

# Package ‘mappoly’

November 23, 2020

**Type** Package

**Title** Genetic Linkage Maps in Autopolyploids

**Version** 0.2.1

**Maintainer** Marcelo Mollinari <mmollin@ncsu.edu>

**Description** Construction of genetic maps in autopolyploid full-sib populations.

Uses pairwise recombination fraction estimation as the first source of information to sequentially position allelic variants in specific homologues. For situations where pairwise analysis has limited power, the algorithm relies on the multilocus likelihood obtained through a hidden Markov model (HMM). For more detail, please see Mollinari and Garcia (2019) <doi:10.1534/g3.119.400378> and Mollinari et al. (2020) <doi:10.1534/g3.119.400620>.

**License** GPL-3

**LazyData** TRUE

**Depends** R (>= 3.5.0)

**Imports** Rcpp (>= 0.12.6), RCurl, plotly, fields, ggpubr, ggsci, rstudioapi, scatterplot3d, dplyr, crayon, cli, magrittr, reshape2, ggplot2, smacof, princurve, dendextend, vcfR, zoo

**LinkingTo** Rcpp

**RoxygenNote** 7.1.1

**SystemRequirements** C++11

**Encoding** UTF-8

**Suggests** testthat, RColorBrewer, updog, covr, polymapR, fitPoly

**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 [aut] (<<https://orcid.org/0000-0002-4106-7346>>), Guilherme 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>>),  
 Katharine Preedy [ctb, cph] (MDS ordering algorithm),  
 Robert Gentleman [cph] (C code for MLE optimization in  
 src/pairwise\_estimation.cpp),  
 Ross Ihaka [cph] (C code for MLE optimization in  
 src/pairwise\_estimation.cpp),  
 R Foundation [cph] (C code for MLE optimization in  
 src/pairwise\_estimation.cpp),  
 R-core [cph] (C code for MLE optimization in  
 src/pairwise\_estimation.cpp)

**Repository** CRAN

**Date/Publication** 2020-11-23 10:50:11 UTC

## R topics documented:

add_marker . . . . .	3
cache_counts_twopt . . . . .	6
calc_genoprob . . . . .	8
calc_genoprob_dist . . . . .	9
calc_genoprob_error . . . . .	10
calc_homoprob . . . . .	11
calc_prefpair_profiles . . . . .	12
check_data_sanity . . . . .	13
drop_marker . . . . .	14
elim_redundant . . . . .	14
est_full_hmm_with_global_error . . . . .	15
est_full_hmm_with_prior_prob . . . . .	17
est_pairwise_rf . . . . .	19
est_rf_hmm . . . . .	21
est_rf_hmm_sequential . . . . .	24
export_data_to_polymapR . . . . .	28
export_map_list . . . . .	29
extract_map . . . . .	29
filter_missing . . . . .	30
filter_segregation . . . . .	31
get_genomic_order . . . . .	32
get_submap . . . . .	32
get_tab_mrks . . . . .	34
group_mappoly . . . . .	35
hexafake . . . . .	36
hexafake.geno.dist . . . . .	37
import_data_from_polymapR . . . . .	38
import_from_updog . . . . .	39
import_phased_maplist_from_polymapR . . . . .	41
loglike_hmm . . . . .	42
make_mat_mappoly . . . . .	43
make_pairs_mappoly . . . . .	44

make_seq_mappoly . . . . .	45
maps.hexafake . . . . .	47
mds_mappoly . . . . .	47
merge_datasets . . . . .	49
merge_maps . . . . .	51
plot.mappoly.homoprob . . . . .	53
plot.mappoly.prefpair.profiles . . . . .	54
plot_genome_vs_map . . . . .	55
plot_map_list . . . . .	56
plot_mrk_info . . . . .	57
poly_cross_simulate . . . . .	58
print_mrk . . . . .	59
read_fitpoly . . . . .	60
read_geno . . . . .	62
read_geno_csv . . . . .	65
read_geno_prob . . . . .	67
read_vcf . . . . .	69
reest_rf . . . . .	72
rev_map . . . . .	73
rf_list_to_matrix . . . . .	73
rf_snp_filter . . . . .	75
segreg_poly . . . . .	77
sim_homologous . . . . .	78
solcap.dose.map . . . . .	79
solcap.err.map . . . . .	80
solcap.mds.map . . . . .	80
solcap.prior.map . . . . .	81
split_and_rephase . . . . .	81
summary_maps . . . . .	82
tetra.solcap . . . . .	83
tetra.solcap.geno.dist . . . . .	84
update_map . . . . .	85
update_missing . . . . .	85

**Index****87**


---

add_marker	<i>Add a single marker to a map</i>
------------	-------------------------------------

---

**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,
  verbose = TRUE
)

```

**Arguments**

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

**Details**

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

**Value**

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

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

<code>m</code>	the ploidy level
<code>n.mrk</code>	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**

```
sub.map<-get_submap(maps.hexafake[[1]], 1:20, reestimate.rf = FALSE)
plot(sub.map, mrk.names = TRUE)
s<-make_seq_mappoly(hexafake, sub.map$info$mrk.names)
tpt <- est_pairwise_rf(s)
rf.matrix <- rf_list_to_matrix(input.twopt = tpt,
                             thresh.LOD.ph = 3,
                             thresh.LOD.rf = 3,
                             shared.alleles = TRUE)
##### Removing marker "M_1" (first) #####
mrk.to.remove <- "M_1"
input.map <- drop_marker(sub.map, mrk.to.remove)
plot(input.map, mrk.names = TRUE)
## Computing conditional probabilities using the resulting map
```

```

genoprob <- calc_genoprob(input.map)
res.add.M_1<-add_marker(input.map = input.map,
                        mrk = "M_1",
                        pos = 0,
                        rf.matrix = rf.matrix,
                        genoprob = genoprob,
                        tol = 10e-4)
plot(res.add.M_1, mrk.names = TRUE)
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_10" (middle or last) #####
mrk.to.remove <- "M_10"
input.map <- drop_marker(sub.map, mrk.to.remove)
plot(input.map, mrk.names = TRUE)
# Computing conditional probabilities using the resulting map
genoprob <- calc_genoprob(input.map)
res.add.M_10<-add_marker(input.map = input.map,
                        mrk = "M_10",
                        pos = "M_9",
                        rf.matrix = rf.matrix,
                        genoprob = genoprob,
                        tol = 10e-4)
plot(res.add.M_10, mrk.names = TRUE)
best.phase <- res.add.M_10$maps[[1]]$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])

```

---

cache_counts_twopt	<i>Frequency of genotypes for two-point recombination fraction estimation</i>
--------------------	---

---

## 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

<code>input.seq</code>	an object of class <code>mappoly.sequence</code>
<code>cached</code>	If TRUE, access the counts for all linkage phase configurations in a internal file (default = FALSE)
<code>cache.prev</code>	an object of class <code>cache.info</code> containing pre-computed genotype frequencies, obtained with <code>cache_counts_twopt</code> (optional, default = NULL)
<code>ncpus</code>	Number of parallel processes to spawn (default = 1)
<code>verbose</code>	If TRUE (default), print the linkage phase configurations. If <code>cached = TRUE</code> , nothing is printed, since all linkage phase configurations will be cached.
<code>joint.prob</code>	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**

```

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

```

---

calc\_genoprob

*Compute conditional probabilities of the genotypes*


---

**Description**

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

**Usage**

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

**Arguments**

input.map	An object of class <code>mappoly.map</code>
step	Maximum distance (in cM) between positions at which the genotype probabilities are calculated, though for <code>step = 0</code> , 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 its 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**

```
## 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]))
```

---

calc_genoprob_dist	<i>Compute conditional probabilities of the genotypes using probability distribution of dosages</i>
--------------------	---

---

**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 <code>mappoly.map</code>
dat.prob	an object of class <code>mappoly.data</code> 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 its 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

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

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

---

calc\_genoprob\_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 <code>mappoly.map</code>
step	Maximum distance (in cM) between positions at which the genotype probabilities are calculated, though for <code>step = 0</code> , 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**

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

---

calc_homoprob	<i>Homolog probabilities</i>
---------------	------------------------------

---

**Description**

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

**Usage**

```
calc_homoprob(input.genoprobs, verbose = TRUE)
```

**Arguments**

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

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**References**

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**

```
## tetraploid example
w1 <- calc_genoprob(solcap.dose.map[[1]])
h.prob <- calc_homoprob(w1)
print(h.prob)
plot(h.prob, ind = 5, use.plotly = FALSE)
## using error modeling (removing noise)
w2 <- calc_genoprob_error(solcap.err.map[[1]])
h.prob2 <- calc_homoprob(w2)
print(h.prob2)
plot(h.prob2, ind = 5, use.plotly = FALSE)
```

---

calc\_prefpair\_profiles

*Preferential pairing profiles*

---

**Description**

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

**Usage**

```
calc_prefpair_profiles(input.genoprobs, verbose = TRUE)
```

**Arguments**

input.genoprobs	an object of class <code>mappoly.genoprob</code>
verbose	if TRUE (default), the current progress is shown; if FALSE, no output is produced

**Author(s)**

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

## References

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

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

---

check_data_sanity	<i>Data sanity check</i>
-------------------	--------------------------

---

## 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

```
check_data_sanity(tetra.solcap)
```

---

drop_marker	<i>Remove markers from a map</i>
-------------	----------------------------------

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

an object of class `mappoly.map`

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**Examples**

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

---

elim_redundant	<i>Eliminate redundant markers</i>
----------------	------------------------------------

---

**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**

<code>input.map</code>	an object of class <code>mappoly.map</code>
<code>error</code>	the assumed global error rate (default = <code>NULL</code> )
<code>tol</code>	the desired accuracy (default = <code>10e-04</code> )
<code>restricted</code>	if <code>TRUE</code> (default), restricts the prior to the possible classes under Mendelian, non double-reduced segregation given dosage of the parents
<code>th.prob</code>	the threshold for using global error or genotype probability distribution if present in the dataset (default = <code>0.95</code> )
<code>verbose</code>	if <code>TRUE</code> , current progress is shown; if <code>FALSE</code> (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:

<code>m</code>	the ploidy level
<code>n.mrk</code>	number of markers
<code>seq.num</code>	a vector containing the (ordered) indices of markers in the map, according to the input file
<code>mrk.names</code>	the names of markers in the map
<code>seq.dose.p</code>	a vector containing the dosage in parent 1 for all markers in the map
<code>seq.dose.q</code>	a vector containing the dosage in parent 2 for all markers in the map
<code>sequence</code>	a vector indicating the sequence (usually chromosome) each marker belongs as informed in the input file. If not available, <code>sequence = NULL</code>
<code>sequence.pos</code>	physical position (usually in megabase) of the markers into the sequence
<code>seq.ref</code>	reference base used for each marker (i.e. A, T, C, G). If not available, <code>seq.ref = NULL</code>
<code>seq.alt</code>	alternative base used for each marker (i.e. A, T, C, G). If not available, <code>seq.ref = NULL</code>
<code>chisq.pval</code>	a vector containing p-values of the chi-squared test of Mendelian segregation for all markers in the map
<code>data.name</code>	name of the dataset of class <code>mappoly.data</code>



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**

```
submap <- get_submap(solcap.dose.map[[1]], mrk.pos = 1:20, verbose = FALSE)
err.submap <- est_full_hmm_with_global_error(submap,
                                           error=0.01,
                                           tol=10e-4,
                                           verbose = TRUE)

err.submap
plot_map_list(list(dose = submap, err = err.submap),
              title = "estimation procedure")
```

---

est\_full\_hmm\_with\_prior\_prob

*Re-estimate genetic map using dosage prior probability distribution*

---

**Description**

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

**Usage**

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

**Arguments**

<code>input.map</code>	an object of class <code>mappoly.map</code>
<code>dat.prob</code>	an object of class <code>mappoly.data</code> containing the probability distribution of the genotypes
<code>phase.config</code>	which phase configuration should be used. "best" (default) will choose the maximum likelihood configuration
<code>tol</code>	the desired accuracy (default = 10e-04)
<code>verbose</code>	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:

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

```
submap <- get_submap(solcap.dose.map[[1]], mrk.pos = 1:20, verbose = FALSE)
prob.submap <- est_full_hmm_with_prior_prob(submap,
                                           dat.prob = tetra.solcap.geno.dist,
                                           tol=10e-4,
                                           verbose = TRUE)

prob.submap
plot_map_list(list(dose = submap, prob = prob.submap),
              title = "estimation procedure")
```

---

est\_pairwise\_rf      *Pairwise two-point analysis*

---

### Description

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

### Usage

```
est_pairwise_rf(
  input.seq,
  count.cache = NULL,
  ncpus = 1L,
  mrk.pairs = NULL,
```

```

n.batches = 1L,
est.type = c("disc", "prob"),
verbose = TRUE,
memory.warning = TRUE,
parallelization.type = c("PSOCK", "FORK"),
tol = .Machine$double.eps^0.25
)

```

### Arguments

<code>input.seq</code>	an object of class <code>mappoly.sequence</code>
<code>count.cache</code>	an object of class <code>cache.info</code> containing pre-computed genotype frequencies, obtained with <code>cache_counts_twopt</code> . If NULL (default), genotype frequencies are internally loaded.
<code>ncpus</code>	Number of parallel processes (cores) to spawn (default = 1)
<code>mrk.pairs</code>	a matrix of dimensions $2*N$ , containing $N$ pairs of markers to be analyzed. If NULL (default), all pairs are considered
<code>n.batches</code>	The number of batches of marker pairs that should be analyzed in parallel. Using <code>n.batches &gt; 1</code> , will usually result in more processing time. However, it will require less memory. See examples.
<code>est.type</code>	Indicates whether to use the discrete ("disc") or the probabilistic ("prob") dosage scoring when estimating the two-point recombination fractions.
<code>verbose</code>	If TRUE (default), current progress is shown; if FALSE, no output is produced
<code>memory.warning</code>	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.
<code>parallelization.type</code>	one of the supported cluster types. This should be either PSOCK (default) or FORK.
<code>tol</code>	the desired accuracy. See <code>optimize()</code> for details

### Value

An object of class `poly.est.two.pts.pairwise` which is a list containing the following components:

<code>data.name</code>	name of the object of class <code>mappoly.data</code> with the raw data
<code>n.mrk</code>	number of markers in the sequence
<code>seq.num</code>	a vector containing the (ordered) indices of markers in the sequence, according to the input file
<code>pairwise</code>	a list of size <code>choose(length(input.seq\$seq.num), 2)</code> , each of them containing a matrix where the name of the rows have the form <code>x-y</code> , where <code>x</code> and <code>y</code> 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.thresh 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

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

all.pairs
plot(all.pairs, 20, 21)
mat <- rf_list_to_matrix(all.pairs)
plot(mat)
```

---

est\_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",
  P = "Parent 1",
  Q = "Parent 2",
  ...
)

```

### Arguments

<code>input.seq</code>	an object of class <code>mappoly.sequence</code>
<code>input.ph</code>	an object of class <code>two.pts.linkage.phases</code> . If not available (default = NULL), it will be computed
<code>thres</code>	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.
<code>twopt</code>	an object of class <code>poly.est.two.pts.pairwise</code> containing two-point information
<code>verbose</code>	if TRUE, current progress is shown; if FALSE (default), no output is produced
<code>tol</code>	the desired accuracy (default = 1e-04)
<code>est.given.0.rf</code>	logical. If TRUE returns a map forcing all recombination fractions equals to 0 (1e-5, for internal use only. Default = FALSE)
<code>reestimate.single.ph.configuration</code>	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
<code>high.prec</code>	logical. If TRUE (default) uses high precision long double numbers in the HMM procedure
<code>x</code>	an object of the class <code>mappoly.map</code>
<code>detailed</code>	logical. if TRUE, prints the linkage phase configuration and the marker position for all maps. If FALSE (default), prints a map summary
<code>...</code>	currently ignored
<code>left.lim</code>	the left limit of the plot (in cM, default = 0).
<code>right.lim</code>	the right limit of the plot (in cM, default = Inf, i.e., will print the entire map)

phase	logical. If TRUE (default) plots the phase configuration for both parents
mrk.names	if TRUE, marker names are displayed (default = FALSE)
cex	The magnification to be used for marker names
config	should be 'best' or the position of the configuration to be plotted. If 'best', plot the configuration with the highest likelihood
P	a string containing the name of parent P
Q	a string containing the name of parent Q

### 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:

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

```

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

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

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

```

---

est\_rf\_hmm\_sequential *Multipoint analysis using Hidden Markov Models: Sequential phase elimination*

---

**Description**

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



**Usage**

```

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

```

**Arguments**

<code>input.seq</code>	an object of class <code>mappoly.sequence</code>
<code>twopt</code>	an object of class <code>poly.est.two.pts.pairwise</code> containing the two-point information
<code>start.set</code>	number of markers to start the phasing procedure (default = 4)
<code>thres.twopt</code>	the LOD threshold used to determine if the linkage phases compared via two-point analysis should be considered for the search space reduction (A.K.A. $\eta$ in <i>Mollinari and Garcia (2019)</i> , default = 5)
<code>thres.hmm</code>	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)
<code>extend.tail</code>	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 <code>info.tail = TRUE</code> , it uses at least <code>extend.tail</code> as the tail length
<code>phase.number.limit</code>	the maximum number of linkage phases of the sub-maps defined by arguments <code>info.tail</code> and <code>extend.tail</code> . Default is 20. If the size exceeds this limit, the marker will not be inserted. If <code>Inf</code> , then it will insert all markers.
<code>sub.map.size.diff.limit</code>	the maximum accepted length difference between the current and the previous sub-map defined by arguments <code>info.tail</code> and <code>extend.tail</code> . If the size exceeds this limit, the marker will not be inserted. If NULL (default), then it will insert all markers.
<code>info.tail</code>	if TRUE (default), it uses the complete informative tail of the chain (i.e. number of markers where all homologous ( <i>ploidy</i> $\times$ 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:

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

```

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

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

## 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)

```

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

```
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)
```

---

export_map_list	<i>Export a genetic map to a CSV file</i>
-----------------	---

---

**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**

<code>map.list</code>	A list of objects or a single object of class <code>mappoly.map</code>
<code>file</code>	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**

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

---

extract_map	<i>Extract the maker position from an object of class 'mappoly.map'</i>
-------------	---

---

**Description**

Extract the maker position from an object of class `'mappoly.map'`

**Usage**

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

**Arguments**

`input.map` An object of class `mappoly.map`

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

**Examples**

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

---

<code>filter_missing</code>	<i>Filter missing genotypes</i>
-----------------------------	---------------------------------

---

**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**

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

---

filter\_segregation      *Filter markers based on chi-square test*

---

**Description**

This function 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.chi.test.seq which contains a list with the following components:

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

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**Examples**

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

---

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, verbose = TRUE)
```

**Arguments**

input.seq      a sequence object of class `mappoly.sequence`  
verbose      if TRUE (default), the current progress is shown; if FALSE, no output is produced

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**Examples**

```
s1<-make_seq_mappoly(tetra.solcap, "all")
o1<-get_genomic_order(s1)
head(o1)
```

---

get\_submap      *Extract sub-map from map*

---

**Description**

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

**Usage**

```
get_submap(
  input.map,
  mrk.pos,
  phase.config = "best",
  reestimate.rf = TRUE,
  reestimate.phase = FALSE,
  thres.twopt = 5,
  thres.hmm = 3,
  extend.tail = 50,
```



```

    tol = 0.1,
    tol.final = 0.001,
    use.high.precision = FALSE,
    verbose = TRUE
)

```

### Arguments

<code>input.map</code>	An object of class <code>mappoly.map</code>
<code>mrk.pos</code>	positions of the markers that should be considered in the new map. This can be in any order
<code>phase.config</code>	which phase configuration should be used. "best" (default) will choose the configuration associated with the maximum likelihood
<code>reestimate.rf</code>	logical. If TRUE (default) the recombination fractions between markers are re-estimated
<code>reestimate.phase</code>	logical. If TRUE, the linkage phase configurations are re-estimated (default = FALSE)
<code>thres.twopt</code>	the LOD threshold used to determine if the linkage phases compared via two-point analysis should be considered (default = 5)
<code>thres.hmm</code>	the threshold used to determine if the linkage phases compared via hmm analysis should be considered (default = 3)
<code>extend.tail</code>	the length of the tail of the chain that should be used to calculate the likelihood of the linkage phases. If <code>info.tail = TRUE</code> , the function uses at least <code>extend.tail</code> as the length of the tail (default = 50)
<code>tol</code>	the desired accuracy during the sequential phase (default = 0.1)
<code>tol.final</code>	the desired accuracy for the final map (default = 10e-04)
<code>use.high.precision</code>	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)
<code>verbose</code>	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

```
## selecting the six first markers in linkage group 1
## re-estimating the recombination fractions and linkage phases
submap1.lg1<-get_submap(input.map = maps.hexafake[[1]],
                       mrk.pos = 1:6, verbose = TRUE,
                       reestimate.phase = TRUE,
                       tol.final = 10e-3)
## no recombination fraction re-estimation: first 20 markers
submap2.lg1<-get_submap(input.map = maps.hexafake[[1]],
                       mrk.pos = 1:20, reestimate.rf = FALSE,
                       verbose = TRUE,
                       tol.final = 10e-3)
plot(maps.hexafake[[1]])
plot(submap1.lg1, mrk.names = TRUE, cex = .8)
plot(submap2.lg1, mrk.names = TRUE, cex = .8)
```

---

get\_tab\_mrks

*Get table of dosage combinations*

---

## Description

Internal function

## Usage

```
get_tab_mrks(x)
```

## Arguments

x                    an object of class `mappoly.map`

## Author(s)

Gabriel Gesteira, <gabrielgesteira@usp.br>

---

group_mappoly	<i>Assign markers to linkage groups</i>
---------------	---

---

**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 <code>mappoly.rf.matrix</code>
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 <code>group_mappoly</code>
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**

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

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

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

lgs
plot(lgs)
```

---

hexafake

*Simulated autohexaploid dataset.*

---

**Description**

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

**Usage**

hexafake

**Format**

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

**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",
  input.type = c("discrete", "probabilistic"),
  prob.thres = 0.95,
  pardose = NULL,
  offspring = NULL,
  filter.non.conforming = TRUE,
  verbose = TRUE
)
```

## Arguments

<code>input.data</code>	a polymapR dataset
<code>ploidy</code>	the ploidy level
<code>parent1</code>	a character string containing the name (or pattern of genotype IDs) of parent 1

parent2	a character string containing the name (or pattern of genotype IDs) of parent 2
input.type	Indicates whether the input is discrete ("disc") or probabilistic ("prob")
prob.thres	threshold probability to assign a dosage to offspring. If the probability is smaller than thresh.parent.geno, the data point is converted to 'NA'.
pardose	matrix of dimensions (n.mrk x 3) containing the name of the markers in the first column, and the dosage of parents 1 and 2 in columns 2 and 3. (see polymapR vignette)
offspring	a character string containing the name (or pattern of genotype IDs) of the offspring individuals. If NULL (default) it considers all individuals as offsprings, except parent1 and parent2.
filter.non.conforming	if TRUE exclude samples with non expected genotypes under no double reduction. Since markers were already filtered in polymapR, the default is FALSE.
verbose	if TRUE (default), the current progress is shown; if FALSE, no output is produced

### 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

```
require(polymapR)
data("screened_data3")
mappoly.data <- import_data_from_polymapR(screened_data3, 4)
plot(mappoly.data)
```

---

import_from_updog	<i>Import from updog</i>
-------------------	--------------------------

---

### 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,
  verbose = TRUE
)
```

**Arguments**

<code>object</code>	the name of the object of class <code>multidog</code>
<code>prob.thres</code>	probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than <code>'prob.thres'</code> are considered as missing data for the dosage calling purposes
<code>filter.non.conforming</code>	if TRUE (default) exclude samples with non expected genotypes under random chromosome pairing and no double reduction
<code>verbose</code>	if TRUE (default), the current progress is shown; if FALSE, no output is produced

**Value**

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

<code>m</code>	ploidy level
<code>n.ind</code>	number individuals
<code>n.mrk</code>	total number of markers
<code>ind.names</code>	the names of the individuals
<code>mrk.names</code>	the names of the markers
<code>dosage.p</code>	a vector containing the dosage in parent P for all <code>n.mrk</code> markers
<code>dosage.q</code>	a vector containing the dosage in parent Q for all <code>n.mrk</code> markers
<code>sequence</code>	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
<code>sequence.pos</code>	physical position of the markers into the sequence
<code>prob.thres</code>	probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than <code>'prob.thres'</code> were considered as missing data in the <code>'geno.dose'</code> matrix
<code>geno.dose</code>	a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by <code>ploidy_level + 1</code>
<code>geno</code>	a <code>data.frame</code> 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 <a href="#">segreg_poly</a>
<code>n.phen</code>	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

```
library("updog")
data("uitdewilligen")
mout = multidog(refmat = t(uitdewilligen$refmat),
                sizemat = t(uitdewilligen$sizemat),
                ploidy = uitdewilligen$ploidy,
                model = "f1",
                p1_id = colnames(t(uitdewilligen$sizemat))[1],
                p2_id = colnames(t(uitdewilligen$sizemat))[2],
                nc = 2)
mydata = import_from_updog(mout)
mydata
plot(mydata)
```

---

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 <code>create_phased_maplist</code> from package <code>polymapR</code>
mappoly.data	a dataset used to obtain <code>maplist</code> , converted into class <code>mappoly.data</code>
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**

```
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 = TRUE, left.lim = 0, right.lim = 20, cex = .7)
```

---

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 <code>mappoly.map</code>
input.data	An object of class <code>mappoly.data</code> , which was used to generate <code>input.map</code>
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**

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

---

make_mat_mappoly	<i>Subset recombination fraction matrices</i>
------------------	---

---

**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 <code>mappoly.rf.matrix</code>
input.seq	an object of class <code>mappoly.sequence</code> , with a sequence of markers contained in <code>input.mat</code>

**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**

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

---

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**

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

**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**

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

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

---

make_seq_mappoly	<i>Create a sequence of markers</i>
------------------	-------------------------------------

---

**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**

<code>input.obj</code>	an object of one of the following classes: <code>mappoly.data</code> , <code>mappoly.map</code> , <code>mappoly.group</code> , <code>mappoly.unique.seq</code> , <code>mappoly.pcmmap</code> or <code>mappoly.pcmmap3d</code>
<code>arg</code>	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 <code>input.object</code> has class <code>mappoly.group</code> ; or v) NULL if <code>input.object</code> has class <code>mappoly.pcmmap</code> , <code>mappoly.pcmmap3d</code> or <code>mappoly.unique.seq</code>
<code>data.name</code>	name of the object of class <code>mappoly.data</code>
<code>genomic.info</code>	optional argument applied for <code>mappoly.group</code> 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 <code>genomic.info</code> = NULL (default), the function returns a sequence containing all markers defined by the grouping function. When <code>genomic.info</code> = 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 <code>genomic.info</code> = 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.
<code>x</code>	an object of the class <code>mappoly.sequence</code>
<code>...</code>	currently ignored

**Value**

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

<code>seq.num</code>	a vector containing the (ordered) indices of markers in the sequence, according to the input file
<code>seq.phases</code>	a list with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases
<code>seq.rf</code>	a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies
<code>loglike</code>	log-likelihood of the corresponding linkage map
<code>data.name</code>	name of the object of class <code>mappoly.data</code> with the raw data
<code>twopt</code>	name of the object of class <code>mappoly.twopt</code> 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

```
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)
```

---

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](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 `MDSmap`, available under GNU GENERAL PUBLIC LICENSE, Version 3, at <https://CRAN.R-project.org/package=MDSmap>

**Usage**

```

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

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

## S3 method for class 'mappoly.pcmmap3d'
print(x, ...)

```

**Arguments**

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

**Value**

A list containing:

<code>M</code>	the input distance map
<code>sm</code>	the unconstrained MDS results
<code>pc</code>	the principal curve results
<code>distmap</code>	a matrix of pairwise distances between loci where the columns are in the estimated order
<code>locimap</code>	a data frame of the loci containing the name and position of each locus in order of increasing distance
<code>length</code>	integer giving the total length of the segment
<code>removed</code>	a vector of the names of loci removed from the analysis
<code>scale</code>	the scaling factor from the MDS



locikey	a data frame showing the number associated with each locus name for interpreting the MDS configuration plot
confplotno	a data frame showing locus name associated with each number on the MDS configuration plots

### Author(s)

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

### References

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

```
s1 <- make_seq_mappoly(hexafake, 1:20)
t1 <- est_pairwise_rf(s1, ncpus = 1)
m1 <- rf_list_to_matrix(t1)
plot(m1, ord = rownames(get_genomic_order(s1)))
mds.ord <- mds_mappoly(m1)
plot(mds.ord)
so <- make_seq_mappoly(mds.ord)
plot(m1, ord = rownames(get_genomic_order(so)))
plot(so$seq.num ~ I(so$sequence.pos/1e6),
      xlab = "Genome Position",
      ylab = "MDS position")
```

---

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

**Value**

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

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

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

---

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 <code>mappoly.map</code> to be merged.
twopt	an object of class <code>poly.est.two.pts.pairwise</code> 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 <code>mappoly.genoprob</code> 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 <code>mappoly.data</code>
ph.thres	the LOD threshold used to define the linkage phase configurations to test

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

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

### Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

### Examples

```
##### Tetraploid example #####
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])
```

---

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,
  verbose = TRUE,
  ...
)
```

**Arguments**

x	an object of class <code>mappoly.homoprob</code>
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 its name. If NULL (default), the function plots the first individual
use.plotly	if TRUE (default), it uses plotly interactive graphics
verbose	if TRUE (default), the current progress is shown; if FALSE, no output is produced
...	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 <code>mappoly.prefpair.profiles</code>
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 <code>mappoly.map</code>
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**

```
## 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)
```

---

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**

<code>map.list</code>	A list of objects or a single object of class <code>mappoly.map</code>
<code>horiz</code>	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
<code>col</code>	a vector of colors for the bars or bar components (default = 'lightgrey') <code>ggstyle</code> produces maps using the default <code>ggplot</code> color palette
<code>title</code>	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

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

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

---

plot_mrk_info	<i>Plot marker information</i>
---------------	--------------------------------

---

## Description

Plots summary statistics for a given marker

## Usage

```
plot_mrk_info(input.data, mrk)
```

## Arguments

input.data	an object of class <code>mappoly.data</code>
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

```
plot_mrk_info(tetra.solcap.geno.dist, 2680)
plot_mrk_info(tetra.solcap.geno.dist, "solcap_snp_c2_23828")
```

---

poly\_cross\_simulate    *Simulate an autopolyploid full-sib population*

---

### Description

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

### Usage

```
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**

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

---

read\_fitpoly

*Data Input in fitPoly format*


---

**Description**

Reads an external data file generated as output of [saveMarkerModels](#). This function creates an object of class `mappoly.data`.

**Usage**

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

**Arguments**

<code>file.in</code>	a character string with the name of (or full path to) the input file
<code>ploidy</code>	the ploidy level
<code>parent1</code>	a character string containing the name (or pattern of genotype IDs) of parent 1
<code>parent2</code>	a character string containing the name (or pattern of genotype IDs) of parent 2
<code>offspring</code>	a character string containing the name (or pattern of genotype IDs) of the offspring individuals. If NULL (default) it considers all individuals as offsprings, except <code>parent1</code> and <code>parent2</code> .
<code>filter.non.conforming</code>	if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function <a href="#">segreg_poly</a> for information on expected classes and their respective frequencies.
<code>elim.redundant</code>	logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to in order to include them in the final map.

parent.geno	indicates whether to use the joint probability 'joint' (default) or the maximum probability of multiple replicates (if available) to assign dosage to parents. If there is one observation per parent, both options will yield the same results.
thresh.parent.geno	threshold probability to assign a dosage to parents. If the probability is smaller than thresh.parent.geno, the marker is discarded.
prob.thres	threshold probability to assign a dosage to offspring. If the probability is smaller than thresh.parent.geno, the data point is converted to 'NA'.
file.type	indicates whether the characters in the input file are separated by 'white spaces' ("table") or by commas ("csv").
verbose	if TRUE (default), the current progress is shown; if FALSE, no output is produced

### Value

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

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

### Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

## References

Voorrips, R.E., Gort, G. & Vosman, B. (2011) Genotype calling in tetraploid species from bi-allelic marker data using mixture models. *BMC Bioinformatics*. <https://doi.org/10.1186/1471-2105-12-172>

## Examples

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

---

read\_genotype

*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_genotype(
  file.in,
  filter.non.conforming = TRUE,
  elim.redundant = TRUE,
  verbose = TRUE
)

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

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

**Arguments**

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

<code>m</code>	ploidy level
<code>n.ind</code>	number individuals
<code>n.mrk</code>	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

```
#### Tetraploid Example
f11 = "https://raw.githubusercontent.com/mmollina/MAPpoly_vignettes/master/data/SolCAP_dosage"
tempf1 <- tempfile()
download.file(f11, destfile = tempf1)
SolCAP.dose <- read_genotype(file.in = tempf1)
print(SolCAP.dose, detailed = TRUE)
plot(SolCAP.dose)
```



---

read_genovcsv	<i>Data Input in CSV format</i>
---------------	---------------------------------

---

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

### Arguments

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

### Details

This is an alternative and a somewhat more straightforward version of the function [read\\_genov](#). 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:

<code>m</code>	ploidy level
<code>n.ind</code>	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

```
#### 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_genovcsv(file.in = tempfl, ploidy = 4)
print(SolCAP.dose, detailed = TRUE)
plot(SolCAP.dose)
```

---

read_genotype	<i>Data Input</i>
---------------	-------------------

---

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

### Arguments

<code>file.in</code>	a character string with the name of (or full path to) the input file which contains the data to be read
<code>prob.thres</code>	probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than <code>prob.thres</code> are considered as missing data for the dosage calling purposes (default = 0.95)
<code>filter.non.conforming</code>	if TRUE (default) converts data points with unexpected genotypes (i.e. no double reduction) to 'NA'. See function <a href="#">segseg_poly</a> for information on expected classes and their respective frequencies.
<code>elim.redundant</code>	logical. If TRUE (default), removes redundant markers during map construction, keeping them annotated to export to the final map.
<code>verbose</code>	if TRUE (default), the current progress is shown; if FALSE, no output is produced

### Details

The first line of the input file contains the string `ploidy` followed by the ploidy level of the parents. The second and third lines 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:

<code>m</code>	ploidy level
<code>n.ind</code>	number individuals
<code>n.mrk</code>	total number of markers
<code>ind.names</code>	the names of the individuals
<code>mrk.names</code>	the names of the markers
<code>dosage.p</code>	a vector containing the dosage in parent P for all <code>n.mrk</code> markers
<code>dosage.q</code>	a vector containing the dosage in parent Q for all <code>n.mrk</code> markers
<code>sequence</code>	a vector indicating which sequence each marker belongs. Zero indicates that the marker was not assigned to any sequence
<code>sequence.pos</code>	physical position of the markers into the sequence
<code>seq.ref</code>	NULL (unused in this type of data)
<code>seq.alt</code>	NULL (unused in this type of data)
<code>all.mrk.depth</code>	NULL (unused in this type of data)
<code>prob.thres</code>	probability threshold to associate a marker call to a dosage. Markers with maximum genotype probability smaller than <code>'prob.thres'</code> were considered as missing data in the <code>'geno.dose'</code> matrix
<code>geno.dose</code>	a matrix containing the dosage for each markers (rows) for each individual (columns). Missing data are represented by <code>ploidy_level + 1</code>
<code>geno</code>	a <code>data.frame</code> 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 <a href="#">segreg_poly</a>
<code>n.phen</code>	number of phenotypic traits
<code>phen</code>	a matrix containing the phenotypic data. The rows correspond to the traits and the columns correspond to the individuals
<code>chisq.pval</code>	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

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

---

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,
    verbose = TRUE
  )

```

### Arguments

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

### 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:

<code>m</code>	ploidy level
<code>n.ind</code>	number individuals
<code>n.mrk</code>	total number of markers
<code>ind.names</code>	the names of the individuals
<code>mrk.names</code>	the names of the markers
<code>dosage.p</code>	a vector containing the dosage in parent P for all <code>n.mrk</code> markers
<code>dosage.q</code>	a vector containing the dosage in parent Q for all <code>n.mrk</code> markers
<code>sequence</code>	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

```
## Hexaploid sweetpotato: Subset of chromosome 3
f1 = "https://github.com/mmollina/MAppoly_vignettes/raw/master/data/sweet_sample_ch3.vcf.gz"
tempf1 <- tempfile(pattern = 'chr3_', fileext = '.vcf.gz')
download.file(f1, destfile = tempf1)
dat.dose.vcf = read_vcf(file = tempf1, parent.1 = "PARENT1", parent.2 = "PARENT2")
print(dat.dose.vcf)
plot(dat.dose.vcf)
```

reest\_rf

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

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

**Usage**

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

**Arguments**

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

**Value**

An updated object of class `mappoly.pcmmap` whose order was used in the `input.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	<i>Reverse map</i>
---------	--------------------

---

## 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

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

---

rf_list_to_matrix	<i>Recombination fraction list to matrix</i>
-------------------	--

---

## 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,
  fact = 1,
  ...
)

```

**Arguments**

input.twopt	an object of class <code>poly.est.two.pts.pairwise</code>
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 <code>mappoly.rf.matrix</code>
...	currently ignored
type	type of matrix that should be printed. Can be one of the following: "rf", for recombination fraction or "lod" for LOD Score
ord	the order in which the markers should be plotted (default = NULL)
rem	which markers should be removed from the heatmap (default = NULL)
main.text	a character string as the title of the heatmap (default = NULL)
index	logical should the name of the markers be printed in the diagonal of the heatmap? (default = FALSE)

fact                    positive integer. factor expressed as number of cells to be aggregated (default = 1, no aggregation)

### Details

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

### Value

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

### Author(s)

Marcelo Mollinari, <mmollin@ncsu.edu>

### References

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

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

---

rf\_snp\_filter

*Remove markers that do not meet a LOD criteria*

---

### Description

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

**Usage**

```
rf_snp_filter(  
  input.twopt,  
  thresh.LOD.ph = 5,  
  thresh.LOD.rf = 5,  
  thresh.rf = 0.15,  
  probs = c(0.05, 1),  
  ncpus = 1L,  
  diagnostic.plot = TRUE  
)
```

**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)

`probs` indicates the probability corresponding to the filtering quantiles. (default = `c(0.05, 1)`)

`ncpus` number of parallel processes (i.e. cores) to spawn (default = 1)

`diagnostic.plot` if TRUE produces a diagnostic plot

**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**

```

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

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

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

```

---

segreg\_poly

*Polysomic segregation frequency*


---

**Description**

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

**Usage**

```
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	<i>Simulate homology groups</i>
----------------	---------------------------------

---

## 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	<i>Resulting maps from <a href="#">tetra.solcap.geno.dist</a></i>
------------------	---

---

**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	<i>Divides map in sub-maps and re-phase them</i>
-------------------	--

---

**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,
  verbose = TRUE
)
```

**Arguments**

<code>input.map</code>	an object of class <code>mappoly.map</code>
<code>twopt</code>	an object of class <code>poly.est.two.pts.pairwise</code> containing the two-point information for the markers contained in <code>input.map</code>
<code>gap.threshold</code>	distance threshold of neighboring markers where the map should be spitted. The default value is 5 cM
<code>remove.single</code>	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)  
**verbose**            if TRUE (default), the current progress is shown; if FALSE, no output is produced

**Value**

An object of class `mappoly.map`

**Author(s)**

Marcelo Mollinari, <mmollin@ncsu.edu>

**References**

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**

```

map <- get_submap(maps.hexafake[[1]], 1:20, reestimate.rf = FALSE, reestimate.phase = FALSE)
tpt <- est_pairwise_rf(make_seq_mappoly(map))
new.map <- split_and_rephase(map, tpt, 5)
map
new.map
plot_map_list(list(old.map = map, new.map = new.map))
  
```

---

summary\_maps

*Summary maps*

---

**Description**

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

**Usage**

```
summary_maps(map.list, verbose = TRUE)
```

**Arguments**

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

**Value**

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

**Author(s)**

Gabriel Gesteira, <gabrielgesteira@usp.br>

**Examples**

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

---

tetra.solcap	<i>Autotetraploid potato dataset.</i>
--------------	---------------------------------------

---

**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](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](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	<i>Update map</i>
------------	-------------------

---

**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.maps, verbose = TRUE)
```

**Arguments**

<code>input.maps</code>	a single map or a list of maps of class <code>mappoly.map</code>
<code>verbose</code>	if TRUE (default), shows information about each update process

**Value**

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

**Author(s)**

Gabriel Gesteira, <gabrielgesteira@usp.br>

**Examples**

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

---

update_missing	<i>Update missing information</i>
----------------	-----------------------------------

---

**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**

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

# Index

- \* **analysis**
  - cache\_counts\_twopt, 6
- \* **datasets**
  - hexafake, 36
  - hexafake.geno.dist, 37
  - maps.hexafake, 47
  - solcap.dose.map, 79
  - solcap.err.map, 80
  - solcap.mds.map, 80
  - solcap.prior.map, 81
  - tetra.solcap, 83
  - tetra.solcap.geno.dist, 84
- \* **two-point**
  - cache\_counts\_twopt, 6
- add\_marker, 3
- cache\_counts\_twopt, 6, 7, 20
- calc\_genoprob, 8
- calc\_genoprob\_dist, 9
- calc\_genoprob\_error, 10
- calc\_homoprob, 11
- calc\_prefpair\_profiles, 12
- check\_data\_sanity, 13
- drop\_marker, 14
- elim\_redundant, 14
- est\_full\_hmm\_with\_global\_error, 15
- est\_full\_hmm\_with\_prior\_prob, 17
- est\_pairwise\_rf, 19, 45
- est\_rf\_hmm, 21
- est\_rf\_hmm\_sequential, 23, 24
- export\_data\_to\_polymapR, 28
- export\_map\_list, 29
- extract\_map, 29
- filter\_missing, 30
- filter\_segregation, 31
- get\_genomic\_order, 32
- get\_submap, 32
- get\_tab\_mrks, 34
- group\_mappoly, 35
- hexafake, 36, 37, 47
- hexafake.geno.dist, 37
- import\_data\_from\_polymapR, 38
- import\_from\_updog, 39
- import\_phased\_maplist\_from\_polymapR, 41
- loglike\_hmm, 42
- make\_mat\_mappoly, 43
- make\_pairs\_mappoly, 44
- make\_seq\_mappoly, 45, 76
- maps.hexafake, 47
- mds\_mappoly, 47, 80
- merge\_datasets, 49
- merge\_maps, 51
- plot.mappoly.data (read\_geno), 62
- plot.mappoly.homoprob, 53
- plot.mappoly.map (est\_rf\_hmm), 21
- plot.mappoly.prefpair\_profiles, 54
- plot.mappoly.rf.matrix (rf\_list\_to\_matrix), 73
- plot.mappoly.sequence (make\_seq\_mappoly), 45
- plot\_genome\_vs\_map, 55
- plot\_map\_list, 56
- plot\_mrk\_info, 57
- poly\_cross\_simulate, 58
- print.mappoly.data (read\_geno), 62
- print.mappoly.map (est\_rf\_hmm), 21
- print.mappoly.pcmmap (mds\_mappoly), 47
- print.mappoly.pcmmap3d (mds\_mappoly), 47
- print.mappoly.rf.matrix (rf\_list\_to\_matrix), 73

print.mappoly.sequence  
    (make\_seq\_mappoly), 45  
print\_mrk, 59

read\_fitpoly, 60  
read\_geno, 59, 62, 65  
read\_geno\_csv, 65  
read\_geno\_prob, 67  
read\_vcf, 69  
reest\_rf, 72  
rev\_map, 73  
rf\_list\_to\_matrix, 4, 44, 73  
rf\_snp\_filter, 75

saveMarkerModels, 60  
segreg\_poly, 40, 60, 63, 65, 67, 68, 70, 77  
sim\_homologous, 78  
solcap.dose.map, 79  
solcap.err.map, 80  
solcap.mds.map, 80  
solcap.prior.map, 81  
split\_and\_rephase, 81  
summary\_maps, 82

tetra.solcap, 79, 80, 83, 84  
tetra.solcap.geno.dist, 81, 84

update\_map, 85  
update\_missing, 85