Planing crosses

Matthias Frisch

1 Graphical genotypes **2**
1.1 Details of haplotype block building . . . 3
1.2 Graphical genotype of the best lines . . . 4

2 Marking reference genotypes **5**

3 Genetic distances of crossing partners **6**

4 Genome-wide marker effects **9**

5 Optimum contribution selection **10**

6 Optimum cross selection **12**
6.1 A complete observed-marker panel . . . 12
6.2 Joint selection of parents and crosses by mean . . . 12
6.3 Pairing at fixed parental contributions . . . 14
6.4 Simulation of the planned DH families . . . 15

7 Optimum cross selection using usefulness **16**
7.1 One score table for both criteria . . . 17
7.2 The usefulness criterion . . . 18

8 Optimum mate allocation **19**
8.1 Mating and parental bounds . . . 19
8.2 Mean-based mate allocation . . . 19
8.3 Usefulness-based mate allocation . . . 20

9 Proportional loss of expected genetic standard deviation **21**
9.1 A diversity constraint in cross selection . . . 22

10 Allocation of progeny to fixed crosses **23**
10.1 Comparison with simulated selected descendants . . . 24

A Mathematical definitions and implemented algorithms **24**
A.1 Marker prediction, segregation variance and usefulness . . . 24
A.2 Parental contributions and equal-sized integer families . . . 25
A.3 Continuous optimum mate allocation . . . 27
A.4 Covariance-based proportional loss of SD . . . 27
A.5 Integer progeny allocation for fixed families . . . 29

1 Graphical genotypes

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")
> st.set.info.level(0)
> data("v-tropmaize-vcf")
> data("v-tropmaize-phe")
> st.load.vcf.data(v.tropmaize.vcf)
> st.load.performance.data(v.tropmaize.phe)
```

The data are then preprocessed.

```
> st.restrict.marker.data( NoAll.MAX = 2    )
> st.restrict.marker.data( MaMis.MAX = 0.05 )
> st.restrict.marker.data( ExHet.MIN = 0.1  )
> st.restrict.marker.data( InMis.MAX = 0.1  )
```

We then make a copy of the preprocessed data for later.

```
> st.copy.marker.data("preprocessed","default")
```

Building haplotype blocks is a two-step procedure. In the first step the borders of the haplotypes are determined. Several options are available. We use here five adjacent markers.

```
> st.copy.marker.data( "hblocks01", "preprocessed" )
> h <- st.def.hblocks( hap      = 5,
+                      hap.unit = 1,
+                      data.set = "hblocks01" )
```

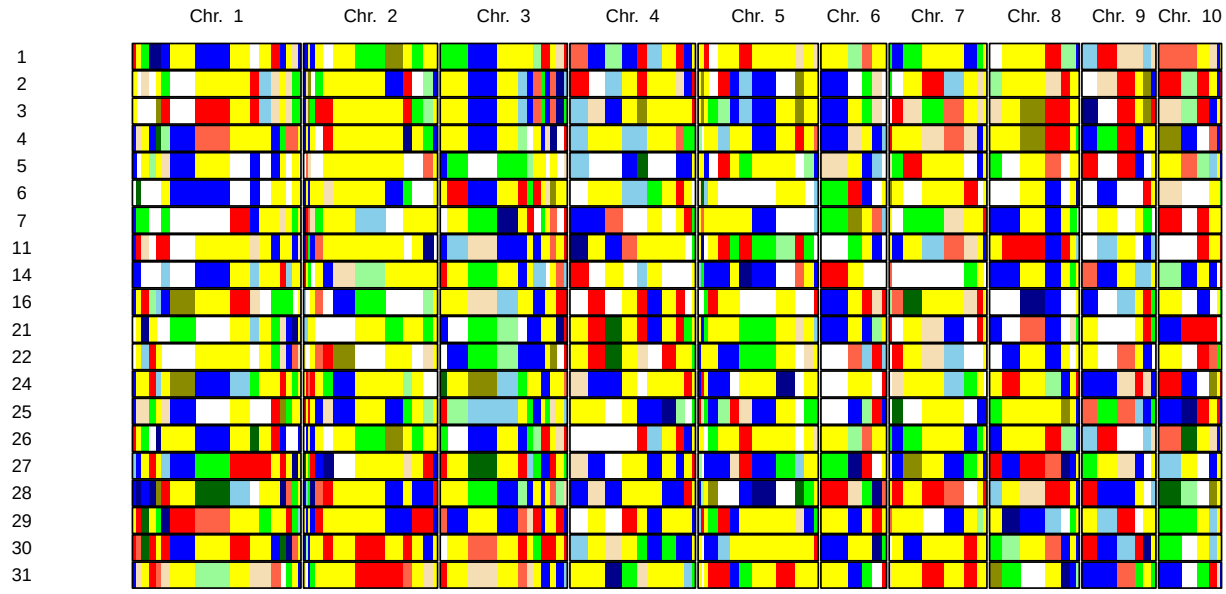
In the second step the variants of the haplotype blocks are determined. For inbred lines, the variants are determined with `st.recode.hil()`.

```
> st.recode.hil ( data.set="hblocks01" )
```

and the plot

```
> st.plot.ggt( f.ind    = 1,
+              l.ind    = 20,
+              data.set = "hblocks01")
```

Here are the graphical genotypes of the first 20 lines in the data set:



1.1 Details of haplotype block building

This is the list of haplotype blocks, five markers are combined to a block, the blocks are subsequently nameed started with `b000000`

```
> h.first <- h[1:10, ]
> h.first
```

```
   Chrom       Pos    Name Class
1      1   3.774747 b000000     b
2      1  10.654315 b000001     b
3      1  20.484429 b000002     b
4      1  36.397726 b000003     b
5      1  47.886392 b000004     b
6      1  55.773202 b000005     b
7      1  75.985652 b000006     b
8      1 145.383079 b000007     b
9      1 198.285440 b000008     b
10     1 213.593020 b000009     b
                                          Markers
1   PZA036131;PZA036142;PZA003931;PZA003934;PZA028692;
2   PZB019151;PZA035191;PZA035201;PZA035203;PZA035521;
3   PZA035583;PZA035571;PZA036631;PZA029219;PZB025153;
4   PZB020583;PZB016622;PZB016621;PZA001926;PZA037422;
5   PZA037421;PZB025171;PZA036732;PZA036731;PZA031681;
6   PZA031682;PZA037461;PZA003281;PZA035611;PZA035612;
7   PZA003789;PZA032404;PZA034072;PZA034073;PZA034074;
8  PHM1062129;PZB010671;PHM172534;PZA035802;PZB001492;
9   PZD000535;PZA034271;PZA034272;PZA036851;PZB011562;
10 PZA032182;PZA031931;PZA0028911;PZA037411;PZA036323;
```

Alternatively we could form haplotyype blocks using a fixed length of the chromosome stretches

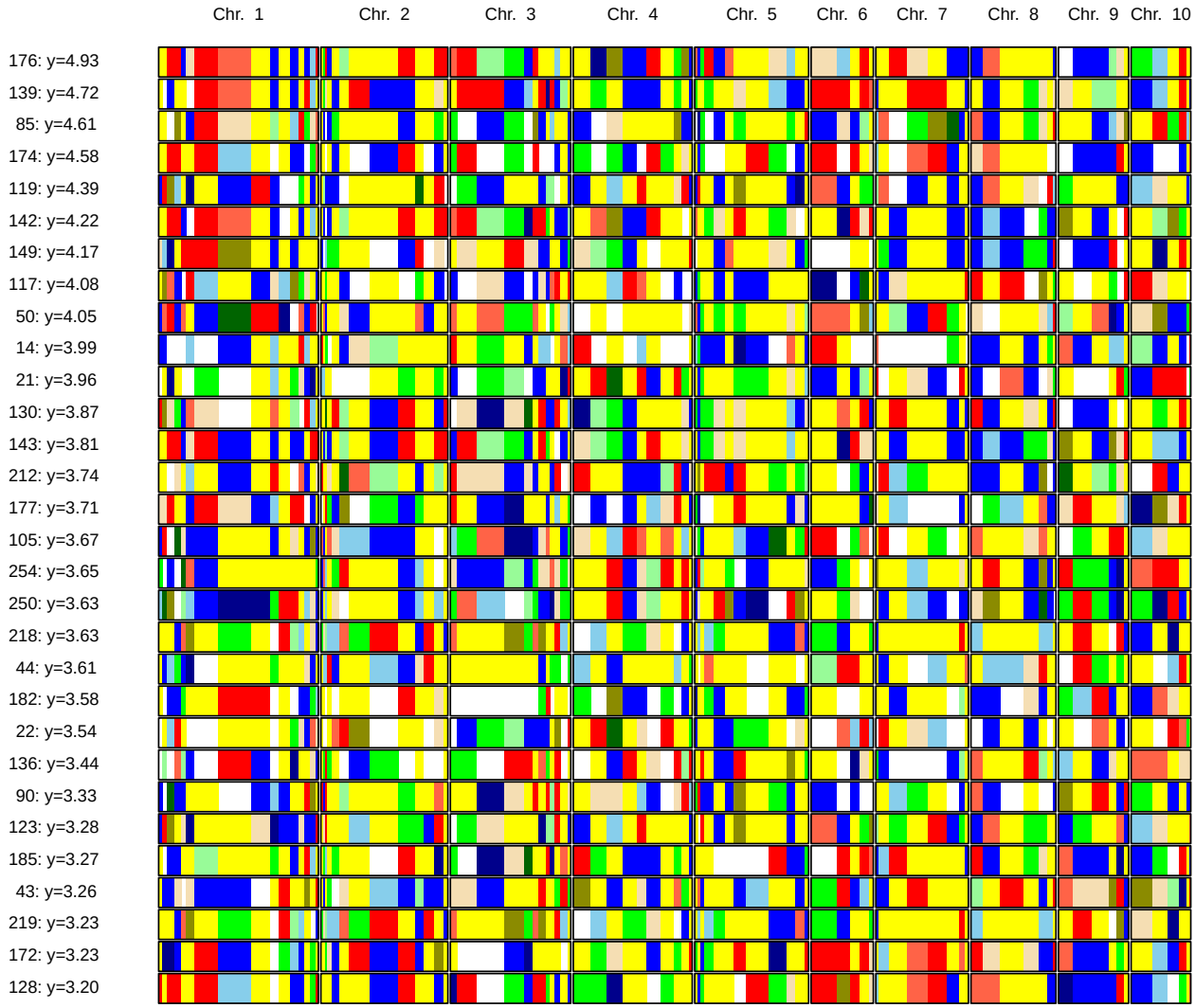
```
> st.copy.marker.data( "hblocks02", "preprocessed" )
> h2 <- st.def.hblocks( hap      = 5,
+                       hap.unit = 2,
+                       data.set = "hblocks02" )
> h2.first <- h2[1:10, ]
> h2.first
```

```
   Chrom        Pos    Name Class
1      1   5.999946 b000000     b
2      1  17.517206 b000001     b
3      1  31.194276 b000002     b
4      1  39.391025 b000003     b
5      1  46.893396 b000004     b
6      1  54.497620 b000005     b
7      1  74.244593 b000006     b
8      1  95.772987 b000007     b
9      1 137.777371 b000008     b
10     1 181.739428 b000009     b
                                                                                Markers
1                       PZA036131;PZA036142;PZA003931;PZA003934;PZA028692;PZB019151;
2  PZA035191;PZA035201;PZA035203;PZA035521;PZA035583;PZA035571;PZA036631;PZA029219;
3                                                    PZB025153;PZB020583;PZB016622;
4                                                    PZB016621;PZA001926;PZA037422;
5                                                    PZA037421;PZB025171;PZA036732;
6                                 PZA036731;PZA031681;PZA031682;PZA037461;PZA003281;
7                                           PZA035611;PZA035612;PZA003789;PZA032404;
8                                         PZA034072;PZA034073;PZA034074;PHM1062129;
9                                                          PZB010671;PHM172534;
10                                                         PZA035802;PZB001492;
```

1.2 Graphical genotype of the best lines

Performance values stored in the marker data set can be used directly by `st.plot.ggt`. With `sort.performance=TRUE`, the individuals are sorted according to their performance before `f.ind` and `l.ind` are applied. Thus, with decreasing sorting, `f.ind=1` and `l.ind=30` select the 30 best lines. With `show.performance=TRUE`, the performance value is added automatically to each individual label.

```
> st.plot.ggt(f.ind                 = 1,
+             l.ind                 = 30,
+             sort.performance      = TRUE,
+             show.performance      = TRUE,
+             performance.decreasing = TRUE,
+             data.set              = "hblocks01")
```



2 Marking reference genotypes

To find a suitable crossing partner, a line can be marked as a reference. Chromosome regions of a line that are similar to the reference line are yellow, other chromosome segments are blue. We first make a copy of the haplotyped data set

```
> st.copy.marker.data( "hblocks03", "hblocks01" )
```

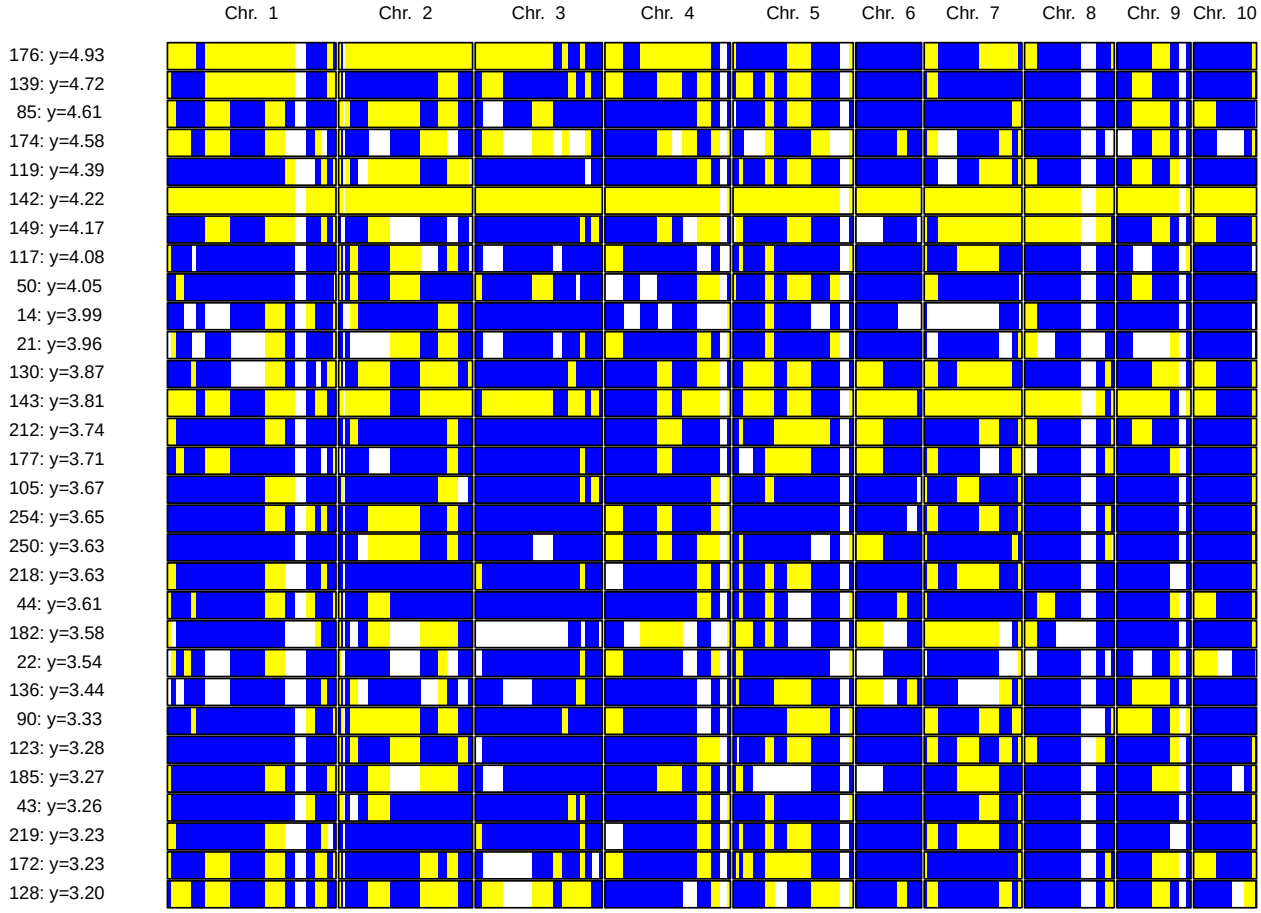
The reference individual can be specified directly by its identifier. Here the individual named `142` is used as reference.

```
> st.recode.ref( reference = "142",
+                data.set  = "hblocks03")
```

We then plot the 30 best lines again. Sorting and performance labels are obtained directly from the performance data stored in the data set.

```
> st.plot.ggt( f.ind                = 1,
+              l.ind                = 30,
+              sort.performance     = TRUE,
+              show.performance     = TRUE,
```

```
+              performance.decreasing = TRUE,
+              data.set               = "hblocks03")
```



3 Genetic distances of crossing partners

We make a copy of the preprocessed marker data individuals.

```
> st.copy.marker.data( "gd01", "preprocessed" )
```

The performance values stored in the SelectionTools data set can be used by `st.select.phen` to give a list of the best 50 lines

```
> best <- st.select.phen( n=50, data.set="gd01" )
```

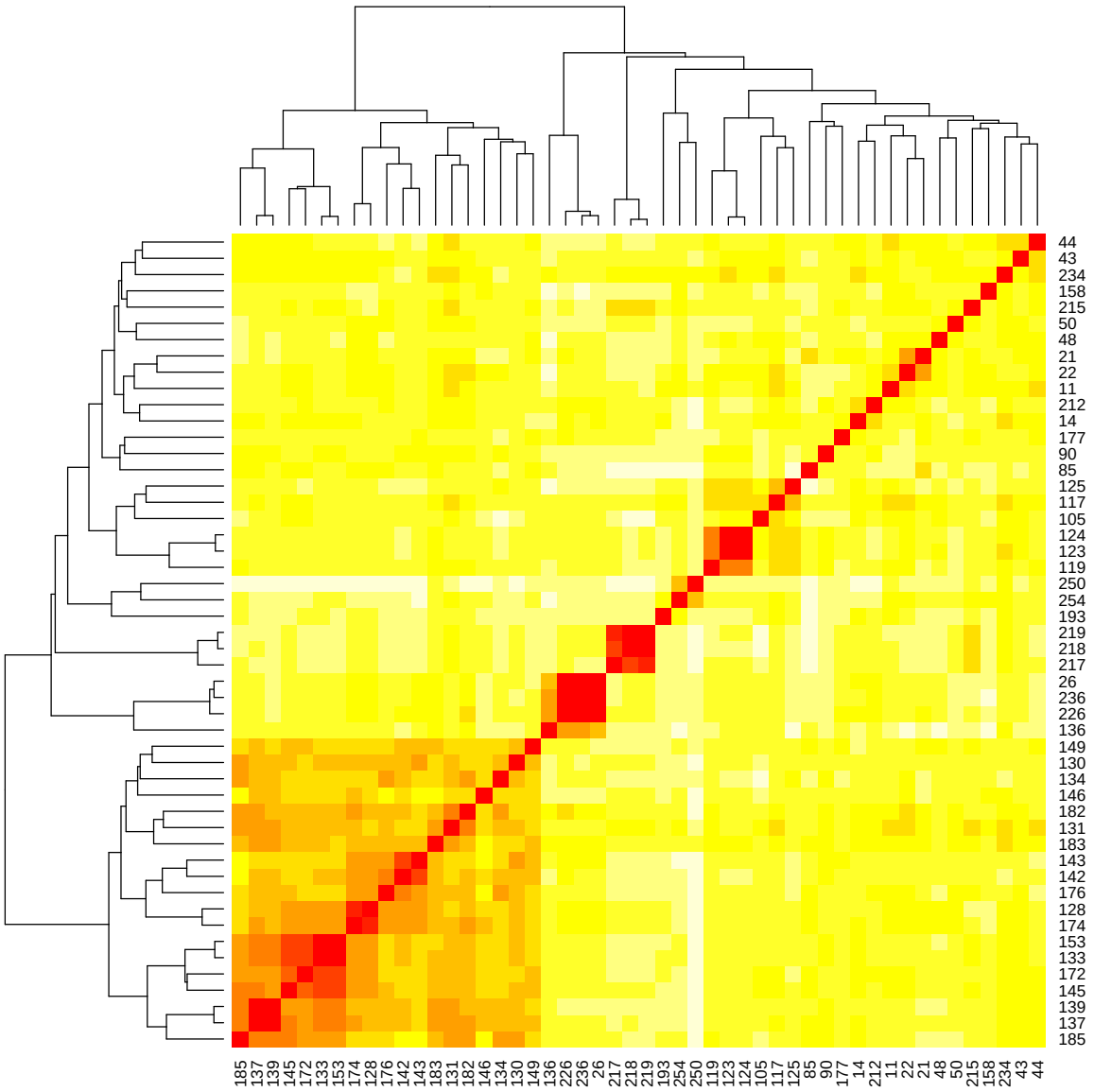
Then a new data set is generated with these 50 lines

```
> st.restrict.marker.data ( ind.list=best$i,
+                           data.set="gd01" )
```

The function `st.genetic.distances` calculates pariwise genetic distances of the genotypes in a a data set. We calculate the Rogers Distance with `measure ="rd"`. With `format="m"` a standard R `dist` object is returned and can be converted to a square matrix for plotting.

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

```
+                                        data.set = "gd01" )
> dm <- as.matrix(dist.mat)
> heatmap( dm, scale="none", col=heat.colors(12) )
```



For ranking the crossing combinations we use the Modified Rogers distance. The pairwise distances are therefore recalculated with `gs.cross.eval.gd` using `dist="mrd"`, and the ranked crossing information is then retrieved with `gs.cross.info`.

```
> gs.cross.eval.gd ( dist="mrd", data.set="gd01" )
> crosses <- gs.cross.info ( sortby="gd", bestn=50, data.set="gd01" )
> distance.crosses <- crosses[1:10, ]
> distance.crosses
```

```
   P1No P2No P1Name P2Name        gd mu mi ma va es
1    10   50     85    254 0.6112693 NA NA NA NA NA
2    45   49    219    250 0.6074501 NA NA NA NA NA
3    34   49    174    250 0.6067799 NA NA NA NA NA
4    31   49    153    250 0.6061281 NA NA NA NA NA
5    44   49    218    250 0.6061281 NA NA NA NA NA
6    18   49    128    250 0.6049901 NA NA NA NA NA
```

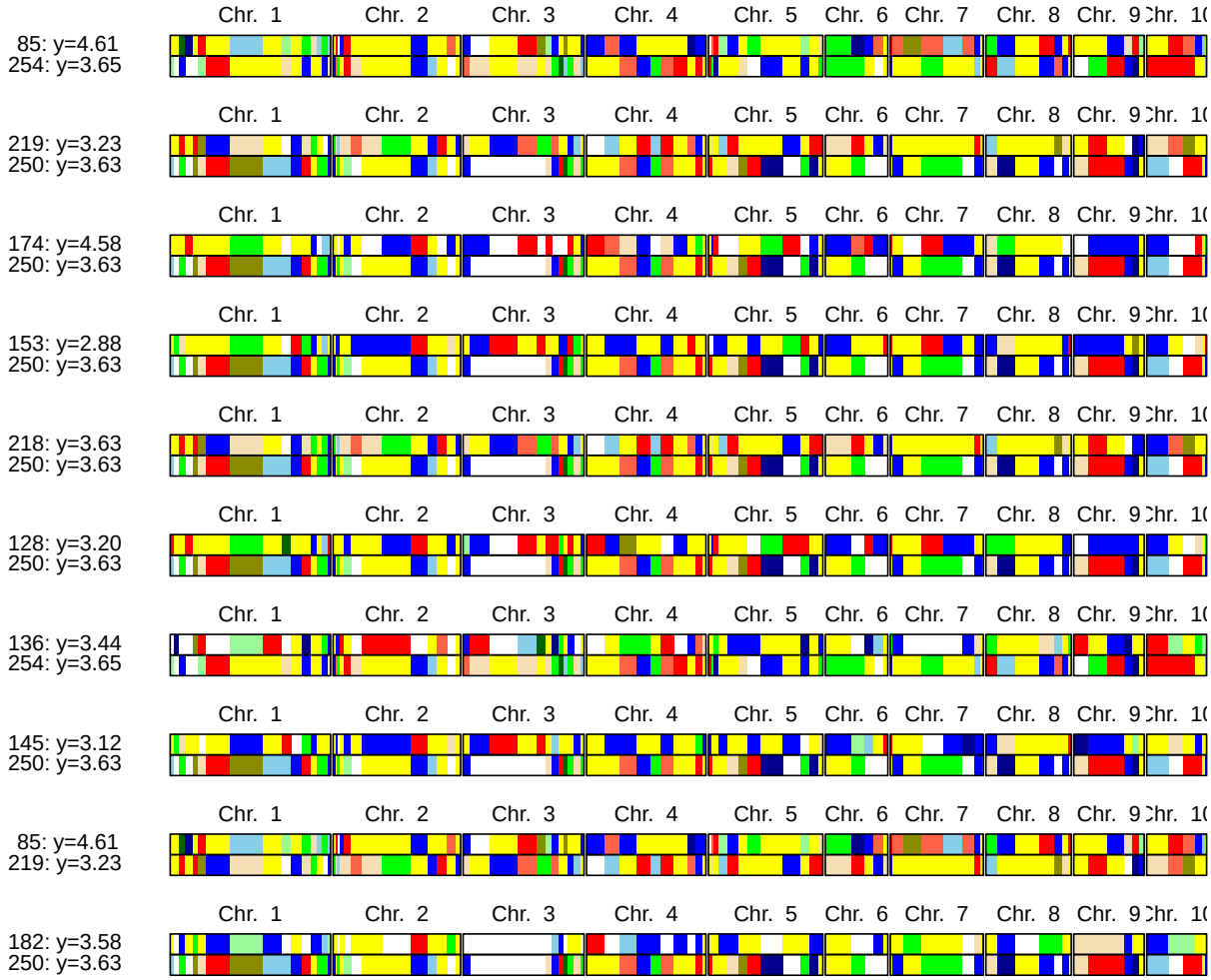
```
7    23   50    136    254 0.6038653 NA NA NA NA NA
8    28   49    145    250 0.6033532 NA NA NA NA NA
9    10   45     85    219 0.6025309 NA NA NA NA NA
10   37   49    182    250 0.6025062 NA NA NA NA NA
```

We now want to inspect the graphical genotypes of the pairs of lines with the greatest genetic distances and generate haplotype blocks

```
> st.def.hblocks( hap=5, data.set="gd01" )
> st.recode.hil ( data.set="gd01" )
```

and plot the 10 crosses with the greatest genetic distance:

```
> st.set.info.level(-2)
> for (i in seq_len(min(10,nrow(crosses)))) {
+     a <- crosses$P1Name[i]
+     b <- crosses$P2Name[i]
+     f.nme <- sprintf("ex-crs-07-%03i.pdf",100+i)
+     pdf(f.nme,width=8,height=0.8,pointsize=10)
+     st.plot.ggt(i.list=c(a,b),
+                 show.performance=TRUE,
+                 cex.chrom=0.8,
+                 p.t=FALSE,
+                 p.s=FALSE,
+                 data.set="gd01")
+     dev.off()
+ }
> st.set.info.level(0)
```



4 Genome-wide marker effects

We make a copy of the processed data set

```
> st.copy.marker.data( "effects01", "preprocessed" )
```

Then genome-wide marker effects are estimated from the marker and performance data. Here we use BLUP as implemented in `gs.esteff.rr`.

```
> gs.esteff.rr( method="BLUP", data.set="effects01" )
```

Once marker effects have been estimated, all possible crosses among the individuals in the data set can be evaluated using several complementary criteria. The expected cross mean is calculated with `gs.cross.eval.mu`. Lower and upper marker-effect based limits are calculated with the functions `gs.cross.eval.mi` and `gs.cross.eval.ma` for progeny values that can be assembled from the parental alleles.

```
> gs.cross.eval.mu( data.set="effects01" )
> gs.cross.eval.mi( data.set="effects01" )
> gs.cross.eval.ma( data.set="effects01" )
```

The segregation variance describes how strongly progeny from a cross are expected to differ genetically. It can be calculated for doubled-haploid or single-seed-descent progeny while taking linkage and recombination into account. The approach implemented here follows Osthushenrich et al. (2017).

```
> gs.cross.eval.va( pop.type="DH", data.set="effects01" )
```

The expected superior progeny criterion combines the expected cross mean with the amount of segregation expected among the progeny. It therefore favours crosses that have a high mean and, at the same time, enough genetic variation to generate outstanding descendants. For large populations the selected fraction is positive and no greater than one. With this fraction, the approach of Zhong and Jannink (2007) is used.

```
> gs.cross.eval.es( alpha=0.25, data.set="effects01" )
```

For small finite progeny populations, the numbers of selected individuals and evaluated individuals can be supplied directly; the corresponding selection-intensity correction follows Burrows (1972). Both counts are positive whole numbers, with `N` no greater than `G`. When all progeny are retained, `N=G` or `alpha=1`, the selection differential is zero and the criterion equals the cross mean. The mean and variance are evaluated before this criterion; a calculated zero variance is valid.

```
> gs.cross.eval.es( N=1, G=4, data.set="effects01" )
```

Genetic distance can be evaluated independently of the marker effects. In this example we use Rogers distance; properties of genetic distance measures used in plant breeding are discussed by Reif et al. (2005).

```
> gs.cross.eval.gd(dist="rd",data.set="effects01")
```

The criteria can be calculated for the same set of crosses and then summarized together. Criteria that have not been evaluated appear as `NA`, whereas an evaluated zero remains zero. Sorting requires an evaluated criterion. The lower marker-effect limit, `sortby="mi"`, is sorted from smallest to largest; the other scores are sorted from largest to smallest. Here the crosses are ranked by expected superior progeny value and the best 20 are returned.

```
> gs.cross.eval.gd(dist="rd",data.set="effects01")
> gs.cross.eval.mu(data.set="effects01")
> gs.cross.eval.ma(data.set="effects01")
> gs.cross.eval.mi(data.set="effects01")
> gs.cross.eval.va(pop.type="DH",data.set="effects01")
> gs.cross.eval.es(N=1,G=4,data.set="effects01")
> crosses <- gs.cross.info(sortby="es",bestn=20,data.set="effects01")
> formatted.crosses <- format(crosses,digits=2,nsmall=2)
> formatted.crosses
```

	P1No	P2No	P1Name	P2Name	gd	mu	mi	ma	va	es
1	123	153	142	176	0.148	3.62	-1.05	6.35	0.032	3.80
2	124	153	143	176	0.163	3.59	-0.97	6.40	0.025	3.76
3	123	124	142	143	0.093	3.55	-0.83	5.90	0.020	3.70
4	118	153	137	176	0.207	3.51	-1.43	6.75	0.029	3.69
5	118	123	137	142	0.212	3.47	-1.16	6.59	0.035	3.66
6	120	153	139	176	0.212	3.48	-1.47	6.75	0.029	3.66
7	120	123	139	142	0.217	3.44	-1.23	6.59	0.036	3.64
8	118	124	137	143	0.226	3.45	-1.28	6.59	0.025	3.61
9	111	153	130	176	0.194	3.43	-1.46	6.59	0.029	3.61
10	100	153	119	176	0.304	3.32	-1.83	6.85	0.080	3.61
11	111	123	130	142	0.200	3.39	-1.30	6.41	0.040	3.60
12	105	153	124	176	0.300	3.39	-1.73	6.78	0.037	3.59
13	109	153	128	176	0.186	3.42	-1.24	6.42	0.025	3.59
14	104	153	123	176	0.297	3.39	-1.66	6.78	0.034	3.58
15	109	123	128	142	0.168	3.38	-1.17	6.07	0.037	3.58
16	120	124	139	143	0.229	3.42	-1.31	6.60	0.024	3.58
17	105	123	124	142	0.317	3.35	-1.65	6.55	0.046	3.57
18	104	123	123	142	0.314	3.35	-1.62	6.52	0.047	3.57
19	151	153	174	176	0.176	3.41	-1.19	6.42	0.021	3.56
20	109	124	128	143	0.183	3.36	-1.30	6.05	0.032	3.54

5 Optimum contribution selection

The predicted genetic values can also be used to choose the proportion of genes contributed by each parent to the next generation. For an additive model, the expected genetic value of the resulting population is the contribution-weighted mean of these parental values. `st.ocs` maximizes this mean subject to an upper bound on group relatedness and lower and upper bounds on each contribution (Wellmann, 2019). These continuous proportions sum to one.

The following analysis uses the same tropical-maize lines and fitted marker effects as the cross evaluations. Predicted genetic values supply the named merit vector. The genomic relationship matrix, `A`, is centered using marker frequencies in this complete candidate set. The matrix and its reference remain fixed when contributions change. Here `K=A/2` is the kinship-scaled matrix, and group relatedness is the quadratic form of `K` with the contribution vector. In this example it is a genomic relatedness index relative to the candidate panel, rather than an absolute probability of identity by descent. Equal contributions to all candidates have an index approximately zero on this centered reference.

```
> values <- gs.estimate.gv(estimation.set="effects01",
+                          prediction.set="effects01")
> merit <- setNames(values$yhat, values$i)
> A <- st.get.A("effects01")
> merit <- merit[rownames(A)]
> K <- A/2
```

The chosen breeding scenario sets `theta=0.02` as the bound on group relatedness on this reference scale. Each contribution has a lower bound of zero and an upper bound of 0.1, so a single parent can supply at

most ten percent of the nominal gene contribution. The merit vector is ordered explicitly to match both dimensions of `K`. The returned `gain` is the contribution-weighted mean of the supplied merit values.

```
> contributions <- st.ocs(
+     merit = merit,                          # Parental genetic values
+     K = K,                                  # Kinship matrix
+     theta = 0.02,                           # Group relatedness limit
+     lower = 0,                              # Minimum parental contribution
+     upper = 0.1                             # Maximum parental contribution
+ )
> contributions
```

```
Optimum contribution selection
Criterion: maximum expected genetic merit
Status: optimal
Expected merit: 2.7698
Group relatedness: 0.0200; requested limit: 0.0200
Minimum relatedness: >-0.0001; lower bound: 0.0000
Gain upper bound: 2.7698; optimality gap: <0.0001
Maximum constraint residual: <0.0001

Parental contributions (236 parents, 64 positive)
  Parent  Contribution
  218           0.0575
  176           0.0433
  44            0.0399
  119           0.0376
  137           0.0368
  85            0.0350
  117           0.0344
  226           0.0325
  50            0.0289
  21            0.0282
  [221 table rows omitted]
  258           0.0000
  259           0.0000
  261           0.0000
  263           0.0000
  264           0.0000
```

Interpretation of the printed summary

The summary above the contribution table reports the objective and the numerical qualification of the solution.

Criterion and status. The criterion is the quantity maximized. `optimal` denotes a certified merit optimum; `maxit` denotes an unresolved iteration or numerical limit. An infeasible request has no solution under the supplied constraints.

Expected merit. This is the contribution-weighted mean of the supplied parental genetic values.

Group relatedness. This is the achieved $c'Kc$, compared with the requested limit `theta`.

Minimum relatedness and lower bound. These describe the separate search for the smallest attainable relatedness under the contribution bounds. Their difference measures its remaining numerical gap.

Gain upper bound and optimality gap. The bound limits the best attainable merit. Its difference from the achieved merit is the gap.

Maximum constraint residual. This is the largest numerical violation of the constraints; values close to zero indicate feasibility within numerical tolerance.

The contribution table identifies the parents and their gene shares, which sum to one. Its displayed rows include the greatest and smallest shares.

The result also reports the minimum-relatedness calculation for the supplied contribution bounds. A requested `theta` below the feasible range is infeasible. The returned status and diagnostics describe feasibility and numerical convergence. These contributions refer to the nominal gene shares before selection within progeny families. A crossing plan additionally specifies parental pairs and integer progeny counts; realized ancestry after selection can differ from the nominal contributions.

6 Optimum cross selection

`st.cross.plan` selects parent pairs for a specified number of families. The default joint mode chooses both parental usage and pairing to maximize the predicted additive population mean, subject to contribution limits and the group relatedness bound. The specified number of crosses refers to distinct unordered pairs of different parents. Here every family has the same progeny count.

6.1 A complete observed-marker panel

The execution example uses a separate panel comprising the first 30 lines in the processed data order. Markers with any missing call among these parents are excluded, leaving 243 markers with complete homozygous founder genotypes. The original intercept and the fitted effects of these retained markers define the prediction model for this example. Merit and the genomic relationship matrix are recalculated for precisely this panel. Its marker set and reference frequencies differ from the preceding 236-line analysis; the relatedness bound therefore has this smaller panel as its reference. Planning and simulation below share the same parents, markers and effect model.

```
> marker.effects <- gs.return.effects(data.set="effects01")
> st.copy.marker.data("cross.parents", "effects01")
> st.restrict.marker.data(ind.list=names(merit)[1:30],
+                         data.set="cross.parents")
> st.restrict.marker.data(MaMis.MAX=0, data.set="cross.parents")
> cross.map <- st.get.map(data.set="cross.parents")
```

Marker-data copies do not retain fitted effects. The marker names from the map identify the original effects attached by `gs.set.effects`, together with the fitted intercept.

```
> cross.effects <- marker.effects[
+   marker.effects$marker %in% c("mu", cross.map$Name), ]
> gs.set.effects(cross.effects, data.set="cross.parents")
> cross.values <- gs.estimate.gv(estimation.set="cross.parents",
+                                prediction.set="cross.parents")
```

The named merit vector follows the parent order of `cross.A`. Dividing this additive relationship matrix by two supplies the kinship scale used in the following plans.

```
> cross.merit <- setNames(cross.values$yhat, cross.values$i)
> cross.A <- st.get.A("cross.parents")
> cross.merit <- cross.merit[rownames(cross.A)]
> cross.K <- cross.A/2
```

6.2 Joint selection of parents and crosses by mean

The following breeding scenario contains 15 distinct families with 100 progeny each. A parental contribution of at most 0.1 permits a parent to occur in at most three of these equal-sized families. Group relatedness is bounded by `theta=0.02` on the 30-parent reference scale.

```
> cross.plan <- st.cross.plan(
+     merit = cross.merit,                    # Parental genetic values
+     K = cross.K,                            # Kinship matrix
+     n.crosses = 15,                         # Number of distinct parent pairs
+     n.progeny = 100,                        # Progeny per family
+     theta = 0.02,                           # Group relatedness limit
+     upper = 0.1                             # Maximum parental contribution
+ )
> cross.plan
```

```
Optimum cross selection (mean)
Criterion: maximum predicted mean
Status: best.found
Families: 15; progeny per family: 100; total progeny: 1500
Expected mean: 2.3854
Group relatedness: 0.0200; requested limit: 0.0200
Average within-cross relatedness: -0.0159
Objective bound: 2.3992; optimality gap: 0.0138
Maximum constraint residual: 0.0000

Chosen crosses
  Family  Parent 1  Parent 2  Progeny    Mean  Kinship
       1  1         5              100  2.2508  -0.0689
       2  1         27             100  2.3523  -0.0447
       3  2         6              100  2.4583  -0.0311
       4  2         30             100  2.2902  -0.0290
       5  3         7              100  2.1269  -0.1213
       6  6         34             100  2.4984   0.0129
       7  11        14             100  2.7394  -0.1286
       8  11        22             100  2.6670   0.1692
       9  11        38             100  2.6248  -0.0542
      10  14        34             100  2.6336  -0.0500
      11  14        39             100  2.3035  -0.0657
      12  16        22             100  2.3135   0.0832
      13  29        31             100  2.0175   0.1063
      14  32        37             100  2.1386  -0.0825
      15  34        41             100  2.3654   0.0664
```

Interpretation of the printed summary

The summary above the chosen crosses describes the breeding scenario, objective and numerical qualification of the plan.

Criterion and status. The criterion names the objective. `best.found` denotes a feasible search result, whereas `optimal` also requires a bound certifying the optimum. `search.limit` means that the search has not found a feasible plan.

Families and progeny. These give the number of distinct crosses, the progeny count per family and the total progeny count.

Expected mean. This is the average predicted cross mean for these equal-sized families. With supplied cross scores, expected usefulness additionally includes the predicted gain from within-family selection; `accuracy` specifies the assumed accuracy of that ranking.

Group relatedness. This is $c'Kc$ for the pooled parental gene shares, compared with the requested limit.

Average within-cross relatedness. This is the average kinship between the two parents of each chosen cross. It concerns pairing and can change while pooled parental contributions remain fixed.

Objective bound and optimality gap. The bound is an upper bound for a maximized objective and a lower bound for the minimized pairing objective. The gap measures the remaining difference from that bound.

Maximum constraint residual. This is the largest numerical violation of the plan constraints.

The cross table gives the parental identifiers, progeny counts, predicted cross means and parental kinships. A usefulness column appears when the corresponding cross scores are supplied.

6.3 Pairing at fixed parental contributions

With `mode="fixed"`, the parental gene shares are supplied explicitly. Their additive predicted mean and group relatedness remain fixed, while the pairing objective is to minimize average within-cross relatedness. This example uses the gene shares of the preceding feasible plan, so they already correspond to integer parental-use counts for the 15 families. Other supplied proportions that cannot be realized by the requested integer family counts are reported as infeasible without automatic rounding. The preceding plan provides the starting mating list through `control=list(start=cross.plan)`.

```
> fixed.plan <- st.cross.plan(
+     merit = cross.merit,                        # Parental genetic values
+     K = cross.K,                                # Kinship matrix
+     n.crosses = 15,                             # Number of distinct parent pairs
+     n.progeny = 100,                            # Progeny per family
+     theta = 0.02,                               # Group relatedness limit
+     upper = 0.1,                                # Maximum parental contribution
+     contributions = cross.plan$contributions,   # Fixed parental gene shares
+     mode = "fixed",                             # Pairing at fixed contributions
+     control = list(                             # Search settings
+         start = cross.plan                      # Starting crossing plan
+     )
+ )
> fixed.plan
```

```
Fixed-contribution mating allocation
Criterion: minimum average within-cross relatedness
Status: best.found
Families: 15; progeny per family: 100; total progeny: 1500
Expected mean: 2.3854
Group relatedness: 0.0200; requested limit: 0.0200
Average within-cross relatedness: -0.1627
Objective bound: -0.1901; optimality gap: 0.0274
Maximum constraint residual: 0.0000

Chosen crosses
  Family  Parent 1  Parent 2  Progeny    Mean  Kinship
       1  1         2              100  2.3328  -0.0856
       2  1         32             100  2.2892  -0.2974
       3  2         34             100  2.4219  -0.1611
       4  3         34             100  2.1981  -0.2146
       5  5         22             100  2.4391  -0.1884
       6  6         7              100  2.4272  -0.1139
       7  6         29             100  2.3693  -0.1475
       8  11        14             100  2.7394  -0.1286
       9  11        27             100  2.5472  -0.1045
      10  11        31             100  2.2524  -0.1580
      11  14        16             100  2.3860  -0.1202
      12  14        37             100  2.3940  -0.1685
      13  22        34             100  2.5612  -0.1926
      14  30        41             100  2.2336  -0.2419
      15  38        39             100  2.1889  -0.1171
```

6.4 Simulation of the planned DH families

The simulation uses complete homozygous founders. `st.set.simpop` initializes them from the marker data set. The parental names in the plan identify the individuals copied into the two temporary parental populations. `cross` creates one F1 from each pair, and `dh` produces the planned number of doubled-haploid descendants from that F1. The seed is set before the first cross.

```
> st.set.simpop(pop.name="Founders", data.set="cross.parents")
> rng.init(seed=1001)
> founder.names <- population.pedigree("Founders")$name
> planned.crosses <- cross.plan$crosses
> for (i in seq_len(nrow(planned.crosses))) {
+     population.copy("P1", "Founders",
+                     start=match(planned.crosses$P1Name[i], founder.names),
+                     n=1)
+     population.copy("P2", "Founders",
+                     start=match(planned.crosses$P2Name[i], founder.names),
+                     n=1)
+     cross("F1", "P1", "P2", NoPg=1, crossing.scheme=3)
+     dh(sprintf("DH%03d", i), "F1",
+        NoPg=planned.crosses$n.progeny[i])
+ }
> population.remove(c("P1", "P2", "F1"))
> families <- population.list()
> families
```

```
    PopName count
1     DH015   100
2     DH014   100
3     DH013   100
4     DH012   100
5     DH011   100
6     DH010   100
7     DH009   100
8     DH008   100
9     DH007   100
10    DH006   100
11    DH005   100
12    DH004   100
13    DH003   100
14    DH002   100
15    DH001   100
16 Founders    30
```

`population.list` reports the simulation populations and their sizes. The temporary parental and F1 populations have been removed. The 15 DH families contain 100 individuals each, giving 1,500 simulated DH individuals; `Founders` retains the 30 parental lines.

`population.concat` moves the individuals from each DH family into one population named `DH`. The separate family populations are removed, while their individuals and pedigrees are retained in the merged population.

```
> for (i in seq_len(nrow(planned.crosses))) {
+     population.concat("DH", sprintf("DH%03d", i))
+ }
> merged.populations <- population.list()
> merged.populations
```

```
   PopName count
1       DH  1500
2 Founders    30
```

The simulation uses the fitted marker effects and intercept transferred by `st.set.simpop`. Genotyping and evaluation provide genetic values for the founders and the merged DH population under this same model.

```
> genotype.population("Founders DH")
> evaluate.population("Founders DH", effName="effect")
> genetic.summary <- summarize.gvalue("Founders DH", effectfile="effect")
> genetic.summary
```

```
          Obs     Mean      SDev      Min      Q10      Med      Max
Founders   30 2.187315 0.2997672 1.653617 1.799855 2.210782 2.805240
DH       1500 2.383872 0.2514697 1.650580 2.040985 2.383192 3.035688
```

The selection gain is the difference between the mean of the simulated DH population and the equal-weight mean of the 30 founder lines. It reflects the choice of crossing parents before selection among the DH individuals, under the fitted additive model.

```
> selection.gain <- genetic.summary["DH", "Mean"] -
+                   genetic.summary["Founders", "Mean"]
> selection.gain
```

```
[1] 0.1965574
```

The plan predicts the mean of the unselected DH population. Its predicted selection gain uses the same founder mean as the simulated gain. The comparison therefore concerns the same genetic model and founder reference.

```
> prediction.comparison <- data.frame(
+     Predicted = c(cross.plan$gain,
+                   cross.plan$gain - genetic.summary["Founders", "Mean"]),
+     Simulated = c(genetic.summary["DH", "Mean"], selection.gain),
+     row.names = c("Population mean", "Selection gain")
+ )
> prediction.comparison$Difference <- prediction.comparison$Simulated -
+                                     prediction.comparison$Predicted
> prediction.comparison
```

```
               Predicted Simulated   Difference
Population mean 2.3853515 2.3838720 -0.001479536
Selection gain  0.1980369 0.1965574 -0.001479536
```

The difference is the simulated value minus the prediction. Finite Mendelian sampling can make the simulated population mean differ from its expectation. This comparison precedes selection among DH lines; the usefulness criterion describes a different, within-family selection scenario.

7 Optimum cross selection using usefulness

The preceding mean criterion can be compared with usefulness while retaining the same 15 families, 100 progeny per family, candidate pairs, parental-use limits and group-relatedness bound. In this example, ten descendants are retained from each family. Equal retained counts give each family the same weight in the predicted mean of the selected descendants.

7.1 One score table for both criteria

`gs.cross.eval.mu` and `gs.cross.eval.va` evaluate the means and DH segregation variances for the complete founder panel and its existing effect model. The `N=10` and `G=100` arguments specify ten retained individuals among 100 progeny. The resulting `es` score uses the finite-family approximation of Burrows (1972); it is not an exact expectation over finite normal order statistics.

```
> gs.cross.eval.mu(data.set="cross.parents")
> gs.cross.eval.va(pop.type="DH", data.set="cross.parents")
> gs.cross.eval.es(N=10, G=100, data.set="cross.parents")
> cross.scores <- gs.cross.info(data.set="cross.parents")
```

Both analyses receive the same values from `cross.scores`. The mean criterion maximizes the average `mu` of the selected crosses. The result also reports their average usefulness. Both searches receive the same earlier feasible `cross.plan` through `control`; each can retain or improve upon that starting crossing list.

```
> mean.plan <- st.cross.plan(
+     merit = cross.merit,                    # Parental genetic values
+     K = cross.K,                            # Kinship matrix
+     n.crosses = 15,                         # Number of distinct parent pairs
+     n.progeny = 100,                        # Progeny per family
+     theta = 0.02,                           # Group relatedness limit
+     upper = 0.1,                            # Maximum parental contribution
+     scores = cross.scores,                  # Precomputed cross scores
+     criterion = "mean",                     # Objective: maximum cross mean
+     control = list(                         # Search settings
+         start = cross.plan                  # Starting crossing plan
+     )
+ )
> mean.plan
```

```
Optimum cross selection (mean)
Criterion: maximum predicted mean
Status: best.found
Families: 15; progeny per family: 100; total progeny: 1500
Expected mean: 2.3854
Expected usefulness: 2.6241; accuracy: 1.0000
Group relatedness: 0.0200; requested limit: 0.0200
Average within-cross relatedness: -0.0159
Objective bound: 2.6318; optimality gap: 0.2464
Maximum constraint residual: 0.0000

Chosen crosses
  Family  Parent 1  Parent 2  Progeny    Mean  Kinship  Usefulness
       1  1         5              100  2.2508  -0.0689      2.4677
       2  1         27             100  2.3523  -0.0447      2.5732
       3  2         6              100  2.4583  -0.0311      2.6836
       4  2         30             100  2.2902  -0.0290      2.5302
       5  3         7              100  2.1269  -0.1213      2.4293
       6  6         34             100  2.4984   0.0129      2.6800
       7  11        14             100  2.7394  -0.1286      2.9392
       8  11        22             100  2.6670   0.1692      2.8504
       9  11        38             100  2.6248  -0.0542      2.8665
      10  14        34             100  2.6336  -0.0500      2.8920
      11  14        39             100  2.3035  -0.0657      2.6186
      12  16        22             100  2.3135   0.0832      2.5542
      13  29        31             100  2.0175   0.1063      2.2565
      14  32        37             100  2.1386  -0.0825      2.4046
      15  34        41             100  2.3654   0.0664      2.6154
```

7.2 The usefulness criterion

The usefulness criterion maximizes the average cross mean plus `accuracy` times the expected selection gain, `es - mu`. The supplied value of one represents perfect within-family ranking under this fitted additive model. An accuracy of zero reduces the objective to the same mean criterion. The score computation, constraints and default search budget are identical in the two calls.

```
> usefulness.plan <- st.cross.plan(
+     merit = cross.merit,                    # Parental genetic values
+     K = cross.K,                            # Kinship matrix
+     n.crosses = 15,                         # Number of distinct parent pairs
+     n.progeny = 100,                        # Progeny per family
+     theta = 0.02,                           # Group relatedness limit
+     upper = 0.1,                            # Maximum parental contribution
+     scores = cross.scores,                  # Precomputed cross scores
+     criterion = "usefulness",               # Objective: maximum usefulness
+     accuracy = 1,                           # Within-family ranking accuracy
+     control = list(                         # Search settings
+         start = cross.plan                  # Starting crossing plan
+     )
+ )
> usefulness.plan
```

```
Optimum cross selection (usefulness)
Criterion: maximum predicted usefulness
Status: best.found
Families: 15; progeny per family: 100; total progeny: 1500
Expected mean: 2.3810
Expected usefulness: 2.6649; accuracy: 1.0000
Group relatedness: 0.0197; requested limit: 0.0200
Average within-cross relatedness: -0.0780
Objective bound: 2.8596; optimality gap: 0.1947
Maximum constraint residual: 0.0000

Chosen crosses
  Family  Parent 1  Parent 2  Progeny    Mean  Kinship  Usefulness
       1  1         22            100  2.4721  -0.0856      2.7121
       2  1         38            100  2.4299  -0.0573      2.6846
       3  2         32            100  2.3382  -0.1653      2.6422
       4  2         38            100  2.4789  -0.0825      2.7414
       5  3         11            100  2.3039  -0.0416      2.6072
       6  5         11            100  2.4457  -0.0028      2.6916
       7  6         21            100  2.4029  -0.1087      2.6662
       8  7         32            100  2.3070  -0.1223      2.6658
       9  14        30            100  2.5019   0.0507      2.7260
      10  14        34            100  2.6336  -0.0500      2.8920
      11  14        37            100  2.3940  -0.1685      2.6840
      12  16        33            100  2.0838  -0.0930      2.3953
      13  22        31            100  2.2458  -0.0605      2.5957
      14  27        41            100  2.3448  -0.0940      2.6309
      15  29        34            100  2.3329  -0.0888      2.6380
```

Both results report the predicted population mean and usefulness, making the trade-off visible for the two feasible crossing lists. Their `best.found` status describes the search result rather than a certified global optimum. The group-relatedness limit still concerns nominal parental contributions before within-family selection. It does not specify the genomic ancestry of the retained descendants.

The same crossing commands apply to `usefulness.plan$crosses`. These expected selected-descendant scores describe the fitted additive model and the declared selection scenario; they do not establish an advantage in field performance.

8 Optimum mate allocation

Optimum mate allocation assigns a continuous fraction of the produced progeny to each allowed cross (Endelman, 2025). The parental contributions follow from those mating fractions. This differs from the preceding optimum cross-selection examples, which specify 15 distinct families with 100 progeny each. Continuous mate allocation has no fixed number of families or integer progeny counts.

8.1 Mating and parental bounds

The following analysis uses the same complete founder panel, kinship matrix and full-precision cross scores. The three columns of `oma.parents` identify each parent and its minimum and maximum gene contribution. The mating table identifies both parents, the cross mean to be optimized, and its allocation bounds. Here each parent and each mating can receive at most one tenth of the corresponding total.

```
> oma.parents <- data.frame(id=rownames(cross.K), min=0, max=0.1)
> oma.mean.matings <- data.frame(
+   parent1=cross.scores$P1Name, parent2=cross.scores$P2Name,
+   merit=cross.scores$mu, min=0, max=0.1)
```

8.2 Mean-based mate allocation

`base="RM"` uses the randomly mated reference, whose value is the mean of all entries in the supplied `cross.K`. The two `dF` values specify an interval for the first-generation average within-mating kinship, called `F1`. The upper endpoint also sets the bound on the second-generation group quantity, `F2`. On this centered reference, the reference value is approximately zero, so `dF=c(0,0.01)` bounds `F1` between zero and 0.01 and `F2` by approximately 0.0199. These are a chosen reference-scale scenario, not absolute probabilities of identity by descent or a recommended biological inbreeding rate.

```
> oma.mean <- st.oma(
+     parents = oma.parents,                    # Parent contribution bounds
+     matings = oma.mean.matings,               # Allowed crosses and scores
+     K = cross.K,                              # Kinship matrix
+     dF = c(0, 0.01),                          # F1 change range; F2 change cap
+     base = "RM"                               # Random-mating reference
+ )
> oma.mean
```

```
OMA: Optimum mate allocation
Status: optimal
Expected merit: 2.3987
Reference F0: >-0.0001; F1: 0.0020
Group F2: 0.0199; upper limit: 0.0199
dF1: 0.0020; dF2: 0.0100
Merit upper bound: 2.3987; gap: <0.0001
Maximum constraint residual: <0.0001

Continuous mating allocations
  Parent 1  Parent 2  Allocation   Merit
  1         2             0.0107  2.3328
  1         3             0.0019  2.1090
  1         4             0.0016  2.1884
  1         5             0.0063  2.2508
  1         6             0.0088  2.4093
  1         7             0.0048  2.3017
  1         11            0.0148  2.4787
  1         14            0.0367  2.5445
  1         16            0.0036  2.1252
  1         21            0.0032  2.2774
```

```
[420 table rows omitted]
38        40            0.0003  2.1747
38        41            0.0023  2.4224
39        40            0.0002  1.7876
39        41            0.0006  2.0352
40        41            0.0003  2.0211
```

Interpretation of the printed summary

Status. `optimal` means that the constraint residual and merit-bound gap satisfy the numerical tolerance. `maxit` denotes an unresolved iteration or numerical limit; `infeasible` denotes a request with no feasible allocation.

Expected merit. This is the allocation-weighted mean of the supplied mating scores: cross means in this analysis and usefulness in the following analysis.

Reference F0 and F1. `F0` is the fixed reference value. Here it is the mean of the complete kinship matrix. `F1` is the allocation-weighted kinship between the parents of the allowed matings.

Group F2 and upper limit. `F2` is $c'Kc$ for the parental gene contributions implied by the mating fractions. Its upper limit follows from the supplied `dF` values and reference.

dF1 and dF2. These express changes relative to `F0` as

$$dF_1 = \frac{F_1 - F_0}{1 - F_0}, \qquad dF_2 = 1 - \sqrt{\frac{1 - F_2}{1 - F_0}}.$$

The first follows the supplied change interval, and the second is bounded by its upper endpoint.

Merit upper bound and gap. The upper bound limits the best attainable merit. The gap is its difference from the achieved merit.

Maximum constraint residual. This reports the largest numerical constraint violation used to assess feasibility.

The allocation column gives fractions of the produced progeny, summing to one. For example, 0.0107 corresponds to approximately 1.07% of the progeny. The merit column contains each mating's supplied score. These continuous fractions are not integer family sizes.

Parental gene contributions are one half of the sum of mating allocations across their two parental slots. The same parent in both slots would supply the full contribution of that self mating; this candidate table contains only distinct parents.

Different mating fractions can give the same parental contributions and the same additive mean. The mean-based optimum can therefore have several equally good mating allocations.

8.3 Usefulness-based mate allocation

The second mating table retains every pair and its bounds. Only the merit column changes to the existing expected superior-progeny score. That score was calculated above for ten retained individuals among 100 produced DH progeny per family, using the same fitted effects and finite-family approximation. The method optimizes the fixed supplied scores; it does not recalculate the selection intensity from the allocation fractions. A later integer allocation with different family sizes would require scores appropriate to those sizes. The contribution constraints refer to nominal gene shares before selection within families.

```
> oma.usefulness.matings <- oma.mean.matings
> oma.usefulness.matings$merit <- cross.scores$es
```

The second call uses the same parent and mating bounds, kinship matrix, reference and `dF` interval as the mean analysis. Its weighted merit therefore represents expected usefulness under that declared scoring model.

```
> oma.usefulness <- st.oma(
+     parents = oma.parents,                    # Parent contribution bounds
+     matings = oma.usefulness.matings,         # Allowed crosses and scores
+     K = cross.K,                              # Kinship matrix
+     dF = c(0, 0.01),                          # F1 change range; F2 change cap
+     base = "RM"                               # Random-mating reference
+ )
> oma.usefulness
```

```
OMA: Optimum mate allocation
Status: optimal
Expected merit: 2.6831
Reference F0: >-0.0001; F1: <0.0001
Group F2: 0.0199; upper limit: 0.0199
dF1: <0.0001; dF2: 0.0100
Merit upper bound: 2.6831; gap: <0.0001
Maximum constraint residual: <0.0001

Continuous mating allocations
  Parent 1  Parent 2  Allocation   Merit
  1         2            <0.0001  2.5882
  1         3            <0.0001  2.3099
  1         4            <0.0001  2.3755
  1         5            <0.0001  2.4677
  1         6            <0.0001  2.6464
  1         7            <0.0001  2.5471
  1         11           <0.0001  2.6839
  1         14            0.0227  2.7570
  1         16           <0.0001  2.3367
  1         21           <0.0001  2.5356
  [420 table rows omitted]
  38        40           <0.0001  2.5211
  38        41           <0.0001  2.7087
  39        40           <0.0001  2.1226
  39        41           <0.0001  2.3380
  40        41           <0.0001  2.3849
```

The returned allocations are continuous fractions. An operational crossing plan additionally needs integer family and progeny counts satisfying its own constraints. These fractions are not rounded automatically and are not an integer family plan for the simulation in Section 6.4. The optimum cross-selection interface remains the corresponding choice when the requested result is a list of equal-sized integer families.

9 Proportional loss of expected genetic standard deviation

The published PropSD criterion of Fernández-González et al. (2026) describes proportional loss of expected genetic standard deviation relative to a fixed reference population. The expectation follows a Gaussian genetic model with covariance proportional to the supplied matrix. Its implementation uses exact moments of sample variance and a two-moment Gamma approximation to expected sample standard deviation. This quantity differs from the standard deviation of the fitted parental merits and from group relatedness. Appendix A gives the mathematical definitions.

The complete 30-parent matrix defines the reference sample here. Each family contributes two parental occurrences to the selected sample, including repeated occurrences of a parent used in several families. Such occurrences represent the same genetic random variable. They are not independent copies.

```
> parent.slots <- c(usefulness.plan$crosses$P1Name,
+                   usefulness.plan$crosses$P2Name)
> diversity.loss <- propsd(
+     G = cross.K,                              # Complete reference covariance
+     parents = parent.slots                    # Parental occurrences in families
+ )
> diversity.loss
```

```
Proportional loss of expected sample SD
PropSD: 0.0256
Expected SD: reference 1.0044; selected 0.9788
Sample occurrences: reference 30.0000; selected 30.0000
Gaussian moments; two-moment Gamma approximation
```

The loss is one minus the selected-to-reference ratio of approximate expected standard deviations. A value of 0.03 represents a 3% loss under this model; negative values are possible when the selected sample has greater expected variation. The reference remains fixed when the selected occurrences change. The reported approximation does not describe realized ancestry or variation after selection within the descendant families.

9.1 A diversity constraint in cross selection

The optional diversity list constrains the same numerical estimate during cross selection. The explicit `family.slots` declaration specifies the sample convention used above. The complete matrix can also contain reference parents outside the eligible crossing panel. The objective remains usefulness and the existing contribution and group-relatedness limits still apply.

```
> diversity.plan <- st.cross.plan(
+     merit = cross.merit,                      # Parental genetic values
+     K = cross.K,                              # Kinship matrix
+     n.crosses = 15,                           # Number of distinct parent pairs
+     n.progeny = 100,                          # Progeny per family
+     theta = 0.02,                             # Group relatedness limit
+     upper = 0.1,                              # Maximum parental contribution
+     scores = cross.scores,                    # Precomputed cross scores
+     criterion = "usefulness",                 # Objective: maximum usefulness
+     accuracy = 1,                             # Within-family ranking accuracy
+     diversity = list(                         # Additional diversity constraint
+         G = cross.K,                          # Complete reference covariance
+         max = 0.025,                          # Strict upper limit on PropSD
+         sample = "family.slots"               # Two parental occurrences per family
+     ),
+     control = list(                           # Search settings
+         starts = 4,                           # Number of search starts
+         maxit = 5000                          # Proposals per search start
+     )
+ )
> diversity.plan
```

```
Optimum cross selection (usefulness)
Criterion: maximum predicted usefulness
Status: best.found
Families: 15; progeny per family: 100; total progeny: 1500
Expected mean: 2.3730
Expected usefulness: 2.6597; accuracy: 1.0000
Group relatedness: 0.0197; requested limit: 0.0200
PropSD (family slots): 0.0148; strict upper limit: 0.0250
Average within-cross relatedness: -0.0601
Objective bound: 2.8596; optimality gap: 0.1999
Maximum constraint residual: 0.0000
```

```
Chosen crosses
  Family  Parent 1  Parent 2  Progeny    Mean  Kinship  Usefulness
       1  1         14            100  2.5445   0.0570      2.7570
       2  1         38            100  2.4299  -0.0573      2.6846
       3  2         21            100  2.3264  -0.0940      2.6401
       4  2         34            100  2.4219  -0.1611      2.6972
       5  3         30            100  2.0664  -0.0510      2.3144
       6  5         22            100  2.4391  -0.1884      2.7195
       7  6         11            100  2.6042  -0.0972      2.8350
       8  7         32            100  2.3070  -0.1223      2.6658
       9  11        38            100  2.6248  -0.0542      2.8665
      10  14        40            100  2.2893  -0.0594      2.6583
      11  14        41            100  2.5370  -0.0867      2.8071
      12  22        37            100  2.3215   0.1293      2.6162
      13  27        34            100  2.4414   0.0685      2.7007
      14  28        32            100  2.0388  -0.1905      2.3958
      15  29        33            100  2.2024   0.0056      2.5369
```

The diversity line reports the computed loss and its strict upper limit. Feasibility requires the displayed estimate before rounding to be below 0.025. The reported search status retains its usual meaning: a feasible heuristic result does not establish a global optimum. The constraint applies to the declared parental sample and the Gamma estimate, rather than a guaranteed loss of genetic variation in a future selected population.

10 Allocation of progeny to fixed crosses

The parent pairs can remain fixed while the produced progeny counts vary. The following allocation distributes 1,500 progeny across the 15 families and retains ten individuals per family. The mean of the selected descendants therefore weights each family by its ten retained individuals. Selection intensity is recalculated for every proposed integer family size from the cross mean and segregation variance; an earlier score at 100 progeny is not reused at other family sizes.

```
> progeny.allocation <- st.cross.allocate(
+     plan = diversity.plan,                    # Fixed parent pairs and constraints
+     scores = cross.scores,                    # Cross means and segregation variances
+     total.progeny = 1500,                     # Total produced progeny budget
+     n.selected = 10,                          # Retained descendants per family
+     min.progeny = 10,                         # Minimum produced count per family
+     max.progeny = 1500,                       # Maximum produced count per family
+     accuracy = 1                              # Within-family ranking accuracy
+ )
> progeny.allocation
```

```
Fixed-pair progeny allocation
Status: optimal
Families: 15; produced: 1500; retained: 150
Predicted retained mean: 2.661; response: 0.2876
Objective upper bound: 2.661; response gap: 6.661e-16
Group relatedness, nominal retained: 0.01975; produced: 0.01939
Family: parents; produced / retained; predicted retained mean
1: 1 x 14; 78 / 10; 2.742
2: 1 x 38; 91 / 10; 2.678
3: 2 x 21; 108 / 10; 2.647
4: 2 x 34; 97 / 10; 2.695
5: 3 x 30; 89 / 10; 2.306
6: 5 x 22; 98 / 10; 2.718
7: 6 x 11; 84 / 10; 2.824
8: 7 x 32; 121 / 10; 2.684
9: 11 x 38; 87 / 10; 2.857
10: 14 x 40; 124 / 10; 2.68
```

```
11: 14 x 41; 95 / 10; 2.803
12: 22 x 37; 102 / 10; 2.618
13: 27 x 34; 92 / 10; 2.695
14: 28 x 32; 120 / 10; 2.413
15: 29 x 33; 114 / 10; 2.549
```

The family table gives produced and retained counts together with the recalculated expected selected-family means. The expected selected mean combines them using retained counts. The budget and integer bounds refer to produced counts. The optimum and numerical gap refer to allocation among these fixed pairs under the finite-family approximation; they do not establish a joint optimum over parent pairs and progeny numbers.

The inherited parental contribution and group-relatedness constraints use nominal retained-family gene shares. Produced-progeny shares and their group relatedness are separate reported quantities. Neither vector predicts the actual genomic ancestry after within-family selection. The family-slot diversity sample remains unchanged when only produced counts vary.

The low-level simulation commands in Section 6.4 already read the produced count of each family from `planned.crosses`. The assignment `planned.crosses <- progeny.allocation$crosses` therefore provides the variable counts to that same crossing and DH-generation loop. The equal-family execution adapter retains its equal-size requirement.

10.1 Comparison with simulated selected descendants

A separate repeated-simulation check uses the same 30 founders, 243 markers, fitted effects and fixed pairs as this allocation. Each of 200 paired cycles produces 1,500 DH descendants under each allocation and retains the ten highest genetic values per family. The table reports the predicted selected mean, simulated mean across cycles, and Monte Carlo standard error (MCSE). The difference uses the paired cycles.

Quantity	Predicted	Simulated	MCSE
Equal families	2.659679	2.655875	0.000516
Allocated families	2.660583	2.656752	0.000489
Allocation difference	0.000904	0.000878	0.000453

The simulated allocation difference is positive, but its 95% Monte Carlo interval, approximately $[-0.000009,\ 0.001765]$, includes zero. This run therefore does not establish a positive allocation benefit at that precision. Both simulated selected means are about 0.0038 below their normal/Burrows predictions. That discrepancy concerns the selection approximation for this fitted-effect model, rather than a failure of the integer allocation optimum. The same marker effects score the predictions and simulations, so the comparison assesses model consistency and does not provide independent field validation. These repeated simulations are validation work, separate from the ordinary short companion example.

A Mathematical definitions and implemented algorithms

The definitions below distinguish parental merit, within-family segregation variance, nominal parental contribution and covariance-based diversity. All relationship matrices retain the reference population used to construct them. Numerical convergence describes the stated model and supplied scores; it does not establish the accuracy of the fitted marker effects. Equation numbers below refer to the cited publications. The sample conventions, constraints and numerical safeguards specific to SelectionTools are identified separately from the established statistical formulas and algorithmic principles.

A.1 Marker prediction, segregation variance and usefulness

For the biallelic additive model, $z_{i\ell} \in \{0,1,2\}$ is the dosage of the effect allele at marker ℓ in parent i, a_0 is the fitted intercept and a_ℓ is its marker effect. The parental prediction and expected cross mean are

$$g_i = a_0 + \sum_\ell a_\ell z_{i\ell}, \qquad \mu_{ij} = a_0 + \frac{1}{2}\sum_\ell a_\ell (z_{i\ell} + z_{j\ell}).$$

The cross mean is the dosage-notation form of Osthushenrich et al. (2017), Equations (1) and (4). Thus $\mu_{ij} = (g_i + g_j)/2$ for the same additive model. The score table retains the cross evaluator's supplied values; the planner does not replace them by values reconstructed from exported parental predictions.

For two completely homozygous parents, the marker contribution difference is $\delta_{ij,\ell} = a_\ell(z_{i\ell} - z_{j\ell})/2$. The linked segregation variance used for DH or completely homozygous SSD lines is

$$v_{ij} = \sum_c \sum_{\ell,k \text{ on } c} \rho_{\ell k}\, \delta_{ij,\ell}\delta_{ij,k}, \qquad \rho_{\ell k} = 2q_{\ell k} - 1.$$

This covariance contraction rewrites Osthushenrich et al. (2017), Equations (2) and (13), using the matching-origin relation in their Equation (16). The sum includes both orders of each marker pair and the diagonal. Different chromosomes have zero covariance. Here $q_{\ell k}$ is the probability that the two loci have matching parental origin. With recombination frequency r and t additional random-intermating generations, the implemented correlations are

$$\rho_{\mathrm{DH}}(r,t) = (1-2r)(1-r)^t, \qquad \rho_{\mathrm{SSD}}(r,t) = \frac{1-2r}{1+2r}(1-r)^t.$$

The DH and SSD correlations are $2q-1$ from Osthushenrich et al. (2017), Equations (20) and (23), respectively; their Equations (19) and (22) give the underlying linkage disequilibria. The SSD expression concerns the limit under repeated selfing; t is not the number of selfing generations. The map distance d in Morgans gives $r = (1-\exp(-2d))/2$ under Haldane or $r = \tanh(2d)/2$ under Kosambi. The first is the inverse of Osthushenrich et al. (2017), Equation (24), following Haldane (1919). The second is Kosambi (1943), Equation (6), p. 173. The unlinked inbred-parent calculation reduces to $\sum_\ell \delta_{ij,\ell}^2$. The C implementation forms the chromosome-specific recombination and origin-probability matrices and contracts them with the parental marker contributions, following Osthushenrich et al. (2017).

Normal truncation and its finite-family correction follow Burrows (1972, pp. 1091–1100). For retention fraction $p \in (0,1)$, normal truncation intensity is

$$i(p) = \frac{\phi\{\Phi^{-1}(1-p)\}}{p},$$

where ϕ and Φ are the standard normal density and distribution function. With R retained individuals from G produced individuals, the implemented Burrows (1972) finite-family approximation is

$$i_B(R,G) = i(R/G) - \frac{1-R/G}{2R(1+1/G)i(R/G)}, \qquad 1 \le R < G.$$

The boundary definitions are $i(1) = i_B(G,G) = 0$. Family usefulness with selection accuracy $a \in [0,1]$ is

$$U_{ij} = \mu_{ij} + a\, i_B(R,G)\sqrt{v_{ij}}.$$

The usefulness criterion of Schnell and Utz (1975) combines the family mean and a within-family selection differential, as described by Osthushenrich et al. (2017, Introduction, pp. 1–2). Zhong and Jannink (2007, Introduction) discuss the corresponding superior-progeny criterion. The accuracy-scaled form above also appears in Fernández-González et al. (2026), Equation (16), with the intensity here supplied by the Burrows calculation. The SD, rather than the variance, enters the differential. The infinite population option uses $i(p)$ instead. The fixed-score planner uses $\mu_{ij} + a(es_{ij} - \mu_{ij})$ from its common supplied table. Changing its common progeny count does not recompute es_{ij}. Variable-size allocation uses μ_{ij} and v_{ij} to recompute the differential for each size.

A.2 Parental contributions and equal-sized integer families

For marker frequencies p_ℓ in the fixed reference, the genomic relationship matrix in this example is

$$W_{i\ell} = z_{i\ell} - 2p_\ell, \qquad A = \frac{WW'}{2\sum_\ell p_\ell(1-p_\ell)}, \qquad K = A/2.$$

This is the first genomic relationship construction of VanRaden (2008, p. 4416; unnumbered first-method formula), written for dosage coding $0,1,2$ rather than $-1,0,1$. The denominator must be positive. No diagonal adjustment is applied. The matrix can be singular and its off-diagonal entries can be negative. The quadratic form $Q(c) = c'Kc$ is a reference-dependent group-relatedness index. Dividing A by two does not itself give an absolute probability of identity by descent; Endelman (2025, Discussion following Equation (22)) also distinguishes genomic correlations from identity-by-descent probabilities.

The merit and coancestry terms in optimum contribution selection follow Meuwissen (1997), Equations [1] and [3], with $K = A/2$. His sex-specific normalizations in Equation [2] become one simplex for the supplied plant-parent pool here, with the stated lower and upper bounds. The problem is

$$\max_c g'c \quad \text{subject to} \quad \mathbf{1}'c = 1, \quad l \le c \le u, \quad Q(c) \le \theta.$$

The inner C solver minimizes $Q(c) - 2tg'c$ over the box-constrained simplex. This scalar tradeoff has the Lagrangian form of Meuwissen (1997), Equation [4]; the reduced-coordinate solver described here is the SelectionTools implementation. An active-set method uses a basis of free directions whose components sum to zero, an eigendecomposition of the reduced quadratic form, and a bounded line search. Rank-deficient directions and active bounds receive explicit treatment. The outer search brackets and bisects the merit–relatedness tradeoff. A separate minimum-relatedness problem tests feasibility; the minimum-relatedness boundary also has an affine rank-based calculation.

For any candidate x, $\lambda \ge 0$, and box-constrained simplex C, convexity gives the merit upper bound

$$g'c \le \lambda\{\theta + Q(x)\} + \max_{y \in C}(g - 2\lambda Kx)'y.$$

The remaining maximization is a bounded linear allocation. Feasibility residuals and the difference between such a bound and the achieved merit govern the reported numerical qualification; an iteration limit does not establish an optimum. The displayed bound follows by applying the first-order convexity inequality of Boyd and Vandenberghe (2004), Equation (3.2), p. 69, to Q and using $Q(c) \le \theta$.

For M equal-sized distinct non-selfed families, $x_e \in \{0,1\}$ selects an allowed unordered pair $e = (i_e, j_e)$. The parental uses and nominal gene contributions are

$$\sum_e x_e = M, \qquad d_i = \sum_e x_e\{\mathbf{1}(i_e = i) + \mathbf{1}(j_e = i)\}, \qquad c_i = \frac{d_i}{2M}.$$

Joint planning maximizes $M^{-1}\sum_e x_e h_e$, with h_e the additive cross mean or supplied usefulness, subject to contribution bounds and $Q(c) \le \theta$. Without cross scores its mean objective equals $g'c$. Fixed-contribution planning instead minimizes $M^{-1}\sum_e x_e K_{i_e j_e}$ while preserving every d_i. This is the kinship-minimizing mate-allocation objective described by Wellmann (2019, section “Mate allocation”; unnumbered objective), with the distinct-pair and integer-usage conventions stated here. A continuous contribution vector is therefore insufficient unless all its uses are integer-realizable in the allowed distinct-pair graph.

The exact search uses the branch-and-bound principle of Land and Doig (1960, Section 3), implemented here by include/exclude branches in score order. Available incident edges and remaining parental capacity prune impossible branches; sums of the best remaining edge scores bound objective values. Continuous contribution relaxations provide additional bounds when applicable. Automatic method selection compares an estimated enumeration workload with the node budget. The alternative heuristic combines multiple initial constructions with local edge changes and a local random-number stream. Fixed-contribution changes rewire two or three pairs while preserving all parental uses. Every retained incumbent is rechecked from its complete selected pairs and original matrix. The exact method can establish optimality after a completed search; a feasible heuristic incumbent has status `best.found`. An unresolved boundary or exhausted budget does not establish infeasibility. Relatedness is recomputed using integer uses and the original matrix, with numerical allowance

$$\varepsilon \max\left\{|\theta|, \sum_{i,j}|K_{ij}|c_i c_j\right\},$$

where ε is `control$tolerance`. Its computed roundoff enclosure must satisfy the bound including this allowance. A zero limit additionally requires a row-residual check independent of ε. These tolerance, roundoff and status rules are implementation safeguards.

A.3 Continuous optimum mate allocation

The continuous model follows the diploid mate-allocation framework of Endelman (2025, section "Convex optimization"). For allowed pairs e, mating fractions x_e are continuous and sum to one. The parental incidence matrix has entries $B_{ie} = \{\mathbf{1}(i_e = i) + \mathbf{1}(j_e = i)\}/2$, so $c = Bx$. This is the allowed-pair incidence representation of Endelman (2025), Equation (8). A self mating, when present in this interface, contributes one to its parent's incidence entry. With supplied mating scores h_e, `st.oma` maximizes $h'x$ under the supplied bounds on x and c. Its two kinship quantities are

$$F_1 = \sum_e x_e K_{i_e j_e}, \qquad F_2 = (Bx)'K(Bx).$$

These are the diploid specializations of Endelman (2025), Equations (14) and (19), with random mating in the following generation for F_2. The complete supplied $N \times N$ matrix defines the fixed reference

$$F_0 = \frac{1}{N}\sum_i (2K_{ii} - 1) \quad \text{for } \texttt{base="current"}, \qquad F_0 = \frac{1}{N^2}\sum_{i,j} K_{ij} \quad \text{for } \texttt{base="RM"}.$$

The current-population expression follows the diploid form of Endelman (2025), Equation (22), with $A = 2K$. The random-mating reference specializes his Equation (16) to uniform contributions. Using the complete supplied reference before restriction to eligible parents is the stated interface convention. The reference must be below one. For `dF=c(dL,dU)`, the constraints are

$$F_0 + (1 - F_0)d_L \le F_1 \le F_0 + (1 - F_0)d_U,$$
$$F_2 \le F_0 + (1 - F_0)d_U(2 - d_U).$$

A scalar `dF` sets an equality for F_1. The reported changes are

$$dF_1 = \frac{F_1 - F_0}{1 - F_0}, \qquad dF_2 = 1 - \sqrt{\frac{1 - F_2}{1 - F_0}}.$$

The bounds and rate expressions are algebraic rearrangements of Endelman (2025), the one- and two-generation recurrences in Equations (10)–(11). The linear F_1 and quadratic F_2 restrictions remain distinct.

The C implementation uses a primal–dual interior-point method with the predictor–corrector principle of Mehrotra (1992). Positive inequality slacks and multipliers enter the perturbed complementarity equations. An affine predictor determines the centering parameter; the corrector includes the predictor's second-order product. A fraction-to-boundary rule and residual line search control the update. The Newton calculation uses a symmetric system of parental dimension, with the quadratic-constraint multiplier represented explicitly, and residual refinement. Scaling and rank decisions aid this calculation; feasibility is checked against the original supplied matrix and limits. Separate merit bounds and constraint residuals govern the status. The parent-dimension Newton formulation, residual checks and status qualification describe its implementation for this problem. Continuous mating fractions are not rounded into integer families.

A.4 Covariance-based proportional loss of SD

The `propsd` evaluator follows the published PropSD estimator of Fernández-González et al. (2026), Equations (14)–(15) and Appendix C, Equations (C1)–(C6). For the complete reference covariance-proportional matrix G, the model is $u \sim N(0, \tau^2 G)$. A literal selected sample $s = (s_1, \ldots, s_m)$ has covariance $G_s = G[s, s]$. Repeated identifiers repeat the same random genetic value; they do not create independent copies. The marker-frequency reference and the complete G remain fixed.

For any $n \times n$ covariance matrix V, $n \ge 2$, the centered matrix and ordinary sample variance with denominator $n - 1$ are

$$H_n = I_n - \mathbf{1}\mathbf{1}'/n, \qquad C = H_n V H_n, \qquad S^2 = \frac{z' H_n z}{n-1}, \quad z \sim N(0, V).$$

The standard Gaussian quadratic-form identities of Magnus (1978), Lemmas 2.2–2.3, pp. 203–204, applied to $H_n/(n-1)$, give the exact moments

$$\mu_V = E(S^2) = \frac{\mathrm{tr}(C)}{n-1}, \qquad v_V = \mathrm{Var}(S^2) = \frac{2\sum_{i,j} C_{ij}^2}{(n-1)^2}.$$

These are also Fernández-González et al. (2026), Equation (15) and Appendix C, Equations (C1)–(C4), in centered-matrix notation. Their two-moment Gamma approximation, Equation (C5), has shape $\alpha_V = \mu_V^2/v_V$ and scale $\beta_V = v_V/\mu_V$. Its half moment approximates the expected sample SD:

$$e(V) = \sqrt{\beta_V}\,\frac{\Gamma(\alpha_V + 1/2)}{\Gamma(\alpha_V)}, \qquad \widehat{\mathrm{PropSD}} = 1 - \frac{e(G_s)}{e(G)}.$$

The Gamma half moment follows from Euler's integral (NIST DLMF, Equation 5.2.1), as worked out in Fernández-González et al. (2026), Equation (C6) and its following simplification. The common τ cancels. A general Gaussian quadratic form is a weighted sum of chi-squares, so Gamma moment matching is an approximation; it is exact when all nonzero eigenvalues of C are equal, including rank-one contrasts. The estimator is a ratio of expected SDs, rather than the expectation of a random ratio.

Integer multiplicities avoid materializing a replicated covariance matrix. With counts $k_i \geq 0$, $m = \sum_i k_i \geq 2$, $w = k/m$, $r = Gw$, and $D_{ij} = G_{ij} - r_i - r_j + w'Gw$, the same moments are

$$\mu_s = \frac{\sum_i k_i D_{ii}}{m-1}, \qquad v_s = \frac{2\sum_{i,j} k_i k_j D_{ij}^2}{(m-1)^2}.$$

These count expressions are an algebraic regrouping of Equation (15) of Fernández-González et al. (2026), with identical repeated-parent rows grouped together. Only positive-count parents enter the selected-sample sums. The total m matters as well as the normalized counts. Continuous contribution fractions alone therefore do not specify this statistic.

The C evaluator checks positive semidefiniteness. Each reference or selected sample uses its own active parental support to form anchored contrasts directly from the original supplied covariance entries, followed by occurrence-weighted centering and local contrast scaling. This preserves small selected-sample contrasts even when the reference contains much larger variances. The reference covariance model remains fixed; the local numerical scaling does not reconstruct marker frequencies. Extended-precision moment sums and separate reference and selected scale factors reduce cancellation and overflow. Moments reported in original covariance units can still exceed the numeric range of R. The SD calculation is

$$e(V) = \sqrt{\mu_V}\exp\{\ell(\alpha_V)\}, \qquad \ell(\alpha) = \log\Gamma(\alpha + 1/2) - \log\Gamma(\alpha) - \tfrac{1}{2}\log\alpha.$$

For $\alpha > 128$, the implemented large-shape expansion is $\ell(\alpha) \simeq -1/(8\alpha) + 1/(192\alpha^3) - 1/(640\alpha^5)$; these coefficients specialize NIST DLMF, Equation 5.11.8, by subtracting the $h = 0$ expansion from the $h = 1/2$ expansion. The cutoff at 128 is an implementation choice; smaller shapes use the log-Gamma functions. A positive reference contrast variance is required. A selected sample with zero contrast variance has $e(G_s) = 0$ and loss one; a one-occurrence sample has undefined sample SD. Negative loss is retained when the selected expected SD exceeds the reference expected SD. Common positive covariance scaling and addition of a common random-intercept covariance leave the ratio unchanged when the covariance matrices remain valid.

The optional cross-planner convention `sample="family.slots"` assigns one occurrence to each parental slot of every selected family, so $k_i = d_i$ and $m = 2M$. The complete supplied diversity matrix defines the reference, including any additional reference parents. The constraint is the strict comparison $\widehat{\mathrm{PropSD}} < b$, where b is `diversity$max`. The direction and strictness of this cutoff follow Fernández-González et al. (2026), Equation (18). It constrains the computed Gamma estimate; it is not an interval proof of biological diversity loss. Trait merits, $c'Kc$ and realized genomic ancestry after within-family selection remain different quantities. The exact search checks the criterion at complete candidate plans; objective bounds obtained by omitting it remain valid relaxations. The joint heuristic includes both relatedness and diversity violations when seeking feasibility and rechecks the criterion on retained plans. Fixed-contribution rewiring leaves the family-slot multiplicities, and hence this diversity criterion, unchanged.

A.5 Integer progeny allocation for fixed families

The produced counts G_e vary while parent pairs and retained counts R_e remain fixed. With total budget T, the integer constraints are

$$L_e \le G_e \le U_e, \qquad G_e \ge R_e \ge 1, \qquad \sum_e G_e = T.$$

Repeated unordered pairs are pooled into one biological family before selection intensity is calculated. Their retained counts and produced bounds are summed, and they have one common prediction accuracy a_e. The retained-population mean is

$$\overline{U}(G) = \sum_e \frac{R_e}{R_+}\{\mu_e + a_e\sqrt{v_e}\, i_B(R_e, G_e)\}, \qquad R_+ = \sum_e R_e.$$

For equal retained counts this is the mean-family usefulness objective of Fernández-González et al. (2026), Equation (17). The displayed retained-count weights and pooling convention define the allocation problem implemented here. The weighting follows retained individuals rather than produced counts. The constant $\sum_e R_e\mu_e/R_+$ is excluded from allocation comparisons, which protects small response differences against large common merit offsets.

For family response utility $u_e(g) = (R_e/R_+)a_e\sqrt{v_e}\, i_B(R_e, g)$, exact dynamic programming uses the finite-stage principle of optimality of Bellman (1954, Section 2 and the basic recurrence in Equation (3.3), pp. 504–505), specialized to the remaining progeny budget:

$$D_0(0) = 0, \quad D_0(b \ne 0) = -\infty, \qquad D_e(b) = \max_{L_e \le g \le U_e}\{D_{e-1}(b-g) + u_e(g)\}.$$

The C recurrence works with additional counts above the family minima. Reachable budget ranges restrict states. Two objective rows and a family-by-budget traceback table recover the integer allocation. With $T' = T - \sum_e L_e$, the worst-case time is $O(M(T'+1)W)$ for maximum count-range width W; traceback storage is $O(M(T'+1))$. No concavity assumption is needed for this enumeration of feasible counts. The shared C intensity kernel also serves the existing finite-family cross evaluator. Upward-rounded recurrence sums give an upper bound on the finite computed utility table; a downward sum for the traced allocation gives the corresponding numerical gap. These bounds do not make the normal-selection or Burrows approximation exact. A completed calculation reports `optimal` only when its gap satisfies the numerical tolerance; an unresolved gap has status `numerically.unresolved`. A resource cap retains a deterministic feasible allocation with `resource.limit` and an infinite unresolved bound.

The same diploid parental-incidence relation as Endelman (2025), Equation (8), gives the nominal retained and produced shares separately

$$c_i^{\text{ret}} = \frac{\sum_e R_e\{\mathbf{1}(i_e = i) + \mathbf{1}(j_e = i)\}}{2R_+}, \qquad c_i^{\text{prod}} = \frac{\sum_e G_e\{\mathbf{1}(i_e = i) + \mathbf{1}(j_e = i)\}}{2T}.$$

The inherited parental bounds and relatedness limit apply to c^{ret}. Both relatedness values are reported; the produced one is a diagnostic in this allocation model. A family-slot PropSD constraint remains fixed because the parent pairs remain fixed. Neither nominal vector predicts the genomic ancestry of selected descendants. The optimum is conditional on the supplied pairs and retained counts, rather than a joint global optimum over pairs and progeny resources.

References

Bellman R (1954) The theory of dynamic programming. Bulletin of the American Mathematical Society 60:503–515.

Boyd S, Vandenberghe L (2004) Convex Optimization. Cambridge University Press, Cambridge.

Burrows PM (1972) Expected selection differentials for directional selection. Biometrics 28:1091–1100

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

Endelman JB (2025) Genomic prediction of heterosis, inbreeding control, and mate allocation in outbred diploid and tetraploid populations. Genetics 229:iyae193.

Fernández-González J, Metwally SM, Isidro y Sánchez J (2026) MateR: a novel genomic mating framework. Genetics 232:iyag013. doi:10.1093/genetics/iyag013

Haldane JBS (1919) The combination of linkage values, and the calculation of distances between the loci of linked factors. Journal of Genetics 8:299–309.

Kosambi DD (1943) The estimation of map distances from recombination values. Annals of Eugenics 12:172–175.

Land AH, Doig AG (1960) An automatic method of solving discrete programming problems. Econometrica 28:497–520.

Magnus JR (1978) The moments of products of quadratic forms in normal variables. Statistica Neerlandica 32:201–210.

Mehrotra S (1992) On the implementation of a primal-dual interior point method. SIAM Journal on Optimization 2:575–601.

Meuwissen THE (1997) Maximizing the response of selection with a predefined rate of inbreeding. Journal of Animal Science 75:934–940.

NIST Digital Library of Mathematical Functions. Chapter 5, Gamma function, Equations 5.2.1 and 5.11.8. National Institute of Standards and Technology. https://dlmf.nist.gov/5 (accessed 12 November 2026).

Osthushenrich T, Frisch M, Herzog E (2017) Genomic selection of crossing partners on basis of the expected mean and variance of their derived lines. Plos One 12(12):e0188839

Reif JC, Melchinger AE, Frisch M (2005) Genetical and Mathematical Properties of Similarity and Dissimilarity Coefficients Applied in Plant Breeding and Seed Bank Management. Crop Science 45:1-7

Schnell FW, Utz HF (1975) F_1-Leistung und Elternwahl in der Züchtung von Selbstbefruchtern. Bericht über die Arbeitstagung der Vereinigung österreichischer Pflanzenzüchter, Gumpenstein, pp. 243–248.

VanRaden PM (2008) Efficient methods to compute genomic predictions. Journal of Dairy Science 91:4414–4423.

Wellmann R (2019) Optimum contribution selection for animal breeding and conservation: the R package optiSel. BMC Bioinformatics 20:25

Zhong S, Jannink JL (2007) Using quantitative trait loci results to discriminate among crosses on the basis of their progeny mean and variance. Genetics 177:567–576