)
```

Arguments

data_obj	A Cape object
geno_obj	A genotype object
marker1	A string indicating the name of the source marker in the interaction. This can also be the name of a covariate.
marker2	Another string indicating the name of the source marker in the interaction. This can also be the name of a covariate. Optional.
pheno_type	One of "eigentraits", "normalized_traits", or "raw_traits", indicating which traits to plot.
plot_type	A letter referring to the desired style of the plot. The choices are the following: "l" - line plots, "p" = points, "b" - bar plots, "h" - heat map.

`error_bars` The type of error bars to plot. Choices are "none" (the default), "se" for standard error, or "sd" for standard deviation.

`ymin` A minimum value for the y axes across all plots. If NULL, each y axis will be determined independently

`ymax` A maximum value for the y axes across all plots. If NULL, each y axis will be dertermined independently

`covar` A vector of strings indicating which covariates, if any, the traits should be adjusted for. If NULL, the covariates specified in the data_obj are used as default. To prevent adjusting for covariates, use "none".

`marker1_label` A string to use as the label for marker1 If NULL, the string used for marker1 will be used.

`marker2_label` A string to use as the label for marker2 If NULL, the string used for marker2 will be used.

`bin_continuous_genotypes` If TRUE, genotypes (and covariate) values will be binned into 0, 0.5, and 1 values. This reduces the number of bins that traits need to be divided into, especially if there are only one or two individuals with a 0.49 genotype, for example. Binning may not be desirable when using the heatmap.

`ref_centered` A logical value indicating whether to center the values on the reference allele. Defaults to TRUE.

`gen_model1` One of "Additive", "Dominant", or "Recessive" indicating how the genotype should be coded for the first marker. If Additive, genotypes are coded as 0 for homozygous reference allele, 1 for homozygous alternate allele, and 0.5 for heterozygous. If Dominant, any allele probability greater than 0.5 is set to 1. If recessive, any allele probability less than or equal to 0.5 is set to 0. In other words, for the dominant coding, heterozygotes are grouped with the homozygous alternate genotypes: 0 vs. (0.5,1). This shows the effect of having any dose of the alternate allele. With a recessive coding, heterozygotes are grouped with the homozygous reference genotypes: (0, 0.5) vs. 1. This shows the effect of having two copies of the alternate allele vs. having fewer than two copies.

`gen_model2` The same as gen_model1, but for the second marker.

`bins_marker1` Only used for heatmap plotting. The number of bins for marker1 if it is a continuously valued marker or covariate. The bins are used to fit a linear model and predict outcomes for a 2D grid of marker1 and marker2 values. This argument can also be a vector of bin values for binning at specific values.

`bins_marker2` The same as bins_marker1, but for marker2.

Details

The "h" option calls `plot_int_heat`, which fits linear models to each trait and both markers specified. It uses those models to predict phenotype values along continuously valued genotype bins and plots the predicted values as a heatmap.

Value

None

Examples

```
## Not run:
marker1 <- dimnames(geno_obj)[[3]][1]
marker2 <- dimnames(geno_obj)[[3]][2]
plot_effects(data_obj, geno_obj, plot_type = "l", error_bars = "se")

## End(Not run)
```

`plot_full_network` *Plot the final epistatic network in a traditional network view.*

Description

This function plots the final results in a layout different to both `plot_variant_influences` and `plot_network`. In this view, the network is plotted with a traditional network layout. The genomic position information in `plot_network` is lost, but in this view it is easier to see the structure of the overall network in terms of hubs and peripheral nodes. In this view, each node is plotted as a pie-chart, and the main effects of the node are indicated as positive, negative, or not-significant (gray). Significant interactions are shown arrows between nodes and colored based on whether they are positive or negative interactions. Colors for positive and negative main effects and interactions are specified in the arguments. The function `get_network` must be run before plotting the network.

Usage

```
plot_full_network(
  data_obj,
  p_or_q = 0.05,
  collapsed_net = TRUE,
  main = NULL,
  color_scheme = c("DO/CC", "other"),
  pos_col = "brown",
  neg_col = "blue",
  bg_col = "gray",
  light_dark = "f",
  node_border_lwd = 1,
  layout_matrix = NULL,
  zoom = 1,
  xshift = 0,
  yshift = 0,
  node_radius = 1,
  label_nodes = TRUE,
  label_offset = 0,
  label_cex = 1,
  legend_radius = 1,
  legend_cex = 1,
  legend_position = "topleft",
```

```
  arrow_offset = node_radius,
  arrow_length = 0.2,
  edge_lwd = 2
)
```

Arguments

data_obj	A Cape object
p_or_q	The maximum p value (or q value if FDR was used) for significant main effects and interactions.
collapsed_net	A logical value indicating whether to show the network condensed into linkage blocks (TRUE) or each individual marker (FALSE)
main	A title for the plot
color_scheme	either "DO/CC" or "other". "DO/CC" uses the official "DO/CC" colors for the DO/CC alleles http://www.csbio.unc.edu/CCstatus/index.py?run=AvailableLines.information "other" uses an unrelated color palette for multiple alleles.
pos_col	The color to use for positive main effects and interactions must be one of "green", "purple", "red", "orange", "blue", "brown", "yellow", "gray" see get_color
neg_col	The color to use for negative main effects and interactions takes the same values as pos_col.
bg_col	The color to be used in pie charts for non-significant main effects. Takes the same values as pos_col
light_dark	Indicates whether pos_col, neg_col, and bg_col should be selected from light colors ("l"), dark colors ("d") or the full spectrum from light to dark ("f")
node_border_lwd	The thickness of the lines around the pie charts
layout_matrix	Users have the option of providing their own layout matrix for the network. This should be a two column matrix indicating the x and y coordinates of each node in the network.
zoom	Allows the user to zoom in and out on the image if the network is either running off the edges of the plot or too small in the middle of the plot.
xshift	A constant by which to shift the x values of all nodes in the network.
yshift	A constant by which to shift the y values of all nodes in the network.
node_radius	The size of the pie chart for each node.
label_nodes	A logical value indicating whether the nodes should be labeled. Users may want to remove labels for large networks.
label_offset	The amount by which to offset the node labels from the center of the nodes.
label_cex	The size of the node labels
legend_radius	The size of the legend indicating which pie piece corresponds to which traits.
legend_cex	The size of the labels in the legend.
legend_position	The position of the legend on the plot
arrow_offset	The distance from the center of the node to the arrow head.
arrow_length	The length of the head of the arrow
edge_lwd	The thickness of the arrows showing the interactions.

Details

For most networks, the default options will be fine, but there is a lot of room for modification if changes are desired

Value

This function invisibly returns a list of length two. The first element contains the igraph network object. The second contains the layout matrix for the network. This can be passed in as an argument ("layout_matrix") which provides more control to the user in the layout. Other network layouts from igraph can also be passed in here.

References

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

Examples

```
## Not run:
plot_full_network(data_obj)

## End(Not run)
```

---

plot_network *Plots cape results as a circular network*

Description

This script plots cape results in a circular network. The chromosomes are arranged in a circle. Main effects are shown in concentric circles around the chromosomes, with each trait in its own circle. Main effects can either be colored as negative or positive, or with parental allele colors for multi-parent populations.

Usage

```
plot_network(
  data_obj,
  marker_pairs = NULL,
  collapsed_net = TRUE,
  trait = NULL,
  trait_labels = NULL,
  color_scheme = c("DO/CC", "other"),
  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",
  show_alleles = TRUE
)
```

Arguments

data_obj	a Cape object
marker_pairs	a two-column matrix identifying which marker pairs should be plotted. This is particularly useful if the network is very dense. The default value, NULL, plots all marker pairs.
collapsed_net	A logical value indicating whether to plot all individual SNPs or linkage blocks calculated by linkage_blocks_network.
trait	A character vector indicating which traits to plot. The default NULL value plots all traits.
trait_labels	A character vector indicating the names of the traits in case the names from the data object are not great for plotting.
color_scheme	A character value of either "DO/CC" or other indicating the color scheme of main effects. If "DO/CC" allele effects can be plotted with the DO/CC colors.
main_lwd	A numeric value indicating the line width for the main effect lines
inter_lwd	A numeric value indicating the line width for the interaction lines
label_cex	A numeric value indicating the size of the labels
percent_bend	A numeric value indicating the amount that the arrows for the interaction effects should be bent. A value of 0 will plot straight lines.
chr_gap	A numeric value indicating the size of the gap plotted between chromosomes.
label_gap	A numeric value indicating the size of the gap the chromosomes and their labels.
positive_col	One of c("green", "purple", "red", "orange", "blue", "brown", "yellow", "gray") indicating the color for positive interactions.
negative_col	One of c("green", "purple", "red", "orange", "blue", "brown", "yellow", "gray") indicating the color for negative interactions. show_alleles A logical value indicating whether to color main effects by their allele values (TRUE) or by whether they are positive or negative (FALSE)
show_alleles	boolean, default is TRUE

Details

Interaction effects are shown as arrows linking chromosomal positions. They are colored based on whether they are positive or negative.

Examples

```
## Not run:
plot_network(data_obj)

## End(Not run)
```

plot_pairscan *Plot the result of the pairwise scan*

Description

This function plots the results of the pairwise scan. It plots a matrix of the the interactions between all pairs of markers.

Usage

```
plot_pairscan(
  data_obj,
  pairscan_obj,
  phenotype = NULL,
  standardized = FALSE,
  show_marker_labels = FALSE,
  show_chr = TRUE,
  label_chr = TRUE,
  show_alleles = TRUE,
  allele_labels = NULL,
  pos_col = "brown",
  neg_col = "blue",
  color_scheme = c("DO/CC", "other"),
  pdf_label = "Pairscan.Regression.pdf"
)
```

Arguments

data_obj	a Cape object
pairscan_obj	a pairscan object from pairscan
phenotype	The names of the phenotypes to be plotted. If NULL, all phenotypes are plotted.
standardized	If TRUE, the standardized effects are plotted. IF FALSE, the effect sizes are plotted.
show_marker_labels	If TRUE marker labels are plotted along the axes. If FALSE, they are omitted.
show_chr	If TRUE, the chromosome boundaries are shown
label_chr	If TRUE, the chromosomes are labeled
show_alleles	If TRUE, the allele of each marker is indicated by color.
allele_labels	Labels for the alleles if other than those stored in the data object.
pos_col	The color to use for positive main effects and interactions must be one of "green", "purple", "red", "orange", "blue", "brown", "yellow", "gray" see get_color
neg_col	The color to use for negative main effects and interactions takes the same values as pos_col.

color_scheme	either "DO/CC" or "other". "DO/CC" uses the official "DO/CC" colors for the DO/CC alleles http://www.csbio.unc.edu/CCstatus/index.py?run=AvailableLines.information "other" uses an unrelated color palette for multiple alleles.
pdf_label	Label for the resulting file. Defaults to "Pairscan.Regression.pdf" if plotting to pdf, "Pairscan.Regression.jpg" otherwise.

Value

Plots to a pdf

Examples

```
## Not run:
plot_pairscan(data_obj, pairscan_obj)

## End(Not run)
```

---

plot_pheno_cor *Plot trait pairs against each other*

Description

This function plots pairs of traits against each other to visualize the correlations between traits.

Usage

```
plot_pheno_cor(
  data_obj,
  pheno_which = NULL,
  color_by = NULL,
  group_labels = NULL,
  text_cex = 1,
  pheno_labels = NULL,
  pt_cex = 1
)
```

Arguments

data_obj	a Cape object
pheno_which	A vector of trait names to plot. The default is to plot all traits.
color_by	A character string indicating a value to color the traits by, for example sex or treatment. It must be one of the covariates. See pheno2covar.
group_labels	A vector of names for the legend indicating the groups for the colored dots.
text_cex	A numeric value indicating the size of the text
pheno_labels	A vector of names for traits to appear in the plot in case the column names are not very pretty.
pt_cex	A numeric value indicating the size of the points.

Examples

```
## Not run:
plot_pheno_cor(data_obj, color_by = "sex", group_labels = c("Female", "Male"))

## End(Not run)
```

`plot_singlescan` *Plot results of single-locus scans*

Description

This function plots the results of `singlescan`

Usage

```
plot_singlescan(
  data_obj,
  singlescan_obj,
  chr = NULL,
  traits = NULL,
  alpha = c(0.01, 0.05),
  standardized = TRUE,
  color_scheme = c("DO/CC", "other"),
  allele_labels = NULL,
  include_covars = TRUE,
  show_selected = FALSE,
  line_type = "l",
  lwd = 1,
  pch = 16,
  cex = 1,
  covar_label_size = 0.7
)
```

Arguments

`data_obj`	a `Cape` object
`singlescan_obj`	a singlescan object from `singlescan`
`chr`	a vector of chromosome names to include in the plot. Defaults to all chromosomes.
`traits`	a vector of trait names to plot. Defaults to all traits.
`alpha`	the alpha significance level. Lines for significance values will only be plotted if n_perm > 0 when `singlescan` was run. And only alpha values specified in `singlescan` can be plotted.
`standardized`	If TRUE t statistics are plotted. If FALSE, effect sizes are plotted.

color_scheme	A character value of either "DO/CC" or other indicating the color scheme of main effects. If "DO/CC" allele effects can be plotted with the DO/CC colors.
allele_labels	A vector of labels for the alleles if different than those stored in the data_object.
include_covars	Whether to include covariates in the plot.
show_selected	If TRUE will indicate which markers were selected for the pairscan. In order for these to be plotted, select_markers_for_pairscan must be run first.
line_type	as defined in plot
lwd	line width, default is 1
pch	see the "points()" R function. Default is 16 (a point).
cex	see the "points()" R function. Default is 1.
covar_label_size	default is 0.7

Examples

```
## Not run:
plot_singlescan(data_obj, singlescan_obj)

## End(Not run)
```

---

plot_svd *Plots eigentraits*

Description

This function plots the results of the singular value decomposition (SVD) on the phenotypes. Gray bars indicate the amount of phenotypic variance accounted for by each eigentrait.

Usage

```
plot_svd(
  data_obj,
  orientation = c("vertical", "horizontal"),
  neg_col = "blue",
  pos_col = "brown",
  light_dark = "f",
  pheno_labels = NULL,
  cex_barplot_axis = 1.7,
  cex_barplot_labels = 2,
  cex_barplot_title = 1.7,
  main = "Eigentrait Contributions to Phenotypes",
  cex_main = 2,
  main_x = 0.5,
  main_y = 0.5,
```

```
  cex_ET = 1.7,
  ET_label_x = 0.5,
  ET_label_y = 0.5,
  pheno_label_pos = 0.5,
  cex_pheno = 1.7,
  pheno_srt = 90,
  percent_total_variance_x = 0.5,
  percent_total_variance_y = 0.5,
  cex_color_scale = 1,
  cex_var_accounted = 2,
  var_accounted_x = 0,
  var_accounted_y = 0,
  show_var_accounted = FALSE,
  just_selected_et = FALSE
)
```

Arguments

data_obj	a Cape object
orientation	string, ("vertical", "horizontal")
neg_col	The color to use for negative main effects and interactions takes the same values as pos_col.
pos_col	The color to use for positive main effects and interactions must be one of "green", "purple", "red", "orange", "blue", "brown", "yellow", "gray" see get_color
light_dark	Indicates whether pos_col, neg_col, and bg_col should be selected from light colors ("l"), dark colors ("d") or the full spectrum from light to dark ("f")
pheno_labels	Vector of phenotype names if other than what is stored in the data object
cex_barplot_axis	Size of axis for the bar plot
cex_barplot_labels	Size of labels for the bar plot
cex_barplot_title	Size of the barplot title
main	Title for the plot. Defaults to "Eigentrait Contributions to Phenotypes"
cex_main	Size of the overall title
main_x	x shift for the overall title
main_y	y shift for the overall title
cex_ET	Size of the eigentrait labels
ET_label_x	x shift for the eigentrait labels
ET_label_y	y shift for the eigentrait labels
pheno_label_pos	x shift for the trait labels
cex_pheno	size of the trait labels
pheno_srt	Rotation factor for the trait labels

percent_total_variance_x
: x shift for the percent total variance labels

percent_total_variance_y
: y shift for the percent total variance labels

cex_color_scale
: label size for the color scal

cex_var_accounted
: size for the variance accounted for labels

var_accounted_x
: x shift for the variance accounted axis label

var_accounted_y
: x shift for the variance accounted axis label

show_var_accounted
: logical

just_selected_et
: logical

Details

Below the bars is a heatmap indicating how each trait contributes to each eigentrait. Colors can be adjusted to suit preferences.

Value

`list("data_obj" = data_obj, "geno_obj" = geno_obj)`

Examples

```
## Not run:
#plot all eigentraits
plot_svd(data_obj)

#plot only eigentraits being run in cape
plot_svd(data_obj, just_selected_et = TRUE)

## End(Not run)
```

---

plot_variant_influences

Plot cape coefficients

Description

This function plots the the cape coefficients between pairs of markers as a heat map. The interactions are shown in the main part of the heatmap while the main effects are shown on the right hand side. Directed interactions are read from the y axis to the x axis. For example an interaction from marker1 to marker2 will be shown in the row corresponding to marker1 and the column corresponding to marker2. Similarly, if marker1 has a main effect on any traits, these will be shown in the row for marker1 and the trait columns.

Usage

```
plot_variant_influences(
  data_obj,
  p_or_q = 0.05,
  min_std_effect = 0,
  plot_all_vals = FALSE,
  standardize = FALSE,
  color_scheme = c("DO/CC", "other"),
  pos_col = "brown",
  neg_col = "blue",
  not_tested_col = "lightgray",
  show_marker_labels = FALSE,
  show_chr = TRUE,
  label_chr = TRUE,
  show_alleles = TRUE,
  scale_effects = c("log10", "sqrt", "none"),
  pheno_width = NULL,
  covar_width = NULL,
  covar_labels = NULL,
  phenotype_labels = NULL,
  show_not_tested = TRUE
)
```

Arguments

data_obj	a Cape object
p_or_q	A threshold indicating the maximum p value (or q value if FDR was used) of significant interactions and main effects
min_std_effect	An optional filter. The plot will exclude all pairs with standardized effects below the number set here.
plot_all_vals	If TRUE will plot all values regardless of significant
standardize	Whether to plot effect sizes (FALSE) or standardized effect sizes (TRUE)
color_scheme	A character value of either "DO/CC" or other indicating the color scheme of main effects. If "DO/CC" allele effects can be plotted with the DO/CC colors.
pos_col	The color to use for positive main effects and interactions must be one of "green", "purple", "red", "orange", "blue", "brown", "yellow", "gray" see get_color
neg_col	The color to use for negative main effects and interactions takes the same values as pos_col.
not_tested_col	The color to use for marker pairs not tested. Takes the same values as pos_col and neg_col
show_marker_labels	Whether to write the marker labels on the plot
show_chr	Whether to show chromosome boundaries
label_chr	Whether to label chromosomes if plotted
show_alleles	If TRUE, the allele of each marker is indicated by color.

scale_effects	One of "log10", "sqrt", "none." If some effects are very large, scaling them can help show contrasts between smaller values. The default is no scaling.
pheno_width	Each marker and trait gets one column in the matrix. If there are many markers, this makes the effects on the traits difficult to see. pheno_width increases the number of columns given to the phenotypes. For example, if pheno_width = 11, the phenotypes will be shown 11 times wider than individual markers.
covar_width	See pheno_width. This is the same effect for covariates.
covar_labels	Labels for covariates if different from those stored in the data object.
phenotype_labels	Labels for traits if different from those stored in the data object
show_not_tested	Whether to color the marker pairs that were not tested. If FALSE, they will not be colored in.

Value

This function invisibly returns the variant influences matrix. shown in the heat map.

Examples

```
## Not run:
plot_variant_influences(data_obj)

## End(Not run)
```

---

qnorm_pheno *Plot trait distributions*

Description

This function plots the quantiles of each trait against quantiles of a theoretical normal distribution. This provides a way to check whether traits are normally distributed

Usage

```
qnorm_pheno(data_obj, pheno_which = NULL, pheno_labels = NULL)
```

Arguments

data_obj	a Cape object
pheno_which	A vector of trait names to plot. The default is to plot all traits.
pheno_labels	A vector of names for traits to appear in the plot in case the column names are not very pretty.

Examples

```
## Not run:
qnorm_pheno(data_obj)

## End(Not run)
```

`qtl2_to_cape` *Convert qtl2 object to cape format*

Description

This function converts a data object constructed by qtl2 using the read_cross() function to cape format. It returns a list in which the first element is the cape data object, and the second element is the cape genotype object.

Usage

```
qtl2_to_cape(cross, genoprobs = NULL, map = NULL, covar = NULL, verbose = TRUE)
```

Arguments

`cross`	a cross object created by the R/qtl2 function read_cross()
`genoprobs`	an optional argument for providing previously calculated genoprobs. if this parameter is missing, genoprobs are calculated by qtl_to_cape.
`map`	The qtl2 map. This can be omitted if the map is included in the cross object as either pmap or gmap. By default the physical map (pmap) is used. If it is missing, the genetic map is used. A provided map will be used preferentially over a map included in the cross object.
`covar`	Optional matrix of any covariates to be included in the analysis.
`verbose`	A logical value indicating whether to print progress to the screen. Defaults to TRUE.

Value

This function returns a list of two elements. The first element is a cape data object. The second element is a cape genotype object.

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

Broman, Karl W., Daniel M. Gatti, Petr Simecek, Nicholas A. Furlotte, Pjotr Prins, Śaunak Sen, Brian S. Yandell, and Gary A. Churchill. "R/qtl2: software for mapping quantitative trait loci with high-dimensional data and multiparent populations." Genetics 211, no. 2 (2019): 495-502.

Examples

```
## Not run:
data_obj <- qtl2_to_cape(cross_obj, genoprobs, map, covar, verbose = TRUE)

## End(Not run)
```

`read_parameters` *Read the parameter file, add missing entries*

Description

This function returns reads in the YAML file and checks for any parameters that might not be included. This may not matter for the given run, but it's handy to be able to check for any and all potential variables.

Usage

```
read_parameters(filename = "cape.parameters.yml", yaml_parameters = NULL)
```

Arguments

`filename`	full path to the .yml file holding CAPE parameters (is not needed if yaml_parameters is provided)
`yaml_parameters`	yaml string holding CAPE parameters (can be NULL)

Value

Returns a named list with all possible options

Examples

```
## Not run:
param_table <- read_parameters()

## End(Not run)
```

`read_population` *Reads in data in the R/qtl csv format*

Description

This function reads in a data file in the r/qtl format It converts letter genotypes to numbers if required. It parses the data into a data object. if filename is left empty, the script will ask the use to choose a file. phenotypes can be specified with a vector of column numbers or character strings. For each phenotype specified with a name, the script will find its location.

Usage

```
read_population(
  filename = NULL,
  pheno_col = NULL,
  geno_col = NULL,
  id_col = NULL,
  delim = ",",
  na_strings = "-",
  check_chr_order = TRUE,
  verbose = TRUE
)
```

Arguments

`filename`	The name of the file to read in
`pheno_col`	Column numbers of desired traits. The default behavior is to read in all traits.
`geno_col`	Column numbers of desired markers. The default behavior is to read in all markers.
`id_col`	The column number of an ID column. This is helpful to specify if the individual IDs are strings. Strings are only allowed in the ID column. All other trait data must be numeric.
`delim`	column delimiter for the file, default is ","
`na_strings`	a character string indicating how NA values are specified, default is "-"
`check_chr_order`	boolean, default is TRUE
`verbose`	A logical value indicating whether to print progress and cross information to the screen. Defaults to TRUE.

Value

This function returns a cape object in a former cape format. It must be updated using `cape2mpp`

References

Broman et al. (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890 doi:10.1093/bioinformatics/btg112

Examples

```
## Not run:
cape_obj <- read_population("cross.csv")
combined_obj <- cape2mpp(cape_obj)
data_obj <- combined_obj$data_obj
geno_obj <- combined_obj$geno_obj

## End(Not run)
```

`remove_ind` *Remove individuals*

Description

Remove individuals

Usage

```
remove_ind(data_obj, ind_to_remove = NULL, names_to_remove = NULL)
```

Arguments

`data_obj`	a Cape object
`ind_to_remove`	Indices of individuals to remove
`names_to_remove`	Names of individuals to remove Only one of ind_to_remove or names_to_remove should be specified.

Value

an updated cape data object with specified individuals removed.

Examples

```
## Not run:
#remove males
covar_info <- get_covar(data_obj)
male_idx <- which(covar_info$covar_table[,"sex"] == 1)
data_obj <- remove_ind(data_obj, ind_to_remove = male_idx)

## End(Not run)
```

remove_kin_ind *Removes individuals from the kinship object to match the cape.obj*

Description

Removes individuals from the kinship object to match the cape.obj

Usage

```
remove_kin_ind(data_obj, kin_obj)
```

Arguments

data_obj	a Cape object
kin_obj	a kinship object

Value

updated kinship object

Examples

```
## Not run:
kin_obj <- remove_kin_id(data_obj, kin_obj)

## End(Not run)
```

remove_markers *Removes genetic markers*

Description

Removes genetic markers

Usage

```
remove_markers(data_obj, markers_to_remove)
```

Arguments

data_obj	a Cape object
markers_to_remove	A vector of marker names to be removed.

Examples

```
## Not run:
#remove markers on chromosome 1
marker_idx <- which(data_obj$chromosome == 1)
data_obj <- remove_markers(data_obj, marker_idx)

## End(Not run)
```

`remove_missing_genotype_data` *Removes individuals and/or markers with missing data*

Description

Because there an be no missing data when calculating the kinship correction, we need a way to remove either individuals or markers with missing data. We also need a way to calculate which of these options will remove the least amount of data.

Usage

```
remove_missing_genotype_data(
  data_obj,
  geno_obj = NULL,
  ind_missing_thresh = 0,
  marker_missing_thresh = 0,
  prioritize = c("fewer", "ind", "marker")
)
```

Arguments

`data_obj`	a Cape object
`geno_obj`	a genotype object
`ind_missing_thresh`	Allowable amount of missing information for an individual. If 10 default, all individuals with any missing data at all will be removed.
`marker_missing_thresh`	Allowable amount of missing information for a marker. If 10 default, all markers with any missing data at all will be removed.
`prioritize`	the basis prioritization is one of "fewer" = calculate whether removing individuals or markers will remove fewer data points, and start with that. "ind" = remove individuals with missing data before considering markers with missing data. "marker" = remove markers with missing data before considering individuals.

Details

For example, if there is one marker with no data at all, we would rather remove that one marker, than all individuals with missing data. Alternatively, if there is one individual with very sparse genotyping, we would prefer to remove that single individual, rather than all markers with missing data.

This function provides a way to calculate whether individuals or markers should be prioritized when removing data. It then removes those individuals or markers.

Value

The cape object is returned with individuals and markers removed. After this step, the function get_geno should return an array with no missing data if ind_missing_thresh and marker_missing_thresh are both 0. If these numbers are higher, no individual or marker will be missing more than the set percentage of data.

details All missing genotype data must either be imputed or removed if using the kinship correction. Running impute_missing_geno prior to running remove_missing_genotype_data ensures that the least possible amount of data are removed before running cape. In some cases, there will be missing genotype data even after running impute_missing_geno, in which case, remove_missing_genotype_data still needs to be run. The function run_cape automatically runs these steps when use_kinship is set to TRUE.

See Also

get_geno, impute_missing_geno, run_cape

Examples

```
## Not run:
#remove entries with more than 10\
#removal of markers
data_obj <- remove_missing_genotype_data(data_obj, geno_obj,
marker_missing_thresh = 10, ind_missing_thresh = 10,
prioritize = "marker")

#remove markers with more than 5\
#more than 50\
#missing data, prioritizing removal of individuals.
data_obj <- remove_missing_genotype_data(data_obj, geno_obj,
ind_missing_thresh = 10, marker_missing_thresh = 50,
prioritize = "ind")

#remove entries witn any missing data prioritizing whichever
#method removes the least amount of data
data_obj <- remove_missing_genotype_data(data_obj, geno_obj)


## End(Not run)
```

remove_unused_markers *Take out markers not used in cape*

Description

This function removes any markers that are not used in cape. This includes markers on the sex chromosomes, mitochondrial markers, and any invariant markers.

Usage

```
remove_unused_markers(data_obj, geno_obj, verbose = FALSE)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object
verbose	A logical value indicating whether to print progress to the screen. Default is FALSE

Value

an updated Cape object (data_obj)

Examples

```
## Not run:
data_obj <- remove_unused_markers(data_obj, geno_obj)

## End(Not run)
```

run_cape *Runs CAPE*

Description

This function takes in a data object and genotype object that have been formatted for cape, as well as a string identifying a parameter file. It runs cape on the data using the parameters specified in the file.

Usage

```
run_cape(
  pheno_obj,
  geno_obj,
  results_file = "cross.RDS",
  p_or_q = 0.05,
  n_cores = 4,
  initialize_only = FALSE,
  verbose = TRUE,
  run_parallel = FALSE,
  param_file = NULL,
  yaml_params = NULL,
  results_path = NULL,
  plot_pdf = TRUE
)
```

Arguments

pheno_obj	the cape object holding the phenotype data returned by
geno_obj	the genotype object
results_file	the name of the saved data_obj RDS file. The base name is used as the base name for all saved RDS files.
p_or_q	A threshold indicating the maximum adjusted p value considered significant. Or, if FDR p value correction is used, the the maximum q value considered significant.
n_cores	The number of CPUs to use if run_parallel is set to TRUE
initialize_only,	If TRUE, cape will not be run. Instead an initialized data object will be returned. This data object will contain normalized and mean-centered trait values, and eigentraits, and will have covariates specified. However, the singlescan, pairscan, etc. will not be run.
verbose	Whether progress should be printed to the screen
run_parallel	boolean, if TRUE runs certain parts of the code as parallel blocks
param_file	yaml full path to the parameter file
yaml_params	yaml string containing the parameters. Either the param_file or yaml_params can be null.
results_path	path that results should be written to.
plot_pdf	boolean, TRUE by default. If FALSE no pdf will be generated by the analysis.

Details

This function assumes you already have all required libraries and functions loaded.

Value

This function invisibly returns the data object with all final data included. In addition, data saved to the data_obj$results_path directory

Examples

```
## Not run:
final_data_obj <- run_cape(pheno_obj, geno_obj)

## End(Not run)
```

`select_eigentraits` *Assign selected eigentraits in the Cape object*

Description

This function is used to identify which eigentraits will be analyzed in the Cape run. After eigentrait decomposition of n traits, there will be n eigentraits. If there are more than two eigentraits, the user may wish to analyze a subset of them. This function specifies which of the eigentraits will be analyzed by Cape. It does this by subsetting the ET matrix to only those eigentraits specified. The traits not selected are deleted from the object.

Usage

```
select_eigentraits(data_obj, traits_which = c(1, 2))
```

Arguments

`data_obj`	a Cape object
`traits_which`	A vector of integers, of at least length two specifying which eigentraits should be analyzed.

Value

updated Cape object

See Also

`plot_svd`

Examples

```
## Not run:
data_obj <- selecct_eigentraits(data_obj, traits_which = 1:3)

## End(Not run)
```

select_markers_for_pairscan *Select markers for the pairwise scan.*

Description

This function selects markers for the pairwise scan. Beause Cape is computationally intensive, pairscans should not be run on large numbers of markers. As a rule of thumb, 1500 markers in a population of 500 individuals takes about 24 hours to run without the kinship correction. The kinship correction increases the time of the analysis, and users may wish to reduce the number of markers scanned even further to accommodate the extra computational burden of the kinship correction.

Usage

```
select_markers_for_pairscan(
  data_obj,
  singlescan_obj,
  geno_obj,
  specific_markers = NULL,
  num_alleles = 50,
  peak_density = 0.5,
  window_size = NULL,
  tolerance = 5,
  plot_peaks = FALSE,
  verbose = FALSE,
  pdf_filename = "Peak.Plots.pdf"
)
```

Arguments

data_obj	a Cape object
singlescan_obj	a singlescan object from singlescan.
geno_obj	a genotype object
specific_markers	A vector of marker names specifying which markers should be selected. If NULL, the function uses main effect size to select markers.
num_alleles	The target number of markers to select if using main effect size
peak_density	The fraction of markers to select under each peak exceeding the current threshold. Should be set higher for populations with low LD. And should be set lower for populations with high LD. Defaults to 0.5, corresponding to 50% of markers selected under each peak.
window_size	The number of markers to use in a smoothing window when calculating main effect peaks. If NULL, the window size is selected automatically based on the number of markers with consecutive rises and falls of main effect size.

tolerance	The allowable deviation from the target marker number in number of markers. For example, If you ask the function to select 100 markers, an set the tolerance to 5, the algorithm will stop when it has selected between 95 and 105 markers.
plot_peaks	Whether to plot the singlescan peaks identified by bin_curve. This can be helpful in determining whether the window_size and peak_density parameters are optimal for the population.
verbose	Whether progress should be printed to the screen
pdf_filename	If plot_peaks is TRUE, this argument specifies the filename to which the peaks are plotted.

Details

This function can select markers either from a pre-defined list input as the argument specific_markers, or can select markers based on their main effect size.

To select markers based on main effect size, this function first identifies effect score peaks using an automated peak detection algorithm. It finds the peaks rising above a starting threshold and samples markers within each peak based on the user-defined sampling density peak_density. Setting peak_density to 0.5 will result in 50% of the markers in a given peak being sampled uniformly at random. Sampling reduces the redundancy among linked markers tested in the pairscan. If LD is relatively low in the population, this density can be increased to 1 to include all markers under a peak. If LD is high, the density can be decreased to reduce redundancy further.

The algorithm compares the number of markers sampled to the target defined by the user in the argument num_alleles. If fewer than the target have been selected, the threshold is lowered, and the process is repeated until the target number of alleles have been selected (plus or minus the number set in tolerance).

If the number of target alleles exceeds the number of markers genotyped, all alleles will be selected automatically.

Value

Returns the Cape object with a new matrix called geno_for_pairscan containing the genotypes of the selected markers for each individual.

See Also

bin_curve, singlescan

Examples

```
## Not run:
#select 100 alleles to run through pairscan
data_obj <- select_markers_for_pairscan(data_obj, singlescan_obj, geno_obj, num_alleles = 100)

## End(Not run)
```

`select_pheno` *This function selects the phenotypes in a Cape object*

Description

Updates the pheno object to include only 'pheno_which' columns. Optionally scale and/or normalize traits.

Usage

```
select_pheno(
  data_obj,
  pheno_which,
  min_entries = 5,
  scale_pheno = FALSE,
  rank_norm_pheno = FALSE
)
```

Arguments

`data_obj`	a Cape object
`pheno_which`	vector of names from the parameters YAML file. This vector should include both traits and covariates. The covariates are assigned after trait selection.
`min_entries`	minimum number of data entries the phenotype needs to have for it to be included. If any trait has fewer than min_entries, It will be removed with a warning.
`scale_pheno`	if TRUE then phenotypes are mean-centered and standardized
`rank_norm_pheno`	if TRUE then phenotypes are rank Z normalized

Value

updated Cape object

Examples

```
## Not run:
data_obj <- select_pheno(data_obj, pheno_which = c("BW_24", "INS_24", "log_GLU_24"))

## End(Not run)
```

singlescan *Runs marker regression on each individual genetic marker*

Description

This function performs marker regression to associate individual markers with traits (or eigentraits). If n_perm is greater than 0, permutations are run to determine effect size thresholds for the alpha values provided. The default alpha values are 0.05 and 0.01. Covariates are specified in the cape parameter file.

Usage

```
singlescan(
  data_obj,
  geno_obj,
  kin_obj = NULL,
  n_perm = 0,
  alpha = c(0.01, 0.05),
  model_family = "gaussian",
  run_parallel = FALSE,
  n_cores = 4,
  verbose = FALSE,
  overwrite_alert = TRUE
)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object.
kin_obj	a kinship object. If NULL, the kinship correction is not performed.
n_perm	integer number of permutations. Permutation results are only used in plot_singlescan. They are not used for any other piece of the Cape analysis and may be safely omitted. The default number of permutations is 0.
alpha	significance level if permutations are being run. If permutations are run effect size thresholds for each alpha level are cacluated using the extreme value distribution.
model_family	A vector indicating the model family of the phenotypes. This can be either "gaussian" or "binomial." If length 1, all phenotypes will be assigned to the same family. Phenotypes can be assigned different model families by providing a vector of the same length as the number of phenotypes, indicating how each phenotype should be modeled.
run_parallel	Whether to run on parallel CPUs
n_cores	The number of CPUs to use if run_parallel is TRUE
verbose	Whether to print progress to the screen
overwrite_alert	Used

Details

model_family indicates the model family of the phenotypes This can be either "gaussian" or "binomial". If this argument is length 1, all phenotypes will be assigned to the same family. Phenotypes can be assigned different model families by providing a vector of the same length as the number of phenotypes, indicating how each phenotype should be modeled.

Value

Returns a list of the singlescan results. The list is of length seven, and has the following elements: alpha: The alpha values set in the argument alpha alpha_thresh: The calculated effect size thresholds at each alpha if permutations are run. ref_allele: The allele used as the reference allele singlescan_effects: The effect sizes (beta coefficients) from the single-locus linear models singlescan_t_stats: The t statistics from the single-locus linear models locus.p_vals: Marker-level p values locus_score_scores: Marker-level test statistics.

See Also

plot_singlescan

Examples

```
## Not run:
singlescan_obj <- singlescan(data_obj, geno_obj, kin_obj)

## End(Not run)
```

---

write_population *Save the cross data in R/qtl CSV format*

Description

This function writes out a cape object in a csv format readable both by read_population in Cape or by read.cross in R/qtl.

Usage

```
write_population(
  data_obj,
  geno_obj,
  ref_allele = "A",
  na = NA,
  filename = "capeData.csv"
)
```

Arguments

data_obj	a Cape object
geno_obj	a genotype object
ref_allele	a character, e.g., "A", that represents the reference allele in the data object
na	either NA or a character used to represent a missing data value in the output
filename	absolute or relative path to the output file's destination

Value

Writes a file to the destination path

References

Broman et al. (2003) R/qtl: QTL mapping in experimental crosses. Bioinformatics 19:889-890 doi:10.1093/bioinformatics/btg112

Examples

```
## Not run:
write_population(data_obj, geno_obj)

## End(Not run)
```

---

write_variant_influences

Write significant cape interactions to a csv file

Description

This function takes in the final data object and writes the variant influences that are at or below the specified significance level.

Usage

```
write_variant_influences(
  data_obj,
  p_or_q = 0.05,
  include_main_effects = TRUE,
  filename = "Variant.Influences.csv",
  delim = ",",
  mark_covar = FALSE,
  write_file = TRUE
)
```

Arguments

data_obj	a Cape object
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.
include_main_effects	Whether to include main effects (TRUE) or only interaction effects (FALSE) in the output table.
filename	A character vector specifying the name of the file.
delim	A character string indicating the delimiter 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.

Details

The columns of the output file are the following: Source: The marker that is the source of the directed interaction Chr: The chromosome on which the source marker lives Position: The genomic position of the source marker Target: If the effect is an interaction, this column lists the marker that is the target of the directed interaction. If the effect is a main effect, this column lists the trait that is the target of the main effect. Chr: The chromosome on which the target marker lives. If the effect is a main effect, this is listed as 0. Position: The genomic position of the target marker. If the effect is a main effect, this is listed as 1. Conditioning: If the effect is a main effect, this column identifies the marker on which the main effect marker was conditioned when it had it's largest main effect. Chr: If the effect is a main effect, this column lists the chromosome on which the conditioning marker lives Position: If the effect is a main effect, this column lists the genomic position of the conditioning marker. Effect: The effect size of the effect, either main effect or interaction. SE: The standard error of the effect, either main effect or interaction. |Effect|/SE: The standardized effect P_empirical: The empirical p value calculated from permutations p_adjusted: The p value adjusted by the method specified in the parameter file.

Value

If write_file is TRUE, this function writes the results table to a file and invisibly returns the table. If write_file is FALSE, the function returns the results table without writing to file.

Examples

```
## Not run:
inf_table <- write_variant_influences(data_obj)

## End(Not run)
```

Index