Package ‘onemap’

May 16, 2025

Title Construction of Genetic Maps in Experimental Crosses

Version 3.2.2

Description Analysis of molecular marker data from model and non-model systems. For the later, it allows statistical analysis by simultaneously estimating linkage and linkage phases (genetic map construction) according to Wu and colleagues (2002) <doi:10.1006/tpbi.2002.1577>. All analysis are based on multi-point approaches using hidden Markov models.

Author Gabriel Margarido [aut],
Marcelo Mollinari [aut],
Cristiane Taniguti [ctb, cre],
Getulio Ferreira [ctb],
Rodrigo Amadeu [ctb],
Jeekin Lau [ctb],
Karl Broman [ctb],
Katharine Preedy [ctb, cph] (MDS ordering algorithm),
Bastian Schiffthaler [ctb, cph] (HMM parallelization),
Augusto Garcia [aut, ctb]

LinkingTo Rcpp (>= 1.0.0)

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, smacof, princurve, parallel, dplyr, tidyr, htmlwidgets, ggpubr, RColorBrewer, dendextend, rebus, vcfR (>= 1.6.0)

Suggests knitr (>= 1.10), rmarkdown, testthat, stringr

VignetteBuilder knitr

Encoding UTF-8

License GPL-3

URL `https://github.com/cristianetaniguti/onemap`

BugReports `https://github.com/Cristianetaniguti/onemap/issues`

Maintainer Cristiane Taniguti <`cht47@cornell.edu`>

Repository CRAN

NeedsCompilation yes

Date/Publication 2025-05-16 14:30:02 UTC

RoxygenNote 7.3.2

Contents

acum 4
add_marker 4
add_redundants 5
Bonferroni_alpha 6
check_data 7
check_twopts 8
combine_onemap 8
compare 10
create_dataframe_for_plot_outcross 12
create_data_bins 12
create_depths_profile 13
create_probs 15
draw_map 17
draw_map2 18
drop_marker 19
edit_order_onemap 20
empty_onemap_obj 21
export_mappoly_genoprob 22
export_viewpoly 22
extract_depth 23
filter_2pts_gaps 24
filter_missing 25
filter_prob 26
find_bins 27
generate_overlapping_batches 28
group 28
group_seq 30
group_upgma 32
haldane 33
keep_only_selected_mks 34
kosambi 34
load_onemap_sequences 35
make_seq 35
map 37
mapmaker_example_f2 39
map_avoid_unlinked 40
map_overlapping_batches 42
map_save_ram 43
marker_type 44
mds_onemap 46

onemap_example_bc . . . 48
onemap_example_f2 . . . 49
onemap_example_out . . . 50
onemap_example_riself . . . 51
onemap_read_vcfR . . . 52
order_seq . . . 54
ord_by_geno . . . 56
parents_haplotypes . . . 57
pick_batch_sizes . . . 58
plot.onemap . . . 59
plot.onemap_progeny_haplotypes . . . 60
plot.onemap_progeny_haplotypes_counts . . . 61
plot.onemap_segreg_test . . . 62
plot_by_segreg_type . . . 63
plot_genome_vs_cm . . . 64
print.compare . . . 64
print.onemap . . . 65
print.onemap_bin . . . 65
print.onemap_segreg_test . . . 66
print.order . . . 66
print.sequence . . . 67
progeny_haplotypes . . . 67
progeny_haplotypes_counts . . . 68
rcd . . . 69
read_mapmaker . . . 71
read_onemap . . . 73
record . . . 75
remove_inds . . . 77
rf_2pts . . . 78
rf_graph_table . . . 79
rf_snp_filter_onemap . . . 81
ripple_seq . . . 83
rm_dupli_mks . . . 84
save_onemap_sequences . . . 85
seeded_map . . . 86
select_segreg . . . 88
seq_by_type . . . 89
seriation . . . 90
set_map_fun . . . 92
simu_example_bc . . . 93
simu_example_f2 . . . 94
simu_example_out . . . 95
sort_by_pos . . . 96
split_2pts . . . 97
split_onemap . . . 97
suggest_lod . . . 98
summary_maps_onemap . . . 99
test_segregation . . . 99

test_segregation_of_a_marker 100
try_seq 101
try_seq_by_seq 103
ug 104
vcf2raw 106
vcf_example_bc 106
vcf_example_f2 107
vcf_example_out 108
vcf_example_riself 109
write_map 110
write_onemap_raw 111

Index **113**

`acum` *Perform gaussian sum*

Description

Perform gaussian sum

Usage

```
acum(w)
```

Arguments

`w`	vector of numbers

`add_marker` *Creates a new sequence by adding markers.*

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>`

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)))
```

`add_redundants` *Add the redundant markers removed by create_data_bins function*

Description

Add the redundant markers removed by create_data_bins function

Usage

```
add_redundants(sequence, onemap.obj, bins)
```

Arguments

`sequence`	object of class sequence
`onemap.obj`	object of class `onemap.obj` before redundant markers were removed
`bins`	object of class `onemap_bin`

Value

New sequence 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`	object of class `onemap` with the raw data.
`twopt`	object of class `rf_2pts` with the 2-point analyses.

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

See Also

`find_bins`

`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
`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
```

`check_data` *Onemap object sanity check*

Description

Based on MAPpoly check_data_sanity function by Marcelo Mollinari

Usage

```
check_data(x)
```

Arguments

`x`	an object of class `onemap`

Value

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

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

Examples

```
data(onemap_example_bc)
check_data(onemap_example_bc)
```

`check_twopts` *Twopts object sanity check*

Description

Based on MAPpoly check_data_sanity function by Marcelo Mollinari

Usage

```
check_twopts(x)
```

Arguments

`x`	an object of class `onemap`

Value

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

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

Examples

```
data(onemap_example_bc)
twopts <- rf_2pts(onemap_example_bc)
check_twopts(twopts)
```

`combine_onemap` *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.

Author(s)

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

References

Lincoln, S. E., Daly, M. J. and Lander, E. S. (1993) Constructing genetic linkage maps with MAPMAKER/EXP Version 3.0: a tutorial and reference manual. *A Whitehead Institute for Biomedical Research Technical Report*.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

See Also

`read_onemap` and `read_mapmaker`.

Examples

```
data("onemap_example_out")
data("vcf_example_out")
combined_data <- combine_onemap(onemap_example_out, vcf_example_out)
```

`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 $\frac{n!}{2}$ possible orders.

Usage

```
compare(input.seq, n.best = 50, tol = 0.001, verbose = 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.

Details

Since the number $\frac{n!}{2}$ 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 recommend 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

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Jiang, C. and Zeng, Z.-B. (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* 101: 47-58.

Lander, E. S., Green, P., Abrahamson, J., Barlow, A., Daly, M. J., Lincoln, S. E. and Newburg, L. (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* 1: 174-181.

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. _Heredity_ 103: 494-502.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002a) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. (2002b). Linkage mapping of sex-specific differences. *Genetical Research* 79: 85-96

See Also

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

Examples

```
#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))
```

`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

`create_data_bins` *New dataset based on 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`, 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.
`error`	matrix containing HMM emission probabilities

Author(s)

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

See Also

`find_bins`

Examples

```
data("onemap_example_f2")
(bins<-find_bins(onemap_example_f2, exact=FALSE))
onemap_bins <- create_data_bins(onemap_example_f2, bins)
```

`create_depths_profile` *Create database and ggplot graphic of allele reads depths*

Description

Create database and ggplot graphic of allele reads depths

Usage

```
create_depths_profile(
  onemap.obj = NULL,
  vcfR.object = NULL,
  vcf = NULL,
  parent1 = NULL,
  parent2 = NULL,
  vcf.par = "AD",
  recovering = FALSE,
  mks = NULL,
  inds = NULL,
  GTfrom = "onemap",
  alpha = 1,
  rds.file = "data.rds",
  y_lim = NULL,
  x_lim = NULL,
  verbose = TRUE
)
```

Arguments

`onemap.obj`	an object of class `onemap`.
`vcfR.object`	object of class vcfR;
`vcf`	path to VCF file.
`parent1`	a character specifying the first parent ID
`parent2`	a character specifying the second parent ID
`vcf.par`	the vcf parameter that store the allele depth information.
`recovering`	logical. If TRUE, all markers in vcf are consider, if FALSE only those in onemap.obj
`mks`	a vector of characters specifying the markers names to be considered or NULL to consider all markers
`inds`	a vector of characters specifying the individual names to be considered or NULL to consider all individuals
`GTfrom`	the graphic should contain the genotypes from onemap.obj or from the vcf? Specify using "onemap", "vcf" or "prob".
`alpha`	define the transparency of the dots in the graphic
`rds.file`	rds file name to store the data frame with values used to build the graphic
`y_lim`	set scale limit for y axis
`x_lim`	set scale limit for x axis
`verbose`	If `TRUE`, print tracing information.

Value

an rds file and a ggplot graphic.

Author(s)

Cristiane Taniguti, <chtaniguti@tamu.edu>

See Also

`onemap_read_vcfR`

`create_probs` *Build genotype probabilities matrix for hmm*

Description

The genotypes probabilities can be calculated considering a global error (default method) or considering a genotype error probability for each genotype. Furthermore, user can provide directly the genotype probability matrix.

Usage

```
create_probs(
  input.obj = NULL,
  global_error = NULL,
  genotypes_errors = NULL,
  genotypes_probs = NULL
)
```

Arguments

`input.obj`	object of class onemap or onemap sequence
`global_error`	a integer specifying the global error value
`genotypes_errors`	a matrix with dimensions (number of individuals) x (number of markers) with genotypes errors values
`genotypes_probs`	a matrix with dimensions (number of individuals)*(number of markers) x possible genotypes (i.e., a ab ba b) with four columns for f2 and outcrossing populations, and two for backcross and RILs).

Details

The genotype probability matrix has number of individuals x number of markers rows and four columns (or two if considering backcross or RILs populations), one for each possible genotype of the population. This format follows the one proposed by MAPpoly.

The genotype probabilities come from SNP calling methods. If you do not have them, you can use a global error or a error value for each genotype. The OneMap until 2.1 version have only the global error option.

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.
`error`	matrix containing HMM emission probabilities

Author(s)

Cristiane Taniguti <`chtaniguti@tamu.edu`>

References

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

See Also

`make_seq`

Examples

```
data(onemap_example_out)
new.data <- create_probs(onemap_example_out, global_error = 10^-5)
```

`draw_map` *Draw a genetic 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.

Value

figure with genetic map draw

Author(s)

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

Examples

```
 #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)
```

`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 = FALSE,
  y.axis = TRUE,
  space = NULL,
  col.group = NULL,
  col.mark = NULL,
  col.tag = NULL,
  output = NULL,
  verbose = TRUE
)
```

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 automatically 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/theirs label(s).
`output`	the name of the output file. The file format can be specified by adding its extension. Available formats: 'bmp', 'jpeg', 'png', 'tiff', 'pdf' and 'eps' (default).
`verbose`	If `TRUE`, print tracing information.

Value

ggplot graphic with genetic map draw

Author(s)

Getulio Caixeta Ferreira, <`getulio.caifer@gmail.com`>

Examples

```
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"),
output = paste0(tempfile(), ".png"))
```

`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

```
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)))
```

`edit_order_onemap` *Edit sequence ordered by reference genome positions comparing to another set order*

Description

Edit sequence ordered by reference genome positions comparing to another set order

Usage

```
edit_order_onemap(input.seq)
```

Arguments

`input.seq`	object of class sequence with alternative order (not genomic order)

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

`empty_onemap_obj` *Produce empty object to avoid code break. Function for internal purpose.*

Description

Produce empty object to avoid code break. Function for internal purpose.

Usage

```
empty_onemap_obj(vcf, P1, P2, cross)
```

Arguments

`vcf`	object of class vcfR
`P1`	character with parent 1 ID
`P2`	character with parent 2 ID
`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.

Value

An empty 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)

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

`export_mappoly_genoprob` *Export genotype probabilities in MAPpoly format (input for QTLpoly)*

Description

Export genotype probabilities in MAPpoly format (input for QTLpoly)

Usage

```
export_mappoly_genoprob(input.map)
```

Arguments

`input.map`	object of class ‘sequence‘

Value

object of class ‘mappoly.genoprob‘

`export_viewpoly` *Export OneMap maps to be visualized in VIEWpoly*

Description

Export OneMap maps to be visualized in VIEWpoly

Usage

```
export_viewpoly(seqs.list)
```

Arguments

`seqs.list`	a list with ‘sequence‘ objects

Value

object of class viewmap

---

extract_depth *Extract allele counts of progeny and parents of vcf file*

Description

Uses vcfR package and onemap object to generates list of vectors with reference allele count and total counts for each marker and genotypes included in onemap object (only available for biallelic sites)

Usage

```
extract_depth(
  vcfR.object = NULL,
  onemap.object = NULL,
  vcf.par = c("GQ", "AD", "DPR, PL", "GL"),
  parent1 = "P1",
  parent2 = "P2",
  f1 = "F1",
  recovering = FALSE
)
```

Arguments

`vcfR.object`	object output from vcfR package
`onemap.object`	onemap object output from read_onemap, read_mapmaker or onemap_read_vcf function
`vcf.par`	vcf format field that contain allele counts informations, the implemented are: AD, DPR, GQ, PL, GL. AD and DPR return a list with allele depth information. GQ returns a matrix with error probability for each genotype. PL return a data.frame with genotypes probabilities for every genotype.
`parent1`	parent 1 identification in vcfR object
`parent2`	parent 2 identification in vcfR object
`f1`	if your cross type is f2, you must define the F1 individual
`recovering`	TRUE/FALSE, if TRUE evaluate all markers from vcf file, if FALSE evaluate only markers in onemap object

Value

list containing the following components:

`palt`	a `matrix` with parent 1 and 2 alternative allele counts.
`pref`	a `matrix` with parent 1 and 2 reference allele counts.
`psize`	a `matrix` with parent 1 and 2 total allele counts.
`oalt`	a `matrix` with progeny alternative allele counts.

`oref`	a `matrix` with progeny reference allele counts.
`osize`	a `matrix` with progeny total allele counts.
`n.mks`	total number of markers.
`n.ind`	total number of individuals in progeny.
`inds`	progeny individuals identification.
`mks`	markers identification.
`onemap.object`	same onemap.object inputted

Author(s)

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

`filter_2pts_gaps` *Filter markers based on 2pts distance*

Description

Filter markers based on 2pts distance

Usage

```
filter_2pts_gaps(input.seq, max.gap = 10)
```

Arguments

`input.seq`	object of class sequence with ordered markers
`max.gap`	maximum gap measured in kosambi centimorgans allowed between adjacent markers. Markers that presents the defined distance between both adjacent neighbors will be removed.

Value

New sequence 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`	object of class `onemap` with the raw data.
`twopt`	object of class `rf_2pts` with the 2-point analyses.

Author(s)

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

`filter_missing` *Filter markers according with a missing data threshold*

Description

Filter markers according with a missing data threshold

Usage

```
filter_missing(
  onemap.obj = NULL,
  threshold = 0.25,
  by = "markers",
  verbose = TRUE
)
```

Arguments

`onemap.obj`	an object of class `onemap`.
`threshold`	a numeric from 0 to 1 to define the threshold of missing data allowed
`by`	character defining if ‘markers‘ or ‘individuals‘ should be filtered
`verbose`	A logical, if TRUE it output progress status information.

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.
`error`	matrix containing HMM emission probabilities

Author(s)

Cristiane Taniguti, <chtaniguti@tamu.edu>

Examples

```
data(onemap_example_out)
filt_obj <- filter_missing(onemap_example_out, threshold=0.25)
```

`filter_prob` *Function filter genotypes by genotype probability*

Description

Function filter genotypes by genotype probability

Usage

```
filter_prob(onemap.obj = NULL, threshold = 0.8, verbose = TRUE)
```

Arguments

`onemap.obj`	an object of class `onemap`.
`threshold`	a numeric from 0 to 1 to define the threshold for the probability of the called genotype (highest probability)
`verbose`	If `TRUE`, print tracing information.

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.
`error`	matrix containing HMM emission probabilities

Author(s)

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

Examples

```
data(onemap_example_out)
filt_obj <- filter_prob(onemap_example_out, threshold=0.8)
```

`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)
```

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.

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`

Examples

```
data("vcf_example_out")
(bins<-find_bins(vcf_example_out, exact=FALSE))
```

`generate_overlapping_batches` *Function to divide the sequence in batches with user defined size*

Description

Function to divide the sequence in batches with user defined size

Usage

```
generate_overlapping_batches(input.seq, size = 50, overlap = 15)
```

Arguments

`input.seq`	an object of class `sequence`.
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches

`group` *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

```
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.

Author(s)

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

References

Lincoln, S. E., Daly, M. J. and Lander, E. S. (1993) Constructing genetic linkage maps with MAPMAKER/EXP Version 3.0: a tutorial and reference manual. *A Whitehead Institute for Biomedical Research Technical Report*.

See Also

`rf_2pts` and `make_seq`

Examples

```
data(onemap_example_out)
twopts <- rf_2pts(onemap_example_out)

all.data <- make_seq(twopts,"all")
link_gr <- group(all.data)
link_gr
print(link_gr, details=FALSE) #omit the names of the markers
```

`group_seq` *Assign markers to preexisting linkage groups*

Description

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

```
group_seq(
  input.2pts,
  seqs = "CHROM",
  unlink.mks = "all",
  repeated = FALSE,
  LOD = NULL,
  max.rf = NULL,
  min_mks = 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`	a 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.
`min_mks`	integer defining the minimum number of markers that a provided sequence (seqs or CHROM) should have to be considered a group.

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@tamu.edu>

See Also

`make_seq` and `group`

Examples

```
data(onemap_example_out) # load OneMap's fake dataset for a outcrossing population
data(vcf_example_out) # load OneMap's fake dataset from a VCF file for a outcrossing population
comb_example <- combine_onemap(onemap_example_out, vcf_example_out) # Combine datasets
twopts <- rf_2pts(comb_example)

out_CHROM <- group_seq(twopts, seqs="CHROM", repeated=FALSE)
out_CHROM

seq1 <- make_seq(twopts, c(1,2,3,4,5,25,26))
seq2 <- make_seq(twopts, c(8,18))
```

```
seq3 <- make_seq(twopts, c(4,16,20,21,24,29))

out_seqs <- group_seq(twopts, seqs=list(seq1,seq2,seq3))
out_seqs
```

`group_upgma` *Assign markers to linkage groups*

Description

Identifies linkage groups of markers using the results of two-point (pairwise) analysis and UPGMA method. Function adapted from MAPpoly package written by Marcelo Mollinari.

Usage

```
group_upgma(input.seq, expected.groups = NULL, inter = TRUE, comp.mat = FALSE)
```

Arguments

`input.seq`	an object of class `mappoly.rf.matrix`
`expected.groups`	when available, inform the number of expected linkage groups (i.e. chromosomes) for the species
`inter`	if `TRUE` (default), plots a dendrogram highlighting the expected groups before continue
`comp.mat`	if `TRUE`, shows a comparison between the reference based and the linkage based grouping, if the sequence information is available (default = FALSE)

Value

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

`data.name`	the referred dataset name
`hc.snp`	a list containing information related to the UPGMA grouping method
`expected.groups`	the number of expected linkage groups
`groups.snp`	the groups to which each of the markers belong
`seq.vs.grouped.snp`	comparison between the genomic group information (when available) and the groups provided by `group_upgma`
`LOD`	minimum LOD Score to declare linkage.
`max.rf`	maximum recombination fraction to declare linkage.
`twopt`	name of the object of class `rf.2ts` used as input, i.e., containing information used to assign markers to linkage groups.

Author(s)

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

Cristiane Taniguti <`chtaniguti@tamu.edu`>

References

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

Examples

```
data("vcf_example_out")
twopts <- rf_2pts(vcf_example_out)
input.seq <- make_seq(twopts, "all")
lgs <- group_upgma(input.seq, expected.groups = 3, comp.mat=TRUE, inter = FALSE)
plot(lgs)
```

`haldane` *Apply Haldane mapping function*

Description

Apply Haldane mapping function

Usage

```
haldane(rcmb)
```

Arguments

`rcmb`	vector of recombination fraction values

Value

vector with centimorgan values

`keep_only_selected_mks` *Keep in the onemap and twopts object only markers in the sequences*

Description

Keep in the onemap and twopts object only markers in the sequences

Usage

```
keep_only_selected_mks(list.sequences = NULL)
```

Arguments

`list.sequences`	a list of objects 'sequence'

Value

a list of objects 'sequences' with internal onemap and twopts objects reduced

Author(s)

Cristiane Taniguti

`kosambi` *Apply Kosambi mapping function*

Description

Apply Kosambi mapping function

Usage

```
kosambi(rcmb)
```

Arguments

`rcmb`	vector of recombination fraction values

Value

vector with centimorgan values

`load_onemap_sequences` *Load list of sequences saved by save_onemap_sequences*

Description

Load list of sequences saved by save_onemap_sequences

Usage

```
load_onemap_sequences(filename)
```

Arguments

`filename`	name of the file to be loaded

`make_seq` *Create a sequence of markers based on other OneMap object types*

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`, `group.upgma` or `order`, this argument has no effect.
`data.name`	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`	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`	object of class `onemap` with the raw data.
`twopt`	object of class `rf_2pts` with the 2-point analyses.

Author(s)

Gabriel Margarido, `<gramarga@gmail.com>`

References

Lander, E. S., Green, P., Abrahamson, J., Barlow, A., Daly, M. J., Lincoln, S. E. and Newburg, L. (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* 1: 174-181.

See Also

`compare`, `try_seq`, `order_seq` and `map`.

Examples

```
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)

all_mark <- make_seq(twopt,"all")
all_mark <- make_seq(twopt,1:30) # same as above, for this data set
groups <- group(all_mark)
LG1 <- make_seq(groups,1)
LG1.ord <- order_seq(LG1)
(LG1.final <- make_seq(LG1.ord)) # safe order
(LG1.final.all <- make_seq(LG1.ord,"force")) # forced order

markers <- make_seq(twopt,c(2,3,12,14))
markers.comp <- compare(markers)
(base.map <- make_seq(markers.comp))
base.map <- make_seq(markers.comp,1,1) # same as above
(extend.map <- try_seq(base.map,30))
(base.map <- make_seq(extend.map,5)) # fifth position is the best
```

`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,
  rm_unlinked = FALSE,
  phase_cores = 1,
  parallelization.type = "PSOCK",
  global_error = NULL,
  genotypes_errors = NULL,
  genotypes_probs = NULL
)
```

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.

`rm_unlinked`	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 and map_avoid_unlinked functions).
`phase_cores`	number of computer cores to be used in analysis
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.
`global_error`	single value to be considered as error probability in HMM emission function
`genotypes_errors`	matrix individuals x markers with error values for each marker
`genotypes_probs`	table containing the probability distribution for each combination of marker × individual. Each line on this table represents the combination of one marker with one individual, and the respective probabilities. The table should contain four three columns (prob(AA), prob(AB) and prob(BB)) and individuals*markers rows.

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`>, with minor changes by Cristiane Taniguti and Bastian Schiffthaler

References

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Jiang, C. and Zeng, Z.-B. (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* 101: 47-58.

Lander, E. S., Green, P., Abrahamson, J., Barlow, A., Daly, M. J., Lincoln, S. E. and Newburg, L. (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* 1: 174-181.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002a) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. (2002b). Linkage mapping of sex-specific differences. *Genetical Research* 79: 85-96

See Also

`make_seq`

Examples

```
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)
```

`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 1 3 ... ..- attr(*, "dimnames")=List of 2 .. ..$: NULL .. ..$: chr [1:66] "M1" "M2" "M3" "M4" ... $ n.ind : num 200 $ n.mar : num 66 $ segr.type : chr [1:66] "A.H.B" "C.A" "D.B" "C.A" ... $ segr.type.num: num [1:66] 1 3 2 3 3 2 1 3 2 1 ... $ input : chr "/home/cristiane/R/x86_64-pc-linux-gnu-library/3.4/onemap/extdata/mapmaker_example_f2.raw" $ n.phe : num 1 $ pheno : num [1:200, 1] 37.6 36.4 37.2 35.8 37.1 ... ..- attr(*, "dimnames")=List of 2 .. ..$: NULL .. ..$: chr "Trait_1" - 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(mapmaker_example_f2)

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

`map_avoid_unlinked` *Repeat HMM if map find unlinked marker*

Description

Repeat HMM if map find unlinked marker

Usage

```
map_avoid_unlinked(
  input.seq,
  size = NULL,
  overlap = NULL,
  phase_cores = 1,
  tol = 1e-04,
  parallelization.type = "PSOCK",
  max.gap = FALSE,
  global_error = NULL,
  genotypes_errors = NULL,
  genotypes_probs = NULL
)
```

Arguments

`input.seq`	object of class sequence
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`tol`	tolerance for the C routine, i.e., the value used to evaluate convergence.
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.
`max.gap`	the marker will be removed if it have gaps higher than this defined threshold in both sides
`global_error`	single value to be considered as error probability in HMM emission function
`genotypes_errors`	matrix individuals x markers with error values for each marker
`genotypes_probs`	table containing the probability distribution for each combination of marker × individual. Each line on this table represents the combination of one marker with one individual, and the respective probabilities. The table should contain four three columns (prob(AA), prob(AB) and prob(BB)) and individuals*markers rows.

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.

Examples

```
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)

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

markers <- make_seq(twopt,c(30,12,3,14,2),phase=c(4,1,4,3)) # incorrect phases
```

```
map_avoid_unlinked(markers)
```

`map_overlapping_batches`

Mapping overlapping batches

Description

Apply the batch mapping algorithm using overlapping windows.

Usage

```
map_overlapping_batches(
  input.seq,
  size = 50,
  overlap = 15,
  phase_cores = 1,
  verbose = FALSE,
  seeds = NULL,
  tol = 1e-04,
  rm_unlinked = TRUE,
  max.gap = FALSE,
  parallelization.type = "PSOCK"
)
```

Arguments

`input.seq`	an object of class `sequence`.
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`verbose`	A logical, if TRUE its output progress status information.
`seeds`	A vector of phase information used as seeds for the first batch
`tol`	tolerance for the C routine, i.e., the value used to evaluate convergence.
`rm_unlinked`	When some pair of markers do not follow the linkage criteria, if `TRUE` one of the markers is removed and map is performed again.
`max.gap`	the marker will be removed if it have gaps higher than this defined threshold in both sides
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.

Details

This algorithm implements the overlapping batch maps for high density marker sets. The mapping problem is reduced to a number of subsets (batches) which carry information forward in order to more accurately estimate recombination fractions and phasing. It is a adapted version of map.overlapping.batches function of BatchMap package. The main differences are that this onemap version do not have the option to reorder the markers according to ripple algorithm and, if the it finds markers that do not reach the linkage criterias, the algorithm remove the problematic marker and repeat the analysis. Than, the output map can have few markers compared with the input.seq.

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 `outcross` with the raw data.
`twopt`	name of the object of class `rf.2pts` with the 2-point analyses.

See Also

`pick_batch_sizes`, `map`

`map_save_ram` *Perform map using background objects with only selected markers. It saves ram memory during the procedure. It is useful if dealing with many markers in total data set.*

Description

Perform map using background objects with only selected markers. It saves ram memory during the procedure. It is useful if dealing with many markers in total data set.

Usage

```
map_save_ram(
  input.seq,
  tol = 1e-04,
  verbose = FALSE,
  rm_unlinked = FALSE,
  phase_cores = 1,
  size = NULL,
```

```
  overlap = NULL,
  parallelization.type = "PSOCK",
  max.gap = FALSE
)
```

Arguments

`input.seq`	object of class sequence
`tol`	tolerance for the C routine, i.e., the value used to evaluate convergence.
`verbose`	If `TRUE`, print tracing information.
`rm_unlinked`	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 and map_avoid_unlinked functions).
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.
`max.gap`	the marker will be removed if it have gaps higher than this defined threshold in both sides

---

`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*):

Type	Cross	Segregation
A.1	ab x cd	1:1:1:1
A.2	ab x ac	1:1:1:1
A.3	ab x co	1:1:1:1
A.4	ao x bo	1:1:1:1
B1.5	ab x ao	1:2:1
B2.6	ao x ab	1:2:1
B3.7	ab x ab	1:2:1
C8	ao x ao	3:1
D1.9	ab x cc	1:1
D1.10	ab x aa	1:1
D1.11	ab x oo	1:1
D1.12	bo x aa	1:1
D1.13	ao x oo	1:1
D2.14	cc x ab	1:1
D2.15	aa x ab	1:1
D2.16	oo x ab	1:1
D2.17	aa x bo	1:1
D2.18	oo x ao	1:1

Value

data.frame with segregation types of all markers in the sequence are displayed on the screen.

Author(s)

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

References

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

See Also

`make_seq`

Examples

```
data(onemap_example_out)
twopts <- rf_2pts(onemap_example_out)
markers.ex <- make_seq(twopts,c(3,6,8,12,16,25))
marker_type(input.seq = 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
```

`mds_onemap` *OneMap interface with MDSMap package with option for multipoint distances estimation*

Description

For a given sequence of markers, apply mds method described in Preedy and Hackett (2016) using MDSMap package to ordering markers and estimates the genetic distances with OneMap multipoint approach. Also gives MDSMap input file format for directly analysis in this package.

Usage

```
mds_onemap(
  input.seq,
  out.file = NULL,
  p = NULL,
  ispc = TRUE,
  displaytext = FALSE,
  weightfn = "lod2",
  mapfn = "haldane",
  ndim = 2,
  rm_unlinked = TRUE,
  size = NULL,
  overlap = NULL,
  phase_cores = 1,
  tol = 1e-05,
  hmm = TRUE,
  parallelization.type = "PSOCK"
)
```

Arguments

`input.seq`	an object of class `sequence`
`out.file`	path to the generated MDSMap input file.
`p`	Integer - the penalty for deviations from the sphere - higher p forces points more closely onto a sphere.
`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'.

`ndim`	number of dimensions to be considered in the multidimensional scaling procedure (default = 2)
`rm_unlinked`	When some pair of markers do not follow the linkage criteria, if `TRUE` one of the markers is removed and mds is performed again.
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`tol`	tolerance for the C routine, i.e., the value used to evaluate convergence.
`hmm`	logical defining if the HMM must be applied to estimate multipoint genetic distances
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.

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@tamu.edu>`

References

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

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. *Heredity* 103: 494-502.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002a) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. (2002b). Linkage mapping of sex-specific differences. *Genetical Research* 79: 85-96

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 1 2 1 2 1 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 38.9 ... ..- 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\qtl` 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
```

`onemap_example_f2` *Simulated data from a F2 population*

Description

Simulated data set 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 3 3 2 1 3 2 1 ... $ n.phe : int 1 $ pheno : num [1:200, 1] 37.6 36.4 37.2 35.8 37.1 ... ..- attr(*, "dimnames")=List of 2 .. ..$: NULL .. ..$: 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)
```

`onemap_example_out` *Data from a full-sib family derived from two outbred parents*

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
```

`onemap_example_riself` *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 vcf file to onemap object*

Description

Converts data from a vcf file to onemap initial object, while identify the appropriate marker segregation patterns.

Usage

```
onemap_read_vcfR(
  vcf = NULL,
  vcfR.object = NULL,
  cross = NULL,
  parent1 = NULL,
  parent2 = NULL,
  f1 = NULL,
  only_biallelic = TRUE,
  output_info_rds = NULL,
  verbose = TRUE
)
```

Arguments

`vcf`	string defining the path to VCF file;
`vcfR.object`	object of class vcfR;
`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. If f2 backcross population, define here the ID of the backcrossed parent.
`parent2`	`string` specifying sample ID of the second parent.
`f1`	`string` if you are working with f2 intercross or backcross populations you may have f1 parents in you vcf, specify its ID here
`only_biallelic`	if TRUE (default) only biallelic markers are considered, if FALSE multiallelic markers are included.
`output_info_rds`	define a name for the file with alleles information.
`verbose`	A logical, if TRUE it output progress status information.

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.

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.
`error`	matrix containing HMM emission probabilities

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

See Also

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

Examples

```
data <- onemap_read_vcfR(vcf=system.file("extdata/vcf_example_out.vcf.gz", package = "onemap"),
                 cross="outcross",
                 parent1=c("P1"),
                 parent2=c("P2"))
```

`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

```
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,
  rm_unlinked = FALSE,
  verbose = FALSE
)
```

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.

`rm_unlinked`	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 and map_avoid_unlinked functions).
`verbose`	A logical, if TRUE its output progress status information.

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 $\frac{n.init!}{2}$ 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

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Jiang, C. and Zeng, Z.-B. (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* 101: 47-58.

Lander, E. S. and Green, P. (1987). Construction of multilocus genetic linkage maps in humans. *Proc. Natl. Acad. Sci. USA* 84: 2363-2367.

Lander, E. S., Green, P., Abrahamson, J., Barlow, A., Daly, M. J., Lincoln, S. E. and Newburg, L. (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* 1: 174-181.

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. *Heredity* 103: 494-502.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002a) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. (2002b). Linkage mapping of sex-specific differences. *Genetical Research* 79: 85-96

See Also

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

Examples

```
#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
```

`ord_by_geno` *Order the markers in a sequence using the genomic position*

Description

Order the markers in a sequence using the genomic position

Usage

```
ord_by_geno(input.seq)
```

Arguments

`input.seq`	object of class ‘sequence‘

Value

An object of class `sequence`

Author(s)

Cristiane Taniguti

`parents_haplotypes` *Generates data.frame with parents estimated haplotypes*

Description

Generates data.frame with parents estimated haplotypes

Usage

```
parents_haplotypes(
  ...,
  group_names = NULL,
  map.function = "kosambi",
  ref_alt_alleles = FALSE
)
```

Arguments

`...`	objects of class sequence
`group_names`	vector of characters defining the group names
`map.function`	"kosambi" or "haldane" according to which was used to build the map
`ref_alt_alleles`	TRUE to return parents haplotypes as reference and alternative ref_alt_alleles codification

Value

data.frame with group ID (group), marker number (mk.number) and names (mk.names), position in centimorgan (dist) and parents haplotypes (P1_1, P1_2, P2_1, P2_2)

Author(s)

Getulio Caixeta Ferreira, <getulio.caifer@gmail.com>
Cristiane Taniguti, <chtaniguti@tamu.edu>

Examples

```
data("onemap_example_out")
twopts <- rf_2pts(onemap_example_out)
lg1 <- make_seq(twopts, 1:5)
lg1.map <- map(lg1)
parents_haplotypes(lg1.map)
```

`pick_batch_sizes` *Picking optimal batch size values*

Description

Suggest an optimal batch size value for use in `map_overlapping_batches`

Usage

```
pick_batch_sizes(input.seq, size = 50, overlap = 15, around = 5)
```

Arguments

`input.seq`	an object of class sequence.
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`around`	The range around the center which is maximally allowed to be searched.

Value

An integer value for the size which most evenly divides batches. In case of ties, bigger batch sizes are preferred.

Author(s)

Bastian Schiffthaler, <bastian.schiffthaler@umu.se>

See Also

`map_overlapping_batches`

Examples

```
  LG <- structure(list(seq.num = seq(1,800)), class = "sequence")
  batchsize <- pick_batch_sizes(LG, 50, 19)
```

`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 of 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 the segregation pattern.

Usage

```
## 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

```
# library(ggplot2)
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
# For details, see the help of ggplot2's function ggsave()
g <- plot(onemap_example_bc)

data(onemap_example_f2) # Loads a fake backcross dataset installed with onemap
plot(onemap_example_f2) # This will show you the graph

# You can store the graphic in an object, then save it with a number of properties
# For details, see the help of ggplot2's function ggsave()
g <- plot(onemap_example_f2)
```

```
data(onemap_example_out) # Loads a fake full-sib dataset installed with onemap
plot(onemap_example_out) # This will show you the graph for all markers
plot(onemap_example_out, all=FALSE) # This will show you the graph splitted for marker types

# You can store the graphic in an object, then save it.
# For details, see the help of ggplot2's function ggsave()
g <- plot(onemap_example_out, all=FALSE)
```

`plot.onemap_progeny_haplotypes`

Plots progeny haplotypes

Description

Figure is generated with the haplotypes for each selected individual. As a representation, the recombination breakpoints are here considered to be in the mean point of the distance between two markers. It is important to highlight that it did not reflects the exact breakpoint position, specially if the genetic map have low resolution.

Usage

```
## S3 method for class 'onemap_progeny_haplotypes'
plot(
  x,
  col = NULL,
  position = "stack",
  show_markers = TRUE,
  main = "Genotypes",
  ncol = 4,
  ...
)
```

Arguments

`x`	object of class onemap_progeny_haplotypes
`col`	Color of parents' homologous.
`position`	"split" or "stack"; if "split" (default) the alleles' are plotted separately. if "stack" the parents' alleles are plotted together.
`show_markers`	logical; if `TRUE`, the markers (default) are plotted.
`main`	An overall title for the plot; default is `NULL`.
`ncol`	number of columns of the facet_wrap
`...`	currently ignored

Value

a ggplot graphic

Author(s)

Getulio Caixeta Ferreira, <`getulio.caifer@gmail.com`>

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

Examples

```
data("onemap_example_out")
twopts <- rf_2pts(onemap_example_out)
lg1 <- make_seq(twopts, 1:5)
lg1.map <- map(lg1)
plot(progeny_haplotypes(lg1.map))
```

`plot.onemap_progeny_haplotypes_counts`

Plot recombination breakpoints counts for each individual

Description

Plot recombination breakpoints counts for each individual

Usage

```
## S3 method for class 'onemap_progeny_haplotypes_counts'
plot(x, by_homolog = FALSE, n.graphics = NULL, ncol = NULL, ...)
```

Arguments

`x`	object of class onemap_progeny_haplotypes_counts
`by_homolog`	logical, if TRUE plots counts by homolog (two for each individuals), if FALSE plots total counts by individual
`n.graphics`	integer defining the number of graphics to be plotted, they separate the individuals in different plots
`ncol`	integer defining the number of columns in plot
`...`	currently ignored

Value

a ggplot graphic

Examples

```
data("onemap_example_out")
twopts <- rf_2pts(onemap_example_out)
lg1 <- make_seq(twopts, 1:5)
lg1.map <- map(lg1)
prog.haplo <- progeny_haplotypes(lg1.map, most_likely = TRUE)
plot(progeny_haplotypes_counts(prog.haplo))
```

`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 -log10(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

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

 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
```

`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

```
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)
```

`plot_genome_vs_cm` *Draws a physical vs cM map*

Description

Provides simple genetic to physical ggplot.

Usage

```
plot_genome_vs_cm(map.list, mapping_function = "kosambi", group.names = NULL)
```

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.
`mapping_function`	either "kosambi" or "haldane"
`group.names`	vector with group name for each sequence object in the map.list

Value

ggplot with cM on x-axis and physical position on y-axis

Author(s)

Jeekin Lau, <`jeekinlau@gmail.com`>

`print.compare` *print method for object class 'compare'*

Description

print method for object class 'compare'

Usage

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

Arguments

`x`	object of class compare
`...`	currently ignored

Value

compare object description

`print.onemap` *Print method for object class 'onemap'*

Description

Print method for object class 'onemap'

Usage

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

Arguments

`x`	object of class onemap
`...`	currently ignored

Value

printed information about onemap object

`print.onemap_bin` *print method for object class 'onemap_bin'*

Description

print method for object class 'onemap_bin'

Usage

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

Arguments

`x`	object of class `onemap_bin`
`...`	currently ignored

Value

No return value, called for side effects

---

print.onemap_segreg_test *Show the results of segregation tests*

Description

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

```
## 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

```
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
```

---

print.order *Print order_seq object*

Description

Print order_seq object

Usage

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

Arguments

`x`	object of class order_seq
`...`	currently ignored

Value

printed information about order_seq object

`print.sequence` *Print method for object class 'sequence'*

Description

Print method for object class 'sequence'

Usage

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

Arguments

`x`	object of class sequence
`...`	corrently ignored

Value

printed information about sequence object

`progeny_haplotypes` *Generate data.frame with genotypes estimated by HMM and its probabilities*

Description

Generate data.frame with genotypes estimated by HMM and its probabilities

Usage

```
progeny_haplotypes(..., ind = 1, group_names = NULL, most_likely = FALSE)
```

Arguments

`...`	Map(s) or list(s) of maps. Object(s) of class sequence.
`ind`	vector with individual index to be evaluated or "all" to include all individuals
`group_names`	Names of the groups.
`most_likely`	logical; if `TRUE`, the most likely genotype receive 1 and all the rest 0. If there are more than one most likely both receive 0.5. if FALSE (default) the genotype probability is plotted.

Value

a data.frame information: individual (ind) and marker ID, group ID (grp), position in centimorgan (pos), genotypes probabilities (prob), parents, and the parents homologs and the allele IDs.

Author(s)

Getulio Caixeta Ferreira, <`getulio.caifer@gmail.com`>

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

Examples

```
data("onemap_example_out")
twopts <- rf_2pts(onemap_example_out)
lg1 <- make_seq(twopts, 1:5)
lg1.map <- map(lg1)
progeny_haplotypes(lg1.map)
```

`progeny_haplotypes_counts`

Plot number of breakpoints by individuals

Description

Generate graphic with the number of break points for each individual considering the most likely genotypes estimated by the HMM. Genotypes with same probability for two genotypes are removed. By now, only available for outcrossing and f2 intercross.

Usage

```
progeny_haplotypes_counts(x)
```

Arguments

`x`	object of class onemap_progeny_haplotypes

Value

a `data.frame` with columns individuals ID (ind), group ID (grp), homolog (homolog) and counts of breakpoints

Examples

```
data("onemap_example_out")
twopts <- rf_2pts(onemap_example_out)
lg1 <- make_seq(twopts, 1:5)
lg1.map <- map(lg1)
progeny_haplotypes_counts(progeny_haplotypes(lg1.map, most_likely = TRUE))
```

`rcd` *Rapid Chain Delineation*

Description

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

Usage

```
rcd(
  input.seq,
  LOD = 0,
  max.rf = 0.5,
  tol = 1e-04,
  rm_unlinked = TRUE,
  size = NULL,
  overlap = NULL,
  phase_cores = 1,
  hmm = TRUE,
  parallelization.type = "PSOCK",
  verbose = TRUE
)
```

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.
`rm_unlinked`	When some pair of markers do not follow the linkage criteria, if `TRUE` one of the markers is removed and rcd is performed again.

`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`hmm`	logical defining if the HMM must be applied to estimate multipoint genetic distances
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.
`verbose`	A logical, if TRUE it output progress status information.

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

Basten, C. J., Weir, B. S. and Zeng, Z.-B. (2005) *QTL Cartographer Version 1.17: A Reference Manual and Tutorial for QTL Mapping*.

Doerge, R. W. (1996) Constructing genetic maps by rapid chain delineation. *Journal of Quantitative Trait Loci* 2: 121-132.

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. *Heredity* 103: 494-502.

See Also

`make_seq`, `map`

Examples

```
#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, hmm = FALSE)

#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, hmm = FALSE)
LG1.rcd
```

`read_mapmaker` *Read data from a Mapmaker raw file*

Description

Imports data from a Mapmaker raw file.

Usage

```
read_mapmaker(file = NULL, dir = NULL, verbose = TRUE)
```

Arguments

`file`	the name of the input file which contains the data to be read.
`dir`	directory where the input file is located.
`verbose`	A logical, if TRUE it output progress status information.

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:

`geno`	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.

`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.

`n.ind`	number of individuals.
`n.mar`	number of markers.
`segr.type`	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.
`segr.type.num`	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.
`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. Currently ignored.
`error`	matrix containing HMM emission probabilities

Author(s)

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

References

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Lincoln, S. E., Daly, M. J. and Lander, E. S. (1993) Constructing genetic linkage maps with MAPMAKER/EXP Version 3.0: a tutorial and reference manual. *A Whitehead Institute for Biomedical Research Technical Report*.

See Also

`mapmaker_example_bc` and `mapmaker_example_f2` raw files in the package source.

Examples

```
map_data <-read_mapmaker(file=system.file("extdata/mapmaker_example_f2.raw", package = "onemap"))
 #Checking 'mapmaker_example_f2'
 data(mapmaker_example_f2)
 names(mapmaker_example_f2)
```

`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

```
read_onemap(inputfile = NULL, dir = NULL, verbose = TRUE)
```

Arguments

`inputfile`	the name of the input file which contains the data to be read.
`dir`	directory where the input file is located.
`verbose`	A logical, if TRUE it output progress status information.

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.
`error`	matrix containing HMM emission probabilities

Author(s)

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

References

Lincoln, S. E., Daly, M. J. and Lander, E. S. (1993) Constructing genetic linkage maps with MAPMAKER/EXP Version 3.0: a tutorial and reference manual. *A Whitehead Institute for Biomedical Research Technical Report*.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

See Also

`combine_onemap` and the `example` directory in the package source.

Examples

```
outcr_data <- read_onemap(inputfile=
system.file("extdata/onemap_example_out.raw", package= "onemap"))
```

`record` *Recombination Counting and Ordering*

Description

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

Usage

```
record(
  input.seq,
  times = 10,
  LOD = 0,
  max.rf = 0.5,
  tol = 1e-04,
  rm_unlinked = TRUE,
  size = NULL,
  overlap = NULL,
  phase_cores = 1,
  hmm = TRUE,
  parallelization.type = "PSOCK",
  verbose = TRUE
)
```

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.
`rm_unlinked`	When some pair of markers do not follow the linkage criteria, if `TRUE` one of the markers is removed and record is performed again.
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`hmm`	logical defining if the HMM must be applied to estimate multipoint genetic distances
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.
`verbose`	A logical, if TRUE it output progress status information.

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

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. *Heredity* 103: 494-502.

Van Os, H., Stam, P., Visser, R.G.F. and Van Eck, H.J. (2005) RECORD: a novel method for ordering loci on a genetic linkage map. *Theoretical and Applied Genetics* 112: 30-40.

See Also

`make_seq` and `map`

Examples

```
##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, hmm = FALSE)

##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, hmm = FALSE)
LG1.rec
```

`remove_inds` *Remove individuals from the onemap object*

Description

Remove individuals from the onemap object

Usage

```
remove_inds(onemap.obj = NULL, rm.ind = NULL, list.seqs = NULL)
```

Arguments

`onemap.obj`	object of class onemap
`rm.ind`	vector of characters with individuals names
`list.seqs`	list of objects of class sequence

Value

An object of class `onemap` without the selected individuals if onemap object is used as input, or a list of objects of class `sequence` without the selected individuals if a list of sequences objects is use as input

Author(s)

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

`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, rm_mks = FALSE)
```

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.
`rm_mks`	logical. If `TRUE` the algorithm will remove the markers for which it found numerical problems to calculates the recombination fraction. The numerical problems can happens because of excess of missing data or segregation deviation.

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 overriden.

Author(s)

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

References

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Examples

```
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
```

`rf_graph_table` *Plots pairwise recombination fractions and LOD Scores in a heatmap*

Description

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

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

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 interative 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 pallete.
`display`	logical. If inter `TRUE` and display `TRUE` interactive graphic is plotted in browser automatically when run the function

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) $\times$ 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.

Value

a ggplot graphic

Author(s)

Rodrigo Amadeu, <`rramadeu@gmail.com`>

Examples

```
##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)


  ##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")
  }
```

`rf_snp_filter_onemap` *Filter markers according with a two-points recombination fraction and LOD threshold. Adapted from MAPpoly.*

Description

Filter markers according with a two-points recombination fraction and LOD threshold. Adapted from MAPpoly.

Usage

```
rf_snp_filter_onemap(
  input.seq,
  thresh.LOD.rf = 5,
```

```
  thresh.rf = 0.15,
  probs = c(0.05, 1)
)
```

Arguments

`input.seq`	an object of class `onemap`.
`thresh.LOD.rf`	LOD score threshold for recombination fraction (default = 5)
`thresh.rf`	threshold for recombination fractions (default = 0.15)
`probs`	indicates the probability corresponding to the filtering quantiles. (default = c(0.05, 1))

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`	object of class `onemap` with the raw data.
`twopt`	object of class `rf_2pts` with the 2-point analyses.

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

Examples

```
 data("vcf_example_out")
 twopts <- rf_2pts(vcf_example_out)
 seq1 <- make_seq(twopts, which(vcf_example_out$CHROM == "1"))
filt_seq <- rf_snp_filter_onemap(seq1, 20, 0.5, c(0.5,1))
```

`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, verbose = TRUE)
```

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.
`verbose`	A logical, if TRUE it output progress status information.

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

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Jiang, C. and Zeng, Z.-B. (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* 101: 47-58.

Lander, E. S., Green, P., Abrahamson, J., Barlow, A., Daly, M. J., Lincoln, S. E. and Newburg, L. (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* 1: 174-181.

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. *Heredity* 103: 494-502.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002a) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. (2002b). Linkage mapping of sex-specific differences. *Genetical Research* 79: 85-96

See Also

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

Examples

```
#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,1)
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)
```

`rm_dupli_mks` *Remove duplicated markers keeping the one with less missing data*

Description

Remove duplicated markers keeping the one with less missing data

Usage

```
rm_dupli_mks(onemap.obj)
```

Arguments

`onemap.obj`	object of class `onemap`

Value

An empty 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)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

`save_onemap_sequences` *Save a list of onemap sequence objects*

Description

The onemap sequence object contains everything users need to reproduce the complete analysis: the input onemap object, the rf_2pts result, and the sequence genetic distance and marker order. Therefore, a list of sequences is the only object users need to save to be able to recover all analysis. But simple saving the list of sequences will save many redundant objects. This redundancy is only considered by R when saving the object. For example, one input object and the rf_2pts result will be saved for every sequence.

Usage

```
save_onemap_sequences(sequences.list, filename)
```

Arguments

`sequences.list`	list of `sequence` objects
`filename`	name of the output file (Ex: my_beautiful_map.RData)

`seeded_map` *Construct the linkage map for a sequence of markers after seeding phases*

Description

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

Usage

```
seeded_map(
  input.seq,
  tol = 1e-04,
  phase_cores = 1,
  seeds,
  verbose = FALSE,
  rm_unlinked = FALSE,
  parallelization.type = "PSOCK"
)
```

Arguments

`input.seq`	an object of class `sequence`.
`tol`	tolerance for the C routine, i.e., the value used to evaluate convergence.
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`seeds`	A vector given the integer encoding of phases for the first *N* positions of the map
`verbose`	A logical, if TRUE it output progress status information.
`rm_unlinked`	When some pair of markers do not follow the linkage criteria, if `TRUE` one of the markers is removed and map is performed again.
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.

Details

Markers are mapped in the order defined in the object `input.seq`. The best combination of linkage phases is also estimated starting from the first position not in the given seeds.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 `outcross` 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>`. Modified to use seeded phases by Bastian Schiffthaler `<bastian.schiffthaler@umu.se>`

References

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Jiang, C. and Zeng, Z.-B. (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* 101: 47-58.

Lander, E. S., Green, P., Abrahamson, J., Barlow, A., Daly, M. J., Lincoln, S. E. and Newburg, L. (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* 1: 174-181.

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002a) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. (2002b). Linkage mapping of sex-specific differences. *Genetical Research* 79: 85-96

See Also

`make_seq`

Examples

```
data(onemap_example_out)
twopt <- rf_2pts(onemap_example_out)

markers <- make_seq(twopt,c(30,12,3,14,2))
seeded_map(markers, seeds = c(4,2))
```

`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 (alpha) to be considered in the test. If NULL, it uses the threshold alpha = 0.05. Bonferroni correction is applied for multiple test correction.

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_out)
# Performs the chi-square test for all markers
Chi <- test_segregation(onemap_example_out)
# 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)
```

`seq_by_type` *Extract marker number by name*

Description

Extract marker number by name

Usage

```
seq_by_type(sequence, mk_type)
```

Arguments

`sequence`	object of class or sequence
`mk_type`	vector of character with marker type to be selected

Value

New sequence object of class `sequence` with selected marker type, 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`	object of class `onemap` with the raw data.
`twopt`	object of class `rf_2pts` with the 2-point analyses.

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

See Also

`make_seq`

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,
  rm_unlinked = TRUE,
  size = NULL,
  overlap = NULL,
  phase_cores = 1,
  hmm = TRUE,
  parallelization.type = "PSOCK",
  verbose = TRUE
)
```

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.
`rm_unlinked`	When some pair of markers do not follow the linkage criteria, if `TRUE` one of the markers is removed and ug is performed again.
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`hmm`	logical defining if the HMM must be applied to estimate multipoint genetic distances
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.
`verbose`	A logical, if TRUE it output progress status information.

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 to 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

Buetow, K. H. and Chakravarti, A. (1987) Multipoint gene mapping using seriation. I. General methods. *American Journal of Human Genetics* 41: 180-188.

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. *Heredity* 103: 494-502.

See Also

`make_seq`, `map`

Examples

```
##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)
```

`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

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

Arguments

`type`	Indicates the function that should be used, which can be `"kosambi"` or `"haldane"`

Value

No return value, called for side effects

Kosambi, D. D. (1944) The estimation of map distance from recombination values. *Annuaire of Eugenetics* 12: 172-175.

Author(s)

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

References

Haldane, J. B. S. (1919) The combination of linkage values and the calculation of distance between the loci of linked factors. *Journal of Genetics* 8: 299-309.

See Also

`kosambi` and `haldane`

`simu_example_bc` *Simulated data from a backcross population*

Description

Simulated data set from a backcross population.

Usage

```
data(simu_example_bc)
```

Format

The format is: List of 11 $ geno : num [1:200, 1:54] 1 2 1 1 2 2 2 1 1 2 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:200] "BC_001" "BC_002" "BC_003" "BC_004" ... .$: chr [1:54] "M001" "M002" "M003" "M004" ... $ n.ind : int 200 $ n.mar : int 54 $ segr.type : chr [1:54] "A.H" "A.H" "A.H" "A.H" ... $ segr.type.num: num [1:54] 8 8 8 8 8 8 8 8 8 8 ... $ n.phe : int 0 $ pheno : NULL $ CHROM : NULL $ POS : NULL $ input : chr "simu_example_bc.raw" $ error : num [1:10800, 1:2] 1 1 1 1 1 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:10800] "M001_BC_001" "M002_BC_001" "M003_BC_001" "M004_BC_001" ... .$: NULL - attr(*, "class")= chr [1:2] "onemap" "backcross"

Details

A simulation of a backcross population of 200 individuals genotyped with 54 markers. There are no missing data. There are two groups, one (Chr01) with a total of 100 cM and the other (Chr10) with 150 cM. The markers are positioned equidistant from each other.

Author(s)

Cristiane Taniguti, `<chtaniguti@usp.br>`

See Also

`read_onemap` and `read_mapmaker`.

Examples

```
data(simu_example_bc)

# perform two-point analyses
twopts <- rf_2pts(simu_example_bc)
twopts
```

`simu_example_f2` *Simulated data from a f2 intercross population*

Description

Simulated data set from a f2 intercross population.

Usage

```
data(simu_example_f2)
```

Format

The format is: List of 11 $ geno : num [1:200, 1:54] 1 2 1 1 2 2 1 1 1 2 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:200] "F2_001" "F2_002" "F2_003" "F2_004" ... .$: chr [1:54] "M001" "M002" "M003" "M004" ... $ n.ind : int 200 $ n.mar : int 54 $ segr.type : chr [1:54] "C.A" "C.A" "C.A" "C.A" ... $ segr.type.num: num [1:54] 7 7 7 7 4 4 7 4 4 4 ... $ n.phe : int 0 $ pheno : NULL $ CHROM : NULL $ POS : NULL $ input : chr "simu_example_f2.raw" $ error : num [1:10800, 1:4] 1 1 1 1 1 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:10800] "M001_F2_001" "M002_F2_001" "M003_F2_001" "M004_F2_001" ... .$: NULL - attr(*, "class")= chr [1:2] "onemap" "f2"

Details

A simulation of a f2 intercross population of 200 individuals genotyped with 54 markers. There are no missing data. There are two groups, one (Chr01) with a total of 100 cM and the other (Chr10) with 150 cM. The markers are positioned equidistant from each other.

Author(s)

Cristiane Taniguti, `<chtaniguti@usp.br>`

See Also

`read_onemap` and `read_mapmaker`.

Examples

```
data(simu_example_f2)

# perform two-point analyses
twopts <- rf_2pts(simu_example_f2)
twopts
```

`simu_example_out` *Simulated data from a outcrossing population*

Description

Simulated data set from a outcrossing population.

Usage

```
data(simu_example_out)
```

Format

The format is: List of 11 $ geno : num [1:200, 1:54] 2 1 2 1 1 2 2 2 1 1 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:200] "F1_001" "F1_002" "F1_003" "F1_004" ... .$: chr [1:54] "M001" "M002" "M003" "M004" ... $ n.ind : int 200 $ n.mar : int 54 $ segr.type : chr [1:54] "D2.16" "D2.17" "D2.17" "D1.9" ... $ segr.type.num: num [1:54] 7 7 7 6 1 3 3 1 7 6 ... $ n.phe : int 0 $ pheno : NULL $ CHROM : NULL $ POS : NULL $ input : chr "simu_example_out.raw" $ error : num [1:10800, 1:4] 1.00e-05 1.00e-05 1.00e-05 1.00 3.33e-06 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:10800] "M001_F1_001" "M002_F1_001" "M003_F1_001" "M004_F1_001" ... .$: NULL - attr(*, "class")= chr [1:2] "onemap" "outcross"

Details

A simulation of a outcrossing population of 200 individuals genotyped with 54 markers. There are no missing data. There are two groups, one (Chr01) with a total of 100 cM and the other (Chr10) with 150 cM. The markers are positioned equidistant from each other.

Author(s)

Cristiane Taniguti, `<chtaniguti@usp.br>`

See Also

`read_onemap` and `read_mapmaker`.

Examples

```
data(simu_example_out)

# perform two-point analyses
twopts <- rf_2pts(simu_example_out)
twopts
```

`sort_by_pos` *Sort markers in onemap object by their position in reference genome*

Description

Sort markers in onemap object by their position in reference genome

Usage

```
sort_by_pos(onemap.obj)
```

Arguments

`onemap.obj`	object of class onemap

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)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

`split_2pts` *Split rf_2pts object by markers*

Description

Split rf_2pts object by markers

Usage

```
split_2pts(twopts.obj, mks)
```

Arguments

`twopts.obj`	object of class rf_2pts
`mks`	markers names (vector of characters) or number (vector of integers) to be removed and added to a new rf_2pts object

Value

An object of class `rf_2pts` with only the selected markers, 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.

Author(s)

Cristiane Taniguti, <`chtaniguti@tamu.edu`>

`split_onemap` *Split onemap data sets*

Description

Receives one onemap object and a vector with markers names to be removed from the input onemap object and inserted in a new one. The output is a list containing the two onemap objects.

Usage

```
split_onemap(onemap.obj = NULL, mks = NULL)
```

Arguments

`onemap.obj`	object of class onemap
`mks`	markers names (vector of characters) or number (vector of integers) to be removed and added to a new onemap object

Value

a list containing in first level the original onemap object without the indicated markers and the second level the new onemap object with only the indicated markers

---

`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

```
suggest_lod(x)
```

Arguments

`x`	an object of class `sequence` or `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
```

`summary_maps_onemap` *Create table with summary information about the linkage map*

Description

Create table with summary information about the linkage map

Usage

```
summary_maps_onemap(map.list, mapping_function = "kosambi")
```

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.
`mapping_function`	either "kosambi" or "haldane"

Value

data.frame with basic summary statistics

Author(s)

Jeekin Lau, `<jeekinlau@gmail.com>`

`test_segregation` *test_segregation*

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, simulate.p.value = FALSE)
```

Arguments

`x`	an object of class `onemap`, with data and additional information.
`simulate.p.value`	a logical indicating whether to compute p-values by Monte Carlo simulation.

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
```

`test_segregation_of_a_marker`
test_segregation_of_a_marker

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, simulate.p.value = FALSE)
```

Arguments

`x`	an object of class `onemap`, with data and additional information.
`marker`	the marker which will be tested for its segregation.
`simulate.p.value`	a logical indicating whether to compute p-values by Monte Carlo simulation.

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

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Jiang, C. and Zeng, Z.-B. (1997). Mapping quantitative trait loci with dominant and missing markers in various crosses from two inbred lines. *Genetica* 101: 47-58.

Lander, E. S., Green, P., Abrahamson, J., Barlow, A., Daly, M. J., Lincoln, S. E. and Newburg, L. (1987) MAPMAKER: An interactive computer package for constructing primary genetic linkage maps of experimental and natural populations. *Genomics* 1: 174-181.

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetic maps. *Heredity* 103: 494-502

Wu, R., Ma, C.-X., Painter, I. and Zeng, Z.-B. (2002a) Simultaneous maximum likelihood estimation of linkage and linkage phases in outcrossing species. *Theoretical Population Biology* 61: 349-363.

Wu, R., Ma, C.-X., Wu, S. S. and Zeng, Z.-B. (2002b). Linkage mapping of sex-specific differences. *Genetical Research* 79: 85-96

See Also

`make_seq` and `compare`.

Examples

```
#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
```

`try_seq_by_seq` *Run try_seq considering previous sequence*

Description

It uses try_seq function repeatedly trying to positioned each marker in a vector of markers into a already ordered sequence. Each marker in the vector `"markers"` is kept in the sequence if the difference of LOD and total group size of the models with and without the marker are below the thresholds `"lod.thr"` and `"cM.thr"`.

Usage

```
try_seq_by_seq(sequence, markers, cM.thr = 10, lod.thr = -10, verbose = TRUE)
```

Arguments

`sequence`	object of class sequence with ordered markers
`markers`	vector of integers defining the marker numbers to be inserted in the `sequence`
`cM.thr`	number defining the threshold for total map size increase when inserting a single marker
`lod.thr`	the difference of LODs between model before and after inserting the marker need to have value higher than the value defined in this argument
`verbose`	A logical, if TRUE it output progress status information.

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.

ug *Unidirectional Growth*

Description

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

Usage

```
ug(
  input.seq,
  LOD = 0,
  max.rf = 0.5,
  tol = 1e-04,
  rm_unlinked = TRUE,
  size = NULL,
  overlap = NULL,
  phase_cores = 1,
  hmm = TRUE,
  parallelization.type = "PSOCK",
  verbose = TRUE
)
```

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.
`rm_unlinked`	When some pair of markers do not follow the linkage criteria, if `TRUE` one of the markers is removed and ug is performed again.
`size`	The center size around which an optimum is to be searched
`overlap`	The desired overlap between batches
`phase_cores`	The number of parallel processes to use when estimating the phase of a marker. (Should be no more than 4)
`hmm`	logical defining if the HMM must be applied to estimate multipoint genetic distances
`parallelization.type`	one of the supported cluster types. This should be either PSOCK (default) or FORK.
`verbose`	A logical, if TRUE it output progress status information.

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`	object of class `onemap` with the raw data.
`twopt`	object of class `rf_2pts` with the 2-point analyses.

Author(s)

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

References

Mollinari, M., Margarido, G. R. A., Vencovsky, R. and Garcia, A. A. F. (2009) Evaluation of algorithms used to order markers on genetics maps. *Heredity* 103: 494-502.

Tan, Y. and Fu, Y. (2006) A novel method for estimating linkage maps. *Genetics* 173: 2383-2390.

See Also

`make_seq`, `map`

Examples

```
#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
```

`vcf2raw` *These functions are defunct and no longer available.*

Description

These functions are defunct and no longer available.

Usage

```
vcf2raw()
```

Value

No return value, called for side effects

`vcf_example_bc` *Data generated from VCF file with biallelic markers from a f2 backcross population*

Description

Simulated biallelic data set for an backcross population

Usage

```
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 chromossome and position informations for each marker. It is included to be used as a 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

```
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 chromossome 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

```
data(vcf_example_f2)

# plot markers informations
plot(vcf_example_f2)
```

`vcf_example_out` *Data generated from VCF file with biallelic markers from a full-sib family derived from two outbred parents*

Description

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

Usage

```
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 chromossome 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

```
data(vcf_example_out)

# plot markers informations
plot(vcf_example_out)
```

`vcf_example_riself` *Data generated from VCF file with biallelic markers from a RIL population produced by selfing*

Description

Simulated biallelic data set for an `ri self` population.

Usage

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

Format

The format is: List of 10 $ geno : num [1:92, 1:25] 3 3 1 3 1 3 3 1 3 1 ... ..- attr(*, "dimnames")=List of 2 .. ..$: chr [1:92] "ID1" "ID3" "ID4" "ID5" ... .$: chr [1:25] "SNP16" "SNP12" "SNP17" "SNP10" ... $ 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" ... $ 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 chromossome and position informations for each marker. It is included to be used as a 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

```
data(vcf_example_riself)
plot(vcf_example_riself)
```

`write_map` *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.

Value

file with genetic map information

Wang S., Basten, C. J. and Zeng Z.-B. (2010) Windows QTL Cartographer 2.5. Department of Statistics, North Carolina State University, Raleigh, NC.

Author(s)

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

References

Broman, K. W., Wu, H., Churchill, G., Sen, S., Yandell, B. (2008) *qtl: Tools for analyzing QTL experiments* R package version 1.09-43

Examples

```
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 ".map" file
    write_map(maps.list, tempfile(fileext = ".map"))

  }
```

`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 chromossome and position of the markers, and number of phenotypes mesured. Third line: Individuals/sample names; Followed lines: marker name, marker type and genotypes. One line for each marker. Final lines: chromossome, position and phenotypes informations. See more about input file format at vignettes.

Usage

```
write_onemap_raw(onemap.obj = NULL, file.name = NULL)
```

Arguments

`onemap.obj`	object of class ‘onemap‘
`file.name`	a character for the onemap raw file name. Default is "out.raw"

Value

a onemap input file

Author(s)

Cristiane Taniguti, `<chtaniguti@tamu.edu>`

See Also

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

Examples

```
data(onemap_example_out)
write_onemap_raw(onemap_example_out, file.name = paste0(tempfile(), ".raw"))
```

Index

- ∗ **IO**
  - combine_onemap, 8
  - read_mapmaker, 71
  - read_onemap, 73
- ∗ **alleles**
  - create_depths_profile, 13
- ∗ **arith**
  - set_map_fun, 92
- ∗ **bins**
  - add_redundants, 5
  - create_data_bins, 12
  - find_bins, 27
- ∗ **datasets**
  - mapmaker_example_f2, 39
  - onemap_example_bc, 48
  - onemap_example_f2, 49
  - onemap_example_out, 50
  - onemap_example_riself, 51
  - simu_example_bc, 93
  - simu_example_f2, 94
  - simu_example_out, 95
  - vcf_example_bc, 106
  - vcf_example_f2, 107
  - vcf_example_out, 108
  - vcf_example_riself, 109
- ∗ **depth**
  - create_depths_profile, 13
- ∗ **dimension**
  - create_data_bins, 12
  - find_bins, 27
- ∗ **misc**
  - group, 28
- ∗ **reduction**
  - create_data_bins, 12
  - find_bins, 27
- ∗ **redundants**
  - add_redundants, 5
- ∗ **rqtl**
  - draw_map, 17
  - draw_map2, 18
  - write_map, 110
- ∗ **utilities**
  - compare, 10
  - make_seq, 35
  - map, 37
  - map_overlapping_batches, 42
  - marker_type, 44
  - order_seq, 54
  - pick_batch_sizes, 58
  - rcd, 69
  - record, 75
  - rf_2pts, 78
  - rf_graph_table, 79
  - ripple_seq, 83
  - seeded_map, 86
  - seriation, 90
  - try_seq, 101
  - ug, 104

acum, 4
add_marker, 4, *20*
add_redundants, 5

Bonferroni_alpha, 6

check_data, 7
check_twopts, 8
combine_onemap, 8, *75*
compare, 10, *36*, *54*, *56*, *84*, *102*
create_data_bins, 12, *28*
create_dataframe_for_plot_outcross, 12
create_depths_profile, 13
create_probs, 15

draw_map, 17
draw_map2, 18
drop_marker, *5*, 19

edit_order_onemap, 20
empty_onemap_obj, 21

export_mappoly_genoprob, 22
export_viewpoly, 22
extract_depth, 23

filter_2pts_gaps, 24
filter_missing, 25
filter_prob, 26
find_bins, *6*, *13*, 27

generate_overlapping_batches, 28
group, 28, *31*
group_seq, 30
group_upgma, 32

haldane, 33, *92*

keep_only_selected_mks, 34
kosambi, 34, *92*

load_onemap_sequences, 35

make_seq, *11*, *16*, *29*, *31*, 35, *39*, *45*, *56*, *71*, *77*, *84*, *87*, *89*, *91*, *102*, *105*
map, *36*, 37, *43*, *70*, *71*, *76*, *77*, *91*, *105*
map_avoid_unlinked, 40
map_overlapping_batches, 42, *58*
map_save_ram, 43
mapmaker_example_f2, 39
marker_type, *11*, 44, *73*
mds_onemap, 46

onemap_example_bc, 48
onemap_example_f2, 49
onemap_example_out, 50
onemap_example_riself, 51
onemap_read_vcfR, *15*, 52
ord_by_geno, 56
order_seq, *36*, 54, *84*

parents_haplotypes, 57
pick_batch_sizes, *43*, 58
plot.onemap, 59
plot.onemap_progeny_haplotypes, 60
plot.onemap_progeny_haplotypes_counts, 61
plot.onemap_segreg_test, 62
plot_by_segreg_type, 63
plot_genome_vs_cm, 64
print.compare, 64
print.onemap, 65
print.onemap_bin, 65
print.onemap_segreg_test, 66
print.order, 66
print.sequence, 67
progeny_haplotypes, 67
progeny_haplotypes_counts, 68

rcd, 69
read_mapmaker, *9*, *48*, 71, *93–95*
read_onemap, *9*, *48*, *50*, *51*, *72*, 73, *93–95*, *107–109*
record, 75
remove_inds, 77
rf_2pts, *29*, 78
rf_graph_table, 79
rf_snp_filter_onemap, 81
ripple_seq, 83
rm_dupli_mks, 84

save_onemap_sequences, 85
seeded_map, 86
select_segreg, 88
seq_by_type, 89
seriation, 90
set_map_fun, 92
simu_example_bc, 93
simu_example_f2, 94
simu_example_out, 95
sort_by_pos, 96
split_2pts, 97
split_onemap, 97
suggest_lod, 98
summary_maps_onemap, 99

test_segregation, 99
test_segregation_of_a_marker, 100
try_seq, *36*, *56*, *84*, 101
try_seq_by_seq, 103

ug, 104

vcf2raw, 106
vcf_example_bc, 106
vcf_example_f2, 107
vcf_example_out, 108
vcf_example_riself, 109

write_map, 110
write_onemap_raw, 111