Package: IBDsim (via r-universe)

March 24, 2025

Type Package

Title Simulation of Chromosomal Regions Shared by Family Members

Version 0.9-8

Date 2019-04-6

Author Magnus Dehli Vigeland

Maintainer Magnus Dehli Vigeland <m.d.vigeland@medisin.uio.no>

Description Simulation of segments shared identical-by-descent (IBD) by pedigree members. Using sex specific recombination rates along the human genome (Kong et. al (2010) <doi:10.1038/nature09525>), phased chromosomes are simulated for all pedigree members, either by unconditional gene dropping or conditional on a specified IBD pattern. Additional functions provide summaries and further analysis of the simulated genomes.

Depends R (>= 3.3), paramlink (>= 1.1)

Imports stats

License GPL (>= 2)

LazyData Yes

NeedsCompilation yes

Date/Publication 2019-04-06 11:42:45 UTC

Additional_repositories https://cranhaven.r-universe.dev

Repository https://cranhaven.r-universe.dev

RemoteUrl https://github.com/cranhaven/cranhaven.r-universe.dev

RemoteRef package/IBDsim

RemoteSha 5ceab74a448847ba586d825e1d1fca03ac7b9309

RemoteSubdir IBDsim

Contents

alleleSummary 2
DecodeMap 3
dominant1 4
IBDsim 4
oneLocusIBD 8
oneLocusJacquard 10
twoLocusIBD 11
twoLocusJacquard 13

Index **15**

`alleleSummary` *Allele sharing summary*

Description

This function summarises the allele flow for selected individuals, for a single genome simulation.

Usage

```
alleleSummary(x, ids, ibd.status = FALSE)
```

Arguments

`x`	A single genome simulation, i.e. a list of simulated chromosomes. Each chromosome is a list, with one entry for each individual. Each of these entries is a list of two matrices (one for each strand). The matrices have 2 columns (start position; allele) and one row for each segment unbroken by recombination.
`ids`	A vector of numerical IDs. If missing, all individuals are included.
`ibd.status`	A logical, meaningful only if `length(ids)==2`. If TRUE the IBD status (number of alleles shared IBD, either 0 1 or 2) of each segment is computed, as well as the breakdown of their parental origin.

Details

This function is useful for downstream analysis of simulations produced by `IBDsim`.

Value

A numerical matrix. Each row corresponds to a chromosomal segment. The first 4 columns describe the segment (chromosome, start, end, length), and are followed by two columns (paternal allele, maternal allele) for each of the selected individuals. If `ibd.status=TRUE` five more columns are added: `ibd`, `ibd_pp`, `ibd_pm`, `ibd_mp` and `ibd_mm`. The first of these indicate the IBD status (0, 1 or 2) in the segment, while the latter 4 give the parental breakdown of this number. For instance, `ibd_pm` is 1 if the _p_aternal allele of the first individual is IBD with the _m_aternal allele of the second individual, and 0 otherwise.

Author(s)

Magnus Dehli Vigeland

See Also

IBDsim

Examples

```
### Sibling simulation (3 sims of chromosomes 1 and 2)
x = nuclearPed(2)
sim = IBDsim(x, sims=3, chromosomes=1:2)

alleleSummary(sim[[1]]) # First sim, summary of all individuals
alleleSummary(sim[[1]], ids=3:4) # Summary of the siblings
alleleSummary(sim[[1]], ids=3:4, ibd.status=TRUE) # IBD breakdown of the siblings

# Trivial example: Summary of the father.
# Being the first founder, his alleles are denoted 1 and 2 everywhere.
fa = alleleSummary(sim[[1]], ids=1)
stopifnot(all(fa[,c('1p', '1m')]==c(1,1,2,2)))
```

DecodeMap — *Decode recombination map*

Description

A recombination map of the human genome, adapted from the dataset published in (Kong et al., 2010).

Usage

```
DecodeMap
```

Format

List of 23 components (one for each chromosome pair). Each is a list of 2 matrices, containing the male and female recombination maps of the given chromosome respectively. The 23rd component has NULL as its first (male) element, as we assume no recombination between sex chromosomes in males. The recombination map matrices have 2 columns, named "Mb" and "cM". The "Mb" column has the base positions of the markers used by (Kong et. al, 2010), and "cM" the genetic distance from the left telomere.

Source

Kong, A. et al. (October 2010) Fine scale recombination rate differences between sexes, populations and individuals. *Nature*, 467, 1099–1103. doi:10.1038/nature09525.

Examples

```
#the first entries of the male map of chromosome 1:
head(DecodeMap[[1]]$male)
```

`dominant1` *Autosomal dominant pedigree*

Description

A three generation pedigree affected with an autosomal dominant disorder.

Usage

```
dominant1
```

Format

A data frame describing the pedigree in standard LINKAGE format (i.e. the columns are: Family ID, individual ID, father, mother, sex, affection status).

Examples

```
x = linkdat(dominant1)
plot(x)
```

`IBDsim` *IBD simulation*

Description

This is the main function of the package. Gene dropping of chromosomes is simulated down the pedigree, either unconditionally or conditional on the 'condition' pattern if this is given. Regions showing the 'query' pattern are collected and summarized.

Usage

```
IBDsim(x, sims, query=NULL, condition=NULL, map="decode",
       chromosomes=NULL, model="chi", merged=TRUE, simdata=NULL,
       skip.recomb = "noninf_founders", seed=NULL, verbose=TRUE)
```

Arguments

`x`	A pedigree in the form of a `linkdat` object.
`sims`	The number of simulations.
`query`, `condition`	Single allele patterns (SAPs), described as lists with numerical entries named "0", "1", "2", "atleast1", "atmost1".
`map`	The genetic map(s) to be used in the simulations: One of the character strings "decode", "uniform.sex.spec", "uniform.sex.aver". (See Details.)
`chromosomes`	A numeric vector indicating chromosome numbers, or either of the words "AUTOSOMAL" or "X". The default is 1:22, i.e. the human autosomes.
`model`	A character indicating the statistical model for recombination: Either "chi" (the default) or "haldane". (See details.)
`merged`	A logical, indicating if overlapping/adjacent regions should be merged or not.
`simdata`	Either NULL, in which case simulation is performed before collecting IBD regions, or an object containing simulation data (typically generated by a previous run of `IBDsim`).
`skip.recomb`	A numeric containing individuals whose meioses should be simulated without recombination (i.e. a random strand is passed on to each offspring). If NULL, nobody is skipped. The default value (the character "noninf_founders") computes the set of pedigree founders that cannot be carriers of the alleles described in the `condition` and/or `query` SAPs.
`seed`	NULL, or an integer to be passed on to `set.seed`).
`verbose`	A logical.

Details

Each simulation starts by unique alleles being distributed to the pedigree founders. In each subsequent meiosis, homologue chromosomes are made to recombine according to a renewal process along the four-strand bundle, with chi square distributed waiting times. (For comparison purposes, Haldane's Poisson model for recombination is also implemented.)

Recombination rates are sex-dependent, and vary along each chromosome according to the recombination map specified by the `map` parameter. By default, the complete Decode map of the human autosome is used (see References). If `map="uniform.sex.spec"`, the genetic chromosome *lengths* are as in the Decode map, but the recombination rate is kept constant along each chromosome. If `map="uniform.sex.aver"`, sex averaged genetic chromosome lengths are used (and constant recombination rates along each chromosome).

IBD patterns are described as combinations of Single Allele Patterns (SAPs). A SAP is a specification for a given allele of the number of copies carried by various individuals, and must be given as a list of numerical vectors (containing ID labels) named '0', '1', '2', 'atleast1' and 'atmost1' (some of these can be absent or NULL; see Examples).

If a condition SAP is given (i.e. if `condition` is non-null), simulation of each complete chromosome set (all autosomes by default) is performed as follows: A 'disease chromosome' is sampled at random (using the physical chromosome lengths as weights), followed by a random 'disease locus' on this chromosome. For this chromosome, gene dropping down the pedigree is carried out in

such a way that the 'disease locus' has the condition SAP. (In a bit more detail: First, the program computes all possible sets of obligate carriers, with suitable weights, and samples one of these. Included in the obligate carriers will be exactly one founder, one of whose alleles is taken as the 'disease allele'. In each meiosis involving obligate carriers, recombination is performed as usual, but the strand carrying the 'disease allele' is always the one passed on.) For the other chromosomes, simulation is done unconditionally.

Value

If the `query` is NULL, no identification of IBD segments is done, and only the simulated genomes are (invisibly) returned.

If `query` is non-null, the segments with the query SAP are identified, and a list of three elements is returned. These are the 'simdata' (the simulated genomes), the 'segments' (a list of matrices describing all identified regions) and finally 'stats' (a matrix with one column per simulation, summarizing the regions from that genome). A summary is printed on the screen, with some or all of the following slots:

`count.all`	The average count of all IBD segments (i.e. counting both random regions and the disease region in case of conditional simulation).
`fraction.all`	The average fraction (in %) of the genome covered by IBD segments.
`average.all`	The average length (in Mb) of IBD segments.
`longest.all`	The average length (in Mb) of the longest IBD segment from each simulated genome.
`count.rand`	The average count of random IBD segments.
`fraction.rand`	The average fraction (in %) of the chromosomes covered by random IBD segments.
`average.rand`	The average length (in Mb) of random IBD segments.
`longest.rand`	The average length (in Mb) of the longest random IBD segment from each simulated genome.
`length.dis`	The average length of the disease segment (only with conditional simulation).
`rank.dis`	The average rank of the disease segment length among all the segments.
`zeroprob`	The percentage of simulations resulting in zero IBD segments.

The term 'IBD segment' in the above always refers to 'IBD segment matching the query SAP'. The suffixes `.dis`, `.rand` and `.all` points to respectively the *disease* IBD segments (only relevant in conditional simulations), the *random*, i.e. non-disease, IBD segments (only relevant in conditional simulations), and *all* IBD segments (in any simulation).

Author(s)

Magnus Dehli Vigeland

References

The Decode map:

Kong, A. et al. (2010) Fine scale recombination rate differences between sexes, populations and individuals. *Nature*, 467, 1099–1103. doi:10.1038/nature09525.

The chi-square model:

Zhao, H., Speed, T. P., McPeek, M. S. (1995) Statistical analysis of crossover interference using the chi-square model. *Genetics*, 139(2), 1045–1056.

Examples

```
# In all examples below, the 'sims' parameter is set to 1 to
# decrease runtime. For realistic results increase to e.g. 1000.

#### An example with a recessive disease in a consanguineous pedigree:

x = linkdat(twoloops)
plot(x)

# If individuals 15, 16 and 17 are available for sequencing, we can
# predict the number and size of disease-compatible IBD segments as
# follows:

sap1 = list('2'=15:16, 'atmost1'=17)
IBDsim(x, sims=1, condition=sap1, query=sap1)

# If 16 is unavailable, his parents and healthy sibling are still
# informative. The regions we are looking for now are those with
# an allele present in 2 copies in 15, 1 copy in 12 and 14, and
# at most one in 17. Note that the condition SAP remains as above.

IBDsim(x, sims=1, condition=sap1,
       query=list('2'=15, '1'=c(12,14), 'atmost1'=17))


#### Example with an autosomal dominant disorder:
y = linkdat(dominant1) # lazy load data
plot(y)

# Suppose a 20 Mb linkage peak is found.
# How often would this occur by chance?

aff = which(dominant1[,'AFF']==2)   # the affected
nonaff = which(dominant1[,'AFF']==1)   # the non-affected
dom_pattern = list('1'=aff, '0'=nonaff)
res = IBDsim(y, sims=1, query=dom_pattern)
mean(res$stats['longest.all',] > 20)   # the estimated p-value


#### Another example: Unconditional simulation of regions shared
# IBD by third cousins. The "zeroprob" entry in the output shows
# the percentage of simulations having no IBD-sharing among the
```

```
# two cousins. (Again: Increase 'sims' for more accurate results.)

x_male = cousinPed(3)
plot(x_male)
IBDsim(x_male, sims=1, query=list('1'=15:16))

# Changing the genders of the individuals connecting the cousins
# can have a big impact on the distribution of IBD segments:

x_female = swapSex(x_male, c(3,4,7,8,11,12))
plot(x_female)
IBDsim(x_female, sims=1, query=list('1'=15:16))

## Given that the two third cousins have at least one segment in
# common, what is the expected length of this segment? We simulate
# conditional on one allele in common between the cousins. The
# "length.dis" entry of the summary shows the average length of
# the disease region (which should be quite a lot larger than
# an average random segment).

sap = list('1'=15:16)
IBDsim(x_male, sims=1, condition=sap, query=sap)


#### Let us look at a different relationship: Half first cousins.
# Two such cousins will on average share 1/8 = 12.5% of the autosome.

z = halfCousinPed(1)
plot(z)
res = IBDsim(z, sims=1, query=list('1'=8:9))
res$stats

# visualizing the spread in total IBD sharing and the number of segments:

IBD_percent = res$stats['fraction.all', ]
IBD_count = res$stats['count.all', ]
hist(IBD_percent)
hist(IBD_count)
```

`oneLocusIBD` *Estimating pairwise IBD coefficients*

Description

Estimates by simulation the IBD coefficients of a non-inbred pairwise relationship.

Usage

```
oneLocusIBD(x, ind1, ind2, Nsim, Xchrom=FALSE, verbose=TRUE, ...)
```

Arguments

`x`	A pedigree in the form of a `linkdat` object.
`ind1, ind2`	Numeric ID labels of the two individuals.
`Nsim`	The number of simulations to be performed.
`Xchrom`	A logical indicating if the locus is X-linked (if TRUE) or autosomal (FALSE).
`verbose`	A logical.
`...`	Further arguments to be passed on to `IBDsim`.

Details

For any pair of non-inbred individuals, the IBD coefficients $\kappa = (\kappa[0], \kappa[1], \kappa[2])$ associated with the relationship, are defined as the probabilities

$$\kappa[i] = Pr(iallelessharedidenticallybydescent).$$

For an X-chromosomal locus, and if at least one of the individuals is male, $\kappa[i]$ is defined only for $i = 0, 1$.

Value

A numeric of length 3 (autosomal) or 2 (X-linked), estimating κ.

Author(s)

Magnus Dehli Vigeland

See Also

`twoLocusIBD`, `oneLocusJacquard`, `twoLocusJacquard`

Examples

```
### Example 1: Full siblings
x <- nuclearPed(2)
Nsim <- 100 # Should be increased substantially

# Autosomal kappa estimate (exact = c(0.25, 0.5, 0.25))
oneLocusIBD(x, ind1=3, ind2=4, Nsim=Nsim)

# X-chromosomal kappa estimate (exact = c(0.5, 0.5))
oneLocusIBD(x, ind1=3, ind2=4, Nsim=Nsim, Xchrom=TRUE)
```

`oneLocusJacquard` *Estimating Jacquard's condensed identity coefficients*

Description

Estimates by simulation Jacquard's 9 condensed identity coefficients for a pairwise relationship. This function is rarely needed, as exact values can be obtained by using `jacquard`.

Usage

```
oneLocusJacquard(x, ind1, ind2, Nsim, verbose=TRUE,...)
```

Arguments

`x`	A pedigree in the form of a `linkdat` object.
`ind1`, `ind2`	Numeric ID labels of the two individuals.
`Nsim`	The number of simulations to be performed.
`verbose`	A logical.
`...`	Further arguments to be passed on to `IBDsim`.

Details

For the definition and further details about these coefficients, see Jacquard (1970).

Value

A numeric of length 9, estimating the condensed Jacquard identity coefficients Δ.

Author(s)

Magnus Dehli Vigeland

See Also

`jacquard`, `oneLocusIBD`, `twoLocusIBD`, `twoLocusJacquard`

Examples

```
### Siblings whose parents are full siblings.
x = fullSibMating(generations=2)
Nsim = 100 # (increase to improve accuracy)

# Estimating the 9 identity coefficients
j_est = oneLocusJacquard(x, ind1=5, ind2=6, Nsim=Nsim)

# Exact: c(2,1,4,1,4,1,7,10,2)/32

# With the "identity" package:
```

```
## Not run:
  j_exact = jacquard(x, 5:6)

## End(Not run)
```

---

twoLocusIBD *Estimating two-locus IBD coefficients*

Description

Estimates by simulation the two-locus IBD coefficients of a non-inbred pairwise relationship.

Usage

```
twoLocusIBD(x, ind1, ind2, rho=NULL, cM=NULL, Nsim, Xchrom=FALSE, verbose=TRUE, ...)
```

Arguments

x	A pedigree in the form of a linkdat object.
ind1, ind2	Numeric ID labels of the two individuals.
rho	NULL, or a number in the interval [0, 0.5]: the recombination fraction between the two loci. If non-NULL, it is converted to centiMorgan using Haldanes map function: `cM = -50*log(1-2*rho)`.
cM	NULL, or a non-negative number: the distance in centiMorgan between the two loci.
Nsim	The number of simulations to be performed.
Xchrom	A logical indicating if the loci are X-linked (if TRUE) or autosomal (FALSE).
verbose	A logical.
...	Further arguments to be passed on to IBDsim.

Details

For a given pair of autosomal loci, the two-locus IBD coefficients of a non-inbred pairwise relationship are defined by

$$\kappa[i,j] = Pr(sharingiallelesIBDatlocus1, andjallelesIBDatlocus2),$$

where $0 \leq i, j \leq 2$. For X-chromosomal loci, and if at least one of the individuals is male, $\kappa[i,j]$ is defined only for $0 \leq i, j \leq 1$.

While the single-locus IBD coefficients $\kappa = (\kappa[0], \kappa[1], \kappa[2])$ (see oneLocusIBD) depend only on the genealogy relating the two individuals, the two-locus coefficients also depend on the genetic distance between the loci. In particular, if the loci are completely linked (rho=0; cM=0) we have $\kappa[i,j] = \kappa[i]$ if i=j and 0 otherwise. If the loci are unlinked (rho=0.5; cM=Inf) we have $\kappa[i,j] = \kappa[i] * \kappa[j]$. (See examples.)

Value

A numerical matrix, where the entry in row a and column b is the estimate of $\kappa[a-1, b-1]$ defined above.

Author(s)

Magnus Dehli Vigeland

See Also

oneLocusIBD, oneLocusJacquard, twoLocusJacquard

Examples

```
### Example 1: Full siblings
x <- nuclearPed(2)
Nsim <- 100 # Should be increased substantially

# One-locus kappa estimates (autosomal and X):
k_est = oneLocusIBD(x, ind1=3, ind2=4, Nsim=Nsim, seed=123)
k_est_X = oneLocusIBD(x, ind1=3, ind2=4, Nsim=Nsim, Xchrom=TRUE, seed=123)

### Two-locus IBD estimation
# Completely linked, autosomal
rho = 0
k2_linked = twoLocusIBD(x, ind1=3, ind2=4, rho=rho, Nsim=Nsim, seed=123)
stopifnot(identical(diag(k2_linked), k_est))

# Completely unlinked, autosomal
rho = 0.5
k2_unlinked = twoLocusIBD(x, ind1=3, ind2=4, rho=rho, Nsim=Nsim, seed=123)
stopifnot(identical(k2_unlinked, outer(k_est, k_est)))

# Recombination rate 10%, autosomal
rho <- 0.1
cM <- -50*log(1-2*rho)
r1 <- twoLocusIBD(x, ind1=3, ind2=4, rho=rho, Nsim=Nsim, seed=17)
r2 <- twoLocusIBD(x, ind1=3, ind2=4, cM=cM, Nsim=Nsim, seed=17)
stopifnot(identical(r1, r2))

### Two-locus IBD on X
# Completely linked, X chromosome
rho = 0
k2_linked_X = twoLocusIBD(x, ind1=3, ind2=4, rho=rho, Nsim=Nsim, Xchrom=TRUE, seed=123)
stopifnot(identical(diag(k2_linked_X), k_est_X))

# Completely unlinked, X chromosome
rho = 0.5
k2_unlinked_X = twoLocusIBD(x, ind1=3, ind2=4, rho=rho, Nsim=Nsim, Xchrom=TRUE, seed=123)
stopifnot(identical(k2_unlinked_X, outer(k_est_X, k_est_X)))

# Recombination rate 10%, X chromosome
```

```
rho <- 0.1
cM <- -50*log(1-2*rho)
r1_X <- twoLocusIBD(x, ind1=3, ind2=4, rho=rho, Nsim=Nsim, Xchrom=TRUE, seed=123)
r2_X <- twoLocusIBD(x, ind1=3, ind2=4, cM=cM, Nsim=Nsim, Xchrom=TRUE, seed=123)
stopifnot(identical(r1_X, r2_X))


### Example 2: Comparing half sib, grandparent and uncle
# These are indistinguishable with unlinked loci, see e.g.
# pages 182-183 in Egeland, Kling and Mostad (2016).
# Each simulations followed by exact counterpart.
x <- addSon(addSon(nuclearPed(2, sex=1:2), 4), 5)
plot(x)
rho <- 0.25
Nsim <- 10 # Should be increased to at least 10000
twoLocusIBD(x, 1, 6, rho=rho, Nsim=Nsim, verbose=FALSE)[2,2]; .5*(1-rho)
twoLocusIBD(x, 8, 6, rho=rho, Nsim=Nsim, verbose=FALSE)[2,2]; .5*(rho^2+(1-rho)^2)
twoLocusIBD(x, 3, 6, rho=rho, Nsim=Nsim, verbose=FALSE)[2,2]; .5*((1-rho)*(rho^2+(1-rho)^2) + rho/2)

### Example 3: X chromosome, granddaughter vs maternal grandfather.
y <- addDaughter(nuclearPed(1, sex=2), 3)
plot(y)
rho <- 0.25
Nsim <-10
twoLocusIBD(y, 1, 5, rho=rho, Nsim=Nsim, Xchrom=TRUE)

# Exact
matrix(c(1-rho, rho, rho, 1-rho)/2, ncol=2)
```

`twoLocusJacquard` *Estimating two-locus Jacquard coefficients*

Description

Estimates by simulation the two-locus version of Jacquard's condensed identity coefficients for a pairwise relationship.

Usage

```
twoLocusJacquard(x, ind1, ind2, rho=NULL, cM=NULL, Nsim, verbose=TRUE,...)
```

Arguments

`x`	A pedigree in the form of a `linkdat` object.
`ind1, ind2`	Numeric ID labels of the two individuals.
`rho`	NULL, or a number in the interval [0, 0.5]: the recombination fraction between the two loci. If non-NULL, it is converted to centiMorgan using Haldanes map function: `cM = -50*log(1-2*rho)`.

cM	NULL, or a non-negative number: the distance in centiMorgan between the two loci. The numeric Inf is allowed, and corresponds to unlinked loci.
Nsim	The number of simulations to be performed.
verbose	A logical.
...	Further arguments to be passed on to IBDsim.

Details

As in the case of IBD coefficients (see twoLocusIBD), we can generalise Jacquard's identity coefficients to two loci: Given any pair of individuals, and any pair of autosomal loci, we define the two-locus Jacquard coefficient $\Delta[i,j]$ (where $1 \leq i,j \leq 9$) by

$$\Delta[i,j] = Pr(Jacquardstate\Sigma[i]atlocus1, andstate\Sigma[j]atlocus2).$$

Value

A numerical 9*9 matrix. The entry in row a and column b is the estimate of $\Delta[a-1,b-1]$ defined above.

Author(s)

Magnus Dehli Vigeland

See Also

jacquard, oneLocusIBD, twoLocusIBD, oneLocusJacquard

Examples

```
### Siblings whose parents are full siblings.
x = fullSibMating(generations=2)
Nsim = 100 # (increase to improve accuracy)

# Estimate of the 9 identity coefficients
j_est = oneLocusJacquard(x, ind1=5, ind2=6, Nsim=Nsim, seed=123)

### Two-locus Jacquard coefficients
# Completely linked loci
rho = 0
j2_linked = twoLocusJacquard(x, ind1=5, ind2=6, rho=rho, Nsim=Nsim, seed=123)
stopifnot(identical(diag(j2_linked), j_est))

# Completely unlinked
rho = 0.5
j2_unlinked = twoLocusJacquard(x, ind1=5, ind2=6, rho=rho, Nsim=Nsim, seed=123)
stopifnot(identical(j2_unlinked, outer(j_est, j_est)))
```

Index

∗ **datasets**
    DecodeMap, 3
    dominant1, 4
∗ **math**
    alleleSummary, 2
    IBDsim, 4
    oneLocusIBD, 8
    oneLocusJacquard, 10
    twoLocusIBD, 11
    twoLocusJacquard, 13

alleleSummary, 2

DecodeMap, 3
dominant1, 4

IBDsim, *2, 3*, 4, *9–11*, *14*

jacquard, *10*, *14*

linkdat, *5*, *9–11*, *13*

oneLocusIBD, 8, *10–12*, *14*
oneLocusJacquard, *9*, 10, *12*, *14*

set.seed, *5*

twoLocusIBD, *9, 10*, 11, *14*
twoLocusJacquard, *9, 10*, *12*, 13