Marker-Assisted Backcrossing

Matthias Frisch

1 Low-level MABC interface **2**

1.1 Baseline backcrossing and marker density . . . 2
1.2 Foreground selection for the target allele . . . 4
1.3 Two-stage selection . . . 5
1.4 Three-stage selection . . . 7

2 High-level MABC workflow **9**

2.1 Input data and terminology . . . 9
2.1.1 Input directories and result files . . . 9
2.1.2 Linkage-map files . . . 10
2.1.3 Parental marker files . . . 10
2.1.4 Generation types . . . 12
2.1.5 Selection strategies . . . 12
2.1.6 Counting marker analyses . . . 13
2.2 `mab.simulate()` . . . 13
2.3 `mab.load.data()` and `mab.tabulate()` . . . 13
2.4 `mab.compare()` . . . 16
2.5 Interpreting MABC results . . . 16
2.6 Planning comparisons . . . 17

3 Advanced high-level examples **17**

3.1 Example 1: six backcross generations with target selection . . . 17
3.2 Example 2: two-stage selection . . . 18
3.3 Example 3: increasing population sizes and three-stage selection . . . 20
3.4 Example 4: comparing population sizes . . . 22
3.5 Example 5: simultaneous introgression of two genes . . . 24
3.6 Example 6: selection strategy and linkage drag . . . 26
3.7 Example 7: introgression of three genes . . . 27
3.8 Example 8: observed marker data for donor and recurrent parent . . . 29
3.9 Example 9: different BC1 plants as donor parent . . . 31
3.10 Example 9a: analysis of input individuals without further simulation . . . 32
3.11 Example 10: selection for a low number of donor segments . . . 34
3.12 Example 11: analysis of chromosome regions . . . 35
3.13 Example 12: preselection for recurrent-parent alleles . . . 37
3.14 Example 13: success factor . . . 41
3.15 Example 14: recoding parental input files . . . 42
3.16 Example 15: minimum number of preselected individuals . . . 44

1 Low-level MABC interface

SelectionTools provides two interfaces for marker-assisted backcrossing (MABC). The low-level interface uses the population-simulation functions of SelectionTools and therefore exposes every crossing, selection and sampling step. The high-level interface, implemented by the `mab.*` functions, describes a complete backcross program by vectors of generation types, population sizes and selection strategies. Both interfaces use the same simulation engine.

The examples in this vignette are based on the MABC studies of Frisch, Bohn and Melchinger (1999) and Frisch and Melchinger (2001). They illustrate recovery of the recurrent-parent genome, linkage drag, marker-data requirements, simultaneous introgression of several target genes, and the use of observed parental or backcross marker genotypes.

The low-level interface is appropriate when individual operations of a breeding program must be controlled explicitly. At this level, foreground selection, flanking-marker selection and genome-wide background selection are separate commands. This makes the interface useful for developing new breeding schemes or for studying exactly which individuals enter each selection stage.

The low-level examples use the package data sets `lib00.map1`, `lib00.map1a`, and `lib00.map2`. They reproduce the structure of the simulations underlying Tables 1–3 of Frisch et al. (1999). In the low-level interface the donor allele is coded as 1 and the recurrent-parent allele as 8.

Simulation size and comparison with published results. The simulation studies of Frisch, Bohn and Melchinger (1999) and Frisch and Melchinger (2001) used 10000 replicate breeding programs. The examples use `NREP <- 10000` for the same sampling precision; the unselected baseline also contains 10000 individuals per generation. Repetitions are distinct from the number of plants per breeding generation. The latter controls selection intensity and is specified separately by `population.size` or the population-size argument of `cross()`. Increasing repetitions reduces sampling variation without changing the breeding scheme. Comparable published tables also require the same map, selection strategy and generations; identical numerical entries are not expected from independently seeded simulations.

Simulation output and random-number state. SelectionTools uses clock-based initialization by default. For reproducible simulations, `rng.init(seed=1001)` initializes R's random-number state and uses R's generator to supply seeds for the internal simulation generators. A call to `set.seed()` alone does not initialize those internal generators. Calling `rng.init()` without a seed returns to clock-based initialization.

The examples select one thread and initialize the generators before simulating. Repeating the same seed, inputs and sequence of calls gives identical results with the same actual thread count, R random-number generator settings, package build and numerical libraries. Results need not agree across different thread counts or platforms. The blue boxes contain output from these seeded runs.

1.1 Baseline backcrossing and marker density

Without selection, repeated backcrossing to the recurrent parent provides the baseline increase in RPG. The original example also compared a map with approximately 80 markers with a reduced map containing approximately 40 markers. Marker effects are scaled so that the marker score estimates the recurrent-parent genome proportion.

```
> library(SelectionTools)
> options(width=72, digits=5)
> st.set.num.threads(1)
> st.output.dir <- file.path(tempdir(), "SelectionTools-masbc")
```

```
> st.data.dir <- st.output.dir
> dir.create(st.output.dir, recursive=TRUE, showWarnings=FALSE)
> NREP <- 10000
> rng.init(seed=1001)
> reset.all()
```

The call to `rng.init(seed=1001)` initializes both R and the internal random-number generators, and `reset.all()` removes state left by previous low-level simulations. The repeated setup is intentional so this example can be run independently.

```
> data("lib00.map1")
> data("lib00.map1a")
> write.table(lib00.map1, file=file.path(st.output.dir,"mab-1.map"),
+             quote=FALSE, row.names=FALSE, col.names=FALSE)
> write.table(lib00.map1a, file=file.path(st.output.dir,"mab-1a.map"),
+             quote=FALSE, row.names=FALSE, col.names=FALSE)
> set.genome.par(
+     no.chrom=10,
+     no.hom=2,
+     chrom.len=c(2.68,1.81,2.08,1.40,1.53,
+                 1.56,0.93,1.44,1.55,1.14)
+ )
> load.linkage.map(file.path(st.output.dir,"mab-1.map"))
```

The package maps are written to temporary files because the low-level linkage-map loader expects a file. The genome definition supplies the ten maize chromosome lengths used in the original MABC examples before the denser map is loaded.

```
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(8))
> cross("F1","P1","P2",NREP)
> gen <- c("P1","F1","BC1","BC2","BC3","BC4","BC5","BC6")
> for (i in 1:7)
+     cross(gen[i+1],gen[i],"P2",NREP)
> rpg <- genome.contribution("BC1 BC2 BC3 BC4 BC5 BC6",8)
> define.effects("markers80","0, mc 8 uniform 0.00625")
> estimated80 <- get.score("BC2","markers80")
> true80 <- evaluate.genome("BC2",8)$ratio
```

The donor and recurrent parents are homozygous for alleles 1 and 8, respectively. Six successive backcrosses provide the no-selection baseline, after which the true recurrent-parent genome proportion and its marker-based estimate are obtained for the denser map.

```
> load.linkage.map(file.path(st.output.dir,"mab-1a.map"))
> define.effects("markers40","0, mc 8 uniform 0.0125")
> estimated40 <- get.score("BC2","markers40")
> true40 <- evaluate.genome("BC2",8)$ratio
> out <- list(
+     recurrent.parent.genome=round(rpg,4),
+     marker.estimate.correlation=c(
+         markers80=cor(estimated80,true80),
+         markers40=cor(estimated40,true40)
+     )
+ )
> print(out)
```

The reduced map is then loaded and the same marker estimate is formed with appropriately rescaled marker effects. The reported correlations show how marker density affects the approximation of the true RPG.

```
$recurrent.parent.genome
      Obs   Mean   SDev    Min    Q10    Med    Max
BC1 10000 0.7489 0.0525 0.5693 0.6806 0.7490 0.9352
BC2 10000 0.8737 0.0423 0.7051 0.8173 0.8755 0.9993
BC3 10000 0.9373 0.0301 0.8023 0.8966 0.9408 1.0000
BC4 10000 0.9685 0.0204 0.8641 0.9411 0.9713 1.0000
BC5 10000 0.9841 0.0136 0.8991 0.9659 0.9871 1.0000
BC6 10000 0.9921 0.0091 0.9355 0.9797 0.9949 1.0000

$marker.estimate.correlation
markers80 markers40
  0.90560   0.86971
```

The point of the marker-density comparison is not to define a new selection strategy, but to show how densely distributed background markers approximate the true RPG recovered from the simulated genome.

1.2 Foreground selection for the target allele

Foreground selection retains plants carrying the target allele. At the low level, each backcross is generated explicitly, all best plants for the target effect are retained, one plant is sampled, and the selected plant is used as the non-recurrent parent of the next backcross.

```
> library(SelectionTools)
> options(width=72, digits=5)
> st.set.num.threads(1)
> st.output.dir <- file.path(tempdir(), "SelectionTools-masbc")
> st.data.dir <- st.output.dir
> dir.create(st.output.dir, recursive=TRUE, showWarnings=FALSE)
> NREP <- 10000
> rng.init(seed=1002)
> reset.all()
```

This independent run again fixes the simulation environment and random-number seed before any populations are created.

```
> data("lib00.map1")
> write.table(lib00.map1, file=file.path(st.output.dir,"mab-1.map"),
+             quote=FALSE, row.names=FALSE, col.names=FALSE)
> set.genome.par(
+     no.chrom=10,
+     no.hom=2,
+     chrom.len=c(2.68,1.81,2.08,1.40,1.53,
+                 1.56,0.93,1.44,1.55,1.14)
+ )
> load.linkage.map(file.path(st.output.dir,"mab-1.map"))
> define.effects("target","0, target 1 uniform 1")
```

The historical map is materialized and loaded, and a target effect is defined so that allele 1 at the target locus identifies plants carrying the donor target allele.

```
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(8))
> cross("F1","P1","P2",1)
> pop <- c("F1","BC1","BC2","BC3","BC4","BC5","BC6")
> se1 <- paste(pop,"s1",sep="")
```

```
> sel <- paste(pop,"se",sep="")
> store <- paste(pop,"st",sep="")
> sel[1] <- "F1"
```

The two homozygous parents generate the F1. Population-name vectors are prepared for the six backcross generations and for the intermediate and stored selected populations.

```
> for (j in 1:NREP) {
+     for (i in 1:6) {
+         cross(pop[i+1],sel[i],"P2",50)
+         select.all.best(se1[i+1],pop[i+1],"target")
+         sample.population(sel[i+1],se1[i+1],1)
+         append.population(store[i+1],sel[i+1])
+     }
+ }
```

For each replicate and generation, a backcross population of 50 plants is produced. All plants with the best target score are retained, one is sampled, and that plant becomes the non-recurrent parent of the next backcross.

```
> out <- round(
+     genome.contribution(
+         "BC1st BC2st BC3st BC4st BC5st BC6st",8
+     ),
+     4
+ )
> print(out)
```

The stored selected plants are finally summarized by their recurrent-parent genome contribution. This is the low-level implementation of pure foreground selection.

```
        Obs   Mean   SDev    Min    Q10    Med    Max
BC1st 10000 0.7397 0.0515 0.5597 0.6731 0.7402 0.9304
BC2st 10000 0.8608 0.0423 0.6852 0.8054 0.8629 0.9796
BC3st 10000 0.9235 0.0303 0.7709 0.8828 0.9260 0.9952
BC4st 10000 0.9556 0.0215 0.8436 0.9261 0.9586 0.9994
BC5st 10000 0.9725 0.0153 0.8860 0.9518 0.9751 0.9995
BC6st 10000 0.9817 0.0111 0.9032 0.9668 0.9839 0.9999
```

1.3 Two-stage selection

Two-stage selection first retains target carriers and then selects among them for the largest recurrent-parent marker score. The low-level interface also allows the number of marker data points to be accumulated with `evaluate.mdp()` and `get.mdp()`.

```
> library(SelectionTools)
> options(width=72, digits=5)
> st.set.num.threads(1)
> st.output.dir <- file.path(tempdir(), "SelectionTools-masbc")
> st.data.dir <- st.output.dir
> dir.create(st.output.dir, recursive=TRUE, showWarnings=FALSE)
> NREP <- 10000
> rng.init(seed=1003)
> reset.all()
```

The two-stage example is again independent and runnable. Resetting the simulation state is especially important because marker-data-point counters are used below.

```
> data("lib00.map1")
> write.table(lib00.map1, file=file.path(st.output.dir,"mab-1.map"),
+             quote=FALSE, row.names=FALSE, col.names=FALSE)
> set.genome.par(
+     no.chrom=10,
+     no.hom=2,
+     chrom.len=c(2.68,1.81,2.08,1.40,1.53,
+                 1.56,0.93,1.44,1.55,1.14)
+ )
> load.linkage.map(file.path(st.output.dir,"mab-1.map"))
> define.effects("target","0, target 1 uniform 1")
> define.effects("markers","0, mc 8 uniform 0.00625")
```

The target effect identifies target carriers, whereas the genome-wide marker effect rewards recurrent-parent allele 8 across the map. This separates foreground and background selection explicitly.

```
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(8))
> cross("F1","P1","P2",1)
> pop <- c("F1","BC1","BC2","BC3","BC4","BC5","BC6",
+          "BC7","BC8","BC9","BC10")
> se1 <- paste(pop,"s1",sep="")
> se2 <- paste(pop,"s2",sep="")
> sel <- paste(pop,"se",sep="")
> store <- paste(pop,"st",sep="")
> sel[1] <- "F1"
> mdp <- matrix(NA_real_,nrow=10,ncol=NREP)
```

Parents and F1 are created and names for ten backcross generations are prepared. The `mdp` matrix stores cumulative marker data points separately for every generation and replicate.

```
> for (j in 1:NREP) {
+     reset.mdp()
+     for (i in 1:10) {
+         cross(pop[i+1],sel[i],"P2",100)
+         select.all.best(se1[i+1],pop[i+1],"target")
+         evaluate.mdp(se1[i+1],sel[i],"P2","markers")
+         mdp[i,j] <- get.mdp()
+         select.all.best(se2[i+1],se1[i+1],"markers")
+         sample.population(sel[i+1],se2[i+1],1)
+         append.population(store[i+1],sel[i+1])
+     }
+ }
```

Within each backcross generation, target carriers are selected first. Marker requirements are then counted for these candidates, background selection chooses the best recurrent-parent marker score, and one plant is sampled for the next generation.

```
> out <- list(
+     recurrent.parent.genome=round(
+         genome.contribution(
+             paste0("BC",1:10,"st",collapse=" "),8
+         ),
+         4
```

```
+     ),
+     mean.marker.data.points=data.frame(
+         generation=paste0("BC",1:10),
+         MDP=round(rowMeans(mdp),1)
+     )
+ )
> print(out)
```

The result combines the realized RPG of the selected plants with the average cumulative marker-data requirement. Thus biological response and genotyping effort can be compared in the same example.

```
$recurrent.parent.genome
         Obs   Mean   SDev    Min    Q10    Med    Max
BC1st  10000 0.8438 0.0281 0.7363 0.8081 0.8432 0.9402
BC2st  10000 0.9573 0.0161 0.8837 0.9358 0.9585 0.9968
BC3st  10000 0.9860 0.0084 0.9366 0.9743 0.9874 0.9998
BC4st  10000 0.9928 0.0054 0.9567 0.9857 0.9943 0.9999
BC5st  10000 0.9948 0.0038 0.9638 0.9900 0.9958 0.9999
BC6st  10000 0.9959 0.0029 0.9728 0.9928 0.9965 0.9999
BC7st  10000 0.9965 0.0023 0.9765 0.9940 0.9969 0.9999
BC8st  10000 0.9968 0.0019 0.9812 0.9945 0.9972 0.9999
BC9st  10000 0.9971 0.0017 0.9812 0.9949 0.9973 0.9999
BC10st 10000 0.9973 0.0016 0.9860 0.9951 0.9975 1.0000

$mean.marker.data.points
   generation    MDP
1         BC1 4003.7
2         BC2 5179.9
3         BC3 5437.1
4         BC4 5475.2
5         BC5 5477.3
6         BC6 5477.4
7         BC7 5477.4
8         BC8 5477.4
9         BC9 5477.4
10       BC10 5477.4
```

1.4 Three-stage selection

The three-stage strategy first selects for the target allele, then for recurrent-parent alleles at flanking markers, and finally for the genome-wide recurrent-parent marker score. The flanking step enriches for recombinants close to the target and thereby reduces linkage drag before the genome-wide background-selection step.

```
> library(SelectionTools)
> options(width=72, digits=5)
> st.set.num.threads(1)
> st.output.dir <- file.path(tempdir(), "SelectionTools-masbc")
> st.data.dir <- st.output.dir
> dir.create(st.output.dir, recursive=TRUE, showWarnings=FALSE)
> NREP <- 10000
> rng.init(seed=1004)
> reset.all()
```

The three-stage simulation uses its own seed and clean simulation state, exactly as the preceding low-level examples.

```
> data("lib00.map2")
> write.table(lib00.map2, file=file.path(st.output.dir,"mab-2.map"),
+             quote=FALSE, row.names=FALSE, col.names=FALSE)
> set.genome.par(
+     no.chrom=10,
+     no.hom=2,
+     chrom.len=c(2.68,1.81,2.08,1.40,1.53,
+                 1.56,0.93,1.44,1.55,1.14)
+ )
> load.linkage.map(file.path(st.output.dir,"mab-2.map"))
> define.effects("target","0, target 1 uniform 1")
> define.effects("flanking","0, flanking 8 uniform 1")
> define.effects("markers","0, mc 8 uniform 1")
```

The second historical map contains the target and its flanking loci. Separate effects are defined for the target, the flanking markers, and the genome-wide recurrent-parent marker score.

```
> init.population("P1",homozygote(1))
> init.population("P2",homozygote(8))
> cross("F1","P1","P2",1)
> pop <- c("F1","BC1","BC2","BC3","BC4","BC5","BC6")
> s1 <- paste(pop,"s1",sep="")
> s2 <- paste(pop,"s2",sep="")
> sel <- paste(pop,"se",sep="")
> store <- paste(pop,"st",sep="")
> sel[1] <- "F1"
> mdp <- matrix(NA_real_,nrow=6,ncol=NREP)
```

Parents and F1 are initialized and population names are prepared for six backcross generations. A marker-data-point matrix again records genotyping effort over replicates.

```
> for (j in 1:NREP) {
+     reset.mdp()
+     for (i in 1:6) {
+         cross(pop[i+1],sel[i],"P2",100)
+         select.all.best(s1[i+1],pop[i+1],"target")
+         evaluate.mdp(s1[i+1],sel[i],"P2","flanking")
+         select.all.best(s2[i+1],s1[i+1],"flanking")
+         evaluate.mdp(s2[i+1],sel[i],"P2","markers")
+         select.n.best(sel[i+1],s2[i+1],"markers",1)
+         append.population(store[i+1],sel[i+1])
+         mdp[i,j] <- get.mdp()
+     }
+ }
```

Each generation performs the three selection stages in order: target carriers, recurrent-parent alleles at flanking markers, and finally genome-wide background selection. The selected plant is stored and used in the next backcross.

```
> out <- list(
+     recurrent.parent.genome=round(
+         genome.contribution(
+             "BC1st BC2st BC3st BC4st BC5st BC6st",8
+         ),
+         4
+     ),
+     mean.marker.data.points=data.frame(
+         generation=paste0("BC",1:6),
```

```
+         MDP=round(rowMeans(mdp),1)
+     )
+ )
> print(out)
```

The final summary reports RPG together with marker-data requirements. Relative to two-stage selection, the additional flanking step is intended to reduce donor chromosome surrounding the target.

```
$recurrent.parent.genome
        Obs   Mean   SDev    Min    Q10    Med    Max
BC1st 10000 0.7993 0.0487 0.6004 0.7331 0.8049 0.9512
BC2st 10000 0.9350 0.0265 0.8050 0.9005 0.9370 0.9979
BC3st 10000 0.9851 0.0101 0.9280 0.9712 0.9872 0.9999
BC4st 10000 0.9922 0.0059 0.9374 0.9842 0.9939 1.0000
BC5st 10000 0.9945 0.0042 0.9594 0.9891 0.9956 1.0000
BC6st 10000 0.9958 0.0032 0.9710 0.9921 0.9965 1.0000

$mean.marker.data.points
  generation    MDP
1        BC1  593.1
2        BC2 1373.6
3        BC3 1797.9
4        BC4 1826.0
5        BC5 1826.1
6        BC6 1826.1
```

These examples illustrate why the low-level interface remains useful even though the high-level interface is much shorter: every intermediate population and every marker-analysis step can be inspected or modified.

2 High-level MABC workflow

The high-level interface comprises four principal functions.

`mab.simulate()` simulates one MABC program and saves the complete result object in an explicitly supplied result file.

`mab.compare()` runs up to nine related scenarios. Scenario A is the base definition; scenarios B through I inherit every setting from A unless the corresponding scenario-specific argument is supplied.

`mab.load.data()` loads one result object from an explicitly supplied result file.

`mab.tabulate()` tabulates one or more saved scenarios and returns the requested summaries.

The high-level interface is convenient for comparing alternative allocations and breeding strategies. The low-level interface is more flexible when the sequence of operations itself is the object of study.

2.1 Input data and terminology

2.1.1 Input directories and result files

The high-level MABC routines obtain linkage maps and parental marker data from `st.input.dir`. Result files are written relative to `st.data.dir`; recoded parental marker

files are written relative to `st.output.dir`. Absolute file names can also be supplied. The high-level routines require explicit `result.file` or `result.files` arguments; result-file names are not generated automatically.

Each simulation example calls `rng.init(seed=...)` after the shared setup to initialize both R and the internal random-number generators. Use the same seed, inputs, call sequence and actual thread count in the same software environment to repeat a result. Standard startup remains clock-based when no seed is requested; `rng.init()` restores that behavior.

The following shared setup prepares the data and directories for all high-level examples. Run it once before selecting an example:

```
library(SelectionTools)
options(width=72, digits=5)
st.set.num.threads(1)

data("masbc.input")
st.input.dir <- file.path(tempdir(), "SelectionTools-masbc-input")
dir.create(st.input.dir, recursive=TRUE, showWarnings=FALSE)
for (file in names(masbc.input))
    writeLines(masbc.input[[file]], file.path(st.input.dir,file))
st.output.dir <- file.path(tempdir(), "SelectionTools-masbc")
st.data.dir <- st.output.dir
dir.create(st.output.dir, recursive=TRUE, showWarnings=FALSE)

NREP <- 10000
```

The package data set `masbc.input` contains the 16 text files used by the high-level examples as a named list of character vectors. It contains `ex-mab1.map` through `ex-mab4.map`, `ex-mabp1.mda`, `ex-mabp2.mda`, `ex-mabp1_m.mda`, `ex-mabp2_m.mda`, and `ex-mabi1.mda` through `ex-mabi8.mda`. The `mab.*` functions use file names, so the shared setup loads `masbc.input` with `data()` and writes its contents to a temporary input directory. No permanent files are created in the user's working directory.

2.1.2 Linkage-map files

A linkage-map file contains three fields: chromosome number, position in cM, and locus name. A short excerpt from `ex-mab1.map` is:

```
1       0       umc94
1       70      umc137
5       23      umc90
5       30      target
5       46      umc27
5       57      umc43
```

The first MABC examples assume homozygous donor and recurrent parents and polymorphism between them at every marker on the map. In this mode, `donor.parent="H"` and `recurrent.parent="H"` are used.

2.1.3 Parental marker files

Observed parental marker genotypes can be supplied instead. The first line contains the genotype name. For a homozygous parent, each following line contains marker name and one allele, for example:

```
01E0103DH03
s1e2855xxx ade
```

```
s1e2083xxx ade
s1e4552xxx gua
s1e4867xxx cyt
```

For a heterozygous parent, two alleles are given:

```
bc1i1
umc94   1  8
bnl805  1  8
umc76   8  8
umc137  8  8
```

If only one allele is present in the file, the genotype is treated as homozygous. When parental files are used, donor and recurrent parent must both be supplied as files. Only markers that occur in the linkage map and in both parental files and that are not coded as missing are retained. The missing-allele symbol is controlled by `missing.allele`; the default is `"9"`.

The following input-data example reads the files directly and verifies the linkage-map and parental-marker file structure.

```
> # Input files have been prepared by the shared setup.
```

The shared setup loads the package, fixes output formatting, selects a single thread, prepares the input directory, and directs results to a temporary directory. `NREP` is fixed at 10000 throughout the vignette.

```
> map <- read.table(
+     file.path(st.input.dir,"ex-mab1.map"),
+     stringsAsFactors=FALSE
+ )
> names(map) <- c("chromosome","position.cM","locus")
```

The map file is ordinary white-space separated text. Reading it directly shows the three required fields: chromosome, position in cM, and locus name.

```
> parent <- read.table(
+     file.path(st.input.dir,"ex-mabp1_m.mda"),
+     stringsAsFactors=FALSE,
+     skip=1
+ )
> parent.name <- scan(
+     file.path(st.input.dir,"ex-mabp1_m.mda"),
+     what="character",
+     n=1,
+     quiet=TRUE
+ )
```

The parental marker file begins with the genotype name and is followed by marker–allele records. The first call reads the marker table, while `scan()` retrieves the name stored on the first line.

```
> out <- list(
+     map=head(map,6),
+     parent.name=parent.name,
+     parent.markers=head(parent,6)
+ )
> print(out)
```

Only a short prefix is printed here. This makes the file structure visible without reproducing the complete map and parental genotype file in the vignette.

```
$map
  chromosome position.cM  locus
1          1           0  umc94
2          1           2 bnl805
3          1          67  umc76
4          1          70 umc137
5          1         141  umc58
6          1         160  umc23

$parent.name
[1] "01E0103DH03"

$parent.markers
          V1  V2
1 s1e2855xxx ade
2 s1e2083xxx ade
3 s1e4552xxx gua
4 s1e4867xxx cyt
5 s1e3305xxx cyt
6 s1e2837xxx cyt
```

2.1.4 Generation types

A high-level breeding program is described by parallel vectors. The implemented generation types are:

`f1`	F1 population
`bc`	backcross population
`s`	selfing population
`dh`	doubled haploid population

The vectors `gen.type`, `population.size`, `sel.strategy`, and `no.selected` must describe the same number of generations. If `no.selected` is omitted, one individual is selected in each generation. `no.preselected`, when supplied, must likewise have one entry per generation.

2.1.5 Selection strategies

The high-level selection code is a sequence of letters describing the selection stages.

`n`	random selection
`t`	donor alleles at target loci
`tb`	target selection followed by genome-wide background selection
`tf`	target selection followed by recurrent-parent alleles at flanking loci
`tfb`	target, flanking-marker, and genome-wide background selection
`tfs`	target and flanking-marker selection followed by selection for a low number of donor chromosome segments
`tr`	target selection followed by recurrent-parent alleles at recipient loci
`trb`	target, recipient-locus, and genome-wide background selection
`trfb`	target, recipient-locus, flanking-marker, and genome-wide background selection
`tfrb`	target, flanking-marker, recipient-locus, and genome-wide background selection

The two-stage strategy `tb` and the three-stage strategy `tfb` correspond to the strategies described by Frisch et al. (1999). `flanking.loci` normally flank the target locus and are used to reduce the donor chromosome segment attached to it, but they need not literally be the nearest flanking markers. `recipient.loci` define recurrent-parent alleles or chromosome regions that must be retained in the final backcrossing product.

2.1.6 Counting marker analyses

The high-level simulator also records marker-data requirements. The papers count individual marker data points (MDP). An HT assay here counts one genotyped individual, not each marker scored on that individual; therefore HT totals must not be compared numerically with published MDP totals. The low-level examples explicitly accumulate MDP.

`t` Foreground selection for the target genes is not counted, because this step requires the same number of target assays for all strategies being compared.

`f` Flanking-marker preselection is counted as single-marker (SM) assays. In advanced generations, markers already fixed for the recurrent allele in the selected non-recurrent parent need not be assayed again.

`b` Genome-wide background selection is counted as one high-throughput (HT) assay per assayed individual, irrespective of marker genotype.

`s` Selection for a low number of donor segments is also counted as one HT assay per assayed individual.

`r` Recipient-locus preselection is treated as an HT assay. In `tr`, `trb`, and `trfb`, one HT assay is counted for every plant carrying the target genes. In `tfrb`, flanking selection precedes the recipient-locus step and the flanking assays are counted as SM assays.

The optional `success.factor` is the probability that a selected plant survives and is fertile. Values below one cause the number of successful selected individuals to be drawn from a binomial distribution; at least one individual is retained.

2.2 `mab.simulate()`

`mab.simulate()` defines one complete program. Its principal inputs are the simulation identifiers, number of repetitions, donor and recurrent parents, linkage map, target/flanking/recipient loci, generation types, population sizes, selection strategies, numbers selected, optional minimum numbers preselected, optional separately evaluated chromosome regions, missing-allele code, success factor, and the explicit result file.

A successful call invisibly returns the same named list that is saved to the result file. Depending on the requested analyses, the result contains target-allele summaries, genome-wide and regional RPG summaries, genome-wide and regional donor-segment counts, linkage drag for each target, and SM and HT marker-analysis counts.

2.3 `mab.load.data()` and `mab.tabulate()`

`mab.load.data()` reads one saved result. The simulation name and run remain part of the workflow identifiers, but the file itself is selected explicitly by `result.file`.

`mab.tabulate()` accepts alternating simulation names and run identifiers in `simulation.names.runs` and one explicitly supplied result file for each scenario. It can return a basic parameter summary, an extended parameter summary, mean and Q10 RPG values, target-allele summaries, regional RPG, donor-segment counts, linkage drag, and SM/HT marker-analysis counts. RPG can be reported as percent (the default) or as a proportion, and the number of displayed decimal places is controlled by `digits`.

The following complete example demonstrates the three functions.

```
> rng.init(seed=2000)
```

The first simulation example uses the shared directory setup and fixes a seed. High-level routines read maps from `st.input.dir` and save result objects in `st.data.dir`.

```
> mab.simulate(
+     simulation.name="core",
+     simulation.run="x",
+     repetitions=NREP,
+     linkage.map="ex-mab1.map",
+     target.loci="target",
+     flanking.loci=c("umc90","umc27"),
+     gen.type=c("f1","bc","bc","bc"),
+     population.size=c(1,100,100,100),
+     sel.strategy=c("n","tfb","tfb","tfb"),
+     result.file="core.RData"
+ )
```

A complete breeding program is described in one `mab.simulate()` call: one F1 generation followed by three backcrosses, target selection, population size 100, and one selected plant per generation.

```
> loaded <- mab.load.data(
+     simulation.name="core",
+     simulation.run="x",
+     result.file="core.RData"
+ )
```

`mab.load.data()` retrieves the saved simulation object explicitly from its result file. Loading and simulation are therefore separate operations.

```
> tab <- mab.tabulate(
+     simulation.names.runs=c("core","x"),
+     ext.summary=TRUE,
+     TargetAlleles=TRUE,
+     RPG.gw.mean=TRUE,
+     RPG.gw.Q10=TRUE,
+     DonorSegments.gw=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files="core.RData"
+ )
```

`mab.tabulate()` requests selected summaries from the same saved result. Here the target-allele state, mean RPG, linkage drag, and marker-analysis counts are requested together.

```
> out <- list(
+     result.components=names(loaded),
+     selected.summary=tab
+ )
> print(out)
```

The printed list demonstrates the two levels of access: the loaded object is available for detailed inspection, whereas the tabulation provides the usual compact MABC summaries.

```
$result.components
 [1] "simulation.name"  "simulation.run"   "repetitions"
 [4] "recurrent.parent" "donor.parent"     "linkage.map"
 [7] "target.loci"      "flanking.loci"    "gen.type"
[10] "population.size"  "sel.strategy"     "no.selected"
[13] "success.factor"   "TargetAlleles"    "RPG.gw"
[16] "DonorSegments.gw" "LinkageDrag1"     "SM"
[19] "HT"

$selected.summary
$selected.summary$ext.summary
         core-x
Reps     "10000"
RP       "H"
DP       "H"
Map      "ex-mab1.map"
Target   "target"
Flm      "umc90/umc27"
Pop type "f1/bc/bc/bc"
Pop size "1/100/100/100"
Sel str  "n/tfb/tfb/tfb"
n Sel    "1/1/1/1"
n PreSel ""
Succ f   "1"

$selected.summary$TargetAlleles.mean
   core-x
P1      2
G0      1
G1      1
G2      1
G3      1

$selected.summary$RPG.gw.mean
   core-x
P1    0.0
G0   50.0
G1   80.0
G2   93.5
G3   98.5

$selected.summary$RPG.gw.Q10
   core-x
P1    0.0
G0   50.0
G1   73.2
G2   90.0
G3   97.1

$selected.summary$DonorSegments.gw.mean
   core-x
P1   20.0
G0   10.0
G1   12.7
G2    8.4
G3    3.8

$selected.summary$LinkageDrag1.mean
   core-x
P1  153.1
G0  153.1
G1   32.5
G2   11.5
```

```
G3   10.5

$selected.summary$SM.mean
      core-x
Total  132.5
G0       0.0
G1     100.1
G2      31.6
G3       0.9

$selected.summary$HT.mean
      core-x
Total   77.5
G0       0.0
G1       6.3
G2      21.9
G3      49.3
```

2.4 `mab.compare()`

`mab.compare()` can run up to nine scenarios, labeled A through I. Scenario A is the complete base definition. A later scenario is activated by giving its `simulation.name`; every other `NULL` setting then inherits the corresponding A setting. Thus only the parameters being changed need to be supplied. `result.files` contains one unique file name for every active scenario in A–I order.

This inheritance mechanism is used repeatedly in the advanced examples below to compare population sizes, selection strategies, parental genotypes, and survival assumptions.

2.5 Interpreting MABC results

The main output quantities have direct breeding interpretations.

Target alleles. `TargetAlleles` summarizes the frequency of donor target alleles among the selected plants over generations.

Genome-wide RPG. `RPG.gw` summarizes recurrent-parent genome content over the complete genome. `mab.tabulate()` can report its mean and its 10th percentile (Q10).

Regional RPG. `RPG.reg1`, `RPG.reg2`, ... summarize RPG within chromosome regions defined by `reg.chr`, `reg.begin`, and `reg.end`. An `reg.end` of zero means the chromosome end.

Donor segments. `DonorSegments.gw` and the corresponding regional objects count donor chromosome segments. Segments are counted per homologue, so the two homologues of a diploid chromosome are counted separately.

Linkage drag. `LinkageDrag1`, `LinkageDrag2`, ... give the donor chromosome segment attached to each target locus. Target loci are evaluated separately and linkage drag is averaged over the two homologues.

Marker analyses. `SM` and `HT` summarize the single-marker and high-throughput marker analyses attributed to the selected strategy.

These quantities allow strategies to be compared simultaneously for genetic outcome and marker-analysis effort.

2.6 Planning comparisons

MABC optimization is usually a comparison problem rather than a single simulation problem. Population size, selection stage, number and position of target genes, density of background markers, starting genotype, preselection intensity, and survival of selected plants can all affect RPG recovery, linkage drag, and marker-data requirements.

The high-level interface is designed around this idea. A useful analysis typically defines one base scenario with `mab.compare()`, changes only the factors of interest in scenarios B through I, and then requests the same columns from `mab.tabulate()` for all scenarios. The examples in the next section illustrate these comparisons with explicitly named result files.

3 Advanced high-level examples

The following subsections present high-level MABC comparisons. After the shared setup in the high-level workflow, each example can be run separately: it uses a fixed seed, creates all result files it needs, and prints the summary shown in the blue box. The number of repetitions is `NREP <- 10000`.

3.1 Example 1: six backcross generations with target selection

This example corresponds to the structure of Table 2 of Frisch et al. (1999). An F1 is followed by six backcross generations and two selfing generations. Strategy `t` selects for the target allele only; marker data points for target assays are not counted.

```
> rng.init(seed=2101)
```

Example 1 initializes both R and the internal random-number generators after the shared setup.

```
> mab.simulate(
+     simulation.name="fbm99-01",
+     simulation.run="x",
+     repetitions=NREP,
+     linkage.map="ex-mab1.map",
+     target.loci="target",
+     flanking.loci=c("umc90","umc27"),
+     gen.type=c("f1","bc","bc","bc","bc","bc","bc","s","s"),
+     population.size=c(1,100,100,100,100,100,100,100,100),
+     sel.strategy=c("n","t","t","t","t","t","t","t","t"),
+     result.file="fbm99-01.RData"
+ )
```

The program follows an F1 with six backcross generations and two selfing generations. Selection strategy `t` retains the target allele but performs no marker-assisted background selection.

```
> out <- mab.tabulate(
+     simulation.names.runs=c("fbm99-01","x"),
+     TargetAlleles=TRUE,
+     RPG.gw.mean=TRUE,
+     LinkageDrag=TRUE,
+     result.files="fbm99-01.RData"
+ )
> print(out)
```

The tabulation reports target-allele recovery, genome-wide RPG, linkage drag, and marker analyses for the saved scenario. These quantities form the baseline for the more intensive strategies below.

```
$par.summary
          fbm99-01-x
Reps      "10000"
DP        "H"
Pop type  "f1/bc/bc/bc/bc/bc/bc/s/s"
Pop size  "1/100/100/100/100/100/100/100/100"
Sel str   "n/t/t/t/t/t/t/t/t"
n Sel     "1/1/1/1/1/1/1/1/1"

$TargetAlleles.mean
   fbm99-01-x
P1          2
G0          1
G1          1
G2          1
G3          1
G4          1
G5          1
G6          1
G7          2
G8          2

$RPG.gw.mean
   fbm99-01-x
P1        0.0
G0       50.0
G1       73.9
G2       86.0
G3       92.3
G4       95.5
G5       97.2
G6       98.2
G7       97.4
G8       97.4

$LinkageDrag1.mean
   fbm99-01-x
P1      153.1
G0      153.1
G1       96.3
G2       68.4
G3       52.4
G4       42.4
G5       35.6
G6       30.7
G7       26.9
G8       26.6
```

3.2 Example 2: two-stage selection

The second historical example corresponds to the RPG comparison in Table 3 of Frisch et al. (1999). Strategy `tb` first selects plants carrying the target allele and then performs genome-wide background selection. The second stage is treated as an HT assay.

The section regenerates both the target-only and two-stage scenarios so that it can be run independently.

```
> rng.init(seed=2102)
```

Example 2 uses a separate seed and result files so that it can be run without Example 1.

```
> mab.simulate(
+     simulation.name="fbm99-01",
+     simulation.run="x",
+     repetitions=NREP,
+     linkage.map="ex-mab1.map",
+     target.loci="target",
+     flanking.loci=c("umc90","umc27"),
+     gen.type=c("f1","bc","bc","bc","bc","bc","bc","s","s"),
+     population.size=c(1,100,100,100,100,100,100,100,100),
+     sel.strategy=c("n","t","t","t","t","t","t","t","t"),
+     result.file="fbm99-01.RData"
+ )
```

The first call reproduces target-only selection over five backcross generations. Keeping this scenario separate provides a direct comparator for two-stage selection.

```
> mab.simulate(
+     simulation.name="fbm99-02",
+     simulation.run="x",
+     repetitions=NREP,
+     linkage.map="ex-mab1.map",
+     target.loci="target",
+     flanking.loci=c("umc90","umc27"),
+     gen.type=c("f1","bc","bc","bc","bc","bc"),
+     population.size=c(1,100,100,100,100,100),
+     sel.strategy=c("n","tb","tb","tb","tb","tb"),
+     result.file="fbm99-02.RData"
+ )
```

The second program changes only the backcross selection strategy to `tb`: target carriers are identified first and genome-wide background selection is then applied among them.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "fbm99-01","x",
+         "fbm99-02","x"
+     ),
+     TargetAlleles=TRUE,
+     RPG.gw.mean=TRUE,
+     RPG.gw.Q10=TRUE,
+     LinkageDrag=TRUE,
+     result.files=c("fbm99-01.RData","fbm99-02.RData")
+ )
> print(out)
```

Both result files are tabulated jointly. The RPG and linkage-drag summaries show the benefit obtained from background selection, while marker-analysis counts quantify the extra genotyping effort.

```
$par.summary
         fbm99-01-x                              fbm99-02-x
Reps     "10000"                                 "10000"
DP       "H"                                     "H"
Pop type "f1/bc/bc/bc/bc/bc/s/s"                 "f1/bc/bc/bc/bc/bc"
Pop size "1/100/100/100/100/100/100/100/100"     "1/100/100/100/100/100"
Sel str  "n/t/t/t/t/t/t/t/t"                     "n/tb/tb/tb/tb/tb"
n Sel    "1/1/1/1/1/1/1/1/1"                     "1/1/1/1/1/1"
```

```
$TargetAlleles.mean
   fbm99-01-x fbm99-02-x
P1          2          2
G0          1          1
G1          1          1
G2          1          1
G3          1          1
G4          1          1
G5          1          1
G6          1         NA
G7          2         NA
G8          2         NA

$RPG.gw.mean
   fbm99-01-x fbm99-02-x
P1        0.0        0.0
G0       50.0       50.0
G1       74.0       84.4
G2       86.1       95.7
G3       92.4       98.6
G4       95.6       99.3
G5       97.2       99.5
G6       98.2         NA
G7       97.4         NA
G8       97.4         NA

$RPG.gw.Q10
   fbm99-01-x fbm99-02-x
P1        0.0        0.0
G0       50.0       50.0
G1       67.4       80.8
G2       80.6       93.6
G3       88.3       97.4
G4       92.7       98.5
G5       95.1       99.0
G6       96.7         NA
G7       95.3         NA
G8       95.2         NA

$LinkageDrag1.mean
   fbm99-01-x fbm99-02-x
P1      153.1      153.1
G0      153.1      153.1
G1       96.2       72.9
G2       67.9       38.2
G3       52.1       18.6
G4       42.2       11.1
G5       35.7       10.2
G6       30.6         NA
G7       26.9         NA
G8       26.6         NA
```

3.3 Example 3: increasing population sizes and three-stage selection

This example follows the principle illustrated in Table 4 of Frisch et al. (1999): population sizes increase from 33 to 100 to 166 across BC1–BC3 and strategy `tfb` combines target, flanking-marker and genome-wide background selection.

```
> rng.init(seed=2103)
```

Example 3 uses its own seed and the shared temporary result directory.

```
> mab.simulate(
+     simulation.name="fbm99-03",
+     simulation.run="x",
+     repetitions=NREP,
+     linkage.map="ex-mab1.map",
+     target.loci="target",
+     flanking.loci=c("umc90","umc27"),
+     gen.type=c("f1","bc","bc","bc"),
+     population.size=c(1,33,100,166),
+     sel.strategy=c("n","tfb","tfb","tfb"),
+     result.file="fbm99-03.RData"
+ )
```

Population sizes increase from 33 to 100 and 166 across the three backcross generations. Strategy `tfb` combines target, flanking-marker, and genome-wide background selection.

```
> out <- mab.tabulate(
+     simulation.names.runs=c("fbm99-03","x"),
+     RPG.gw.Q10=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files="fbm99-03.RData"
+ )
> print(out)
```

The requested summaries emphasize the lower tail of RPG, linkage drag, and marker analyses. These are the quantities used to evaluate whether increasing population size improves the chance of obtaining favorable recombinants.

```
$par.summary
         fbm99-03-x
Reps     "10000"
DP       "H"
Pop type "f1/bc/bc/bc"
Pop size "1/33/100/166"
Sel str  "n/tfb/tfb/tfb"
n Sel    "1/1/1/1"

$RPG.gw.Q10
   fbm99-03-x
P1        0.0
G0       50.0
G1       72.3
G2       88.1
G3       96.9

$LinkageDrag1.mean
   fbm99-03-x
P1      153.1
G0      153.1
G1       44.1
G2       12.1
G3       10.6

$SM.mean
      fbm99-03-x
Total       81.4
G0           0.0
G1          33.1
```

```
G2          44.8
G3           3.5

$HT.mean
      fbm99-03-x
Total       94.7
G0           0.0
G1           3.3
G2          11.7
G3          79.7
```

3.4 Example 4: comparing population sizes

`mab.compare()` is used to define six otherwise identical `tb` programs with population sizes 60, 80, 100, 125, 150 and 200 in every backcross generation, covering the larger population sizes in Table 3 of Frisch et al. (1999). Only the scenario-specific population sizes and names need to be supplied for scenarios B through F.

```
> rng.init(seed=2104)
```

The comparison uses the shared setup and `NREP`, with its own fixed seed.

```
> mab.compare(
+     a.simulation.name="s01-060",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.linkage.map="ex-mab1.map",
+     a.target.loci="target",
+     a.flanking.loci=c("umc90","umc27"),
+     a.gen.type=c("f1","bc","bc","bc"),
+     a.population.size=c(1,60,60,60),
+     a.sel.strategy=c("n","tb","tb","tb"),
+     b.simulation.name="s01-080",
+     b.population.size=c(1,80,80,80),
+     c.simulation.name="s01-100",
+     c.population.size=c(1,100,100,100),
+     d.simulation.name="s01-125",
+     d.population.size=c(1,125,125,125),
+     e.simulation.name="s01-150",
+     e.population.size=c(1,150,150,150),
+     f.simulation.name="s01-200",
+     f.population.size=c(1,200,200,200),
+     result.files=c(
+         "s01-060.RData",
+         "s01-080.RData",
+         "s01-100.RData",
+         "s01-125.RData",
+         "s01-150.RData",
+         "s01-200.RData"
+     )
+ )
```

`mab.compare()` defines the 60-plant program as scenario A. Scenarios B through F inherit every unspecified setting and replace only their names and population sizes, giving directly comparable programs of 80, 100, 125, 150, and 200 plants.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "s01-060","x",
+         "s01-080","x",
+         "s01-100","x",
+         "s01-125","x",
+         "s01-150","x",
+         "s01-200","x"
+     ),
+     RPG.gw.Q10=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files=c(
+         "s01-060.RData",
+         "s01-080.RData",
+         "s01-100.RData",
+         "s01-125.RData",
+         "s01-150.RData",
+         "s01-200.RData"
+     )
+ )
> print(out)
```

The six result files are tabulated together for the 10% RPG quantile, linkage drag, and marker analyses. Joint tabulation is the convenient way to compare alternative program designs.

```
$par.summary
         s01-060-x     s01-080-x     s01-100-x       s01-125-x
Reps     "10000"       "10000"       "10000"         "10000"
DP       "H"           "H"           "H"             "H"
Pop type "f1/bc/bc/bc" "f1/bc/bc/bc" "f1/bc/bc/bc"   "f1/bc/bc/bc"
Pop size "1/60/60/60"  "1/80/80/80"  "1/100/100/100" "1/125/125/125"
Sel str  "n/tb/tb/tb"  "n/tb/tb/tb"  "n/tb/tb/tb"    "n/tb/tb/tb"
n Sel    "1/1/1/1"     "1/1/1/1"     "1/1/1/1"       "1/1/1/1"
         s01-150-x       s01-200-x
Reps     "10000"         "10000"
DP       "H"             "H"
Pop type "f1/bc/bc/bc"   "f1/bc/bc/bc"
Pop size "1/150/150/150" "1/200/200/200"
Sel str  "n/tb/tb/tb"    "n/tb/tb/tb"
n Sel    "1/1/1/1"       "1/1/1/1"

$RPG.gw.Q10
   s01-060-x s01-080-x s01-100-x s01-125-x s01-150-x s01-200-x
P1       0.0       0.0       0.0       0.0       0.0       0.0
G0      50.0      50.0      50.0      50.0      50.0      50.0
G1      79.8      80.3      80.9      81.3      81.6      82.1
G2      92.8      93.2      93.6      93.9      94.2      94.5
G3      97.1      97.3      97.4      97.5      97.6      97.8

$LinkageDrag1.mean
   s01-060-x s01-080-x s01-100-x s01-125-x s01-150-x s01-200-x
P1     153.1     153.1     153.1     153.1     153.1     153.1
G0     153.1     153.1     153.1     153.1     153.1     153.1
G1      75.3      73.9      72.8      71.8      71.3      70.1
G2      41.6      39.7      38.2      36.7      35.5      33.8
G3      22.1      20.0      18.5      17.0      16.2      14.6

$SM.mean
      s01-060-x s01-080-x s01-100-x s01-125-x s01-150-x s01-200-x
Total         0         0         0         0         0         0
G0            0         0         0         0         0         0
```

```
G1            0         0         0         0         0         0
G2            0         0         0         0         0         0
G3            0         0         0         0         0         0

$HT.mean
      s01-060-x s01-080-x s01-100-x s01-125-x s01-150-x s01-200-x
Total      90.1     119.9     150.1     187.4       225     300.1
G0          0.0       0.0       0.0       0.0         0       0.0
G1         30.1      40.0      50.0      62.4        75     100.1
G2         30.0      40.0      50.1      62.5        75     100.0
G3         30.0      39.9      50.0      62.5        75     100.0
```

3.5 Example 5: simultaneous introgression of two genes

This example corresponds to the structure of Table 5 of Frisch and Melchinger (2001). Two target genes are introgressed simultaneously. Four flanking markers define the target regions, and the population increases from 60 to 120 to 180 across BC1–BC3. Arbitrary numbers of target genes and flanking markers can be defined in the same way. The example ends at BC3; Table 5 additionally includes a final selfing generation and reports BC3-S1, so its final RPG and marker-analysis values are not directly comparable to the BC3 values here.

```
> rng.init(seed=2105)
```

The two-gene introgression scenario uses the shared setup and its own seed.

```
> mab.simulate(
+     simulation.name="fm01u-01",
+     simulation.run="x",
+     repetitions=NREP,
+     linkage.map="ex-mab2.map",
+     target.loci=c("target1","target2"),
+     flanking.loci=c("umc90","umc27","umc111","bnl1407"),
+     gen.type=c("f1","bc","bc","bc"),
+     population.size=c(1,60,120,180),
+     sel.strategy=c("n","tfb","tfb","tfb"),
+     result.file="fm01u-01.RData"
+ )
```

Two target loci and four flanking markers are supplied. Three backcross generations use increasing population sizes and `tfb` selection to recover both target alleles while controlling linkage drag and background recovery.

```
> out <- mab.tabulate(
+     simulation.names.runs=c("fm01u-01","x"),
+     TargetAlleles=TRUE,
+     RPG.gw.mean=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files="fm01u-01.RData"
+ )
> print(out)
```

The result table requests target-allele recovery, RPG, linkage drag, and marker analyses. The same high-level interface therefore extends from one target locus to multiple target loci without changing the simulation workflow.

```
$par.summary
          fm01u-01-x
Reps      "10000"
DP        "H"
Pop type "f1/bc/bc/bc"
Pop size "1/60/120/180"
Sel str  "n/tfb/tfb/tfb"
n Sel    "1/1/1/1"

$TargetAlleles.mean
   fm01u-01-x
P1           4
G0           2
G1           2
G2           2
G3           2

$RPG.gw.mean
   fm01u-01-x
P1         0.0
G0        50.0
G1        76.8
G2        90.2
G3        96.5

$LinkageDrag1.mean
   fm01u-01-x
P1      153.1
G0      153.1
G1       61.1
G2       25.8
G3       12.6

$LinkageDrag2.mean
   fm01u-01-x
P1        93.1
G0        93.1
G1        51.8
G2        24.2
G3        12.3

$SM.mean
      fm01u-01-x
Total      176.9
G0           0.0
G1          60.0
G2          71.5
G3          45.4

$HT.mean
      fm01u-01-x
Total       19.8
G0           0.0
G1           2.8
G2           4.3
G3          12.6
```

3.6 Example 6: selection strategy and linkage drag

Three otherwise identical programs compare target-only selection (`t`), target plus background selection (`tb`), and target plus flanking-marker plus background selection (`tfb`). The comparison shows how selection strategy changes RPG, linkage drag, and marker-analysis requirements.

```
> rng.init(seed=2106)
```

Example 6 starts from the same simulation environment for all strategies in the comparison.

```
> mab.compare(
+     a.simulation.name="ld-t",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.linkage.map="ex-mab1.map",
+     a.target.loci="target",
+     a.flanking.loci=c("umc90","umc27"),
+     a.gen.type=c("f1","bc","bc","bc"),
+     a.population.size=c(1,100,100,100),
+     a.sel.strategy=c("n","t","t","t"),
+     a.no.selected=c(1,1,1,1),
+     b.simulation.name="ld-tb",
+     b.sel.strategy=c("n","tb","tb","tb"),
+     c.simulation.name="ld-tfb",
+     c.sel.strategy=c("n","tfb","tfb","tfb"),
+     result.files=c("ld-t.RData","ld-tb.RData","ld-tfb.RData")
+ )
```

The base scenario uses target-only selection. Two inherited scenarios replace that strategy by `tb` and `tfb`, so the effect of adding background and flanking-marker selection can be examined without changing other design parameters.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "ld-t","x",
+         "ld-tb","x",
+         "ld-tfb","x"
+     ),
+     RPG.gw.Q10=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files=c("ld-t.RData","ld-tb.RData","ld-tfb.RData")
+ )
> print(out)
```

The joint table focuses on RPG, linkage drag, and marker analyses. This exposes the central trade-off between faster recurrent-genome recovery, reduced donor segment around the target, and genotyping effort.

```
$par.summary
         ld-t-x          ld-tb-x         ld-tfb-x
Reps     "10000"         "10000"         "10000"
DP       "H"             "H"             "H"
Pop type "f1/bc/bc/bc"   "f1/bc/bc/bc"   "f1/bc/bc/bc"
Pop size "1/100/100/100" "1/100/100/100" "1/100/100/100"
Sel str  "n/t/t/t"       "n/tb/tb/tb"    "n/tfb/tfb/tfb"
n Sel    "1/1/1/1"       "1/1/1/1"       "1/1/1/1"
```

```
$RPG.gw.Q10
   ld-t-x ld-tb-x ld-tfb-x
P1    0.0     0.0      0.0
G0   50.0    50.0     50.0
G1   67.4    80.8     73.2
G2   80.4    93.5     90.0
G3   88.2    97.4     97.1

$LinkageDrag1.mean
   ld-t-x ld-tb-x ld-tfb-x
P1  153.1   153.1    153.1
G0  153.1   153.1    153.1
G1   96.9    73.0     33.1
G2   68.4    38.0     11.5
G3   52.1    18.5     10.5

$SM.mean
      ld-t-x ld-tb-x ld-tfb-x
Total      0       0    133.0
G0         0       0      0.0
G1         0       0     99.9
G2         0       0     32.1
G3         0       0      0.9

$HT.mean
      ld-t-x ld-tb-x ld-tfb-x
Total      0   149.8     77.2
G0         0     0.0      0.0
G1         0    49.9      6.4
G2         0    50.0     21.5
G3         0    49.9     49.4
```

3.7 Example 7: introgression of three genes

Three target genes are distributed over chromosomes 1, 5 and 7, each with two flanking markers. Four scenarios compare `t`, `tb`, `tfb`, and a program that uses a larger BC1 population followed by `tb` in later generations.

```
> rng.init(seed=2107)
```

The three-gene comparison uses one seed and a common set of four result files.

```
> mab.compare(
+     a.simulation.name="fm013g-01",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.linkage.map="ex-mab3.map",
+     a.target.loci=c("target1","target2","target3"),
+     a.flanking.loci=c(
+         "umc90","umc27",
+         "umc111","bnl1407",
+         "umc37","bnl1518"
+     ),
+     a.gen.type=c("f1","bc","bc","bc"),
+     a.population.size=c(1,100,100,100),
+     a.sel.strategy=c("n","t","t","t"),
+     b.simulation.name="fm013g-02",
+     b.sel.strategy=c("n","tb","tb","tb"),
```

```
+     c.simulation.name="fm013g-03",
+     c.sel.strategy=c("n","tfb","tfb","tfb"),
+     d.simulation.name="fm013g-04",
+     d.population.size=c(1,200,100,100),
+     d.sel.strategy=c("n","tfb","tb","tb"),
+     result.files=c(
+         "fm013g-01.RData",
+         "fm013g-02.RData",
+         "fm013g-03.RData",
+         "fm013g-04.RData"
+     )
+ )
```

Scenario A introduces three targets with target-only selection. The inherited scenarios compare `tb`, `tfb`, and a design with a larger BC1 population followed by less intensive selection in later generations.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "fm013g-01","x",
+         "fm013g-02","x",
+         "fm013g-03","x",
+         "fm013g-04","x"
+     ),
+     RPG.gw.Q10=TRUE,
+     TargetAlleles=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files=c(
+         "fm013g-01.RData",
+         "fm013g-02.RData",
+         "fm013g-03.RData",
+         "fm013g-04.RData"
+     )
+ )
> print(out)
```

All four programs are summarized together for target-allele recovery, the 10% RPG quantile, linkage drag, and marker analyses. The example shows how population size and selection intensity can be varied jointly for multi-gene introgression.

```
$par.summary
         fm013g-01-x     fm013g-02-x     fm013g-03-x
Reps     "10000"         "10000"         "10000"
DP       "H"             "H"             "H"
Pop type "f1/bc/bc/bc"   "f1/bc/bc/bc"   "f1/bc/bc/bc"
Pop size "1/100/100/100" "1/100/100/100" "1/100/100/100"
Sel str  "n/t/t/t"       "n/tb/tb/tb"    "n/tfb/tfb/tfb"
n Sel    "1/1/1/1"       "1/1/1/1"       "1/1/1/1"
         fm013g-04-x
Reps     "10000"
DP       "H"
Pop type "f1/bc/bc/bc"
Pop size "1/200/100/100"
Sel str  "n/tfb/tb/tb"
n Sel    "1/1/1/1"

$TargetAlleles.mean
   fm013g-01-x fm013g-02-x fm013g-03-x fm013g-04-x
P1           6           6           6           6
G0           3           3           3           3
```

```
G1           3           3           3           3
G2           3           3           3           3
G3           3           3           3           3

$RPG.gw.Q10
   fm013g-01-x fm013g-02-x fm013g-03-x fm013g-04-x
P1         0.0         0.0         0.0         0.0
G0        50.0        50.0        50.0        50.0
G1        65.4        74.7        69.4        70.2
G2        77.7        87.6        83.5        86.6
G3        85.1        93.2        91.2        93.3

$LinkageDrag1.mean
   fm013g-01-x fm013g-02-x fm013g-03-x fm013g-04-x
P1       153.1       153.1       153.1       153.1
G0       153.1       153.1       153.1       153.1
G1        96.3        79.1        71.7        64.0
G2        68.5        47.8        37.8        39.2
G3        52.3        32.1        22.5        27.0

$LinkageDrag2.mean
   fm013g-01-x fm013g-02-x fm013g-03-x fm013g-04-x
P1        93.1        93.1        93.1        93.1
G0        93.1        93.1        93.1        93.1
G1        74.0        67.4        57.6        54.0
G2        60.1        46.6        35.3        38.6
G3        49.9        30.4        21.2        25.8

$LinkageDrag3.mean
   fm013g-01-x fm013g-02-x fm013g-03-x fm013g-04-x
P1       268.1       268.1       268.1       268.1
G0       268.1       268.1       268.1       268.1
G1       140.1       115.9        96.7        87.4
G2        88.8        62.4        42.9        48.5
G3        63.4        36.4        24.8        30.2

$SM.mean
      fm013g-01-x fm013g-02-x fm013g-03-x fm013g-04-x
Total           0           0       150.5       150.1
G0              0           0         0.0         0.0
G1              0           0        74.9       150.1
G2              0           0        47.6         0.0
G3              0           0        28.0         0.0

$HT.mean
      fm013g-01-x fm013g-02-x fm013g-03-x fm013g-04-x
Total           0        37.5         8.2        27.5
G0              0         0.0         0.0         0.0
G1              0        12.5         2.2         2.5
G2              0        12.5         2.6        12.5
G3              0        12.4         3.3        12.5
```

3.8 Example 8: observed marker data for donor and recurrent parent

Instead of assuming two fully polymorphic homozygous lines, observed SNP marker files define both parents. If one allele is supplied at a marker, the parent is treated as homozygous there. Only markers with known map positions and usable marker data in both parents are retained. Missing alleles are coded by `"9"` in this example.

The map `ex-mab4.map` contains 320 marker rows and the parental files contain the corresponding observed alleles. The target is `s7e2190xxx`, flanked by `s7e5163xxx` and

s7e4815xxx.

```
> rng.init(seed=2108)
```

This example uses observed parental marker files rather than idealized homozygous allele codes, while using the shared setup and its own fixed seed.

```
> mab.compare(
+     a.simulation.name="marker-01",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.donor.parent="ex-mabp1_m.mda",
+     a.recurrent.parent="ex-mabp2_m.mda",
+     a.linkage.map="ex-mab4.map",
+     a.target.loci="s7e2190xxx",
+     a.flanking.loci=c("s7e5163xxx","s7e4815xxx"),
+     a.gen.type=c("f1","bc","bc"),
+     a.population.size=c(1,25,25),
+     a.sel.strategy=c("n","t","t"),
+     a.missing.allele="9",
+     b.simulation.name="marker-02",
+     b.population.size=c(1,50,50),
+     b.sel.strategy=c("n","tb","tb"),
+     result.files=c("marker-01.RData","marker-02.RData")
+ )
```

The donor and recurrent-parent files, observed linkage map, target locus, flanking loci, and missing-allele symbol are supplied to `mab.compare()`. A second scenario doubles the backcross population size and adds background selection.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "marker-01","x",
+         "marker-02","x"
+     ),
+     RPG.gw.Q10=TRUE,
+     DonorSegments.gw=TRUE,
+     LinkageDrag=TRUE,
+     result.files=c("marker-01.RData","marker-02.RData")
+ )
> print(out)
```

Joint tabulation compares RPG, donor-segment number, and linkage drag. These summaries demonstrate that the high-level interface can work from actual parental marker genotypes as well as simulated homogeneous parents.

```
$par.summary
         marker-01-x      marker-02-x
Reps     "10000"          "10000"
DP       "ex-mabp1_m.mda" "ex-mabp1_m.mda"
Pop type "f1/bc/bc"       "f1/bc/bc"
Pop size "1/25/25"        "1/50/50"
Sel str  "n/t/t"          "n/tb/tb"
n Sel    "1/1/1"          "1/1/1"

$RPG.gw.Q10
   marker-01-x marker-02-x
P1        89.0        89.0
G0        94.5        94.5
```

```
G1        95.6        97.7
G2        96.9        99.0

$DonorSegments.gw.mean
   marker-01-x marker-02-x
P1        44.0        44.0
G0        22.0        22.0
G1        13.9         7.6
G2         9.2         2.8

$LinkageDrag1.mean
   marker-01-x marker-02-x
P1         9.7         9.7
G0         9.7         9.7
G1         9.4         9.1
G2         9.2         7.7
```

3.9 Example 9: different BC1 plants as donor parent

The non-recurrent starting parent need not be a homozygous line. Here six different BC1 plants, each represented by a two-allele marker file, are backcrossed for two additional generations. The selected BC3 descendants are compared for RPG, linkage drag, and number of donor segments.

```
> rng.init(seed=2109)
```

Each observed BC1 donor comparison uses the shared setup, its own seed, and the supplied individual marker files.

```
> mab.compare(
+     a.simulation.name="bc1i1tfb",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.donor.parent="ex-mabi1.mda",
+     a.recurrent.parent="ex-mabp2.mda",
+     a.linkage.map="ex-mab1.map",
+     a.target.loci="target",
+     a.flanking.loci=c("umc90","umc27"),
+     a.gen.type=c("bc","bc"),
+     a.population.size=c(100,100),
+     a.sel.strategy=c("tfb","tfb"),
+     b.simulation.name="bc1i2tfb",
+     b.donor.parent="ex-mabi2.mda",
+     c.simulation.name="bc1i3tfb",
+     c.donor.parent="ex-mabi3.mda",
+     d.simulation.name="bc1i4tfb",
+     d.donor.parent="ex-mabi4.mda",
+     e.simulation.name="bc1i5tfb",
+     e.donor.parent="ex-mabi5.mda",
+     f.simulation.name="bc1i6tfb",
+     f.donor.parent="ex-mabi6.mda",
+     result.files=paste0("bc1i",1:6,"tfb.RData")
+ )
```

Scenario A defines the common recurrent parent, map, target, flanking markers, and two subsequent backcrosses. Scenarios B–F inherit these settings and differ only in the BC1 donor genotype file.

```
> out <- mab.tabulate(
+     simulation.names.runs=as.vector(rbind(
+         paste0("bc1i",1:6,"tfb"),
+         rep("x",6)
+     )),
+     RPG.gw.Q10=TRUE,
+     DonorSegments.gw=TRUE,
+     LinkageDrag=TRUE,
+     result.files=paste0("bc1i",1:6,"tfb.RData")
+ )
> print(out)
```

The six resulting programs are compared for RPG, genome-wide donor-segment number, and linkage drag. This permits candidate BC1 plants to be evaluated by their expected consequences in later backcross generations.

```
$par.summary
         bc1i1tfb-x     bc1i2tfb-x     bc1i3tfb-x     bc1i4tfb-x
Reps     "10000"        "10000"        "10000"        "10000"
DP       "ex-mabi1.mda" "ex-mabi2.mda" "ex-mabi3.mda" "ex-mabi4.mda"
Pop type "bc/bc"        "bc/bc"        "bc/bc"        "bc/bc"
Pop size "100/100"      "100/100"      "100/100"      "100/100"
Sel str  "tfb/tfb"      "tfb/tfb"      "tfb/tfb"      "tfb/tfb"
n Sel    "1/1"          "1/1"          "1/1"          "1/1"
         bc1i5tfb-x     bc1i6tfb-x
Reps     "10000"        "10000"
DP       "ex-mabi5.mda" "ex-mabi6.mda"
Pop type "bc/bc"        "bc/bc"
Pop size "100/100"      "100/100"
Sel str  "tfb/tfb"      "tfb/tfb"
n Sel    "1/1"          "1/1"

$RPG.gw.Q10
   bc1i1tfb-x bc1i2tfb-x bc1i3tfb-x bc1i4tfb-x bc1i5tfb-x bc1i6tfb-x
P1       81.7       81.1       82.7       79.0       77.0       76.0
G0       90.0       89.5       90.3       88.5       89.8       89.3
G1       97.2       97.3       97.6       96.9       97.4       97.2

$DonorSegments.gw.mean
   bc1i1tfb-x bc1i2tfb-x bc1i3tfb-x bc1i4tfb-x bc1i5tfb-x bc1i6tfb-x
P1       12.0        9.0       11.0       12.0        9.0        9.0
G0        8.5        7.2        7.9        9.1        7.9        7.8
G1        3.7        3.2        3.3        4.3        3.9        3.6

$LinkageDrag1.mean
   bc1i1tfb-x bc1i2tfb-x bc1i3tfb-x bc1i4tfb-x bc1i5tfb-x bc1i6tfb-x
P1       38.0       38.0       26.5       38.0      109.0      126.6
G0       12.0       12.1       11.6       11.9       11.5       11.4
G1       10.8       10.8       10.8       10.6       10.8       10.8
```

3.10 Example 9a: analysis of input individuals without further simulation

If `gen.type`, `population.size`, and `sel.strategy` are omitted, the supplied parental/input individuals themselves can be analyzed. This allows the six BC1 plants from the preceding example to be compared without simulating additional generations.

```
> rng.init(seed=2110)
```

The analysis-only version uses the same observed BC1 donor files but does not define future generations.

```
> mab.compare(
+     a.simulation.name="bc1i1",
+     a.simulation.run="x",
+     a.donor.parent="ex-mabi1.mda",
+     a.recurrent.parent="ex-mabp2.mda",
+     a.linkage.map="ex-mab1.map",
+     a.target.loci="target",
+     a.flanking.loci=c("umc90","umc27"),
+     b.simulation.name="bc1i2",
+     b.donor.parent="ex-mabi2.mda",
+     c.simulation.name="bc1i3",
+     c.donor.parent="ex-mabi3.mda",
+     d.simulation.name="bc1i4",
+     d.donor.parent="ex-mabi4.mda",
+     e.simulation.name="bc1i5",
+     e.donor.parent="ex-mabi5.mda",
+     f.simulation.name="bc1i6",
+     f.donor.parent="ex-mabi6.mda",
+     result.files=paste0("bc1i",1:6,".RData")
+ )
```

`mab.compare()` therefore imports and evaluates the six input genotypes without simulating additional progeny. The inherited scenarios again differ only in the donor-parent file.

```
> out <- mab.tabulate(
+     simulation.names.runs=as.vector(rbind(
+         paste0("bc1i",1:6),
+         rep("x",6)
+     )),
+     RPG.gw.Q10=TRUE,
+     DonorSegments.gw=TRUE,
+     LinkageDrag=TRUE,
+     result.files=paste0("bc1i",1:6,".RData")
+ )
> print(out)
```

The tabulation compares the existing individuals for RPG, donor segments, and linkage drag. This is useful when selection decisions must be made from already genotyped backcross plants.

```
$par.summary
         bc1i1-x        bc1i2-x        bc1i3-x        bc1i4-x
Reps     "1000"         "1000"         "1000"         "1000"
DP       "ex-mabi1.mda" "ex-mabi2.mda" "ex-mabi3.mda" "ex-mabi4.mda"
Pop type ""             ""             ""             ""
Pop size ""             ""             ""             ""
Sel str  ""             ""             ""             ""
n Sel    ""             ""             ""             ""
         bc1i5-x        bc1i6-x
Reps     "1000"         "1000"
DP       "ex-mabi5.mda" "ex-mabi6.mda"
Pop type ""             ""
Pop size ""             ""
Sel str  ""             ""
n Sel    ""             ""

$RPG.gw.Q10
   bc1i1-x bc1i2-x bc1i3-x bc1i4-x bc1i5-x bc1i6-x
P1    81.7    81.1    82.7      79      77      76
```

```
$DonorSegments.gw.mean
   bc1i1-x bc1i2-x bc1i3-x bc1i4-x bc1i5-x bc1i6-x
P1      12       9      11      12       9       9

$LinkageDrag1.mean
   bc1i1-x bc1i2-x bc1i3-x bc1i4-x bc1i5-x bc1i6-x
P1      38      38    26.5      38     109   126.6
```

3.11 Example 10: selection for a low number of donor segments

Strategy `tfs` combines target selection, flanking-marker preselection, and selection for a low number of donor chromosome segments. The final segment-selection step is treated as an HT assay. The same six BC1 starting plants are compared.

```
> rng.init(seed=2111)
```

The donor-segment example uses the shared setup, its own seed, and the same six observed BC1 donor candidates.

```
> mab.compare(
+     a.simulation.name="bc1i1tfs",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.donor.parent="ex-mabi1.mda",
+     a.recurrent.parent="ex-mabp2.mda",
+     a.linkage.map="ex-mab1.map",
+     a.target.loci="target",
+     a.flanking.loci=c("umc90","umc27"),
+     a.gen.type=c("bc","bc"),
+     a.population.size=c(100,100),
+     a.sel.strategy=c("tfs","tfs"),
+     a.no.selected=c(1,1),
+     b.simulation.name="bc1i2tfs",
+     b.donor.parent="ex-mabi2.mda",
+     c.simulation.name="bc1i3tfs",
+     c.donor.parent="ex-mabi3.mda",
+     d.simulation.name="bc1i4tfs",
+     d.donor.parent="ex-mabi4.mda",
+     e.simulation.name="bc1i5tfs",
+     e.donor.parent="ex-mabi5.mda",
+     f.simulation.name="bc1i6tfs",
+     f.donor.parent="ex-mabi6.mda",
+     result.files=paste0("bc1i",1:6,"tfs.RData")
+ )
```

Strategy `tfs` applies target and flanking-marker selection and then favors plants with few donor chromosome segments. The six inherited scenarios differ only in which observed BC1 genotype starts the program.

```
> out <- mab.tabulate(
+     simulation.names.runs=as.vector(rbind(
+         paste0("bc1i",1:6,"tfs"),
+         rep("x",6)
+     )),
+     RPG.gw.Q10=TRUE,
+     DonorSegments.gw=TRUE,
```

```
+     LinkageDrag=TRUE,
+     result.files=paste0("bc1i",1:6,"tfs.RData")
+ )
> print(out)
```

The resulting RPG, donor-segment, and linkage-drag summaries show how explicit selection against donor segments can be used as an alternative background criterion.

```
$par.summary
         bc1i1tfs-x     bc1i2tfs-x     bc1i3tfs-x     bc1i4tfs-x
Reps     "10000"        "10000"        "10000"        "10000"
DP       "ex-mabi1.mda" "ex-mabi2.mda" "ex-mabi3.mda" "ex-mabi4.mda"
Pop type "bc/bc"        "bc/bc"        "bc/bc"        "bc/bc"
Pop size "100/100"      "100/100"      "100/100"      "100/100"
Sel str  "tfs/tfs"      "tfs/tfs"      "tfs/tfs"      "tfs/tfs"
n Sel    "1/1"          "1/1"          "1/1"          "1/1"
         bc1i5tfs-x     bc1i6tfs-x
Reps     "10000"        "10000"
DP       "ex-mabi5.mda" "ex-mabi6.mda"
Pop type "bc/bc"        "bc/bc"
Pop size "100/100"      "100/100"
Sel str  "tfs/tfs"      "tfs/tfs"
n Sel    "1/1"          "1/1"

$RPG.gw.Q10
   bc1i1tfs-x bc1i2tfs-x bc1i3tfs-x bc1i4tfs-x bc1i5tfs-x bc1i6tfs-x
P1       81.7       81.1       82.7       79.0       77.0       76.0
G0       89.1       88.3       89.2       87.0       87.2       86.9
G1       96.4       96.2       96.1       95.1       95.3       95.4

$DonorSegments.gw.mean
   bc1i1tfs-x bc1i2tfs-x bc1i3tfs-x bc1i4tfs-x bc1i5tfs-x bc1i6tfs-x
P1       12.0        9.0       11.0       12.0        9.0        9.0
G0        7.9        6.5        7.2        8.3        6.9        6.7
G1        2.7        2.2        2.4        3.1        2.8        2.5

$LinkageDrag1.mean
   bc1i1tfs-x bc1i2tfs-x bc1i3tfs-x bc1i4tfs-x bc1i5tfs-x bc1i6tfs-x
P1       38.0       38.0       26.5       38.0      109.0      126.6
G0       12.1       12.1       11.7       12.0       11.5       11.5
G1       10.8       10.8       10.7       10.6       10.8       10.9
```

3.12 Example 11: analysis of chromosome regions

RPG and donor segments can be evaluated for user-defined regions. The example defines chromosome 1, chromosome 5, and the interval 0–60 cM on chromosome 5. A zero value for `reg.end` means the complete chromosome.

```
> rng.init(seed=2112)
```

This regional-analysis example uses a target-only program with large backcross populations and its own simulation seed.

```
> mab.simulate(
+     simulation.name="regions-1",
+     simulation.run="x",
+     repetitions=NREP,
```

```
+     linkage.map="ex-mab1.map",
+     target.loci="target",
+     flanking.loci=c("umc90","umc27"),
+     gen.type=c("f1","bc","bc","bc"),
+     population.size=c(1,500,500,500),
+     sel.strategy=c("n","t","t","t"),
+     no.selected=c(1,1,1,1),
+     reg.chr=c(1,5,5),
+     reg.begin=c(0,0,0),
+     reg.end=c(0,0,60),
+     result.file="regions-1.RData"
+ )
```

Three regions are requested: complete chromosome 1, complete target chromosome 5, and the 0–60 cM interval on chromosome 5. Region definitions are stored with the simulation and evaluated along with genome-wide quantities.

```
> out <- mab.tabulate(
+     simulation.names.runs=c("regions-1","x"),
+     ext.summary=TRUE,
+     RPG.gw.mean=TRUE,
+     RPG.reg.mean=TRUE,
+     DonorSegments.gw=TRUE,
+     DonorSegments.reg=TRUE,
+     result.files="regions-1.RData"
+ )
> print(out)
```

The tabulation asks for mean RPG and donor-segment counts both genome-wide and by region. Regional output is useful when recovery near the target or in another biologically important chromosome segment matters separately from whole-genome recovery.

```
$ext.summary
          regions-1-x
Reps      "10000"
RP        "H"
DP        "H"
Map       "ex-mab1.map"
Target    "target"
Flm       "umc90/umc27"
Pop type  "f1/bc/bc/bc"
Pop size  "1/500/500/500"
Sel str   "n/t/t/t"
n Sel     "1/1/1/1"
n PreSel  ""
Succ f    "1"
Reg chr   "1/5/5"
Reg beg   "0/0/0"
Reg end   "0/0/60"

$RPG.gw.mean
   regions-1-x
P1         0.0
G0        50.0
G1        73.9
G2        86.1
G3        92.4

$RPG.reg1.mean
   regions-1-x
P1         0.0
```

```
G0        50.0
G1        75.0
G2        87.4
G3        93.7

$RPG.reg2.mean
   regions-1-x
P1         0.0
G0        50.0
G1        63.7
G2        72.8
G3        78.9

$RPG.reg3.mean
   regions-1-x
P1         0.0
G0         1.3
G1         1.6
G2         1.9
G3         2.0

$DonorSegments.gw.mean
   regions-1-x
P1        20.0
G0        10.0
G1        13.2
G2        11.1
G3         8.2

$DonorSegments.reg1.mean
   regions-1-x
P1         2.0
G0         1.0
G1         1.8
G2         1.6
G3         1.1

$DonorSegments.reg2.mean
   regions-1-x
P1         2.0
G0         1.0
G1         1.4
G2         1.6
G3         1.6

$DonorSegments.reg3.mean
   regions-1-x
P1         2.0
G0         1.0
G1         1.4
G2         1.6
G3         1.6
```

3.13 Example 12: preselection for recurrent-parent alleles

`recipient.loci` defines markers at which recurrent-parent alleles must be retained. The historical example places these loci on chromosome 2 and therefore also evaluates RPG for chromosome 2 separately.

The recipient-locus comparison uses strategies `tr`, `trb`, `trfb`, and `tfrb`. In `tr` and `trb`, recipient-locus selection follows target selection. `trfb` adds flanking and background selection

after the recipient-locus step. `tfrb` performs flanking selection before recipient-locus and background selection. The comparison includes these strategies together with `t`, `tf`, and `tfb`. To run the simulations as independent units, the example runs each strategy with its own fixed seed and tabulates the seven result files jointly.

```
> recipient <- c(
+     "bnl845","umc53","umc6","umc34","umc131",
+     "umc98a","umc4","umc10","bnl844b","umc36"
+ )
```

The recipient-locus comparison first defines the common recipient-locus panel after the shared setup. These markers represent recurrent-parent alleles that should be retained.

```
> run.recipient <- function(name,strategy,seed) {
+     rng.init(seed=seed)
+     mab.simulate(
+         simulation.name=name,
+         simulation.run="x",
+         repetitions=NREP,
+         linkage.map="ex-mab1.map",
+         target.loci="target",
+         flanking.loci=c("umc90","umc27"),
+         recipient.loci=recipient,
+         gen.type=c("f1","bc","bc","bc"),
+         population.size=c(1,1000,1000,1000),
+         sel.strategy=c("n",strategy,strategy,strategy),
+         no.selected=c(1,1,1,1),
+         reg.chr=c(1,2,5,5),
+         reg.begin=c(0,0,0,0),
+         reg.end=c(0,0,0,60),
+         result.file=paste0(name,".RData")
+     )
+ }
```

The helper function packages the common breeding design. Only the scenario name, selection strategy, and seed vary; every strategy is therefore simulated as a standalone program with the same populations, regions, target, flanking markers, and recipient loci.

```
> run.recipient("ss-t",   "t",   21131)
> run.recipient("ss-tf",  "tf",  21132)
> run.recipient("ss-tr",  "tr",  21133)
> run.recipient("ss-tfb", "tfb", 21134)
> run.recipient("ss-trb", "trb", 21135)
> run.recipient("ss-trfb","trfb",21136)
> run.recipient("ss-tfrb","tfrb",21137)
```

Seven calls cover `t`, `tf`, `tr`, `tfb`, `trb`, `trfb`, and `tfrb`. Separate calls initialize both R and the internal random-number generators for each scenario while preserving a joint comparison.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "ss-t","x",
+         "ss-tf","x",
+         "ss-tr","x",
+         "ss-tfb","x",
+         "ss-trb","x",
+         "ss-trfb","x",
+         "ss-tfrb","x"
```

```
+     ),
+     ext.summary=TRUE,
+     RPG.gw.mean=TRUE,
+     RPG.reg.mean=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files=c(
+         "ss-t.RData","ss-tf.RData","ss-tr.RData",
+         "ss-tfb.RData","ss-trb.RData",
+         "ss-trfb.RData","ss-tfrb.RData"
+     )
+ )
> print(out)
```

The seven saved result files are tabulated together. Extended parameter summaries, genome-wide and regional RPG, linkage drag, and marker analyses reveal how the ordering and inclusion of recipient-, flanking-, and background-selection stages change the program.

```
$ext.summary
         ss-t-x              ss-tf-x             ss-tr-x
Reps     "10000"             "10000"             "10000"
RP       "H"                 "H"                 "H"
DP       "H"                 "H"                 "H"
Map      "ex-mab1.map"       "ex-mab1.map"       "ex-mab1.map"
Target   "target"            "target"            "target"
Flm      "umc90/umc27"       "umc90/umc27"       "umc90/umc27"
Pop type "f1/bc/bc/bc"       "f1/bc/bc/bc"       "f1/bc/bc/bc"
Pop size "1/1000/1000/1000" "1/1000/1000/1000" "1/1000/1000/1000"
Sel str  "n/t/t/t"           "n/tf/tf/tf"        "n/tr/tr/tr"
n Sel    "1/1/1/1"           "1/1/1/1"           "1/1/1/1"
n PreSel ""                  ""                  ""
Succ f   "1"                 "1"                 "1"
Reg chr  "1/2/5/5"           "1/2/5/5"           "1/2/5/5"
Reg beg  "0/0/0/0"           "0/0/0/0"           "0/0/0/0"
Reg end  "0/0/0/60"          "0/0/0/60"          "0/0/0/60"
         ss-tfb-x            ss-trb-x            ss-trfb-x
Reps     "10000"             "10000"             "10000"
RP       "H"                 "H"                 "H"
DP       "H"                 "H"                 "H"
Map      "ex-mab1.map"       "ex-mab1.map"       "ex-mab1.map"
Target   "target"            "target"            "target"
Flm      "umc90/umc27"       "umc90/umc27"       "umc90/umc27"
Pop type "f1/bc/bc/bc"       "f1/bc/bc/bc"       "f1/bc/bc/bc"
Pop size "1/1000/1000/1000" "1/1000/1000/1000" "1/1000/1000/1000"
Sel str  "n/tfb/tfb/tfb"     "n/trb/trb/trb"     "n/trfb/trfb/trfb"
n Sel    "1/1/1/1"           "1/1/1/1"           "1/1/1/1"
n PreSel ""                  ""                  ""
Succ f   "1"                 "1"                 "1"
Reg chr  "1/2/5/5"           "1/2/5/5"           "1/2/5/5"
Reg beg  "0/0/0/0"           "0/0/0/0"           "0/0/0/0"
Reg end  "0/0/0/60"          "0/0/0/60"          "0/0/0/60"
         ss-tfrb-x
Reps     "10000"
RP       "H"
DP       "H"
Map      "ex-mab1.map"
Target   "target"
Flm      "umc90/umc27"
Pop type "f1/bc/bc/bc"
Pop size "1/1000/1000/1000"
Sel str  "n/tfrb/tfrb/tfrb"
n Sel    "1/1/1/1"
n PreSel ""
```

```
Succ f   "1"
Reg chr  "1/2/5/5"
Reg beg  "0/0/0/0"
Reg end  "0/0/0/60"

$RPG.gw.mean
   ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
P1    0.0     0.0     0.0      0.0      0.0       0.0       0.0
G0   50.0    50.0    50.0     50.0     50.0      50.0      50.0
G1   74.0    76.0    76.6     80.4     86.4      82.4      78.4
G2   86.1    87.7    87.4     95.9     97.3      96.4      96.1
G3   92.3    93.6    93.0     98.3     99.0      98.9      98.9

$RPG.reg1.mean
   ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
P1    0.0     0.0     0.0      0.0      0.0       0.0       0.0
G0   50.0    50.0    50.0     50.0     50.0      50.0      50.0
G1   75.1    74.9    75.2     79.8     85.4      79.7      75.5
G2   87.5    87.5    87.4     97.0     97.7      96.0      95.5
G3   93.7    93.8    93.8     99.0     99.1      99.0      98.9

$RPG.reg2.mean
   ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
P1    0.0     0.0     0.0      0.0      0.0       0.0       0.0
G0   50.0    50.0    50.0     50.0     50.0      50.0      50.0
G1   75.0    75.0    99.6     74.9     99.5      99.5      90.6
G2   87.5    87.5    99.8     87.5     99.8      99.8      99.6
G3   93.9    93.7    99.9     93.7     99.9      99.9      99.8

$RPG.reg3.mean
   ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
P1    0.0     0.0     0.0      0.0      0.0       0.0       0.0
G0   50.0    50.0    50.0     50.0     50.0      50.0      50.0
G1   63.7    85.3    63.9     87.9     73.1      81.7      85.3
G2   72.8    89.7    72.9     94.7     90.2      94.1      93.6
G3   78.9    92.2    79.0     96.1     96.0      96.1      96.1

$RPG.reg4.mean
   ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
P1    0.0     0.0     0.0      0.0      0.0       0.0       0.0
G0    1.3     1.3     1.3      1.3      1.3       1.3       1.3
G1    1.6     2.2     1.6      2.2      1.9       2.1       2.2
G2    1.9     2.3     1.9      2.4      2.3       2.4       2.4
G3    2.0     2.4     2.0      2.5      2.4       2.5       2.5

$LinkageDrag1.mean
   ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
P1  153.1   153.1   153.1    153.1    153.1     153.1     153.1
G0  153.1   153.1   153.1    153.1    153.1     153.1     153.1
G1   97.0    11.9    96.7     11.8     71.3      32.9      11.8
G2   68.5    11.0    68.4     10.9     27.0      11.0      10.9
G3   52.6    10.5    52.3     10.3     11.0      10.5      10.3

$SM.mean
      ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
Total      0  1006.7       0   1004.8        0         0    1005.4
G0         0     0.0       0      0.0        0         0       0.0
G1         0  1000.2       0   1000.1        0         0     999.8
G2         0     6.4       0      4.7        0         0       5.7
G3         0     0.0       0      0.0        0         0       0.0

$HT.mean
      ss-t-x ss-tf-x ss-tr-x ss-tfb-x ss-trb-x ss-trfb-x ss-tfrb-x
Total      0       0  1499.8   1000.5   1499.9    1500.0    1000.5
G0         0       0     0.0      0.0      0.0       0.0       0.0
```

```
G1         0       0   500.0      5.3    499.8     499.8       5.5
G2         0       0   499.9    495.3    500.1     500.1     495.0
G3         0       0   499.9    499.9    500.0     500.1     500.0
```

3.14 Example 13: success factor

The `success.factor` is the probability that a selected plant contributes to the next generation, thereby representing survival and fertility. The example compares ten selected plants with success probability 1, ten selected plants with probability 0.5, and five selected plants with probability 1.

```
> rng.init(seed=2114)
```

The success-factor comparison is self-contained and fixes one common random seed.

```
> mab.compare(
+     a.simulation.name="success-1",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.linkage.map="ex-mab1.map",
+     a.target.loci="target",
+     a.flanking.loci=c("umc90","umc27"),
+     a.gen.type=c("f1","bc","bc","bc","bc","bc"),
+     a.population.size=c(1,100,100,100,100,100),
+     a.sel.strategy=c("n","tb","tb","tb","tb","tb"),
+     a.no.selected=c(1,10,10,10,10,10),
+     a.success.factor=1,
+     b.simulation.name="success-2",
+     b.no.selected=c(1,10,10,10,10,10),
+     b.success.factor=0.5,
+     c.simulation.name="success-3",
+     c.no.selected=c(1,5,5,5,5,5),
+     c.success.factor=1,
+     result.files=c(
+         "success-1.RData",
+         "success-2.RData",
+         "success-3.RData"
+     )
+ )
```

Scenario A selects ten plants per generation with success factor 1. Scenario B gives each selected plant only 0.5 probability of contributing to the next generation, while scenario C selects five plants with success factor 1.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "success-1","x",
+         "success-2","x",
+         "success-3","x"
+     ),
+     ext.summary=TRUE,
+     RPG.gw.mean=TRUE,
+     RPG.gw.Q10=TRUE,
+     result.files=c(
+         "success-1.RData",
+         "success-2.RData",
+         "success-3.RData"
```

```
+     )
+ )
> print(out)
```

The joint summary reports mean and lower-tail RPG. This separates the effect of biological success or fertility from the nominal number of plants selected by the breeding program.

```
$ext.summary
         success-1-x             success-2-x
Reps     "10000"                 "10000"
RP       "H"                     "H"
DP       "H"                     "H"
Map      "ex-mab1.map"           "ex-mab1.map"
Target   "target"                "target"
Flm      "umc90/umc27"           "umc90/umc27"
Pop type "f1/bc/bc/bc/bc/bc"     "f1/bc/bc/bc/bc/bc"
Pop size "1/100/100/100/100/100" "1/100/100/100/100/100"
Sel str  "n/tb/tb/tb/tb/tb"      "n/tb/tb/tb/tb/tb"
n Sel    "1/10/10/10/10/10"      "1/10/10/10/10/10"
n PreSel ""                      ""
Succ f   "1"                     "0.5"
         success-3-x
Reps     "10000"
RP       "H"
DP       "H"
Map      "ex-mab1.map"
Target   "target"
Flm      "umc90/umc27"
Pop type "f1/bc/bc/bc/bc/bc"
Pop size "1/100/100/100/100/100"
Sel str  "n/tb/tb/tb/tb/tb"
n Sel    "1/5/5/5/5/5"
n PreSel ""
Succ f   "1"

$RPG.gw.mean
   success-1-x success-2-x success-3-x
P1         0.0         0.0         0.0
G0        50.0        50.0        50.0
G1        80.4        80.4        81.9
G2        92.9        92.9        94.2
G3        97.5        97.4        98.1
G4        99.0        98.9        99.2
G5        99.4        99.4        99.5

$RPG.gw.Q10
   success-1-x success-2-x success-3-x
P1         0.0         0.0         0.0
G0        50.0        50.0        50.0
G1        76.6        76.6        78.3
G2        90.4        90.3        91.8
G3        96.0        95.9        96.8
G4        98.1        98.0        98.3
G5        98.8        98.8        98.9
```

3.15 Example 14: recoding parental input files

With `recode.infiles=TRUE`, parental marker data can be converted to the internal donor/recurrent-parent coding and written to an explicitly supplied `recode.file`. In the

recoded representation the recurrent-parent allele is coded as 8. Both `result.file` and `recode.file` must be supplied explicitly.

```
> rng.init(seed=2115)
```

The recoding example uses observed parental marker data and a separate seed but does not need to simulate a breeding program.

```
> mab.simulate(
+     simulation.name="rec-01",
+     simulation.run="x",
+     donor.parent="ex-mabp1_m.mda",
+     recurrent.parent="ex-mabp2_m.mda",
+     linkage.map="ex-mab4.map",
+     result.file="rec-01.RData",
+     recode.infiles=TRUE,
+     recode.file="recoded-parents.mda"
+ )
```

With `recode.infiles=TRUE`, `mab.simulate()` creates recoded parental input in which alleles are expressed relative to donor and recurrent parents. The linkage map and missing-allele code determine which marker records can be used.

```
> out <- list(
+     result=names(mab.load.data(
+         simulation.name="rec-01",
+         simulation.run="x",
+         result.file="rec-01.RData"
+     )),
+     recoded.file=head(
+         readLines(file.path(st.output.dir,"recoded-parents.mda")),
+         12
+     )
+ )
> print(out)
```

The example then locates the recoded output and prints its first lines. This makes the generated file format visible without filling the vignette with the complete recoded marker matrix.

```
$result
 [1] "simulation.name"  "simulation.run"   "repetitions"
 [4] "recurrent.parent" "donor.parent"     "linkage.map"
 [7] "sel.strategy"     "no.selected"      "success.factor"
[10] "RPG.gw"           "DonorSegments.gw"

$recoded.file
 [1] "01E0103DH03 01E0109DH02" "s1e0038xxx.1  1 0"
 [3] "s1e0038xxx.8  0 1"       "s1e1093xxx.1  0 0"
 [5] "s1e1093xxx.8  1 1"       "s1e2309xxx.1  0 0"
 [7] "s1e2309xxx.8  1 1"       "s1e2406xxx.1  0 0"
 [9] "s1e2406xxx.8  1 1"       "s1e2486xxx.1  0 0"
[11] "s1e2486xxx.8  1 1"       "s1e2522xxx.1  0 0"
```

3.16 Example 15: minimum number of preselected individuals

For strategies containing the flanking-marker step `f`, the default is to retain all individuals having the maximum number of favorable flanking markers. This can produce a very small preselected set. For example, if only two plants show recombination on both sides of a target locus, only those two would proceed to the HT assay in a `tfb` strategy.

`no.preselected` imposes a minimum size on that intermediate set. If five plants are requested and only two attain the best flanking-marker class, three additional plants from the next-best class are added. The example compares the default behavior with a minimum of five preselected plants in each backcross generation.

```
> rng.init(seed=2116)
```

The final example again defines all common settings locally so the comparison can be regenerated by itself.

```
> mab.compare(
+     a.simulation.name="sel-50-0a-01",
+     a.simulation.run="x",
+     a.repetitions=NREP,
+     a.linkage.map="ex-mab1.map",
+     a.target.loci="target",
+     a.flanking.loci=c("umc90","umc27"),
+     a.gen.type=c("f1","bc","bc","bc"),
+     a.population.size=c(1,50,50,50),
+     a.sel.strategy=c("n","tfb","tfb","tfb"),
+     b.simulation.name="sel-50-05-01",
+     b.no.preselected=c(1,5,5,5),
+     result.files=c("sel-50-0a-01.RData","sel-50-05-01.RData")
+ )
```

Both scenarios use `tfb` with population size 50. Scenario B sets `no.preselected=5`, forcing at least five plants to pass the flanking-marker preselection even when fewer individuals carry the maximum number of favorable flanking alleles.

```
> out <- mab.tabulate(
+     simulation.names.runs=c(
+         "sel-50-0a-01","x",
+         "sel-50-05-01","x"
+     ),
+     ext.summary=TRUE,
+     RPG.gw.Q10=TRUE,
+     LinkageDrag=TRUE,
+     MarkerAnalyses=TRUE,
+     result.files=c("sel-50-0a-01.RData","sel-50-05-01.RData")
+ )
> print(out)
```

The two programs are compared for the lower RPG quantile, linkage drag, and marker analyses. The example shows explicitly how relaxing the flanking preselection cutoff changes both selection opportunity and genotyping effort.

```
$ext.summary
        sel-50-0a-01-x  sel-50-05-01-x
Reps     "10000"         "10000"
RP       "H"             "H"
DP       "H"             "H"
```

```
Map      "ex-mab1.map"   "ex-mab1.map"
Target   "target"        "target"
Flm      "umc90/umc27"   "umc90/umc27"
Pop type "f1/bc/bc/bc"   "f1/bc/bc/bc"
Pop size "1/50/50/50"    "1/50/50/50"
Sel str  "n/tfb/tfb/tfb" "n/tfb/tfb/tfb"
n Sel    "1/1/1/1"       "1/1/1/1"
n PreSel ""              "1/5/5/5"
Succ f   "1"             "1"

$RPG.gw.Q10
   sel-50-0a-01-x sel-50-05-01-x
P1            0.0            0.0
G0           50.0           50.0
G1           73.3           75.7
G2           87.9           89.7
G3           96.2           96.2

$LinkageDrag1.mean
   sel-50-0a-01-x sel-50-05-01-x
P1          153.1          153.1
G0          153.1          153.1
G1           39.6           50.0
G2           13.7           23.3
G3           10.9           14.3

$SM.mean
      sel-50-0a-01-x sel-50-05-01-x
Total           73.2           89.5
G0               0.0            0.0
G1              50.1           50.0
G2              20.2           26.7
G3               3.0           12.8

$HT.mean
      sel-50-0a-01-x sel-50-05-01-x
Total           36.2           30.9
G0               0.0            0.0
G1               4.1            5.6
G2               9.5            8.7
G3              22.7           16.6
```

References

Frisch M, Bohn M, Melchinger AE (1999) Comparison of selection strategies for marker-assisted backcrossing of a gene. Crop Science 39:1295–1301. doi:10.2135/cropsci1999.3951295x.

Frisch M, Melchinger AE (2001) Marker-assisted backcrossing for simultaneous introgression of two genes. Crop Science 41:1716–1725. doi:10.2135/cropsci2001.1716.