Package ‘onemap’

February 17, 2020

Title  Construction of Genetic Maps in Experimental Crosses: Full-Sib, RILs, F2 and Backcrosses

Version  2.1.3

Description  Analysis of molecular marker data from model (backcrosses, F2 and recombinant inbred lines) and non-model systems (i. e. outcrossing species). For the later, it allows statistical analysis by simultaneously estimating linkage and linkage phases (genetic map construction) according to Wu et al. (2002) <doi:10.1006/tpbi.2002.1577>. All analysis are based on multipoint approaches using hidden Markov models.

Date  2020-02-14

Author  Gabriel Margarido [aut], Marcelo Mollinari [aut], Karl Broman [ctb], Getulio Ferreira [ctb], Rodrigo Amadeu [ctb], Cristiane Taniguti [ctb, cre], Augusto Garcia [aut, ctb]

LinkingTo  Rcpp (>= 0.10.5)

Depends  R (>= 3.6.0)

Imports  ggplot2 (>= 2.2.1), plotly (>= 4.7.1), reshape2 (>= 1.4.1), Rcpp (>= 0.10.5), graphics, methods, stats, utils, grDevices, MDSMap

Suggests  qtl (>= 1.36-6), knitr (>= 1.10), rmarkdown, htmlwidgets, vcfR (>= 1.6.0),

VignetteBuilder  knitr

Encoding  UTF-8

License  GPL-3

URL  https://github.com/augusto-garcia/onemap

BugReports  https://github.com/augusto-garcia/onemap/wiki

Maintainer  Cristiane Taniguti <chtaniguti@usp.br>
R topics documented:

add_marker ............................................................. 3
Bonferroni_alpha ...................................................... 4
combine_onemap ......................................................... 5
compare ................................................................. 6
create_dataframe_for_plot_outcross ................................. 8
create_data_bins ....................................................... 9
draw_map ............................................................ 10
draw_map2 ............................................................. 11
drop_marker ........................................................... 13
find_bins ............................................................. 14
group ................................................................. 15
group_seq ............................................................ 16
make_seq ............................................................. 18
map ................................................................. 20
mapmaker_example_bc .................................................. 22
mapmaker_example_f2 ................................................... 23
map_func ............................................................ 24
marker_type ........................................................... 25
mds_onemap ........................................................... 26
onemap_example_bc ....................................................... 28
onemap_example_f2 ....................................................... 29
onemap_example_out .................................................... 30
onemap_example_riself ................................................ 31
onemap_read_vcfR ..................................................... 32
order_seq ............................................................ 33
plot.onemap ........................................................... 36
plot.onemap_segreg_test .............................................. 37
plot_by_segreg_type ................................................... 38
print.onemap_segreg_test ............................................. 39
rcd ................................................................. 40
read_mapmaker ........................................................ 42
read_onemap .......................................................... 43
record ............................................................... 45
rf_2pts ............................................................. 47
rf_graph_table ........................................................ 48
ripple_seq ........................................................... 50
select_segreg ......................................................... 52
seriation ............................................................. 53
add_marker

Description

Creates a new sequence by adding markers from a predetermined one. The markers are added in the end of the sequence.

Usage

add_marker(input.seq, mrks)

Arguments

input.seq an object of class sequence.
mrks a vector containing the markers to be added from the sequence.

Value

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

- seq.num a vector containing the (ordered) indices of markers in the sequence, according to the input file.
- seq.phases a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.
- seq.rf a vector with the recombination fractions between markers in the sequence. -1 means that there are no estimated recombination fractions.
- seq.like log-likelihood of the corresponding linkage map.
- data.name name of the object of class onemap with the raw data.
- twopt name of the object of class rf_2pts with the 2-point analyses.

@author Marcelo Mollinari, <mmollina@usp.br>
Bonferroni_alpha

See Also
drop_marker

Examples
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
(LG1 <- make_seq(groups,1))
(LG.aug<-add_marker(LG1, c(4,7)))

twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
(LG1 <- make_seq(groups,1))
(LG.aug<-add_marker(LG1, c(4,7)))

Bonferroni_alpha  Calculates individual significance level to be used to achieve a global alpha (with Bonferroni)

Description
It shows the alpha value to be used in each chi-square segregation test, in order to achieve a given global type I error. To do so, it uses Bonferroni’s criteria.

Usage
Bonferroni_alpha(x, global.alpha = 0.05)

Arguments
x an object of class onemap_segreg_test

Arguments
x an object of class onemap_segreg_test

global.alpha the global alpha that

Value
the alpha value for each test (numeric)

Examples
data(onemap_example_bc) # Loads a fake backcross dataset installed with onemap
Chi <- test_segregation(onemap_example_bc) # Performs the chi-square test for all markers
print(Chi) # Shows the results of the Chi-square tests
Bonferroni_alpha (Chi) # Shows the individual alpha level to be used
Combine OneMap datasets

Description

Merge two or more OneMap datasets from the same cross type. Creates an object of class onemap.

Usage

combine_onemap(...)

Arguments

... Two or more onemap dataset objects of the same cross type.

Details

Given a set of OneMap datasets, all from the same cross type (full-sib, backcross, F2 intercross or recombinant inbred lines obtained by self- or sib-mating), merges marker and phenotype information to create a single onemap object.

If sample IDs are present in all datasets (the standard new format), not all individuals need to be genotyped in all datasets - the merged dataset will contain all available information, with missing data elsewhere. If sample IDs are missing in at least one dataset, it is required that all datasets have the same number of individuals, and it is assumed that they are arranged in the same order in every dataset.

Value

An object of class onemap, i.e., a list with the following components:

- geno: a matrix with integers indicating the genotypes read for each marker. Each column contains data for a marker and each row represents an individual.
- n.ind: number of individuals.
- n.mar: number of markers.
- segr.type: a vector with the segregation type of each marker, as strings.
- segr.type.num: a vector with the segregation type of each marker, represented in a simplified manner as integers, i.e. 1 corresponds to markers of type "A"; 2 corresponds to markers of type "B1.5"; 3 corresponds to markers of type "B2.6"; 4 corresponds to markers of type "B3.7"; 5 corresponds to markers of type "C.8"; 6 corresponds to markers of type "D1" and 7 corresponds to markers of type "D2". Markers for F2 intercrosses are coded as 1; all other crosses are left as NA.
- input: a string indicating that this is a combined dataset.
- n.phe: number of phenotypes.
- pheno: a matrix with phenotypic values. Each column contains data for a trait and each row represents an individual.
compare

Compare all possible orders (exhaustive search) for a given sequence of markers

Description

For a given sequence with \( n \) markers, computes the multipoint likelihood of all \( \frac{n!}{2^n} \) possible orders.

Usage

\[
\text{compare}(\text{input.seq}, \text{n.best} = 50, \text{tol} = 0.001, \text{verbose} = \text{FALSE})
\]

Arguments

- **input.seq**: an object of class `sequence`.
- **n.best**: the number of best orders to store in object (defaults to 50).
- **tol**: tolerance for the C routine, i.e., the value used to evaluate convergence.
- **verbose**: if FALSE (default), simplified output is displayed. if TRUE, detailed output is displayed.

See Also

`read_onemap` and `read_mapmaker`.

Examples

```r
## Not run:
combined_data <- combine_onemap(onemap_data1, onemap_data2)

## End(Not run)
```
Details

Since the number \( n! \) is large even for moderate values of \( n \), this function is to be used only for sequences with relatively few markers. If markers were genotyped in an outcross population, linkage phases need to be estimated and therefore more states need to be visited in the Markov chain; when segregation types are D1, D2 and C, computation can required a very long time (specially when markers linked in repulsion are involved), so we recomand to use this function up to 6 or 7 markers. For inbred-based populations, up to 10 or 11 markers can be ordered with this function, since linkage phase are known. The multipoint likelihood is calculated according to Wu et al. (2002b) (Eqs. 7a to 11), assuming that the recombination fraction is the same in both parents. Hidden Markov chain codes adapted from Broman et al. (2008) were used.

Value

An object of class `compare`, which is a list containing the following components:

- `best.ord` a matrix containing the best orders.
- `best.ord.rf` a matrix with recombination frequencies for the corresponding best orders.
- `best.ord.phase` a matrix with linkage phases for the best orders.
- `best.ord.like` a vector with log-likelihood values for the best orders.
- `best.ord.LOD` a vector with LOD Score values for the best orders.
- `data.name` name of the object of class `onemap` with the raw data.
- `twopt` name of the object of class `rf_2pts` with the 2-point analyses.

Author(s)

Marcelo Mollinari, <mmollina@usp.br>

References


See Also

`marker_type` for details about segregation types and `make_seq`.
create_dataframe_for_plot_outcross

Create a dataframe suitable for a ggplot2 graphic

Description
An internal function that prepares a dataframe suitable for drawing a graphic of raw data using ggplot2, i.e., a data frame with long format

Usage
create_dataframe_for_plot_outcross(x)

Arguments
x an object of classes onemap and outcross, with data and additional information

Value
a dataframe

Examples

## Not run:
#outcrossing example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
markers <- make_seq(twopt,c(12,14,15,26,28))
(markers.comp <- compare(markers))
(markers.comp <- compare(markers,verbose=TRUE))

#F2 example
data(onemap_example_f2)
twopt <- rf_2pts(onemap_example_f2)
markers <- make_seq(twopt,c(17,26,29,30,44,46,55))
(markers.comp <- compare(markers))
(markers.comp <- compare(markers,verbose=TRUE))

## End(Not run)
create_data_bins

Description

Creates a new dataset based on onemap_bin object

Usage

create_data_bins(input.obj, bins)

Arguments

input.obj  an object of class onemap.
bins       an object of class onemap_bin.

Details

Given a onemap_bin object, creates a new data set where the redundant markers are collapsed into bins and represented by the marker with the lower amount of missing data among those on the bin.

Value

an object of class onemap.

Author(s)

Marcelo Mollinari, <mmollina@usp.br>

See Also

find_bins

Examples

## Not run:
load(url("https://github.com/mmollina/data/raw/master/fake_big_data_f2.RData"))
fake.big.data.f2
(bins <- find_bins(fake.big.data.f2, exact=FALSE))
(new.data <- create_data_bins(fake.big.data.f2, bins))
## End(Not run)
draw_map  

Description

Provides a simple draw of a genetic map.

Usage

draw_map(
    map.list,
    horizontal = FALSE,
    names = FALSE,
    grid = FALSE,
    cex.mrk = 1,
    cex.grp = 0.75
)

Arguments

map.list a map, i.e. an object of class sequence with a predefined order, linkage phases, recombination fraction and likelihood; also it could be a list of maps.
horizontal if TRUE, indicates that the map should be plotted horizontally. Default is FALSE
names if TRUE, displays the names of the markers. Default is FALSE
grid if TRUE, displays a grid in the background. Default is FALSE
cex.mrk the magnification to be used for markers.
cex.grp the magnification to be used for group axis annotation.

Author(s)

Marcelo Mollinari, <mmollina@usp.br>

Examples

## Not run:
#outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
lg<-group(make_seq(twopt, "all"))
maps<-vector("list", lg$n.groups)
for(i in 1:lg$n.groups)
   maps[[i]]<- make_seq(order_seq(input.seq= make_seq(lg,i),twopt.alg = "rcd"), "force")
draw_map(maps, grid=TRUE)
draw_map(maps, grid=TRUE, horizontal=TRUE)
#F2 example
data(onemap_example_f2)
twopt<-rf_2pts(onemap_example_f2)
lg<-group(make_seq(twopt, "all"))
maps<-vector("list", lg$n.groups)
for(i in 1:lg$n.groups)
  maps[[i]]<- make_seq(order_seq(input.seq= make_seq(lg,i), twopt.alg = "rcd"), "force")
draw_map(maps, grid=TRUE)
draw_map(maps, grid=TRUE, horizontal=TRUE)

## End(Not run)

draw_map2

Draw a linkage map

Description

Provides a simple draw of a linkage map.

Usage

draw_map2(
  ..., 
  tag = NULL, 
  id = TRUE, 
  pos = TRUE, 
  cex.label = NULL, 
  main = NULL, 
  group.names = NULL, 
  centered = F, 
  y.axis = TRUE, 
  space = NULL, 
  col.group = NULL, 
  col.mark = NULL, 
  col.tag = NULL, 
  output = NULL 
)

Arguments

... map(s). Object(s) of class sequence and/or data.frame. If data.frame, it must have two columns: column 1: marker id; column 2: position (cM) (numeric).
tag name(s) of the marker(s) to highlight. If "all", all markers will be highlighted. Default is NULL.
?id logical. If TRUE (default), shows name(s) of tagged marker(s).
pos logical. If TRUE (default), shows position(s) of tagged marker(s).
cex.label the magnification used for label(s) of tagged marker(s). If NULL (default), the
cex will be automatically calculated to avoid overlapping.
main an overall title for the plot. Default is NULL.
group.names name(s) to identify the group(s). If NULL (default), the name(s) of the sequence(s)
will be used.
centered logical. If TRUE, the group(s) will be aligned in the center. If FALSE (default),
the group(s) will be aligned at the top.
y.axis logical. If TRUE (default), shows y axis. If centered = TRUE, the y axis will
always be hidden.
space numerical. Spacing between groups. If NULL (default), the spacing will be auto-
matically calculated to avoid overlapping.
col.group the color used for group(s).
col.mark the color used for marker(s).
col.tag the color used for highlighted marker(s) and its/their label(s).
output the name of the output file. The file format can be specified by adding its exten-

Author(s)
Getulio Caixeta Ferreira, <getulio.caifer@gmail.com>

Examples

```r
## Not run:
data("onemap_example_out")
twopt <- rf_2pts(onemap_example_out)
lg<-group(make_seq(twopt, "all"))
seq1<make_seq(order_seq(input.seq= make_seq(lg,1),twopt.alg = "rcd"), "force")
seq2<make_seq(order_seq(input.seq= make_seq(lg,2),twopt.alg = "rcd"), "force")
seq3<make_seq(order_seq(input.seq= make_seq(lg,3),twopt.alg = "rcd"), "force")
draw_map2(seq1,seq2,seq3,tag = c("M1","M2","M3","M4","M5"))

data("onemap_example_f2")
twopt <- rf_2pts(onemap_example_f2)
lg<-group(make_seq(twopt, "all"))
seq<-list(  make_seq(order_seq(input.seq= make_seq(lg,1),twopt.alg = "rcd"), "force"),
make_seq(order_seq(input.seq= make_seq(lg,2),twopt.alg = "rcd"), "force"),
make_seq(order_seq(input.seq= make_seq(lg,3),twopt.alg = "rcd"), "force")
  )
draw_map2(seq,tag = "all",group.names = c("Chr 1","Chr 2","Chr 3"),main="Linkage Map")

## End(Not run)
```
**drop_marker**  
Creates a new sequence by dropping markers.

**Description**  
Creates a new sequence by dropping markers from a predetermined one.

**Usage**  
`drop_marker(input.seq, mrks)`

**Arguments**

- **input.seq**: an object of class sequence.
- **mrks**: a vector containing the markers to be removed from the sequence.

**Value**

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

- **seq.num**: a vector containing the (ordered) indices of markers in the sequence, according to the input file.
- **seq.phases**: a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.
- **seq.rf**: a vector with the recombination fractions between markers in the sequence. -1 means that there are no estimated recombination fractions.
- **seq.like**: log-likelihood of the corresponding linkage map.
- **data.name**: name of the object of class onemap with the raw data.
- **twopt**: name of the object of class rf_2pts with the 2-point analyses.

@author Marcelo Mollinari, <mmollina@usp.br>

**See Also**

- `add_marker`

**Examples**

```r
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
(LG1 <- make_seq(groups,1))
(LG.aug<-drop_marker(LG1, c(10,14)))
```
find_bins

Allocate markers into bins

Description

Function to allocate markers with redundant information into bins. Within each bin, the pairwise recombination fraction between markers is zero.

Usage

find_bins(input.obj, exact = TRUE, ch = NULL)

Arguments

input.obj an object of class onemap.

exact logical. If TRUE, it only allocates markers with the exact same information into bins, including missing data; if FALSE, missing data are not considered when allocating markers. In the latter case, the marker with the lowest amount of missing data is taken as the representative marker on that bin.

ch not used in this OneMap version. Chromosome for which the analysis should be performed. If NULL the analysis is performed for all chromosomes.

Value

An object of class onemap_bin, which is a list containing the following components:

bins a list containing the bins. Each element of the list is a table whose lines indicate the name of the marker, the bin in which that particular marker was allocated and the percentage of missing data. The name of each element of the list corresponds to the marker with the lower amount of missing data among those on the bin

n.mar total number of markers.

n.ind number individuals

exact.search logical; indicates if the search was performed with the argument exact=TRUE or exact=FALSE

Author(s)

Marcelo Mollinari, <mmollina@usp.br>

See Also

create_data_bins
group

Examples

```r
## Not run:
load(url("https://github.com/mmollina/data/raw/master/fake_big_data_f2.RData"))
fake.big.data.f2
(bins<-find_bins(fake.big.data.f2, exact=FALSE))
## End(Not run)
```

```
Assign markers to linkage groups

Description

Identifies linkage groups of markers, using results from two-point (pairwise) analysis and the transitive property of linkage.

Usage

```r
group(input.seq, LOD = NULL, max.rf = NULL, verbose = TRUE)
```

Arguments

- `input.seq`: an object of class `sequence`.
- `LOD`: a (positive) real number used as minimum LOD score (threshold) to declare linkage.
- `max.rf`: a real number (usually smaller than 0.5) used as maximum recombination fraction to declare linkage.
- `verbose`: logical. If `TRUE`, current progress is shown; if `FALSE`, no output is produced.

Details

If the arguments specifying thresholds used to group markers, i.e., minimum LOD Score and maximum recombination fraction, are `NULL` (default), the values used are those contained in object `input.seq`. If not using `NULL`, the new values override the ones in object `input.seq`.

Value

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

- `data.name`: name of the object of class `onemap` that contains the raw data.
- `twopt`: name of the object of class `rf.2ts` used as input, i.e., containing information used to assign markers to linkage groups.
- `marnames`: marker names, according to the input file.
- `n.mar`: total number of markers.
- `LOD`: minimum LOD Score to declare linkage.
- `max.rf`: maximum recombination fraction to declare linkage.
- `n.groups`: number of linkage groups found.
- `groups`: number of the linkage group to which each marker is assigned.
Assign markers to preexisting linkage groups

Identifies linkage groups of markers combining input sequences objects with unlinked markers from rf_2pts object. The results from two-point (pairwise) analysis and the transitive property of linkage are used for grouping, as group function.

Usage

```r
group_seq(
  input.2pts,
  seqs = "CHROM",
  unlink.mks = "all",
  repeated = FALSE,
  LOD = NULL,
  max.rf = NULL
)
```
Arguments

- **input.2pts**: an object of class `rf_2pts`.
- **seqs**: a list of objects of class `sequence` or the string "CHROM" if there is CHROM information available in the input data file.
- **unlink.mks**: an object of class `sequence` with the number of the markers to be grouped with the preexisting sequences defined by `seqs` parameter. Using the string "all", all remaining markers of the `rf_2pts` object will be tested.
- **repeated**: logical. If `TRUE`, markers grouped in more than one of the sequences are kept in the output sequences. If `FALSE`, they are removed of the output sequences.
- **LOD**: a (positive) real number used as minimum LOD score (threshold) to declare linkage.
- **max.rf**: a real number (usually smaller than 0.5) used as maximum recombination fraction to declare linkage.

Details

If the arguments specifying thresholds used to group markers, i.e., minimum LOD Score and maximum recombination fraction, are `NULL` (default), the values used are those contained in object `input.2pts`. If not using `NULL`, the new values override the ones in object `input.2pts`.

Value

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

- **data.name**: name of the object of class `onemap` that contains the raw data.
- **twopt**: name of the object of class `rf.2ts` used as input, i.e., containing information used to assign markers to linkage groups.
- **mk.names**: marker names, according to the input file.
- **input.seqs**: list with the numbers of the markers in each inputted sequence
- **input.unlink.mks**: numbers of the unlinked markers in inputted sequence
- **out.seqs**: list with the numbers of the markers in each outputted sequence
- **n.unlinked**: number of markers that remained unlinked
- **n.repeated**: number of markers which repeated in more than one group
- **n.mar**: total number of markers evaluated
- **LOD**: minimum LOD Score to declare linkage.
- **max.rf**: maximum recombination fraction to declare linkage.
- **sequences**: list of outputted sequences
- **repeated**: list with the number of the markers that are repeated in each outputted sequence
- **unlinked**: number of the markers which remained unlinked

Author(s)

Cristiane Taniguti, <chtaniguti@usp.br>
make_seq

Create a sequence of markers

Description

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

Usage

make_seq(input.obj, arg = NULL, phase = NULL, data.name = NULL, twopt = NULL)

Arguments

input.obj  
an object of class onemap, rf_2pts, group, compare, try or order.
arg  
its value depends on the type of object input.obj. For a onemap object, arg must be a string corresponding to one of the reference sequences on which markers are anchored (usually chromosomes). This requires that CHROM information be available in the input data file. It can also be a vector of integers specifying which markers comprise the sequence. For an object rf_2pts, arg can be the string "all", resulting in a sequence with all markers in the raw data (generally done for grouping markers); otherwise, it must be a vector of integers specifying which markers comprise the sequence. For an object of class group, arg must be an integer specifying the group. For a compare object, arg is an integer indicating the corresponding order (arranged according to the likelihood); if NULL (default), the best order is taken. For an object of class try, arg must be an integer less than or equal to the length of the original sequence plus one;
the sequence obtained will be that with the additional marker in the position indicated by arg. Finally, for an order object, arg is a string: "safe" means the order that contains only markers mapped with the provided threshold; "force" means the order with all markers.

phase

its value is also dependent on the type of input.obj. For an rf_2pts or onemap object, phase can be a vector with user-defined linkage phases (its length is equal to the number of markers minus one); if NULL (default), other functions will try to find the best linkage phases. For example, if phase takes on the vector c(1,2,3,4), the sequence of linkage phases will be coupling/coupling, coupling/repulsion, repulsion/coupling and repulsion/repulsion for a sequence of five markers. If input.obj is of class compare or try, this argument indicates which combination of linkage phases should be chosen, for the particular order given by argument arg. In both cases, NULL (default) makes the best combination to be taken. If input.obj is of class, group or order, this argument has no effect.

data.name

a string indicating the name of the object which contains the raw data. This does not have to be defined by the user: it is here for compatibility issues when calling make_seq from inside other functions.

twopt

a string indicating the name of the object which contains the two-point information. This does not have to be defined by the user: it is here for compatibility issues when calling make_seq from inside other functions.

Value

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

seq.num

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

seq.phases

a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.

seq.rf

a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies.

seq.like

log-likelihood of the corresponding linkage map.

data.name

name of the object of class onemap with the raw data.

twopt

name of the object of class rf_2pts with the 2-point analyses.

Author(s)

Gabriel Margarido, <gramarga@gmail.com>

References

map

Construct the linkage map for a sequence of markers

Description

Estimates the multipoint log-likelihood, linkage phases and recombination frequencies for a sequence of markers in a given order.

Usage

map(input.seq, tol = 1e-04, verbose = FALSE, mds.seq = FALSE)

Arguments

input.seq an object of class sequence.
tol tolerance for the C routine, i.e., the value used to evaluate convergence.
verbose If TRUE, print tracing information.
mds.seq When some pair of markers do not follow the linkage criteria, if TRUE one of the markers is removed and returns a vector with remaining marker numbers (useful for mds_onemap function).
Details

Markers are mapped in the order defined in the object input.seq. If this object also contains a user-defined combination of linkage phases, recombination frequencies and log-likelihood are estimated for that particular case. Otherwise, the best linkage phase combination is also estimated. The multipoint likelihood is calculated according to Wu et al. (2002b)(Eqs. 7a to 11), assuming that the recombination fraction is the same in both parents. Hidden Markov chain codes adapted from Broman et al. (2008) were used.

Value

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

- seq.num: a vector containing the (ordered) indices of markers in the sequence, according to the input file.
- seq.phases: a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.
- seq.rf: a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies.
- seq.like: log-likelihood of the corresponding linkage map.
- data.name: name of the object of class onemap with the raw data.
- twopt: name of the object of class rf_2pts with the 2-point analyses.

Author(s)

Adapted from Karl Broman (package 'qtl') by Gabriel R A Margarido, <gramarga@usp.br> and Marcelo Mollinari, <mmollina@gmail.com>

References


See Also

make_seq
Examples

```r
## Not run:
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)

markers <- make_seq(twopt,c(30,12,3,14,2)) # correct phases
map(markers)

markers <- make_seq(twopt,c(30,12,3,14,2),phase=c(4,1,4,3)) # incorrect phases
map(markers)
## End(Not run)
```

mapmaker_example_bc

Simulated data from a backcross population

Description

Simulated data set from a backcross population.

Usage

```r
data(mapmaker_example_bc)
```

Format

The format is: List of 8

```r
$ geno : num [1:150, 1:67] 1 2 1 1 2 1 1 2 1 1 ... - attr(*, "dimnames")=List of 2 ...$ : NULL ...
$ : chr [1:167] "M1" "M2" "M3" "M4" ... $ n.ind : num 150 $ n.mar : num 67
$ segr.type : chr [1:67] "A.H" "A.H" "A.H" "A.H" ... $ segr.type.num: logi [1:67] NA NA NA NA NA NA NA NA ... $ input : chr "inst/extdata/mapmaker_example_bc.raw" $ n.phe : num 1 $ pheno : num [1:150, 1] 40.8 39.5 37.9 34.2 38.9 ... ...
```

Details

A total of 150 individuals were genotyped for 67 markers with 15% of missing data. There is one quantitative phenotype to show how to use onemap output as R\qtl input.

Author(s)

Marcelo Mollinari, <mmollina@usp.br>

See Also

read_onemap and read_mapmaker.
Examples

data(mapmaker_example_bc)

# perform two-point analyses
twopts <- rf_2pts(mapmaker_example_bc)
twopts

mapmaker_example_f2  Simulated data from a F2 population

Description

Simulated data set from a F2 population.

Usage

data("mapmaker_example_f2")

Format

The format is: List of 8 $ geno : num [1:200, 1:66] 1 3 2 2 1 0 3 1 3 ... - attr(*, "dimnames")=List

Details

A total of 200 individuals were genotyped for 66 markers (36 co-dominant, i.e. a, ab or b and 30
dominant i.e. c or a and d or b) with 15% of missing data. There is one quantitative phenotype
to show how to use onemap output as R\texttt{qt1} and QTL Cartographer input. Also, it is used for the
analysis in the tutorial that comes with OneMap.

Examples

data(mapmaker_example_f2)

# perform two-point analyses
twopts <- rf_2pts(mapmaker_example_f2)
twopts
map_func

*Mapping functions Haldane and Kosambi*

**Description**

Functions to convert recombination fractions to distance in cM (centiMorgans).

**Usage**

```r
haldane(rcmb)
kosambi(rcmb)
```

**Arguments**

- `rcmb` A recombination fraction between two markers, i.e., a number between 0 and 0.5.

**Details**

Haldane mapping function is defined as

\[ d_M = -\frac{1}{2} \ln(1 - 2r), \]

for \(0 \leq r \leq 0.5\), where \(r\) stands for the recombination fraction in \(rcmb\). Kosambi mapping function is

\[ d_M = \frac{1}{4} \ln \left[ \frac{1 + 2r}{1 - 2r} \right], \]

for \(0 \leq r \leq 0.5\), where \(r\) is defined as above.

**Value**

Both functions return a number with a distance measured in cM.

**Author(s)**

Gabriel R A Margarido, <gramarga@gmail.com>

**References**


marker_type

Examples

# little difference for small recombination fractions
haldane(0.05)
kosambi(0.05)

# greater difference as recombination fraction increases
haldane(0.35)
kosambi(0.35)

marker_type     Informs the segregation patterns of markers

Description

Informs the type of segregation of all markers from an object of class sequence. For outcross populations it uses the notation by Wu et al., 2002. For backcrosses, F2s and RILs, it uses the traditional notation from MAPMAKER i.e. AA, AB, BB, not AA and not BB.

Usage

marker_type(input.seq)

Arguments

input.seq       an object of class sequence.

Details

The segregation types are (Wu et al., 2002):

<table>
<thead>
<tr>
<th>Type</th>
<th>Cross</th>
<th>Segregation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A.1</td>
<td>ab x cd</td>
<td>1:1:1:1</td>
</tr>
<tr>
<td>A.2</td>
<td>ab x ac</td>
<td>1:1:1:1</td>
</tr>
<tr>
<td>A.3</td>
<td>ab x co</td>
<td>1:1:1:1</td>
</tr>
<tr>
<td>A.4</td>
<td>ao x bo</td>
<td>1:1:1:1</td>
</tr>
<tr>
<td>B1.5</td>
<td>ab x ao</td>
<td>1:2:1</td>
</tr>
<tr>
<td>B2.6</td>
<td>ao x ab</td>
<td>1:2:1</td>
</tr>
<tr>
<td>B3.7</td>
<td>ab x ab</td>
<td>1:2:1</td>
</tr>
<tr>
<td>C8</td>
<td>ao x ao</td>
<td>3:1</td>
</tr>
<tr>
<td>D1.9</td>
<td>ab x cc</td>
<td>1:1</td>
</tr>
<tr>
<td>D1.10</td>
<td>ab x aa</td>
<td>1:1</td>
</tr>
<tr>
<td>D1.11</td>
<td>ab x oo</td>
<td>1:1</td>
</tr>
<tr>
<td>D1.12</td>
<td>bo x aa</td>
<td>1:1</td>
</tr>
<tr>
<td>D1.13</td>
<td>ao x oo</td>
<td>1:1</td>
</tr>
<tr>
<td>D2.14</td>
<td>cc x ab</td>
<td>1:1</td>
</tr>
<tr>
<td>D2.15</td>
<td>aa x ab</td>
<td>1:1</td>
</tr>
<tr>
<td>D2.16</td>
<td>oo x ab</td>
<td>1:1</td>
</tr>
<tr>
<td>D2.17</td>
<td>aa x bo</td>
<td>1:1</td>
</tr>
<tr>
<td>D2.18</td>
<td>oo x ao</td>
<td>1:1</td>
</tr>
</tbody>
</table>
Value
Nothing is returned. Segregation types of all markers in the sequence are displayed on the screen.

Author(s)
Gabriel R A Margarido, <gramarga@gmail.com>

References

See Also
make_seq

Examples
```r
data(onemap_example_out)
twopts <- rf_2pts(onemap_example_out)
markers.ex <- make_seq(twopts,c(3,6,8,12,16,25))
marker_type(markers.ex) # segregation type for some markers

data(onemap_example_f2)
twopts <- rf_2pts(onemap_example_f2)
all_mrk<-make_seq(twopts, "all")
lgs<-group(all_mrk)
lg1<-make_seq(lgs,1)
marker_type(lg1) # segregation type for linkage group 1
```
p = NULL,  
n = NULL,  
ispc = TRUE,  
displaytext = FALSE,  
weightfn = "lod2",  
mapfn = "haldane",  
hmm = TRUE,  
mds.seq = TRUE
)

Arguments

input.seq         an object of class sequence
out.file          path to the generated MDSMap input file.
mds.graph.file    path to the graphic generated by MDSMap
p                 Integer - the penalty for deviations from the sphere - higher p forces points more closely onto a sphere.
n                 Vector of integers or strings containing markers to be omitted from the analysis.
ispc              Logical determining the method to be used to estimate the map. By default this is TRUE and the method of principal curves will be used. If FALSE then the constrained MDS method will be used.
displaytext      Shows markers names in analysis graphic view
weightfn         Character string specifying the values to use for the weight matrix in the MDS 'lod2’ or 'lod’.
mapfn             Character string specifying the map function to use on the recombination fractions 'haldane’ is default, 'kosambi’ or 'none’.
hmm               Multipoint genetic distance estimation
mds.seq           When some pair of markers do not follow the linkage criteria, if TRUE one of the markers is removed and mds is performed again.

Details

For better description about MDS method, see MDSMap package vignette.

Value

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

seq.num           a vector containing the (ordered) indices of markers in the sequence, according to the input file.
seq.phases        a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.
seq.rf            a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies.
seq.like          log-likelihood of the corresponding linkage map.
data.name          name of the object of class onemap with the raw data.
twopt             name of the object of class rf_2pts with the 2-point analyses.
Author(s)
Cristiane Taniguti, <chtaniguti@usp.br>

References

See Also
https://CRAN.R-project.org/package=MDSMap.

onemap_example_bc Simulated data from a backcross population

Description
Simulated data set from a backcross population.

Usage
data(onemap_example_bc)

Format
The format is: List of 10 $ geno : num [1:150, 1:67] 1 2 1 2 1 2 1 2 1 2 ... - attr(*, "dimnames")=List of 2 .. $ : chr [1:150] "ID1" "ID2" "ID3" "ID4" ... ..$ : chr [1:67] "M1" "M2" "M3" "M4" ... $ n.ind : int 150 $ n.mar : int 67 $ segr.type : chr [1:67] "A.H" "A.H" "A.H" "A.H" ... $ segr.type.num: logi [1:67] NA NA NA NA NA NA ... $ n.phe : int 1 $ pheno : num [1:150, 1] 40.8 39.5 37.9 34.2 ... $ attr(*, "dimnames")=List of 2 .. $ : NULL ..$ : chr "Trait_1" $ CHROM : NULL $ POS : NULL $ input : chr "onemap_example_bc.raw" - attr(*, "class")= chr [1:2] "onemap" "backcross"

Details
A total of 150 individuals were genotyped for 67 markers with 15% of missing data. There is one quantitative phenotype to show how to use onemap output as R\qt1 input.
Author(s)

Marcelo Mollinari, <mmollina@usp.br>

See Also

read_onemap and read_mapmaker.

Examples

data(onemap_example_bc)

# perform two-point analyses
twopts <- rf_2pts(onemap_example_bc)
twopts

Description

Simulated data from a F2 population.

Usage

 data("onemap_example_f2")

Format

The format is: List of 10 $ geno : num [1:200, 1:66] 1 3 2 2 1 0 3 1 1 3 ... ..- attr(*, "dimnames")=List of 2 ...$ : chr [1:200] "IND1" "IND2" "IND3" "IND4" ... ..$ : chr [1:66] "M1" "M2" "M3" "M4" ... $ n.ind : int 200 $ n.mar : int 66 $ segr.type : chr [1:66] "A.H.B" "C.A" "D.B" "C.A" ... $ segr.type.num: num [1:66] 1 3 2 2 1 0 3 1 1 3 ... $ n.phe : int 1 $ pheno : num [1:200, 1] 37.6 36.4 37.2 35.8 37.1 ... ..- attr(*, "dimnames")=List of 2 ...$ : chr "Trait_1" $ CHROM : NULL $ POS : NULL $ input : chr "/home/cristiane/R/x86_64-pc-linux-gnu-library/3.4/onemap/extdata/onemap_example_f2.raw" - attr(*, "class")= chr [1:2] "onemap" "f2"

Details

A total of 200 individuals were genotyped for 66 markers (36 co-dominant, i.e. a, ab or b and 30 dominant i.e. c or a and d or b) with 15% of missing data. There is one quantitative phenotype to show how to use onemap output as R\qtl and QTL Cartographer input. Also, it is used for the analysis in the tutorial that comes with OneMap.

Examples

data(onemap_example_f2)
plot(onemap_example_f2)
Description

Simulated data set for an outcross, i.e., an F1 population obtained by crossing two non-homozygous parents.

Usage

data(onemap_example_out)

Format

An object of class onemap.

Details

A total of 100 F1 individuals were genotyped for 30 markers. The data currently contains only genotype information (no phenotypes). It is included to be used as a reference in order to understand how a data file needs to be. Also, it is used for the analysis in the tutorial that comes with OneMap.

Author(s)

Gabriel R A Margarido, <gramarga@gmail.com>

See Also

read_onemap for details about objects of class onemap.

Examples

data(onemap_example_out)

# perform two-point analyses
twopts <- rf_2pts(onemap_example_out)
twopts
Simulated data from a RIL population produced by selfing.

Description

Simulated biallelic data set for an ri self population.

Usage

data("onemap_example_riself")

Format

The format is: List of 10 $ geno : num [1:100, 1:68] 3 1 3 1 1 1 1 1 1 1 ... - attr(*, "dimnames")=List of 2 .. ..$ : chr [1:100] "ID1" "ID2" "ID3" "ID4" ... .. ..$ : chr [1:68] "M1" "M2" "M3" "M4" ... $ n.ind : int 100 $ n.mar : int 68 $ segr.type : chr [1:68] "A.B" "A.B" "A.B" "A.B" ... $ segr.type.num: logi [1:68] NA NA NA NA NA NA ... $ n.phe : int 0 $ pheno : NULL $ CHROM : NULL $ POS : NULL $ input : chr "onemap_example_riself.raw" - attr(*, "class")= chr [1:2] "onemap" "riself"

Details

A total of 100 F1 individuals were genotyped for 68 markers. The data currently contains only genotype information (no phenotypes). It is included to be used as a reference in order to understand how a data file needs to be.

Author(s)

Cristiane Taniguti, <chtaniguti@usp.br>

See Also

read_onemap for details about objects of class onemap.

Examples

data(onemap_example_riself)
plot(onemap_example_riself)
onemap_read_vcfR  
Convert vcfR object to onemap object

Description

Converts data from a vcfR package to onemap initial object, while trying to identify the appropriate marker segregation patterns.

Usage

```r
onemap_read_vcfR(
  vcfR.object = NULL,
  cross = c("outcross", "f2 intercross", "f2 backcross", "ri self", "ri sib"),
  parent1 = NULL,
  parent2 = NULL
)
```

Arguments

- **vcfR.object**: object of class 'vcfR' from vcfR package.
- **cross**: type of cross. Must be one of: "outcross" for full-sibs; "f2 intercross" for an F2 intercross progeny; "f2 backcross"; "ri self" for recombinant inbred lines by self-mating; or "ri sib" for recombinant inbred lines by sib-mating.
- **parent1**: string specifying sample ID of the first parent.
- **parent2**: string specifying sample ID of the second parent.

Details

Only biallelic SNPs and indels for diploid variant sites are considered.

Genotype information on the parents is required for all cross types. For full-sib progenies, both outbred parents must be genotyped. For backcrosses, F2 intercrosses and recombinant inbred lines, the original inbred lines must be genotyped. Particularly for backcross progenies, the recurrent line must be provided as the first parent in the function arguments.

Marker type is determined based on parental genotypes. Variants for which parent genotypes cannot be determined are discarded.

Reference sequence ID and position for each variant site are also stored.

Author(s)

Cristiane Taniguti, <chtaniguti@usp.br>

See Also

read_onemap for a description of the output object of class onemap.
order_seq

Examples

```r
## Not run:
vcfR.object <- vcfR::read.vcfR(system.file("extdata/vcf_example_out.vcf", package = "onemap"))
data <- onemap_read_vcfR(vcfR.object=vcfR.object,
cross="outcross",
parent1=c("P1"),
parent2=c("P2"))

## End(Not run)
```

---

order_seq

Search for the best order of markers combining compare and try_seq functions

Description

For a given sequence of markers, this function first uses the compare function to create a framework for a subset of informative markers. Then, it tries to map remaining ones using the try_seq function.

Usage

```r
order_seq(
  input.seq, n.init = 5,
  subset.search = c("twopt", "sample"),
  subset.n.try = 30,
  subset.THRES = 3,
  twopt.alg = c("rec", "rcd", "ser", "ug"),
  THRES = 3,
  touchdown = FALSE,
  tol = 0.1
)
```

Arguments

- `input.seq`: an object of class `sequence`.
- `n.init`: the number of markers to be used in the compare step (defaults to 5).
- `subset.search`: a character string indicating which method should be used to search for a subset of informative markers for the `compare` step. It is used for backcross, $F_2$ or RIL populations, but not for outcrosses. See the Details section.
- `subset.n.try`: integer. The number of times to repeat the subset search procedure. It is only used if `subset.search`="sample". See the Details section.
- `subset.THRES`: numerical. The threshold for the subset search procedure. It is only used if `subset.search`="sample". See the Details section.
twopt.alg  a character string indicating which two-point algorithm should be used if subset.search="twopt". See the Details section.

THRES  threshold to be used when positioning markers in the try_seq step.

touchdown  logical. If FALSE (default), the try_seq step is run only once, with the value of THRES. If TRUE, try_seq runs with THRES and then once more, with THRES-1. The latter calculations take longer, but usually are able to map more markers.

tol  tolerance number for the C routine, i.e., the value used to evaluate convergence of the EM algorithm.

Details

For outcrossing populations, the initial subset and the order in which remaining markers will be used in the try_seq step is given by the degree of informativeness of markers (i.e markers of type A, B, C and D, in this order).

For backcrosses, F2s or RILs, two methods can be used for choosing the initial subset: i) "sample" randomly chooses a number of markers, indicated by n.init, and calculates the multipoint log-likelihood of the \( \frac{n!}{2} \) possible orders. If the LOD Score of the second best order is greater than subset.THRES, than it takes the best order to proceed with the try_seq step. If not, the procedure is repeated. The maximum number of times to repeat this procedure is given by the subset.n.try argument. ii) "twopt" uses a two-point based algorithm, given by the option "twopt.alg", to construct a two-point based map. The options are "rec" for RECORD algorithm, "rcd" for Rapid Chain Delineation, "ser" for Seriation and "ug" for Unidirectional Growth. Then, equally spaced markers are taken from this map. The "compare" step will then be applied on this subset of markers.

In both cases, the order in which the other markers will be used in the try_seq step is given by marker types (i.e. co-dominant before dominant) and by the missing information on each marker.

After running the compare and try_seq steps, which result in a "safe" order, markers that could not be mapped are "forced" into the map, resulting in a map with all markers positioned.

Value

An object of class order, which is a list containing the following components:

ord  an object of class sequence containing the "safe" order.

mrk.unpos  a vector with unpositioned markers (if they exist).

LOD.unpos  a matrix with LOD-Scores for unmapped markers, if any, for each position in the "safe" order.

THRES  the same as the input value, just for printing.

ord.all  an object of class sequence containing the "forced" order, i.e., the best order with all markers.

data.name  name of the object of class onemap with the raw data.

twopt  name of the object of class rf_2pts with the 2-point analyses.

Author(s)

Gabriel R A Margarido, <gramarga@usp.br> and Marcelo Mollinari, <mmollina@gmail.com>
References


See Also

`make_seq`, `compare` and `try_seq`.

Examples

```r
## Not run:
# outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG2 <- make_seq(groups,2)
LG2.ord <- order_seq(LG2,touchdown=TRUE)
LG2.ord
make_seq(LG2.ord) # get safe sequence
make_seq(LG2.ord,"force") # get forced sequence

# F2 example
data(onemap_example_f2)
twopt <- rf_2pts(onemap_example_f2)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG3 <- make_seq(groups,3)
LG3.ord <- order_seq(LG3, subset.search = "twopt", twopt.alg = "rcd", touchdown=TRUE)
LG3.ord
make_seq(LG3.ord) # get safe sequence
ord.1<-make_seq(LG3.ord,"force") # get forced sequence

LG3.ord.s <- order_seq(LG3, subset.search = "sample", touchdown=TRUE)
```

LG3.ord.s
make_seq(LG3.ord) # get safe sequence
ord.2<-make_seq(LG3.ord,"force") # get forced sequence
rbind(ord.1$seq.num, ord.2$seq.num) # probably, the same order for this dataset

## End(Not run)

---

### plot.onemap

**Draw a graphic of raw data for any OneMap population**

**Description**

Shows a heatmap (in ggplot2, a graphic of geom "tile") for raw data. Lines correspond to markers and columns to individuals. The function can plot a graph for all marker types, depending on the cross type (dominant/codominant markers, in all combinations). The function receives a onemap object of class onemap, reads information from genotypes from this object, converts it to a long dataframe format using function melt() from package reshape2() or internal function create_dataframe_for_plot_outcross(), converts numbers from the object to genetic notation (according to the cross type), then plots the graphic. If there is more than 20 markers, removes y labels. For outcross populations, it can show all markers together, or it can split them according to the segregation pattern.

**Usage**

```r
## S3 method for class 'onemap'
plot(x, all = TRUE, ...)
```

**Arguments**

- **x**: an object of class onemap, with data and additional information
- **all**: a TRUE/FALSE option to indicate if results will be plotted together (if TRUE) or splitted based on their segregation pattern. Only used for outcross populations.
- **...**: currently ignored

**Value**

a ggplot graphic

**Examples**

```r
## Not run:
data(onemap_example_bc) # Loads a fake backcross dataset installed with onemap
plot(onemap_example_bc) # This will show you the graph

# You can store the graphic in an object, then save it with a number of properties
```
plot.onemap_segreg_test

Plot p-values for chi-square tests of expected segregation

Description

Draw a graphic showing the p-values (re-scaled to \(-\log_{10}(p-values)\)) associated with the chi-square tests for the expected segregation patterns for all markers in a dataset. It includes a vertical line showing the threshold for declaring statistical significance if Bonferroni’s correction is considered, as well as the percentage of markers that will be discarded if this criterion is used.

Usage

## S3 method for class 'onemap_segreg_test'
plot(x, order = TRUE, ...)

Arguments

x              an object of class onemap_segreg_test (produced by onemap’s function test_segregation()), i. e., after performing segregation tests
order          a variable to define if p-values will be ordered in the plot
...            currently ignored
plot_by_segreg_type

Value

a ggplot graphic

Examples

data(onemap_example_bc) # load OneMap's fake dataset for a backcross population
BC.seg <- test_segregation(onemap_example_bc) # Applies chi-square tests
print(BC.seg) # Shows the results
plot(BC.seg) # Plot the graph, ordering the p-values
plot(BC.seg, order=FALSE) # Plot the graph showing the results keeping the order in the dataset
# You can store the graphic in an object, then save it.
# For details, see the help of ggplot2's function ggsave()
# g <- plot(BC.seg)
# ggplot2::ggsave("SegregationTests.jpg", g, width=7, height=5, dpi=600)

data(onemap_example_out) # load OneMap's fake dataset for an outcrossing population
Out.seg <- test_segregation(onemap_example_out) # Applies chi-square tests
print(Out.seg) # Shows the results
plot(Out.seg) # Plot the graph, ordering the p-values
plot(Out.seg, order=FALSE) # Plot the graph showing the results keeping the order in the dataset
# You can store the graphic in an object, then save it.
# For details, see the help of ggplot2's function ggsave()
g <- plot(Out.seg)
ggplot2::ggsave("SegregationTests.jpg", g, width=7, height=5, dpi=600)

plot_by_segreg_type

Draw a graphic showing the number of markers of each segregation pattern.

Description

The function receives an object of class onemap. For outcrossing populations, it can show detailed information (all 18 possible categories), or a simplified version.

Usage

plot_by_segreg_type(x, subcateg = TRUE)

Arguments

x an object of class onemap
subcateg a TRUE/FALSE option to indicate if results will be plotted showing all possible categories (only for outcrossing populations)

Value

a ggplot graphic
Examples

```r
data(onemap_example_out) # Outcrossing data
plot_by_segreg_type(onemap_example_out)
plot_by_segreg_type(onemap_example_out, subcateg=FALSE)

data(onemap_example_bc)
plot_by_segreg_type(onemap_example_bc)

data(mapmaker_example_f2)
plot_by_segreg_type(mapmaker_example_f2)

# You can store the graphic in an object, then save it.
# For details, see the help of ggplot2's function ggsave()
# data(onemap_example_out) # Outcrossing data
# g <- plot_by_segreg_type(onemap_example_out)
# ggplot2::ggsave("SegregationTypes.jpg", g, width=7, height=4, dpi=600)
```

Description

Show the results of segregation tests

It shows the results of Chisquare tests performed for all markers in a onemap object of cross type outcross, backcross, F2 intercross or recombinant inbred lines.

Usage

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

Arguments

- `x` an object of class onemap_segreg_test
- `...` currently ignored

Value

a dataframe with marker name, H0 hypothesis, chi-square statistics, p-values, and

Examples

```r
data(onemap_example_out) # Loads a fake outcross dataset installed with onemap
Chi <- test_segregation(onemap_example_out) # Performs the chi-square test for all markers
print(Chi) # Shows the results
```
Description

Implements the marker ordering algorithm *Rapid Chain Delineation* (Doerge, 1996).

Usage

```r
cd(input.seq, LOD = 0, max.rf = 0.5, tol = 1e-04)
```

Arguments

- `input.seq`: an object of class `sequence`.
- `LOD`: minimum LOD-Score threshold used when constructing the pairwise recombination fraction matrix.
- `max.rf`: maximum recombination fraction threshold used as the LOD value above.
- `tol`: tolerance for the C routine, i.e., the value used to evaluate convergence.

Details

*Rapid Chain Delineation* (RCD) is an algorithm for marker ordering in linkage groups. It is not an exhaustive search method and, therefore, is not computationally intensive. However, it does not guarantee that the best order is always found. The only requirement is a matrix with recombination fractions between markers. Next is an excerpt from QTL Cartographer Version 1.17 Manual describing the RCD algorithm (Basten et al., 2005):

*The linkage group is initiated with the pair of markers having the smallest recombination fraction. The remaining markers are placed in a “pool” awaiting placement on the map. The linkage group is extended by adding markers from the pool of unlinked markers. Each terminal marker of the linkage group is a candidate for extension of the chain: The unlinked marker that has the smallest recombination fraction with either is added to the chain subject to the provision that the recombination fraction is statistically significant at a prespecified level. This process is repeated as long as markers can be added to the chain.*

After determining the order with RCD, the final map is constructed using the multipoint approach (function `map`).

Value

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

- `seq.num`: a vector containing the (ordered) indices of markers in the sequence, according to the input file.
- `seq.phases`: a vector with the linkage phases between markers in the sequence, in corresponding positions. −1 means that there are no defined linkage phases.
- `seq.rf`: a vector with the recombination frequencies between markers in the sequence. −1 means that there are no estimated recombination frequencies.
seq.like log-likelihood of the corresponding linkage map.
data.name name of the object of class onemap with the raw data.
twopt name of the object of class rf_2pts with the 2-point analyses.

Author(s)

Gabriel R A Margarido, <gramarga@gmail.com>

References


See Also

make_seq, map

Examples

## Not run:
#outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.rcd <- rcd(LG1)

#F2 example
data(onemap_example_f2)
twopt <- rf_2pts(onemap_example_f2)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.rcd <- rcd(LG1)
LG1.rcd

## End(Not run)
read_mapmaker

Read data from a Mapmaker raw file

Description
Imports data from a Mapmaker raw file.

Usage
read_mapmaker(dir, file)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dir</td>
<td>directory where the input file is located.</td>
</tr>
<tr>
<td>file</td>
<td>the name of the input file which contains the data to be read.</td>
</tr>
</tbody>
</table>

Details
For details about MAPMAKER files see Lincoln et al. (1993). The current version supports backcross, F2s and RIL populations. The file can contain phenotypic data, but it will not be used in the analysis.

Value
An object of class onemap, i.e., a list with the following components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>geno</td>
<td>a matrix with integers indicating the genotypes read for each marker in onemap fashion. Each column contains data for a marker and each row represents an individual.</td>
</tr>
<tr>
<td>geno.mmk</td>
<td>a matrix with integers indicating the genotypes read for each marker in MAPMAKER/EXP fashion, i.e., 1, 2, 3: AA, AB, BB, respectively; 3, 4: BB, not BB, respectively; 1, 5: AA, not AA, respectively. Each column contains data for a marker and each row represents an individual.</td>
</tr>
<tr>
<td>n.ind</td>
<td>number of individuals.</td>
</tr>
<tr>
<td>n.mar</td>
<td>number of markers.</td>
</tr>
<tr>
<td>segr.type</td>
<td>a vector with the segregation type of each marker, as strings. Segregation types were adapted from outcross segregation types, using the same notation. For details see read_onemap.</td>
</tr>
<tr>
<td>segr.type.num</td>
<td>a vector with the segregation type of each marker, represented in a simplified manner as integers. Segregation types were adapted from outcross segregation types. For details see read_onemap.</td>
</tr>
<tr>
<td>input</td>
<td>the name of the input file.</td>
</tr>
<tr>
<td>n.phe</td>
<td>number of phenotypes.</td>
</tr>
<tr>
<td>pheno</td>
<td>a matrix with phenotypic values. Each column contains data for a trait and each row represents an individual. Currently ignored.</td>
</tr>
</tbody>
</table>
**read_onemap**

**Read data from all types of progenies supported by OneMap**

**Description**

Imports data derived from outbred parents (full-sib family) or inbred parents (backcross, F2 intercross and recombinant inbred lines obtained by self- or sib-mating). Creates an object of class onemap.

**Usage**

```r
read_onemap(dir, inputfile)
```

**Arguments**

- `dir` directory where the input file is located.
- `inputfile` the name of the input file which contains the data to be read.

---

**Author(s)**

Adapted from Karl Broman (package *qtl*) by Marcelo Mollinari, <mmollina@usp.br>

**References**


**See Also**

`mapmaker_example_bc` and `mapmaker_example_f2` directory in the package source.

**Examples**

```r
## Not run:
map_data <- read_mapmaker(dir="work_directory",file="data_file.txt")
# Checking 'mapmaker_example_f2'
data(mapmaker_example_f2)
names(mapmaker_example_f2)

## End(Not run)
```
Details

The file format is similar to that used by MAPMAKER/EXP (Lincoln et al., 1993). The first line indicates the cross type and is structured as data type {cross}, where cross must be one of "outcross", "f2 intercross", "f2 backcross", "ri self" or "ri sib". The second line contains five integers: i) the number of individuals; ii) the number of markers; iii) an indicator variable taking the value 1 if there is CHROM information, i.e., if markers are anchored on any reference sequence, and 0 otherwise; iv) a similar 1/0 variable indicating whether there is POS information for markers; and v) the number of phenotypic traits.

The next line contains sample IDs, separated by empty spaces or tabs. Addition of this sample ID requirement makes it possible for separate input datasets to be merged.

Next comes the genotype data for all markers. Each new marker is initiated with a "*" (without the quotes) followed by the marker name, without any space between them. Each marker name is followed by the corresponding segregation type, which may be: "A.1", "A.2", "A.3", "A.4", "B1.5", "B2.6", "B3.7", "C.8", "D1.9", "D1.10", "D1.11", "D1.12", "D1.13", "D2.14", "D2.15", "D2.16", "D2.17" or "D2.18" (without quotes), for full-sibs [see marker_type and Wu et al. (2002) for details]. Other cross types have special marker types: "A.H" for backcrosses; "A.H.B" for F2 intercrosses; and "A.B" for recombinant inbred lines.

After the segregation type comes the genotype data for the corresponding marker. Depending on the segregation type, genotypes may be denoted by ac, ad, bc, bd, a, ba, b, bc, ab and o, in several possible combinations. To make things easier, we have followed exactly the notation used by Wu et al. (2002). Allowed values for backcrosses are a and ab; for F2 crosses they are a, ab and b; for RILs they may be a and b. Genotypes must be separated by a space. Missing values are denoted by "-".

If there is physical information for markers, i.e., if they are anchored at specific positions in reference sequences (usually chromosomes), this is included immediately after the marker data. These lines start with special keywords *CHROM and *POS and contain strings and integers, respectively, indicating the reference sequence and position for each marker. These also need to be separated by spaces.

Finally, if there is phenotypic data, it will be added just after the marker or CHROM/POS data. They need to be separated by spaces as well, using the same symbol for missing information.

The example directory in the package distribution contains an example data file to be read with this function. Further instructions can be found at the tutorial distributed along with this package.

Value

An object of class onemap, i.e., a list with the following components:

- **geno**: a matrix with integers indicating the genotypes read for each marker. Each column contains data for a marker and each row represents an individual.
- **n.ind**: number of individuals.
- **n.mar**: number of markers.
- **segr.type**: a vector with the segregation type of each marker, as strings.
- **segr.type.num**: a vector with the segregation type of each marker, represented in a simplified manner as integers, i.e., 1 corresponds to markers of type "A"; 2 corresponds to markers of type "B1.5"; 3 corresponds to markers of type "B2.6"; 4 corresponds to markers of type "B3.7"; 5 corresponds to markers of type "C.8"; 6
corresponds to markers of type "D1" and 7 corresponds to markers of type "D2". Markers for F2 intercrosses are coded as 1; all other crosses are left as NA.

input
the name of the input file.

n.phe
number of phenotypes.

pheno
a matrix with phenotypic values. Each column contains data for a trait and each row represents an individual.

Author(s)
Gabriel R A Margarido, <gramarga@gmail.com>

References


See Also
combine_onemap and the example directory in the package source.

Examples

```r
## Not run:
outcr_data <- read_onemap(dir="work_directory", inputfile="data_file.txt")

## End(Not run)
```

Description

Implements the marker ordering algorithm *Recombination Counting and Ordering* (Van Os et al., 2005).

Usage

```r
record(input.seq, times = 10, LOD = 0, max.rf = 0.5, tol = 1e-04)
```
Arguments

- **input.seq**: an object of class sequence.
- **times**: integer. Number of replicates of the RECORD procedure.
- **LOD**: minimum LOD-Score threshold used when constructing the pairwise recombination fraction matrix.
- **max.rf**: maximum recombination fraction threshold used as the LOD value above.
- **tol**: tolerance for the C routine, i.e., the value used to evaluate convergence.

Details

*Recombination Counting and Ordering (RECORD)* is an algorithm for marker ordering in linkage groups. It is not an exhaustive search method and, therefore, is not computationally intensive. However, it does not guarantee that the best order is always found. The only requirement is a matrix with recombination fractions between markers.

After determining the order with RECORD, the final map is constructed using the multipoint approach (function `map`).

Value

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

- **seq.num**: a vector containing the (ordered) indices of markers in the sequence, according to the input file.
- **seq.phases**: a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.
- **seq.rf**: a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies.
- **seq.like**: log-likelihood of the corresponding linkage map.
- **data.name**: name of the object of class `onemap` with the raw data.
- **twopt**: name of the object of class `rf_2pts` with the 2-point analyses.

Author(s)

Marcelo Mollinari, <mmollina@usp.br>

References


See Also

`make_seq` and `map`
## Examples

```r
## Not run:
## outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.rec <- record(LG1)

## F2 example
data(onemap_example_f2)
twopt <- rf_2pts(onemap_example_f2)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.rec <- record(LG1)
LG1.rec

## End(Not run)
```

---

**rf_2pts**  
*Two-point analysis between genetic markers*

### Description

Performs the two-point (pairwise) analysis proposed by Wu *et al.* (2002) between all pairs of markers.

### Usage

```
rf_2pts(input.obj, LOD = 3, max.rf = 0.5, verbose = TRUE)
```

### Arguments

- **input.obj**: an object of class `onemap`.
- **LOD**: minimum LOD Score to declare linkage (defaults to 3).
- **max.rf**: maximum recombination fraction to declare linkage (defaults to 0.50).
- **verbose**: logical. If TRUE, current progress is shown; if FALSE, no output is produced.

### Details

For `n` markers, there are \( \frac{n(n - 1)}{2} \) pairs of markers to be analyzed. Therefore, completion of the two-point analyses can take a long time.
Value

An object of class rf_2pts, which is a list containing the following components:

- **n.mar**: total number of markers.
- **LOD**: minimum LOD Score to declare linkage.
- **max.rf**: maximum recombination fraction to declare linkage.
- **input**: the name of the input file.
- **analysis**: an array with the complete results of the two-point analysis for each pair of markers.

Note

The thresholds used for LOD and max. rf will be used in subsequent analyses, but can be overridden.

Author(s)

Gabriel R A Margarido <gramarga@gmail.com> and Marcelo Mollinari <mmollina@usp.br>

References


Examples

```r
data(onemap_example_out)
twopts <- rf_2pts(onemap_example_out, LOD=3, max.rf=0.5) # perform two-point analyses
twopts
print(twopts, c("M1", "M2")) # detailed results for markers 1 and 2
```

Description

Plots pairwise recombination fractions and LOD Scores in a heatmap.

Plots a matrix of pairwise recombination fraction or LOD Scores using a color scale. Any value of the matrix can be easily accessed using an interactive plotly-html interface, helping users to check for possible problems.
**Usage**

```r
def rf_graph_table(
    input.seq,
    graph.LOD = FALSE,
    main = NULL,
    inter = FALSE,
    html.file = NULL,
    mrk.axis = "numbers",
    lab.xy = NULL,
    n.colors = 4
)
```

**Arguments**

- `input.seq`: an object of class `sequence` with a predefined order.
- `graph.LOD`: logical. If TRUE, displays the LOD heatmap, otherwise, displays the recombination fraction heatmap.
- `main`: character. The title of the plot.
- `inter`: logical. If TRUE, an interactive HTML graphic is plotted. Otherwise, a default graphic device is used.
- `html.file`: character naming the html file with iterative graphic.
- `mrk.axis`: character, "names" to display marker names in the axis, "numbers" to display marker numbers and "none" to display axis free of labels.
- `lab.xy`: character vector with length 2, first component is the label of x axis and second of the y axis.
- `n.colors`: integer. Number of colors in the palette.

**Details**

The color scale varies from red (small distances or big LODs) to purple. When hover on a cell, a dialog box is displayed with some information about corresponding markers for that cell (line (y) × column (x)). They are: i) the name of the markers; ii) the number of the markers on the data set; iii) the segregation types; iv) the recombination fraction between the markers and v) the LOD-Score for each possible linkage phase calculated via two-point analysis. For neighbor markers, the multipoint recombination fraction is printed; otherwise, the two-point recombination fraction is printed. For markers of type `D1` and `D2`, it is impossible to calculate recombination fraction via two-point analysis and, therefore, the corresponding cell will be empty (white color). For cells on the diagonal of the matrix, the name, the number and the type of the marker are printed, as well as the percentage of missing data for that marker.

**Author(s)**

Rodrigo Amadeu, <rramadeu@gmail.com>
Examples

```r
## Not run:
## outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.rcd <- rcd(LG1)
rf_graph_table(LG1.rcd, inter=FALSE)

## Now, using interactive plotly
rf_graph_table(LG1.rcd, inter=TRUE, html.file= "LG1.rcd.html")

## F2 example
data(onemap_example_f2)
twopt <- rf_2pts(onemap_example_f2)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)

#"pre-allocate" an empty list of length groups$n.groups (3, in this case)
maps.list<-vector("list", groups$n.groups)

for(i in 1:groups$n.groups){
  #create linkage group i
  LG.cur <- make_seq(groups,i)
  #ordering
  map.cur<-order_seq(LG.cur, subset.search = "sample")
  #assign the map of the i-th group to the maps.list
  maps.list[[i]]<-make_seq(map.cur, "force")
}

##Plot LOD/recombination fraction matrices for each group
require(gridExtra)
plot1 <- rf_graph_table(maps.list[[1]], main="Group 1",inter=FALSE)
plot2 <- rf_graph_table(maps.list[[2]], main="Group 2",inter=FALSE)
plot3 <- rf_graph_table(maps.list[[3]], main="Group 3",inter=FALSE)
grid.arrange(plot1, plot2, plot3, nrow=3)

## End(Not run)
```

### ripple_seq

Compares and displays plausible alternative orders for a given linkage group.

**Description**

For a given sequence of ordered markers, computes the multipoint likelihood of alternative orders, by shuffling subsets (windows) of markers within the sequence. For each position of the window, all possible \((ws)!\) orders are compared.
Usage

ripple_seq(input.seq, ws = 4, ext.w = NULL, LOD = 3, tol = 0.1)

Arguments

input.seq  
an object of class sequence with a predefined order.

ws  
an integer specifying the length of the window size (defaults to 4).

ext.w  
an integer specifying how many markers should be considered in the vicinity of the permuted window. If ext.w=NULL all markers in the sequence are considered. In this version, it is used only in backcross, F_2 or RIL crosses.

LOD  
threshold for the LOD-Score, so that alternative orders with LOD less then or equal to this threshold will be displayed.

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

Details

Large values for the window size make computations very slow, specially if there are many partially informative markers.

Value

This function does not return any value; it just produces text output to suggest alternative orders.

Author(s)

Gabriel R A Margarido, <gramarga@gmail.com> and Marcelo Mollinari, <mmollina@usp.br>

References


See Also

make_seq, compare, try_seq and order_seq.
Examples

```r
## Not run:
#Outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
markers <- make_seq(twopt, c(27, 16, 20, 4, 19, 21, 23, 9, 24, 29))
markers.map <- map(markers)
ripple_seq(markers.map)

#F2 example
data(onemap_example_f2)
twopt <- rf_2pts(onemap_example_f2)
all_mark <- make_seq(twopt, "all")
groups <- group(all_mark)
LG3 <- make_seq(groups, 3)
LG3.ord <- order_seq(LG3, subset.search = "twopt", twopt.alg = "rcd", touchdown = TRUE)
LG3.ord
make_seq(LG3.ord) # get safe sequence
ord.1<-make_seq(LG3.ord, "force") # get forced sequence
ripple_seq(ord.1, ws=5)

## End(Not run)
```

select_segreg  

Show markers with/without segregation distortion

Description

A function to shows which marker have segregation distortion if Bonferroni’s correction is applied for the Chi-square tests of mendelian segregation.

Usage

`select_segreg(x, distorted = FALSE, numbers = FALSE, threshold = NULL)`

Arguments

- `x` : an object of class `onemap_segreg_test`
- `distorted` : a TRUE/FALSE variable to show distorted or non-distorted markers
- `numbers` : a TRUE/FALSE variable to show the numbers or the names of the markers
- `threshold` : a number between 0 and 1 to specify the threshold to be considered in the test. If NULL, it uses the threshold defined by bonferroni correction with alpha = 0.05

Value

a vector with marker names or numbers, according to the option for "distorted" and "numbers"
Examples

# Loads a fake backcross dataset installed with onemap
data(onemap_example_bc)
# Performs the chi-square test for all markers
Chi <- test_segregation(onemap_example_bc)
# To show non-distorted markers
select_segreg(Chi)
# To show markers with segregation distortion
select_segreg(Chi, distorted=TRUE)
# To show the numbers of the markers with segregation distortion
select_segreg(Chi, distorted=TRUE, numbers=TRUE)

seriation  Seriation

Description

Implements the marker ordering algorithm Seriation (Buetow & Chakravarti, 1987).

Usage

seriation(input.seq, LOD = 0, max.rf = 0.5, tol = 1e-04)

Arguments

input.seq  an object of class sequence.
LOD  minimum LOD-Score threshold used when constructing the pairwise recombination fraction matrix.
max.rf  maximum recombination fraction threshold used as the LOD value above.
tol  tolerance for the C routine, i.e., the value used to evaluate convergence.

Details

Seriation is an algorithm for marker ordering in linkage groups. It is not an exhaustive search method and, therefore, is not computationally intensive. However, it does not guarantee that the best order is always found. The only requirement is a matrix with recombination fractions between markers.

NOTE: When there are too many pairs of markers with the same value in the recombination fraction matrix, it can result in ties during the ordination process and the Seriation algorithm may not work properly. This is particularly relevant for outcrossing populations with mixture of markers of type D1 and D2. When this occurs, the function shows the following error message: There are too many ties in the ordination process - please, consider using another ordering algorithm.

After determining the order with Seriation, the final map is constructed using the multipoint approach (function map).
Value

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

- `seq.num` a vector containing the (ordered) indices of markers in the sequence, according to the input file.
- `seq.phases` a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.
- `seq.rf` a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies.
- `seq.like` log-likelihood of the corresponding linkage map.
- `data.name` name of the object of class `onemap` with the raw data.
- `twopt` name of the object of class `rf_2pts` with the 2-point analyses.

Author(s)

Gabriel R A Margarido, <gramarga@gmail.com>

References


See Also

`make_seq`, `map`

Examples

```r
## Not run:
## outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt, "all")
groups <- group(all_mark)
LG3 <- make_seq(groups, 3)
LG3.ser <- seriation(LG3)

## F2 example
data(onemap_example_f2)
twopt <- rf_2pts(onemap_example_f2)
all_mark <- make_seq(twopt, "all")
groups <- group(all_mark)
LG1 <- make_seq(groups, 1)
LG1.ser <- seriation(LG1)
LG1.ser

## End(Not run)
```
**set_map_fun**

*Defines the default mapping function*

**Description**

Defines the function that should be used to display the genetic map through the analysis.

**Usage**

```r
set_map_fun(type = c("kosambi", "haldane"))
```

**Arguments**

- **type**
  Indicates the function that should be used, which can be "kosambi" or "haldane".

**Author(s)**

Marcelo Mollinari, <mmollina@usp.br>

**References**


**See Also**

- `kosambi` and `haldane`

---

**suggest_lod**

*Suggests a LOD Score for two point tests*

**Description**

It suggests a LOD Score for declaring statistical significance for two-point tests for linkage between all pairs of markers, considering that multiple tests are being performed.

**Usage**

```r
suggest_lod(x)
```

**Arguments**

- **x**
  an object of class `onemap`
Details

In a somehow naive approach, the function calculates the number of two-point tests that will be performed for all markers in the data set, and then using this to calculate the global alpha required to control type I error using Bonferroni’s correction.

From this global alpha, the corresponding quantile from the chi-square distribution is taken and then converted to LOD Score.

This can be seen as just an initial approximation to help users to select a LOD Score for two point tests.

Value

the suggested LOD to be used for testing linkage

Examples

data(onemap_example_bc) # Loads a fake backcross dataset installed with onemap
suggest_lod(onemap_example_bc) # An value that should be used to start the analysis

date

Description

Using OneMap internal function test_segregation_of_a_marker(), performs the Chi-square test to check if all markers in a dataset are following the expected segregation pattern, i. e., 1:1:1:1 (A), 1:2:1 (B), 3:1 (C) and 1:1 (D) according to OneMap’s notation.

Usage

test_segregation(x)

Arguments

x an object of class onemap, with data and additional information.

Details

First, it identifies the correct segregation pattern and corresponding H0 hypothesis, and then tests it.

Value

an object of class onemap_segreg_test, which is a list with marker name, H0 hypothesis being tested, the chi-square statistics, the associated p-values and the % of individuals genotyped. To see the object, it is necessary to print it.
Examples

data(onemap_example_out) # Loads a fake outcross dataset installed with onemap
Chi <- test_segregation(onemap_example_out) # Performs the chi-square test for all markers
print(Chi) # Shows the results

data(onemap_example_bc) # Loads a fake backcross dataset installed with onemap
test_segregation_of_a_marker(onemap_example_bc,1)
data(onemap_example_out) # Loads a fake outcross dataset installed with onemap
test_segregation_of_a_marker(onemap_example_out,1)

Description

Applies the chi-square test to check if markers are following the expected segregation pattern, i.e., 1:1:1:1 (A), 1:2:1 (B), 3:1 (C) and 1:1 (D) according to OneMap's notation. It does not use Yate’s correction.

Usage

test_segregation_of_a_marker(x, marker)

Arguments

x an object of class onemap, with data and additional information.
marker the marker which will be tested for its segregation.

Details

First, the function selects the correct segregation pattern, then it defines the H0 hypothesis, and then tests it, together with percentage of missing data.

Value

a list with the H0 hypothesis being tested, the chi-square statistics, the associated p-values, and the % of individuals genotyped.

@examples data(onemap_example_bc) # Loads a fake backcross dataset installed with onemap
test_segregation_of_a_marker(onemap_example_bc,1)
data(onemap_example_out) # Loads a fake outcross dataset installed with onemap
test_segregation_of_a_marker(onemap_example_out,1)
try_seq

Try to map a marker into every possible position between markers in a given map

Description

For a given linkage map, tries do add an additional unpositioned marker. This function estimates parameters for all possible maps including the new marker in all possible positions, while keeping the original linkage map unaltered.

Usage

try_seq(input.seq, mrk, tol = 0.1, pos = NULL, verbose = FALSE)

Arguments

input.seq an object of class sequence with a predefined order.

mrk the index of the marker to be tried, according to the input file.

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

pos defines in which position the new marker mrk should be placed for the diagnostic graphic. If NULL (default), the marker is placed on the best position i.e. the one which results LOD = 0.00

verbose if FALSE (default), simplified output is displayed. if TRUE, detailed output is displayed.

Value

An object of class try, which is a list containing the following components:

ord a list containing results for every linkage map estimated. These results include linkage phases, recombination frequencies and log-likelihoods.

LOD a vector with LOD-Scores for each position where the additional marker is placed. This Score is based on the best combination of linkage phases for each map.

try_ord a matrix with the orders of all linkage maps.

data.name name of the object of class onemap with the raw data.

twopt name of the object of class rf_2pts with the 2-point analyses.

Author(s)

Marcelo Mollinari, <mmollina@usp.br>
References


See Also

`make_seq` and `compare`.

Examples

```r
## Not run:
#outcrossing example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
markers <- make_seq(twopt,c(2,3,12,14))
markers.comp <- compare(markers)
base.map <- make_seq(markers.comp,1)

extend.map <- try_seq(base.map,30)
extend.map
print(extend.map,5) # best position
print(extend.map,4) # second best position

#F2 example
data(mapmaker_example_f2)
twopt <- rf_2pts(mapmaker_example_f2)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG3 <- make_seq(groups,3)
LG3.ord <- order_seq(LG3, subset.search = "twopt", twopt.alg = "rcd", touchdown=TRUE)
LG3.ord
safe.map<-make_seq(LG3.ord,"safe")
extend.map <- try_seq(safe.map,64)
extend.map
(new.map<-make_seq(extend.map,14)) # best position
```
Unidirectional Growth

Description

Implements the marker ordering algorithm *Unidirectional Growth* (Tan & Fu, 2006).

Usage

```r
ug(input.seq, LOD = 0, max.rf = 0.5, tol = 1e-04)
```

Arguments

- `input.seq`: an object of class `sequence`.
- `LOD`: minimum LOD-Score threshold used when constructing the pairwise recombination fraction matrix.
- `max.rf`: maximum recombination fraction threshold used as the LOD value above.
- `tol`: tolerance for the C routine, i.e., the value used to evaluate convergence.

Details

*Unidirectional Growth* (UG) is an algorithm for marker ordering in linkage groups. It is not an exhaustive search method and, therefore, is not computationally intensive. However, it does not guarantee that the best order is always found. The only requirement is a matrix with recombination fractions between markers.

After determining the order with UG, the final map is constructed using the multipoint approach (function `map`).

Value

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

- `seq.num`: a vector containing the (ordered) indices of markers in the sequence, according to the input file.
- `seq.phases`: a vector with the linkage phases between markers in the sequence, in corresponding positions. -1 means that there are no defined linkage phases.
- `seq.rf`: a vector with the recombination frequencies between markers in the sequence. -1 means that there are no estimated recombination frequencies.
- `seq.like`: log-likelihood of the corresponding linkage map.
- `data.name`: name of the object of class `onemap` with the raw data.
- `twopt`: name of the object of class `rf_2pts` with the 2-point analyses.
Author(s)

Marcelo Mollinari, <mmollina@usp.br>

References


See Also

`make_seq, map`

Examples

```r
## Not run:
# outcross example
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.ug <- ug(LG1)

# F2 example
data(mapmaker_example_f2)
twopt <- rf_2pts(mapmaker_example_f2)
all_mark <- make_seq(twopt,"all")
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.ug <- ug(LG1)
LG1.ug

## End(Not run)
```

vcf2raw

Convert variants from a VCF file to OneMap file format

Description

Converts data from a standard VCF (Variant Call Format) file to the input format required by OneMap, while trying to identify the appropriate marker segregation patterns.
Usage

```r
vcf2raw(
  input = NULL,
  output = NULL,
  cross = c("outcross", "f2 intercross", "f2 backcross", "ri self", "ri sib"),
  parent1 = NULL,
  parent2 = NULL,
  min_class = 1
)
```

Arguments

- **input**: path to the input VCF file.
- **output**: path to the output OneMap file.
- **cross**: type of cross. Must be one of: "outcross" for full-sibs; "f2 intercross" for an F2 intercross progeny; "f2 backcross"; "ri self" for recombinant inbred lines by self-mating; or "ri sib" for recombinant inbred lines by sib-mating.
- **parent1**: string or vector of strings specifying sample ID(s) of the first parent.
- **parent2**: string or vector of strings specifying sample ID(s) of the second parent.
- **min_class**: a real number between 0.0 and 1.0. For each parent and each variant site, defines the proportion of parent samples that must be of the same genotype for it to be assigned to the corresponding parent.

Details

The input VCF file must be sorted, compressed and tabix indexed. Please check functions `bgzip` and `indexTabix` of package `Rsamtools` for details.

Each variant in the VCF file is processed independently. Only biallelic SNPs and indels for diploid variant sites are considered.

Genotype information on the parents is required for all cross types. For full-sib progenies, both outbred parents must be genotyped. For backcrosses, F2 intercrosses and recombinant inbred lines, the original inbred lines must be genotyped. Particularly for backcross progenies, the recurrent line must be provided as the first parent in the function arguments.

First, samples corresponding to both parents of the progeny are parsed and their genotypes identified, given that their replicates are concordant above a threshold given by `min_class`. This allows replicates of the parents to be used, which is common in sequencing plates. In detail, each parent will be called an heterozygote only if `min_class*number of replicates` samples or more are heterozygous. The same is valid for homozygous calls. Whenever there are different genotypes among replicates, heterozygosity is checked first. The default value (1.0) requires that all replicates be of the same genotype. If each parent is represented by a single sample, this parameter has no effect.

Next, marker type is determined based on parental genotypes. Finally, progeny genotypes are identified and output is produced. Variants for which parent genotypes cannot be determined are discarded.

Reference sequence ID and position for each variant site are stored as special fields denoted CHROM and POS.
### Description

Simulated biallelic data set for an backcross population

### Usage

```r
data("vcf_example_bc")
```

### Format

An object of class `onemap`.

### Details

A total of 142 backcross individuals were genotyped with 25 markers. The data was generated from a VCF file. It contains chromosome and position informations for each marker. It is included to be used as an example to understand how to convert VCF file to OneMap input data with the functions `vcf2raw` and `onemap_read_vcfR`.

### Author(s)

Cristiane Hayumi Taniguti, <chaytaniguti@gmail.com>
See Also

`read_onemap` for details about objects of class `onemap`.

Examples

```r
data(vcf_example_bc)
plot(vcf_example_bc)
```

---

vcf_example_f2

*Data generated from VCF file with biallelic markers from a f2 intercross population*

Description

Simulated biallelic data set for an f2 population

Usage

`data(vcf_example_f2)`

Format

An object of class `onemap`.

Details

A total of 192 F2 individuals were genotyped with 25 markers. The data was generated from a VCF file. It contains chromosome and position informations for each marker. It is included to be used as a reference in order to understand how to convert VCF file to OneMap input data. Also, it is used for the analysis in the tutorial that comes with OneMap.

Author(s)

Cristiane Hayumi Taniguti, <chaytaniguti@gmail.com>

See Also

`read_onemap` for details about objects of class `onemap`.

Examples

```r
data(vcf_example_f2)

# plot markers informations
plot(vcf_example_f2)
```
Description

Simulated biallelic data set for an outcross, i.e., an F1 population obtained by crossing two non-homozygous parents.

Usage

```r
data(vcf_example_out)
```

Format

An object of class onemap.

Details

A total of 92 F1 individuals were genotyped with 27 markers. The data was generated from a VCF file. It contains chromosome and position informations for each marker. It is included to be used as a reference in order to understand how to convert VCF file to OneMap input data. Also, it is used for the analysis in the tutorial that comes with OneMap.

Author(s)

Cristiane Hayumi Taniguti, <chaytaniguti@gmail.com>

See Also

`read_onemap` for details about objects of class onemap.

Examples

```r
data(vcf_example_out)

# plot markers informations
plot(vcf_example_out)
```
Description

Simulated biallelic data set for an RIL population.

Usage

```r
data("vcf_example_riself")
```

Format

The format is: List of 10

```r
$ geno : num [1:92, 1:25] 3 3 1 3 3 1 3 3 1 3 1 ... attr(*, "dimnames")=List of 2

$ n.ind : int 92

$ n.mar : int 25

$ segr.type : chr [1:25] "A.B" "A.B" "A.B" "A.B" ...

$ segr.type.num: logi [1:25] NA NA NA NA NA NA ...

$ n.phe : int 0

$ pheno : NULL

$ CHROM : chr [1:25] "1" "1" "1" "1" "1" ...

$ POS : int [1:25] 1791 6606 9001 11326 11702 15533 17151 18637 19146 19220 ...

$ input : chr "vcf_example_riself.raw" - attr(*, "class")= chr [1:2] "onemap" "riself"
```

Details

A total of 92 rils individuals were genotyped with 25 markers. The data was generated from a VCF file. It contains chromosome and position informations for each marker. It is included to be used as an example in order to understand how to convert VCF file to OneMap input data with the functions `vcf2raw` and `onemap_read_vcfR`.

Author(s)

Cristiane Hayumi Taniguti, <chaytaniguti@gmail.com>

See Also

`read_onemap` for details about objects of class `onemap`.

Examples

```r
data(vcf_example_riself)
plot(vcf_example_riself)
```
Write a genetic map to a file

Description
Write a genetic map to a file, base on a given map, or a list of maps. The output file can be used as an input to perform QTL mapping using the package R/qtl. It is also possible to create an output to be used with QTLCartographer program.

Usage
write_map(map.list, file.out)

Arguments
map.list a map, i.e. an object of class sequence with a predefined order, linkage phases, recombination fraction and likelihood or a list of maps.
file.out output map file.

Details
This function is available only for backcross, F2 and RILs.

Author(s)
Marcelo Mollinari, <mmollina@usp.br>

References

Examples
```r
## Not run:
data(mapmaker_example_f2)
twopt<rf_2pts(mapmaker_example_f2)
lg<group(make_seq(twopt, "all"))

## "pre-allocate" an empty list of length lg$n.groups (3, in this case)
maps.list<-vector("list", lg$n.groups)
for(i in 1:lg$n.groups){
  #create linkage group i
  LG.cur <- make_seq(lg,i)
}
## ordering
map.cur <- order_seq(LG.cur, subset.search = "sample")
# assign the map of the i-th group to the maps.list
maps.list[[i]] <- make_seq(map.cur, "force")
}

## write maps.list to "mapmaker_example_f2.map" file
write_map(map.list, "mapmaker_example_f2.map")

## Using R/qtl
## you must install the package 'qtl'
## install.packages("qtl")

require(qtl)
file <- paste(system.file("example", package="onemap"), "mapmaker_example_f2.raw", sep="/"")
dat1 <- read.cross("mm", file=file, mapfile="mapmaker_example_f2.map")
newmap <- est.map(dat1, tol=1e-6, map.function="kosambi")

(logliks <- sapply(newmap, attr, "loglik"))
plot.map(dat1, newmap)

## Using R/qtl to generate QTL Cartographer input files (.map and .cro)
write.cross(dat1, format="qtlcart", filestem="mapmaker_example_f2")

## End(Not run)

---

write_onemap_raw

Convert onemap object to onemap raw file

Description

Converts onemap R object to onemap input file. The input file brings information about the mapping population: First line: cross type, it can be "outcrossing", "f2 intercross", "f2 backcross", "ri self" or "ri sib". Second line: number of individuals, number of markers, presence (1) or absence (0) of chromosome and position of the markers, and number of phenotypes measured. Third line: Individuals/sample names; Followed lines: marker name, marker type and genotypes. One line for each marker. Final lines: chromosome, position and phenotypes informations. See more about input file format at vignettes.

Usage

write_onemap_raw(
  onemap.obj = NULL,
  file.name = "out.raw",
  cross = c("outcross", "f2 backcross", "f2 intercross", "ri self", "ri sib")
)

Arguments

- `onemap.obj` object of class `onemap`
- `file.name` a character for the onemap raw file name. Default is "out.raw"
- `cross` a character describing the cross type. It can be "outcrossing", "f2 intercross", "f2 backcross", "ri self" or "ri sib"

Author(s)

Cristiane Taniguti, <chtaniguti@usp.br>

See Also

`read_onemap` for a description of the output object of class onemap.

Examples

```r
## Not run:
data(onemap_example_out)
write_onemap_raw(onemap_example_out, file.name = "onemap_example_out.raw", cross="outcross")
## End(Not run)
```
Index

*Topic IO
  combine_onemap, 5
  read_mapmaker, 42
  read_onemap, 43
  vcf2raw, 61

*Topic arith
  map_func, 24
  set_map_fun, 55

*Topic bins
  create_data_bins, 9
  find_bins, 14

*Topic datasets
  mapmaker_example_bc, 22
  mapmaker_example_f2, 23
  onemap_example_bc, 28
  onemap_example_f2, 29
  onemap_example_out, 30
  onemap_example_riself, 31
  vcf_example_bc, 63
  vcf_example_f2, 64
  vcf_example_out, 65
  vcf_example_riself, 66

*Topic dimension
  create_data_bins, 9
  find_bins, 14

*Topic manip
  marker_type, 25

*Topic misc
  group, 15

*Topic reduction
  create_data_bins, 9
  find_bins, 14

*Topic rqtl
  draw_map, 10
  draw_map2, 11
  drop_marker, 4, 13
  find_bins, 9, 14
  group, 15, 18
  group_seq, 16
  haldane, 55
  haldane (map_func), 24
  kosambi, 55
  kosambi (map_func), 24
  make_seq, 7, 16, 18, 18, 21, 26, 35, 41, 46, 51, 54, 59, 61
  map, 20, 20, 40, 41, 46, 53, 54, 60, 61
  map_func, 24
  mapmaker_example_bc, 22
  map, 20
  marker_type, 25
  order_seq, 33
  rcd, 40
  record, 45
  rf_2pts, 47
  rf_graph_table, 48
  ripple_seq, 50
  seriation, 53
  try_seq, 58
  ug, 60

add_marker, 3, 13
Bonferroni_alpha, 4
combine_onemap, 5, 45
compare, 6, 20, 33, 35, 51, 59
create_data_bins, 9, 14
create_dataframe_for_plot_outcross, 8
draw_map, 10
draw_map2, 11
drop_marker, 4, 13
find_bins, 9, 14

haldane, 55
haldane (map_func), 24
kosambi, 55
kosambi (map_func), 24
make_seq, 7, 16, 18, 18, 21, 26, 35, 41, 46, 51, 54, 59, 61
map, 20, 20, 40, 41, 46, 53, 54, 60, 61
map_func, 24
mapmaker_example_bc, 22
INDEX

mapmaker_example_f2, 23
marker_type, 7, 25, 44
mds_onemap, 26

onemap_example_bc, 28
onemap_example_f2, 29
onemap_example_out, 30
onemap_example_riself, 31
onemap_read_vcfR, 32
order_seq, 20, 33, 51

plot.onemap, 36
plot.onemap_segreg_test, 37
plot_by_segreg_type, 38
print.onemap_segreg_test, 39

rcd, 40
read_mapmaker, 6, 22, 29, 42
read_onemap, 6, 22, 29–31, 42, 43, 64–66
record, 45
rf_2pts, 16, 47
rf_graph_table, 48
ripple_seq, 50

select_segreg, 52
seriation, 53
set_map_fun, 55
suggest_lod, 55

test_segregation, 56
test_segregation_of_a_marker, 57
try_seq, 20, 35, 51, 58

ug, 60

vcf2raw, 61
vcf_example_bc, 63
vcf_example_f2, 64
vcf_example_out, 65
vcf_example_riself, 66

write_map, 67
write_onemap_raw, 68