Package 'mappoly'

September 14, 2020

Type Package

Title Construction of Genetic Linkage Maps in Autopolyploids

Version 0.2.0

Maintainer Marcelo Mollinari <mmollin@ncsu.edu>

Description Software to construct genetic maps in autopolyploid full sib populations.

License GPL-3 | file LICENSE

LazyData TRUE

Depends R (>= 3.5.0)

Imports Rcpp (>= 0.12.6), RCurl, plotly, fields, ggpubr, ggsci, rstudioapi, plot3D, memuse, dplyr, crayon, cli, magrittr, reshape, ggplot2, smacof, princurve, dendextend

LinkingTo Rcpp

RoxygenNote 7.1.1

SystemRequirements C++11

Encoding UTF-8

Suggests testthat, RColorBrewer, updog

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

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

NeedsCompilation yes

Author Marcelo Mollinari [aut, cre] (<https://orcid.org/0000-0002-7001-8498>), Gabriel Gesteira [ctb] (<https://orcid.org/0000-0002-4106-7346>), Guilhereme Pereira [ctb] (<https://orcid.org/0000-0002-7106-8630>), Augusto Garcia [ctb] (<https://orcid.org/0000-0003-0634-3277>), Zhao-Bang Zeng [ctb] (<https://orcid.org/0000-0002-3115-1149>)

R topics documented:

`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

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

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

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:

m	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.p	a vector containing the dosage in parent 1 for all markers in the map
seq.dose.q	a vector containing the dosage in parent 2 for all markers in the map
sequence	a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, sequence = NULL
sequence.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>

Examples

```
## Not run:
sub.map<-get_submap(maps.hexafake[[1]], 1:50, 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)
 best.phase <- res.add.M_1$maps[[1]]$seq.ph
 names.id<-names(best.phase$P)
 plot_compare_haplotypes(m = 6,
                         hom.allele.p1 = best.phase$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_20" (middle) #######
mrk.to.remove <- "M_20"
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_20<-add_marker(input.map = input.map,
                        mrk = "M_20",
                        pos = "M_19",
                        rf.matrix = rf.matrix,
                        genoprob = genoprob,
                        tol = 10e-4)
 plot(res.add.M_20, mrk.names = TRUE)
 best.phase <- res.add.M_20$maps[[1]]$seq.ph
 names.id<-names(best.phase$P)
 plot_compare_haplotypes(m = 6,
                         hom.allele.p1 = best.phase$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_53" (last) #######
mrk.to.remove <- "M_53"
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_53<-add_marker(input.map = input.map,
                         mrk = "M_53",
                         pos = "M_52",
                         rf.matrix = rf.matrix,
                         genoprob = genoprob,
                         tol = 10e-4)
 plot(res.add.M_53, mrk.names = TRUE)
 best.phase <- res.add.M_53$maps[[1]]$seq.ph
 names.id<-names(best.phase$P)
 plot_compare_haplotypes(m = 6,
                          hom.allele.p1 = best.phase$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])

## End(Not run)
```

`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

```
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 a 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, <gabrielgesteira@usp.br>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
  all.mrk<-make_seq_mappoly(hexafake, 'all')
  ## local computation
  counts<-cache_counts_twopt(all.mrk, ncpus = 8)
 ## load from internal file of web-stored counts (especially important for high ploidy levels)
  counts.cached<-cache_counts_twopt(all.mrk, cached = TRUE)

## End(Not run)
```

---

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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
## 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]))

## hexaploid example
probs.h<-calc_genoprob(input.map = maps.hexafake[[1]],
                           verbose = TRUE)
probs.h
## displaying individual 1, 400 genotypic states
## (rows) across linkage group 1 (columns)
image(t(probs.h$probs[,,1]))

## End(Not run)
```

---

calc_genoprob_dist *Compute conditional probabilities of the genotypes using probability distribution of dosages*

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

```
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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
## 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]))

## hexaploid example
probs.h<-calc_genoprob_dist(input.map = maps.hexafake[[1]],
                              dat.prob = hexafake.geno.dist,
                              verbose = TRUE)
probs.h
## displaying individual 1, 400 genotypic states
## (rows) across linkage group 1 (columns)
image(t(probs.h$probs[,,1]))
```

```
## End(Not run)
```

`calc_genoprob_error` *Compute conditional probabilities of the genotypes using global error*

Description

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

Usage

```
calc_genoprob_error(
  input.map,
  step = 0,
  phase.config = "best",
  error = 0.01,
  th.prob = 0.95,
  restricted = TRUE,
  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 maximum likelihood configuration
`error`	the assumed global error rate (default = 0.01)
`th.prob`	the threshold for using global error or genotype probability distribution contained in the dataset (default = 0.95)
`restricted`	if `TRUE` (default), restricts the prior to the possible classes under Mendelian non double-reduced segregation given the parental dosages
`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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
    probs<-calc_genoprob(input.map = solcap.dose.map[[1]],
                         verbose = TRUE)
    probs.error<-calc_genoprob_error(input.map = solcap.err.map[[1]],
                         error = 0.05,
                         verbose = TRUE)
  op<-par(mfrow = c(2,1))
  ## Example: individual 11
  ind<-11
  ## posterior probabilities with no error modeling
  pr1<-probs$probs[,,ind]
  d1<-probs$map
  image(t(pr1),
        col = RColorBrewer::brewer.pal(n=9 , name = "YlOrRd"),
        axes=FALSE,
        xlab = "Markers",
        ylab = " ",
        main = paste("LG_1, ind ", ind))
  axis(side = 1, at = d1/max(d1),
       labels =rep("", length(d1)), las=2)
  axis(side = 2, at = seq(0,1,length.out = nrow(pr1)),
       labels = rownames(pr1), las=2, cex.axis=.5)

  ## posterior probabilities with error modeling
  pr2<-probs.error$probs[,,ind]
  d2<-probs.error$map
  image(t(pr2),
        col=RColorBrewer::brewer.pal(n=9 , name = "YlOrRd"),
        axes=FALSE,
        xlab = "Markers",
        ylab = " ",
        main = paste("LG_1, ind ", ind, " - w/ error"))
  axis(side = 1, at = d2/max(d2),
       labels =rep("", length(d2)), las=2)
  axis(side = 2, at = seq(0,1,length.out = nrow(pr2)),
       labels = rownames(pr2), las=2, cex.axis=.5)
  par(op)

## End(Not run)
```

`calc_homoprob` *Homolog probabilities*

Description

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

Usage

```
calc_homoprob(input.genoprobs)
```

Arguments

input.genoprobs	an object of class mappoly.genoprob

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari M., Olukolu B. A., Pereira G. da S., Khan A., Gemenet D., Yencho G. C., Zeng Z-B. (2020), Unraveling the Hexaploid Sweetpotato Inheritance Using Ultra-Dense Multilocus Mapping, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400620

Examples

```
## Not run:
  ## tetraploid solcap example
  w2<-lapply(solcap.dose.map, calc_genoprob)
  h.prob.solcap<-calc_homoprob(w2)
  print(h.prob.solcap)
  plot(h.prob.solcap, ind = "ind_10")
  plot(h.prob.solcap, stack = TRUE, ind = 5)
  plot(h.prob.solcap, stack = TRUE, ind = 5, lg = "all")

  w3<-lapply(solcap.err.map, calc_genoprob_error, error = 0.05)
  h.prob.solcap.err<-calc_homoprob(w3)
  plot(h.prob.solcap, lg = 1, ind = 100, use.plotly = FALSE)
  plot(h.prob.solcap.err, lg = 1, ind = 100, use.plotly = FALSE)

  ## hexaploid example
  w1 <- lapply(maps.hexafake, calc_genoprob)
  h.prob <- calc_homoprob(w1)
  print(h.prob)
  plot(h.prob)
  plot(h.prob, lg = 1, ind = 5, use.plotly = FALSE)
  plot(h.prob, lg = c(1,3), ind = 15, use.plotly = FALSE)
  plot(h.prob, lg = "all")


## End(Not run)
```

---

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)
```

Arguments

input.genoprobs	an object of class mappoly.genoprob

Author(s)

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

References

Mollinari M., Olukolu B. A., Pereira G. da S., Khan A., Gemenet D., Yencho G. C., Zeng Z-B. (2020), Unraveling the Hexaploid Sweetpotato Inheritance Using Ultra-Dense Multilocus Mapping, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400620

Examples

```
## Not run:
  ## hexaploid example
  w1 <- lapply(maps.hexafake, calc_genoprob)
  x1 <- calc_prefpair_profiles(w1)
  print(x1)
  plot(x1, min.y.prof = 0.05, max.y.prof = .15, thresh = 0.01)

  ## tetraploid example
  w2 <- lapply(solcap.err.map, calc_genoprob_error, error = 0.05)
  x2 <- calc_prefpair_profiles(w2)
  print(x2)
  plot(x2, min.y.prof = 0.15, max.y.prof = .50)
  plot(x2, type = "hom.pairs", min.y.prof = 0.15, max.y.prof = .50)

## End(Not run)
```

---

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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
#### 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)
check_data_sanity(SolCAP.dose)

#### Hexaploid example
fl2 = "https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/hexafake"
tempfl <- tempfile()
download.file(fl2, destfile = tempfl)
hexa.dose <- read_geno(file.in  = tempfl)
check_data_sanity(hexa.dose)

## End(Not run)
```

---

drop_marker *Remove markers from a map*

Description

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

Usage

```
drop_marker(input.map, mrk)
```

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

Value

an object of class `mappoly.map`

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

```
## Not run:
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)

## End(Not run)
```

---

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, <gabrielgesteira@usp.br>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
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

```
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:

`m`	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.p	a vector containing the dosage in parent 1 for all markers in the map
seq.dose.q	a vector containing the dosage in parent 2 for all markers in the map
sequence	a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, sequence = NULL
sequence.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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
  seq1.20<-make_seq_mappoly(hexafake, 1:20)
  counts<-cache_counts_twopt(seq1.20, cached = TRUE)
  subset.pairs<-est_pairwise_rf(seq1.20, counts)
  subset.map <- est_rf_hmm_sequential(input.seq  = seq1.20,
                                      thres.twopt = 5,
                                      thres.hmm = 10,
                                      extend.tail = 10,
                                      tol = 0.1,
                                      tol.final = 10e-4,
                                      twopt = subset.pairs,
                                      verbose = TRUE)
  subset.map
  plot(subset.map)
  subset.map.reest<-est_full_hmm_with_global_error(subset.map,
                                                   error=0.01,
```

```
                                                    tol=10e-4,
                                                    verbose = TRUE)
    subset.map.reest
    plot(subset.map.reest)

## End(Not run)
```

`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

```
est_full_hmm_with_prior_prob(
  input.map,
  dat.prob = NULL,
  phase.config = "best",
  tol = 0.001,
  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 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:

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

`m`	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.p`	a vector containing the dosage in parent 1 for all markers in the map
`seq.dose.q`	a vector containing the dosage in parent 2 for all markers in the map

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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
 ## Not run:
solcap.p<-vector("list", 12)
names(solcap.p)<-names(solcap.dose.map)
for(i in 1:12){
   cat("Lg ", i, "...")
   solcap.p[[i]] <- est_full_hmm_with_prior_prob(solcap.dose.map[[i]],
                                                 dat.prob = tetra.solcap.geno.dist,
                                                 verbose = FALSE)
   cat("\n")
}
w<-NULL
for(i in 1:12)
  w<-c(w, c(solcap.dose.map[i],
            solcap.p[i]))
names(w) <- apply(expand.grid(c("dose", "prior"), paste0("LG_", 1:12),
                              stringsAsFactors = FALSE)[,2:1], 1, paste,
                  collapse = "_")
op <- par(cex.axis = .7)
plot_map_list(w, horiz = FALSE, col = rep(gg_color_hue(2), 12))
```

```
 par(op)
 legend("bottomright", legend = c("Dosage based", "Prob. based"),
        pch=15, col = rep(gg_color_hue(2)))

## End(Not run)
```

`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

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

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.
`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`	The number of batches of marker pairs that should be analyzed in parallel. Using `n.batches` > 1, will usually result in more processing time. However, it will require less memory. See examples.
`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.
`tol`	the desired accuracy. See `optimize()` for details

Value

An object of class `poly.est.two.pts.pairwise` 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 `choose(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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
## Tetraploid example:
all.mrk <- make_seq_mappoly(tetra.solcap, 'all')
red.mrk <- elim_redundant(all.mrk)
unique.mrks <- make_seq_mappoly(red.mrk)
# will take ~ 13 min
all.pairs <- est_pairwise_rf(input.seq = unique.mrks,
                             ncpus = 7,
                             verbose=TRUE)
 all.pairs
 plot(all.pairs, 90, 91)
 mat <- rf_list_to_matrix(all.pairs)
 plot(mat)

 ## Hexaploid example
fl = "https://github.com/mmollina/MAPpoly_vignettes/raw/master/data/BT/sweetpotato_chr1.vcf.gz"
 tempfl <- tempfile(pattern = 'chr1_', fileext = '.vcf.gz')
 download.file(fl, destfile = tempfl)
 dat.dose.vcf = read_vcf(file = tempfl, parent.1 = "PARENT1", parent.2 = "PARENT2")

 ## Filtering dataset by marker
```

```
 dat.filt.mrk <- filter_missing(input.data = dat.dose.vcf,
                                type = "marker",
                                filter.thres = 0.10,
                                inter = FALSE)
 ## Filtering dataset by individual
 dat.filt.ind <- filter_missing(input.data = dat.filt.mrk,
                                type = "individual",
                                filter.thres = 0.10,
                                inter = FALSE)
## Segregation test
pval.bonf <- 0.05/dat.filt.ind$n.mrk
mrks.chi.filt <- filter_segregation(dat.filt.ind,
                                    chisq.pval.thres =  pval.bonf,
                                    inter = FALSE)
seq.ch1<-make_seq_mappoly(mrks.chi.filt)
plot(seq.ch1)
## will take ~  19 min / peak of memory usage ~ 10GB
all.pairs.1 <- est_pairwise_rf(input.seq = seq.ch1,
                               ncpus = 7,
                               verbose=TRUE)
## same thing, but it will take ~  21 min / peak of memory usage ~ 6GB
all.pairs.2 <- est_pairwise_rf(input.seq = seq.ch1,
                               ncpus = 7,
                               n.batch = 10,
                               verbose=TRUE)
 plot(all.pairs, 90, 91)
 mat <- rf_list_to_matrix(all.pairs.1)
 plot(mat)

## End(Not run)
```

`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

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

## S3 method for class 'mappoly.map'
plot(
  x,
  left.lim = 0,
  right.lim = Inf,
  phase = TRUE,
  mrk.names = FALSE,
  cex = 1,
  config = "best",
  ...
)
```

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 poly.est.two.pts.pairwise 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

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:

`m`	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.p`	a vector containing the dosage in parent 1 for all markers in the map
`seq.dose.q`	a vector containing the dosage in parent 2 for all markers in the map
`sequence`	a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, `sequence = NULL`
`sequence.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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
   mrk.subset<-make_seq_mappoly(hexafake, 1:50)
   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
   subset.map <- est_rf_hmm(input.seq = unique.mrks,
                            thres = 2,
                            twopt = subset.pairs,
                            verbose = TRUE,
                            tol = 0.1,
                            est.given.0.rf = FALSE)

   ## Re-estimating the map with the most likely configuration
   subset.map1 <- est_rf_hmm_single(input.seq = unique.mrks,
                                    input.ph.single = subset.map$maps[[1]]$seq.ph,
                                    tol = 10e-3,
                                    verbose = TRUE)

   subset.map$maps[[1]]$seq.ph <- subset.map1$seq.ph

   plot(subset.map)

    ## 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(m = 6, h1 = ph.P[names(ph.P.est)], h2 = ph.P.est)
    compare_haplotypes(m = 6, h1 = ph.Q[names(ph.Q.est)], h2 = ph.Q.est)

## End(Not run)
```

---

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

```
est_rf_hmm_sequential(
  input.seq,
  twopt,
  start.set = 4,
  thres.twopt = 5,
  thres.hmm = 50,
  extend.tail = NULL,
  phase.number.limit = Inf,
  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 `poly.est.two.pts.pairwise` 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. η 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`. If the size exceeds this limit, the marker will not be inserted. If `NULL`, then it will insert all markers (default = Inf)
`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`, then it will insert all markers (default = Inf)
`info.tail`	if `TRUE` (default), it uses the complete informative tail of the chain (i.e. number of markers where all homologous ($ploidy x 2$) 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	If TRUE, the expansion of the current submap is shown;
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:

m	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.p	a vector containing the dosage in parent 1 for all markers in the map
seq.dose.q	a vector containing the dosage in parent 2 for all markers in the map
sequence	a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, sequence = NULL
sequence.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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
   #### Autotetraploid example
   s1<-make_seq_mappoly(tetra.solcap, 'seq1')
   red.mrk<-elim_redundant(s1)
   s1.unique.mrks<-make_seq_mappoly(red.mrk)
   s1.pairs<-est_pairwise_rf(input.seq = s1.unique.mrks,
                             ncpus = 7,
                             verbose=TRUE)
   unique.gen.ord<-get_genomic_order(s1.unique.mrks)
   ## Selecting a subset of 100 markers at the beginning of chromosome 1
   s1.gen.subset<-make_seq_mappoly(tetra.solcap, rownames(unique.gen.ord)[1:100])
   s1.gen.subset.map <- est_rf_hmm_sequential(input.seq = s1.gen.subset,
                                              start.set = 10,
                                              thres.twopt = 10,
                                              thres.hmm = 10,
                                              extend.tail = 30,
                                              info.tail = TRUE,
                                              twopt = s1.pairs,
                                              sub.map.size.diff.limit = 5,
                                              phase.number.limit = 40,
                                             reestimate.single.ph.configuration = TRUE,
                                              tol = 10e-3,
                                              tol.final = 10e-5)
    print(s1.gen.subset.map, detailed = TRUE)
    plot(s1.gen.subset.map)
    plot(s1.gen.subset.map, phase = FALSE)

   #### Autohexaploid example
   mrk.subset<-make_seq_mappoly(hexafake, 1:50)
   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,
                                       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)

## 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(m = 6, h1 = ph.P[names(ph.P.est)], h2 = ph.P.est)
compare_haplotypes(m = 6, h1 = ph.Q[names(ph.Q.est)], h2 = ph.Q.est)

## End(Not run)
```

`export_data_to_polymapR` *Export data to* `polymapR`

Description

Export data to `polymapR`

Usage

```
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>

Examples

```
## Not run:
require(polymapR)
dat<-export_data_to_polymapR(hexafake)
F1checked <- checkF1(dosage_matrix = dat,
                     parent1 = "P1",
                     parent2 = "P2",
                     F1 = colnames(dat)[-c(1:2)],
                     polysomic = TRUE,
                     disomic = FALSE,
                     mixed = FALSE,
                     ploidy = 6)
 head(F1checked$checked_F1)
 PCA_progeny(dosage_matrix = dat,
             highlight = list(c("P1", "P2")),
```

```
             colors = "red")

## End(Not run)
```

`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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
export_map_list(solcap.err.map[[1]], file = "")
export_map_list(maps.hexafake)

## End(Not run)
```

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

```
 ## Not run:
 x <- maps.hexafake[[1]]$info$sequence.pos/1e6
 y <- extract_map(maps.hexafake[[1]])
 plot(y~x, ylab = "Map position (cM)", xlab = "Genome Position (Mbp)")

## End(Not run)
```

filter_missing *Filter missing genotypes*

Description

Excludes markers or individuals based on their proportion of missing data

Usage

```
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) '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 (default), it plots markers or individuals vs. frequency of missing data

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

```
## Not run:
    plot(tetra.solcap)
    dat.filt.mrk <- filter_missing(input.data = tetra.solcap,
                                   type = "marker",
                                   filter.thres = 0.1)
    plot(dat.filt.mrk)

## End(Not run)
```

---

filter_segregation *Filter markers based on chi-square test*

Description

This function filter 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 chromosome bivalent pairing and no double reduction.

Usage

```
filter_segregation(input.data, chisq.pval.thres = 1e-04, inter = TRUE)
```

Arguments

input.data	name of input object (class mappoly.data)
chisq.pval.thres	p-value threshold used for chi-square tests (default = 10e-05)
inter	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

```
## Not run:
    mrks.chi.filt <- filter_segregation(input.data = tetra.solcap,
                                        chisq.pval.thres = 0.05/mydata$n.mrk,
                                        inter = TRUE)
    seq.init<-make_seq_mappoly(mrks.chi.filt)

## End(Not run)
```

---

get_genomic_order *Get the genomic position of markers in a sequence*

Description

This functions gets the genomic position of markers in a sequence and return an ordered data frame with the name and position of each marker

Usage

```
get_genomic_order(input.seq)
```

Arguments

input.seq	a sequence object of class mappoly.sequence

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

```
## Not run:
s1<-make_seq_mappoly(tetra.solcap, "all")
o1<-get_genomic_order(s1)
head(o1)

## End(Not run)
```

---

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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
   ## selecting the 20 first markers in linkage group 1
   ####
   ## re-estimating recombination fractions
   submap1.lg1<-get_submap(input.map = maps.hexafake[[1]],
                           mrk.pos = 1:20, verbose = TRUE,
                           tol.final = 10e-3)
   ## re-estimating the recombination fractions and linkage phases
   submap2.lg1<-get_submap(input.map = maps.hexafake[[1]],
                           mrk.pos = 1:20, verbose = TRUE,
                           reestimate.phase = TRUE,
                           tol.final = 10e-3)
  ## no recombination fraction re-estimation
  submap3.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 = T, cex = .8)
 plot(submap2.lg1, mrk.names = T, cex = .8)
 plot(submap3.lg1, mrk.names = T, cex = .8)

## End(Not run)
```

---

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, <gabrielgesteira@usp.br>

group_mappoly *Assign markers to linkage groups*

Description

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

Usage

```
group_mappoly(
  input.mat,
  expected.groups = NULL,
  inter = TRUE,
  comp.mat = 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 sequence information is available (default = FALSE)
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

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
   all.mrk <- make_seq_mappoly(hexafake, 'all')
   red.mrk <- elim_redundant(all.mrk)
   unique.mrks <- make_seq_mappoly(red.mrk)
   counts <- cache_counts_twopt(unique.mrks, cached = TRUE)
   ##will take ~ 7 minutes
   all.pairs <- est_pairwise_rf(input.seq = unique.mrks,
                                count.cache = counts,
                                ncpus = 7,
                                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 = 3,
                        inter = TRUE,
                        comp.mat = TRUE, #this data has physical information
                        verbose = TRUE)
   lgs
   plot(lgs)
   lg1 <- make_seq_mappoly(lgs, 1)
   lg2 <- make_seq_mappoly(lgs, 2)
   lg3 <- make_seq_mappoly(lgs, 3)

   ##Plot matrices
   m1<-make_mat_mappoly(input.seq = lg1, input.mat = mat.full)
   m2<-make_mat_mappoly(input.seq = lg2, input.mat = mat.full)
   m3<-make_mat_mappoly(input.seq = lg3, input.mat = mat.full)
   op<-par(mfrow = c(1,3), pty = "s")
   plot(m1, main.text = "LG1", index = FALSE)
   plot(m2, main.text = "LG2", index = FALSE)
   plot(m3, main.text = "LG3", index = FALSE)
   par(op)

## End(Not run)
```

---

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:

m 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.p a vector containing the dosage in parent P for all `n.mrk` markers

dosage.q a vector containing the dosage in parent Q for all `n.mrk` markers

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

sequence.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:

m 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.p a vector containing the dosage in parent P for all `n.mrk` markers

dosage.q a vector containing the dosage in parent Q for all `n.mrk` markers

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

sequence.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

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

`import_data_from_polymapR`

Import data from polymapR

Description

Function to import datasets from polymapR

Usage

```
import_data_from_polymapR(
  input.data,
  ploidy,
  parent1 = "P1",
  parent2 = "P2",
  filter.non.conforming = TRUE
)
```

Arguments

`input.data`	a `polymapR` dataset
`ploidy`	the ploidy level
`parent1`	name of parent 1
`parent2`	name of parent 2
`filter.non.conforming`	if `TRUE` (default) exclude samples with non expected genotypes under no double reduction

Author(s)

Marcelo Mollinari <mmollin@ncsu.edu>

References

Bourke PM et al: (2019) PolymapR — linkage analysis and genetic map construction from F1 populations of outcrossing polyploids. _Bioinformatics_ 34:3496–3502. https://doi.org/10.1093/bioinformatics/bty1002

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
require(polymapR)
data("screened_data3")
mappoly.data <- import_data_from_polymapR(screened_data3, 4)
plot(mappoly.data)

## End(Not run)
```

`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`

Usage

```
import_from_updog(object, prob.thres = NULL, filter.non.conforming = FALSE)
```

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

Value

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

`m`	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.p	a vector containing the dosage in parent P for all n.mrk markers
dosage.q	a vector containing the dosage in parent Q for all n.mrk markers
sequence	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
sequence.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
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. 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, <gabrielgesteira@usp.br>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
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 = 4)
mydata = import_from_updog(mout)
mydata
plot(mydata)
mydata = import_from_updog(mout, filter.non.conforming = TRUE)
mydata
plot(mydata)

## End(Not run)
```

`import_phased_maplist_from_polymapR` *Import phased map list from polymapR*

Description

Function to import phased map lists from polymapR

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

Author(s)

Marcelo Mollinari <mmollin@ncsu.edu>

References

Bourke PM et al: (2019) PolymapR — linkage analysis and genetic map construction from F1 populations of outcrossing polyploids. _Bioinformatics_ 34:3496–3502. https://doi.org/10.1093/bioinformatics/bty1002

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
require(polymapR)
## Loading polymapR example
data("integrated.maplist", "screened_data3", "marker_assignments_P1","marker_assignments_P2")
maplist <- create_phased_maplist(maplist = integrated.maplist,
                                 dosage_matrix.conv = screened_data3,
                                 marker_assignment.1=marker_assignments_P1,
                                 marker_assignment.2=marker_assignments_P2,
                                 ploidy = 4)
 ## Importing polymapR dataset
 mappoly.data <- import_data_from_polymapR(screened_data3, 4)
 plot(mappoly.data)

 ## Importing polymapR phased maplist
 mappoly.maplist <- import_phased_maplist_from_polymapR(maplist, mappoly.data)
 plot_map_list(mappoly.maplist)
 ## plot phased map
 plot(mappoly.maplist[[1]])
```

```
## plot a segment of phased map (from 0 to 20 cM)
plot(mappoly.maplist[[1]], mrk.names = T, left.lim = 0, right.lim = 20, cex = .7)
plot(mappoly.maplist[[2]])
plot(mappoly.maplist[[3]])
plot(mappoly.maplist[[4]])
plot(mappoly.maplist[[5]])

## Computing conditional genotype probabilities
genoprob0 <- lapply(mappoly.maplist, calc_genoprob, step = 1)

## Computing preferential pairing profiles
pref.pair0 <- calc_prefpair_profiles(genoprob0)
plot(pref.pair0, min.y.prof = .25, max.y.prof = 0.4, P = "P1", Q = "P2")

## Computing homolog probabilities
h.prob0<-calc_homoprob(genoprob0)
plot(h.prob0, ind = "F1_030") ## plot haplotype of individual "F1_030"

#### Computing conditional genotype probabilities including error
genoprob1 <- lapply(mappoly.maplist, calc_genoprob_error, step = 1, error = 0.05)

## Computing preferential pairing profiles
pref.pair1 <- calc_prefpair_profiles(genoprob1)
plot(pref.pair1, min.y.prof = .25, max.y.prof = 0.4, P = "P1", Q = "P2")

## Computing homolog probabilities
h.prob1<-calc_homoprob(genoprob1)
plot(h.prob1, ind = "F1_030") ## plot haplotype of individual "F1_030"

#### Reestimating recombination fractions using HMM
cl <- parallel::makeCluster(5)
parallel::clusterEvalQ(cl, require(mappoly))
parallel::clusterExport(cl,  "mappoly.data")
reest.maps <- parallel::parLapply(cl, mappoly.maplist,
                                  est_full_hmm_with_global_error,
                                  error = 0.05)
parallel::stopCluster(cl)

## Computing conditional genotype probabilities
genoprob2 <- lapply(reest.maps, calc_genoprob_error, step = 1, error = 0.05)

## Computing preferential pairing profiles
pref.pair2 <- calc_prefpair_profiles(genoprob2)
plot(pref.pair2, min.y.prof = .25, max.y.prof = 0.4, P = "P1", Q = "P2")

## Computing homolog probabilities
h.prob2<-calc_homoprob(genoprob2)
plot(h.prob2, ind = "F1_030")

#### Reconstructing the map using MAPpoly
s <- make_seq_mappoly(mappoly.data, "all")
tpt <- est_pairwise_rf(input.seq = s, ncpus = 7)
mat <- rf_list_to_matrix(make_pairs_mappoly(tpt, s))
grs <- group_mappoly(input.mat = mat,
                     expected.groups = 5,
                     inter = TRUE)
grs
```

```
LG <- vector("list", 5)
op <- par(mfrow = c(2,3))
for(i in 1:5){
  s.temp <-  make_seq_mappoly(grs, arg = i)
  tpt.temp <- make_pairs_mappoly(tpt, s.temp)
  sf<-rf_snp_filter(input.twopt = tpt.temp,
                    thresh.LOD.ph = 1,
                    thresh.LOD.rf = 1,
                    thresh.perc = 0.02)
  M <- make_mat_mappoly(input.mat = mat, sf)
  o <- mds_mappoly(M)
  so<-make_seq_mappoly(o)
  plot(M, ord = so$seq.mrk.names, main.text = paste("LG", i), index = FALSE)
  LG[[i]] <- list(s = so, tpt = tpt.temp)
  cat("\n")
}
par(op)
MAPs <- vector("list", 5)
for(i in 1:5){
MAPs[[i]] <- est_rf_hmm_sequential(input.seq = LG[[i]]$s,
                                   start.set = 6,
                                   thres.twopt = 10,
                                   thres.hmm = 50,
                                   extend.tail = 30,
                                   twopt = LG[[i]]$tpt,
                                   verbose = TRUE,
                                   tol = 10e-2,
                                   tol.final = 10e-4,
                                   phase.number.limit = 20,
                                   sub.map.size.diff.limit =  5,
                                   info.tail = TRUE,
                                   reestimate.single.ph.configuration = TRUE)
}
cl <- parallel::makeCluster(5)
parallel::clusterEvalQ(cl, require(mappoly))
parallel::clusterExport(cl,  "mappoly.data")
recons.maps <- parallel::parLapply(cl, MAPs,
                                   est_full_hmm_with_global_error,
                                   error = 0.05)
parallel::stopCluster(cl)

## Comparing resulting maps
## polymapR
summary_maps(mappoly.maplist)

## MAPpoly
summary_maps(recons.maps)

## Computing conditional genotype probabilities
genoprob3 <- lapply(recons.maps,
                    calc_genoprob_error,
                    step = 1,
                    error = 0.05)

## Computing preferential pairing profiles
pref.pair3 <- calc_prefpair_profiles(genoprob3)
plot(pref.pair3, min.y.prof = .25, max.y.prof = 0.4, P = "P1", Q = "P2")
```

```
## Comparing homolog probabilities with different mapping approaches
h.prob3<-calc_homoprob(genoprob3)
## plot haplotype of individual 10 (polymapR)
plot(h.prob0, ind = "F1_030", use.plotly = FALSE)
## plot haplotype of individual 10 (polymapR + HMM error modeling)
plot(h.prob1, ind = "F1_030", use.plotly = FALSE)
## plot haplotype of individual 10 (reestimated: MAPpoly)
plot(h.prob2, ind = "F1_030", use.plotly = FALSE)
## plot haplotype of individual 10 (reconstructed: MAPpoly)
plot(h.prob3, ind = "F1_030", use.plotly = FALSE)

## End(Not run)
```

`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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
 hexa.map1<-reest_rf(solcap.dose.map[[1]], verbose = FALSE, tol = 10e-4)
 hexa.map2<-loglike_hmm(hexa.map1)
 hexa.map1;hexa.map2

## End(Not run)
```

make_mat_mappoly *Subset recombination fraction matrices*

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

```
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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
   # sequence with 100 markers
   mrk.seq<-make_seq_mappoly(hexafake, 1:100)
   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)

## End(Not run)
```

make_pairs_mappoly *Subset pairwise recombination fractions*

Description

Get a subset of an object of class `poly.est.two.pts.pairwise` (i.e. recombination fraction) and LOD score statistics for all possible linkage phase combinations based on a sequence of markers.

Usage

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

Arguments

`input.twopt`	an object of class `poly.est.two.pts.pairwise`
`input.seq`	an object of class `mappoly.sequence`, with a sequence of markers contained in `input.twopt`

Value

an object of class `poly.est.two.pts.pairwise` which is a subset of `input.twopt`. See `est_pairwise_rf` for details

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
   all.mrk<-make_seq_mappoly(hexafake, sort(sample(1:1500, 200)))
   red.mrk<-elim_redundant(all.mrk)
   unique.mrks<-make_seq_mappoly(red.mrk)
   all.pairs<-est_pairwise_rf(input.seq = unique.mrks,
                              ncpus = 7,
                              verbose=TRUE)

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

   lgs <- group_mappoly(input.mat = mat.full,
                        expected.groups = 3,
                        verbose=TRUE)
   lgs
   plot(lgs)
   lg1 <- make_seq_mappoly(lgs, 1)
```

```
lg2 <- make_seq_mappoly(lgs, 2)
lg3 <- make_seq_mappoly(lgs, 3)

##Plot matrices
p1<-make_pairs_mappoly(input.seq = lg1, input.twopt = all.pairs)
p2<-make_pairs_mappoly(input.seq = lg2, input.twopt = all.pairs)
p3<-make_pairs_mappoly(input.seq = lg3, input.twopt = all.pairs)

m1<-rf_list_to_matrix(input.twopt = p1)
m2<-rf_list_to_matrix(input.twopt = p2)
m3<-rf_list_to_matrix(input.twopt = p3)

op<-par(mfrow = c(1,3), pty = "s")
plot(m1, main.text = "LG1")
plot(m2, main.text = "LG2")
plot(m3, main.text = "LG3")
par(op)

## End(Not run)
```

---

make_seq_mappoly *Create a sequence of markers*

Description

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

Usage

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

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

## S3 method for class 'mappoly.sequence'
plot(x, ...)
```

Arguments

input.obj	an object of one of the following classes: mappoly.data, mappoly.map, mappoly.group, mappoly.unique.seq, mappoly.pcmap or mappoly.pcmap3d
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.pcmap, mappoly.pcmap3d or mappoly.unique.seq
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, <gabrielgesteira@usp.br>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
    all.mrk<-make_seq_mappoly(hexafake, 'all')
    seq1.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)
    #same thing
    (some.mrk.names<-hexafake$mrk.names[c(1,4,28,32,45)])
    some.mrk.2<-make_seq_mappoly(hexafake, some.mrk.names)
    identical(some.mrk.1, some.mrk.2)

    ## Removing redundant markers and makeing a new sequence
    red.mrk<-elim_redundant(all.mrk)
```

```
unique.mrks<-make_seq_mappoly(red.mrk)

## Making a sequence using the intersection between groups and genomic information
s <- make_seq_mappoly(tetra.solcap, 'all')
tpt <- est_pairwise_rf(input.seq = s,
                       ncpus = 7)
mat <- rf_list_to_matrix(tpt)
grs <- group_mappoly(input.mat = mat,
                     expected.groups = 12,
                     comp.mat = FALSE)
seq1 = make_seq_mappoly(grs, arg = 1, genomic.info = 1)

## End(Not run)
```

`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

```
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 `TetraploidSNPMap`, available under GNU GENERAL PUBLIC LICENSE, Version 3, at https://github.com/BiomathematicsAndStatisticsScotland/TetraploidSNPMap

Usage

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

```
)

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

## S3 method for class 'mappoly.pcmap3d'
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 TetraploidSNPMap codes

References

Preedy, K. F., & Hackett, C. A. (2016). A rapid marker ordering approach for high-density genetic linkage maps in experimental autotetraploid populations using multidimensional scaling. _Theoretical and Applied Genetics_, 129(11), 2117-2132. https://doi.org/10.1007/s00122-016-2761-8

Examples

```
## Not run:
   s1 <- make_seq_mappoly(hexafake, 'seq1')
   t1 <- est_pairwise_rf(s1, ncpus = 7)
   m1 <- rf_list_to_matrix(t1)
   plot(m1, ord = rownames(get_genomic_order(s1)))

   ## Removing disruptive SNPs
   s1f <- rf_snp_filter(t1, 5, 5, 0.15, thresh.perc = 0.05)
   m1f <- make_mat_mappoly(m1, s1f)
   mds.ord <- mds_mappoly(m1f)
   plot(mds.ord)
   so <- make_seq_mappoly(mds.ord)
   plot(m1f, ord = rownames(get_genomic_order(so)))
   plot(so$seq.num ~ I(so$sequence.pos/1e6),
        xlab = "Genome Position",
        ylab = "MDS position")

## End(Not run)
```

`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:

`m`	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.p	a vector containing the dosage in parent P for all n.mrk markers
dosage.q	a vector containing the dosage in parent Q for all n.mrk markers
sequence	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
sequence.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, <gabrielgesteira@usp.br>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
## Loading three chromosomes of sweetpotato dataset (SNPs anchored to Ipomoea trifida genome)
dat <- NULL
for(i in 1:3){
  cat("Loading chromosome", i, "...\n")
  invisible(capture.output(y <- {
    tempfl <- tempfile(pattern = paste0("ch", i), fileext = ".vcf.gz")
   x <- "https://github.com/mmollina/MAPpoly_vignettes/raw/master/data/BT/sweetpotato_chr"
    address <- paste0(x, i, ".vcf.gz")
    download.file(url = address, destfile = tempfl)
   dattemp <- read_vcf(file = tempfl, parent.1 = "PARENT1", parent.2 = "PARENT2", ploidy = 6)
```

```
    dat <- merge_datasets(dat, dattemp)
  }))
  cat("\n")
}
## Filtering dataset by marker
dat <- filter_missing(input.data = dat, type = "marker",
                      filter.thres = 0.05, inter = FALSE)

## Filtering dataset by individual
dat <- filter_missing(input.data = dat, type = "individual",
                      filter.thres = 0.05, inter = TRUE)
print(dat, detailed = TRUE)

## Segregation test
pval.bonf <- 0.05/dat$n.mrk
mrks.chi.filt <- filter_segregation(dat,
                                    chisq.pval.thres =  pval.bonf,
                                    inter = TRUE)
seq.init<-make_seq_mappoly(mrks.chi.filt)
length(seq.init$seq.mrk.names)
plot(seq.init)
print(seq.init, detailed = TRUE)

## End(Not run)
```

`merge_maps` *Merge two maps*

Description

Estimates the linkage phase and recombination fraction between pre-built maps and creates a new map by merging them.

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 `poly.est.two.pts.pairwise` 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:

m	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.p	a vector containing the dosage in parent 1 for all markers in the map
seq.dose.q	a vector containing the dosage in parent 2 for all markers in the map
sequence	a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, sequence = NULL
sequence.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>

Examples

```
## Not run:
#### Tetraploid example #####
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$P)
compare_haplotypes(m = 4, best.phase$P[names.id],
                   full.map$maps[[1]]$seq.ph$P[names.id])
compare_haplotypes(m = 4, best.phase$Q[names.id],
                   full.map$maps[[1]]$seq.ph$Q[names.id])

#### Hexaploid example #####
map1<-get_submap(maps.hexafake[[1]], 1:5)
map2<-get_submap(maps.hexafake[[1]], 6:15)
map3<-get_submap(maps.hexafake[[1]], 16:30)
full.map<-get_submap(maps.hexafake[[1]], 1:30)
s<-make_seq_mappoly(hexafake, 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$P)
compare_haplotypes(m = 6, best.phase$P[names.id],
                   full.map$maps[[1]]$seq.ph$P[names.id])
compare_haplotypes(m = 6, best.phase$Q[names.id],
                   full.map$maps[[1]]$seq.ph$Q[names.id])

## End(Not run)
```

---

plot.mappoly.homoprob *Plots mappoly.homoprob*

Description

Plots mappoly.homoprob

Usage

```
## S3 method for class 'mappoly.homoprob'
plot(x, stack = FALSE, lg = NULL, ind = NULL, use.plotly = 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)
lg	indicates which linkage group should be plotted. If NULL (default), it plots the first linkage group. If "all", it plots all linkage groups
ind	indicates which individuals should be plotted. It can be the position of the individuals in the dataset or it's name. If NULL (default), the function plots the first individual
use.plotly	if TRUE (default), it uses plotly interactive graphics
...	unused arguments

---

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,
  P = "P",
  Q = "Q",
  ...
)
```

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)
P	a string containing the name of parent P
Q	a string containing the name of parent Q
...	unused arguments

plot_genome_vs_map *Physical versus genetic distance*

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

```
plot_genome_vs_map(map.list, phase.config = "best", same.ch.lg = FALSE)
```

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`.

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
## tetraploid example
plot_genome_vs_map(solcap.mds.map)
plot_genome_vs_map(solcap.mds.map, same.ch.lg = TRUE)

## hexaploid example
plot_genome_vs_map(maps.hexafake)
plot_genome_vs_map(maps.hexafake, same.ch.lg = TRUE)

## End(Not run)
```

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 the bars or bar components (default = 'lightgrey') 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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
## 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")

## Comparing mapping approaches
```

```
w<-NULL
for(i in 1:12)
  w<-c(w, c(solcap.dose.map[i],
            solcap.prior.map[i],
            solcap.err.map[i]))

names(w) <- apply(expand.grid(c("dose", "prior", "error"), paste0("LG_", 1:12),
                              stringsAsFactors = FALSE)[,2:1], 1, paste,
                  collapse = "_")

op <- par(cex.axis = .7)
z<-plot_map_list(w, horiz = FALSE, col = rep(gg_color_hue(3), 12))
par(op)
legend("bottomright", legend = c("Dosage based", "Prior", "Error"),
       pch=15, col = rep(gg_color_hue(3)))
head(z)

## End(Not run)
```

`plot_mrk_info` *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

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
plot_mrk_info(tetra.solcap.geno.dist, 2680)
plot_mrk_info(tetra.solcap.geno.dist, "solcap_snp_c2_23828")

## End(Not run)
```

`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

```
poly_cross_simulate(
  m,
  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

`m`	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://www.wur.nl/en/show/Software-PedigreeSim.htm

Value

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

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
h.temp<-sim_homologous(m=6, n.mrk=20, max.d=3, max.ph=3, seed=123)
fake.poly.dat<-poly_cross_simulate(m=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 'mappoly.data'
mrks	marker sequence index (integer vector)

Examples

```
 ## Not run:
 print_mrk(tetra.solcap.geno.dist, 1:5)
 print_mrk(hexafake, 256)

## End(Not run)
```

read_geno *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

```
read_geno(file.in, filter.non.conforming = TRUE, elim.redundant = TRUE)

## S3 method for class 'mappoly.data'
print(x, detailed = FALSE, ...)

## S3 method for class 'mappoly.data'
plot(x, thresh.line = 1e-05, ...)
```

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.
x	an object of class mappoly.data
detailed	if available, print the number of markers per sequence (default = FALSE)
...	currently ignored
thresh.line	position of a threshold line for p values of the segregation test (default = 10e-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 nind and nmrk followed by the number of individuals in the dataset and the total number of markers, respectively. Lines number 4 and 5 contain the strings mrknames and indnames 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 sequence 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:

m	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.p	a vector containing the dosage in parent P for all n.mrk markers
dosage.q	a vector containing the dosage in parent Q for all n.mrk markers
sequence	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
sequence.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

Mollinari M., Olukolu B. A., Pereira G. da S., Khan A., Gemenet D., Yencho G. C., Zeng Z-B. (2020), Unraveling the Hexaploid Sweetpotato Inheritance Using Ultra-Dense Multilocus Mapping, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400620

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:

#### 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)

#### Hexaploid example
fl2 = "https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/hexafake"
tempfl <- tempfile()
download.file(fl2, destfile = tempfl)
hexa.dose <- read_geno(file.in  = tempfl)
print(hexa.dose, detailed = TRUE)
plot(hexa.dose)

## End(Not run)
```

---

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

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

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 sequence 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:

`m`	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.p`	a vector containing the dosage in parent P for all `n.mrk` markers
`dosage.q`	a vector containing the dosage in parent Q for all `n.mrk` markers
`sequence`	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
`sequence.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

Mollinari M., Olukolu B. A., Pereira G. da S., Khan A., Gemenet D., Yencho G. C., Zeng Z-B. (2020), Unraveling the Hexaploid Sweetpotato Inheritance Using Ultra-Dense Multilocus Mapping, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400620

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
#### 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)

## End(Not run)
```

`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

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

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 contains the strings `nind` and `nmrk` followed by the number of individuals in the dataset and the total number of markers, respectively. Lines number 4 and 5 contain the string `mrknames` and `indnames` 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 sequence 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:

m	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.p	a vector containing the dosage in parent P for all n.mrk markers
dosage.q	a vector containing the dosage in parent Q for all n.mrk markers
sequence	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
sequence.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
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. 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

`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

Mollinari M., Olukolu B. A., Pereira G. da S., Khan A., Gemenet D., Yencho G. C., Zeng Z-B. (2020), Unraveling the Hexaploid Sweetpotato Inheritance Using Ultra-Dense Multilocus Mapping, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400620

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
#### Tetraploid Example
ft="https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/SolCAP"
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)
## save dataset for future uses
saveRDS(SolCAP.dose.prob, file = "solcap.rds")
SolCAP.dose.prob <- readRDS("solcap.rds")

#### Hexaploid example
fh="https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/hexafake_geno_dist"
tempfl <- tempfile()
download.file(fh, destfile = tempfl)
hexa.dose.prob <- read_geno_prob(file.in  = tempfl, prob.thres = 0.8)
print(hexa.dose.prob, detailed = TRUE)
plot(hexa.dose.prob)
## save dataset for future uses
saveRDS(hexa.dose.prob, file = "hexa.rds")
hexa.dose.prob <- readRDS("hexa.rds")

## End(Not run)
```

read_vcf *Data Input VCF*

Description

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

Usage

```
read_vcf(
  file.in,
  parent.1,
  parent.2,
  ploidy = NA,
  filter.non.conforming = TRUE,
  thresh.line = 0.05,
  min.gt.depth = 0,
  min.av.depth = 0,
  max.missing = 1,
  elim.redundant = TRUE
)
```

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.

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:

`m`	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.p`	a vector containing the dosage in parent P for all `n.mrk` markers
`dosage.q`	a vector containing the dosage in parent Q for all `n.mrk` markers
`sequence`	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
`sequence.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`
`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

Author(s)

Gabriel Gesteira, <gabrielgesteira@usp.br>

References

Mollinari M., Olukolu B. A., Pereira G. da S., Khan A., Gemenet D., Yencho G. C., Zeng Z-B. (2020), Unraveling the Hexaploid Sweetpotato Inheritance Using Ultra-Dense Multilocus Mapping, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400620

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
fl = "https://github.com/mmollina/MAPpoly_vignettes/raw/master/data/BT/sweetpotato_chr1.vcf.gz"
tempfl <- tempfile(pattern = 'chr1_', fileext = '.vcf.gz')
download.file(fl, destfile = tempfl)
dat.dose.vcf = read_vcf(file = tempfl, parent.1 = "PARENT1", parent.2 = "PARENT2")
plot(dat.dose.vcf)

## Loading full sweetpotato dataset (SNPs anchored to Ipomoea trifida genome)
## Needs ~ 3GB
dat <- NULL
for(i in 1:15){
  cat("Loading chromosome", i, "...\n")
  invisible(capture.output(y <- {
    tempfl <- tempfile(pattern = paste0("ch", i), fileext = ".vcf.gz")
   x <- "https://github.com/mmollina/MAPpoly_vignettes/raw/master/data/BT/sweetpotato_chr"
    address <- paste0(x, i, ".vcf.gz")
    download.file(url = address, destfile = tempfl)
   dattemp <- read_vcf(file = tempfl, parent.1 = "PARENT1", parent.2 = "PARENT2", ploidy = 6)
    dat <- merge_datasets(dat, dattemp)
  }))
  cat("\n")
}
## Filtering dataset by marker
dat <- filter_missing(input.data = dat, type = "marker",
                      filter.thres = 0.05, inter = TRUE)

## Filtering dataset by individual
dat <- filter_missing(input.data = dat, type = "individual",
                      filter.thres = 0.05, inter = TRUE)
print(dat, detailed = TRUE)

## Segregation test
pval.bonf <- 0.05/dat$n.mrk
mrks.chi.filt <- filter_segregation(dat,
                                    chisq.pval.thres =  pval.bonf,
                                    inter = TRUE)
seq.init<-make_seq_mappoly(mrks.chi.filt)
length(seq.init$seq.mrk.names)
plot(seq.init)
print(seq.init, detailed = TRUE)

## End(Not run)
```

`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

```
reest_rf(
```

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

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) or 'ols' (Ordinary Least Squares) 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.

Value

An updated object of class mappoly.map

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Stam P (1993) Construction of integrated genetic-linkage maps by means of a new computer package: Joinmap. _Plant J_ 3:739–744 https://doi.org/10.1111/j.1365-313X.1993.00739.x

---

rev_map *Reverse map*

Description

Provides the reverse of a given map.

Usage

```
rev_map(input.map)
```

Arguments

input.map	an object of class mappoly.map

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

Examples

```
## Not run:
    plot_genome_vs_map(solcap.mds.map[[1]])
    plot_genome_vs_map(rev_map(solcap.mds.map[[1]]))

## End(Not run)
```

---

rf_list_to_matrix *Recombination fraction list to matrix*

Description

Transforms the recombination fraction list contained in an object of class poly.est.two.pts.pairwise 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,
  ...
)
```

Arguments

input.twopt	an object of class poly.est.two.pts.pairwise
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)

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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
  all.mrk<-make_seq_mappoly(hexafake, 'all')
  red.mrk<-elim_redundant(all.mrk)
  unique.mrks<-make_seq_mappoly(red.mrk)
  all.pairs<-est_pairwise_rf(input.seq = unique.mrks,
                             ncpus = 7,
                             verbose=TRUE)
```

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

    ## Filtered matrix
    mat.filt<-rf_list_to_matrix(input.twopt=all.pairs,
                                thresh.LOD.ph = 5,
                                thresh.LOD.rf = 5,
                                thresh.rf = 0.5,
                                verbose = TRUE)
    plot(mat.filt)
    plot(mat.filt, type = "lod")

## End(Not run)
```

`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

```
rf_snp_filter(
  input.twopt,
  thresh.LOD.ph = 5,
  thresh.LOD.rf = 5,
  thresh.rf = 0.15,
  thresh.perc = 0.05,
  remove.fp = NULL,
  ncpus = 1L
)
```

Arguments

`input.twopt`	an object of class `poly.est.two.pts.pairwise`
`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) #'
`thresh.perc`	threshold for the percentage of the pairwise marker combinations that should be considered in order to keep the marker. For example, `thresh.perc = 0.05` means that at least 5% of the pairwise combinations should be present in order to keep the marker (default = 0.05)
`remove.fp`	numeric value from 0.0 to 0.5 (default = NULL). When defined, this parameter identifies and removes markers that presented more than 90% of its pairwise recombination fractions below `remove.fp` value throughout the linkage group
`ncpus`	number of parallel processes (i.e. cores) to spawn (default = 1)

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, <gabrielgesteira@usp.br>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
   all.mrk<-make_seq_mappoly(hexafake, 'all')
   red.mrk<-elim_redundant(all.mrk)
   unique.mrks<-make_seq_mappoly(red.mrk)
   all.pairs<-est_pairwise_rf(input.seq = unique.mrks,
                              ncpus = 7,
                              verbose=TRUE)

   ## Full recombination fraction matrix
   mat.full<-rf_list_to_matrix(input.twopt=all.pairs)
   plot(mat.full)
   lgs <- group_mappoly(input.mat = mat.full,
                        expected.groups = 3,
                        inter = TRUE,
                        comp.mat = TRUE, #this data has physical information
                        verbose=TRUE)
   lgs
   plot(lgs)
   lg1 <- make_seq_mappoly(lgs, 1)
   lg2 <- make_seq_mappoly(lgs, 2)
   lg3 <- make_seq_mappoly(lgs, 3)

   ##Plot matrices
   p1<-make_pairs_mappoly(input.seq = lg1, input.twopt = all.pairs)
   p2<-make_pairs_mappoly(input.seq = lg2, input.twopt = all.pairs)
   p3<-make_pairs_mappoly(input.seq = lg3, input.twopt = all.pairs)

   m1<-rf_list_to_matrix(input.twopt = p1)
   m2<-rf_list_to_matrix(input.twopt = p2)
   m3<-rf_list_to_matrix(input.twopt = p3)

   op<-par(mfrow = c(1,3), pty = "s")
   plot(m1, main.text = "LG1")
   plot(m2, main.text = "LG2")
```

```
plot(m3, main.text = "LG3")
par(op)

## Removing disruptive SNPs
lg1.filt<-rf_snp_filter(p1, 5, 5, 0.15, thresh.perc = 0.05)
lg2.filt<-rf_snp_filter(p2, 5, 5, 0.15, thresh.perc = 0.05)
lg3.filt<-rf_snp_filter(p3, 5, 5, 0.15, thresh.perc = 0.05)

p1.filt<-make_pairs_mappoly(input.seq = lg1.filt, input.twopt = all.pairs)
p2.filt<-make_pairs_mappoly(input.seq = lg2.filt, input.twopt = all.pairs)
p3.filt<-make_pairs_mappoly(input.seq = lg3.filt, input.twopt = all.pairs)

m1.filt<-rf_list_to_matrix(input.twopt = p1.filt)
m2.filt<-rf_list_to_matrix(input.twopt = p2.filt)
m3.filt<-rf_list_to_matrix(input.twopt = p3.filt)

op<-par(mfrow = c(2,3), pty = "s")
plot(m1, main.text = "LG1")
plot(m2, main.text = "LG2")
plot(m3, main.text = "LG3")
plot(m1.filt, main.text = "LG1.filt")
plot(m2.filt, main.text = "LG2.filt")
plot(m3.filt, main.text = "LG3.filt")
par(op)

## End(Not run)
```

`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

```
segreg_poly(m, dP, dQ)
```

Arguments

`m`	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

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Serang O, Mollinari M, Garcia AAF (2012) Efficient Exact Maximum a Posteriori Computation for Bayesian SNP Genotyping in Polyploids. _PLoS ONE_ 7(2): e30906. https://doi.org/10.1371/journal.pone.0030906

Examples

```
# autohexaploid with two and three doses in parents P and Q,
# respectively
seg<-segreg_poly(m=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

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

Arguments

m	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 = m + 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
`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)
`hom.allele.q`	Analogously to `hom.allele.p`
`q`	Analogously to `p`

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

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

`solcap.dose.map` *Resulting maps from* `tetra.solcap`

Description

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

Usage

```
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.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

```
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

```
split_and_rephase(
  input.map,
  twopt,
  gap.threshold = 5,
  remove.single = TRUE,
  phase.config = "best",
  tol.final = 0.001
)
```

Arguments

`input.map`	an object of class `mappoly.map`
`twopt`	an object of class `poly.est.two.pts.pairwise` 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
`remove.single`	Should isolated markers be removed?
`phase.config`	which phase configuration should be used. "best" (default) will choose the phase configuration associated with the maximum likelihood
`tol.final`	the desired accuracy for the final map (default = 10e-04)

Value

An object of class `mappoly.map`

Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

References

Mollinari, M., and Garcia, A. A. F. (2019) Linkage analysis and haplotype phasing in experimental autopolyploid populations with high ploidy level using hidden Markov models, _G3: Genes, Genomes, Genetics_. https://doi.org/10.1534/g3.119.400378

Examples

```
## Not run:
map <- solcap.err.map[[1]]
tpt <- est_pairwise_rf(make_seq_mappoly(map))
new.map <- split_and_rephase(map, tpt, 5)
plot_map_list(list(map, new.map))
map
new.map
plot_map_list(list(old.map = map, new.map = new.map))

## End(Not run)
```

---

summary_maps *Summary map*

Description

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

Usage

```
summary_maps(map.list)
```

Arguments

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

Value

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

Author(s)

Gabriel Gesteira, <gabrielgesteira@usp.br>

Examples

```
## Not run:
(tetra.sum <- summary_maps(solcap.err.map))
formattable::formattable(tetra.sum)
(hexa.sum <- summary_maps(maps.hexafake))
formattable::formattable(hexa.sum)

## End(Not run)
```

`tetra.solcap` *Autotetraploid potato dataset.*

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:

m 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.p a vector containing the dosage in parent P for all `n.mrk` markers

dosage.q a vector containing the dosage in parent Q for all `n.mrk` markers

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

sequence.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:

m 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.p a vector containing the dosage in parent P for all `n.mrk` markers

dosage.q a vector containing the dosage in parent Q for all `n.mrk` markers

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

sequence.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

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

update_map *Update map*

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

```
update_map(input.map)
```

Arguments

input.map	an map object of class mappoly.map

Value

an updated object of class mappoly.map, containing the original map plus redundant markers

Author(s)

Gabriel Gesteira, <gabrielgesteira@usp.br>

Examples

```
 ## Not run:
 orig.map   <- solcap.err.map
 up.map <- lapply(solcap.err.map, update_map)
 formattable::formattable(summary_maps(orig.map))
 formattable::formattable(summary_maps(up.map))

## End(Not run)
```

update_missing *Update missing information*

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

```
## Not run:
data.updated = update_missing(tetra.solcap.geno.dist, prob.thres = 0.5)
print(tetra.solcap.geno.dist)
print(data.updated)

## End(Not run)
```

Index