Linkage disequilibrium based haplotype blocks

Matthias Frisch

1 Calculate and plot linkage disequilibrium **2**
1.1 Population structure 2
1.2 Calculate LD 3
1.3 Plot LD 4

2 Haplotype blocks **5**

3 Details of building haplotype blocks **8**

4 Using predefined haplotype definitions **9**

5 LD definitions implemented for homozygous data **10**

6 Gametic phase of the input data **10**
6.1 Known gametic phase 10
6.2 Unknown gametic phase 11

7 References **11**

1 Calculate and plot linkage disequilibrium

We consider 264 tropical maize lines from CIMMYT's Drought Tolerance Maize for Africa project that were analyzed with 1135 SNP markers. Data from Crossa et al. (2010). The original marker data are available from the Genetics webpage. Thanks to Dr. Jose Crossa and Dr. Raman Babu for providing the map data and the permission to use this data set as an example for data analysis. We load the package and the tropical maize data

```
> library("SelectionTools")
> vignette.data <- new.env(parent=emptyenv())
> data("v-tropmaize-vcf", package="SelectionTools", envir=vignette.data)
> st.load.vcf.data(vignette.data$v.tropmaize.vcf, data.set="base")
```

1.1 Population structure

For the population-structure analysis we first make a separate copy of the complete data and apply the same marker and individual filters used in the

```
> st.copy.marker.data("clustering", "base")
> st.restrict.marker.data(NoAll.MAX=2, MaMis.MAX=0.1, data.set="clustering")
> st.restrict.marker.data(InMis.MAX=0.1, data.set="clustering")
> st.restrict.marker.data(ExHet.MIN=0.1, data.set="clustering")
```

Rogers distances are calculated with

```
> dist.mat <- st.genetic.distances( measure="rd",
+                                   format="m",
+                                   data.set="clustering")
```

Four principal coordinates are obtained with `cmdscale()`. The percentages shown in the coordinate labels are computed from the eigenvalues returned by `cmdscale()`.

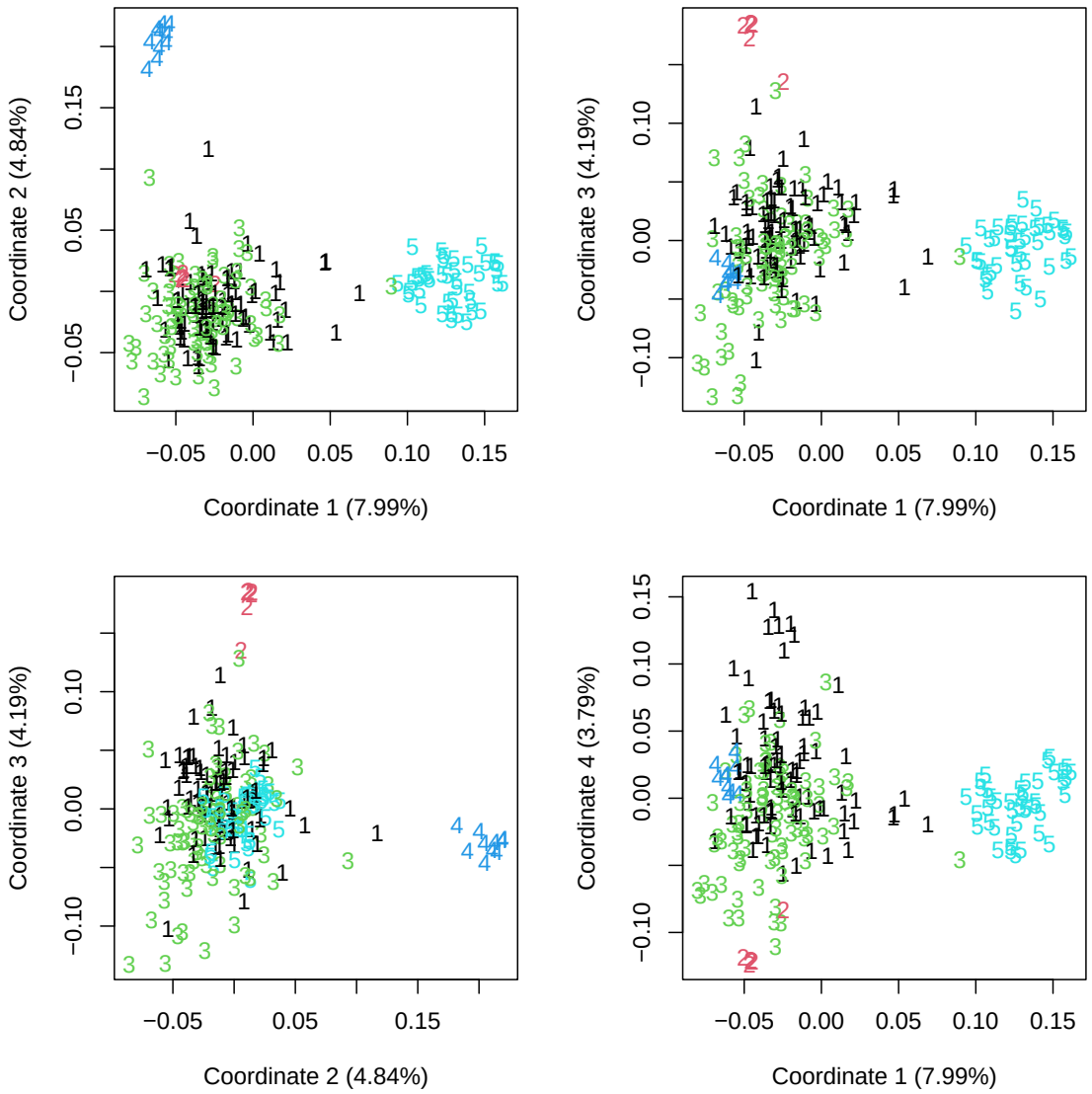
```
> pco <- cmdscale( dist.mat,
+                  k  = 4,
+                  eig= TRUE)
> PC <- pco$points
> var.exp <- round(pco$eig / sum(pco$eig) * 100, 2)
```

Five groups are constructed by K-means clustering of the distance profiles.

```
> set.seed(1)
> clus <- kmeans(as.matrix(dist.mat), centers=5, nstart=25)
> clusters <- clus$cluster
> cl.c <- as.character(clusters)
```

and make a two dimensional principal coordinate plots

```
> cl <- function(i)
+   paste0("Coordinate ", i, " (", sprintf("%.2f", var.exp[i]), "%)")
> par(mfrow=c(2,2), mar=c(4,4,2,2))
> plot(PC[,1], PC[,2], pch=cl.c, col=clusters, xlab=cl(1), ylab=cl(2))
> plot(PC[,1], PC[,3], pch=cl.c, col=clusters, xlab=cl(1), ylab=cl(3))
> plot(PC[,2], PC[,3], pch=cl.c, col=clusters, xlab=cl(2), ylab=cl(3))
> plot(PC[,1], PC[,4], pch=cl.c, col=clusters, xlab=cl(1), ylab=cl(4))
```



We look for the cluster that contains the line 142

```
> target.cluster <- unname(clusters["142"])
> target.ind <- names(clusters)[clusters == target.cluster]
```

Then we copy the base data set and restrict it to only those individuals belonging to the cluster to which line 142 is assigned to

```
> st.copy.marker.data("q3", "base")
> st.restrict.marker.data(ind.list=target.ind, data.set="q3")
> st.restrict.marker.data(MaMis.MAX=0.1, data.set="q3")
> st.restrict.marker.data(InMis.MAX=0.1, data.set="q3")
> st.restrict.marker.data(ExHet.MIN=0.1, data.set="q3")
```

We use this data set for our further analyses.

1.2 Calculate LD

Pairwise linkage disequilibrium is calculated within chromosomes with `st.calc.ld()`. The current implementation provides the frequency-weighted multiallelic measures `"r2"` and `"Dp"`. The returned data frame contains the chromosome, the two locus positions within the chromosome, the marker names, and the LD value.

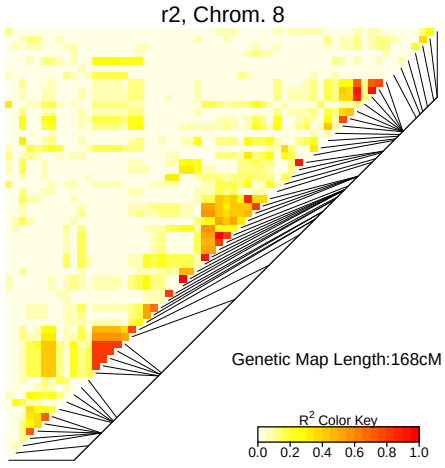
```
> ld <- st.calc.ld( ld.measure="r2",
+                   data.set="q3")
> ld[16042:16052,]
```

	Chrom	Locus1	Locus2	Name1	Name2	LD
16042	8	13	14	PZA033812	PZA033811	0.833895
16043	8	13	15	PZA033812	PZB021143	0.833895
16044	8	13	16	PZA033812	PZB000414	0.833895
16045	8	13	17	PZA033812	PZA004173	0.820898
16046	8	13	18	PZA033812	PZA004521	0.594679
16047	8	13	19	PZA033812	PZA037361	0.466027
16048	8	13	20	PZA033812	PZA035432	0.068560
16049	8	13	21	PZA033812	PZB018831	0.038208
16050	8	13	22	PZA033812	PZB018833	0.056772
16051	8	13	23	PZA033812	PZA036382	0.006269
16052	8	13	24	PZA033812	PZA036372	0.007459

1.3 Plot LD

`st.LDplot.ld()` converts the pairwise list for one chromosome into a matrix and the corresponding map positions are provided by `st.LDplot.map`. The heat map is drawn with the SelectionTools function `st.LDheatmap`, which uses standard R graphics and requires no additional plotting package.

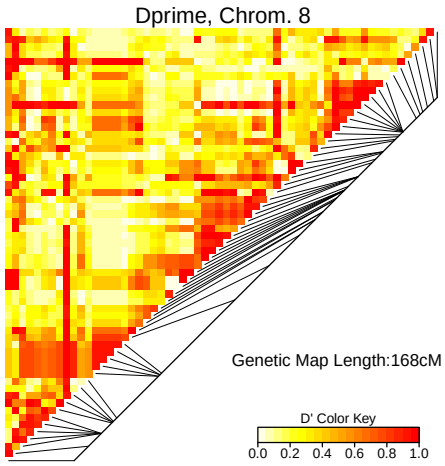
```
> chromosome <- 8
> l <- st.LDplot.ld(ld, chromosome)
> m <- st.LDplot.map(chromosome, data.set="q3")
> st.LDheatmap(l, m,
+              distances="genetical",
+              LDmeasure="r2",
+              title=paste("r2, Chrom.", chromosome),
+              col=heat.colors(20))
```



The measure `"Dp"` is calculated and plotted in the same way.

```
> ld <- st.calc.ld(ld.measure="Dp", data.set="q3")
> l <- st.LDplot.ld(ld, chrom=chromosome)
> m <- st.LDplot.map(chrom=chromosome, data.set="q3")
> st.LDheatmap(l, m,
+              distances="genetical",
+              LDmeasure="Dprime",
```

```
+              title=paste("Dprime, Chrom.", chromosome),
+              col=heat.colors(20))
```



2 Haplotype blocks

For repeated plotting we use the following utility function.

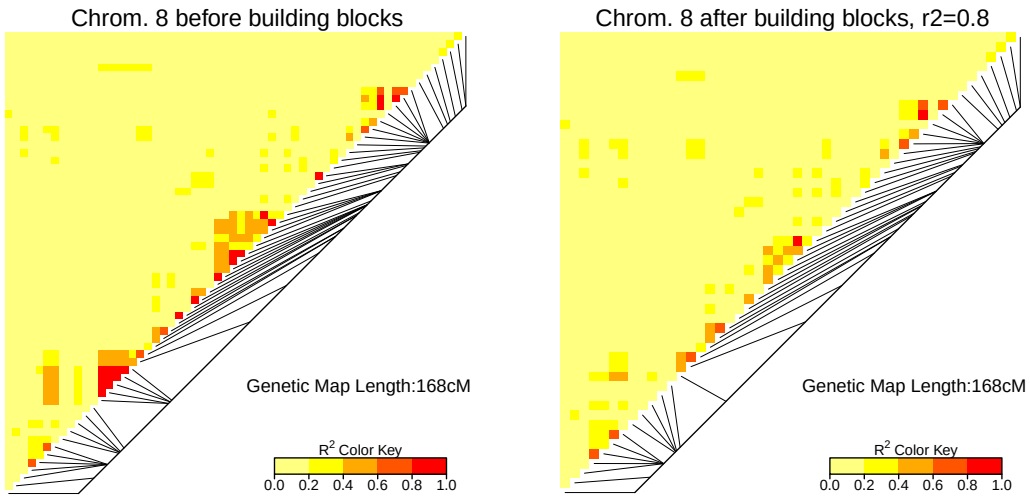
```
> LDplot <- function(ld.list, chrom, data.set, ld.measure="r2",
+                    title="", no.colors=20)
+ {
+   l <- st.LDplot.ld(ld.list, chrom)
+   m <- st.LDplot.map(chrom, data.set)
+   if ("Dp" == ld.measure) LDmeasure <- "Dprime" else LDmeasure <- "r2"
+   st.LDheatmap(l, m,
+                distances="genetical",
+                LDmeasure=LDmeasure,
+                title=paste("Chrom.", chrom, title),
+                col=heat.colors(no.colors))
+ }
```

For LD-based haploblock block construction, `st.calc.ld()` is called first and stores the pairwise LD values in the selected data set. Haplotype blocks are then constructed separately for every chromosome. The algorithm proceeds as follows:

1. Among unassigned markers, find the adjacent marker pair with the greatest LD value exceeding `ld.threshold`.
2. Start a new block with this pair. If no such pair remains, each remaining unassigned marker becomes a single-marker block.
3. Try to extend the block to the left and to the right. With `ld.criterion="flanking"`, the criterion is the LD between the two markers flanking the candidate block. With `"average"`, it is the average pairwise LD in the candidate block.
4. Extend the side with the larger criterion if it is at least the threshold, and repeat until the block can no longer be extended.
5. Repeat until all markers on the chromosome have been assigned to blocks.

The following example uses a threshold of 0.8 and the flanking-marker criterion. The block boundaries are defined with `st.def.hblocks()` and `st.recode.hil()` replaces the marker representation by haplotype-block variants whose alleles are the observed phased marker sequences.

```
> st.copy.marker.data("t3", "q3")
> ld <- st.calc.ld(ld.measure="r2", data.set="t3")
> LDplot(ld, chrom=8, data.set="t3", no.colors=5, title="before building blocks")
> st.def.hblocks(ld.threshold=0.8,
+                ld.criterion="flanking",
+                data.set="t3")
> st.recode.hil(data.set="t3")
> ld <- st.calc.ld(ld.measure="r2", data.set="t3")
> LDplot(ld, chrom=8, data.set="t3", no.colors=5, title="after building blocks, r2=0.8")
```

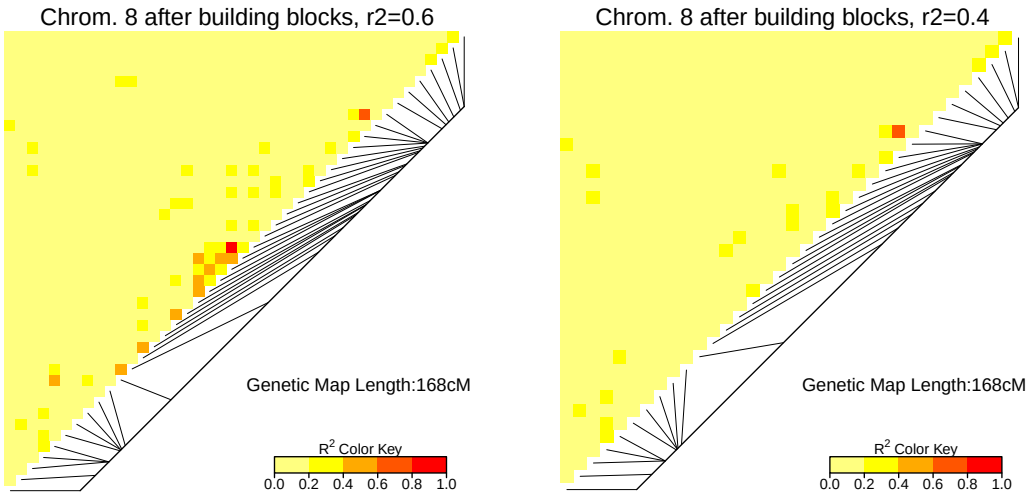


Lower thresholds result in larger blocks in this example. Each threshold is applied to a fresh copy of the original marker data set before haplotype recoding. Here we use an `r2=0.6`

```
> st.copy.marker.data("t3r06", "q3")
> st.calc.ld(ld.measure="r2", data.set="t3r06")
> st.def.hblocks(ld.threshold=0.6,
+                ld.criterion="flanking",
+                data.set="t3r06")
> st.recode.hil(data.set="t3r06")
> ld06 <- st.calc.ld(ld.measure="r2", data.set="t3r06")
> LDplot(ld06, chrom=8, data.set="t3r06",
+        no.colors=5, title="after building blocks, r2=0.6")
```

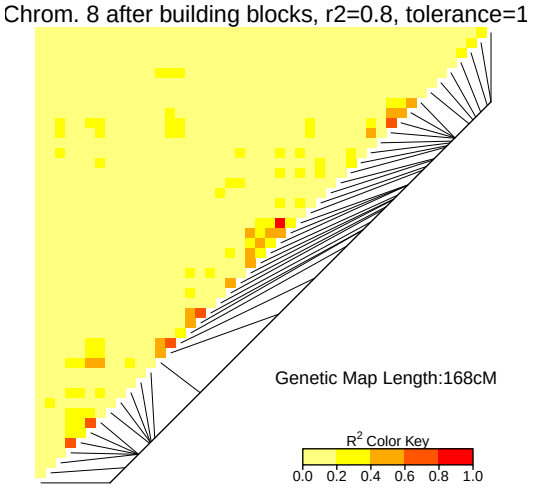
and a threshold of 0.4

```
> st.copy.marker.data("t3r04", "q3")
> st.calc.ld(ld.measure="r2", data.set="t3r04")
> st.def.hblocks(ld.threshold=0.4,
+                ld.criterion="flanking",
+                data.set="t3r04")
> st.recode.hil(data.set="t3r04")
> ld04 <- st.calc.ld(ld.measure="r2", data.set="t3r04")
> LDplot(ld04, chrom=8, data.set="t3r04",
+        no.colors=5, title="after building blocks, r2=0.4")
```



The `tolerance` argument allows the extension decision to look beyond a candidate boundary. For the flanking criterion, SelectionTools uses the best flanking LD value found within up to `tolerance` additional markers on the side being tested. This can bridge a marker with low LD when a marker just beyond it has sufficiently high LD.

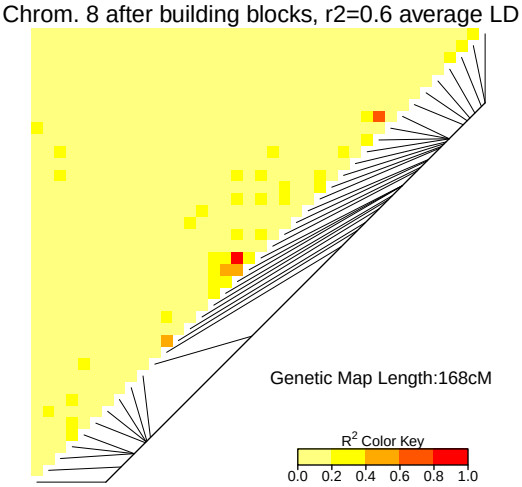
```
> st.copy.marker.data("t3tol", "q3")
> st.calc.ld(ld.measure="r2", data.set="t3tol")
> st.def.hblocks(ld.threshold=0.8,
+                tolerance=1,
+                ld.criterion="flanking",
+                data.set="t3tol")
> st.recode.hil(data.set="t3tol")
> ldtol <- st.calc.ld(ld.measure="r2", data.set="t3tol")
> LDplot(ldtol, chrom=8, data.set="t3tol",
+        no.colors=5, title="after building blocks, r2=0.8, tolerance=1")
```



Instead of LD between markers flanking the candidate block, the average pairwise LD within the candidate block can be used.

```
> st.copy.marker.data("t3avg", "q3")
> st.calc.ld(ld.measure="r2", data.set="t3avg")
> st.def.hblocks(ld.threshold=0.6,
+                ld.criterion="average",
+                data.set="t3avg")
> st.recode.hil(data.set="t3avg")
> ldavg <- st.calc.ld(ld.measure="r2", data.set="t3avg")
```

```
> LDplot(ldavg, chrom=8, data.set="t3avg",
+        no.colors=5, title="after building blocks, r2=0.6 average LD")
```



3 Details of building haplotype blocks

The following output illustrates the relationship between the LD table, original marker genotypes, the block definition returned by `st.def.hblocks()`, and the haplotype alleles returned by `st.recode.hil()`.

```
> options(width=120)
> st.copy.marker.data("t3detail", "q3")
> ld <- st.calc.ld(ld.measure="r2", data.set="t3detail")
```

```
> ld[8==ld$Chrom,][642:650,]
      Chrom Locus1 Locus2     Name1     Name2       LD
16041     8     12     60 PZA004817 PZA035371 0.018484
16042     8     13     14 PZA033812 PZA033811 0.833895
16043     8     13     15 PZA033812 PZB021143 0.833895
16044     8     13     16 PZA033812 PZB000414 0.833895
16045     8     13     17 PZA033812 PZA004173 0.820898
16046     8     13     18 PZA033812 PZA004521 0.594679
16047     8     13     19 PZA033812 PZA037361 0.466027
16048     8     13     20 PZA033812 PZA035432 0.068560
16049     8     13     21 PZA033812 PZB018831 0.038208
```

```
> x1 <- st.marker.data.statistics(data.set="t3detail")
```

```
> x1$genotypes[459:463,1:15]
    Mar/Ind  69 128 129 130 131 132 133 134 135 137 138 139 140 141
459 PZA033812 2/2 2/2 1/1 2/2 2/2 2/2 2/2 1/1 1/1 2/2 1/1 2/2 1/1 2/2
460 PZA033811 2/2 2/2 1/1 2/2 2/2 2/2 2/2 1/1 1/1 2/2 1/1 2/2 1/1 2/2
461 PZB021143 1/1 1/1 2/2 1/1 1/1 1/1 1/1 2/2 2/2 1/1 2/2 1/1 2/2 1/1
462 PZB000414 1/1 1/1 2/2 1/1 1/1 1/1 1/1 2/2 2/2 1/1 2/2 1/1 2/2 1/1
463 PZA004173 1/1 1/1 2/2 1/1 1/1 1/1 1/1 2/2 2/2 1/1 2/2 1/1 2/2 1/1

> x1$genotypes[459:463,39:44]
    182 183 184 185 186 187
459 2/2 2/2  -- 2/2 1/1 1/1
460 2/2 2/2  -- 2/2 1/1 1/1
461 1/1 1/1  -- 1/1 2/2 2/2
462 1/1 1/1  -- 1/1 2/2 2/2
463  -- 1/1 2/2 1/1 2/2 2/2
```

```
> h <- st.def.hblocks(ld.threshold=0.8,
+                     ld.criterion="flanking",
+                     data.set="t3detail")
```

```
> h[8==h$Chrom,][10:20,]
    Chrom       Pos    Name Class                                          Markers
426     8  22.64331 b000425     b                                       PZB014751;
427     8  22.97866 b000426     b                                       PZD000343;
428     8  24.53067 b000427     b                                       PZA004817;
429     8  44.55478 b000428     b PZA033812;PZA033811;PZB021143;PZB000414;PZA004173;
430     8  79.18913 b000429     b                                       PZA004521;
431     8  95.02281 b000430     b                                       PZA037361;
432     8 109.16423 b000431     b                                       PZA035432;
433     8 116.30544 b000432     b                                       PZB018831;
434     8 116.30563 b000433     b                                       PZB018833;
435     8 119.83192 b000434     b                                 PZA036382;PZA036372;
436     8 125.31375 b000435     b                                 PZB005911;PZB005921;
```

```
> r <- st.recode.hil(data.set="t3detail")
```

```
> r[1138:1147,]
       Block AlleleNr     AlleleDef
1138 b000427        3            -1
1139 b000428        1     2;2;1;1;1
1140 b000428        2     1;1;2;2;2
1141 b000428        3 -1;-1;-1;-1;2
1142 b000428        4    2;2;1;1;-1
1143 b000429        1             2
1144 b000429        2             1
1145 b000429        3            -1
1146 b000430        1             2
1147 b000430        2             1
```

```
> x2 <- st.marker.data.statistics(data.set="t3detail")
```

```
> x2$genotypes[429,1:15]
    Mar/Ind  69 128 129 130 131 132 133 134 135 137 138 139 140 141
429 b000428 1/1 1/1 2/2 1/1 1/1 1/1 1/1 2/2 2/2 1/1 2/2 1/1 2/2 1/1

> x2$genotypes[429,39:44]
    182 183 184 185 186 187
429  -- 1/1  -- 1/1 2/2 2/2
```

4 Using predefined haplotype definitions

Predefined block boundaries can be transferred to another marker data set with `st.set.hblocks()`. The `haplotype.list` argument can be the data frame returned by `st.def.hblocks()` or a manually constructed data frame. The current wrapper uses the `Markers` column to construct the positional block assignment. Consequently, the target data set must contain the corresponding markers in the same order and with the same total marker count; the function does not realign the block definition by marker name.

```
> # Use the globally filtered clustering data so both populations have the
> # same marker set and marker order. The comparison population is the
```

```
> # largest of the four clusters other than the target cluster.
> other.sizes <- table(clusters[clusters != target.cluster])
> other.cluster <- as.integer(names(other.sizes)[which.max(other.sizes)])
> other.ind <- names(clusters)[clusters == other.cluster]
> st.copy.marker.data("qtarget", "clustering")
> st.copy.marker.data("qother", "clustering")
> st.restrict.marker.data(ind.list=target.ind, data.set="qtarget")
> st.restrict.marker.data(ind.list=other.ind, data.set="qother")
> st.copy.marker.data("ttarget", "qtarget")
> st.copy.marker.data("tother", "qother")
> st.calc.ld(ld.measure="r2", data.set="ttarget")
> h <- st.def.hblocks(ld.threshold=0.8,
+                     tolerance=3,
+                     ld.criterion="flanking",
+                     data.set="ttarget")
> h2 <- st.set.hblocks(haplotype.list=h,
+                      hap.symbol="c",
+                      data.set="tother")
> st.recode.hil(data.set="tother")
```

5 LD definitions implemented for homozygous data

For an allele x at one locus and an allele y at a second locus, let f_x and f_y denote allele frequencies and f_{xy} the frequency of the corresponding haplotype. The gametic disequilibrium coefficient is

$$D_{xy} = f_{xy} - f_x f_y.$$

SelectionTools implements two normalized measures. For an allele pair,

$$r^2_{xy} = \frac{D^2_{xy}}{f_x(1-f_x)f_y(1-f_y)},$$

and

$$D'_{xy} = \left| \frac{D_{xy}}{D_{\max}} \right|,$$

where the maximum possible magnitude is determined from the allele frequencies and the sign of D_{xy}. For multiallelic markers, the current implementation combines allele-pair values using the equilibrium-frequency weights $f_x f_y$:

$$r^2 = \sum_x \sum_y f_x f_y r^2_{xy}, \qquad D' = \sum_x \sum_y f_x f_y D'_{xy}.$$

The behavior and interpretation of different LD measures, including their dependence on allele frequencies, are reviewed by Hedrick (1987).

6 Gametic phase of the input data

6.1 Known gametic phase

The options `ld.measure="r2"` and `ld.measure="Dp"` calculate haplotype frequencies directly from the two homologues stored in the marker data. Thus, the allele order in the input is interpreted as known gametic phase.

If `A/T` is recorded at the first locus and `G/C` at the second locus, the first stored homologue contributes haplotype `AG` and the second contributes `TC`. The maternal or paternal label of a homologue is irrelevant for LD; what matters is that alleles belonging to the same homologue are stored together across loci. For completely homozygous inbred lines the phase is immaterial. The direct calculation also supports multiallelic loci.

6.2 Unknown gametic phase

For heterozygous material with unknown phase, SelectionTools provides the additional options `ld.measure="r2-estimate-phases"` and `ld.measure="Dp-estimate-phases"`. These options estimate two-locus haplotype frequencies for biallelic markers before calculating r^2 or D'. SelectionTools performs this maximum-likelihood estimation internally; Clayton and Leung (2007) discuss haplotype estimation for association data.

For two biallelic loci with genotypes AA, Aa, aa and BB, Bb, bb, the nine observable two-locus genotype classes can be written as

	BB	Bb	bb
AA	a	b	c
Aa	d	e	f
aa	g	h	i

where e denotes the double heterozygotes. SelectionTools estimates the four haplotype frequencies by maximum likelihood with an expectation-maximization algorithm. If the current frequency estimates are h_{00}, h_{01}, h_{10}, and h_{11}, the probability that a double heterozygote carries the coupling diplotype is

$$q = \frac{h_{00}h_{11}}{h_{00}h_{11} + h_{01}h_{10}}.$$

The expectation step uses q to split the double heterozygotes between the two possible diplotypes. The maximization step recalculates the four haplotype frequencies from the expected haplotype counts. The steps are repeated to convergence. Several deterministic starting values are tried and the converged solution with the greatest observed-data likelihood is retained. The resulting haplotype frequencies are then used in the same r^2 or D' definitions described above.

The phase-estimation algorithm requires exactly two biological alleles at every marker in the selected data set. Missing alleles are not counted as biological alleles. Individuals with missing data at either member of a marker pair are omitted from the corresponding two-locus genotype table. Consequently, preprocessing to retain polymorphic biallelic markers may be required:

```
> st.copy.marker.data("q3phase", "q3")
> st.restrict.marker.data(NoAll.MAX=2, data.set="q3phase")
> st.restrict.marker.data(ExHet.MIN=0.001, data.set="q3phase")
> ld.phase.r2 <- st.calc.ld(ld.measure="r2-estimate-phases",
+                           data.set="q3phase")
> ld.phase.Dp <- st.calc.ld(ld.measure="Dp-estimate-phases",
+                           data.set="q3phase")
```

The phase-estimation model is intended for heterozygous material for which the two-locus gametic phase is unknown. Its assumptions should be considered when interpreting LD in populations affected by strong departures from the random-mating setting underlying the estimator. Hedrick (1987) provides a general review of LD measures and their interpretation.

7 References

Crossa J, de los Campos G, Pérez P, Gianola D, Burgueño J, et al. (2010) Prediction of genetic values of quantitative traits in plant breeding using pedigree and molecular markers. Genetics 186:713–724.

Clayton D, Leung HT (2007) An R package for analysis of whole-genome association studies. Human Heredity 64:45–51.

Hedrick PW (1987) Gametic disequilibrium measures: proceed with caution. Genetics 117:331–341.