Constructing genetic linkage map

To construct a linkage map based on a ourtcrossing F1 population, we can use onemap and lep-map3.

https://statgen-esalq.github.io/tutorials/onemap/Outcrossing_Populations.html#content

https://sourceforge.net/projects/lep-map3/

I have tried onemap. Its map construction step was super slow and after one day I got an error telling me that some markers in the group are not linked.

At the same time, I tried lep-map3, which worked lilke a charm. So here I wrote down how I worked with lep-map3.

First, install the pacakge

Just download the source package from the link above and copy it to a new folder in your own directory on our P drive

then unzip the package

unzip binary+code.zip

Install finished!

Next, make the input file for lep-map3. It requires one pedigree.txt file and a vcf file

I used R to make the pedigree.txt file, so in R, write below codes:

```
#import sample names. The name list was done by last post (processing GBS data)

sample_name <- read.table("sample_name.txt", header = F, stringsAsFactors = F)

# remove 305-309 rows, 305-309 are two Sikem and three Rabiosa parents.
myname <- sample_name[-c(305:309), ]

# make the pedigree file for lep-map3
myline1 <- c("CHR", "POS", rep("SR", 306))
myline2 <- c("CHR", "POS", "PRabiosa", "PSikem", myname[1:304])
myline3 <- c("CHR", "POS", "0", "0", rep("PSikem", 304))
myline4 <- c("CHR", "POS", "0", "0", rep("PRabiosa", 304))
myline5 <- c("CHR", "POS", "2", "1", rep("0", 304))
myline6 <- c("CHR", "POS", rep("0", 306))

mypedigree <- rbind.data.frame(myline1, myline2, myline3, myline4, myline5, myline6)

write.table(mypedigree, "pedigree.txt", quote = F, sep = "\t", row.names = F, col.names = F)
```

So, as you can see, the pedigree.txt file is a 6-line tab-delimited file. The first line is family name, here it is SR, meaning Sikem and Rabiosa.

The second line is individual names, which are the sample names. Here, notice that, two parent individuals are put at the first two places then follows progeny, but in my vcf file, the last two columns are two parents. So, in order to match the sample order, I need to modify my vcf file.

Line 3 and Line 4, just show the mother and father of each progeny.

Line 5 shows the sex, male 1, femal 2, unknown 0.

Line 6, phenotypes, if none, just enter 0.

See lep-map3 wiki page: https://sourceforge.net/p/lep-map3/wiki/LM3%20Home/#parentcall2

Then, modify my vcf file as I mentioned before.

On The Dude, activate my samtools environment

conda activate samtools

Use bcftools to modifiy the vcf file. The idea is to subset the two parents and all progeny from the vcf file and store them in two independent vcf files. Then merge these two vcf files but with the parent vcf at the first place.

 # reshape the vcf file. put parents to the first two columns
bcftools view -s PRabiosa,PSikem tmp_vcf/rabiosa_ps_pr_renamed.filtered.vcf > tmp_vcf/rabiosa_ps_pr_renamed.filtered_parent.vcf
bcftools view -s ^PRabiosa,PSikem tmp_vcf/rabiosa_ps_pr_renamed.filtered.vcf > tmp_vcf/rabiosa_ps_pr_renamed.filtered_progeny.vcf
bgzip -c tmp_vcf/rabiosa_ps_pr_renamed.filtered_parent.vcf > tmp_vcf/rabiosa_ps_pr_renamed.filtered_parent.vcf.gz
bgzip -c tmp_vcf/rabiosa_ps_pr_renamed.filtered_progeny.vcf > tmp_vcf/rabiosa_ps_pr_renamed.filtered_progeny.vcf.gz
bcftools index --threads 10 tmp_vcf/rabiosa_ps_pr_renamed.filtered_parent.vcf.gz
bcftools index --threads 10 tmp_vcf/rabiosa_ps_pr_renamed.filtered_progeny.vcf.gz
bcftools merge tmp_vcf/rabiosa_ps_pr_renamed.filtered_parent.vcf.gz tmp_vcf/rabiosa_ps_pr_renamed.filtered_progeny.vcf.gz > tmp_vcf
/rabiosa_ps_pr_renamed.filtered_lepmap3.vcf

Finally, I have both pedigree.txt and rabiosa_ps_pr_renamed.filtered_lepmap3.vcf ready for lep-map3. Now, start lep-map3. See the pipeline below.
Source from: https://academic.oup.com/bioinformatics/article/33/23/3726/4061277

blocked URL

Below is my codes for running lep-map3 and I learnt a lot from this tutorial, https://avikarn.com/2019-04-17-Genetic-Mapping-in-Lep-MAP3/#-installing-lep-map3-

Also, the wiki page of lep-map3 is a good source of instruction.

#!/bin/bash

 # 09.08.2021

 # using lep-map3 to construct linkage map

mypath=/home/yutachen/public/Yutangchen/lepmap3/bin # store the path of where I have lepmap3 installed in a variable

 # import data
java -cp $mypath ParentCall2 data = pedigree.txt vcfFile = rabiosa_ps_pr_renamed.filtered_lepmap3.vcf > p.call # java cp path module options, this is the pattern how you use lep-map3

 # filter data
java -cp $mypath Filtering2 data=p.call removeNonInformative=1 dataTolerance=0.000001 missingLimit=60 > p_fil.call # removeNonInformative removes none informative markers (homologous for both parents), dataTolerance removes markers with distorted segregating pattern, set the value a little bit lower to discard less markers, missingLimit, value greater than 1 means discard markers having more then value individuals genotype not called (missing genotypes)

 # assign markers to chromosomes
java -cp $mypath SeparateChromosomes2 data=p_fil.call lodLimit=20 theta=0.2 numThreads=5 sizeLimit= 200 > map.txt # always play with the lodLimit (LOD of two-point test) and theta (recombination fraction) to get the ideal number of groups
sort map.txt|uniq -c|sort -n

 # merge single markers
java -cp $mypath JoinSingles2All map=map.txt data=p_fil.call lodLimit=3 theta=0.2 > map_js.txt # some markers were not assigned to any group, so use this module to try to reassign single markers to groups, also play with lod and theta to get the best results you think.
sort map_js.txt|uniq -c|sort -n
cut -f 1 map_js.txt | sort | uniq -c | sort -n

 # order markers in each group
java -cp $mypath OrderMarkers2 data=p_fil.call map=map_js.txt outputPhasedData=1 numThreads=20 useKosambi=1 > order.txt # order markers in each group

 # extract the row number of each SNP in the p.call file
cut -f 1,2 p.call | awk '(NR>=7)' > snps.txt
 # match marker name with contig and coordinate
awk -vFS="\t" -vOFS="\t" '(NR==FNR){s[NR-1]=$0}(NR!=FNR){if ($1 in s) $1=s[$1];print}' snps.txt order.txt > order.mapped # I have to admit that these codes were copied from lep-map3's wiki page (I hate awk but I have to learn and use it)

 # build map for each parent

java -cp $mypath OrderMarkers2 data=p_fil.call map=map.txt informativeMask=1 numThreads=20 useKosambi=1 > Sikem_order.txt
java -cp $mypath OrderMarkers2 data=p_fil.call map=map.txt informativeMask=2 numThreads=20 useKosambi=1 > Rabiosa_order.txt

There are 8880 markers imported to lep-map3, after filtering, 3452 markers were left. After assigning markers to chromosomes and reassigning, 7 large groups were identified. There are still 17 single markers. Markers numbers for each group, see below. In total, 3435 markers were assigned to 7 groups.

17 0

367 7
390 6
428 5
494 4
549 3
562 2
645 1

I upgraded my codes, now I construct one map for one group separately in parallel and also output the interval file:

seq 1 7 | parallel -j 7 "java -cp $mypath OrderMarkers2 data=p_fil.call map=map_js.txt outputPhasedData=1 numThreads=5 useKosambi=1 chromosome={} calculateIntervals=order{}.int > order_{}.txt" ***# this is ultra fast, all maps finish within 0.5 hour***

Then I concatenated all maps and convert the phase information in the file to genotypes

cat order_?.txt > order_x.txt

awk -vfullData=1 -f map2genotypes.awk order_x.txt > genotypes_x.txt

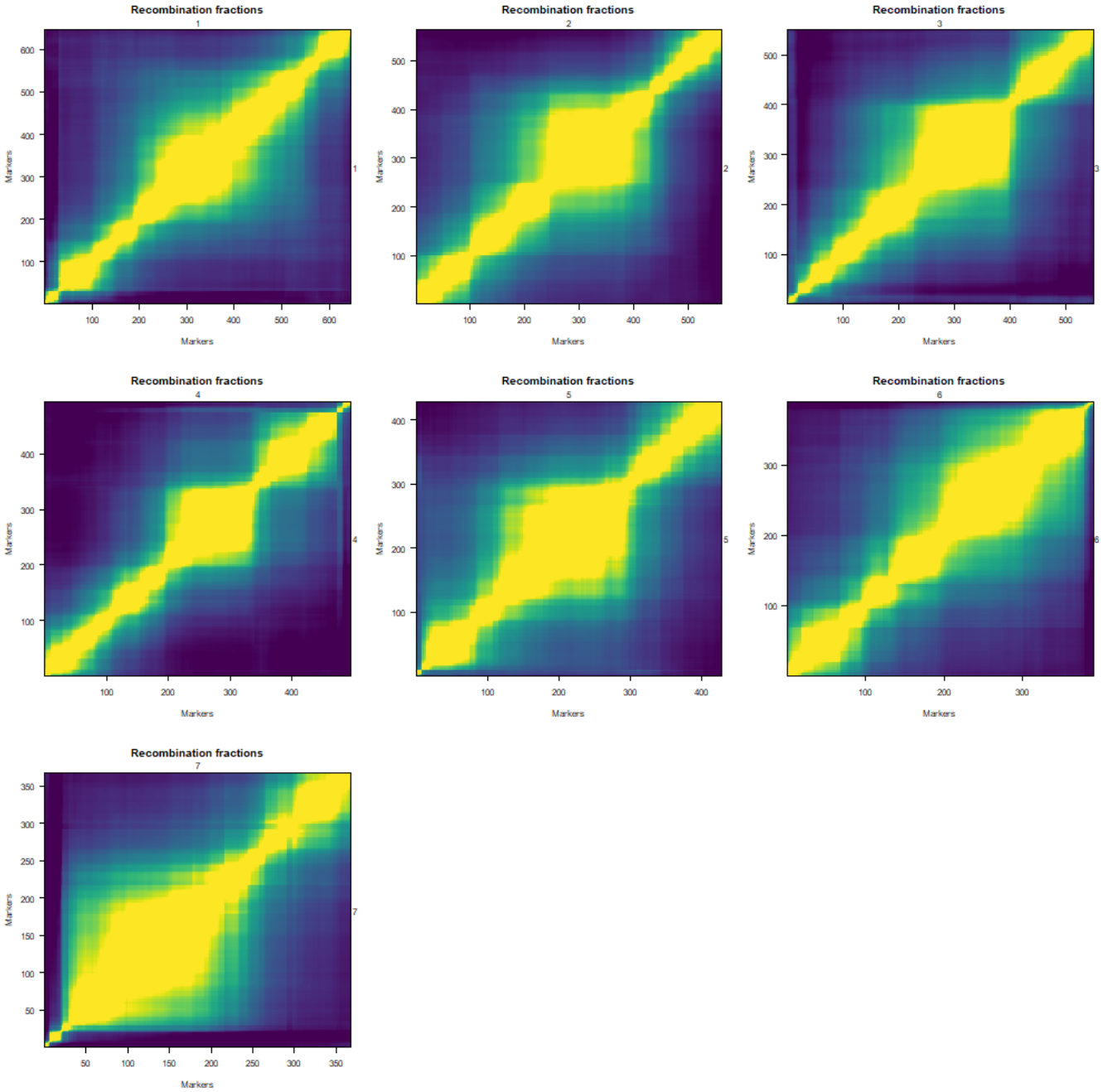
Also, extract marker name and their position from the concatenated file

grep -v "#" order_x.txt | cut -f 1,2 > LGs.txt

Now import genotypes_x.txt and LGs.txt to R to make the csvr format for Rqtl and then plot the recombination graph to verify maps. My R scripts below,

```
# use Rqtl rfplot

# install Rqtl
install.packages("qtl")
# load Rqtl
library(qtl)

# set working directory
setwd("C:/Users/yutachen/Desktop/R/vcf")

# import the genotype data from lepmap3
mytest <- read.table("genotypes_x.txt", header = F, sep = "\t", stringsAsFactors = F)

# import LGs information
LGs <- read.table("LGs.txt", header = F, stringsAsFactors = F)

# convert the format to csvr format for R/qtl
dim(mytest) # 3441 rows and 310 columns

# exclude the first 7 rows and first 4 columns
mygeno <- mytest[-c(1:6), -c(1:4)]

# in my genome, convert "1 2" and "2 1" to H, "1 1" to A and "2 2" to B

for (i in 1:dim(mygeno)[2]){
  mygeno[, i][mygeno[, i] == "1 2"] <- "H"
  mygeno[, i][mygeno[, i] == "2 1"] <- "H"
  mygeno[, i][mygeno[, i] == "1 1"] <- "A"
  mygeno[, i][mygeno[, i] == "2 2"] <- "B"
}

# clolumn combine marker name, linkage group, position and genotype
mycsvr <- cbind.data.frame(LGs$V1,
                           c(rep(1, 645), rep(2, 562), rep(3, 549), rep(4, 494), rep(5, 428), rep(6, 390), rep(7, 367)),
                           LGs$V2,
                           mygeno)

# add one row of phenotype
mycsvr <- rbind.data.frame(c("pheno", "", "", rep(0, 306)), mycsvr)

# write it out as a comma delimited file
write.table(mycsvr, "mycsvr.csv", sep = ",", row.names = F, col.names = F, quote = F)

# now import mycsvr with read.cross funtion in R/qtl
mydata <- read.cross(format = "csvr", file = "mycsvr.csv")
```

```
# plot recombination fraction graph
pdf("RS_lep_map3_rqtl_rf_plot.pdf", 9, 9)
# png("RS_lep_map3_rqtl_rf_plot.png", 1000, 1000)
layout(matrix(1:9, 3, 3, byrow = T))
for (i in 1:7){
  plotRF(mydata, i, what = "rf")
}
dev.off()
```



Above is the recombination fraction (rf) plot of our 7 linkage maps. The plot is actually a matrix of ordered markers, x and y axes are the same ordered markers in one linkage map. The cross point of two makers shows the recombination fraction between them. The darker the color, the higher the recombination frequency. So, along the diagonal line, we expect to see light color (lower rf), but off the diagonal, we expect to see darker color. and the further off the diagonal, the darker the color should be, meaning higher rf. So, we can conclude that, globally, the order of markers in each group is fine, however, we cannot see more detailed local order information from this plot. I think I can continue with these maps.