Package ‘mappoly’

January 5, 2023

Type Package
Title Genetic Linkage Maps in Autopolyploids
Version 0.3.3
Maintainer Marcelo Mollinari <mmollin@ncsu.edu>
Description Construction of genetic maps in autopolyploid full-sib populations. Uses pairwise recombination fraction estimation as the first source of information to sequentially position allelic variants in specific homologous chromosomes. For situations where pairwise analysis has limited power, the algorithm relies on the multilocus likelihood obtained through a hidden Markov model (HMM). For more detail, please see Mollinari and Garcia (2019) <doi:10.1534/g3.119.400378> and Mollinari et al. (2020) <doi:10.1534/g3.119.400620>.
License GPL-3
LazyData TRUE
LazyDataCompression xz
Depends R (>= 4.0.0)
Imports Rcpp (>= 0.12.6), RcppParallel, RCurl, fields, ggpdb, ggsci, rstudapi, plot3D, dplyr, crayon, cli, magrittr, reshape2, ggplot2, smacof, princurve, dendextend, vcfR, zoo, plotly
LinkingTo Rcpp, RcppParallel
RoxygenNote 7.2.3
SystemRequirements C++11, GNU make
Encoding UTF-8
Suggests testthat, updog, fitPoly, polymapR, AGHmatrix, gatepoints, knitr, rmarkdown
URL https://github.com/mmollina/MAPpoly
BugReports https://github.com/mmollina/MAPpoly/issues
VignetteBuilder knitr
NeedsCompilation yes
Author  Marcelo Mollinari [aut, cre] (<https://orcid.org/0000-0002-7001-8498>),
Gabriel Gesteira [aut] (<https://orcid.org/0000-0002-4106-7346>),
Guilhereme Pereira [ctb] (<https://orcid.org/0000-0002-7106-8638>),
Augusto Garcia [ctb] (<https://orcid.org/0000-0003-0634-3277>),
Zhao-Bang Zeng [ctb] (<https://orcid.org/0000-0002-3115-1149>),
Katharine Preedy [ctb, cph] (MDS ordering algorithm),
Robert Gentleman [cph] (C code for MLE optimization in
src/pairwise_estimation.cpp),
Ross Ihaka [cph] (C code for MLE optimization in
src/pairwise_estimation.cpp),
R Foundation [cph] (C code for MLE optimization in
src/pairwise_estimation.cpp),
R-core [cph] (C code for MLE optimization in
src/pairwise_estimation.cpp)

Repository  CRAN
Date/Publication  2023-01-05 20:10:02 UTC

R topics documented:

add_marker . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
cache_counts_twopt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
calc_genoprob . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 8
calc_genoprob_dist . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 9
calc_genoprob_error . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10
calc_genoprob_one_parent . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
calc_homologprob . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 13
calc_prefpair_profiles . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 14
check_data_sanity . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
compare_maps . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 15
drop_marker . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 16
elim_redundant . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 17
est_full_hmm_with_global_error . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 18
est_full_hmm_with_prior_prob . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 20
est_pairwise_rf . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 22
est_pairwise_rf2 . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 24
est_rf_hmm . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 25
est_rf_hmm_sequential . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 28
export_data_to_polymapR . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 31
export_map_list . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 32
export_qtlpoly . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 33
extract_map . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
filter_individuals . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 34
filter_missing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 35
filter_segregation . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 36
find_blocks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 37
get_genomic_order . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
get_submap . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 39
**add_marker**

Add a single marker to a map

**Description**

Creates a new map by adding a marker in a given position in a pre-built map.

**Usage**

```r
code
add_marker(
  input.map,
  mrk,
  pos,
  rf.matrix,
  genoprob = NULL,
  phase.config = "best",
  tol = 0.001,
  r.test = NULL,
  verbose = TRUE
)
```

**Arguments**

- **input.map**: an object of class `mappoly.map`
- **mrk**: the name of the marker to be inserted
- **pos**: the name of the marker after which the new marker should be added. One also can inform the numeric position (between markers) were the new marker should be added. To insert a marker at the beginning of a map, use `pos = 0`
- **rf.matrix**: an object of class `mappoly.rf.matrix` containing the recombination fractions and the number of homologues sharing alleles between pairwise markers on `input.map`. It is important that `shared.alleles = TRUE` in function `rf_list_to_matrix` when computing `rf.matrix`.
- **genoprob**: an object of class `mappoly.genoprob` containing the genotype probabilities for all marker positions on `input.map`
- **phase.config**: which phase configuration should be used. "best" (default) will choose the maximum likelihood configuration
- **tol**: the desired accuracy (default = `10e-04`)
- **r.test**: for internal use only
- **verbose**: if `TRUE` (default), the current progress is shown; if `FALSE`, no output is produced

**Details**

`add_marker` splits the input map into two sub-maps to the left and the right of the given position. Using the genotype probabilities, it computes the log-likelihood of all possible linkage phases under a two-point threshold inherited from function `rf_list_to_matrix`. 
Value

A list of class mappoly.map with two elements:

i) info: a list containing information about the map, regardless of the linkage phase configuration:

- **ploidy**: the ploidy level
- **n.mrk**: number of markers
- **seq.num**: a vector containing the (ordered) indices of markers in the map, according to the input file
- **mrk.names**: the names of markers in the map
- **seq.dose.p1**: a vector containing the dosage in parent 1 for all markers in the map
- **seq.dose.p2**: a vector containing the dosage in parent 2 for all markers in the map
- **chrom**: a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, chrom = NULL
- **genome.pos**: physical position (usually in megabase) of the markers into the sequence
- **seq.ref**: reference base used for each marker (i.e. A, T, C, G). If not available, seq.ref = NULL
- **seq.alt**: alternative base used for each marker (i.e. A, T, C, G). If not available, seq.alt = NULL
- **chisq.pval**: a vector containing p-values of the chi-squared test of Mendelian segregation for all markers in the map
- **data.name**: name of the dataset of class mappoly.data
- **ph.thres**: the LOD threshold used to define the linkage phase configurations to test

ii) a list of maps with possible linkage phase configuration. Each map in the list is also a list containing

- **seq.num**: a vector containing the (ordered) indices of markers in the map, according to the input file
- **seq.rf**: a vector of size (n.mrk - 1) containing a sequence of recombination fraction between the adjacent markers in the map
- **seq.ph**: linkage phase configuration for all markers in both parents
- **loglike**: the hmm-based multipoint likelihood

Author(s)

Marcelo Mollinari, mmollin@ncsu.edu

Examples

```r
sub.map <- get_submap(maps.hexafake[[1]], 1:20, reestimate.rf = FALSE)
plot(sub.map, mrk.names = TRUE)
s <- make_seq_mappoly(hexafake, sub.map$info$mrk.names)
tpt <- est_pairwise_rf(s)
rf.matrix <- rf_list_to_matrix(input.twopt = tpt,
```
thresh.LOD.ph = 3,
thresh.LOD.rf = 3,
shared.alleles = TRUE)

###### Removing marker "M_1" (first) #######

mrk.to.remove <- "M_1"
input.map <- drop_marker(sub.map, mrk.to.remove)
plot(input.map, mrk.names = TRUE)
## Computing conditional probabilities using the resulting map
genoprob <- calc_genoprob(input.map)
res.add.M_1 <- add_marker(input.map = input.map,
                        mrk = "M_1",
pos = 0,
rf.matrix = rf.matrix,
genoprob = genoprob,
tol = 10e-4)
plot(res.add.M_1, mrk.names = TRUE)

names.id <- names(best.phase$P)
plot_compare_haplotypes(ploidy = 6,
hom.allele.p1 = best.phase$P[names.id],
hom.allele.q1 = best.phase$Q[names.id],
hom.allele.p2 = sub.map$maps[[1]]$seq.ph$P[names.id],
hom.allele.q2 = sub.map$maps[[1]]$seq.ph$Q[names.id])


###### Removing marker "M_10" (middle or last) #######

mrk.to.remove <- "M_10"
input.map <- drop_marker(sub.map, mrk.to.remove)
plot(input.map, mrk.names = TRUE)
## Computing conditional probabilities using the resulting map
genoprob <- calc_genoprob(input.map)
res.add.M_10 <- add_marker(input.map = input.map,
                        mrk = "M_10",
pos = "M_9",
rf.matrix = rf.matrix,
genoprob = genoprob,
tol = 10e-4)
plot(res.add.M_10, mrk.names = TRUE)

names.id <- names(best.phase$P)
plot_compare_haplotypes(ploidy = 6,
hom.allele.p1 = best.phase$P[names.id],
hom.allele.q1 = best.phase$Q[names.id],
hom.allele.p2 = sub.map$maps[[1]]$seq.ph$P[names.id],
hom.allele.q2 = sub.map$maps[[1]]$seq.ph$Q[names.id])

---

```
cache_counts_twopt

Frequency of genotypes for two-point recombination fraction estimation
```
Description

Returns the frequency of each genotype for two-point reduction of dimensionality. The frequency is calculated for all pairwise combinations and for all possible linkage phase configurations.

Usage

```r
cache_counts_twopt(
  input.seq,
  cached = FALSE,
  cache.prev = NULL,
  ncpus = 1L,
  verbose = TRUE,
  joint.prob = FALSE
)
```

Arguments

- `input.seq`: an object of class `mappoly.sequence`
- `cached`: If `TRUE`, access the counts for all linkage phase configurations in an internal file (default = `FALSE`)
- `cache.prev`: an object of class `cache.info` containing pre-computed genotype frequencies, obtained with `cache_counts_twopt` (optional, default = `NULL`)
- `ncpus`: Number of parallel processes to spawn (default = 1)
- `verbose`: If `TRUE` (default), print the linkage phase configurations. If `cached = TRUE`, nothing is printed, since all linkage phase configurations will be cached.
- `joint.prob`: If `FALSE` (default), returns the frequency of genotypes for transition probabilities (conditional probabilities). If `TRUE` returns the frequency for joint probabilities. The latter is especially important to compute the Fisher’s Information for a pair of markers.

Value

An object of class `cache.info` which contains one (conditional probabilities) or two (both conditional and joint probabilities) lists. Each list contains all pairs of dosages between parents for all markers in the sequence. The names in each list are of the form ‘A-B-C-D’, where: A represents the dosage in parent 1, marker k; B represents the dosage in parent 1, marker k+1; C represents the dosage in parent 2, marker k; and D represents the dosage in parent 2, marker k+1. For each list, the frequencies were computed for all possible linkage phase configurations. The frequencies for each linkage phase configuration are distributed in matrices whose names represents the number of homologous chromosomes that share alleles. The rows on these matrices represents the dosages in markers k and k+1 for an individual in the offspring. See Table 3 of S3 Appendix in Mollinari and Garcia (2019) for an example.

Author(s)

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

Examples
```r
all.mrk <- make_seq_mappoly(tetra.solcap, 1:20)
## local computation
counts <- cache_counts_twopt(all.mrk, ncpus = 1)
## load from internal file or web-stored counts (especially important for high ploidy levels)
counts.cached <- cache_counts_twopt(all.mrk, cached = TRUE)
```

**calc_genoprob**

*Compute conditional probabilities of the genotypes*

**Description**
Conditional genotype probabilities are calculated for each marker position and each individual given a map.

**Usage**
```
calc_genoprob(input.map, step = 0, phase.config = "best", verbose = TRUE)
```

**Arguments**
- `input.map`: An object of class `mappoly.map`
- `step`: Maximum distance (in cM) between positions at which the genotype probabilities are calculated, though for `step = 0`, probabilities are calculated only at the marker locations.
- `phase.config`: which phase configuration should be used. "best" (default) will choose the phase configuration associated with the maximum likelihood
- `verbose`: if TRUE (default), current progress is shown; if FALSE, no output is produced

**Value**
An object of class `mappoly.genoprob` which has two elements: a tridimensional array containing the probabilities of all possible genotypes for each individual in each marker position; and the marker sequence with it’s recombination frequencies

**Author(s)**
Marcelo Mollinari, <mmollin@ncsu.edu>
References


Examples

```r
## tetraploid example
probs.t <- calc_genoprob(input.map = solcap.dose.map[[1]],
                         verbose = TRUE)
probs.t
## displaying individual 1, 36 genotypic states
## (rows) across linkage group 1 (columns)
image(t(probs.t$probs[,1]))
```

### Description

Conditional genotype probabilities are calculated for each marker position and each individual given a map. In this function, the probabilities are not calculated between markers.

### Usage

```r
calc_genoprob_dist(
  input.map, 
  dat.prob = NULL, 
  phase.config = "best", 
  verbose = TRUE
)
```

### Arguments

- **input.map**: An object of class `mappoly.map`
- **dat.prob**: an object of class `mappoly.data` containing the probability distribution of the genotypes
- **phase.config**: which phase configuration should be used. "best" (default) will choose the phase configuration with the maximum likelihood
- **verbose**: if TRUE (default), the current progress is shown; if FALSE, no output is produced

### Value

An object of class `mappoly.genoprob` which has two elements: a tridimensional array containing the probabilities of all possible genotypes for each individual in each marker position; and the marker sequence with it’s recombination frequencies.
Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

```r
## tetraploid example
probs.t <- calc_genoprob_dist(input.map = solcap.prior.map[[1]],
                               dat.prob = tetra.solcap.geno.dist,
                               verbose = TRUE)
probs.t
```

```
## displaying individual 1, 36 genotypic states
## (rows) across linkage group 1 (columns)
image(t(probs.t$probs[,1]))
```
**calc_genoprob_one_parent**

Compute conditional probabilities of the genotype (one informative parent)

**Description**

Conditional genotype probabilities are calculated for each marker position and each individual given a map

**Value**

An object of class `mappoly.genoprob` which has two elements: a tridimensional array containing the probabilities of all possible genotypes for each individual in each marker position; and the marker sequence with its recombination frequencies

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**References**


**Examples**

```r
probs.error <- calc_genoprob_error(input.map = solcap.err.map[[1]],
                                   error = 0.05,
                                   verbose = TRUE)
```
Usage

calc_genoprob_one_parent(
  input.map,
  step = 0,
  info.parent = 1,
  uninfo.parent = 2,
  global.err = 0,
  phase.config = "best",
  verbose = TRUE
)

Arguments

input.map An object of class mappoly.map (with exceptions)
step Maximum distance (in cM) between positions at which the genotype probabilities are calculated, though for step = 0, probabilities are calculated only at the marker locations.
info.parent index for informative parent
uninfo.parent index for uninformative parent
global.err the assumed global error rate (default = 0.0)
phase.config which phase configuration should be used. "best" (default) will choose the phase configuration associated with the maximum likelihood
verbose if TRUE (default), current progress is shown; if FALSE, no output is produced

Value

An object of class 'mappoly.genoprob' which has two elements: a tridimensional array containing the probabilities of all possible genotypes for each individual in each marker position; and the marker sequence with it’s recombination frequencies

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

## tetraploid example
map <- solcap.dose.map[[1]]
s <- make_seq_mappoly(map)
map1 <- est_rf_hmm_single_one_parent(input.seq = s,
  input.ph.single = map$maps[[1]]$seq.ph,
  info.parent = 1,
calc_homologprob

```r
plot(map1)
probs <- calc_genoprob_one_parent(input.map = map1,
       info.parent = 1,
       uninfo.parent = 2,
       step = 1)
probs
## displaying individual 1, 6 genotypic states
## (rows) across linkage group 1 (columns)
image(t(probs$probs[,2]))
```

---

**calc_homologprob**

**Homolog probabilities**

**Description**

Compute homolog probabilities for all individuals in the full-sib population given a map and conditional genotype probabilities.

**Usage**

```r
calc_homologprob(input.genoprobs, verbose = TRUE)
```

**Arguments**

- `input.genoprobs`:
  - an object of class `mappoly.genoprob`
- `verbose`:
  - if `TRUE` (default), the current progress is shown; if `FALSE`, no output is produced

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**References**


**Examples**

```r
## tetraploid example
w1 <- calc_genoprob(solcap.dose.map[[1]])
h.prob <- calc_homologprob(w1)
print(h.prob)
plot(h.prob, ind = 5, use.plotly = FALSE)
```
calc_prefpair_profiles

Preferential pairing profiles

Description

Given the genotype conditional probabilities for a map, this function computes the probability profiles for all possible homolog pairing configurations in both parents.

Usage

calc_prefpair_profiles(input.genoprobs, verbose = TRUE)

Arguments

input.genoprobs

an object of class mappoly.genoprob

verbose

if TRUE (default), the current progress is shown; if FALSE, no output is produced

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu> and Guilherme Pereira, <g.pereira@cgiar.org>

References


Examples

## tetraploid example
w1 <- lapply(solcap.dose.map[1:12], calc_genoprob)
x1 <- calc_prefpair_profiles(w1)
print(x1)
plot(x1)
check_data_sanity

Data sanity check

Description
Checks the consistency of a dataset

Usage
check_data_sanity(x)

Arguments
x an object of class mappoly.data

Value
if consistent, returns 0. If not consistent, returns a vector with a number of tests, where TRUE indicates a failed test.

Author(s)
Marcelo Mollinari, <mmollin@ncsu.edu>

References

Examples
check_data_sanity(tetra.solcap)

compare_maps

Compare a list of maps

Description
Compare lengths, density, maximum gaps and log likelihoods in a list of maps. In order to make the maps comparable, the function uses the intersection of markers among maps.

Usage
compare_maps(...)


Arguments

... a list of objects of class mappoly.map

Value

A data frame where the lines correspond to the maps in the order provided in input list list

Description

This function creates a new map by removing markers from an existing one.

Usage

drop_marker(input.map, mrk, verbose = TRUE)

Arguments

input.map an object of class mappoly.map
mrk a vector containing markers to be removed from the input map, identified by their names or positions
verbose if TRUE (default), the current progress is shown; if FALSE, no output is produced

Value

an object of class mappoly.map

Author(s)

Marcelo Mollinari,<mmollin@ncsu.edu>

Examples

sub.map <- get_submap(maps.hexafake[[1]], 1:50, reestimate.rf = FALSE)
plot(sub.map, mrk.names = TRUE)
mrk.to.remove <- c("M_1", "M_23", "M_34")
red.map <- drop_marker(sub.map, mrk.to.remove)
plot(red.map, mrk.names = TRUE)
elim_redundant

Eliminate redundant markers

Description

Eliminate markers with identical dosage information for all individuals.

Usage

elim_redundant(input.seq, data = NULL)

Arguments

input.seq  an object of class mappoly.sequence
data       name of the dataset that contains sequence markers (optional, default = NULL)

Value

An object of class mappoly.unique.seq which is a list containing the following components:

unique.seq    an object of class mappoly.sequence with the redundant markers removed
kept          a vector containing the name of the informative markers
eliminated    a vector containing the name of the non-informative (eliminated) markers

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>, with minor modifications by Gabriel Gesteira, <gdesiqu@ncsu.edu>

References


Examples

```r
all.mrk <- make_seq_mappoly(hexafake, 'all')
red.mrk <- elim_redundant(all.mrk)
plot(red.mrk)
unique.mrks <- make_seq_mappoly(red.mrk)
```
est_full_hmm_with_global_error

Re-estimate genetic map given a global genotyping error

Description

This function considers a global error when re-estimating a genetic map using Hidden Markov models. Since this function uses the whole transition space in the HMM, its computation can take a while, especially for hexaploid maps.

Usage

```r
est_full_hmm_with_global_error(
  input.map,  
  error = NULL,  
  tol = 0.001,  
  restricted = TRUE,  
  th.prob = 0.95,  
  verbose = FALSE
)
```

Arguments

- `input.map`: an object of class `mappoly.map`
- `error`: the assumed global error rate (default = NULL)
- `tol`: the desired accuracy (default = 10e-04)
- `restricted`: if TRUE (default), restricts the prior to the possible classes under Mendelian, non double-reduced segregation given dosage of the parents
- `th.prob`: the threshold for using global error or genotype probability distribution if present in the dataset (default = 0.95)
- `verbose`: if TRUE, current progress is shown; if FALSE (default), no output is produced

Value

A list of class `mappoly.map` with two elements:

i) info: a list containing information about the map, regardless of the linkage phase configuration:

- `ploidy`: the ploidy level
- `n.mrk`: number of markers
- `seq.num`: a vector containing the (ordered) indices of markers in the map, according to the input file
- `mrk.names`: the names of markers in the map
- `seq.dose.p1`: a vector containing the dosage in parent 1 for all markers in the map
- `seq.dose.p2`: a vector containing the dosage in parent 2 for all markers in the map
est_full_hmm_with_global_error

chrom a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, chrom = NULL

genome.pos physical position (usually in megabase) of the markers into the sequence

seq.ref reference base used for each marker (i.e. A, T, C, G). If not available, seq.ref = NULL

seq.alt alternative base used for each marker (i.e. A, T, C, G). If not available, seq.ref = NULL

chisq.pval a vector containing p-values of the chi-squared test of Mendelian segregation for all markers in the map

data.name name of the dataset of class mappoly.data

ph.thres the LOD threshold used to define the linkage phase configurations to test

ii) a list of maps with possible linkage phase configuration. Each map in the list is also a list containing

seq.num a vector containing the (ordered) indices of markers in the map, according to the input file

seq.rf a vector of size (n.mrk - 1) containing a sequence of recombination fraction between the adjacent markers in the map

seq.ph linkage phase configuration for all markers in both parents

loglike the hmm-based multipoint likelihood

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

submap <- get_submap(solcap.dose.map[[1]], mrk.pos = 1:20, verbose = FALSE)
err.submap <- est_full_hmm_with_global_error(submap,
error = 0.01,
tol = 10e-4,
verbose = TRUE)
err.submap
plot_map_list(list(dose = submap, err = err.submap),
title = "estimation procedure")
est_full_hmm_with_prior_prob

Re-estimate genetic map using dosage prior probability distribution

Description

This function considers dosage prior distribution when re-estimating a genetic map using Hidden Markov models

Usage

```r
est_full_hmm_with_prior_prob(
  input.map,
  dat.prob = NULL,
  phase.config = "best",
  tol = 0.001,
  verbose = FALSE
)
```

Arguments

- `input.map`: an object of class `mappoly.map`
- `dat.prob`: an object of class `mappoly.data` containing the probability distribution of the genotypes
- `phase.config`: which phase configuration should be used. "best" (default) will choose the maximum likelihood configuration
- `tol`: the desired accuracy (default = 10e-04)
- `verbose`: if TRUE, current progress is shown; if FALSE (default), no output is produced

Value

A list of class `mappoly.map` with two elements:

- `info`: a list containing information about the map, regardless of the linkage phase configuration:
  - `ploidy`: the ploidy level
  - `n.mrk`: number of markers
  - `seq.num`: a vector containing the (ordered) indices of markers in the map, according to the input file
  - `mrk.names`: the names of markers in the map
  - `seq.dose.p1`: a vector containing the dosage in parent 1 for all markers in the map
  - `seq.dose.p2`: a vector containing the dosage in parent 2 for all markers in the map
  - `chrom`: a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, `chrom = NULL`
  - `genome.pos`: physical position (usually in megabase) of the markers into the sequence
The function `est_full_hmm_with_prior_prob` takes the following arguments:

- `seq.ref`: reference base used for each marker (i.e. A, T, C, G). If not available, `seq.ref = NULL`.
- `seq.alt`: alternative base used for each marker (i.e. A, T, C, G). If not available, `seq.ref = NULL`.
- `chisq.pval`: a vector containing p-values of the chi-squared test of Mendelian segregation for all markers in the map.
- `data.name`: name of the dataset of class `mappoly.data`.
- `ph.thresh`: the LOD threshold used to define the linkage phase configurations to test.

II) A list of maps with possible linkage phase configuration. Each map in the list is also a list containing:

- `seq.num`: a vector containing the (ordered) indices of markers in the map, according to the input file.
- `seq.rf`: a vector of size `(n.mrk - 1)` containing a sequence of recombination fraction between the adjacent markers in the map.
- `seq.ph`: linkage phase configuration for all markers in both parents.
- `loglike`: the hmm-based multipoint likelihood.

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**References**


**Examples**

```r
submap <- get_submap(solcap.dose.map[[1]], mrk.pos = 1:20, verbose = FALSE)
prob.submap <- est_full_hmm_with_prior_prob(submap, 
                                          dat.prob = tetra.solcap.geno.dist, 
                                          tol = 1e-4, 
                                          verbose = TRUE)

prob.submap
plot_map_list(list(dose = submap, prob = prob.submap), 
              title = "estimation procedure")
```
est_pairwise_rf  
Pairwise two-point analysis

Description

Performs the two-point pairwise analysis between all markers in a sequence. For each pair, the function estimates the recombination fraction for all possible linkage phase configurations and associated LOD Scores.

Usage

```r
est_pairwise_rf(  
  input.seq,  
  count.cache = NULL,  
  count.matrix = NULL,  
  ncpus = 1L,  
  mrk.pairs = NULL,  
  n.batches = 1L,  
  est.type = c("disc", "prob"),  
  verbose = TRUE,  
  memory.warning = TRUE,  
  parallelization.type = c("PSOCK", "FORK"),  
  tol = .Machine$double.eps^0.25,  
  ll = FALSE  
)
```

Arguments

- `input.seq`: an object of class `mappoly.sequence`
- `count.cache`: an object of class `cache.info` containing pre-computed genotype frequencies, obtained with `cache_counts_twopt`. If NULL (default), genotype frequencies are internally loaded.
- `count.matrix`: similar to `count.cache`, but in matrix format. Mostly for internal use.
- `ncpus`: Number of parallel processes (cores) to spawn (default = 1)
- `mrk.pairs`: a matrix of dimensions 2*N, containing N pairs of markers to be analyzed. If NULL (default), all pairs are considered
- `n.batches`: deprecated. Not available on MAPpoly 0.3.0 or higher
- `est.type`: Indicates whether to use the discrete ("disc") or the probabilistic ("prob") dosage scoring when estimating the two-point recombination fractions.
- `verbose`: If TRUE (default), current progress is shown; if FALSE, no output is produced
- `memory.warning`: if TRUE, prints a memory warning if the number of markers is greater than 10000 for ploidy levels up to 4, and 3000 for ploidy levels > 4.
- `parallelization.type`: one of the supported cluster types. This should be either PSOCK (default) or FORK.
est_pairwise_rf

tol the desired accuracy. See optimize() for details
11 will return log-likelihood instead of LOD scores. (for internal use)

Value

An object of class mappoly.twopt which is a list containing the following components:

- **data.name**: name of the object of class mappoly.data with the raw data
- **n.mrk**: number of markers in the sequence
- **seq.num**: a vector containing the (ordered) indices of markers in the sequence, according to the input file
- **pairwise**: a list of size \( \text{choose}(\text{length(input.seq$seq.num)}, 2) \), each of them containing a matrix where the name of the rows have the form x-y, where x and y indicate how many homologues share the same allelic variant in parents P and Q, respectively (see Mollinari and Garcia, 2019 for notation). The first column indicates the LOD Score in relation to the most likely linkage phase configuration. The second column shows the estimated recombination fraction for each configuration, and the third indicates the LOD Score comparing the likelihood under no linkage \((r = 0.5)\) with the estimated recombination fraction (evidence of linkage).

- **chisq.pval.thres**: threshold used to perform the segregation tests
- **chisq.pval**: p-values associated with the performed segregation tests

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

```r
## Tetraploid example (first 50 markers)
all.mrk <- make_seq_mappoly(tetra.solcap, 1:50)
red.mrk <- elim_redundant(all.mrk)
unique.mrks <- make_seq_mappoly(red.mrk)
all.pairs <- est_pairwise_rf(input.seq = unique.mrks,
                          ncpus = 1,
                          verbose = TRUE)
all.pairs
plot(all.pairs, 20, 21)
mat <- rf_list_to_matrix(all.pairs)
plot(mat)
```
est_pairwise_rf2  

Pairwise two-point analysis - RcppParallel version

Description
Performs the two-point pairwise analysis between all markers in a sequence. For each pair, the function estimates the recombination fraction for all possible linkage phase configurations and associated LOD Scores.

Usage

```r
est_pairwise_rf2(
  input.seq,
  ncpus = 1L,
  mrk.pairs = NULL,
  verbose = TRUE,
  tol = .Machine$double.eps^0.25
)
```

Arguments

- `input.seq`: an object of class `mappoly.sequence`
- `ncpus`: Number of parallel processes (cores) to spawn (default = 1)
- `mrk.pairs`: a matrix of dimensions 2*N, containing N pairs of markers to be analyzed. If `NULL` (default), all pairs are considered
- `verbose`: If `TRUE` (default), current progress is shown; if `FALSE`, no output is produced
- `tol`: the desired accuracy. See `optimize()` for details

Details
Differently from `est_pairwise_rf` this function returns only the values associated to the best linkage phase configuration.

Value
An object of class `mappoly.twopt2`

Author(s)
Marcelo Mollinari, `<mmollin@ncsu.edu>`

References
### Examples

```r
## Tetraploid example
all.mrk <- make_seq_mappoly(tetra.solcap, 100:200)
all.pairs <- est_pairwise_rf2(input.seq = all.mrk, ncpus = 2)
m <- rf_list_to_matrix(all.pairs)
plot(m, fact = 2)
```

---

**est_rf_hmm**

*Multipoint analysis using Hidden Markov Models in autopolyploids*

**Description**

Performs the multipoint analysis proposed by *Mollinari and Garcia (2019)* in a sequence of markers.

**Usage**

```r
est_rf_hmm(
  input.seq,
  input.ph = NULL,
  thres = 0.5,
  twopt = NULL,
  verbose = FALSE,
  tol = 1e-04,
  est.given.0.rf = FALSE,
  reestimate.single.ph.configuration = TRUE,
  high.prec = TRUE
)
```

**S3 method for class 'mappoly.map'**

```r
print(x, detailed = FALSE, ...)
```

**S3 method for class 'mappoly.map'**

```r
plot(
  x,
  left.lim = 0,
  right.lim = Inf,
  phase = TRUE,
  mrk.names = FALSE,
  cex = 1,
  config = "best",
  P = "Parent 1",
  Q = "Parent 2",
  xlim = NULL,
  ...
)
```
Arguments

input.seq  an object of class mappoly.sequence
input.ph  an object of class two.pts.linkage.phases. If not available (default = NULL), it will be computed
thres  LOD Score threshold used to determine if the linkage phases compared via two-point analysis should be considered. Smaller values will result in smaller number of linkage phase configurations to be evaluated by the multipoint algorithm.
twopt  an object of class mappoly.twopt containing two-point information
verbose  if TRUE, current progress is shown; if FALSE (default), no output is produced
tol  the desired accuracy (default = 1e-04)
est.given.0.rf  logical. If TRUE returns a map forcing all recombination fractions equals to 0 (1e-5, for internal use only. Default = FALSE)
reestimate.single.ph.configuration  logical. If TRUE returns a map without re-estimating the map parameters for cases where there is only one possible linkage phase configuration. This argument is intended to be used in a sequential map construction
high.prec  logical. If TRUE (default) uses high precision long double numbers in the HMM procedure
x  an object of the class mappoly.map
detailed  logical. if TRUE, prints the linkage phase configuration and the marker position for all maps. If FALSE (default), prints a map summary
... currently ignored
left.lim  the left limit of the plot (in cM, default = 0).
right.lim  the right limit of the plot (in cM, default = Inf, i.e., will print the entire map)
phase  logical. If TRUE (default) plots the phase configuration for both parents
mrk.names  if TRUE, marker names are displayed (default = FALSE)
cex  The magnification to be used for marker names
config  should be 'best' or the position of the configuration to be plotted. If 'best', plot the configuration with the highest likelihood
P  a string containing the name of parent P
Q  a string containing the name of parent Q
xlim  range of the x-axis. If xlim = NULL (default) it uses the map range.

Details

This function first enumerates a set of linkage phase configurations based on two-point recombination fraction information using a threshold provided by the user (argument thresh). After that, for each configuration, it reconstructs the genetic map using the HMM approach described in Mollinari and Garcia (2019). As result, it returns the multipoint likelihood for each configuration in form of LOD Score comparing each configuration to the most likely one. It is recommended to use a small number of markers (e.g. 50 markers for hexaploids) since the possible linkage phase combinations bounded only by the two-point information can be huge. Also, it can be quite sensible to small changes in 'thresh'. For a large number of markers, please see est_rf_hmm_sequential.
Value

A list of class `mappoly.map` with two elements:

i) info: a list containing information about the map, regardless of the linkage phase configuration:

- **ploidy** the ploidy level
- **n.mrk** number of markers
- **seq.num** a vector containing the (ordered) indices of markers in the map, according to the input file
- **mrk.names** the names of markers in the map
- **seq.dose.p1** a vector containing the dosage in parent 1 for all markers in the map
- **seq.dose.p2** a vector containing the dosage in parent 2 for all markers in the map
- **chrom** a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, `chrom = NULL`
- **genome.pos** physical position (usually in megabase) of the markers into the sequence
- **seq.ref** reference base used for each marker (i.e. A, T, C, G). If not available, `seq.ref = NULL`
- **seq.alt** alternative base used for each marker (i.e. A, T, C, G). If not available, `seq.alt = NULL`
- **chisq.pval** a vector containing p-values of the chi-squared test of Mendelian segregation for all markers in the map
- **data.name** name of the dataset of class `mappoly.data`
- **ph.thres** the LOD threshold used to define the linkage phase configurations to test

ii) a list of maps with possible linkage phase configuration. Each map in the list is also a list containing

- **seq.num** a vector containing the (ordered) indices of markers in the map, according to the input file
- **seq.rf** a vector of size `(n.mrk - 1)` containing a sequence of recombination fraction between the adjacent markers in the map
- **seq.ph** linkage phase configuration for all markers in both parents
- **loglike** the hmm-based multipoint likelihood

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Examples

```r
mrk.subset <- make_seq_mappoly(hexafake, 1:10)
red.mrk <- elim_redundant(mrk.subset)
unique.mrks <- make_seq_mappoly(red.mrk)
subset.pairs <- est_pairwise_rf(input.seq = unique.mrks,
                                 ncpus = 1,
                                 verbose = TRUE)

## Estimating subset map with a low tolerance for the E.M. procedure
## for CRAN testing purposes
subset.map <- est_rf_hmm(input.seq = unique.mrks,
                          thres = 2,
                          twopt = subset.pairs,
                          verbose = TRUE,
                          tol = 0.1,
                          est.given.0.rf = FALSE)

subset.map
## linkage phase configuration with highest likelihood
plot(subset.map, mrk.names = TRUE, config = "best")
## the second one
plot(subset.map, mrk.names = TRUE, config = 2)
```

---

**est_rf_hmm_sequential**  *Multipoint analysis using Hidden Markov Models: Sequential phase elimination*

**Description**

Performs the multipoint analysis proposed by *Mollinari and Garcia (2019)* in a sequence of markers removing unlikely phases using sequential multipoint information.

**Usage**

```r
est_rf_hmm_sequential(
  input.seq,
  twopt,
  start.set = 4,
  thres.twopt = 5,
  thres.hmm = 50,
  extend.tail = NULL,
  phase.number.limit = 20,
  sub.map.size.diff.limit = Inf,
  info.tail = TRUE,
  reestimate.single.ph.configuration = FALSE,
  tol = 0.1,
  tol.final = 0.001,
  verbose = TRUE,
)```

detailed.verbose = FALSE,
high.prec = FALSE)

Arguments

input.seq  an object of class mappoly.sequence
twopt      an object of class mappoly.twopt containing the two-point information
start.set  number of markers to start the phasing procedure (default = 4)
thres.twopt the LOD threshold used to determine if the linkage phases compared via two-point analysis should be considered for the search space reduction (A.K.A. \( \eta \) in Mollinari and Garcia (2019), default = 5)
thres.hmm   the LOD threshold used to determine if the linkage phases compared via hmm analysis should be evaluated in the next round of marker inclusion (default = 50)
extend.tail the length of the chain’s tail that should be used to calculate the likelihood of the map. If NULL (default), the function uses all markers positioned. Even if info.tail = TRUE, it uses at least extend.tail as the tail length
phase.number.limit the maximum number of linkage phases of the sub-maps defined by arguments info.tail and extend.tail. Default is 20. If the size exceeds this limit, the marker will not be inserted. If Inf, then it will insert all markers.
sub.map.size.diff.limit the maximum accepted length difference between the current and the previous sub-map defined by arguments info.tail and extend.tail. If the size exceeds this limit, the marker will not be inserted. If NULL(default), then it will insert all markers.
info.tail   if TRUE (default), it uses the complete informative tail of the chain (i.e. number of markers where all homologous (ploidyx2) can be distinguished) to calculate the map likelihood
reestimate.single.ph.configuration logical. If FALSE (default) returns a map without re-estimating the map parameters in cases where there are only one possible linkage phase configuration
tol         the desired accuracy during the sequential phase (default = 10e-02)
tol.final   the desired accuracy for the final map (default = 10e-04)
verbose     If TRUE (default), current progress is shown; if FALSE, no output is produced
detailed.verbose
high.prec   logical. If TRUE uses high precision (long double) numbers in the HMM procedure implemented in C++, which can take a long time to perform (default = FALSE)

Details

This function sequentially includes markers into a map given an ordered sequence. It uses two-point information to eliminate unlikely linkage phase configurations given thres.twopt. The search is made within a window of size extend.tail. For the remaining configurations, the HMM-based likelihood is computed and the ones that pass the HMM threshold (thres.hmm) are eliminated.
Value

A list of class mappoly.map with two elements:

i) info: a list containing information about the map, regardless of the linkage phase configuration:

- **ploidy**: the ploidy level
- **n.mrk**: number of markers
- **seq.num**: a vector containing the (ordered) indices of markers in the map, according to the input file
- **mrk.names**: the names of markers in the map
- **seq.dose.p1**: a vector containing the dosage in parent 1 for all markers in the map
- **seq.dose.p2**: a vector containing the dosage in parent 2 for all markers in the map
- **chrom**: a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, chrom = NULL
- **genome.pos**: physical position (usually in megabase) of the markers into the sequence
- **seq.ref**: reference base used for each marker (i.e. A, T, C, G). If not available, seq.ref = NULL
- **seq.alt**: alternative base used for each marker (i.e. A, T, C, G). If not available, seq.alt = NULL
- **chisq.pval**: a vector containing p-values of the chi-squared test of Mendelian segregation for all markers in the map
- **data.name**: name of the dataset of class mappoly.data
- **ph.thres**: the LOD threshold used to define the linkage phase configurations to test

ii) a list of maps with possible linkage phase configuration. Each map in the list is also a list containing

- **seq.num**: a vector containing the (ordered) indices of markers in the map, according to the input file
- **seq.rf**: a vector of size \((n.mrk - 1)\) containing a sequence of recombination fraction between the adjacent markers in the map
- **seq.ph**: linkage phase configuration for all markers in both parents
- **loglike**: the hmm-based multipoint likelihood

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

**Examples**

```r
mrk.subset <- make_seq_mappoly(hexafake, 1:20)
red.mrk <- elim_redundant(mrk.subset)
unique.mrks <- make_seq_mappoly(red.mrk)
subset.pairs <- est_pairwise_rf(input.seq = unique.mrks,
                                ncpus = 1,
                                verbose = TRUE)
subset.map <- est_rf_hmm_sequential(input.seq = unique.mrks,
                                      thres.twopt = 5,
                                      thres.hmm = 10,
                                      extend.tail = 10,
                                      tol = 0.1,
                                      tol.final = 10e-3,
                                      phase.number.limit = 5,
                                      twopt = subset.pairs,
                                      verbose = TRUE)

print(subset.map, detailed = TRUE)
plot(subset.map)
plot(subset.map, left.lim = 0, right.lim = 1, mrk.names = TRUE)
plot(subset.map, phase = FALSE)
```

```r
## Retrieving simulated linkage phase
ph.P <- maps.hexafake[[1]]$maps[[1]]$seq.ph$P
ph.Q <- maps.hexafake[[1]]$maps[[1]]$seq.ph$Q
## Estimated linkage phase
ph.P.est <- subset.map$maps[[1]]$seq.ph$P
ph.Q.est <- subset.map$maps[[1]]$seq.ph$Q
compare_haplotypes(ploidy = 6, h1 = ph.P[names(ph.P.est)], h2 = ph.P.est)
compare_haplotypes(ploidy = 6, h1 = ph.Q[names(ph.Q.est)], h2 = ph.Q.est)
```

---

**Description**


**Usage**

```r
export_data_to_polymapR(data.in)
```

**Arguments**

- `data.in`: an object of class `mappoly.data`
Value

a dosage matrix

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

export_map_list

Export a genetic map to a CSV file

Description

Function to export genetic linkage map(s) generated by MAPpoly. The map(s) should be passed as a single object or a list of objects of class mappoly.map.

Usage

export_map_list(map.list, file = "map_output.csv")

Arguments

map.list A list of objects or a single object of class mappoly.map

file either a character string naming a file or a connection open for writing. "" indicates output to the console.

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

export_map_list(solcap.err.map[[1]], file = "")
Description

Compute homolog probabilities for all individuals in the full-sib population given a map and conditional genotype probabilities, and exports the results to be used for QTL mapping in the QTLpoly package.

Usage

```r
export_qtlpoly(input.genoprobs, verbose = TRUE)
```

Arguments

- `input.genoprobs`: an object of class `mappoly.genoprob`
- `verbose`: if TRUE (default), the current progress is shown; if FALSE, no output is produced

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

```r
## tetraploid example
w1 <- calc_genoprob(solcap.dose.map[[1]])
h.prob <- export_qtlpoly(w1)
```
filter_individuals

Filter out individuals

Description

This function removes individuals from the data set. Individuals can be user-defined or can be accessed via interactive kinship analysis.

Usage

filter_individuals(
  input.data,
  ind.to.remove = NULL,
  inter = TRUE,
  verbose = TRUE
)

Arguments

input.data  name of input object (class mappoly.data)
ind.to.remove individuals to be removed. If NULL it opens an interactive graphic to proceed with the individual selection
inter  if TRUE, expects user-input to proceed with filtering
verbose  if TRUE (default), shows the filtered out individuals

extract_map

Extract the maker position from an object of class ‘mappoly.map’

Description

Extract the maker position from an object of class ‘mappoly.map’

Usage

extract_map(input.map, phase.config = "best")

Arguments

input.map  An object of class mappoly.map
phase.config  which phase configuration should be used. "best" (default) will choose the maximum likelihood configuration

Examples

x <- maps.hexafake[[1]]$info$genome.pos/1e6
y <- extract_map(maps.hexafake[[1]])
plot(y~x, ylab = "Map position (cM)", xlab = "Genome Position (Mbp)")
filter_missing

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Description

Excludes markers or individuals based on their proportion of missing data

Usage

```r
filter_missing(
  input.data,
  type = c("marker", "individual"),
  filter.thres = 0.2,
  inter = TRUE
)
```

Arguments

- **input.data**: an object of class `mappoly.data`
- **type**: one of the following options: `"marker"` (filter out markers based on their percentage of missing data, default) or `"individual"` (filter out individuals based on their percentage of missing data). Please notice that removing individuals with certain amount of data can change some marker parameters (such as depth), and can also change the estimated genotypes for other individuals. So be careful when removing individuals.
- **filter.thres**: maximum percentage of missing data (default = 0.2)
- **inter**: if `TRUE`, expects user-input to proceed with filtering

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

```r
plot(tetra.solcap)
dat.filt.mrk <- filter_missing(input.data = tetra.solcap,
                               type = "marker",
                               filter.thres = 0.1,
                               inter = TRUE)
plot(dat.filt.mrk)
```
filter_segregation

Filter markers based on chi-square test

Description

This function filters markers based on p-values of a chi-square test. The chi-square test assumes
that markers follow the expected segregation patterns under Mendelian inheritance, random chro-
mosome bivalent pairing and no double reduction.

Usage

filter_segregation(input.data, chisq.pval.thres = NULL, inter = TRUE)

Arguments

input.data       name of input object (class mappoly.data)
chisq.pval.thres p-value threshold used for chi-square tests (default = Bonferroni approximation
inter          with global alpha of 0.05, i.e., 0.05/n.mrk)
                if TRUE (default), plots distorted vs. non-distorted markers

Value

An object of class mappoly.chitest.seq which contains a list with the following components:

keep     markers that follow Mendelian segregation pattern
exclude  markers with distorted segregation
chisq.pval.thres threshold p-value used for chi-square tests
data.name  input dataset used to perform the chi-square tests

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

mrks.chi.filt <- filter_segregation(input.data = tetra.solcap,
                                  chisq.pval.thres = 0.05/tetra.solcap$n.mrk,
                                  inter = TRUE)

seq.init <- make_seq_mappoly(mrks.chi.filt)
**find_blocks**

Allocate markers into linkage blocks

**Description**

Function to allocate markers into linkage blocks. This is an EXPERIMENTAL FUNCTION and should be used with caution.

**Usage**

```r
find_blocks(
  input.seq, 
  clustering.type = c("rf", "genome"), 
  rf.limit = 1e-04, 
  genome.block.threshold = 10000, 
  rf.mat = NULL, 
  ncpus = 1, 
  ph.thres = 3, 
  phase.number.limit = 10, 
  error = 0.05, 
  verbose = TRUE, 
  tol = 0.01, 
  tol.err = 0.001 
)
```

**Arguments**

- **input.seq** an object of class mappoly.sequence.
- **clustering.type** if 'rf', it uses UPGMA clusterization based on the recombination fraction matrix to assemble blocks. Linkage blocks are assembled by cutting the clusterization tree at rf.limit. If 'genome', it splits the marker sequence at neighbor markers more than 'genome.block.threshold' apart.
- **rf.limit** the maximum value to consider linked markers in case of 'clustering.type = rf'
- **genome.block.threshold** the threshold to assume markers are in the same linkage block, to be considered when allocating markers into blocks in case of 'clustering.type = genome'
- **rf.mat** an object of class mappoly.rf.matrix.
- **ncpus** Number of parallel processes to spawn
- **ph.thres** the threshold used to sequentially phase markers. Used in thres.twopt and thres.hmm. See *est_rf_hmm_sequential* for details.
- **phase.number.limit** the maximum number of linkage phases of the sub-maps. The default is 10. See *est_rf_hmm_sequential* for details.
find_blocks

error
the assumed global genotyping error rate. If NULL (default) it does not include an error in the block estimation.

verbose
if TRUE (default), the current progress is shown; if FALSE, no output is produced.

tol
tolerance for the C routine, i.e., the value used to evaluate convergence.

tol.err
tolerance for the C routine, i.e., the value used to evaluate convergence, including the global genotyping error in the model.

Value
a list containing 1: a list of blocks in form of mappoly.map objects; 2: a vector containing markers that were not included into blocks.

Author(s)
Marcelo Mollinari, <mmollin@ncsu.edu>

Examples
## Not run:
## Selecting 50 markers in chromosome 5
s5 <- make_seq_mappoly(tetra.solcap, "seq5")
s5 <- make_seq_mappoly(tetra.solcap, s5$seq.mrk.names[1:50])
tpt5 <- est_pairwise_rf(s5)
m5 <- rf_list_to_matrix(tpt5, 3, 3)
fb.rf <- find_blocks(s5, rf.mat = m5, verbose = FALSE, ncpus = 2)
bl.rf <- fb.rf$blocks
plot_map_list(bl.rf)

## Merging resulting maps
map.merge <- merge_maps(bl.rf, tpt5)
plot(map.merge, mrk.names = T)

## Comparing linkage phases with pre assembled map
id <- na.omit(match(map.merge$info$mrk.names, solcap.err.map[[5]]$info$mrk.names))
map.orig <- get_submap(solcap.err.map[[5]], mrk.pos = id)

p1.m$<map.merge$maps[[1]]$seq.ph$P
p2.m$<map.merge$maps[[1]]$seq.ph$Q
names(p1.m) <- names(p2.m) <- map.merge$info$mrk.names

p1.o$<map.orig$maps[[1]]$seq.ph$P
p2.o$<map.orig$maps[[1]]$seq.ph$Q
names(p1.o) <- names(p2.o) <- map.orig$info$mrk.names
n <- intersect(names(p1.m), names(p1.o))
plot_compare_haplotypes(4, p1.o[n], p2.o[n], p1.m[n], p2.m[n])

## Using genome
fb.geno <- find_blocks(s5, clustering.type = "genome", genome.block.threshold = 10^4)
plot_map_list(fb.geno$blocks)
splt <- lapply(fb.geno$blocks, split_mappoly, 1)
plot_map_list(splt)

## End(Not run)
get_genomic_order

Get the genomic position of markers in a sequence

Description

This function gets the genomic position of markers in a sequence and returns an ordered data frame with the name and position of each marker.

Usage

get_genomic_order(input.seq, verbose = TRUE)

## S3 method for class 'mappoly.geno.ord'
print(x, ...)

## S3 method for class 'mappoly.geno.ord'
plot(x, ...)

Arguments

input.seq  a sequence object of class mappoly.sequence
verbose  if TRUE (default), the current progress is shown; if FALSE, no output is produced
x  an object of the class mappoly.geno.ord
...  currently ignored

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

s1 <- make_seq_mappoly(tetra.solcap, "all")
o1 <- get_genomic_order(s1)
plot(o1)
s.geno.ord <- make_seq_mappoly(o1)

get_submap

Extract sub-map from map

Description

Given a pre-constructed map, it extracts a sub-map for a provided sequence of marker positions. Optionally, it can update the linkage phase configurations and respective recombination fractions.
Usage

get_submap(
  input.map,  
  mrk.pos,   
  phase.config = "best",  
  reestimate.rf = TRUE,  
  reestimate.phase = FALSE,  
  thres.twopt = 5,  
  thres.hmm = 3,  
  extend.tail = 50,  
  tol = 0.1,  
  tol.final = 0.001,  
  use.high.precision = FALSE,  
  verbose = TRUE
)

Arguments

input.map An object of class mappoly.map
mrk.pos positions of the markers that should be considered in the new map. This can be in any order
phase.config which phase configuration should be used. "best" (default) will choose the configuration associated with the maximum likelihood
reestimate.rf logical. If TRUE (default) the recombination fractions between markers are re-estimated
reestimate.phase logical. If TRUE, the linkage phase configurations are re-estimated (default = FALSE)
thres.twopt the LOD threshold used to determine if the linkage phases compared via two-point analysis should be considered (default = 5)
thres.hmm the threshold used to determine if the linkage phases compared via hmm analysis should be considered (default = 3)
extend.tail the length of the tail of the chain that should be used to calculate the likelihood of the linkage phases. If info.tail = TRUE, the function uses at least extend.tail as the length of the tail (default = 50)
tol the desired accuracy during the sequential phase (default = 0.1)
tol.final the desired accuracy for the final map (default = 10e-04)
use.high.precision logical. If TRUE uses high precision (long double) numbers in the HMM procedure implemented in C++, which can take a long time to perform (default = FALSE)
verbose If TRUE (default), current progress is shown; if FALSE, no output is produced

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>
References


Examples

```r
## selecting the six first markers in linkage group 1
## re-estimating the recombination fractions and linkage phases
submap1.lg1 <- get_submap(input.map = maps.hexafake[[1]],
                           mrk.pos = 1:6, verbose = TRUE,
                           reestimate.phase = TRUE,
                           tol.final = 10e-3)

## no recombination fraction re-estimation: first 20 markers
submap2.lg1 <- get_submap(input.map = maps.hexafake[[1]],
                           mrk.pos = 1:20, reestimate.rf = FALSE,
                           verbose = TRUE,
                           tol.final = 10e-3)

plot(maps.hexafake[[1]])
plot(submap1.lg1, mrk.names = TRUE, cex = .8)
plot(submap2.lg1, mrk.names = TRUE, cex = .8)
```

---

**get_tab_mrks**

*Get table of dosage combinations*

**Description**

Internal function

**Usage**

`get_tab_mrks(x)`

**Arguments**

- `x` an object of class `mappoly.map`

**Author(s)**

Gabriel Gesteira, <gdesiqu@ncsu.edu>
Assign markers to linkage groups

Description

Identifies linkage groups of markers using the results of two-point (pairwise) analysis.

Usage

```r
group_mappoly(
  input.mat,
  expected.groups = NULL,
  inter = TRUE,
  comp.mat = FALSE,
  LODweight = FALSE,
  verbose = TRUE
)
```

Arguments

- `input.mat` an object of class `mappoly.rf.matrix`
- `expected.groups` when available, inform the number of expected linkage groups (i.e. chromosomes) for the species
- `inter` if `TRUE` (default), plots a dendrogram highlighting the expected groups before continue
- `comp.mat` if `TRUE`, shows a comparison between the reference based and the linkage based grouping, if the chromosome information is available (default = `FALSE`)
- `LODweight` if `TRUE`, clusterization is weighted by the square of the LOD Score
- `verbose` logical. If `TRUE` (default), current progress is shown; if `FALSE`, no output is produced

Value

Returns an object of class `mappoly.group`, which is a list containing the following components:

- `data.name` the referred dataset name
- `hc.snp` a list containing information related to the UPGMA grouping method
- `expected.groups` the number of expected linkage groups
- `groups.snp` the groups to which each of the markers belong
- `seq.vs.grouped.snp` comparison between the genomic group information (when available) and the groups provided by `group_mappoly`
- `chisq.pval.thres` the threshold used on the segregation test when reading the dataset
- `chisq.pval` the p-values associated with the segregation test for all markers in the sequence
hexafake

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

```r
## Getting first 20 markers from two linkage groups
all.mrk <- make_seq_mappoly(hexafake, c(1:20,601:620))
red.mrk <- elim_redundant(all.mrk)
unique.mrks <- make_seq_mappoly(red.mrk)
counts <- cache_counts_twopt(unique.mrks, cached = TRUE)
all.pairs <- est_pairwise_rf(input.seq = unique.mrks,
                           count.cache = counts,
                           ncpus = 1,
                           verbose = TRUE)

## Full recombination fraction matrix
mat.full <- rf_list_to_matrix(input.twopt = all.pairs)
plot(mat.full, index = FALSE)

lgs <- group_mappoly(input.mat = mat.full,
                      expected.groups = 2,
                      inter = TRUE,
                      comp.mat = TRUE, #this data has physical information
                      verbose = TRUE)

lgs
plot(lgs)
```

hexafake  

*Simulated autohexaploid dataset.*

Description

A dataset of a hypothetical autohexaploid full-sib population containing three homology groups

Usage

hexafake
Format

An object of class `mappoly.data` which contains a list with the following components:

- **plody** ploidy level = 6
- **n.ind** number individuals = 300
- **n.mrk** total number of markers = 1500
- **ind.names** the names of the individuals
- **mrk.names** the names of the markers
- **dosage.p1** a vector containing the dosage in parent P for all `n.mrk` markers
- **dosage.p2** a vector containing the dosage in parent Q for all `n.mrk` markers
- **chrom** a vector indicating the chromosome each marker belongs. Zero indicates that the marker was not assigned to any chromosome
- **genome.pos** Physical position of the markers into the sequence
- **geno.dose** a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by `ploidy_level + 1 = 7`
- **n.phen** There are no phenotypes in this simulation
- **phen** There are no phenotypes in this simulation
- **chisq.pval** vector containing p-values for all markers associated to the chi-square test for the expected segregation patterns under Mendelian segregation

---

**hexafake.geno.dist**

Simulated autohexaploid dataset with genotype probabilities.

Description

A dataset of a hypothetical autohexaploid full-sib population containing three homology groups. This dataset contains the probability distribution of the genotypes and 2% of missing data, but is essentially the same dataset found in `hexafake`

Usage

- `hexafake.geno.dist`

Format

An object of class `mappoly.data` which contains a list with the following components:

- **plody** ploidy level = 6
- **n.ind** number individuals = 300
- **n.mrk** total number of markers = 1500
- **ind.names** the names of the individuals
- **mrk.names** the names of the markers
**import_data_from_polymapR**

*Import data from polymapR*

**Description**

Function to import datasets from polymapR.

**Usage**

```r
import_data_from_polymapR(
  input.data,
  ploidy,
  parent1 = "P1",
  parent2 = "P2",
  input.type = c("discrete", "probabilistic"),
  prob.thres = 0.95,
  pardose = NULL,
  offspring = NULL,
  filter.non.conforming = TRUE,
  verbose = TRUE
)
```

**Arguments**

- `input.data` a polymapR dataset
- `ploidy` the ploidy level
- `parent1` a character string containing the name (or pattern of genotype IDs) of parent 1
- `parent2` a character string containing the name (or pattern of genotype IDs) of parent 2
- `input.type` the type of input data as a character string: "discrete" or "probabilistic"
- `prob.thres` the probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than `prob.thres` are considered as missing data for the dosage calling purposes
- `pardose` the dosage in parent P for all markers
- `offspring` the dosage in parent Q for all markers
- `filter.non.conforming` logical, if `TRUE` the function will filter markers that do not conform to the input type
- `verbose` logical, if `TRUE` the function will print progress messages

---

**import_data_from_polymapR**

*a vector containing the dosage in parent P for all markers*

**dosage.p1**

*a vector containing the dosage in parent Q for all markers*

**dosage.p2**

*a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence*

**chrom**

*Physical position of the markers into the sequence*

**genome.pos**

*The genome position of the markers into the sequence*

**prob.thres = 0.95**

*The probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than 'prob.thres' are considered as missing data for the dosage calling purposes*

**geno**

*a data.frame containing the probability distribution for each combination of marker and offspring. The first two columns represent the marker and the offspring, respectively. The remaining columns represent the probability associated to each one of the possible dosages*

**geno.dose**

*a matrix containing the dosage for each marker (rows) for each individual (columns). Missing data are represented by ploidy_level + 1 = 7*

**n.phen**

*There are no phenotypes in this simulation*

**phen**

*There are no phenotypes in this simulation*
**parent2**  
a character string containing the name (or pattern of genotype IDs) of parent 2

**input.type**  
Indicates whether the input is discrete ("disc") or probabilistic ("prob")

**prob.thres**  
threshold probability to assign a dosage to offspring. If the probability is smaller than `thresh.parent.geno`, the data point is converted to 'NA'.

**pardose**  
matrix of dimensions (n.mrk x 3) containing the name of the markers in the first column, and the dosage of parents 1 and 2 in columns 2 and 3. (see polymapR vignette)

**offspring**  
a character string containing the name (or pattern of genotype IDs) of the offspring individuals. If NULL (default) it considers all individuals as offspring, except `parent1` and `parent2`.

**filter.non.conforming**  
if TRUE exclude samples with non expected genotypes under no double reduction. Since markers were already filtered in polymapR, the default is FALSE.

**verbose**  
if TRUE (default), the current progress is shown; if FALSE, no output is produced

**Details**


**Author(s)**

Marcelo Mollinari &lt;mmollin@ncsu.edu&gt;

**References**


---

**import_from_updog**  
*Import from updog*

**Description**

Read objects with information related to genotype calling in polyploids. Currently this function supports output objects created with the updog (output of `multidog` function) package. This function creates an object of class `mappoly.data`
import_from_updog

Usage

import_from_updog(
  object,  
  prob.thres = 0.95,  
  filter.non.conforming = TRUE,  
  chrom = NULL,  
  genome.pos = NULL,  
  verbose = TRUE  
)

Arguments

object the name of the object of class multidog
prob.thres probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than 'prob.thres' are considered as missing data for the dosage calling purposes
filter.non.conforming if TRUE (default) exclude samples with non expected genotypes under random chromosome pairing and no double reduction
chrom a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
genome.pos vector with physical position of the markers into the sequence
verbose if TRUE (default), the current progress is shown; if FALSE, no output is produced

Value

An object of class mappoly.data which contains a list with the following components:

ploidy ploidy level
n.ind number individuals
n.mrk total number of markers
ind.names the names of the individuals
mrk.names the names of the markers
dosage.p1 a vector containing the dosage in parent P for all n.mrk markers
dosage.p2 a vector containing the dosage in parent Q for all n.mrk markers
chrom a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
genome.pos physical position of the markers into the sequence
prob.thres probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than 'prob.thres' were considered as missing data in the 'geno.dose' matrix
geno.dose a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1
import_phased_maplist_from_polymapR

Description

Function to import phased map lists from polymapR

genot a data.frame containing the probability distribution for each combination of marker and offspring. The first two columns represent the marker and the offspring, respectively. The remaining elements represent the probability associated to each one of the possible dosages. Missing data are converted from NA to the expected segregation ratio using function segreg_poly

n.phen number of phenotypic traits

phen a matrix containing the phenotypic data. The rows correspond to the traits and the columns correspond to the individuals

chisq.pval a vector containing p-values related to the chi-squared test of Mendelian segregation performed for all markers

Author(s)

Gabriel Gesteira, <gdesiqu@ncsu.edu>

References


Examples

```r
if(requireNamespace("updog", quietly = TRUE)){
library("updog")
data("uitdewilligen")
mout = multidog(refmat = t(uitdewilligen$refmat),
sizemat = t(uitdewilligen$sizemat),
ploidy = uitdewilligen$ploidy,
model = "f1",
p1_id = colnames(t(uitdewilligen$sizemat))[1],
p2_id = colnames(t(uitdewilligen$sizemat))[2],
nc = 2)
mydata = import_from_updog(mout)
mydata
plot(mydata)
}
```
loglike_hmm

Usage

import_phased_maplist_from_polymapR(maplist, mappoly.data, ploidy = NULL)

Arguments

maplist a list of phased maps obtained using function create_phased_maplist from package polymapR
mappoly.data a dataset used to obtain maplist, converted into class mappoly.data
ploidy the ploidy level

Details


Author(s)

Marcelo Mollinari <mmollin@ncsu.edu>

References


loglike_hmm Multipoint log-likelihood computation

Description

Update the multipoint log-likelihood of a given map using the method proposed by Mollinari and Garcia (2019).

Usage

loglike_hmm(input.map, input.data = NULL, verbose = FALSE)

Arguments

input.map An object of class mappoly.map
input.data An object of class mappoly.data, which was used to generate input.map
verbose If TRUE, map information is shown; if FALSE(default), no output is produced
Author(s)
Marcelo Mollinari, <mmollin@ncsu.edu>

References

Examples

```r
hexa.map <- loglike_hmm(maps.hexafake[[1]])
hexa.map
```

Description
Get a subset of an object of class mappoly.rf.matrix, i.e. recombination fraction and LOD score matrices based in a sequence of markers.

Usage
```r
make_mat_mappoly(input.mat, input.seq)
```

Arguments
- `input.mat`: an object of class mappoly.rf.matrix
- `input.seq`: an object of class mappoly.sequence, with a sequence of markers contained in `input.mat`

Value
an object of class mappoly.rf.matrix, which is a subset of 'input.mat'. See `rf_list_to_matrix` for details

Author(s)
Marcelo Mollinari, <mmollin@ncsu.edu>

References
Examples

```r
# sequence with 20 markers
mrk.seq <- make_seq_mappoly(hexafake, 1:20)
mrk.pairs <- est_pairwise_rf(input.seq = mrk.seq,
                             verbose = TRUE)
  ## Full recombination fraction matrix
mat <- rf_list_to_matrix(input.twopt = mrk.pairs)
plot(mat)
  ## Matrix subset
id <- make_seq_mappoly(hexafake, 1:10)
mat.sub <- make_mat_mappoly(mat, id)
plot(mat.sub)
```

Description

Get a subset of an object of class `mappoly.twopt` or `mappoly.twopt2` (i.e., recombination fraction) and LOD score statistics for all possible linkage phase combinations based on a sequence of markers.

Usage

```r
make_pairs_mappoly(input.twopt, input.seq)
```

Arguments

- `input.twopt`: an object of class `mappoly.twopt`
- `input.seq`: an object of class `mappoly.sequence`, with a sequence of markers contained in `input.twopt`

Value

an object of class `mappoly.twopt` which is a subset of `input.twopt`. See `est_pairwise_rf` for details

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Examples

```r
## selecting some markers along the genome
some.mrk <- make_seq_mappoly(hexafake, seq(1, 1500, 30))
all.pairs <- est_pairwise_rf(input.seq = some.mrk)
mat.full <- rf_list_to_matrix(input.twopt = all.pairs)
plot(mat.full)

## selecting two-point information for chromosome 1
mrks.1 <- make_seq_mappoly(hexafake, names(which(some.mrk$chrom == 1)))
p1 <- make_pairs_mappoly(input.seq = mrks.1, input.twopt = all.pairs)
m1 <- rf_list_to_matrix(input.twopt = p1)
plot(m1, main.text = "LG1")
```

### make_seq_mappoly

Create a sequence of markers

**Description**

Makes a sequence of markers based on an object of another class.

**Usage**

```r
make_seq_mappoly(input.obj, arg = NULL, data.name = NULL, genomic.info = NULL)
```

**Arguments**

- `input.obj`: an object of one of the following classes: mappoly.data, mappoly.map, mappoly.group, mappoly.unique.seq, mappoly.pcmapped, mappoly.pcmapped3d, or mappoly.geno.ord
- `arg`: can be one of the following objects: i) a string 'all', resulting in a sequence with all markers in the raw data; ii) a string or a vector of strings 'seqx', where x is the sequence (x = 0 indicates unassigned markers); iii) a vector of integers specifying which markers comprise the sequence; iv) an integer representing linkage group if input.object has class mappoly.group; or v) NULL if input.object has class mappoly.pcmapped, mappoly.pcmapped3d, mappoly.unique.seq, or mappoly.geno.ord
- `data.name`: name of the object of class mappoly.data
- `genomic.info`: optional argument applied for mappoly.group objects only. This argument can be NULL, or can hold the numeric combination of sequences from genomic information to be used when making the sequences. When genomic.info = NULL (default), the function returns a sequence containing all markers defined
by the grouping function. When genomic.info = 1, the function returns a sequence with markers that matched the intersection between grouping function and genomic information, considering the sequence from genomic information that holds the maximum number of markers matching the group; when genomic.info = c(1,2), the function returns a sequence with markers that matched the intersection between grouping function and genomic information, considering two sequences from genomic information that presented the maximum number of markers matching the group; and so on.

x an object of the class mappoly.sequence

... currently ignored

Value

An object of class mappoly.sequence, which is a list containing the following components:

- seq.num a vector containing the (ordered) indices of markers in the sequence, according to the input file
- seq.phases a list with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases
- seq.rf a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies
- loglike log-likelihood of the corresponding linkage map
- data.name name of the object of class mappoly.data with the raw data
- twopt name of the object of class mappoly.twopt with the 2-point analyses. -1 means that the twopt estimates were not computed

Author(s)

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

References


Examples

```r
all.mrk <- make_seq_mappoly(hexafake, 'all')
ext1.mrk <- make_seq_mappoly(hexafake, 'seq1')
plot(seq1.mrk)
some.mrk.pos <- c(1,4,28,32,45)
(some.mrk.1 <- make_seq_mappoly(hexafake, some.mrk.pos))
plot(some.mrk.1)
```
**maps.hexafake**

*Resulting maps from hexafake*

**Description**

A list containing three linkage groups estimated using the procedure available in [MAPpoly’s tutorial](https://mmollina.github.io/MAPpoly/#estimating_the_map_for_a_given_order)

**Usage**

```r
maps.hexafake
```

**Format**

A list containing three objects of class `mappoly.map`, each one representing one linkage group in the simulated data.

---

**mds_mappoly**

*Estimates loci position using Multidimensional Scaling*

**Description**

Estimates loci position using Multidimensional Scaling proposed by Preedy and Hackett (2016). The code is an adaptation from the package MDSmap, available under GNU GENERAL PUBLIC LICENSE, Version 3, at [https://CRAN.R-project.org/package=MDSMap](https://CRAN.R-project.org/package=MDSMap)

**Usage**

```r
mds_mappoly(
  input.mat,
  p = NULL,
  n = NULL,
  ndim = 2,
  weight.exponent = 2,
  verbose = TRUE
)
```

```r
## S3 method for class 'mappoly.pcmapper'
print(x, ...)
```

```r
## S3 method for class 'mappoly.pcmapper3d'
print(x, ...)
```
Arguments

input.mat: an object of class mappoly.input.matrix
p: integer. The smoothing parameter for the principal curve. If NULL (default) this will be done using the leave-one-out cross validation
n: vector of integers or strings containing loci to be omitted from the analysis
ndim: number of dimensions to be considered in the multidimensional scaling procedure (default = 2)
weight.exponent: the exponent that should be used in the LOD score values to weight the MDS procedure (default = 2)
verbose: if TRUE (default), display information about the analysis
x: an object of class mappoly.mds
...: currently ignored

Value

A list containing:

M: the input distance map
sm: the unconstrained MDS results
pc: the principal curve results
distmap: a matrix of pairwise distances between loci where the columns are in the estimated order
locimap: a data frame of the loci containing the name and position of each locus in order of increasing distance
length: integer giving the total length of the segment
removed: a vector of the names of loci removed from the analysis
scale: the scaling factor from the MDS
locikey: a data frame showing the number associated with each locus name for interpreting the MDS configuration plot
confplotno: a data frame showing locus name associated with each number on the MDS configuration plots

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu> mostly adapted from MDSmap codes, written by Katharine F. Preedy, <katharine.preedy@bioss.ac.uk>

References

merge_datasets

Merge datasets

Description

This function merges two datasets of class mappoly.data. This can be useful when individuals of a population were genotyped using two or more techniques and have datasets in different files or formats. Please notice that the datasets should contain the same number of individuals and they must be represented identically in both datasets (e.g. Ind_1 in both datasets, not Ind_1 in one dataset and ind_1 or Ind.1 in the other).

Usage

merge_datasets(dat.1 = NULL, dat.2 = NULL)

Arguments

dat.1 the first dataset of class mappoly.data to be merged
dat.2 the second dataset of class mappoly.data to be merged (default = NULL); if dat.2 = NULL, the function returns dat.1 only

Value

An object of class mappoly.data which contains all markers from both datasets. It will be a list with the following components:

ploidy ploidy level
n.ind number individuals
n.mrk total number of markers
ind.names the names of the individuals
mrk.names the names of the markers

Examples

```r
s1 <- make_seq_mappoly(hexafake, 1:20)
t1 <- est_pairwise_rf(s1, ncpus = 1)
m1 <- rf_list_to_matrix(t1)
o1 <- get_genomic_order(s1)
s.go <- make_seq_mappoly(o1)
plot(m1, ord = s.go$seq.mrk.names)
mds.ord <- mds_mappoly(m1)
plot(mds.ord)
s.o <- make_seq_mappoly(mds.ord)
plot(m1, ord = s.o$seq.mrk.names)
plot(s.o$seq.num ~ I(s.o$genome.pos/1e6),
     xlab = "Genome Position",
ylab = "MDS position")
```
merge_datasets

dosage.p1 a vector containing the dosage in parent P for all n.mrk markers
dosage.p2 a vector containing the dosage in parent Q for all n.mrk markers
chrom a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
genome.pos Physical position of the markers into the sequence
seq.ref if one or both datasets originated from read_vcf, it keeps reference alleles from sequencing platform, otherwise is NULL
seq.alt if one or both datasets originated from read_vcf, it keeps alternative alleles from sequencing platform, otherwise is NULL
all.mrk.depth if one or both datasets originated from read_vcf, it keeps marker read depths from sequencing, otherwise is NULL
prob.thres (unused field)
geno.dose a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1
geno if both datasets contain genotype distribution information, the final object will contain 'geno'. This is set to NULL otherwise
nphen (0)
phen (NULL)
chisq.pval a vector containing p-values related to the chi-squared test of Mendelian segregation performed for all markers in both datasets
kept if elim.redundant = TRUE when reading any dataset, holds all non-redundant markers
elim.correspondence if elim.redundant = TRUE when reading any dataset, holds all non-redundant markers and its equivalence to the redundant ones

Author(s)

Gabriel Gesteira, <gdesiqu@ncsu.edu>

References


Examples

## Loading a subset of SNPs from chromosomes 3 and 12 of sweetpotato dataset
## (SNPs anchored to Ipomoea trifida genome)
dat <- NULL
for(i in c(3, 12)){
  cat("Loading chromosome", i, "...\n")
  tempfl <- tempfile(pattern = paste0("ch", i), fileext = ".vcf.gz")
  x <- "https://github.com/mmollina/MAPpoly_vignettes/raw/master/data/sweet_sample_ch"
merge_maps

merge_maps is a function that merges two maps. It takes a list of objects of class mappoly.map and an object of class mappoly.twopt containing the two-point information for all pairs of markers present in the original maps. It also takes a threshold used to determine if the linkage phases compared via two-point analysis should be considered for the search space reduction, a list of objects of class mappoly.genoprob containing the genotype probabilities for the maps to be merged, a threshold used to determine which linkage phase configurations should be returned when merging two maps, and the desired accuracy.

Usage

merge_maps(map.list, twopt, thres.twopt = 10, genoprob.list = NULL, thres.hmm = "best", tol = 1e-04)

Arguments

map.list: a list of objects of class mappoly.map to be merged.
twopt: an object of class mappoly.twopt containing the two-point information for all pairs of markers present in the original maps.
thres.twopt: the threshold used to determine if the linkage phases compared via two-point analysis should be considered for the search space reduction (default = 3).
genoprob.list: a list of objects of class mappoly.genoprob containing the genotype probabilities for the maps to be merged. If NULL (default), the probabilities are computed.
thres.hmm: the threshold used to determine which linkage phase configurations should be returned when merging two maps. If "best" (default), returns only the best linkage phase configuration. NOTE: if merging multiple maps, it always uses the "best" linkage phase configuration at each block insertion.
tol: the desired accuracy (default = 10e-04)
Details

merge_maps uses two-point information, under a given LOD threshold, to reduce the linkage phase search space. The remaining linkage phases are tested using the genotype probabilities.

Value

A list of class mappoly.map with two elements:

i) info: a list containing information about the map, regardless of the linkage phase configuration:

- **ploidy**: the ploidy level
- **n.mrk**: number of markers
- **seq.num**: a vector containing the (ordered) indices of markers in the map, according to the input file
- **mrk.names**: the names of markers in the map
- **seq.dose.p1**: a vector containing the dosage in parent 1 for all markers in the map
- **seq.dose.p2**: a vector containing the dosage in parent 2 for all markers in the map
- **chrom**: a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, `chrom = NULL`
- **genome.pos**: physical position (usually in megabase) of the markers into the sequence
- **seq.ref**: reference base used for each marker (i.e. A, T, C, G). If not available, `seq.ref = NULL`
- **seq.alt**: alternative base used for each marker (i.e. A, T, C, G). If not available, `seq.alt = NULL`
- **chisq.pval**: a vector containing p-values of the chi-squared test of Mendelian segregation for all markers in the map
- **data.name**: name of the dataset of class mappoly.data
- **ph.thres**: the LOD threshold used to define the linkage phase configurations to test

ii) a list of maps with possible linkage phase configuration. Each map in the list is also a list containing

- **seq.num**: a vector containing the (ordered) indices of markers in the map, according to the input file
- **seq.rf**: a vector of size `(n.mrk - 1)` containing a sequence of recombination fraction between the adjacent markers in the map
- **seq.ph**: linkage phase configuration for all markers in both parents
- **loglike**: the hmm-based multipoint likelihood

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>
Examples

#### Tetraploid example ####

```r
map1 <- get_submap(solcap.dose.map[[1]], 1:5)
map2 <- get_submap(solcap.dose.map[[1]], 6:15)
map3 <- get_submap(solcap.dose.map[[1]], 16:30)
full.map <- get_submap(solcap.dose.map[[1]], 1:30)
s <- make_seq_mappoly(tetra.solcap, full.map$maps[[1]]$seq.num)
twopt <- est_pairwise_rf(input.seq = s)
merged.maps <- merge_maps(map.list = list(map1, map2, map3),
                         twopt = twopt,
                         thres.twopt = 3)
plot(merged.maps, mrk.names = TRUE)
plot(full.map, mrk.names = TRUE)
best.phase <- merged.maps$maps[[1]]$seq.ph
names.id <- names(best.phase$P)
compare_haplotypes(ploidy = 4, best.phase$P[names.id],
                   full.map$maps[[1]]$seq.ph$P[names.id])
compare_haplotypes(ploidy = 4, best.phase$Q[names.id],
                   full.map$maps[[1]]$seq.ph$Q[names.id])
```

---

### Description

Plots mappoly.homoprob

### Usage

```r
## S3 method for class 'mappoly.homoprob'
plot(
    x, stack = FALSE, lg = NULL, ind = NULL, use.plotly = TRUE,
    verbose = TRUE, ...
)
```

### Arguments

- `x`: an object of class mappoly.homoprob
- `stack`: logical. If TRUE, probability profiles of all homologues are stacked in the plot (default = FALSE)
plot.mappoly.prefpair.profiles

Plots mappoly.prefpair.profiles

Description

Plots mappoly.prefpair.profiles

Usage

## S3 method for class 'mappoly.prefpair.profiles'
plot(
  x,
  type = c("pair.configs", "hom.pairs"),
  min.y.prof = 0,
  max.y.prof = 1,
  thresh = 0.01,
  P1 = "P1",
  P2 = "P2",
  ...)

Arguments

x
  an object of class mappoly.prefpair.profiles

type
  a character string indicating which type of graphic is plotted: "pair.configs" (default) plots the preferential pairing profile for the pairing configurations or "hom.pairs" plots the preferential pairing profile for the homolog pairs

min.y.prof
  lower bound for y axis on the probability profile graphic (default = 0)

max.y.prof
  upper bound for y axis on the probability profile graphic (default = 1)

thresh
  threshold for chi-square test (default = 0.01)

P1
  a string containing the name of parent P1

P2
  a string containing the name of parent P2

... unused arguments
plot_genome_vs_map

Description

This function plots scatterplot(s) of physical distance (in Mbp) versus the genetic distance (in cM). Map(s) should be passed as a single object or a list of objects of class mappoly.map.

Usage

```r
plot_genome_vs_map(
  map.list,
  phase.config = "best",
  same.ch.lg = FALSE,
  alpha = 1/5,
  size = 3
)
```

Arguments

- **map.list**: A list or a single object of class mappoly.map
- **phase.config**: A vector containing which phase configuration should be plotted. If 'best' (default), plots the configuration with the highest likelihood for all elements in 'map.list'
- **same.ch.lg**: Logical. If TRUE displays only the scatterplots between the chromosomes and linkage groups with the same number. Default is FALSE.
- **alpha**: transparency factor for SNPs points
- **size**: size of the SNP points

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

```r
plot_genome_vs_map(solcap.mds.map, same.ch.lg = TRUE)
plot_genome_vs_map(solcap.mds.map, same.ch.lg = FALSE,
  alpha = 1, size = 1/2)
```
### plot_GIC  
*Genotypic information content*

**Description**
This function plots the genotypic information content given an object of class `mappoly.homoprob`.

**Usage**
```
plot_GIC(hprobs, P = "P1", Q = "P2")
```

**Arguments**
- `hprobs`: an object of class `mappoly.homoprob`
- `P`: a string containing the name of parent P
- `Q`: a string containing the name of parent Q

**Examples**
```
w <- lapply(solcap.err.map[1:3], calc_genoprob)
hs.probs <- calc_homologprob(w)
plot_GIC(hs.probs)
```

### plot_map_list  
*Plot a genetic map*

**Description**
This function plots a genetic linkage map(s) generated by `MAPpoly`. The map(s) should be passed as a single object or a list of objects of class `mappoly.map`.

**Usage**
```
plot_map_list(
  map.list, 
  horiz = TRUE, 
  col = "lightgray", 
  title = "Linkage group"
)
```
Arguments

map.list A list of objects or a single object of class mappoly.map
horiz logical. If FALSE, the maps are plotted vertically with the first map to the left. If TRUE (default), the maps are plotted horizontally with the first at the bottom
col a vector of colors for each linkage group. (default = 'lightgray') ggstyle produces maps using the default ggplot color palette.
title a title (string) for the maps (default = 'Linkage group')

Value

A data.frame object containing the name of the markers and their genetic position

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

```r
## hexafake map
plot_map_list(maps.hexafake, horiz = FALSE)
plot_map_list(maps.hexafake, col = c("#999999", "#E69F00", "#56B4E9"))

## solcap map
plot_map_list(solcap.dose.map, col = "ggstyle")
plot_map_list(solcap.dose.map, col = "mp_pallet3", horiz = FALSE)
```

---

Plot marker information

Description

Plots summary statistics for a given marker

Usage

plot_mrk_info(input.data, mrk)

Arguments

input.data an object of class mappoly.data
mrk marker name or position in the dataset
**poly_cross_simulate**

**Author(s)**
Marcelo Mollinari, <mmollin@ncsu.edu>

**References**

**Examples**

```r
plot_mrk_info(tetra.solcap.geno.dist, 2680)
plot_mrk_info(tetra.solcap.geno.dist, "solcap.snp.c2.23828")
```

---

**poly_cross_simulate**  
*Simulate an autopolyploid full-sib population*

**Description**
Simulate an autopolyploid full-sib population with one or two informative parents under random chromosome segregation.

**Usage**

```r
poly_cross_simulate(
  ploidy,  
  rf.vec,  
  n.mrk,  
  n.ind,  
  hom.allele,  
  draw = FALSE,  
  file = "output.pdf",  
  seed = NULL,  
  width = 12,  
  height = 6,  
  prob.P = NULL,  
  prob.Q = NULL
)
```

**Arguments**

- **ploidy**: ploidy level. Must be an even number
- **rf.vec**: vector containing the recombination fractions between adjacent markers. If a single recombination fraction is provided, it is repeated \( n.mrk - 1 \) times
- **n.mrk**: number of markers
n.ind number of individuals in the offspring
hom.allele a list containing the linkage phase information for both parents
draw if TRUE, draws a graphical representation of the parental map, including the linkage phase configuration, in a pdf output (default = FALSE)
file name of the output file. It is ignored if draw = TRUE
seed random number generator seed (default = NULL)
width the width of the graphics region in inches (default = 12)
height the height of the graphics region in inches (default = 6)
prob.P a vector indicating the proportion of preferential pairing in parent P (currently ignored)
prob.Q a vector indicating the proportion of preferential pairing in parent Q (currently ignored)

Details

hom.allele.p and hom.allele.q are lists of vectors containing linkage phase configurations. Each vector contains the numbers of the homologous chromosomes in which the alleles are located. For instance, a vector containing (1, 3, 4) means that the marker has three doses located in the chromosomes 1, 3 and 4. For zero doses, use 0. For more sophisticated simulations, we strongly recommend using PedigreeSim V2.0 [https://github.com/PBR/pedigreeSim](https://github.com/PBR/pedigreeSim)

Value

an object of class mappoly.data. See `read_geno` for more information

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

```r
h.temp <- sim_homologous(ploidy = 6, n.mrk = 20, max.d = 3, max.ph = 3, seed = 123)
fake.poly.dat <- poly_cross_simulate(ploidy = 6, rf.vec = .05, n.mrk = 20,
                                   n.ind = 200, h.temp, seed = 123)
plot(fake.poly.dat)
```
**print_mrk**

---

**Summary of a set of markers**

**Description**

Returns information related to a given set of markers

**Usage**

```
print_mrk(input.data, mrks)
```

**Arguments**

- `input.data`: an object of 'mappoly.data'
- `mrks`: marker sequence index (integer vector)

**Examples**

```
print_mrk(tetra.solcap.geno.dist, 1:5)
print_mrk(hexafake, 256)
```

---

**read_fitpoly**

---

**Data Input in fitPoly format**

**Description**

Reads an external data file generated as output of `saveMarkerModels`. This function creates an object of class `mappoly.data`.

**Usage**

```
read_fitpoly(
  file.in,
  ploidy,
  parent1,
  parent2,
  offspring = NULL,
  filter.non.conforming = TRUE,
  elim.redundant = TRUE,
  parent.geno = c("joint", "max"),
  thresh.parent.geno = 0.95,
  prob.thres = 0.95,
  file.type = c("table", "csv"),
  verbose = TRUE
)
```
Arguments

file.in: a character string with the name of (or full path to) the input file
ploidy: the ploidy level
parent1: a character string containing the name (or pattern of genotype IDs) of parent 1
parent2: a character string containing the name (or pattern of genotype IDs) of parent 2
offspring: a character string containing the name (or pattern of genotype IDs) of the offspring individuals. If NULL (default) it considers all individuals as offsprings, except parent1 and parent2.
filter.non.conforming: if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function segreg_poly for information on expected classes and their respective frequencies.
elim.redundant: logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to in order to include them in the final map.
parent.geno: indicates whether to use the joint probability 'joint' (default) or the maximum probability of multiple replicates (if available) to assign dosage to parents. If there is one observation per parent, both options will yield the same results.
thresh.parent.geno: threshold probability to assign a dosage to parents. If the probability is smaller than thresh.parent.geno, the marker is discarded.
prob.thres: threshold probability to assign a dosage to offspring. If the probability is smaller than prob.thres, the data point is converted to 'NA'.
file.type: indicates whether the characters in the input file are separated by 'white spaces' ("table") or by commas ("csv").
verbose: if TRUE (default), the current progress is shown; if FALSE, no output is produced

Value

An object of class mappoly.data which contains a list with the following components:

ploidy: ploidy level
n.ind: number individuals
n.mrk: total number of markers
ind.names: the names of the individuals
mrk.names: the names of the markers
dosage.p1: a vector containing the dosage in parent P for all n.mrk markers
dosage.p2: a vector containing the dosage in parent Q for all n.mrk markers
chrom: a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
genome.pos: Physical position of the markers into the sequence
seq.ref: NULL (unused in this type of data)
seq.alt: NULL (unused in this type of data)
**read_geno**

### Data Input

Reads an external data file. The format of the file is described in the Details section. This function creates an object of class `mappoly.data`.

#### Description

- `all.mrk.depth`: NULL (unused in this type of data)
- `geno.dose`: a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by `ploidy_level + 1`
- `n.phen`: number of phenotypic traits
- `phen`: a matrix containing the phenotypic data. The rows correspond to the traits and the columns correspond to the individuals
- `kept`: if `elim.redundant = TRUE`, holds all non-redundant markers
- `elim.correspondence`: if `elim.redundant = TRUE`, holds all non-redundant markers and its equivalence to the redundant ones

#### Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

#### References


#### Examples

```r
#### Tetraploid Example
ft <- "https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/fitpoly.dat"
tempfl <- tempfile()
download.file(ft, destfile = tempfl)
fitpoly.dat <- read_fitpoly(file.in = tempfl, ploidy = 4, parent1 = "P1", parent2 = "P2", verbose = TRUE)
print(fitpoly.dat, detailed = TRUE)
plot(fitpoly.dat)
plot_mrk_info(fitpoly.dat, 37)
```
Usage

```r
read_geno(
  file.in,  # a character string with the name of (or full path to) the input file which contains
  filter.non.conforming = TRUE,  # if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function segreg_poly for information on expected classes and their respective frequencies.
  elim.redundant = TRUE,  # logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.
  verbose = TRUE  # if TRUE (default), the current progress is shown; if FALSE, no output is produced
)
```

Arguments

- `file.in`: a character string with the name of (or full path to) the input file which contains
  the data to be read
- `filter.non.conforming`: if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function `segreg_poly` for information on expected classes and their respective frequencies.
- `elim.redundant`: logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.
- `verbose`: if TRUE (default), the current progress is shown; if FALSE, no output is produced
- `x`: an object of class `mappoly.data`
- `detailed`: if available, print the number of markers per sequence (default = FALSE)
- `thresh.line`: position of a threshold line for p values of the segregation test (default = 1e-06)

Details

The first line of the input file contains the string `ploidy` followed by the ploidy level of the parents. The second and third lines contain the strings `n.ind` and `n.mrk` followed by the number of individuals in the dataset and the total number of markers, respectively. Lines number 4 and 5 contain the strings `mrk.names` and `ind.names` followed by a sequence of the names of the markers and the name of the individuals, respectively. Lines 6 and 7 contain the strings `dosageP` and `dosageQ` followed by a sequence of numbers containing the dosage of all markers in parent P and Q. Line 8, contains the string `seq` followed by a sequence of integer numbers indicating the chromosome each marker belongs. It can be any 'a priori' information regarding the physical distance between markers. For example, these numbers could refer to chromosomes, scaffolds or even contigs, in which the markers are positioned. If this information is not available for a particular marker, NA should be used. If this information is not available for any of the markers, the string `seq` should be followed by a single NA. Line number 9 contains the string `seqpos` followed by the physical position of the markers into the sequence. The physical position can be given in any unity of physical genomic distance (base pairs, for instance). However, the user should be able to make decisions based on these values, such as the occurrence of crossing overs, etc. Line number 10 should contain
the string nphen followed by the number of phenotypic traits. Line number 11 is skipped (Usually used as a spacer). The next elements are strings containing the name of the phenotypic trait with no space characters followed by the phenotypic values. The number of lines should be the same number of phenotypic traits. NA represents missing values. The line number 12 + nphen is skipped. Finally, the last element is a table containing the dosage for each marker (rows) for each individual (columns). NA represents missing values.

Value

An object of class mappoly.data which contains a list with the following components:

- ploidy: ploidy level
- n.ind: number individuals
- n.mrk: total number of markers
- ind.names: the names of the individuals
- mrk.names: the names of the markers
- dosage.p1: a vector containing the dosage in parent P for all n.mrk markers
- dosage.p2: a vector containing the dosage in parent Q for all n.mrk markers
- chrom: a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
- genome.pos: Physical position of the markers into the sequence
- seq.ref: NULL (unused in this type of data)
- seq.alt: NULL (unused in this type of data)
- all.mrk.depth: NULL (unused in this type of data)
- geno.dose: a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1
- n.phen: number of phenotypic traits
- phen: a matrix containing the phenotypic data. The rows correspond to the traits and the columns correspond to the individuals
- kept: if elim.redundant = TRUE, holds all non-redundant markers
- elim.correspondence: if elim.redundant = TRUE, holds all non-redundant markers and its equivalence to the redundant ones

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

#### Tetraploid Example
fl1 = "https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/SolCAP_dosage"

tempfl <- tempfile()
download.file(fl1, destfile = tempfl)
SolCAP.dose <- read_geno(file.in = tempfl)
print(SolCAP.dose, detailed = TRUE)
plot(SolCAP.dose)

---

read_geno_csv  
*Data Input in CSV format*

Description

Reads an external comma-separated values (CSV) data file. The format of the file is described in the Details section. This function creates an object of class mappoly.data.

Usage

```r
read_geno_csv(
  file.in,
  ploidy,
  filter.non.conforming = TRUE,
  elim.redundant = TRUE,
  verbose = TRUE
)
```

Arguments

- `file.in` a character string with the name of (or full path to) the input file containing the data to be read
- `ploidy` the ploidy level
- `filter.non.conforming` if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function `segreg_poly` for information on expected classes and their respective frequencies.
- `elim.redundant` logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.
- `verbose` if TRUE (default), the current progress is shown; if FALSE, no output is produced
read_geno_csv

Details

This is an alternative and a somewhat more straightforward version of the function read_geno. The input is a standard CSV file where the rows represent the markers, except for the first row which is used as a header. The first five columns contain the marker names, the dosage in parents 1 and 2, the chromosome information (i.e. chromosome, scaffold, contig, etc) and the position of the marker within the sequence. The remaining columns contain the dosage of the full-sib population. A tetraploid example of such file can be found in the Examples section.

Value

An object of class mappoly.data which contains a list with the following components:

- ploidy: ploidy level
- n.ind: number individuals
- n.mrk: total number of markers
- ind.names: the names of the individuals
- mrk.names: the names of the markers
- dosage.p1: a vector containing the dosage in parent P for all n.mrk markers
- dosage.p2: a vector containing the dosage in parent Q for all n.mrk markers
- chrom: a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
- genome.pos: Physical position of the markers into the sequence
- seq.ref: NULL (unused in this type of data)
- seq.alt: NULL (unused in this type of data)
- all.mrk.depth: NULL (unused in this type of data)
- geno.dose: a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1
- n.phen: number of phenotypic traits
- phen: a matrix containing the phenotypic data. The rows correspond to the traits and the columns correspond to the individuals
- kept: if elim.redundant = TRUE, holds all non-redundant markers
- elim.correspondence: if elim.redundant = TRUE, holds all non-redundant markers and its equivalence to the redundant ones

Author(s)

Marcelo Mollinari, <mollin@ncsu.edu>, with minor changes by Gabriel Gesteira, <gdesiqu@ncsu.edu>
References


Examples

```r
#### Tetraploid Example
ft = "https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/tetra_solcap.csv"
tempfl <- tempfile()
download.file(ft, destfile = tempfl)
SolCAP.dose <- read_geno_csv(file.in = tempfl, ploidy = 4)
print(SolCAP.dose, detailed = TRUE)
plot(SolCAP.dose)
```

---

read_geno_prob  Data Input

Description

Reads an external data file. The format of the file is described in the Details section. This function creates an object of class mappoly.data

Usage

```r
read_geno_prob(
  file.in,
  prob.thres = 0.95,
  filter.non.conforming = TRUE,
  elim.redundant = TRUE,
  verbose = TRUE
)
```

Arguments

- `file.in` a character string with the name of (or full path to) the input file which contains the data to be read
- `prob.thres` probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than `prob.thres` are considered as missing data for the dosage calling purposes (default = 0.95)
filter.non.conforming
if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function `segreg.poly` for information on expected classes and their respective frequencies.

elim.redundant
logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.

verbose
if TRUE (default), the current progress is shown; if FALSE, no output is produced.

Details

The first line of the input file contains the string `ploidy` followed by the ploidy level of the parents. The second and third lines contain the strings `n.ind` and `n.mrk` followed by the number of individuals in the dataset and the total number of markers, respectively. Lines number 4 and 5 contain the string `mrk.names` and `ind.names` followed by a sequence of the names of the markers and the name of the individuals, respectively. Lines 6 and 7 contain the strings `dosageP` and `dosageQ` followed by a sequence of numbers containing the dosage of all markers in parent P and Q. Line 8, contains the string `seq` followed by a sequence of integer numbers indicating the chromosome each marker belongs. It can be any 'a priori' information regarding the physical distance between markers. For example, these numbers could refer to chromosomes, scaffolds or even contigs, in which the markers are positioned. If this information is not available for a particular marker, NA should be used. If this information is not available for any of the markers, the string `seq` should be followed by a single `NA`. Line number 9 contains the string `seqpos` followed by the physical position of the markers into the sequence. The physical position can be given in any unity of physical genomic distance (base pairs, for instance). However, the user should be able to make decisions based on these values, such as the occurrence of crossing overs, etc. Line number 10 should contain the string `nphen` followed by the number of phenotypic traits. Line number 11 is skipped (Usually used as a spacer). The next elements are strings containing the name of the phenotypic trait with no space characters followed by the phenotypic values. The number of lines should be the same number of phenotypic traits. NA represents missing values. The line number 12 + `nphen` is skipped. Finally, the last element is a table containing the probability distribution for each combination of marker and offspring. The first two columns represent the marker and the offspring, respectively. The remaining elements represent the probability associated with each one of the possible dosages. NA represents missing data.

Value

an object of class `mappoly.data` which contains a list with the following components:

- **ploidy**: ploidy level
- **n.ind**: number individuals
- **n.mrk**: total number of markers
- **ind.names**: the names of the individuals
- **mrk.names**: the names of the markers
- **dosage.p1**: a vector containing the dosage in parent P for all `n.mrk` markers
- **dosage.p2**: a vector containing the dosage in parent Q for all `n.mrk` markers
- **chrom**: a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
- **genome.pos**: physical position of the markers into the sequence
seq.ref  NULL (unused in this type of data)
seq.alt   NULL (unused in this type of data)
all.mrk.depth NULL (unused in this type of data)
prob.thres probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than ‘prob.thres’ were considered as missing data in the ‘geno.dose’ matrix

geno.dose  a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1

prob.thres  probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than ‘prob.thres’ were considered as missing data in the ‘geno.dose’ matrix

geno.dose  a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1

prob.thres  probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than ‘prob.thres’ were considered as missing data in the ‘geno.dose’ matrix

Author(s)
Marcelo Mollinari, <mmollin@ncsu.edu>

References


Examples

#### Tetraploid Example
```r
ft = "https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/hexa_sample"
tempfl <- tempfile()
download.file(ft, destfile = tempfl)
SolCAP.dose.prob <- read_geno_prob(file.in = tempfl)
print(SolCAP.dose.prob, detailed = TRUE)
plot(SolCAP.dose.prob)
```
Description

Reads an external VCF file and creates an object of class `mappoly.data`.

Usage

```r
read_vcf(
  file.in,  # a character string with the name of (or full path to) the input file which contains the data (VCF format)
  parent.1,  # a character string containing the name of parent 1
  parent.2,  # a character string containing the name of parent 2
  ploidy = NA,  # the species ploidy (optional, it will be automatically detected)
  filter.non.conforming = TRUE,  # if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function `segreg_poly` for information on expected classes and their respective frequencies.
  thresh.line = 0.05,  # threshold used for p-values on segregation test (default = 0.05)
  min.gt.depth = 0,  # minimum genotype depth to keep information. If the genotype depth is below `min.gt.depth`, it will be replaced with NA (default = 0)
  min.av.depth = 0,  # minimum average depth to keep markers (default = 0)
  max.missing = 1,  # maximum proportion of missing data to keep markers (range = 0-1; default = 1)
  elim.redundant = TRUE,  # logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.
  verbose = TRUE,  # if TRUE (default), the current progress is shown; if FALSE, no output is produced
  read.geno.prob = FALSE,  # if TRUE (default), converts genotype probabilities to probabilities (optional, it will be automatically detected)
  prob.thres = 0.95  # if TRUE (default), converts genotype probabilities to probabilities (optional, it will be automatically detected)
)
```

Arguments

- `file.in`: a character string with the name of (or full path to) the input file which contains the data (VCF format).
- `parent.1`: a character string containing the name of parent 1.
- `parent.2`: a character string containing the name of parent 2.
- `ploidy`: the species ploidy (optional, it will be automatically detected).
- `filter.non.conforming`: if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function `segreg_poly` for information on expected classes and their respective frequencies.
- `thresh.line`: threshold used for p-values on segregation test (default = 0.05).
- `min.gt.depth`: minimum genotype depth to keep information. If the genotype depth is below `min.gt.depth`, it will be replaced with NA (default = 0).
- `min.av.depth`: minimum average depth to keep markers (default = 0).
- `max.missing`: maximum proportion of missing data to keep markers (range = 0-1; default = 1).
- `elim.redundant`: logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.
- `verbose`: if TRUE (default), the current progress is shown; if FALSE, no output is produced.
- `read.geno.prob`: if TRUE (default), converts genotype probabilities to probabilities (optional, it will be automatically detected).
- `prob.thres`: if TRUE (default), converts genotype probabilities to probabilities (optional, it will be automatically detected).
read.geno.prob  If genotypic probabilities are available (PL field), generates a probability-based dataframe (default = FALSE).

prob.thres    probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than prob.thres are considered as missing data for the dosage calling purposes (default = 0.95)

Details

This function can handle .vcf files versions 4.0 or higher. The ploidy can be automatically detected, but it is highly recommended that you inform it to check for mismatches. All individual and marker names will be kept as they are in the .vcf file.

Value

An object of class mappoly.data which contains a list with the following components:

- **ploidy**  ploidy level
- **n.ind**  number individuals
- **n.mrk**  total number of markers
- **ind.names**  the names of the individuals
- **mrk.names**  the names of the markers
- **dosage.p1**  a vector containing the dosage in parent P for all n.mrk markers
- **dosage.p2**  a vector containing the dosage in parent Q for all n.mrk markers
- **chrom**  a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
- **genome.pos**  Physical position of the markers into the sequence
- **seq.ref**  Reference base used for each marker (i.e. A, T, C, G)
- **seq.alt**  Alternative base used for each marker (i.e. A, T, C, G)
- **prob.thres**  (unused field)
- **geno.dose**  a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1
- **geno**  a dataframe containing all genotypic probabilities columns for each marker and individual combination (rows). Missing data are represented by ploidy_level + 1
- **nphen**  (unused field)
- **phen**  (unused field)
- **all.mrk.depth**  DP information for all markers on VCF file
- **chisq.pval**  a vector containing p-values related to the chi-squared test of Mendelian segregation performed for all markers
- **kept**  if elim.redundant = TRUE, holds all non-redundant markers
- **elim.correspondence**  if elim.redundant = TRUE, holds all non-redundant markers and its equivalence to the redundant ones
reest_rf

Author(s)
Gabriel Gesteira, <gdesiqu@ncsu.edu>

References


Examples

```r
## Hexaploid sweetpotato: Subset of chromosome 3
fl = "https://github.com/mmollina/MAPpoly_vignettes/raw/master/data/sweet_sample_ch3.vcf.gz"
fl <- tempfile(pattern = "chr3.", fileext = ".vcf.gz")
download.file(fl, destfile = fl)
dat.dose.vcf = read_vcf(file = fl, parent.1 = "PARENT1", parent.2 = "PARENT2")
print(dat.dose.vcf)
plot(dat.dose.vcf)
```

---

reest_rf

**Re-estimate the recombination fractions in a genetic map**

Description

This function re-estimates the recombination fractions between all markers in a given map.

Usage

```r
reest_rf(
  input.map,
  input.mat = NULL,
  tol = 0.01,
  phase.config = "all",
  method = c("hmm", "ols", "wMDS_to_1D_pc"),
  weight = TRUE,
  verbose = TRUE,
  high.prec = FALSE,
  max.rf.to.break.EM = 0.5,
  input.mds = NULL
)```

Arguments

input.map An object of class mappoly.map
input.mat An object of class mappoly.rf.matrix
tol tolerance for determining convergence (default = 10e-03)
phase.config which phase configuration should be used. "best" (default) will choose the maximum likelihood configuration
method indicates whether to use 'hmm' (Hidden Markov Models), 'ols' (Ordinary Least Squares) or 'wMDS_to_1D_pc' (weighted MDS followed by fitting a one dimensional principal curve) to re-estimate the recombination fractions.
weight if TRUE (default), it uses the LOD scores to perform a weighted regression when the Ordinary Least Squares is chosen
verbose if TRUE (default), current progress is shown; if FALSE, no output is produced
high.prec logical. If TRUE uses high precision (long double) numbers in the HMM procedure implemented in C++, which can take a long time to perform (default = FALSE)
max.rf.to.break.EM for internal use only.
input.mds An object of class mappoly.map

Value

An updated object of class mappoly.pcmap whose order was used in the input.map

References


Description

Provides the reverse of a given map.

Usage

rev_map(input.map)

Arguments

input.map an object of class mappoly.map
rf_list_to_matrix

Author(s)
Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

plot_genome_vs_map(solcap.mds.map[[1]])
plot_genome_vs_map(rev_map(solcap.mds.map[[1]]))

---

rf_list_to_matrix  Recombination fraction list to matrix

Description

Transforms the recombination fraction list contained in an object of class mappoly.twopt or mappoly.twopt2 into a recombination fraction matrix

Usage

rf_list_to_matrix(
  input.twopt,
  thresh.LOD.ph = 0,
  thresh.LOD.rf = 0,
  thresh.rf = 0.5,
  ncpus = 1L,
  shared.alleles = FALSE,
  verbose = TRUE
)

## S3 method for class 'mappoly.rf.matrix'
print(x, ...)

## S3 method for class 'mappoly.rf.matrix'
plot(
  x,
  type = c("rf", "lod"),
  ord = NULL,
  rem = NULL,
  main.text = NULL,
  index = FALSE,
  fact = 1,
  ...
)

rf_list_to_matrix

Arguments

- **input.twopt**: an object of class `mappoly.twopt` or `mappoly.twopt2`
- **thresh.LOD.ph**: LOD score threshold for linkage phase configurations (default = 0)
- **thresh.LOD.rf**: LOD score threshold for recombination fractions (default = 0)
- **thresh.rf**: the threshold used for recombination fraction filtering (default = 0.5)
- **ncpus**: number of parallel processes (i.e. cores) to spawn (default = 1)
- **shared.alleles**: if TRUE, computes two matrices (for both parents) indicating the number of homologues that share alleles (default = FALSE)
- **verbose**: if TRUE (default), current progress is shown; if FALSE, no output is produced
- **x**: an object of class `mappoly.rf.matrix`
- **...**: currently ignored
- **type**: type of matrix that should be printed. Can be one of the following: "rf", for recombination fraction or "lod" for LOD Score
- **ord**: the order in which the markers should be plotted (default = NULL)
- **rem**: which markers should be removed from the heatmap (default = NULL)
- **main.text**: a character string as the title of the heatmap (default = NULL)
- **index**: logical should the name of the markers be printed in the diagonal of the heatmap? (default = FALSE)
- **fact**: positive integer. factor expressed as number of cells to be aggregated (default = 1, no aggregation)

Details

**thresh.LOD.ph** should be set in order to only select recombination fractions that have LOD scores associated to the linkage phase configuration higher than **thresh.LOD.ph** when compared to the second most likely linkage phase configuration.

Value

A list containing two matrices. The first one contains the filtered recombination fraction and the second one contains the information matrix.

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Examples

```r
all.mrk <- make_seq_mappoly(hexafake, 1:20)
red.mrk <- elim_redundant(all.mrk)
unique.mrks <- make_seq_mappoly(red.mrk)
all.pairs <- est_pairwise_rf(input.seq = unique.mrks,
  ncpus = 1,
  verbose = TRUE)

## Full recombination fraction matrix
mat.full <- rf_list_to_matrix(input.twopt = all.pairs)
plot(mat.full)
plot(mat.full, type = "lod")
```

---

**rf_snp_filter**

Remove markers that do not meet a LOD criteria

**Description**

Remove markers that do not meet a LOD and recombination fraction criteria for at least a percentage of the pairwise marker combinations. It also removes markers with strong evidence of linkage across the whole linkage group (false positive).

**Usage**

```r
rf_snp_filter(
  input.twopt,
  thresh.LOD.ph = 5,
  thresh.LOD.rf = 5,
  thresh.rf = 0.15,
  probs = c(0.05, 1),
  diag.markers = NULL,
  mrk.order = NULL,
  ncpus = 1L,
  diagnostic.plot = TRUE,
  breaks = 100
)
```

**Arguments**

- `input.twopt` an object of class `mappoly.twopt`
- `thresh.LOD.ph` LOD score threshold for linkage phase configuration (default = 5)
- `thresh.LOD.rf` LOD score threshold for recombination fraction (default = 5)
- `thresh.rf` threshold for recombination fractions (default = 0.15)
- `probs` indicates the probability corresponding to the filtering quantiles. (default = `c(0.05, 1)`)
diag.markers  A window where marker pairs should be considered. If NULL (default), all markers are considered.
mrk.order  marker order. Only has effect if `diag.markers’ is not NULL
ncpus  number of parallel processes (i.e. cores) to spawn (default = 1)
diagnostic.plot  if TRUE produces a diagnostic plot
breaks  number of cells for the histogram

Details

thresh.LOD.ph should be set in order to only select recombination fractions that have LOD scores associated to the linkage phase configuration higher than thresh.LOD.ph when compared to the second most likely linkage phase configuration. That action usually eliminates markers that are unlinked to the set of analyzed markers.

Value

A filtered object of class mappoly.sequence. See make_seq_mappoly for details

Author(s)

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

References


Examples

```r
all.mrk <- make_seq_mappoly(hexafake, 1:20)
red.mrk <- elim_redundant(all.mrk)
unique.mrks <- make_seq_mappoly(red.mrk)
all.pairs <- est_pairwise_rf(input.seq = unique.mrks,
                           ncpus = 1,
                           verbose = TRUE)

## Full recombination fraction matrix
mat.full <- rf_list_to_matrix(input.twopt = all.pairs)
plot(mat.full)

## Removing disruptive SNPs
tpt.filt <- rf_snp_filter(all.pairs, 2, 2, 0.07, probs = c(0.15, 1))
p1.filt <- make_pairs_mappoly(input.seq = tpt.filt, input.twopt = all.pairs)
m1.filt <- rf_list_to_matrix(input.twopt = p1.filt)
plot(mat.full, main.text = "LG1")
plot(m1.filt, main.text = "LG1.filt")
```
**segreg_poly**

*Polysomic segregation frequency*

**Description**

Computes the polysomic segregation frequency given a ploidy level and the dosage of the locus in both parents. It does not consider double reduction.

**Usage**

```r
segreg_poly(ploidy, dP, dQ)
```

**Arguments**

- **ploidy**: the ploidy level
- **dP**: the dosage in parent P
- **dQ**: the dosage in parent Q

**Value**

A vector containing the expected segregation frequency for all possible genotypic classes.

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**References**


**Examples**

```r
# autohexaploid with two and three doses in parents P and Q, 
# respectively
seg <- segreg_poly(ploidy = 6, dP = 2, dQ = 3)
barplot(seg, las = 2)
```
**sim_homologous**  
*Simulate homology groups*

**Description**

Simulate two homology groups (one for each parent) and their linkage phase configuration.

**Usage**

```r
sim_homologous(
  ploidy,  
  n.mrk,  
  min.d = 0,  
  max.d = ploidy + 1,  
  prob.dose = NULL,  
  max.ph,  
  restriction = TRUE,  
  seed = NULL
)
```

**Arguments**

- **ploidy**: ploidy level. Must be an even number
- **n.mrk**: number of markers
- **min.d**: minimum dosage to be simulated (default = 0)
- **max.d**: maximum dosage to be simulated (default = ploidy + 1)
- **prob.dose**: a vector indicating the proportion of markers for different dosage to be simulated (default = NULL)
- **max.ph**: maximum phase difference
- **restriction**: if TRUE (default), avoid cases where it is impossible to estimate recombination fraction and/or linkage phases via two-point analysis
- **seed**: random number generator seed

**Details**

This function prevents the simulation of linkage phase configurations which are impossible to estimate via two point methods.

**Value**

a list containing the following components:

- **hom.allele.p**: a list of vectors containing linkage phase configurations. Each vector contains the numbers of the homologous chromosomes in which the alleles are located. For instance, a vector containing (1, 3, 4) means that the marker has three doses located in the chromosomes 1, 3 and 4. For zero doses, use 0
solcap.dose.map

\( p \) contains the indices of the starting positions of the dosages, considering that the vectors contained in \( p \) are concatenated. Markers with no doses (zero doses are also considered)

\( \text{hom.allele.q} \) Analogously to \( \text{hom.allele.p} \)

\( q \) Analogously to \( p \)

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**References**


**Examples**

```r
h.temp <- sim_homologous(ploidy = 6, n.mrk = 20, max.d = 3, max.ph = 3,
                       seed = 123)
```

---

**Description**

A list containing 12 linkage groups estimated using genomic order and dosage call

**Usage**

```r
solcap.dose.map
```

**Format**

A list containing 12 objects of class `mappoly.map`, each one representing one linkage group in the `tetra.solcap` dataset.
solcap.mds.map

---

**solcap.err.map**  
*Resulting maps from tetra.solcap*

---

**Description**

A list containing 12 linkage groups estimated using genomic order, dosage call and global calling error

**Usage**

solcap.err.map

**Format**

A list containing 12 objects of class mappoly.map, each one representing one linkage group in the tetra.solcap dataset.

---

**solcap.mds.map**  
*Resulting maps from tetra.solcap*

---

**Description**

A list containing 12 linkage groups estimated using mds_mappoly order and dosage call

**Usage**

solcap.mds.map

**Format**

A list containing 12 objects of class mappoly.map, each one representing one linkage group in the tetra.solcap dataset.
solcap.prior.map  

**Resulting maps from tetra.solcap.geno.dist**

**Description**

A list containing 12 linkage groups estimated using genomic order and prior probability distribution.

**Usage**

```r
solcap.prior.map
```

**Format**

A list containing 12 objects of class mappoly.map, each one representing one linkage group in the tetra.solcap.geno.dist dataset.

---

split_and_rephase  

**Divides map in sub-maps and re-phase them**

**Description**

The function splits the input map in sub-maps given a distance threshold of neighboring markers and evaluates alternative phases between the sub-maps.

**Usage**

```r
split_and_rephase(
  input.map,  
  twopt,  
  gap.threshold = 5,  
  size.rem.cluster = 1,  
  phase.config = "best",  
  tol.final = 0.001,  
  verbose = TRUE
)
```

**Arguments**

- `input.map`: an object of class mappoly.map
- `twopt`: an object of class mappoly.twopt containing the two-point information for the markers contained in `input.map`
- `gap.threshold`: distance threshold of neighboring markers where the map should be spitted. The default value is 5 cM
size.rem.cluster
the size of the marker cluster (in number of markers) from which the cluster should be removed. The default value is 1

phase.config which phase configuration should be used. "best" (default) will choose the maximum likelihood phase configuration

tol.final the desired accuracy for the final map (default = 10e-04)

verbose if TRUE (default), the current progress is shown; if FALSE, no output is produced

Value
An object of class mappoly.map

Author(s)
Marcelo Mollinari, <mmollin@ncsu.edu>

References

Examples
map <- get_submap(solcap.dose.map[[1]], 1:20, verbose = FALSE)
tpt <- est_pairwise_rf(make_seq_mappoly(map))
new.map <- split_and_rephase(map, tpt, 1, 1)
map
new.map
plot_map_list(list(old.map = map, new.map = new.map), col = "ggstyle")

summary_maps
Summary maps

Description
This function generates a brief summary table of a list of mappoly.map objects

Usage
summary_maps(map.list, verbose = TRUE)

Arguments
map.list a list of objects of class mappoly.map
verbose if TRUE (default), the current progress is shown; if FALSE, no output is produced
tetra.solcap

Value

a data frame containing a brief summary of all maps contained in map.list

Author(s)

Gabriel Gesteira, <gdesiqu@ncsu.edu>

Examples

tetra.sum <- summary_maps(solcap.err.map)
tetra.sum

Description

A dataset of the B2721 population which derived from a cross between two tetraploid potato varieties: Atlantic × B1829-5. The population comprises 160 offsprings genotyped with the SolCAP Infinium 8303 potato array. The original data set can be found in [The Solanaceae Coordinated Agricultural Project (SolCAP) webpage](http://solcap.msu.edu/potato_infinium.shtml) The dataset also contains the genomic order of the SNPs from the Solanum tuberosum genome version 4.03. The genotype calling was performed using the fitPoly R package.

Usage

tetra.solcap

Format

An object of class mappoly.data which contains a list with the following components:

- **ploidy**  ploidy level = 4
- **n.ind**   number individuals = 160
- **n.mrk**   total number of markers = 4017
- **ind.names**  the names of the individuals
- **mrk.names**  the names of the markers
- **dosage.p1**  a vector containing the dosage in parent P for all n.mrk markers
- **dosage.p2**  a vector containing the dosage in parent Q for all n.mrk markers
- **chrom**   a vector indicating the chromosome each marker belongs. Zero indicates that the marker was not assigned to any sequence
- **genome.pos**  Physical position of the markers into the sequence
- **geno.dose**  a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by ploidy_level + 1 = 5
n.phen  There are no phenotypes in this simulation
phen There are no phenotypes in this simulation
chisq.pval vector containing p-values for all markers associated to the chi-square test for the expected segregation patterns under Mendelian segregation

tetra.solcap.geno.dist

*Autotetraploid potato dataset with genotype probabilities.*

**Description**

A dataset of the B2721 population which derived from a cross between two tetraploid potato varieties: Atlantic × B1829-5. The population comprises 160 offsprings genotyped with the SolCAP Infinium 8303 potato array. The original data set can be found in [The Solanaceae Coordinated Agricultural Project (SolCAP) webpage](http://solcap.msu.edu/potato_infinium.shtml) The dataset also contains the genomic order of the SNPs from the Solanum tuberosum genome version 4.03. The genotype calling was performed using the fitPoly R package. Although this dataset contains the probability distribution of the genotypes, it is essentially the same dataset found in `tetra.solcap`

**Usage**

tetra.solcap.geno.dist

**Format**

An object of class `mappoly.data` which contains a list with the following components:

- **ploidy**  ploidy level = 4
- **n.ind**  number individuals = 160
- **n.mrk**  total number of markers = 4017
- **ind.names**  the names of the individuals
- **mrk.names**  the names of the markers
- **dosage.p1**  a vector containing the dosage in parent P for all `n.mrk` markers
- **dosage.p2**  a vector containing the dosage in parent Q for all `n.mrk` markers
- **chrom**  a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
- **genome.pos**  Physical position of the markers into the sequence
- **prob.thres = 0.95**  probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than 'prob.thres' are considered as missing data for the dosage calling purposes
- **geno**  a data.frame containing the probability distribution for each combination of marker and offspring. The first two columns represent the marker and the offspring, respectively. The remaining elements represent the probability associated to each one of the possible dosages
**update_map**

**geno.dose** a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by \( ploidy\_level + 1 = 5 \)

**n.phen** There are no phenotypes in this simulation

**phen** There are no phenotypes in this simulation

---

**Description**

This function takes an object of class `mappoly.map` and checks for removed redundant markers in the original dataset. Once redundant markers are found, they are re-added to the map in their respective equivalent positions and another HMM round is performed.

**Usage**

```r
update_map(input.maps, verbose = TRUE)
```

**Arguments**

- `input.maps`: a single map or a list of maps of class `mappoly.map`
- `verbose`: if TRUE (default), shows information about each update process

**Value**

an updated map (or list of maps) of class `mappoly.map`, containing the original map(s) plus redundant markers

**Author(s)**

Gabriel Gesteira, <gdesiqu@ncsu.edu>

**Examples**

```r
orig.map <- solcap.err.map
up.map <- lapply(solcap.err.map, update_map)
summary_maps(orig.map)
summary_maps(up.map)
```
Description

Updates the missing data in the dosage matrix of an object of class mappoly.data given a new probability threshold.

Usage

update_missing(input.data, prob.thres = 0.95)

Arguments

- **input.data**: an object of class mappoly.data
- **prob.thres**: probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than 'prob.thres' are considered as missing data for the dosage calling purposes.

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

```r
data.updated = update_missing(hexafake.geno.dist, prob.thres = 0.5)
print(hexafake.geno.dist)
print(data.updated)
```
Index

* analysis
  * cache_counts_twopt, 6

* datasets
  hexafake, 43
  hexafake.geno.dist, 44
  maps.hexafake, 54
  solcap.dose.map, 87
  solcap.err.map, 88
  solcap.mds.map, 88
  solcap.prior.map, 89
  tetra.solcap, 91
  tetra.solcap.geno.dist, 92

* two-point
  * cache_counts_twopt, 6

add_marker, 4
  cache_counts_twopt, 6, 7, 22
  calc_genoprob, 8
  calc_genoprob_dist, 9
  calc_genoprob_error, 10
  calc_genoprob_one_parent, 11
  calc_homologprob, 13
  calc_prefpair_profiles, 14
  check_data_sanity, 15
  compare_maps, 15

drop_marker, 16
  elim_redundant, 17
  est_full_hmm_with_global_error, 18
  est_full_hmm_with_prior_prob, 20
  est_pairwise_rf, 22, 51
  est_pairwise_rf2, 24
  est_rf_hmm, 25
  est_rf_hmm_sequential, 26, 28, 37
  export_data_to_polymapR, 31
  export_map_list, 32
  export_qtlpoly, 33
  extract_map, 34

filter_individuals, 34
filter_missing, 35
filter_segregation, 36
find_blocks, 37
get_genomic_order, 39
get_submap, 39
get_tab_mrks, 41
group_mappoly, 42
hexafake, 43, 44, 54
hexafake.geno.dist, 44
import_data_from_polymapR, 45
import_from_updog, 46
import_phased_maplist_from_polymapR, 48
loglike_hmm, 49
make_mat_mappoly, 50
make_pairs_mappoly, 51
make_seq_mappoly, 52, 84
maps.hexafake, 54
mds_mappoly, 54, 88
merge_datasets, 56
merge_maps, 58
plot.mappoly.data (read_geno), 69
plot.mappoly.geno.ord (get_genomic_order), 39
plot.mappoly.homoprob, 60
plot.mappoly.map (est_rf_hmm), 25
plot.mappoly.prefpair.profiles, 61
plot.mappoly.rf.matrix (rf_list_to_matrix), 81
plot.mappoly.sequence (make_seq_mappoly), 52
plot_genome_vs_map, 62
plot_GIC, 63
plot_map_list, 63
plot_mrk_info, 64
poly_cross_simulate, 65
print.mappoly.data(read_geno), 69
print.mappoly.geno.ord
  (get_genomic_order), 39
print.mappoly.map(est_rf_hmm), 25
print.mappoly.pcmap(mds_mappoly), 54
print.mappoly.pcmap3d(mds_mappoly), 54
print.mappoly.rf.matrix
  (rf_list_to_matrix), 81
print.mappoly.sequence
  (make_seq_mappoly), 52
print_mrkl, 67
read_fitpoly, 67
read_geno, 66, 69, 73
read_geno_csv, 72
read_geno_prob, 74
read_vcf, 77
reest_rf, 79
rev_map, 80
rf_list_to_matrix, 4, 50, 81
rf_snp_filter, 83
saveMarkerModels, 67
segreg_poly, 48, 68, 70, 72, 75–77, 85
sim_homologous, 86
solcap.dose.map, 87
solcap.err.map, 88
solcap.mds.map, 88
solcap.prior.map, 89
split_and_rephase, 89
summary_maps, 90
tetra.solcap, 87, 88, 91, 92
tetra.solcap.geno.dist, 89, 92
update_map, 93
update_missing, 94