Introduction to MAPpoly

Marcelo Mollinari

2022-01-09

MAPpoly is an R package to construct genetic maps in autopolyploids with even ploidy levels. This quick start guide will present some essential functions to construct a tetraploid potato map. Please refer to MAPpoly's Reference Manual and the Extended Tutorial for a comprehensive description of all functions.

Reading data set

There are several functions to read discrete and probabilistic dosage-based genotype data sets in MAPpoly. You can read a data set from TXT, CSV, VCF, fitPoly-generated or import it from the R packages polyRAD, polymapR, and updog. The data set distributed along with MAPpoly is a subset of markers from (Pereira et al., 2021) in CSV format. Let us read it into MAPpoly

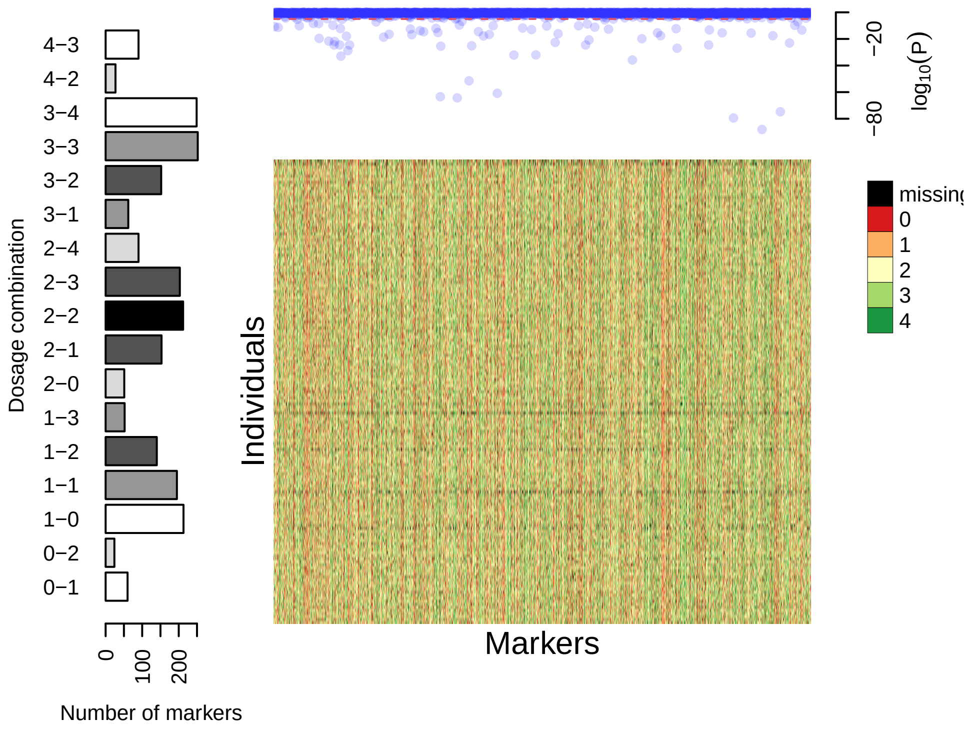
```r
file.name <- system.file("extdata/potato_example.csv", package = "mappoly")
dat <- read_geno_csv(file.in = file.name, ploidy = 4)
```

```
## Reading the following data:
##     Ploidy level: 4
##     No. individuals:  153
##     No. markers:  2225
##     No. informative markers:  2225 (100%)
##     ...
##     Done with reading.
##     Filtering non-conforming markers.
##     ...
##     Done with filtering.
```

```r
print(dat, detailed = T)
```

```
## This is an object of class 'mappoly.data'
##     Ploidy level:                            4
##     No. individuals:                         153
##     No. markers:                             2225
##     Missing data under  prob. threshold: 1.07%
##
##     ----------
##     No. markers per chromosome:
##        seq No.mrk
##          1    258
##         10    114
##         11    150
##         12    122
##          2    214
##          3    197
##          4    207
##          5    168
##          6    202
##          7    221
##          8    179
##          9    179
##     ----------
##     Markers with no chromosome information: 14
##     ----------
##     No. of markers per dosage combination in both parents:
##     P1 P2 freq
##      0  1   60
##      0  2   24
##      1  0  213
##      1  1  195
##      1  2  140
##      1  3   52
##      2  0   51
##      2  1  153
##      2  2  212
##      2  3  203
##      2  4   90
##      3  1   62
##      3  2  152
##      3  3  252
##      3  4  249
##      4  2   27
##      4  3   90
```

```
plot(dat)
```



The output figure shows a bar plot on the left-hand side with the number of markers in each allele dosage combination in P_1 and P_2, respectively. The upper-right plot contains the $\log_{10}(p - value)$ from χ^2 tests for all markers, considering the expected segregation patterns under Mendelian inheritance.

Data quality control

Quality control (QC) procedures are fundamental to identify:

- Individuals from crosses other than $P_1 \times P_2$
- Individuals and markers that exceeds a defined threshold of missing data points
- Markers with distorted segregation
- Markers with the same genotypic information (redundant markers). Removed markers are positioned into the final map.

Depending on the data set, these procedures can be conducted in any order. Let us first remove individuals from crosses other than $P_1 \times P_2$. When using the interactive function, the user needs to select a polygon around the individuals to be removed by clicking its vertices and pressing Esc.

```
dat <- filter_individuals(dat)
```

```
## Initial data:
##  Number of Individuals: 155
##  Number of Markers: 2225
##
## Missing data check:
##  Total SNPs: 2225
##   0 SNPs dropped due to missing data threshold of 1
##  Total of: 2225  SNPs
## MAF check:
##  No SNPs with MAF below 0
## Monomorphic check:
##  No monomorphic SNPs
## Summary check:
##  Initial:  2225 SNPs
##  Final:  2225  SNPs ( 0  SNPs removed)
##
## Completed! Time = 0.51  seconds
```

Now, let us filter out markers and individuals with more than 5% of missing data. You can update the threshold interactively

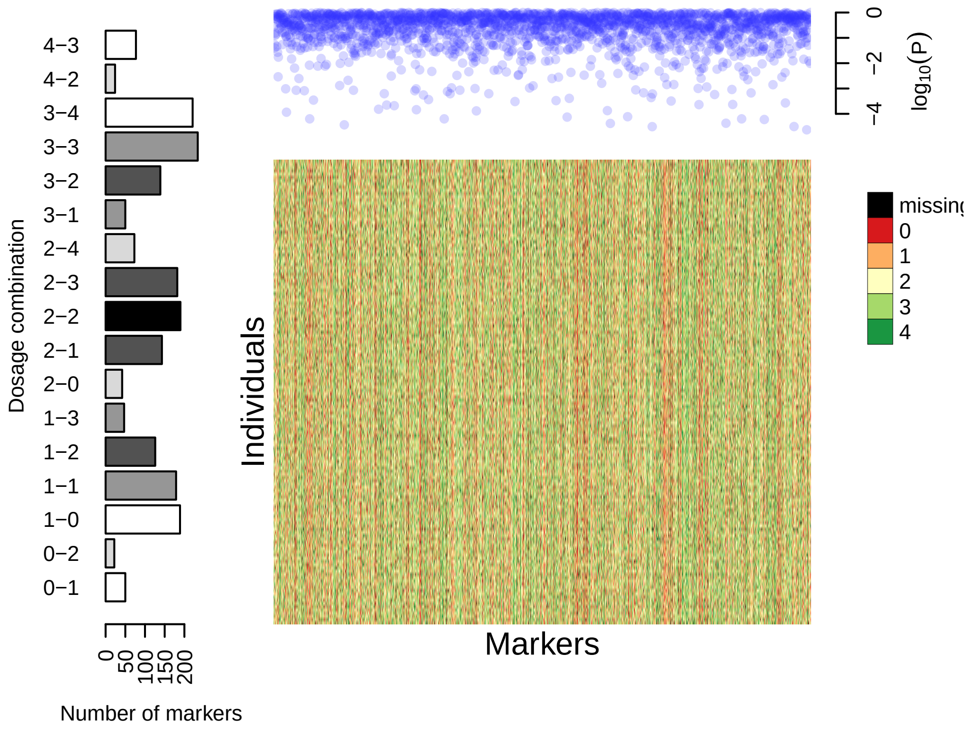
```
dat <- filter_missing(dat, type = "marker", filter.thres = .05)
dat <- filter_missing(dat, type = "individual", filter.thres = .05)
```

Finally, we can filter out markers with distorted segregation and redundant information. At this point, we do not consider preferential pairing and double reduction.

```
seq.filt <- filter_segregation(dat, chisq.pval.thres = 0.05/dat$n.mrk)
seq.filt <- make_seq_mappoly(seq.filt)
seq.red  <- elim_redundant(seq.filt)
```

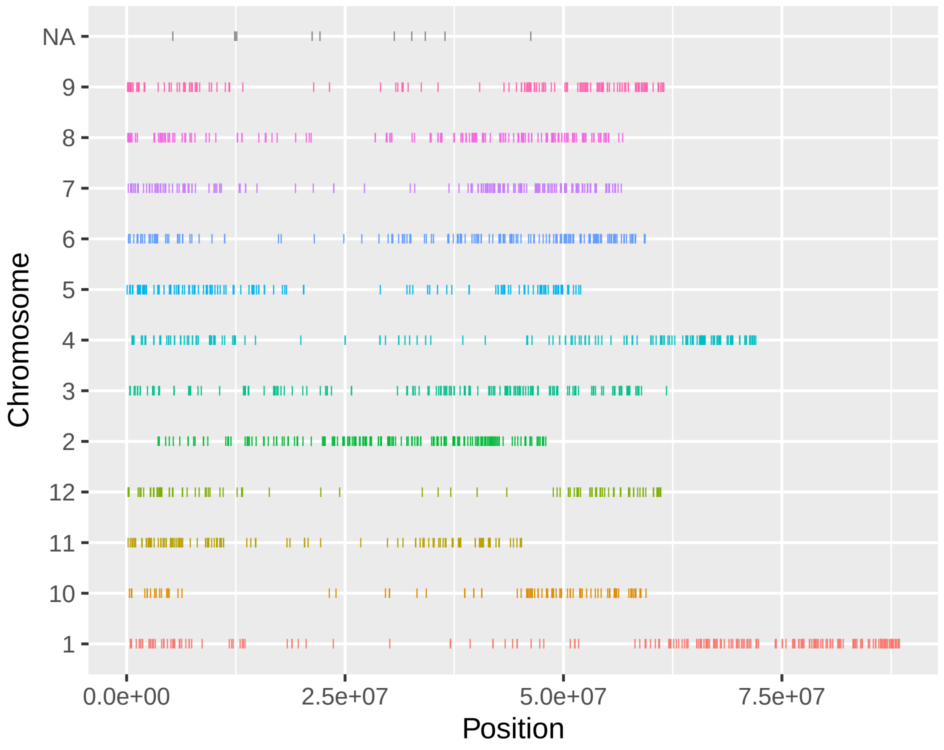
After filtering, 1990 markers were left to be mapped. Now, let us create a sequence of ordered markers to proceed with the analysis. You can also plot the data set for a specific sequence of markers

```
seq.init <- make_seq_mappoly(seq.red)
plot(seq.init)
```



and check the distribution of the markers in the reference genome, if the information is available

```
go <- get_genomic_order(input.seq = seq.init) ## get genomic order of the sequence
plot(go)
```



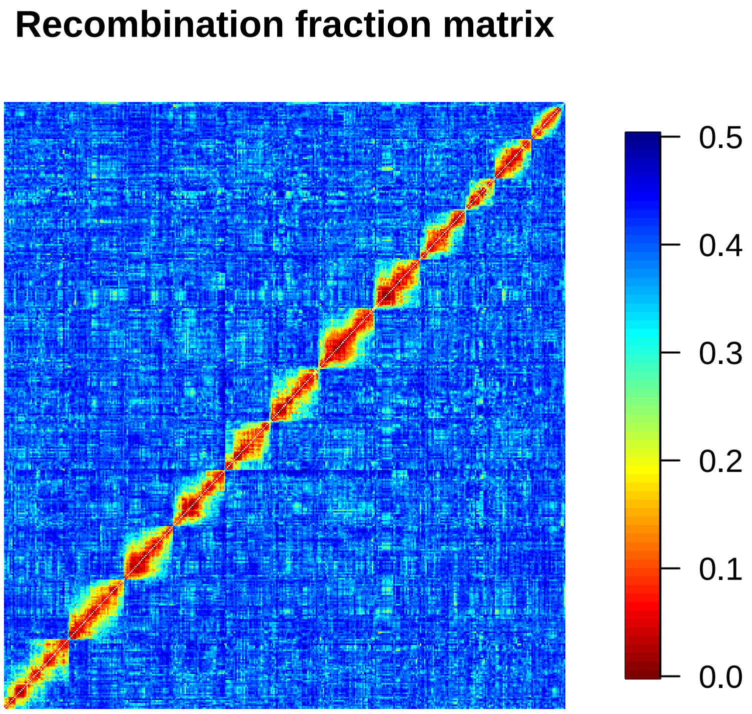
Two-point analysis

The two-point analysis calculates the pairwise recombination fraction in a sequence of markers. At this point of the analysis, where we have many markers, we use the function `est_pairwise_rf2`, which has a less detailed output than the original `est_pairwise_rf` (which will be used later) but can handle tens of thousands of markers, even when using a personal computer. Nevertheless, the analysis can take a while depending on the number of markers if few cores are available.

```
ncores <- parallel::detectCores() - 1
tpt <- est_pairwise_rf2(seq.init, ncpus = ncores)
```

```
## Going RcppParallel mode.
## ...
## Done with pairwise computation.
## ~~~~~~~
## Reorganizing results
## ...
```

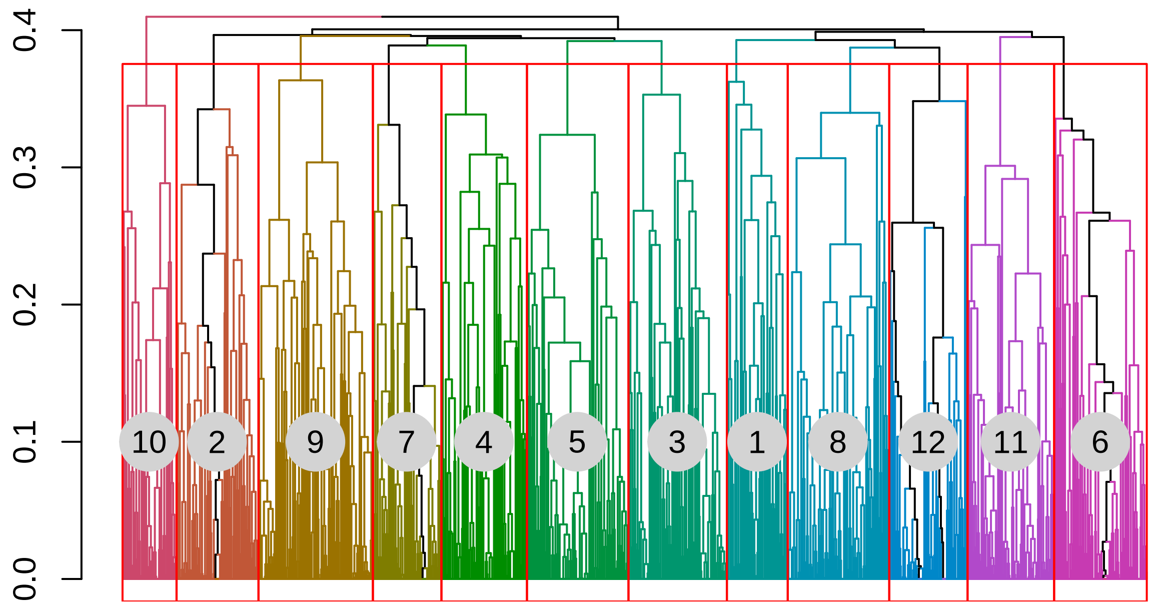
```
m <- rf_list_to_matrix(tpt) ## converts rec. frac. list into a matrix
sgo <- make_seq_mappoly(go) ## creates a sequence of markers in the genome order
plot(m, ord = sgo, fact = 5) ## plots a rec. frac. matrix using the genome order, averaging neighbor ce
```



Grouping

We can cluster the markers in linkage groups by using function `group_mappoly`. The function uses the recombination fraction matrix and UPGMA method to group markers. Use the option `comp.mat = TRUE` to compare the linkage-based clustering results with the chromosome information. If your data set does not contain chromosome information, use the option `comp.mat = FALSE`. You also can use the interactive version to change the number of expected groups

```
g <- group_mappoly(m, expected.groups = 12, comp.mat = TRUE)
plot(g)
```



```
g
```

```
##   This is an object of class 'mappoly.group'
##   ------------------------------------------
##   Criteria used to assign markers to groups:
##
##     - Number of markers:          1988
##     - Number of linkage groups:   12
##     - Number of markers per linkage groups:
##     group n.mrk
##         1   118
##         2   159
##         3   191
##         4   166
##         5   197
##         6   180
##         7   133
##         8   197
##         9   222
##        10   105
##        11   168
##        12   152
##   ------------------------------------------
##     10   9   4   5   2   3  11   7   1  12   6   8 NoChr
## 1   88   0   3   5   7   5   3   1   2   2   0   1     1
## 2    0 147   1   0   2   0   1   2   2   0   1   2     1
## 3    0   1 166   3   0   3   3   2   2   0   4   2     5
## 4    1   1   4 140   3   2   4   3   5   0   2   1     0
## 5    1   1   3   1 174   5   0   1   2   0   0   8     1
## 6    0   4   0   2   6 155   1   1   6   0   3   2     0
## 7    6   2   1   1   0   0 118   0   2   1   1   1     0
## 8    1   3   2   1   0   2   1 179   2   3   1   1     1
## 9    1   1   2   5   2   0   0   6 204   0   0   1     0
## 10   0   1   2   0   0   1   0   1   1  98   0   0     1
## 11   2   0   0   0   1   1   0   0   0   1 160   2     1
## 12   2    0    0    1    1    1    0    0    1   1    2 142      1
##    ------------------------------------------
```

In the table above, the rows indicate linkage groups obtained using linkage information and, the columns are the chromosomes in the reference genome. Notice the diagonal indicating the concordance between the two sources of information.

Ordering markers

Markers are ordered within linkage groups. In this tutorial, we will show the step-by-step procedure using Linkage Group 1 (LG1). You can do the same for the remaining linkage groups.

Since we had a good concordance between genome and linkage information, we will use only markers assigned to a particular linkage group using both sources of information. We will do that using `genomic.info = 1` , so the function uses the intersection of the markers assigned using linkage and the chromosome with the highest number of allocated markers. To use only the linkage information, do not use the argument `genomic.info`. You also need the recombination fraction matrix for that group.

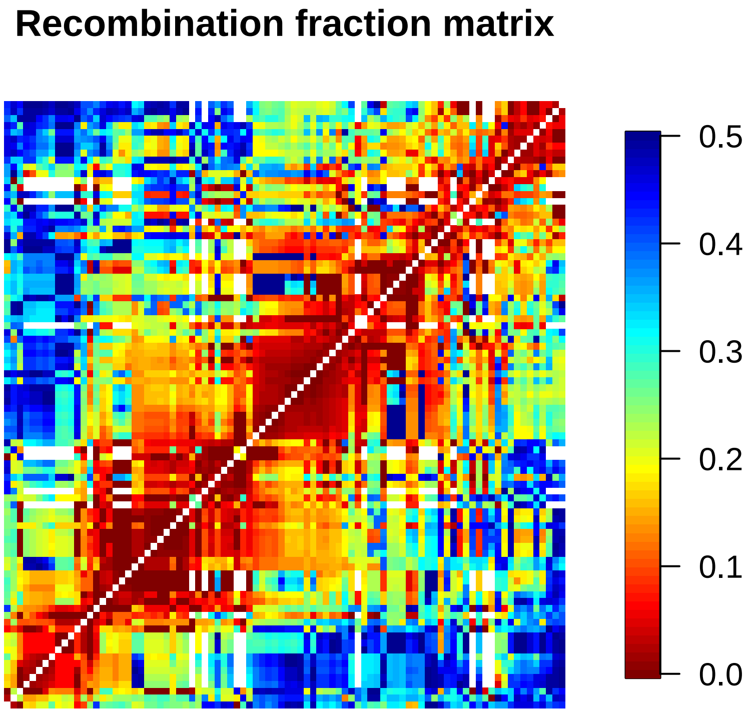
```
s1 <- make_seq_mappoly(g, 1, ## Select LG1
                       genomic.info = 1)
m1 <- make_mat_mappoly(m, s1)
```

Let us order the markers in the sequence using the MDS algorithm.

```
mds.o1 <- mds_mappoly(input.mat = m1)
```

```
## Stress: 0.30656
## Mean Nearest Neighbour Fit: 1.87647
```

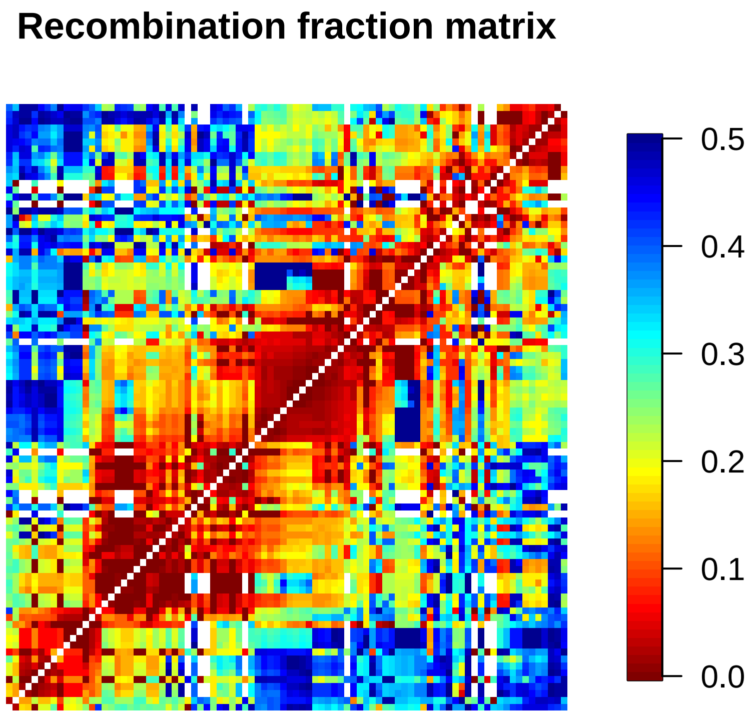
```
s1.mds <- make_seq_mappoly(mds.o1)
plot(m1, ord = s1.mds)
```



Usually, at this point, the user can use diagnostic plots to remove markers that disturb the ordering procedure. We didn't use that procedure in this tutorial, but we encourage the user to check the example in `?mds_mappoly`. Now, let us use the reference genome to order the markers.

You also can order the markers the reference genome

```r
gen.o1 <- get_genomic_order(s1)
s1.gen <- make_seq_mappoly(gen.o1)
plot(m1, ord = s1.gen)
```



For the sake of this short tutorial, let us use the MDS order (`s1.mds`) to phase the markers. Still, you can also try the genome order (`s1.gen`) and compare the resulting maps using function `compare_maps`, and you will notice that the genomic order yields a better map since its likelihood is higher.

Phasing markers and estimating multilocus recombination fractions.

Estimating the genetic map for a given order involves the computation of recombination fraction between adjacent markers and inferring the linkage phase configuration of those markers in both parents. The core function to perform these tasks in `MAPpoly` is `est_rf_hmm_sequential`. This function uses the pairwise recombination fraction as the first source of information to sequentially position allelic variants in specific parental homologs. The algorithm relies on the likelihood obtained through a hidden Markov model (HMM) for situations where pairwise analysis has limited power. Once all markers are positioned, the final map is reconstructed using the HMM multipoint algorithm. For a detailed description of the `est_rf_hmm_sequential` arguments, please refer to MAPpoly's Reference Manual and the Extended Tutorial.

First we need to calculate the pairwise recombination fraction for markers in sequence `s1.mds` using `est_pairwise_rf`, which contains the information necessary to the proper working of the phasing algorithm.

```r
tpt1 <- est_pairwise_rf(s1.mds, ncpus = ncores)
lg1.map <- est_rf_hmm_sequential(input.seq = s1.mds,
                                 start.set = 3,
                                 thres.twopt = 10,
                                 thres.hmm = 20,
                                 extend.tail = 50,
                                 info.tail = TRUE,
                                 twopt = tpt1,
                                 sub.map.size.diff.limit = 5,
                                 phase.number.limit = 20,
                                 reestimate.single.ph.configuration = TRUE,
                                 tol = 10e-3,
                                 tol.final = 10e-4)
```

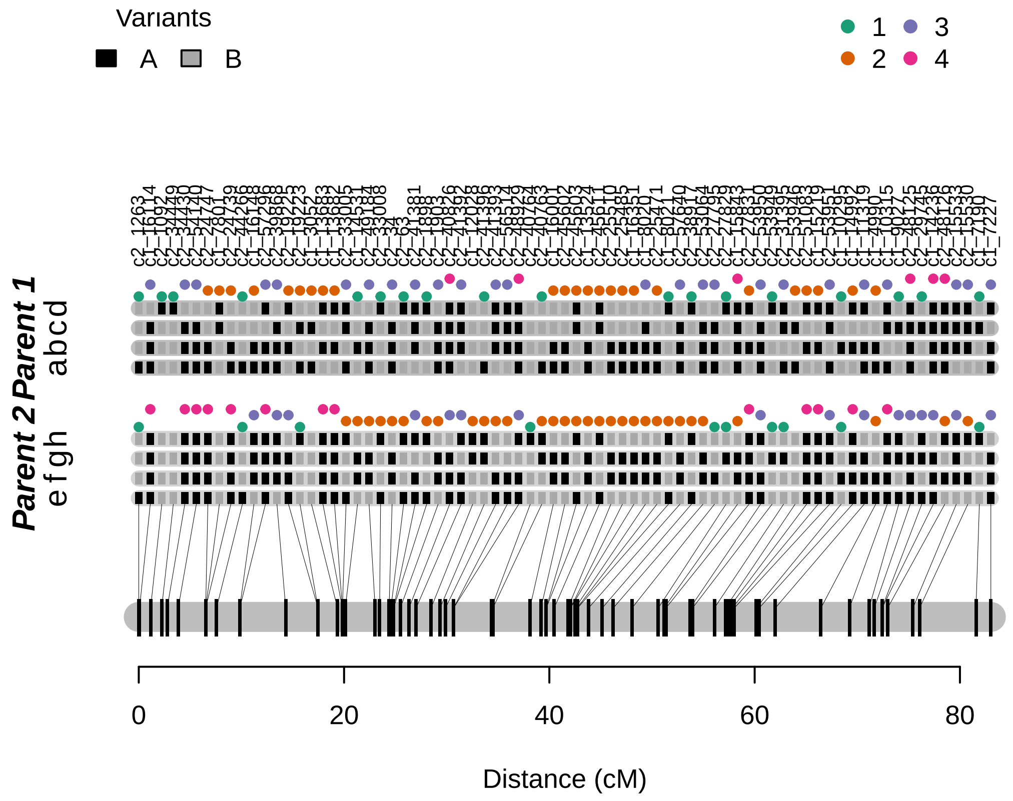
```
## ============================================================ Initial sequence ==

## ================================================== Done with initial sequence ==

## ================================== Reestimating final recombination fractions ==

## ================================================================================
```

Now, use the functions `print` and `plot` to view the map results:

```
print(lg1.map)
```

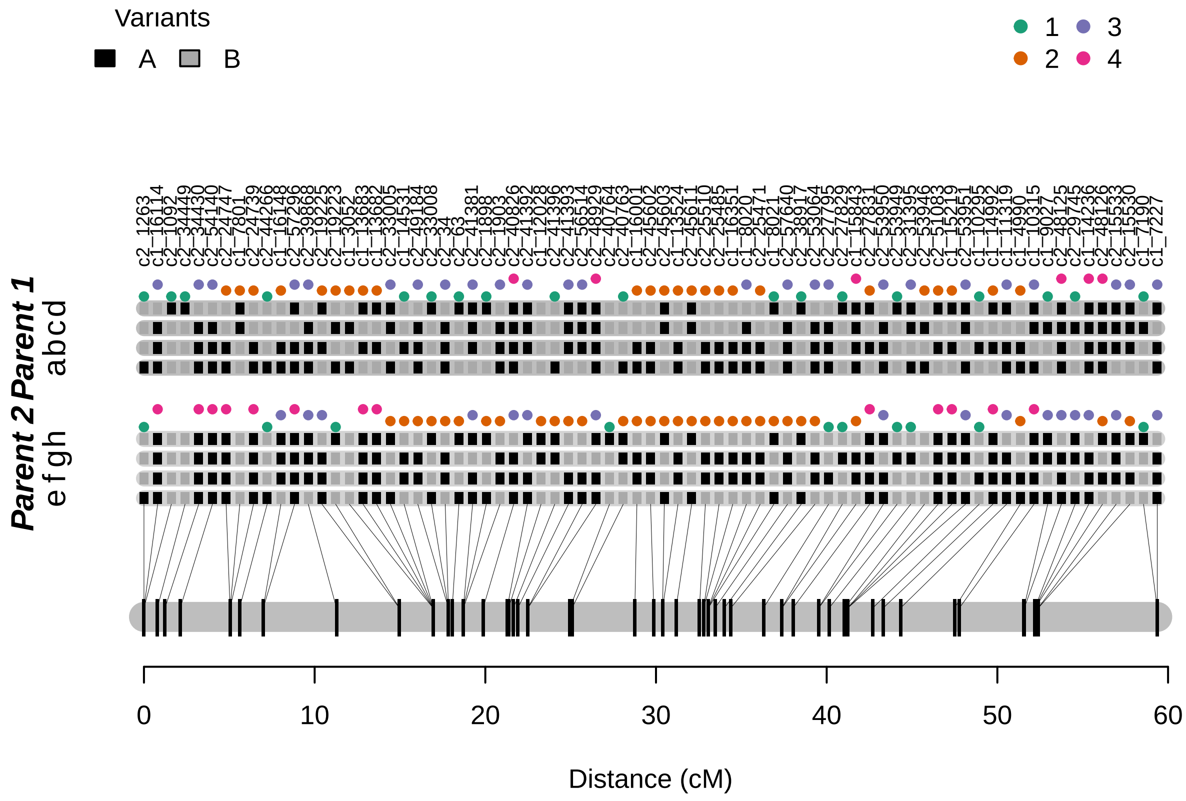
```
## This is an object of class 'mappoly.map'
##     Ploidy level:     4
##     No. individuals:  144
##     No. markers:  75
##     No. linkage phases:   1
##
##     ---------------------------------------------
##     Number of linkage phase configurations:  1
##     ---------------------------------------------
##     Linkage phase configuration:  1
##        map length:    83.01
##        log-likelihood:    -1015.28
##        LOD:       0
##     ~~~~~~~~~~~~~~~~~~
```

```
plot(lg1.map, mrk.names = TRUE, cex = 0.7)
```



Now let us update the recombination fractions by allowing a global error in the HMM recombination fraction re-estimation. Using this approach, the genetic map's length will be updated by removing spurious recombination events. This procedure can be applied using either the probability distribution provided by the genotype calling software using function `est_full_hmm_with_prior_prob` or assuming a global genotype error like the following example

```
lg1.map.up <- est_full_hmm_with_global_error(input.map = lg1.map, error = 0.05,
                                              verbose = TRUE)
plot(lg1.map.up, mrk.names = TRUE, cex = 0.7)
```

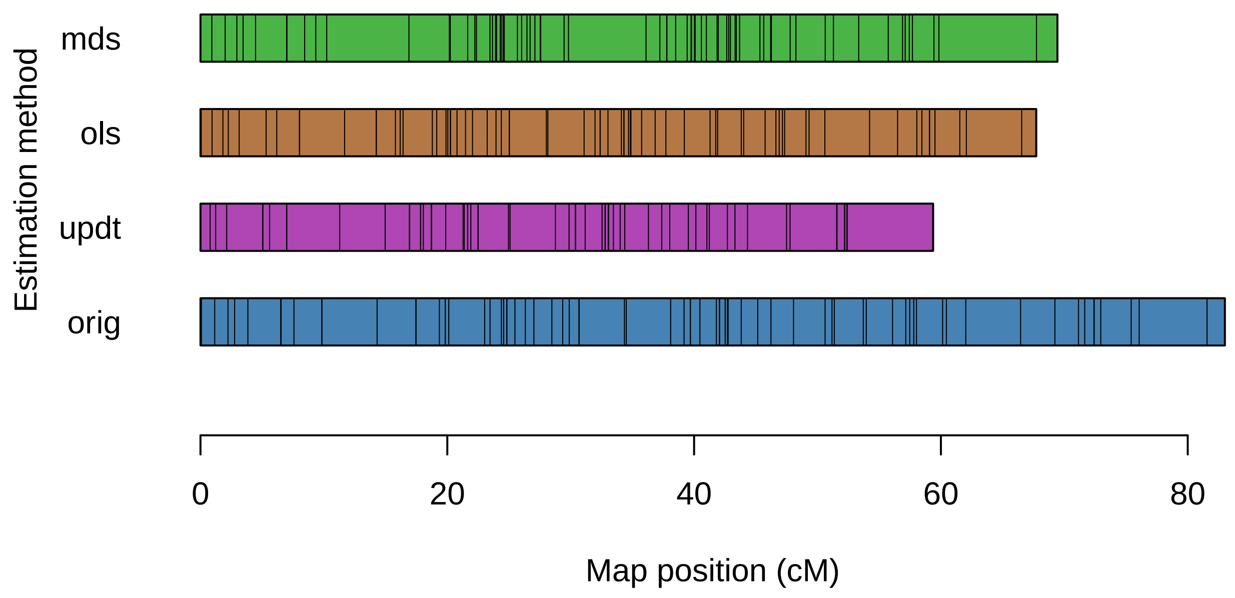


We can also use the ordinary least squares (OLS) method and the weighted MDS followed by fitting a one dimensional principal curve (wMDS_to_1D_pc)

```
lg1.map.ols <- reest_rf(lg1.map, m1, method = "ols")
lg1.map.mds <- reest_rf(lg1.map, m1, method = "wMDS_to_1D_pc", input.mds = mds.o1)
```

Now let us create a list with the maps and plot the results

```
map.list.lg1  <- list(orig = lg1.map,
                      updt = lg1.map.up,
                      ols = lg1.map.ols,
                      mds = lg1.map.mds)
plot_map_list(map.list.lg1, col = "ggstyle", title = "Estimation method")
```



## Homolog probability and preferential pairing profile

To use the genetic map in conjunction with QTL analysis software, we need to obtain the homolog probability for all linkage groups for all individuals in the full-sib population. In this short guide, we will proceed only with one linkage group, but this procedure should be applied to all chromosomes in real situations. Let us use the updated map

```
g1 <- calc_genoprob_error(lg1.map.up, step = 1, error = 0.05)
```

```
## Ploidy level:4
## Number of individuals:144
##  ................................................
##  ................................................
##  ..........................................
```

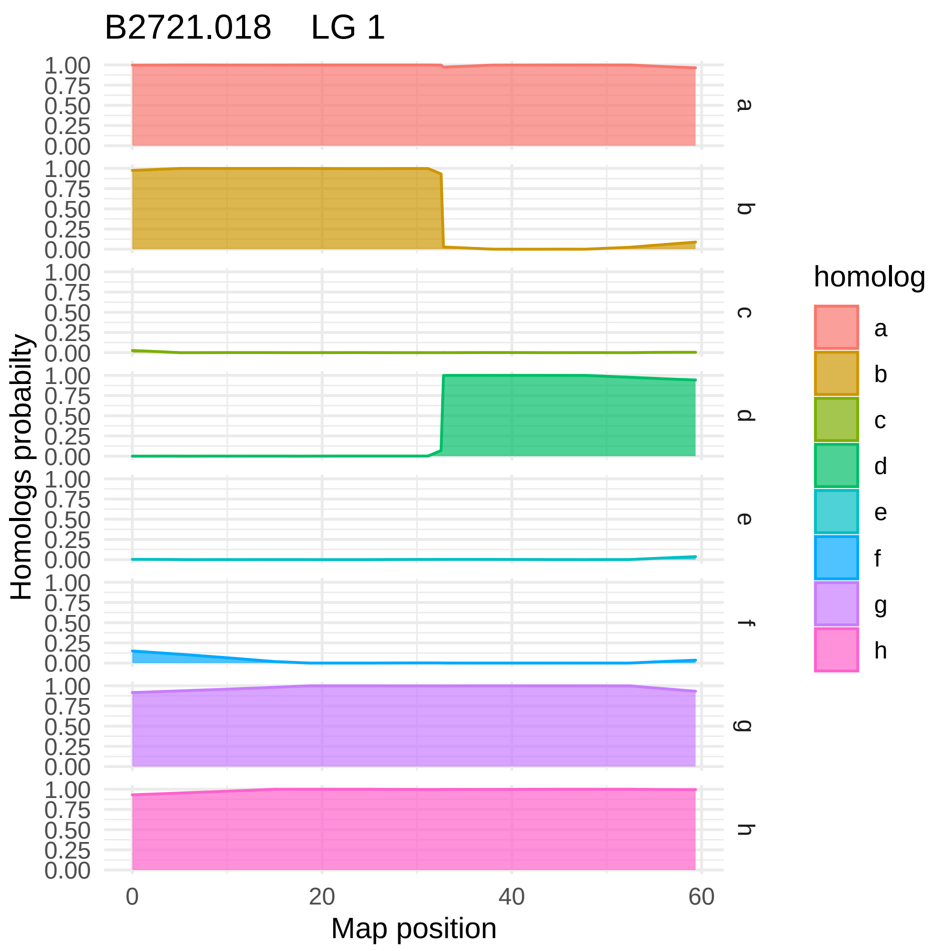
```
to.qtlpoly <- export_qtlpoly(g1) #export to QTLpoly
```

```
##
## Linkage group  1 ...
```

```
h1 <- calc_homologprob(g1)
```

```
##
## Linkage group  1 ...
```

```
plot(h1, lg = 1, ind = 10, use.plotly = FALSE)
```



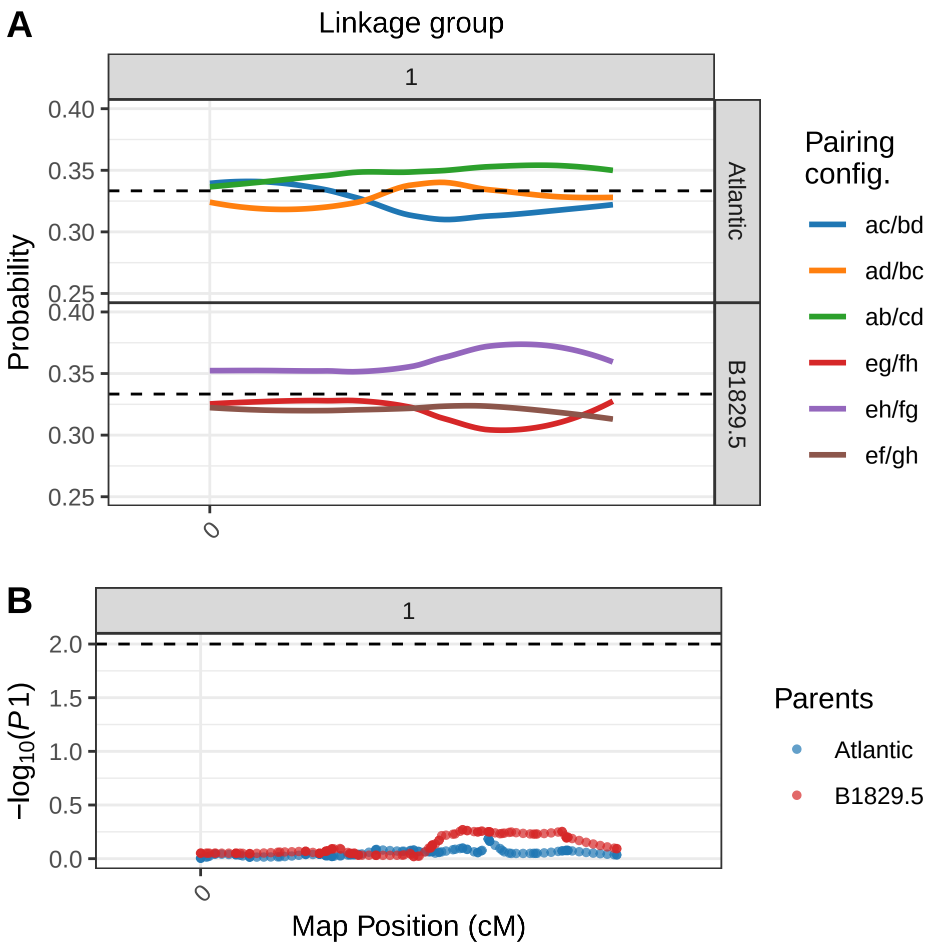
Now let us compute the preferential pairing profile for linkage group 1

```
p1 = calc_prefpair_profiles(g1)
```

```
##
## Linkage group  1 ...
```

```
plot(p1, min.y.prof = 0.25, max.y.prof = 0.4, P1 = "Atlantic", P2 = "B1829.5")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



Exporting a phased map

It is possible to export a phased map to an external CSV file using

```
export_map_list(lg1.map.up, file = "output_file.csv")
```

You can also print the same output on the screen using

```
export_map_list(lg1.map.up, file = "")
```

For a script with a complete analysis of the data set presented here, please refer to the Complete script

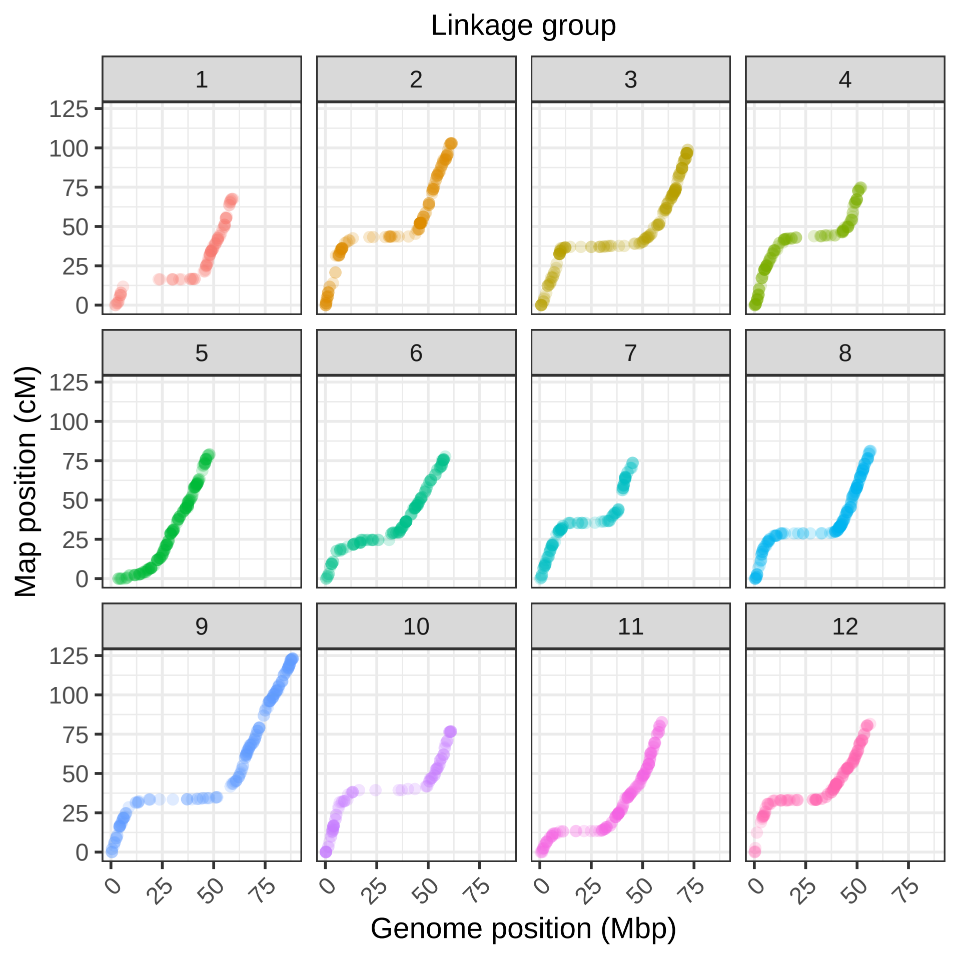
Utility functions

To use the next functions, let us load the complete genetic map

```
in.file <- "https://github.com/mmollina/SCRI/raw/main/docs/tetra/maps_updated.rda"
download.file(in.file,"map_file")
load("map_file")
```

Plot genome vs. map

```
plot_genome_vs_map(MAPs.up, same.ch.lg = TRUE)
```



Map summary

```
summary_maps(MAPs.up)
```

```
##
## Your dataset contains removed (redundant) markers. Once finished the maps, remember to add them back
```

```
##       LG Genomic sequence Map length (cM) Markers/cM Simplex Double-simplex
## 1      1               10           67.55       1.24      15             14
## 2      2                9           102.9       1.43      32             42
## 3      3                4           98.72       1.67      46             32
## 4      4                5           74.72       1.83      63             24
## 5      5                2           79.07        2.2      54             52
## 6      6                3            77.6       1.96      58              8
## 7      7               11            73.6        1.6      30             21
## 8      8                7           81.22       2.18      55             40
## 9      9                1          123.15       1.62      43             47
## 10    10               12           76.82       1.24      44             16
## 11    11                6           82.68       1.91      30             29
## 12    12                8           81.51       1.74      28             47
## 13 Total             <NA>         1019.54       1.72     498            372
##    Multiplex Total Max gap
## 1         55    84    7.43
## 2         73   147   10.78
## 3          87   165    6.17
## 4          50   137    5.42
## 5          68   174    5.36
## 6          86   152    6.86
## 7          67   118   12.19
## 8          82   177    4.17
## 9         109   199    7.35
## 10         35    95    5.32
## 11         99   158     5.4
## 12         67   142    9.83
## 13        878  1748    7.19
```