HMMancestry

November 13, 2015

Type Package

Title R package using the Forward-Backward algorithm to infer genotypes, recombination hotspots, and gene conversion tracts from low-coverage next-generation sequence data

Version 2.1

Date 2015-11-14

Author Tyler D. Hether

Maintainer Tyler D. Hether <tyler.hether@gmail.com>

Description to be added

License GPL (>= 2)

LazyLoad yes

Repository http://github.com/tylerhether/HMMancestry

Date/Publication 2012-08-16 13:47:50

Depends plyr (>= 1.8.3), lattice (>= 0.20-33), reshape (>= 0.8.5), reshape2 (>= 1.4.1)

Imports Rcpp (>= 0.11.3)

LinkingTo Rcpp

R topics documented:

`est_maxLnL` *Numerically Estimating the Maximum LnL parameter values from the data*

Description

Infers the genome-wide recombination rate (Morgans / bp) and the assignment probabilty directly from the data using maximum likelihood estimates.

Usage

```
est_maxLnL(dat, ploidy = "diploid", initial_p_assign = "NULL",
  initial_scale = "NULL", tolerance = 0.001, n_coarse_steps = 5,
  n_iterations = 30, plot = FALSE)
```

Arguments

`dat`	a data.frame with five columns: • `Ind` The individual ID • `Chr` The chromosome ID • `Snp` The (sorted) snp location (in bps) • `p0` The number of reads that mapped to parent 0 • `p1` The number of reads that mapped to parent 1 Note that the names of the columns can be different and there can be additional columns to the right (`est_maxLnL` only uses the first 5 columns of the data.frame).
`ploidy`	The ploidy to analyze. If `ploidy="diploid"` (default) then `fb_diploid` is used to infer states and estimate likelihood. If `ploidy="haploid"` then `fb_haploid` will be used.
`initial_p_assign`	The initial assignment probability (see details). If "NULL" (default), then a coarse search will be performed with `n_coarse_steps` equally spaced from 0.95 to 0.999.
`initial_scale`	The initial genome-wide recombination rate (Morgans / bp). If "NULL" (default), then a coarse search will be performed `n_coarse_steps` equally spaced from 1e-06 to 1e-04.
`tolerance`	The tolerance used to stop the search.
`n_coarse_steps`	The size of the 1D (if either `initial_p_assign` or `initial_scale` is "NULL") or 2D grid (if both are "NULL"). Increasing `n_coarse_steps` can greatly increase computation time.
`n_iterations`	The number of iterations during the fine scale parameter estimate serach (see details).

Details

`est_maxLnL` has a coarse and fine scale method to estimate the genome-wide recombination rate, $\hat{c}$, and assignement probability, $\hat{p}$. The coarse scale uses either `fb_haploid` or `fb_diploid` output and estimates the natural log-likelihood, LnL, of the data across a coarse (in parameter space) grid of varying p or c values and the fine scale uses a two-variable Newton-Raphson method to hone in on a closer estimate. For further details see Hether et al. (in prep).

Value

A data.frame containing the following columns:

- p_assign_hat An estimate of the assignment probability
- scale_hat An estimate of mean recombination rate
- n_iterations The number of iterations carried out

Author(s)

Tyler D. Hether

References

Hether, T.D., C. G. Wiench1, and P.A. Hohenlohe (in review). 2015. Novel molecular and analytical tools for efficient estimation of rates of meiotic crossover, non-crossover and gene conversion

P. Deuflhard, Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms. Springer Series in Computational Mathematics, Vol. 35. Springer, Berlin, 2004.

See Also

fb_haploid, fb_haploid

Examples

```
# Simulating 30 haploid recombinants
set.seed(1234567)        # For reproducibility
n_spores <- 30           # number of recombinants
l <- 1000                # number of snps to simulate
c <- 1e-06               # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*1e2       # snps are evenly spaced 100bps apart
p_a <- 0.985             # assignment probability
coverage <- 1            # mean coverage
# Now simulate
sim <- sim_en_masse(n.spores=n_spores, scale=c, snps=snps,
p.assign=p_a, mu.rate=0, f.cross=1, f.convert=0,
length.conversion=0, coverage=coverage)
# Now estmate params
res <- est_maxLnL(sim, ploidy="haploid", plot=TRUE)
res
```

---

fb_diploid *Inferring hidden ancestry states from diploids*

Description

Uses the forward-backward algorithm to estimate ancestry along a given chromosome for a given genotyped diploid.

Usage

```
fb_diploid(snp_locations, p0, p1, p_assign, scale)
```

Arguments

snp_locations	a numeric vector specifying the locations of each snp (in bps). This vector is assumed to be ordered (sorted from smallest to largest snp).
p0	a vector specifying the number of reads that mapped to parent 0. p0 is assumed to be in the smae order as snp_locations.
p1	a vector specifying the number of reads that mapped to parent 1. p1 is assumed to be in the smae order as snp_locations.
p_assign	a value specifying the assignment probabilty (see details).
scale	a numeric specifying the the genome-wide recombination rate (Morgans / bp). scale is assumed to be between 0 and 1 but in practice it is usually quite small.

Details

This is an extension of fb_haploid with three hidden states: two homozygous states and a heterozygous state.

Value

a dataframe with the following columns:

- snp_locations
- p0
- p1
- The emission matrix (3 columns) specifying the emmision probability of belong to the only parent 0 (emiss.1), both parents (emiss.1), and only parent 2 (emiss.3)
- the forward probability matrix (3 columns) giving the scaled forward probabilities of belong to the only parent 0 (forward.1), both parents (forward.2), and only parent 2 (forward.3)
- the backward probability matrix (3 columns) giving the scaled backward probabilities of belong to the only parent 0 (backward.1), both parents (backward.2), and only parent 2 (backward.3)
- Fscale The forward scaling factor for each snp
- Bscale The backward scaling factor for each snp
- posterior probability matrix (3 columns) posterior probabilities for belong to each parental state
- states_inferred a vector of length snp_locations giving the state with the highest posterior.
- lnL a numeric giving the total likelihood of the data (see fb_haploid).

Author(s)

Tyler D. Hether

References

Hether, T.D., C. G. Wiench1, and P.A. Hohenlohe (in review). 2015. Novel molecular and analytical tools for efficient estimation of rates of meiotic crossover, non-crossover and gene conversion

Drubin, R. S. Eddy, A. Krogh, and G. Mitchison. 1998. Biological Sequence Analysis: Probabilistic Models of proteins and nucleic acids. Cambridge University Press, Cambridge CB2 8RU, UK.

See Also

fb_haploid

Examples

```
set.seed(1234567)        # For reproducibility
n_spores <- 1            # number of spores
l <- 75                  # number of snps to simulate
c <- 3.5e-05             # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*1.3e4     # snps are evenly spaced 20kbp apart
p_a <- 0.95              # assignment probability
coverage <- 2.1          # mean coverage
# Now simulate two haploids
sim1 <- sim_en_masse(n.spores=n_spores, scale=c, snps=snps,
 p.assign=p_a, mu.rate=0, f.cross=0.8, f.convert=0.3,
 length.conversion=2e3, coverage=coverage)
sim2 <- sim_en_masse(n.spores=n_spores, scale=c, snps=snps,
 p.assign=p_a, mu.rate=0, f.cross=0.8, f.convert=0.3,
 length.conversion=2e3, coverage=coverage)
# Now merge the two haploids to make a diploid
p0 <- sim1$p0+sim2$p0
p1 <- sim1$p1+sim2$p1
res <- fb_diploid(snp_locations=sim1$Snp, p0=p0, p1=p1, p_assign=p_a, scale=c)
res
```

`fb_haploid` *Inferring hidden ancestry states from haploids*

Description

Uses the forward-backward algorithm to estimate ancestry along a given chromosome for a given genotyped tetrad or simulated data.

Usage

```
fb_haploid(snp_locations, p0, p1, p_assign, scale)
```

Arguments

`snp_locations`	a numeric vector specifying the locations of each snp (in bps). This vector is assumed to be ordered (sorted from smallest to largest snp).
`p0`	a vector specifying the number of reads that mapped to parent 0. p0 is assumed to be in the smae order as snp_locations.
`p1`	a vector specifying the number of reads that mapped to parent 1. p1 is assumed to be in the smae order as snp_locations.
`p_assign`	a value specifying the assignment probabilty (see details).
`scale`	a numeric specifying the the genome-wide recombination rate (Morgans / bp). `scale` is assumed to be between 0 and 1 but in practice it is usually quite small.

Details

The function `fb_haploid` attempts to estimate parental genotypic 'states' along a chromosome given empirical or simulated F2 cross data. Next-generation data inherits both sequencing error and missing data – especially when sequencing coverage is low. Also, parental populations can be polymorphic with respect to gene frequencies before admixture. In these cases the parental state is ambiguous or unknown. `fb_haploid` takes these uncertainties into account and estimates the most likely sequence of ancestry.

The three steps taken in `fb_haploid` are:

1. Calculate forward probabilities for each state (5' to 3')
2. Calculate backward probabilities for each state (3' to 5')
3. From these two probabilities, calculate the posterior probability that a snp location is of a given parental state.

The two element vector of forward probabilities, f_i, for each parental state at each position i is calculated as:

$$f_i = \mathbf{e_i}\ \mathbf{T_i}\ \mathbf{f_{i-1}}$$

where e_i is the emission probabilities for each state (see below), T_i is a 2-by-2 matrix describing the transition (recombination) probabilities between two states, and f_{i-1} is the forward probability at the previous position along the chromosome (5' of position i). It is assumed that each state is equally likely to occur at the first position.

The emmission probabilities are calculated independently for each snp and depends on the sequence reads assigned to parent "0" and parent "1" and `p_assign`. The emission probabilities are calculated using the binomial equation. For example, the emission probability for parental state "0" at snp position i is:

$$e_i^{(0)} = \binom{n}{k_0} p^{k0} (1-p)^{n-k_0}$$

where n is the sum of the number of reads from both parents at snp i and p is the assignment probability, `p_assign`.

Like the emission probabilities, the transition probabilities are also calculated for each snp position. The transition probabilities in matrix $\mathbf{T_i}$ are calcuated from the genome-wide recombination rate, `scale`, and the physical distance (in bps) between the two snps. Specifically, Haldane's function is used to estimate the probabilty of getting an odd number of crossovers between snp i and snp $i-1$. For more details, see Hether et al. (in prep).

To avoid underflow, we rescale the forward (and backward) probabilities each iteration to that they sum to unity.

The backward probabilities are calculated similarly but in the 3' to 5' direction. It is assumed that the backward probability is equally likely in each state. Following Durbin et al. (1998) the backward probability at snp position i is:

$$b_i = \mathbf{T_i}\ \mathbf{e_{i+1}}\ \mathbf{b_{i+1}}$$

Again, these probabilities are rescaled to avoid underflow.

The posterior probability that the state at position i is k given the observed sequence read counts for each parent at position i, x_i is calculated by:

$$P(\pi_i = k \mid x_i) = \frac{f_i b_i}{\sum_{k=0}^{1} f_i^{(k)} b_i^{(k)}}$$

Ties in posterior probabilities are rare with sufficient coverage and/or signal. However, in the event of a tie state 1 is picked.

Finally, we can determine the log likelihood of the whole sequence of observations by summing up the log of all the scale factors in the forward probability calculation.

Value

a dataframe with the following columns:

- `snp_locations`
- `p0`
- `p1`
- The emission matrix (2 columns) specifying the emmision probability of belong to the parent 0 (emiss.1) and parent 2 (emiss.2)
- the forward probability matrix (2 columns) giving the scaled forward probabilities of belong to the parent 0 (forward.1) and parent 2 (forward.2)
- the backward probability matrix (2 columns) giving the scaled backward probabilities of belong to the parent 0 (backward.1) and parent 2 (backward.2)
- `Fscale` The forward scaling factor for each snp
- `Bscale` The backward scaling factor for each snp
- posterior probability matrix (2 columns) posterior probabilities for belong to each parental state
- states_inferred a vector of length snp_locations giving the state with the highest posterior.
- lnL a numeric giving the total likelihood of the data (see details).

Author(s)

Tyler D. Hether

References

Hether, T.D., C. G. Wiench1, and P.A. Hohenlohe (in review). 2015. Novel molecular and analytical tools for efficient estimation of rates of meiotic crossover, non-crossover and gene conversion

Drubin, R. S. Eddy, A. Krogh, and G. Mitchison. 1998. Biological Sequence Analysis: Probabilistic Models of proteins and nucleic acids. Cambridge University Press, Cambridge CB2 8RU, UK.

See Also

`fb_diploid`

Examples

```
set.seed(1234567)        # For reproducibility
n_spores <- 1            # number of spores
l <- 75                  # number of snps to simulate
c <- 3.5e-05             # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*1.3e4     # snps are evenly spaced 20kbp apart
p_a <- 0.95              # assignment probability
coverage <- 2.1          # mean coverage
# Now simulate
sim1 <- sim_en_masse(n.spores=n_spores, scale=c, snps=snps,
 p.assign=p_a, mu.rate=0, f.cross=0.8, f.convert=0.3,
 length.conversion=2e3, coverage=coverage)
fb_haploid(snp_locations=sim1$Snp, p0=sim1$p0, p1=sim1$p1, p_assign=p_a, scale=c)
```

---

id_recombination_events

Identify crossover points from state sequences and snp locations

---

Description

This is a simple function that takes a vector of parental states along a chromosome and identifies where states have changed.

Usage

```
id_recombination_events(snps_genotypes_df)
```

Arguments

`snps_genotypes_df`
: a data.frame with two columns:
  - snps a vector of ordered snp locations (in bps)
  - states a vector of corresponding inferred states, either haploid (2 states = 0 or 1) or diploid with three states possible (0,1,2)

Value

a data.frame containing the midpoint between focal snp i and snp $i-1$ and whether their states were the same (0) or different (1).

Author(s)

Tyler D. Hether

See Also

fb_haploid, fb_diploid

Examples

```
df <- data.frame(snps=100*(1:10),
   states=c(rep(0,4), rep(1,6)))
id_recombination_events(df)
```

`infer_tracts` *Identify crossover, non-crossover, and gene conversion tracts*

Description

Infers different types of tracts (crossover, non-crossover, and gene conversion) along a chromosome

Usage

```
infer_tracts(dat, threshold_size = 2500)
```

Arguments

`data`	A data.frame with 7 columns: 1. c("Tetrad", "Chr", "Snp", "one", "two", "three", "four") 2. `Tetrad` specifying the tetrad ID 3. `Chr` giving the chromosome name 4. `Snp` a vector of snp locations (in bps) 5. `one` the inferred states for spore 1 6. `two` the inferred states for spore 2 7. `three` the inferred states for spore 3 8. `four` the inferred states for spore 4
`threshold_size`	The size (in bps) of the threshold (see details)

Details

Uses the 3 step classification scheme described in Hether et al. (in review) to identify the location of these specific CO, NCO, and telomeric gene conversion tracts. Specifically, `infer_tracts` attempts to identify the following tracts:

- 2_2 a tract with 2:2 segregation
- COyesGC a region of gene conversion that was associated with a crossover event
- COnoGC a region between a crossover event that did not have a detectable gene conversion
- NCO a non-crossover; a gene conversion tract without crossing
- GC_tel a gene conversion tract located at the chromosome end

Value

A data.frame containing the following columns:

1. type The type of inferred tract
2. start_snp The starting snp position in base pairs
3. end_snp The ending snp position in base pairs
4. extent The size of the tract. For COnoGC, this extent is the s spanning region between flanking CO events.

Author(s)

Tyler D. Hether

References

Hether, T.D., C. G. Wiench1, and P.A. Hohenlohe (in review). 2015. Novel molecular and analytical tools for efficient estimation of rates of meiotic crossover, non-crossover and gene conversion

Examples

```
set.seed(1234567)        # For reproducibility
n_tetrads <- 3           # number of tetrads
l <- 1000                # number of snps to simulate
c <- 3e-05               # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*250       # snps are evenly spaced 250 bp apart
p_a <- 0.95              # assignment probability
coverage <- 1            # mean coverage
# simulate tetrads
tetrad <- sim_tetrad(n.tetrads=n_tetrads, scale=c, snps=snps,
 p.assign=p_a, mu.rate=1e-03, f.cross=0.6, f.convert=0.8,
 length.conversion=2e3, coverage=coverage)
#' # Example 1 -- infer tracts directly from simulated data
inf_tracts_sim <- infer_tracts(tetrad)
inf_tracts_sim
#' # Example 2 -- infer tracts from inferred data
inf_states <- ddply(tetrad, .(Tetrad, Spore, Chr),
    function(x){
    return(fb_haploid(snp_locations=x$Snp, p0=x$p0,
    p1=x$p1, p_assign=p_a, scale=c))})
inf_tracts_inf_states <- infer_tracts(inf_states)
inf_tracts_inf_states
```

`make_parents` *Simulate haploid parental genotypes along a chromosome*

Description

This function creates divergent genotypes between two parents. The first parent's genotypes are denoted by a vector of zeros of length `L` while the second parent's genotypes are a vector of ones.

Usage

```
make_parents(snps)
```

Arguments

`snps`	a vector gving the locations of snps along a chromosome

Value

an object of class `parent.genomes` giving the parental ancestry at each snp.

Author(s)

Tyler D. Hether

References

none.

See Also

recombine_index, recobmine

Examples

```
make_parents(snps=c(1:10)*10)
```

`recombine` *Simulate recombination (crossovers or non-crossovers) along a chromosome*

Description

This function simulates recombination between two parent chromosomes generated from class `parent.genomes`.

Usage

```
recombine(parents, r.index, mu.rate = 0, f.cross = 0.5, f.convert = 0,
  length.conversion = 20)
```

Arguments

`parents`	an object of class `parent.genomes` specifying the parental ancestry at each simulated snp.
`r.index`	a vector of length `snps-1` specifying whether a recombination is to be simulated (1) or not (0) in between two adjacent snps.
`mu.rate`	a numeric between 0 and 1 (inclusive) specifying the per snp mutation rate.
`f.cross`	a numeric between 0 and 1 (inclusive) giving the frequency of recombination events that result in crossing over. This is same as 1 minus the frequency of non-crossovers.
`f.convert`	a numeric between 0 and 1 (inclusive) that gives the frequency of gene conversion during recombination.
`length.conversion`	an integer specifying the mean (and variance) of a given gene conversion tract (in bps).

Details

This function simulates crossover events (with or without gene conversions) and non-crossover events given the parental snps and location of recombination events. First, parental chromosomes are duplicated into 2 pairs of sister chromatids. During this step mutations occur at rate `mu.rate` that switch parental alleles. Second, recombination is simulated by systematically going down the length of the chromosome. At recombination points, idenitfied in `r.index`, two non-sister chromatids are picked at random. Third, a CO or NCO is simulated; this is done at a frequency of f.cross or 1-f.cross, respectively. If a CO occurs, the genotypes of the 3' end of the non-sister chromatids are

simply switched. If a NCO occurs, no such switch takes place. Note that is is impossible to detect NCO events without an associated gene conversion tract. Following every CO or NCO event gene conversion is simulated at a rate of f.convert. During a gene conversion, one of the two unpicked chromatids' genotypes are switched. The length of the gene conversion tract is sampled from a poission distribution with mean (variance) of length.conversion. Note that the gene conversion tract will stop at the end of the chromosme if necessary.

Value

an object of class `recombine` that contains the following data:

- parents input data
- r.index input data
- m.rate input data
- f.cross input data
- f.convert input data
- length.conversion input data
- chromatids.mutated_snps a list giving the genotypes of 4 chromatids (two pairs of sister chromatids) following mutation
- chromatids.recombined a list giving the genotypes of 4 chromatids following recombination

Author(s)

Tyler D. Hether

See Also

`recombine_index`

Examples

```
set.seed(1234567)        # For reproducibility
l <- 50                  # number of snps to simulate
c <- 3e-05               # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*1e4       # snps are evenly spaced 10kbp apart
p <- make_parents(snps)  # make the parents
#
# The recombination indeces are:
r_points <- recombine_index(scale=c, snps=snps)
#
# Now recombine the parents:
recomb_sim <- recombine(parents=p, r.index=r_points, mu.rate=1e-05,
f.cross=0.6, f.convert=0.2, length.conversion=15000)
recomb_sim
```

recombine_index *Simulate recombination points along a chromosome.*

Description

Simulate recombination events given the vector of snp locations and recombination rate(s). Haldane's function is used to simulate the probabilty of getting an odd number of crossover events between any two snps given their physical distance and the specified recombination rate.

Usage

```
recombine_index(scale, snps)
```

Arguments

scale	either vector of length 1 specifying the genome-wide recombination rate (Morgans/bp) or a vector of length snps-1 specifying the recombination rate between all snps. In either case rates must be positive.
snps	a vector gving the locations of snps along a chromosome

Value

a vector of length snps-1 specifying the location of recombiantion points.

Author(s)

Tyler D. Hether

See Also

make_parents

Examples

```
set.seed(1234567)
# Simulating recombination across 1000 loci randomly spaced
# along a 200kbp chromosome
chromSize <- 2e5  # Chromosome size
l <- 1e3  # number of loci to simulate
c <- 1e-04  # Morgans/bp
snps <- sort(sample(size=l, 1:chromSize, replace=FALSE))
# Find recombination points between all the snps
recomb_points <- recombine_index(scale=c, snps=snps)
recomb_points
```

`simulate_coverage` *Simulate sequencing across simulated data:*

Description

This function populates simulated data of class `recombine` with read counts given a specified coverage and assignment probability.

Usage

```
simulate_coverage(simdata, p.assign, coverage)
```

Arguments

`simdata`	an object of class `recombine`
`p.assign`	a numeric between 0 and 1 (inclusive) that gives the probability of correct sequencing assignment. A value of 1 means that no sequencing error or ancestral polymorphism is to be simulated.
`coverage`	the mean (and variance) of sequencing read coverage to be simlated. `coverage` is sampled from a poisson distribution.

Value

an object of class `simulated.coverage` providing the a list of simulated sequencing reads for each parental type for each of the four haploid recombinants. There are four lists in the output – one for each recombinant. Within each list the read counts for parent 0 and parent 1 are given as well as the snp locations and given (simulated) states.

Author(s)

Tyler D. Hether

See Also

`recombine`

Examples

```
set.seed(1234567)        # For reproducibility
l <- 50                  # number of snps to simulate
c <- 3e-05               # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*1e4       # snps are evenly spaced 10kbp apart
p_a <- 0.97              # assignment probability
coverage <- 1            # mean coverage
p <- make_parents(snps)  # make the parents
#
# The recombination indeces are:
r_points <- recombine_index(scale=c, snps=snps)
#
# Now recombine the parents:
recomb_sim <- recombine(parents=p, r.index=r_points, mu.rate=1e-05,
f.cross=0.6, f.convert=0.2, length.conversion=15000)
```

```
# And finally simulated read coverage for each haploid recombinant
sim_cov <- simulate_coverage(simdata=recomb_sim,
 p.assign=p_a, coverage=coverage)
sim_cov
```

`sim_en_masse` *Simulate random spores en masse*

Description

This is a wrapper function of many other functions that simulates a given number of haploids each recombinant between two parents. As the name implies, information of the all four products are lost since spores are random. See `\line{sim_tetrad}` if all four haploid recombinants are desired.

Usage

```
sim_en_masse(n.spores, scale, snps, p.assign, mu.rate, f.cross, f.convert,
  length.conversion, coverage, chr.name = "I")
```

Arguments

`n.spores`	An integer specifying the number of spores to simulate.
`p.assign`	a numeric between 0 and 1 (inclusive) that gives the probability of correct sequencing assignment. A value of 1 means that no sequencing error or ancestral polymorphism is to be simulated.
`mu.rate`	a numeric between 0 and 1 (inclusive) specifying the per snp mutation rate.
`f.cross`	a numeric between 0 and 1 (inclusive) giving the frequency of recombination events that result in crossing over. This is same as 1 minus the frequency of non-crossovers.
`f.convert`	a numeric between 0 and 1 (inclusive) that gives the frequency of gene conversion during recombination.
`length.conversion`	an integer specifying the mean (and variance) of a given gene conversion tract (in bps).
`chr.name`	a numeric or character value specifying the chromosome name (default to "I")
`scale`	either vector of length 1 specifying the genome-wide recombination rate (Morgans/bp) or a vector of length `snps-1` specifying the recombination rate between all snps. In either case rates must be positive.
`snps`	a vector gving the locations of snps along a chromosome
`r.index`	a vector of length `snps-1` specifying whether a recombination is to be simulated (1) or not (0) in between two adjacent snps.

Value

A data.frame of class `en.masse`. Each row contains the following information: which contains three elements:

- Spore The spore ID (1:length(n.spores))
- Chr The chromosome name

- Snp The snp location (in bps)
- p0 The number of reads that mapped to parent 0
- p1 The number of reads that mapped to parent 1
- states_given The simulated states (0 or 1)

Author(s)

Tyler D. Hether

See Also

recombine, make_parents, simulate_coverage, recombine_index, id_hotspots

Examples

```
# Simulating 5 haploid recombinants
set.seed(1234567)        # For reproducibility
n_spores <- 5          # number of tetrads (meiosis events)
l <- 75                  # number of snps to simulate
c <- 3.5e-05             # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*1.3e4     # snps are evenly spaced 20kbp apart
p_a <- 0.95              # assignment probability
coverage <- 2.1          # mean coverage
# Now simulate
sim5 <- sim_en_masse(n.spores=n_spores, scale=c, snps=snps,
p.assign=p_a, mu.rate=0, f.cross=0.8, f.convert=0.3,
length.conversion=2e3, coverage=coverage)
sim5
```

---

sim_tetrad *Simulate recombination across a given number of meiosis events*

Description

This is a wrapper function of many other functions that simulates a given number of tetrads each with four haploid spores that are recombinants between two parents.

Usage

```
sim_tetrad(n.tetrads, scale, snps, p.assign, mu.rate, f.cross, f.convert,
  length.conversion, coverage, chr.name = "I")
```

Arguments

n.tetrads	An integer specifying the number of tetrads to simulate.
p.assign	a numeric between 0 and 1 (inclusive) that gives the probability of correct sequencing assignment. A value of 1 means that no sequencing error or ancestral polymorphism is to be simulated.
mu.rate	a numeric between 0 and 1 (inclusive) specifying the per snp mutation rate.
f.cross	a numeric between 0 and 1 (inclusive) giving the frequency of recombination events that result in crossing over. This is same as 1 minus the frequency of non-crossovers.

f.convert	a numeric between 0 and 1 (inclusive) that gives the frequency of gene conversion during recombination.
length.conversion	an integer specifying the mean (and variance) of a given gene conversion tract (in bps).
chr.name	a numeric or character value specifying the chromosome name (default to "I")
scale	either vector of length 1 specifying the genome-wide recombination rate (Morgans/bp) or a vector of length snps-1 specifying the recombination rate between all snps. In either case rates must be positive.
snps	a vector gving the locations of snps along a chromosome
r.index	a vector of length snps-1 specifying whether a recombination is to be simulated (1) or not (0) in between two adjacent snps.

Value

A data.frame of class tetrad. Each row contains the following information:

- Tetrad The tetrad ID
- Spore The spore ID (1:4)
- Chr The chromosome name
- Snp The snp location (in bps)
- p0 The number of reads that mapped to parent 0
- p1 The number of reads that mapped to parent 1
- states_given The simulated states (0 or 1)

Author(s)

Tyler D. Hether

See Also

recombine, make_parents, simulate_coverage, recombine_index, id_hotspots

Examples

```
# Simulating 100 recombination events
set.seed(1234567)        # For reproducibility
n_tetrads <- 100         # number of tetrads (meiosis events)
l <- 50                  # number of snps to simulate
c <- 3e-05               # recombination rate between snps (Morgan/bp)
snps <- c(1:l)*2e4       # snps are evenly spaced 20kbp apart
p_a <- 0.97              # assignment probability
coverage <- 1            # mean coverage
# Now simulate
sim100 <- sim_tetrad(n.tetrads=n_tetrads, scale=c, snps=snps,
p.assign=p_a, mu.rate=0, f.cross=0.8, f.convert=0.3,
length.conversion=2e3, coverage=coverage)
sim100
```

Index