Analyzing polymapR's example using MAPpoly

Marcelo Mollinari

2018-10-11

Contents

Context

Recently, Bourke et al. (2018) released an R package for constructing genetic maps in species with ploidy level up to 6 called polymapR. The software uses pairwise recombination fractions coupled with the MDS algorithm (Preddy and Hackett, 2016) to assemble linkage groups, order, and phase genetic markers. The final genetic distance between markers is obtained using the projection of the MDS configuration onto a principal curve, as also proposed by Preddy and Hackett (2016). Although the method is fast and provides reliable results under certain ciscunstances (i.e., presence of informative markers), we noticed that, in order to avoid data with high levels of noise, the analytic procedure described in polymapR's tutorial removes a considerable portion of the markers. Here, we use our MAPpoly software to construct a genetic map using the same data set presented in polymapR's tutorial example and compare the results.

Overview of our strategy

We assembled the linkage groups (LGs) using the full recombination fraction matrix and standard UPGMA procedures. Within each linkage group, we also relied on MDS algorithm (Preddy and Hackett, 2016) implementation to order markers. For a given order, we applied our sequential algorithm (Mollinari and Garcia, 2018) which uses pairwise recombination fraction as the first source of information to sequentially

position the allelic variants in specific homologous chromosomes. For situations where pairwise analysis has limited power, it relies on the likelihood obtained through a hidden Markov model (HMM). Once all markers were positioned, the final map was reconstructed using the HMM multilocus algorithm. Thus, differently from `polymapR`, our method does not rely on single dose markers to build a homologous framework.

Using our procedure we were able to increase the number of phased markers in the final map by about 2.56 fold. Also, the recombination fractions were multilocus estimated (i.e., using the maximum likelihood of the HMM). Finally, we included a global genotype error rate of 0.05 in the HMM emission function to reestimate the final map. In real situations, this error could be marker/offspring specific and is obtained using several genotype-calling methods available, such as fitTetra, SuperMASSA and updog. The final length of the maps was somewhat different due to the different estimation procedures: `MAPpoly` yield a longer map when compared to `polymapR` (MAPpoly: 101.76 cM/lg and polymapR: 90.2 cM/lg, in average). We compared our phasing results with `polymapR`'s result and, for the markers contained in both maps, the results were precisely the same for all linkage groups. However, since our method does not rely on single dose markers to perform the phasing, we could include 2514 markers in the final map (from a total of 2873 markers, 256 % more than `polymapR` result, which included 982 markers). In terms of performance, since our program use multilocus analisys and model genotype errors, it takes about 30 minutes to automaticaly build the map. On the other hand, `polymapR` uses only two point analysis, making the recombination fraction computation faster. Thus, using the script from polymapR's tutorial, the map construction takes less than one minute.

Conclusion

Depending on the amount of information available in the data set, `polymapR` can produce adequate results. However, a high level of "noise-to-signal" ratio is expected in polyploid genotypic data sets. `MAPpoly`'s approach extract more information from the data set and is capable of modeling genotype errors due to its multilocus nature. Moreover, it does not depend on single dose markers to provide a phasing framework. It is worthwhile to mention that the construction of complex polyploid maps could be challenging if the adequate computational power is not available. In our experience, we were able to construct an ultra-saturated map of the hexaploid sweetpotato containing more than 38,000 SNPs, where 55% where simplex and 45% multiplex.

Map construction

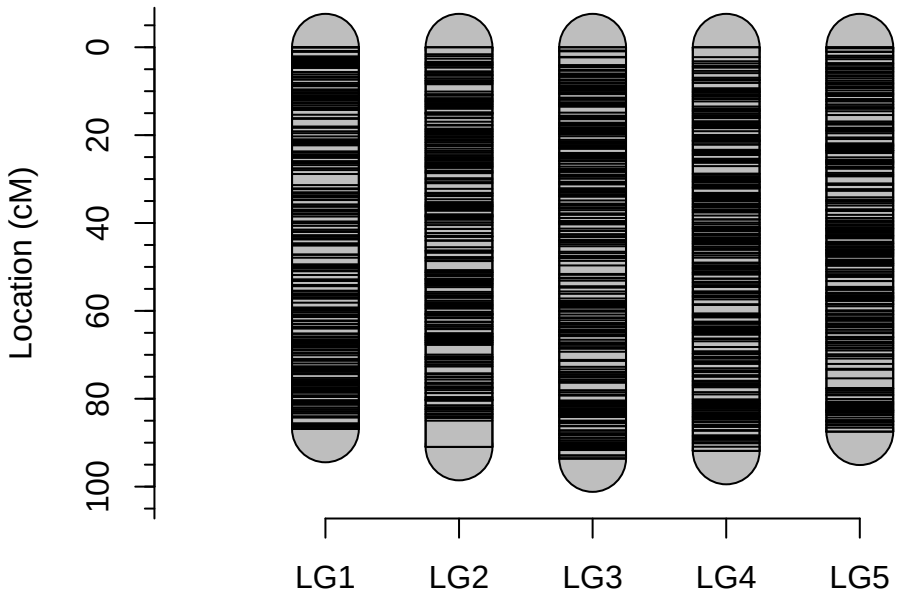
Loading `polymapR` results

There are two map results in `polymapR`'s tutorial: the first one is called "integrated map"", which contains markers from both parents, but not totaly phased. The second one is the"phased map", with less but phased markers.

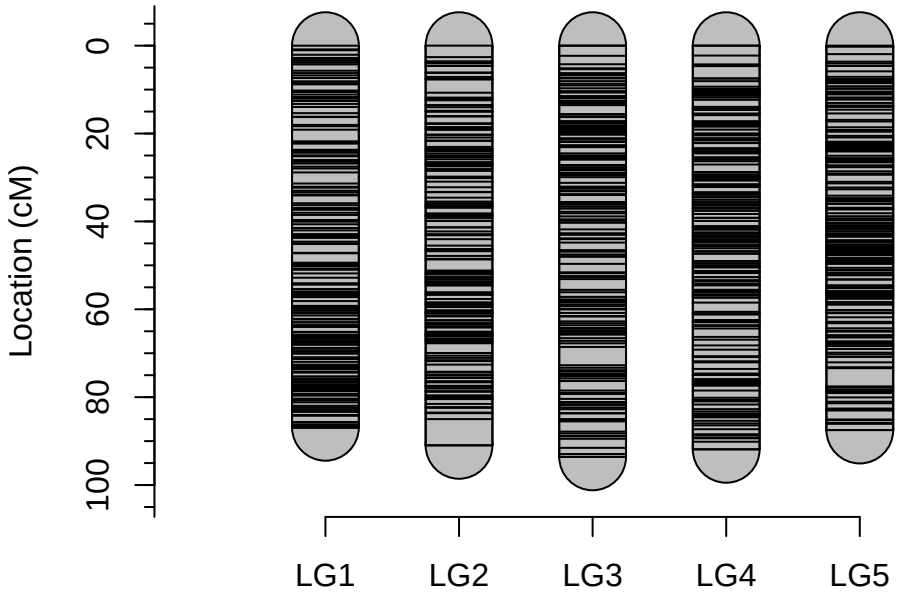
```
require(polymapR)
```

```
## Loading required package: polymapR
```

```
load("~/repos/tutorials/polymapR_example/polymapR_exemple.rda")
plot_map(maplist = integrated.maplist)
```




```
plot_map(maplist = phased.maplist)
```



Summary for both maps:

The number of markers, length and markers density are presented bellow:

```
polymapR.integrated.maps
```

```
##              LG1    LG2    LG3    LG4   LG5
## n.markers 273.00 268.00 284.00 296.00 287.0
## length     86.90  91.00  93.60  91.90  87.5
## cM/mrk      0.32   0.34   0.33   0.31   0.3
```

```
polymapR.phased.maps
```

```
##              LG1    LG2    LG3    LG4   LG5
```

```
## n.markers 201.00 178.00 180.00 206.00 217.0
## length     86.90  91.00  93.60  91.90  87.5
## cM/mrk      0.43   0.51   0.52   0.45   0.4
```

Map construction using `MAPPoly`

This is a very breif tutorial. For a detailed procedure to build a map using `MAPPoly`, please refer to this tutorial.

Loading and converting data

Let us load `MAPPoly` and convert the data set used in `polymapR` tutorial. We will use the `"segregating_data"` data set which was already filtered for no-segregating markers. Also, all the segregation types were converted in their simplest form. This is not a requirement for `MAPPoly` but we are using the same data set to compare results

```
require(mappoly)
source("~/repos/tutorials/polymapR_example/polymapR_to_mappoly.R")
data("segregating_data")
```

Summary of both data sets:

```
tetra.data<-polymapR_to_mappoly(segregating_data)
```

```
##
## ####Marker dosage frequencies:
##
##         P2_0   P2_1   P2_2   P2_3
## -----  -----  -----  -----  -----
## P1_0       0    366    230      0
## P1_1     370    630    386    132
## P1_2     232    352    175      0
##
## markers not converted: 2873
##
## markers 1 parent converted: 0
##
## markers 2 parents converted: 0
##
## non-segregating markers deleted: 0
```

```
print(tetra.data, detailed = TRUE)
```

```
## This is an object of class 'mappoly.data'
##     Ploidy level:    4
##     No. individuals:    207
##     No. markers:        2873
```

```
##
##     No. markers per sequence: not available
##     ----------
##     No. of markers per dosage in both parents:
##     dP dQ freq
##      0  1  366
##      0  2  230
##      1  0  370
##      1  1  630
##      1  2  386
##      1  3  132
##      2  0  232
##      2  1  352
##      2  2  175
```

Recombination fraction estimation

Now, let us compute the recombination fraction between all markers in the data set.

```
s<-make_seq_mappoly(tetra.data, arg = "all")
counts.web<-cache_counts_twopt(s, get.from.web = TRUE)
```

```
## Internet conectivety ok.
## Loading genotype counts from web
```

```
all.pairs<-est_pairwise_rf(input.seq = s,
                           count.cache = counts.web,
                           n.clusters = 16, #toke approx. 9 minutes using 24 CPUs in BRC cluster
                           verbose=TRUE)
```

```
## INFO: Using  16  CPUs for calculation.
## INFO: Done with 4125628  pairs of markers
## INFO: Calculation took: 941.294 seconds
```

```
mat.full<-rf_list_to_matrix(input.twopt=all.pairs)
```

```
## INFO: Going singlemode. Using one CPU.
```

```
mat.full
```

```
##   This is an object of class 'mappoly.rf.matrix'
##
##   Criteria used to filter markers:
##
##       Configuration phase LOD:            0
##       Recombination fraction LOD:         0
##       Maximum recombination fraction:     0.5
##
```

```
##   No. markers:              2873
##   Percentage filled:        90.2 %
```

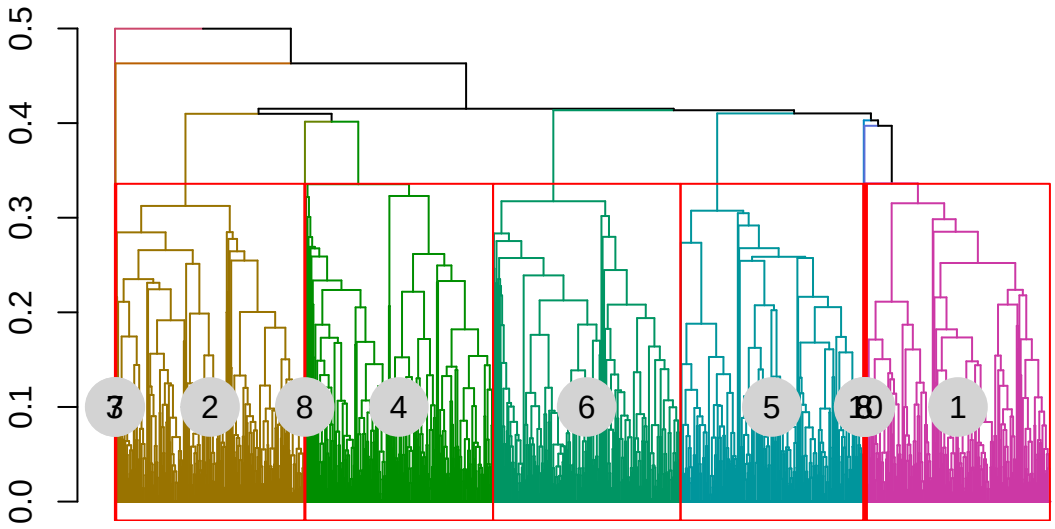
Grouping

Assigning markers to linkage groups is an interactive task in MAPpoly. Since we expect five linkage groups, we tried to use 5 as the expected number. However, due to some markers that were not clearly positioned in any group, we divided the resulting dendrogram into a larger number of groups and picked the densest five clusters.

```
lgs<-group_mappoly(input.mat = mat.full,
                   input.seq = s,
                   expected.groups = 11,
                   comp.mat = FALSE,
                   inter = FALSE)
lgs
```

```
##   This is an object of class 'mappoly.group'
##   -------------------------------------------
##   Criteria used to assign markers to groups:
##
##     - Number of markers =         2873
##     - Number of linkage groups = 11
##     - Number of markers per linkage groups:
##     group n.mrk
##         1   563
##         2   579
##         3     1
##         4   577
##         5   562
##         6   576
##         7     3
##         8     2
##         9     2
##        10     6
##        11     2
##   -------------------------------------------
```

```
plot(lgs)
```




```
## Selecting groups
LGS<-lapply(c(1,2,4,5,6), function(x, lgs) make_seq_mappoly(lgs, x), lgs)
```

Filtering and ordering markers within groups

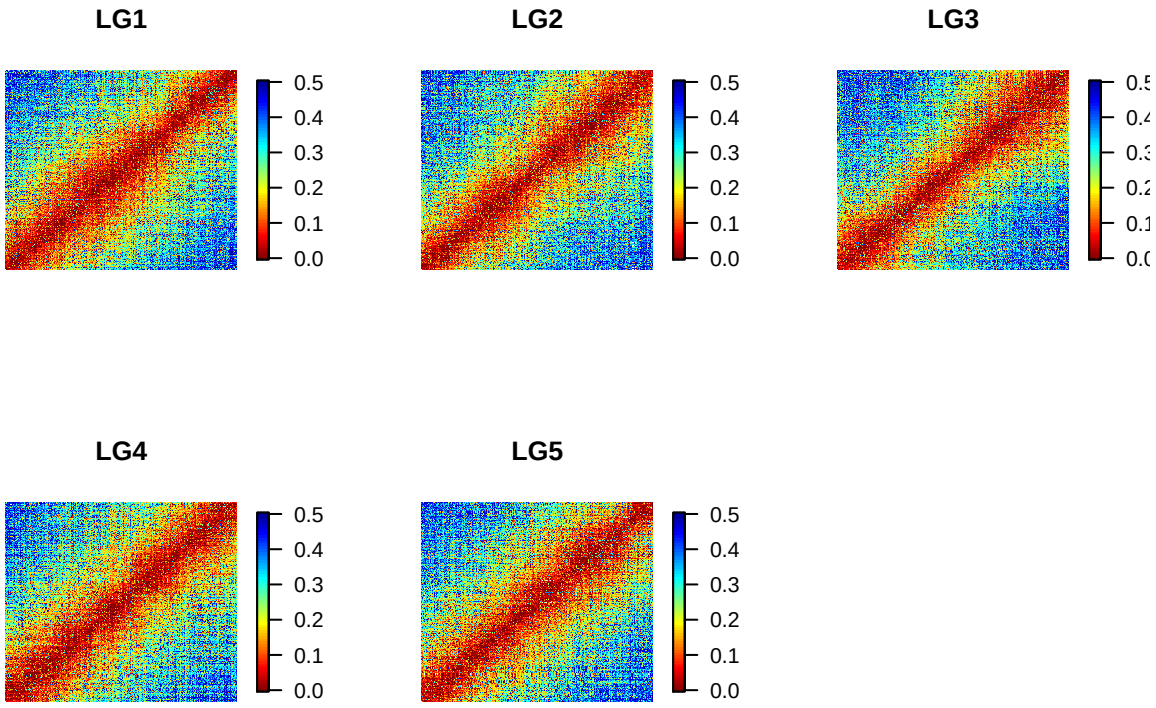
Next, we use the function `rf_snp_filter` to remove markers that do not meet a LOD and recombination fraction criteria for at least 10% of the pairwise markers combinations within each linkage group. After that, we obtain the recombination fraction matrices for all groups and use the MDS algorithm to order the markers.

```
## INFO: Going singlemode. Using one CPU.
## INFO: Going singlemode. Using one CPU.
## INFO: Going singlemode. Using one CPU.
## INFO: Going singlemode. Using one CPU.
## INFO: Going singlemode. Using one CPU.
```

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
```

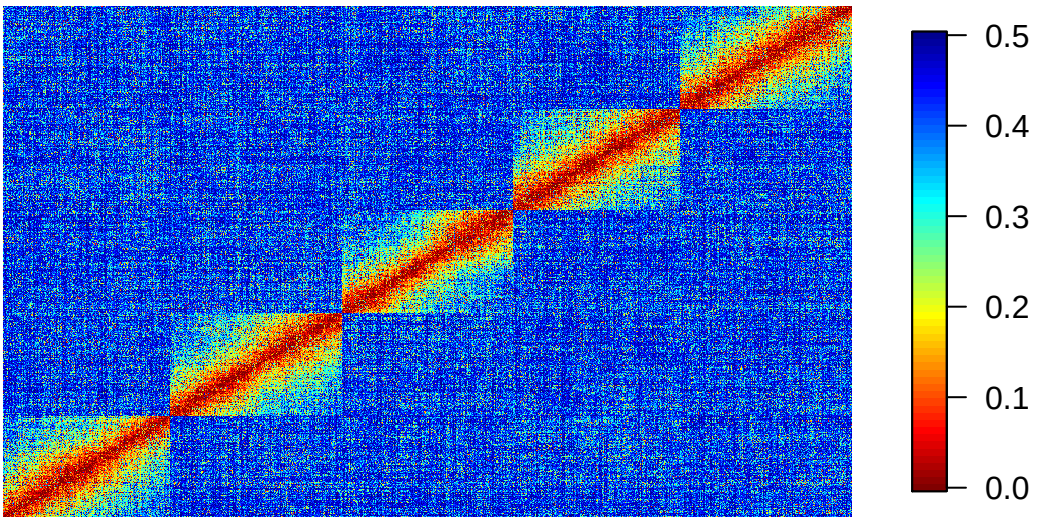
Now, let us plot the reordered recombination fraction

```
op <- par(mfrow = c(2, 3), pty = "s")
for(i in 1:5)
  plot(M.filt[[i]], ord = MDS.seq[[i]]$seq.mrk.names,
       main.text = paste0("LG", i), index = FALSE)
par(op)
```



Now, the full recombination fraction matrix

```
o<-unlist(sapply(MDS.seq, function(x) x$seq.mrk.names))
plot(mat.full, ord = o)
```



Phasing and multilocus reconstruction

Given the MDS order, we estimate the phase and the recombination fraction between all adjacent markers. After that, we reestimate the map using the function `est_full_hmm_with_global_error`, which includes a global genotype error in the HMM model.

```
## function for parallel computation
my.phase.func<-function(X)
{
  return(est_rf_hmm_sequential(input.seq = X[[1]],
                               thres.twopt = 10,
                               thres.hmm = 10,
                               extend.tail = 20,
                               twopt = X[[2]],
                               verbose = TRUE,
                               tol = 10e-3,
                               tol.final = 10e-4,
                               phase.number.limit = 20,
                               sub.map.size.diff.limit = 4,
                               info.tail = TRUE,
                               reestimate.single.ph.configuration = TRUE,
                               high.prec = FALSE))

}
## list with ordered sequences and pairwise recombination fractions
X<-vector("list", 5)
for(i in 1:5)
  X[[i]]<-list(MDS.seq[[i]], P.filt[[i]])

## Running phasing algorithm in 5 CPUs
system.time({
  cl <- parallel::makeCluster(5)
  parallel::clusterEvalQ(cl, require(mappoly))
  parallel::clusterExport(cl, "tetra.data")
  MAPS <- parallel::parLapply(cl,X,my.phase.func)
  MAPs.res <- parallel::parLapply(cl, MAPS, est_full_hmm_with_global_error,
                                  error = 0.05, tol = 10e-4)
  parallel::stopCluster(cl)
})
```

```
##    user  system elapsed
##   1.152   0.214 439.125
```

Comapring maps

Now, let us compare the results from both programs.

```
id<-c(4,3,1,2,5) ##This is to match the linkage group ordering in polymapR tutorial
nm<-pal<-comp.PHQ<-comp.PHP<-vector("list", 5)
for(i in 1:5)
{
  YP<-phased.maplist[[id[i]]][,3:6]
  rownames(YP)<-phased.maplist[[id[i]]][,1]
  YQ<-phased.maplist[[id[i]]][,7:10]
  rownames(YQ)<-phased.maplist[[id[i]]][,1]
  mrk.names<-tetra.data$mrk.names[MAPS[[i]]$maps[[1]]$seq.num]
  nm[[i]]<-intersect(rownames(YQ), mrk.names)
  AP<-MAPS[[i]]$maps[[1]]$seq.ph$P
  AQ<-MAPS[[i]]$maps[[1]]$seq.ph$Q
  names(AQ)<-names(AP)<-mrk.names
  BP<-ph_matrix_to_list(YP)
  names(BP)<-rownames(YP)
  BQ<-ph_matrix_to_list(YQ)
  names(BQ)<-rownames(YQ)
  comp.PHP[[i]]<-compare_haplotypes(m = 4, h1 = AP[nm[[i]]], h2 = BP[nm[[i]]])
  comp.PHQ[[i]]<-compare_haplotypes(m = 4, h1 = AQ[nm[[i]]], h2 = BQ[nm[[i]]])
  pal[[i]]<-mrk.names%in%rownames(YQ)+1
}

res<-rbind(sapply(phased.maplist, nrow),
           sapply(MAPS, function(x) x$info$n.mrk),
           sapply(phased.maplist, function(x) round(max(x$position,1))),
           sapply(MAPs.res, function(x) round(sum(imf_h(x$maps[[1]]$seq.rf)),1)),
           sapply(nm, length),
           sapply(comp.PHP, function(x) x$is.same.haplo),
           sapply(comp.PHQ, function(x) x$is.same.haplo))
dimnames(res)<-list(c("n.mrk.polymapR",  "n.mrk.MAPPoly" , "polymapR.length", "MAPPoly.length","intersec
```

The columns in the following table present the number of markers assigned to each LG (rows) using `polymapR` and `MAPpoly`, respectively, followed by the length of the resulting maps, the number of markers shared between the maps and whether the estimated linkage phase was the same within the shared markers (1) or not (0).

```
t(res)
```

```
##     n.mrk.polymapR n.mrk.MAPPoly polymapR.length MAPPoly.length intersect
## LG1            201           500              87          103.2       186
## LG2            178           512              91          103.9       166
## LG3            180           496              94          101.8       176
## LG4            206           497              92           97.2       164
## LG5            217           509              87          102.7       202
##     is.same.phase.P1 is.same.phase.P2
## LG1                1                1
```

```
## LG2                1                1
## LG3                1                1
## LG4                1                1
## LG5                1                1
```

Next, we compare the likelihoods of the maps produced by both programs using only the shared markers. Since the phase is the same in both cases, here we are evaluating the order obtained using MDS. Notice that, the multilocus likelihood is available only in `MAPpoly`. Thus we reestimated the maps produced by `polymapR` with our HMM procedure.

```
a<-b<-P.MAPs.temp<-M.MAPs.temp<-vector("list", 5)
for(i in 1:5)
{
  a[[i]]<-integrated.maplist[[id[i]]][,1:2]
  b[[i]]<-data.frame(marker = tetra.data$mrk.names[MAPS[[i]]$maps[[1]]$seq.num],
                     position = cumsum(mappoly::imf_h(c(0, MAPS[[i]]$maps[[1]]$seq.rf))))
  s.compP<-mappoly::make_seq_mappoly(tetra.data, arg = intersect(a[[i]]$marker, b[[i]]$marker))
  s.compM<-mappoly::make_seq_mappoly(tetra.data, arg = intersect(b[[i]]$marker, a[[i]]$marker))
  oP<-match(s.compP$seq.num, MAPs.res[[i]]$maps[[1]]$seq.num)
  oM<-match(s.compM$seq.num, MAPs.res[[i]]$maps[[1]]$seq.num)
  P.MAPs.temp[[i]]<-mappoly::get_submap(MAPs.res[[i]], mrk.pos = oP,
                                        reestimate.rf = TRUE,
                                        tol.final = 10e-2, verbose = FALSE)
  M.MAPs.temp[[i]]<-mappoly::get_submap(MAPs.res[[i]], mrk.pos = oM,
                                        reestimate.rf = TRUE,
                                        tol.final = 10e-2, verbose = FALSE)
}
```

Except from LG 2, in the order obtained using `MAPpoly` was superior to the order obtained using `polymapR`.

```
## [[1]]
## [1] TRUE
##
## [[2]]
## [1] TRUE
##
## [[3]]
## [1] TRUE
##
## [[4]]
## [1] TRUE
##
## [[5]]
## [1] TRUE
```

```
##       polymapR  MAPPoly
## [1,]  76.09846   0.0000
```

```
## [2,]   0.00000 218.3207
## [3,]  28.90145   0.0000
## [4,] 208.45842   0.0000
## [5,]  74.74032   0.0000
```

Now, let us compare both maps. Maps to the left were produced by `polymapR`; maps to the left were produced by `MAPpoly`. Shared markers (red dots) are linked by black lines. Private markers are represented by blue dots.