Package ‘mappoly2’

January 29, 2024

Title Genetic Linkage Maps in Autopolyploids

Version 2.0.0

Maintainer Marcelo Mollinari <mmollin@ncsu.edu>

Description Enhanced and streamlined genetic mapping in polyploid organisms.
Building upon the foundational capabilities of 'mappoly', it
introduces several key improvements and features that
make it more efficient and user-friendly, especially for
integration with interactive applications like Shiny.

License MIT + file LICENSE

LazyData TRUE

LazyDataCompression xz

Depends R (>= 4.2.0)

Imports Rcpp (>= 1.0.0),
RcppParallel,
assertthat,
ggplot2,
dendextend,
princurve,
smacof,
fields,
cli,
crayon,
magrittr,
rstudioapi,
AGHmatrix,
gatepoints,
CMplot,
reshape2,
dplyr,
tibble,
eulerr

LinkingTo Rcpp, RcppProgress, RcppArmadillo, RcppParallel

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

SystemRequirements GNU make

Suggests testthat,
knitr,
rmarkdown,
viridis

URL https://github.com/mmollina/mappoly2

BugReports https://github.com/mmollina/mappoly2/issues

VignetteBuilder knitr

R topics documented:

`add_marker` *Add a Marker to a Pre-Mapped Sequence*

Description

This function adds a specified marker to a pre-mapped genetic sequence within a `mappoly2.sequence` object. It assesses the best position for the marker in the sequence based on phase configurations and recombination fractions.

Usage

```
add_marker(
  x,
  mrk,
  lg = NULL,
  type = c("mds", "genome", "custom"),
  parent = c("p1p2", "p1", "p2"),
  reestimate.map.and.haplo = FALSE,
  verbose = TRUE,
  tol = 0.001,
  thresh.LOD.ph = 5,
  thresh.LOD.rf = 5,
  thresh.rf = 0.5,
  max.phases = 5,
  thresh.LOD.ph.to.insert = 10,
  thresh.rf.to.add = NULL
)
```

Arguments

`x`	A `mappoly2.sequence` object.
`mrk`	The name of the marker to be added.
`lg`	The linkage group to which the marker will be added.
`type`	The type of sequence ('mds', 'genome', or 'custom').
`parent`	Specifies the parent phase ('p1p2', 'p1', or 'p2').
`reestimate.map.and.haplo`	Logical; if `TRUE`, re-estimates the map and haplotype probabilities after adding the marker (default is `FALSE`).
`verbose`	Logical; if `TRUE`, function prints messages during execution (default is `TRUE`).
`tol`	Tolerance level for re-estimating the map.
`thresh.LOD.ph`	LOD threshold for considering phase configurations (default = 5).
`thresh.LOD.rf`	LOD threshold for recombination fraction (default = 5).
`thresh.rf`	Recombination fraction threshold (default = 0.5).

`max.phases`	The maximum number of phase configurations to consider (default = 5).
`thresh.LOD.ph.to.insert`	Threshold LOD for inserting the marker into the map (default = 10).
`thresh.rf.to.add`	Recombination fraction threshold for adding the marker (default is the maximum of existing recombination fractions).

Details

The function operates as follows:

1. Validates the input object and parameters.
2. Uses `test_one_marker` to evaluate all possible phase configurations for the specified marker.
3. Checks if the best phase configuration meets the specified LOD and recombination fraction thresholds.
4. Determines the optimal position for the marker within the linkage group based on the phase test results.
5. Inserts the marker into the sequence at the determined position.
6. Optionally re-estimates the map if `reestimate.map.and.haplo` is `TRUE`.

Value

Returns the modified `mappoly2.sequence` object with the new marker added.

See Also

`drop_marker` for removing markers from a sequence.

`alfa_bc` *Autotetraploid alfalfa BC dataset.*

Description

This dataset comprises 93 offspring from a cross between two tetraploid alfalfa parents, I195 and F1.85.209, the latter being derived from the cross between I195 and J432. The biparental population was genotyped with the alfalfa DArTag panel described in Zhao et al., (2023)

Usage

```
alfa_bc
```

Format

This list is of class `mappoly2.data` and contains the following components:

ploidy.p1 The ploidy level of parent P1 (I195) is 4.

ploidy.p2 The ploidy level of parent P2 (F1.85.209) is 4.

n.ind The total number of individuals is 93.

n.mrk The total number of unique markers is 1923, filtered for redundancy.

ind.names A character vector of the individuals' names.

mrk.names A character vector of the markers' names.

name.p1 The name of parent P1 is I195.

name.p2 The name of parent P2 is F1.85.209.

dosage.p1 A named integer vector containing the dosage in parent P1 for all `n.mrk` markers.

dosage.p2 A named integer vector containing the dosage in parent P2 for all `n.mrk` markers.

chrom A named character vector indicating the chromosome each marker belongs to.

genome.pos A named numeric vector containing the physical position of the markers in the sequence.

ref A character vector of the reference allele.

alt A character vector of the alternate allele.

geno.dose A numeric matrix containing the dosage for each marker (rows) for each individual (columns).

redundant A data frame containing two character vectors, `kept` and `removed`, of markers filtered for redundancy.

QAQC.values A list containing quality assurance and quality control values with the following components:

- A data frame with statistics for each marker, including `miss` (missing data rate), `chisq.pval` (chi-squared test p-value), and `read.depth` (read depth).
- A data frame with statistics for each individual, including `miss` (missing data rate) and `full.sib` (indicator of non-belonging to the analyzed bi-parental cross, generated by `filter_individuals`).

`alfa_f1` *Autotetraploid alfalfa F1 dataset.*

Description

This dataset comprises 184 offspring from a cross between two tetraploid alfalfa parents, I195 and J432, which are resistant and susceptible to Aphanomyces euteiches, respectively. The biparental population was genotyped with the alfalfa DArTag panel described in Zhao et al., (2023)

Usage

```
alfa_f1
```

Format

This list is of class `mappoly2.data` and contains the following components:

ploidy.p1 The ploidy level of parent P1 (I195) is 4.

ploidy.p2 The ploidy level of parent P2 (J432) is 4.

n.ind The total number of individuals is 184.

n.mrk The total number of unique markers is 2795, filtered for redundancy.

ind.names A character vector of the individuals' names.

mrk.names A character vector of the markers' names.

name.p1 The name of parent P1 is I195.

name.p2 The name of parent P2 is J432.

dosage.p1 A named integer vector containing the dosage in parent P1 for all `n.mrk` markers.

dosage.p2 A named integer vector containing the dosage in parent P2 for all `n.mrk` markers.

chrom A named character vector indicating the chromosome each marker belongs to.

genome.pos A named numeric vector containing the physical position of the markers in the sequence.

ref A character vector of the reference allele.

alt A character vector of the alternate allele.

geno.dose A numeric matrix containing the dosage for each marker (rows) for each individual (columns).

redundant A data frame containing two character vectors, `kept` and `removed`, of markers filtered for redundancy.

QAQC.values A list containing quality assurance and quality control values with the following components:

- A data frame with statistics for each marker, including `miss` (missing data rate), `chisq.pval` (chi-squared test p-value), and `read.depth` (read depth).
- A data frame with statistics for each individual, including `miss` (missing data rate) and `full.sib` (indicator of non-belonging to the analyzed bi-parental cross, generated by `filter_individuals`).

`augment_phased_map` *Augment a Phased Genetic Map with Unprocessed Markers*

Description

This function efficiently phases unprocessed markers in a given phased genetic map, circumventing the need for complete HMM-based recomputation of the map. It is designed to update a genetic map with new marker data while maintaining the integrity and structure of the existing map.

Usage

```
augment_phased_map(
  x,
  lg = NULL,
  type = c("mds", "genome"),
  ncpus = 1,
  thresh.LOD.ph = 5,
  thresh.LOD.rf = 5,
  thresh.rf = 0.5,
  max.phases = 5,
  thresh.LOD.ph.to.insert = 10,
  thresh.dist.to.insert = NULL,
  reestimate.hmm = TRUE,
  tol = 0.01,
  final.tol = 0.001,
  final.error = NULL,
  verbose = TRUE
)
```

Arguments

`x`	An object of class `mappoly2.sequence`
`lg`	Optional vector specifying the linkage groups to be processed. If NULL, all linkage groups in the object are considered.
`type`	The type of genetic data to be processed, either 'mds' or 'genome'.
`ncpus`	The number of CPU cores to use for parallel processing.
`thresh.LOD.ph`	Threshold for the LOD (Logarithm of the Odds) score for phasing.
`thresh.LOD.rf`	Threshold for the LOD score for recombination fractions.
`thresh.rf`	Threshold for recombination fraction.
`max.phases`	The maximum number of phase configurations allowed for a marker.
`thresh.LOD.ph.to.insert`	Threshold LOD score for inserting a marker into the map.
`thresh.dist.to.insert`	Optional threshold for marker distance (in cM) when inserting markers. If NULL, the maximum distance in the map is used.
`reestimate.hmm`	Logical flag indicating whether to reestimate the HMM (Hidden Markov Model) after marker insertion.
`tol`	Tolerance level for numerical computations.
`final.tol`	Final tolerance level for HMM reestimation.
`final.error`	Final error level for HMM reestimation.
`verbose`	Logical flag indicating whether to print detailed output during function execution.

Details

The function works by first identifying unprocessed markers in the genetic map and then using a phasing algorithm to integrate these markers into the existing map. It can handle large datasets and is optimized for performance with options for parallel processing.

Value

Returns the updated genetic map object with newly phased markers.

`B2721` *Autotetraploid potato dataset.*

Description

This dataset comprises 156 offspring from a cross between two tetraploid potato varieties, Atlantic and B1829-5. It is genotyped with the SolCAP Infinium 8303 potato array, and the genomic order of SNPs from the Solanum tuberosum genome version 4.03 is included. Genotype calling was performed using the ClusterCall R package. The original dataset can be found in Pereira et al., (2021)

Usage

```
B2721
```

Format

This list is of class `mappoly2.data` and contains the following components:

ploidy.p1 The ploidy level of parent P1 (Atlantic) is 4.

ploidy.p2 The ploidy level of parent P2 (B1829-5) is 4.

n.ind The total number of individuals is 156.

n.mrk The total number of unique markers is 6511, filtered for redundancy.

ind.names A character vector of the individuals' names.

mrk.names A character vector of the markers' names.

name.p1 The name of parent P1 is Atlantic.

name.p2 The name of parent P2 is B1829-5.

dosage.p1 A named integer vector containing the dosage in parent P1 for all `n.mrk` markers.

dosage.p2 A named integer vector containing the dosage in parent P2 for all `n.mrk` markers.

chrom A named character vector indicating the chromosome each marker belongs to.

genome.pos A named numeric vector containing the physical position of the markers in the sequence.

ref A character vector of the reference allele.

alt A character vector of the alternate allele.

geno.dose A numeric matrix containing the dosage for each marker (rows) for each individual (columns).

redundant A data frame containing two character vectors, `kept` and `removed`, of markers filtered for redundancy.

QAQC.values A list containing quality assurance and quality control values with the following components:

- A data frame with statistics for each marker, including `miss` (missing data rate), `chisq.pval` (chi-squared test p-value), and `read.depth` (read depth).
- A data frame with statistics for each individual, including `miss` (missing data rate) and `full.sib` (indicator of non-belonging to the analyzed bi-parental cross, generated by `filter_individuals`).

`calc_consensus_haplo` *Calculate Haplotype Probabilities for Consensus Maps*

Description

This function calculates haplotype probabilities for each linkage group in a consensus map dataset of interconnected F1 populations. It utilizes parallel processing for efficient computation when multiple CPU cores are available.

Usage

```
calc_consensus_haplo(x, ncpus = 1)
```

Arguments

`x`	An object of class 'mappoly2.consensus.map', representing consensus map data for interconnected F1 populations.
`ncpus`	The number of CPU cores to use for parallel processing, defaults to 1. If more than 1 core is specified, parallel processing is used.

Details

The function processes the consensus map data to calculate haplotype probabilities using a biallelic model. It leverages parallel processing capabilities when multiple cores are available, significantly improving efficiency on large datasets.

Value

The input object `x` with haplotype probabilities calculated for each consensus map.

`calc_haplotypes` *Calculate Haplotype Probabilities Using Hidden Markov Models*

Description

This function calculates haplotypes for each linkage group in a genetic mapping dataset. It utilizes Hidden Markov Models and supports both sequential and parallel processing for efficient computation.

Usage

```
calc_haplotypes(
  x,
  lg = NULL,
  type = c("mds", "genome"),
  parent = c("p1p2", "p1", "p2"),
  phase.conf = "all",
  verbose = TRUE,
  ncpus = 1
)
```

Arguments

`x`	An object representing genetic mapping data, typically of a specific class that stores genetic information.
`lg`	Optional vector specifying the linkage group indices to process. If NULL, all linkage groups in `x` are processed.
`type`	Character vector indicating the type of map to process, either "mds" or "genome".
`parent`	Character vector specifying the parent or parents to be considered in the haplotype calculation. Options are "p1p2" (both parents), "p1" (first parent), and "p2" (second parent).
`phase.conf`	A configuration parameter for phase calculation.
`verbose`	Logical value; if TRUE, progress messages will be printed.
`ncpus`	The number of CPU cores to use for parallel processing. Defaults to 1.

Details

The function processes the genetic data to calculate haplotypes for each specified linkage group using Hidden Markov Models. It can handle large datasets efficiently by using parallel processing capabilities.

Value

The input object `x` with haplotypes calculated for the specified linkage groups.

Author(s)

Marcelo Mollinari, `<mmollin@ncsu.edu>`

`compare_order` *Compare Marker Orders from MDS and Genome Maps*

Description

This function compares two pre-built maps, one using MDS (multidimensional scaling) and the other using genome order, based on their HMM (hidden Markov model) multilocus-based likelihoods. It serves as an objective function to assist in selecting the best map by ensuring that only markers present in both maps are considered for a proper comparison.

Usage

```
compare_order(
  x,
  parent = c("p1p2", "p1", "p2"),
  ncpus = 1,
  error = 0,
  verbose = TRUE,
  tol = 0.01
)
```

Arguments

`x`	A `mappoly2.sequence` object containing the pre-built maps.
`parent`	The parent phase to consider ('p1p2', 'p1', or 'p2').
`ncpus`	Integer specifying the number of CPU cores for parallel processing. Default is 1 (sequential processing).
`error`	Optional error parameter for comparison adjustments.
`verbose`	Logical; if `TRUE`, prints messages during the function execution.
`tol`	Tolerance level for likelihood comparison.

Value

A matrix comparing the HMM likelihoods of markers across MDS and genome maps.

drop_marker *Drop Marker(s) from a Sequence*

Description

This function removes specified marker(s) from a given linkage group and type within a `mappoly2.sequence` object. It can optionally re-estimate the map after marker removal.

Usage

```
drop_marker(
  x,
  mrk,
  lg = NULL,
  type = c("mds", "genome", "custom"),
  parent = c("p1p2", "p1", "p2"),
  reestimate.map.and.haplo = FALSE,
  verbose = TRUE,
  tol = 0.001
)
```

Arguments

`x`	A `mappoly2.sequence` object.
`mrk`	Marker(s) to be dropped, specified by name or index.
`lg`	Linkage group from which the marker(s) should be removed. If `NULL` (default), all groups are considered.
`type`	Type of sequence ('mds', 'genome', or 'custom') to specify which part of the sequence the marker(s) should be removed from.
`parent`	Indicates the parent phase to consider ('p1p2', 'p1', or 'p2').
`reestimate.map.and.haplo`	Logical; if `TRUE`, the genetic map and haplotype probabilities are re-estimated after marker removal (default is `FALSE`).
`verbose`	Logical; if `TRUE`, function will print messages during execution (default is `TRUE`).
`tol`	Tolerance level used in map re-estimation (applicable if `reestimate.map.and.haplo` is `TRUE`).

Details

The function identifies and removes the specified marker(s) from the linkage group and type within the `mappoly2.sequence` object. If `reestimate.map.and.haplo` is `TRUE`, it also re-estimates the genetic map using the provided tolerance level. The function validates the presence of the marker(s) and handles missing or incorrect marker specifications gracefully.

Value

Returns a modified `mappoly2.sequence` object with the specified marker(s) removed.

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

See Also

add_marker for adding marker in a sequence.

---

estimate_consensus_map *Estimate Consensus Genetic Map*

Description

This function estimates a consensus genetic map among different biparental populations.

Usage

```
estimate_consensus_map(
  x,
  err = 0,
  ncpus = 1,
  verbose = TRUE,
  detailed_verbose = FALSE,
  tol = 0.001,
  ret_H0 = FALSE
)
```

Arguments

x	A list of 'mappoly2.prepared.integrated.data' objects.
err	Error rate to be used in the HMM map estimation (default is 0.0).
ncpus	Number of CPUs to use for parallel processing (default is 1). Automatically adjusted to not exceed the number of available cores.
verbose	Logical; if TRUE, enables the printing of progress messages (default is TRUE).
detailed_verbose	Logical; if TRUE, enables the printing of detailed progress messages (default is FALSE).
tol	Tolerance level for the estimation convergence (default is 10e-4).
ret_H0	Logical; if TRUE, returns null hypothesis estimates (default is FALSE).

Value

A list of results from the consensus map estimation.

`filter_data` *Filter Genetic Data Based on Quality Metrics*

Description

This function filters genetic data in a `mappoly2.data` object based on various quality control metrics. It applies thresholds for missing data rates, chi-squared p-values, and read depth to markers and individuals, and optionally plots the screening process.

Usage

```
filter_data(
  x,
  mrk.thresh = 0.1,
  ind.thresh = 0.1,
  chisq.pval.thresh = NULL,
  read.depth.thresh = c(5, 1000),
  plot.screening = TRUE
)
```

Arguments

`x`	A `mappoly2.data` object containing genetic data.
`mrk.thresh`	A numeric threshold for the missing data rate in markers (default is 0.10).
`ind.thresh`	A numeric threshold for the missing data rate in individuals (default is 0.10).
`chisq.pval.thresh`	A numeric threshold for chi-squared test p-values in markers (default is NULL, which sets the threshold using a Bonferroni approximation).
`read.depth.thresh`	A numeric vector with two values indicating the lower and upper bounds for acceptable read depths in markers (default is c(5, 1000)).
`plot.screening`	Logical, if TRUE (default), plots are generated to visually represent the screening process.

Details

The function first validates the input object, then applies the specified thresholds to filter out markers and individuals based on missing data rates, chi-squared p-values, and read depths. It updates the object with the results of this filtering and optionally generates plots to visualize the data before and after filtering.

Value

Returns the input `mappoly2.data` object with additional components:

- `screened.data`: A list containing the thresholds used for selection and the names of markers and individuals that met the specified criteria.
- Class attribute `screened` is also appended to the object.

Examples

```
filtered_data <- filter_data(B2721)
```

`filter_individuals` *Filter out individuals*

Description

This function removes individuals from the input dataset, either by specifying them manually or by using interactive kinship analysis.

Usage

```
filter_individuals(
  x,
  ind.to.remove = NULL,
  inter = TRUE,
  type = c("Gmat", "PCA"),
  verbose = TRUE
)
```

Arguments

x	The name of the input object (class `mappoly.data`).
ind.to.remove	A character vector containing the names of the individuals to be removed. If `NULL`, the function opens an interactive graphic to allow for individual selection.
inter	If `TRUE`, the function expects user input to proceed with filtering.
type	A character string specifying the procedure to be used for detecting outlier offspring. Options include "Gmat", which utilizes the genomic kinship matrix, and "PCA", which employs principal component analysis on the dosage matrix.
verbose	If `TRUE` (default), the function shows the list of filtered out individuals.

Author(s)

Marcelo Mollinari, <`mmollin@ncsu.edu`>

`genome_order` *Get the Genomic Position of Markers in a Sequence*

Description

This internal function retrieves and orders the genomic position of markers in a given genetic sequence. It is primarily used within a larger analytical context.

Usage

```
genome_order(x, mrk.names, verbose = TRUE)
```

Arguments

`x`	An object of class `mappoly2.sequence`.
`mrk.names`	A vector of marker names for which the genomic position is required.
`verbose`	Logical; if TRUE, progress messages will be printed.

Details

The function checks for the availability of genomic position information in the provided data object. If available, it orders the markers based on their chromosome and sequence position. The function handles cases where only chromosome information or only sequence position information is available.

Value

Returns a data frame with the genomic position of the specified markers, ordered by chromosome and sequence position.

Author(s)

Marcelo Mollinari, `<mmollin@ncsu.edu>`

`group` *Assign Markers to Linkage Groups*

Description

This function identifies linkage groups of markers using the results from two-point (pairwise) analysis in a `mappoly2.data` or `mappoly2.sequence` object.

Usage

```
group(
  x = NULL,
  expected.groups = NULL,
  inter = TRUE,
  comp.mat = FALSE,
  LODweight = FALSE
)
```

Arguments

`x`	A `mappoly2.data` object that contains recombination fraction information.
`expected.groups`	The expected number of linkage groups (e.g., chromosomes) for the species, if known. This parameter is mandatory unless `inter` is set to `TRUE`.
`inter`	Logical; if `TRUE`, the function allows interactive determination of the expected number of groups by plotting a dendrogram (default is `TRUE`).
`comp.mat`	Logical; if `TRUE`, displays a comparison matrix between the reference-based and linkage-based groupings if chromosome information is available (default is `FALSE`).
`LODweight`	Logical; if `TRUE`, clustering is weighted by the square of the LOD score (default is `FALSE`).

Details

The function first validates the input object and then performs hierarchical clustering on the recombination fraction matrix. If `LODweight` is `TRUE`, the clustering is weighted by the square of the LOD scores. The function can interactively determine the number of expected groups and allows users to confirm or adjust this number. A dendrogram is plotted for visual inspection. The function also optionally compares the assigned groups with genomic group information.

Value

An object of class `mappoly2.group`, which includes:

hc.snp A hierarchical clustering object generated by UPGMA or average method.

expected.groups The specified or interactively determined number of expected linkage groups.

groups.snp The assigned linkage groups for each marker.

seq.vs.grouped.snp Comparison matrix between genomic group information and the assigned groups, if `comp.mat` is `TRUE`.

Author(s)

Marcelo Mollinari, <`mmollin@ncsu.edu`>

`make_sequence` *Create a Genetic Sequence from a Screened or Grouped Object*

Description

This function generates a genetic sequence based on the provided screened or grouped object. It allows for the creation of sequences using specific markers, linkage groups, and chromosome information.

Usage

```
make_sequence(x, lg = NULL, ch = NULL, mrk.id.list = NULL)
```

Arguments

`x`	An object of class 'screened' or 'mappoly2.group'.
`lg`	A list specifying the linkage groups, if applicable.
`ch`	A list specifying the chromosomes, if applicable.
`mrk.id.list`	A list or vector of marker IDs to be used in the sequence.

Details

The function first checks if `mrk.id.list` is provided and is not null, and uses the marker information to initiate the sequence. If `lg` and `ch` are provided, they are used to determine the sequence structure. The function performs various checks and validations on the input parameters and the structure of the `x` object.

```
    If `x` is of class 'mappoly2.group', the function uses data within the
   group object for sequence creation. If `lg` and `ch` are null, the function
  uses only the results of the UPGMA (Unweighted Pair Group Method with Arithmetic Mean).
```

Value

Returns a sequence object based on the provided parameters. The sequence is constructed using internal function `.map_skeleton`.

`mapping` *Multi-Locus Map Estimation for mappoly2 Data Objects*

Description

This function performs multi-locus map estimation on mappoly2 data objects, supporting both sequential and parallel processing. It is designed to efficiently handle large datasets, making it suitable for complex genetic mapping tasks.

Usage

```
mapping(
  x,
  lg = NULL,
  type = c("mds", "genome", "custom"),
  parent = c("p1p2", "p1", "p2"),
  recompute.from.pairwise = FALSE,
  phase.conf = "all",
  rf = NULL,
  error = 0,
  ncpus = 1,
  verbose = TRUE,
  tol = 0.001,
  ret_H0 = FALSE
)
```

Arguments

`x`	An object of class `mappoly2.sequence`, representing the genetic map data.
`lg`	Optional vector specifying linkage group indices to be processed. If NULL, all linkage groups in the data object are considered.
`type`	A character vector indicating the type of mapping to perform. Options include "mds" (multi-dimensional scaling), "genome", and "custom". Default is c("mds", "genome", "custom").
`parent`	A character vector specifying the parent or parents to be considered in the mapping process. Options are "p1p2" (both parents), "p1" (first parent), and "p2" (second parent). Default is c("p1p2", "p1", "p2").
`recompute.from.pairwise`	A logical value indicating whether to recompute the map from pairwise data.
`phase.conf`	A configuration parameter for phase determination.
`rf`	Recombination fraction, used in the mapping calculations.
`error`	Error tolerance in the mapping process.
`ncpus`	Integer specifying the number of CPU cores for parallel processing. Default is 1 (sequential processing).

`verbose`	Logical value; if TRUE, detailed progress information is printed during processing. Default is TRUE.
`tol`	Tolerance level for the mapping algorithm.
`ret_H0`	Logical; if TRUE, hypothesis testing results are returned.

Details

The function processes each specified linkage group, performing phase determination and map estimation based on the provided parameters. It can utilize parallel processing to enhance performance on large datasets.

Value

Returns an updated `mappoly2.sequence` data object with mapped linkage groups.

Author(s)

Marcelo Mollinari, `<mmollin@ncsu.edu>`

`mappoly2` *mappoly2: A package for constructing genetic maps in autopolyploids*

Description

mappoly2 provides ...

mappoly main functions

The mappoly2 functions ...

`mappoly_to_csv_mappoly2` *Convert MAPpoly Data to CSV*

Description

This function takes MAPpoly data objects and writes them to CSV files. Each object in the list is processed individually, and a CSV file is generated for each. The user can specify the path for saving the CSV files, and custom names for the parents in the dataset.

Usage

```
mappoly_to_csv_mappoly2(x, path = NULL, parent.names = NULL)
```

Arguments

`x`	A list of MAPpoly data objects or a single MAPpoly data object. The function checks if `x` is a list, and if not, it converts it into a list.
`path`	Optional; a string specifying the directory where the CSV files will be saved. If not provided, files are saved in the current working directory.
`parent.names`	Optional; a matrix of names for the parent genotypes. If not provided, default names are generated in the format P1, P2, etc. The matrix should have two columns and a number of rows equal to the length of `x`.

Details

This function processes each MAPpoly data object, extracting relevant data for CSV output. It handles missing reference and alternate sequences, and ensures that the provided parent names match the structure of the data. The function creates one CSV file per MAPpoly data object.

Value

Invisible NULL. The function is used for its side effect of writing files.

`map_summary` *Summarize Genetic Mapping Data*

Description

This function provides a summary of genetic mapping data, including information about marker distribution, map lengths, and gap sizes. It's designed to work with specific types of genetic data structures in R, offering a concise overview of the mapping results.

Usage

```
map_summary(
  x,
  type = c("both", "mds", "genome"),
  parent = c("p1p2", "p1", "p2")
)
```

Arguments

`x`	A genetic mapping data object, typically containing information about maps, markers, and related genetic data.
`type`	The type of summary required: 'both', 'mds', or 'genome'. 'mds' refers to Multi-Dimensional Scaling, and 'genome' refers to the entire genomic data. 'both' will provide information for both mds and genome.
`parent`	Specifies which parent's data to use in the summary. Options are 'p1p2' (both parents), 'p1' (first parent), or 'p2' (second parent).

Details

The function processes the mapping data based on the specified 'type' and 'parent' parameters. It calculates various statistics including the maximum gap size in the map and the distribution of different types of markers. The output is formatted as a table for easy viewing and interpretation.

Value

An invisible data frame containing the mapping summary, including linkage groups (LG), chromosome data (Chrom), map lengths, number of markers per centiMorgan (Markers/cM), information on simplex markers for each parent, double-simplex, multiplex, total marker count, and the maximum gap size.

---

mds *Estimates Loci Position Using Multidimensional Scaling*

Description

This function estimates loci positions using the Multidimensional Scaling (MDS) method. The method is adapted from the package `MDSmap` and is based on the approach proposed by Preedy and Hackett (2016).

Usage

```
mds(
  x,
  mrk.id,
  p = NULL,
  n = NULL,
  ndim = 2,
  weight.exponent = 2,
  verbose = TRUE
)
```

Arguments

`x`	An object of class `mappoly2.sequence`.
`mrk.id`	A vector of marker IDs to be used in the analysis.
`p`	Integer; the smoothing parameter for the principal curve. If `NULL` (default), this will be determined using leave-one-out cross-validation.
`n`	Vector of integers or strings containing loci to be omitted from the analysis.
`ndim`	Integer; the number of dimensions to be considered in the MDS procedure (default = 2).
`weight.exponent`	Integer; the exponent used in the LOD score values to weight the MDS procedure (default = 2).
`verbose`	Boolean; if `TRUE` (default), displays information about the analysis.

Details

The function employs MDS to estimate the positions of loci on a genetic map. It includes options for excluding certain loci, adjusting the number of dimensions, and setting weighting factors for LOD scores. The results include detailed information about the estimated positions and relationships between loci.

Value

A list containing various components related to MDS results, including the input distance map, unconstrained MDS results, principal curve results, matrix of pairwise distances, data frame of loci positions, total length of the segment, vector of removed loci, scaling factor, and data frames for interpreting MDS plots.

Author(s)

Marcelo Mollinari, adapted from MDSmap codes by Katharine F. Preedy

References

Preedy, K. F., & Hackett, C. A. (2016). A rapid marker ordering approach for high-density genetic linkage maps in experimental autotetraploid populations using multidimensional scaling. *Theoretical and Applied Genetics*, 129(11), 2117-2132. doi:10.1007/s0012201627618

`merge_datasets` *Merge Multiple Genomic Datasets*

Description

This function takes any number of genomic datasets of class 'mappoly2.data' and merges them. If only one dataset is provided, the function will issue a warning and return the original dataset. All datasets need to be of class 'mappoly2.data'. Additional data screening and filtering is performed, including computing chi-square p-values, screening non-conforming markers, and filtering redundant markers.

Usage

```
merge_datasets(..., filter.non.conforming = TRUE, filter.redundant = TRUE)
```

Arguments

`...`	Comma-separated list of datasets of class 'mappoly2.data'.
`filter.non.conforming`	if `TRUE` (default), data points with unexpected genotypes (i.e. double reduction) are converted to 'NA'. See the `segreg_poly` function for information on expected classes and their respective frequencies.
`filter.redundant`	logical. If `TRUE` (default), removes redundant markers during map construction, keeping them annotated to export to the final map.

Value

The merged and filtered dataset if more than one dataset was provided; the original dataset if only one was provided.

Examples

```
## Not run:
data1 <- subset(B2721, type = "marker", n = 200)
data1 <- subset(data1, type = "individual", n = 80)
data2 <- subset(B2721, type = "marker", n = 300)
data2 <- subset(data2, type = "individual", n = 100)
data3 <- subset(B2721, type = "marker", n = 400)
data3 <- subset(data3, type = "individual", n = 100)

merged_data <- merge_multiple_datasets(data1, data2, data3)

merged_data

plot(merged_data)

## End(Not run)
```

`merge_single_parent_maps` *Merge Single Parent Genetic Maps*

Description

This function merges two genetic maps built from markers that are exclusively informative in isolated parents. It facilitates unified analysis and visualization of distinct genetic data.

Usage

```
merge_single_parent_maps(
  x,
  lg = NULL,
  type = c("mds", "genome", "custom"),
  hmm.reconstruction = TRUE,
  rf = NULL,
  error = 0,
  ncpus = 1,
  verbose = TRUE,
  tol = 0.001,
  ret_H0 = FALSE
)
```

Arguments

`x`	An object of class `mappoly2.sequence` containing maps constructed for each parent separately.
`lg`	Optional vector specifying the linkage groups to be processed. If NULL, all linkage groups in `x` are considered.
`type`	The type of genetic maps to be merged, options include "mds", "genome", or "custom".
`hmm.reconstruction`	Logical; if TRUE, HMM-based reconstruction of the merged map is performed.

`rf`	Recombination fraction, used in the merging calculations.
`error`	Error tolerance in the merging process.
`ncpus`	Integer specifying the number of CPU cores for parallel processing.
`verbose`	Logical; if TRUE, progress messages will be printed.
`tol`	Tolerance level for the merging algorithm.
`ret_H0`	Logical; if TRUE, hypothesis testing results are returned.

Details

The function merges separate genetic maps for individual parents into a single map for each linkage group. It handles the alignment and integration of markers from both parents and optionally performs HMM-based reconstruction of the merged maps.

Value

Returns the `mappoly2.sequence` object with the merged genetic maps for the specified linkage groups.

Author(s)

Marcelo Mollinari, `<mmollin@ncsu.edu>`

`order_sequence` *Order Genetic Sequence in a Mapping Data Object*

Description

This function orders the genetic sequence within a mapping data object. It can operate on different types of genetic data (e.g., MDS or genomic) and allows for customization of the ordering process through various parameters.

Usage

```
order_sequence(
  x,
  lg = NULL,
  type = c("mds", "genome"),
  p = NULL,
  n = NULL,
  ndim = 2,
  weight.exponent = 2,
  verbose = TRUE
)
```

Arguments

`x`	A mapping data object that contains genetic mapping information.
`lg`	An optional vector specifying the linkage groups to be ordered. If NULL, all linkage groups in the data object are considered.
`type`	The type of genetic data to be ordered, either 'mds' for Multi-Dimensional Scaling or 'genome' for genomic data.
`p`	An optional parameter for the ordering function, specific to the method being used (e.g., MDS).
`n`	An optional parameter for the ordering function, specific to the method being used.
`ndim`	The number of dimensions to be used in the ordering process. This parameter is primarily relevant for MDS.
`weight.exponent`	The exponent for weighting in the ordering process.
`verbose`	A logical value indicating whether to print detailed output during the execution of the function.

Details

The function iterates over the specified linkage groups (or all groups if none are specified) and applies either MDS or genomic ordering based on the 'type' parameter. Additional parameters like 'p', 'n', and 'ndim' are used to fine-tune the ordering process.

Value

The function returns the modified mapping data object with updated order of the genetic sequence.

`pairwise_phasing` *Phasing Based on Pairwise Recombination Fraction Estimation*

Description

This function performs phasing based on pairwise recombination fraction estimation for genetic maps in a mappoly2 sequence object. It is designed to facilitate the phasing process using detailed genetic data.

Usage

```
pairwise_phasing(
  x,
  lg = NULL,
  type = c("mds", "genome"),
  parent = c("p1p2", "p1", "p2"),
  thresh.LOD.ph = 3,
  thresh.LOD.rf = 3,
  thresh.rf = 0.5,
  max.search.expansion.p1 = 10,
  max.search.expansion.p2 = max.search.expansion.p1,
  verbose = TRUE
)
```

Arguments

`x`	An object representing genetic mapping data, typically a mappoly2 sequence object.
`lg`	Optional vector specifying the linkage groups to be processed. If NULL, all linkage groups in `x` are considered.
`type`	The type of genetic maps to be processed, options include "mds", "genome", or "custom".
`parent`	Specifies which parent's data to use in the phasing process. Options are "p1p2" (both parents), "p1" (first parent), or "p2" (second parent).
`thresh.LOD.ph`	Threshold for the LOD (Logarithm of the Odds) score for phasing.
`thresh.LOD.rf`	Threshold for the LOD score for recombination fractions.
`thresh.rf`	Threshold for recombination fraction.
`max.search.expansion.p1`	The maximum number of search expansions for parent 1.
`max.search.expansion.p2`	The maximum number of search expansions for parent 2. Defaults to the same as `max.search.expansion.p1`.
`verbose`	Logical; if TRUE, progress messages will be printed.

Details

The function iterates over the specified linkage groups and performs phasing based on pairwise recombination fraction estimation. It supports phasing for individual parents as well as both parents together. The function handles the alignment and integration of phasing data across different maps.

Value

Returns the updated mappoly2 sequence object with the phased genetic maps.

Author(s)

Marcelo Mollinari, <`mmollin@ncsu.edu`>

`pairwise_rf` *Pairwise Two-Point Analysis*

Description

Performs a pairwise two-point analysis for selected markers in a dataset. The function estimates the recombination fraction for all possible linkage phase configurations and their respective LOD Scores, returning the most likely one.

Usage

```
pairwise_rf(
  x,
  mrk.scope = c("all", "per.chrom", "chrom"),
  chrom = NULL,
  ncpus = 1L,
  verbose = TRUE,
  tol = .Machine$double.eps^0.25
)
```

Arguments

`x`	An object of class `mappoly2.data`
`mrk.scope`	Specifies the range of markers for which the pairwise recombination fractions will be calculated. Acceptable values are "all", "per.chrom", and "chrom". See details for more information.
`chrom`	Specifies the particular chromosome for which the pairwise recombination fractions will be calculated. This argument is required when the `mrk.scope` argument is set to `"chrom"`.
`ncpus`	Number of parallel processes (cores) to use (default = 1)
`verbose`	Logical; if TRUE, progress messages will be printed.
`tol`	Desired accuracy. See `optimize()` for details

Details

The `mrk.scope` argument allows for the customization of the analysis scope. Options are:

- `"all"`: Analyze all markers in the dataset.
- `"per.chrom"`: Analyze markers within each chromosome.
- `"chrom"`: Analyze markers in a specific chromosome. Requires specifying the chromosome.

Additional arguments (`chrom` or `sequence`) must be specified when using `"chrom"` or `"seq"` options, respectively.

Value

An updated object of class `mappoly2.data` with a slot called 'pairwise.rf' containing:

- `"rec.mat"` the recombination fraction matrix
- `"lod.mat"` the LOD Scrore associated to the recombination fraction matrix
- `"lod.ph.mat"` the LOD Scrore associated to the second most likely linkage phase configuration
- `"Sh.p1"` and `"Sh.p2"` the number of homologs that share the alternate alleles for the estimated linkage phase configurationns for parents 1 and 2, respectively.

Examples

```
dat <- subset(B2721, perc = .1)
dat <- filter_data(dat, mrk.thresh = .08, ind.thresh = 0.06)
dat <- pairwise_rf(dat, mrk.scope = "chrom", chrom = "ch10")
```

plot.mappoly2.consensus.map *Plot Consensus Map*

Description

This function plots consensus genetic maps along with individual population maps. It allows the option to plot only the consensus map or include individual population maps.

Usage

```
## S3 method for class 'mappoly2.consensus.map'
plot(x, only.consensus = FALSE, col = "lightgray", ...)
```

Arguments

x	A list containing elements of 'mappoly2.prepared.integrated.data' class, which includes both individual maps and consensus map data.
only.consensus	Logical, if TRUE, only the consensus map is plotted; if FALSE, individual population maps are included (default is FALSE).
col	The color used for plotting the consensus map markers when 'only.consensus' is TRUE (default is "lightgray").
...	Additional arguments, not used in this method

Value

A ggplot object representing the plotted genetic map(s).

plot.mappoly2.data *Plot Data from mappoly2.data Object*

Description

Visualizes data from a `mappoly2.data` object, which may also have "screened" and "pairwise.rf" classes. This function provides various plotting options including recombination frequency matrices, screened data, marker density, and raw data, depending on the type of data available in the object and the selected options.

Usage

```
## S3 method for class 'mappoly2.data'
plot(x, type = c("rf", "screened", "density", "raw"), chrom = NULL, ...)
```

Arguments

`x`	A `mappoly2.data` object containing genetic mapping data. The object can also be of class "screened" and "pairwise.rf".
`type`	A character string specifying the type of plot to generate. It can be one of "rf" (recombination frequency), "screened", "density", or "raw". Default is "rf".
`chrom`	Optional; a vector of chromosome numbers or names to subset the data before plotting. If `NULL`, all chromosomes are considered.
`...`	Additional arguments, not used in this method

Details

The function can visualize data in four different ways based on the `type` parameter:

- "rf": Plots a recombination frequency matrix, available for objects with the "pairwise.rf" class.
- "screened": Shows screened data for objects with the "screened" class.
- "density": Generates a density plot of markers, requiring genome position information.
- "raw": Displays raw data from the `mappoly2.data` object. The function handles data selection based on the provided chromosome information (`chrom`).

Value

The function does not return a value but generates a plot.

Examples

```
plot(B2721)
```

`plot_consensus_haplo` *Plot Consensus Map with Homolog Probabilities*

Description

This function generates a density plot for homolog probabilities across map positions for a specified individual within a given linkage group. It uses haplotype probability data from a consensus map to create the plot, with different colors representing different parents.

Usage

```
plot_consensus_haplo(x, lg = 1, ind = 1, ...)
```

Arguments

`x`	A list containing consensus map data, specifically `consensus.map`, `ph`, `pedigree`, and `haploprob` structures.
`lg`	An integer specifying the linkage group to plot. Defaults to 1.
`ind`	An identifier for the individual to plot. This can be either a numeric index or a character string corresponding to the individual's name. If it is a character string, the function matches it with the rownames of the pedigree. Defaults to 1.
`...`	Additional arguments to pass on to the plotting function.

Value

A ggplot object representing the density plot of homolog probabilities for the specified individual across map positions.

`plot_genome_vs_map` *Plot Physical vs. Genetic Distance*

Description

This function creates scatterplots to compare physical distance (in Mbp) with genetic distance (in cM). It accepts either a single object or a list of objects of class `mappoly.map`, plotting the relationship for each map provided.

Usage

```
plot_genome_vs_map(
  x,
  type = c("mds", "genome"),
  parent = c("p1p2", "p1", "p2"),
  same.ch.lg = FALSE,
  alpha = 1/2,
  size = 2
)
```

Arguments

`x`	A single object or a list of objects of class `mappoly.map`.
`type`	Character vector indicating the type of genetic map to be used for the analysis. Options include "mds", "genome", or "custom". Default is c("mds", "genome").
`parent`	Character vector specifying the parent or parents to be considered in the analysis. Options are "p1p2" (both parents), "p1" (first parent), or "p2" (second parent). Default is c("p1p2", "p1", "p2").
`same.ch.lg`	Logical; if `TRUE`, only scatterplots for chromosomes and linkage groups with the same number are displayed. Default is `FALSE`.
`alpha`	Numeric; transparency factor for SNP points in the scatterplot. Default is 1/5.
`size`	Numeric; size of the SNP points in the scatterplot. Default is 3.

Details

The function generates scatterplots to visually compare the physical distance (measured in megabase pairs, Mbp) and the genetic distance (measured in centiMorgans, cM) for each map. This helps in understanding the relationship between physical and genetic distances in genetic studies.

Value

A ggplot object representing the scatterplot(s) of physical vs. genetic distances.

plot_haplotypes *Plot Haplotype Probabilities for Genetic Maps*

Description

This function plots haplotype probabilities for specified linkage groups in a genetic mapping dataset. It supports visualization of haplotype probabilities for individual or multiple linkage groups.

Usage

```
plot_haplotypes(
  x,
  lg = NULL,
  ind = 1,
  type = c("mds", "genome"),
  parent = c("p1p2", "p1", "p2")
)
```

Arguments

`x`	An object representing genetic mapping data, typically of a class storing genetic information.
`lg`	Optional vector specifying the linkage group indices to plot. If NULL, the first linkage group is processed.
`ind`	Index or name of the individual for which haplotypes are to be plotted. Should be a single numeric index or a character name.
`type`	Character vector indicating the type of map to process, either "mds" or "genome".
`parent`	Character vector specifying the parent or parents to be considered in the haplotype probability visualization. Options are "p1p2", "p1", and "p2".

Value

A ggplot object representing the plotted haplotype probabilities.

plot_map *Plot Genetic Map*

Description

This function visualizes a genetic map for a specified linkage group. It supports various types of genetic maps and offers customization options for the display.

Usage

```
plot_map(
  x,
  lg = 1,
  type = c("mds", "genome"),
  parent = c("p1p2", "p1", "p2"),
  left.lim = 0,
  right.lim = Inf,
  phase = TRUE,
  mrk.names = FALSE,
  plot.dose = TRUE,
  homolog.names.adj = 3,
  cex = 1,
  xlim = NULL,
  main = "",
  ...
)
```

Arguments

`x`	An object representing genetic mapping data.
`lg`	The linkage group to be visualized, default is the first linkage group (lg = 1).
`type`	The type of genetic map to process, either "mds" or "genome".
`parent`	Specifies which parent's data to use in the visualization. Options are "p1p2" (both parents), "p1" (first parent), or "p2" (second parent).
`left.lim`	The left limit for the plotting area, default is 0.
`right.lim`	The right limit for the plotting area, default is Inf.
`phase`	Logical; if TRUE, phases are included in the plot.
`mrk.names`	Logical; if TRUE, marker names are displayed on the plot.
`plot.dose`	Logical; if TRUE, doses are plotted.
`homolog.names.adj`	Adjustment for homolog names in the plot.
`cex`	Character expansion size for text in the plot.
`xlim`	The limits for the x-axis. Can be set to NULL for automatic adjustment.
`main`	The main title for the plot.
`...`	Additional graphical parameters.

Details

The function creates a detailed plot of a genetic map for a given linkage group. It can display various features such as phases, marker names, and doses, and allows for customization of the plot's appearance.

Value

The function does not return a value but generates a plot of the genetic map.

plot_map_list *Plot a List of Genetic Maps*

Description

This function visualizes a list of genetic maps from a mappoly2 sequence object. It supports both horizontal and vertical orientations and allows for customization of plot colors.

Usage

```
plot_map_list(
  x,
  horiz = TRUE,
  type = c("mds", "genome"),
  parent = c("p1p2", "p1", "p2"),
  col = "lightblue"
)
```

Arguments

x	A mappoly2 sequence object containing genetic map data.
horiz	Logical; if TRUE, the maps are plotted horizontally, otherwise vertically.
type	The type of genetic maps to be plotted, either "mds" or "genome".
parent	Specifies which parent's data to use in the visualization. Options are "p1p2" (both parents), "p1" (first parent), or "p2" (second parent).
col	The color used for plotting the maps, default is "lightgray". Can be a vector of colors to apply different colors to each map.

Details

The function iterates over the linkage groups in the provided mappoly2 sequence object and creates a plot (or a series of plots) showing the genetic map(s) for the specified linkage groups. The plot orientation can be set to either horizontal or vertical, and the color of the plots can be customized.

Value

The function does not return a value but generates a plot or a series of plots representing the genetic maps. It invisibly returns a data frame containing marker positions and linkage group information.

plot_mds *Plot Multi-Dimensional Scaling (MDS) Maps*

Description

This function visualizes the results of Multi-Dimensional Scaling (MDS) analysis on genetic mapping data. It provides options to display either two-dimensional or three-dimensional MDS plots based on the number of dimensions in the MDS object.

Usage

```
plot_mds(
  x,
  lg = 1,
  D1lim = NULL,
  D2lim = NULL,
  D3lim = NULL,
  displaytext = FALSE
)
```

Arguments

`x`	A genetic mapping data object containing MDS information.
`lg`	A numeric value specifying the linkage group to be visualized. Defaults to the first linkage group.
`D1lim`	Optional range for the first dimension of the MDS plot.
`D2lim`	Optional range for the second dimension of the MDS plot.
`D3lim`	Optional range for the third dimension of the MDS plot, applicable only for three-dimensional plots.
`displaytext`	A logical value indicating whether to display text labels on the MDS plot. Defaults to `FALSE`.

Details

Depending on the dimensionality of the MDS object (`ndim`), this function calls either `plot_pcmap` for two-dimensional data or `plot_pcmap3d` for three-dimensional data. It visualizes the spatial arrangement of markers in the specified dimensions, providing insights into the genetic structure.

Value

The function does not return a value but generates a plot of the MDS analysis results for the specified linkage group.

`plot_mds_vs_genome` *Plot MDS vs. Genome*

Description

This function generates a plot comparing MDS (Multi-Dimensional Scaling) positions to genome positions for genetic markers in a `mappoly2.sequence` object. It uses functions from the `mappoly2` package to extract marker data and `ggplot2` for visualization.

Usage

```
plot_mds_vs_genome(x, alpha = 1/2, size = 2)
```

Arguments

`x`	A `mappoly2.sequence` object containing marker and map data. This object typically results from mapping or marker analysis processes in the `mappoly2` package.
`alpha`	The transparency level of the points in the plot, with 1 being fully opaque and 0 being fully transparent. Defaults to 1/2.
`size`	The size of the points in the plot. Defaults to 2.

Value

A ggplot object representing the MDS versus genome position plot. Each linkage group is displayed in a separate panel.

`plot_multi_map` *Plot Multiple Genetic Maps in a Grid Layout*

Description

This function creates a visual representation of genetic maps from different biparental populations. Each panel in the grid corresponds to a linkage group or chromosome. The maps are displayed with the marker positions in centimorgans on the Y axis, and the distinct biparental populations on the X axis. A color gradient from light to dark blue indicates the frequency of markers shared across populations, providing insight into genetic similarities and differences.

Usage

```
plot_multi_map(x)
```

Arguments

`x`	A list of 'mappoly2.sequence' objects representing the genetic data of biparental populations.

Value

A ggplot object representing the genetic maps in a grid layout. Each panel shows a linkage group or chromosome with marker positions and shared frequency across populations.

`plot_rf_matrix` *Plot Recombination Fraction Matrices for Genetic Maps*

Description

This function visualizes the recombination fraction matrices for genetic maps. It allows the user to plot these matrices for different types of genetic data (e.g., MDS, genome, custom) and for specified linkage groups.

Usage

```
plot_rf_matrix(x, lg = NULL, type = c("mds", "genome", "custom"), fact = 1)
```

Arguments

`x`	A `mappoly2.data` object that contains recombination fraction information.
`lg`	Optional vector specifying the linkage groups for which the recombination fraction matrices should be plotted. If NULL, matrices for all linkage groups are plotted.
`type`	The type of genetic data to be visualized. Can be 'mds', 'genome', or 'custom'. This parameter determines how the matrices are processed and displayed.
`fact`	A numeric factor used for scaling or aggregating the matrix data. Defaults to 1 (no scaling).

Details

The function processes the genetic mapping data based on the specified 'type' and 'lg' parameters. It then uses `plot_rf_matrix_one` to plot individual matrices for each linkage group. The visualization helps in understanding the recombination patterns and genetic distances between markers.

Value

The function does not return a value but generates a series of plots, each representing the recombination fraction matrix for a specified linkage group.

`plot_shared_markers` *Plot Shared Markers in Genetic Maps*

Description

This function visualizes shared markers across multiple genetic maps in a 'mappoly2.prepared.integrated.data' object. It uses Euler diagrams to represent the intersection of markers.

Usage

```
plot_shared_markers(x)
```

Arguments

`x`	An object of class 'mappoly2.prepared.integrated.data'. It should contain individual maps from which shared markers are to be identified.

Details

The function extracts markers from each map within the provided 'mappoly2.prepared.integrated.data' object and then plots an Euler diagram to show the intersections (shared markers) among these maps. This visualization helps in understanding the overlap of genetic information across different maps.

Value

An Euler diagram plot showing shared markers across the individual maps in `x`.

`prepare_to_integrate` *Prepare Genetic Map Data for Integration*

Description

This function prepares multiple biparental genetic maps for integration into a unified multi-population genetic map. It primarily utilizes an internal function `match_homologs` to match and rearrange homologs across shared parents in different populations, ensuring consistency in the integrated map estimation. Markers with inconsistencies, such as varying dosages or phasing, are removed. The function currently supports only genome-ordered maps and operates on a single chromosome.

Usage

```
prepare_to_integrate(x, lg = NULL, type = c("genome", "mds"), verbose = TRUE)
```

Arguments

`x`	A list of genetic map objects.
`lg`	Linkage group to be analyzed, default is NULL.
`type`	Type of the map, either "genome" or "mds" (default is "genome").
`verbose`	Logical, indicating whether to show detailed messages (default is TRUE).

Value

A list of prepared data for building an integrated genetic map.

`read_geno_csv` *Read Genetic Marker Data from a CSV File*

Description

This function reads genetic marker data from a comma-separated values (CSV) file and returns an object of class `mappoly2.data`.

Usage

```
read_geno_csv(
  file.in,
  ploidy.p1,
  ploidy.p2 = ploidy.p1,
  name.p1 = NULL,
  name.p2 = NULL,
  filter.non.conforming = TRUE,
  filter.redundant = TRUE,
  verbose = TRUE
)
```

Arguments

`file.in`	A character string specifying the name or full path to the input file.
`ploidy.p1`	The ploidy level of parent 1.
`ploidy.p2`	The ploidy level of parent 2.
`name.p1`	The name of parent 1.
`name.p2`	The name of parent 2.
`filter.non.conforming`	Logical. If `TRUE` (default), data points with unexpected genotypes (e.g., double reduction) are converted to 'NA'. Refer to the `segreg_poly` function for details on expected genotype classes and their frequencies.
`filter.redundant`	Logical. If `TRUE` (default), removes redundant markers during map construction, retaining annotations for export in the final map.
`verbose`	Logical. If `TRUE` (default), displays progress updates; if `FALSE`, no output is provided.

Details

The CSV file should have rows representing markers, with the first row serving as the header. The first seven columns are expected to contain the marker names, the dosages in parents 1 and 2, chromosome information (e.g., chromosome, scaffold, contig), the position of the marker within the sequence, and the alternate and reference alleles, if available. In the absence of allele information, the values should be NA. The remaining columns should contain the dosage for each member of the full-sib population. See the `Examples` section for a tetraploid example.

Value

Returns an object of class `mappoly2.data` containing a list with the following components:

ploidy.p1 Ploidy level of the first parent.

ploidy.p2 Ploidy level of the second parent.

n.ind Number of individuals.

n.mrk Total number of markers.

ind.names Names or identifiers of the individuals.

mrk.names Names or identifiers of the genetic markers.

name.p1 Name or identifier of the first parent.

name.p2 Name or identifier of the second parent.

dosage.p1 Dosage for the first parent.

dosage.p2 Dosage for the second parent.

chrom Chromosome numbers for all markers.

genome.pos Physical positions on the genome for the genetic markers.

ref Reference DNA sequence data for the genetic markers.

alt Alternate DNA sequence data for the genetic markers.

all.mrk.depth Depth of coverage for all genetic markers. NULL when using CSV input files.

geno.dose A matrix containing the dosage for each marker (rows) for each individual (columns).

redundant A list of non-redundant markers and their equivalent redundant markers if `filter.redundant` is TRUE.

QAQC.values A list containing quality assurance and quality control values with the following components:

- **$markers** A data frame with statistics for each marker, including `miss` (missing data rate), `chisq.pval` (chi-squared test p-value), and `read.depth` (read depth).
- **$individuals** A data frame with statistics for each individual, including `miss` (missing data rate) and `full.sib` (indicator of non-belonging to the analyzed bi-parental cross, generated by `filter_individuals`).

Author(s)

Marcelo Mollinari, <`mmollin@ncsu.edu`>

Examples

```
tempfl <- list.files(system.file('extdata', package = 'mappoly2'),
                     full.names = TRUE)
alfalfa.bc <- read_geno_csv(file.in = tempfl,
                            ploidy.p1 = 4,
                            name.p1 = "I195",
                            name.p2 = "F1.85.209")
print(alfalfa.bc, detailed = TRUE)
plot(alfalfa.bc)
```

refine_map *Refine Genetic Map by Removing Problematic Markers*

Description

Identifies and removes markers from a genetic map that contribute to significant gaps or inaccuracies, potentially due to misplaced markers, incorrect phasing, or anomalous behavior.

Usage

```
refine_map(
  x,
  lg = NULL,
  type = c("mds", "genome"),
  parent = c("p1p2", "p1", "p2"),
  gap.threshold = 5,
  size.rem.cluster = 1,
  ncpus = 1,
  reestimate.hmm.map = TRUE,
  recompute.haplotype.prob = TRUE,
  verbose = TRUE,
  tol = 0.001,
  error = NULL
)
```

Arguments

`x`	A genetic mapping object, typically of a class "mappoly2.sequence".
`lg`	Optional vector specifying the linkage group indices to process. If NULL, all linkage groups in `x` are processed.
`type`	Character vector indicating the type of map to process, either "mds" or "genome".
`parent`	Character vector specifying the parent or parents to be considered in the marker removal process. Options are "p1p2" (both parents), "p1" (first parent), and "p2" (second parent).
`gap.threshold`	The threshold for identifying significant gaps in the map. Markers causing gaps larger than this threshold will be considered for removal.
`size.rem.cluster`	Minimum number of consecutive markers to retain. Segments with fewer markers than this threshold will be removed.
`ncpus`	The number of CPU cores to use for parallel processing. Defaults to 1.
`reestimate.hmm.map`	If TRUE, re-estimates the hidden Markov model for the map after marker removal.
`recompute.haplotype.prob`	If TRUE, recalculates haplotype probabilities after marker removal.
`verbose`	Logical; if `TRUE`, function prints messages during execution (default is `TRUE`).
`tol`	Tolerance level for the mapping algorithm.
`error`	Optional; error rate to be used in the hidden Markov model estimation. If NULL, it uses the error rate from the input object.

Details

The function analyzes the genetic map to identify markers causing gaps larger than the specified threshold. It then removes these markers and, optionally, re-estimates the hidden Markov model and recalculates haplotype probabilities to refine the map.

Value

An updated genetic mapping object with problematic markers removed and, if specified, re-estimated HMM and recalculated haplotype probabilities.

`rev_map` *Reverse the Order of a Genetic Map*

Description

This function reverses the order of a genetic map in a `mappoly` object. It is typically used to align the MDS order with the genomic order.

Usage

```
rev_map(
  x,
  lg,
  type = c("mds", "genome", "custom"),
  parent = c("p1p2", "p1", "p2")
)
```

Arguments

`x`	A `mappoly` object containing the genetic map(s) to be reversed.
`lg`	An integer or vector specifying the linkage group(s) to be reversed.
`type`	Character vector indicating the type of genetic map to be reversed. Options include "mds", "genome", or "custom". Default is c("mds", "genome", "custom").
`parent`	A character vector specifying the parent or parents to be considered in the mapping process. Options are "p1p2" (both parents), "p1" (first parent), and "p2" (second parent). Default is c("p1p2", "p1", "p2").

Details

The function modifies the order of markers in the specified linkage group(s) of the genetic map. It reverses the positions of markers along with associated data such as recombination fractions and haplotype probabilities, ensuring that the genetic map's orientation is consistent with genomic data.

Value

Returns the `mappoly` object with the order of the specified linkage group(s) reversed.

rf_filter	*Remove Markers Not Meeting LOD and Recombination Fraction Criteria*

Description

This function removes markers from a `mappoly2.data` or `mappoly2.sequence` object that do not meet specified LOD (logarithm of odds) and recombination fraction criteria. It is designed to filter out markers that are unlikely to be linked or show strong evidence of linkage across an entire linkage group, which might indicate false positives.

Usage

```
rf_filter(
  x,
  thresh.LOD.ph = 5,
  thresh.LOD.rf = 5,
  thresh.rf = 0.15,
  probs = c(0.05, 1),
  lg = NULL,
  type = c("mds", "genome"),
  diag.markers = NULL,
  mrk.order = NULL,
  diagnostic.plot = TRUE,
  breaks = 100
)
```

Arguments

`x`	An object of class `mappoly2.data` or `mappoly2.sequence`.
`thresh.LOD.ph`	LOD score threshold for linkage phase configuration. Typically set to eliminate markers that are unlinked to the analyzed set (default = 5).
`thresh.LOD.rf`	LOD score threshold for recombination fraction (default = 5).
`thresh.rf`	Recombination fraction threshold (default = 0.15).
`probs`	A numeric vector indicating the probability corresponding to the filtering quantiles (default = c(0.05, 1)).
`lg`	A vector of linkage groups to be processed. If NULL (default), all groups are considered.
`type`	A character vector specifying the method for linkage group analysis. Options are "mds" for multidimensional scaling and "genome" for genomic analysis. This parameter only has an effect if `lg` is not NULL.
`diag.markers`	A vector specifying a window of marker pairs to consider. If NULL (default), all markers are considered.
`mrk.order`	Marker order vector. This parameter is only used if `diag.markers` is not NULL.
`diagnostic.plot`	Logical; if `TRUE`, generates a diagnostic plot (default is TRUE).
`breaks`	The number of cells for the histogram in the diagnostic plot (default = 100).

Details

The function first checks the type of the input object and applies the relevant filtering criteria based on LOD scores and recombination fractions. It optionally produces a diagnostic plot to visualize the filtering process.

Value

A filtered object of the same class as the input (`mappoly2.data` or `mappoly2.sequence`).

Author(s)

Marcelo Mollinari, <`mmollin@ncsu.edu`> with updates by Gabriel Gesteira, <`gdesiqu@ncsu.edu`>

Examples

```
## Not run:
  # Assuming `my_data` is a valid mappoly2.data or mappoly2.sequence object
  filtered_data <- rf_filter(my_data, thresh.LOD.ph = 5, thresh.LOD.rf = 5)

## End(Not run)
```

`subset.mappoly2.data` *Subset mappoly2.data Object*

Description

This function creates a subset of either individuals or markers from an object of the `mappoly2.data` class, based on specified criteria.

Usage

```
## S3 method for class 'mappoly2.data'
subset(
  x,
  type = c("marker", "individual"),
  perc = 0.1,
  n = NULL,
  select.mrk = NULL,
  select.ind = NULL,
  seed = NULL,
  filter.non.conforming = TRUE,
  filter.redundant = TRUE,
  ...
)
```

Arguments

`x`	A `mappoly2.data` object from which the subset is to be selected.
`type`	Character string, either "marker" or "individual", indicating whether markers or individuals should be subset. Default is "marker".

`perc`	Numeric; the proportion of individuals or markers to be sampled. This is used only if `n` is NULL. Default is 0.1.
`n`	Integer; the number of individuals or markers to be sampled. If NULL, `perc` must be specified.
`select.mrk`	Character vector containing the names of the markers to select. Effective only if `type = "marker"`, and both `n` and `perc` are NULL.
`select.ind`	Character vector containing the names of the individuals to select. Effective only if `type = "individual"`, and both `n` and `perc` are NULL.
`seed`	Integer or NULL; sets the seed for random number generation for reproducible sampling.
`filter.non.conforming`	Logical; if TRUE (default), data points with unexpected genotypes (e.g., double reduction) are converted to 'NA'.
`filter.redundant`	Logical; if TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.
`...`	Additional arguments, currently not used.

Details

The function allows for flexible subsetting of `mappoly2.data` objects based on the number or proportion of individuals or markers. It supports random sampling and specific selection based on provided lists. Additional filtering options are available to handle non-conforming data and redundant markers.

Value

A `mappoly2.data` object that contains the selected subset of individuals or markers.

Index