OneMap Hidden Markov Model parallelization

Cristiane Hayumi Taniguti from Statistical Genetics Lab Department of Genetics Luiz de Queiroz College of Agriculture University of São Paulo

The goal here is to optimize the speed of OneMap Hidden Markov Model (HMM) (Margarido et al., 2007) algorithm to genetic distance estimation. For this, we implemented in OneMap two approaches: `parmap` function and the approach proposed in BatchMap package (Schiffthaler et al., 2017), a fork from OneMap.

The `parmap` splits an ordered sequence into groups according to the number of cores available to the analyses. The groups have an overlap of markers in their edges, to be possible the later joint of them. Here we use these overlap markers also to compare the estimated genetic distance between groups and see how the process can affect the maps distances.

The approach implemented in BatchMap (and now in OneMap) also divides the sequence into batches, but the parallelization is limited by the number of possible phases. Therefore, a maximum of four cores can be used to estimate genetic distances for outcrossing species.

To be a fair comparison, here we use four cores for both approaches.

Tests:

All scenarios have 1 chromosome with 100 cM, 5 family F1, progeny size of 150 individuals, 504 markers, 6 cores, and overlap of 5 markers.

- **Scenario 1**: 28 markers of each one of the 18 possible types, without missing data
- **Scenario 2**: 168 markers of D1.10, D2.15, and B3.7 types, without missing data
- **Scenario 3**: 28 markers of each one of the 18 possible types, 25% of missing data
- **Scenario 4**: 168 markers of D1.10, D2.15 and B3.7 types, 25% of missing data

Measures:

- Total size
- Recombination fraction difference between overlap markers
- Time to run

Another test applied was to increase the tolerance value of the HMM. All scenarios were repeated with tolerance value of 10^{-3} (the default value is 10^{-5}).

Packages

```
library(parallel)
library(onemap) # Since version 2.2.0
library(tidyverse)
```

Functions

```
runcomp <- function(n.marker, mk.types, n.types, miss.perc,tol=10E-5){
  n.fam <- 5
  int.tot.size <- par.tot.size <- int.time <- vector()
  par.time <- int.cms <- par.cms <-  diff2 <- vector()
  batch.tot.size  <- batch.time <- batch.cms <- vector()
```

```r
for(w in 1:n.fam){
  run_pedsim(chromosome = c("Chr1"), n.marker = n.marker,
             tot.size.cm = c(100), centromere = c(50),
             n.ind = 200, mk.types = mk.types,
             n.types = n.types, pop = "F1", path.pedsim = "~/Programs/PedigreeSim/",
             name.mapfile = "mapfile.map", name.founderfile="founderfile.gen",
             name.chromfile="sim.chrom", name.parfile="sim.par",
             name.out="sim_out")

  pedsim2raw(cross="outcross", genofile = "sim_out_genotypes.dat",
             parent1 = "P1", parent2 = "P2",
             out.file = "sim_out.example1.raw", miss.perc = miss.perc)

  df <- read_onemap("sim_out.example1.raw")

  twopts <- rf_2pts(df)

  seq1 <- make_seq(twopts, "all")

  batch_size <- pick_batch_sizes(input.seq = seq1,
                                 size = 80,
                                 overlap = 30,
                                 around = 10)

  batch.time <- rbind(batch.time,
                      system.time(batch.map <- map_overlapping_batches(input.seq = seq1,
                                  size = batch_size,
                                  phase_cores = 4,
                                  overlap = 30, rm_unlinked = T)))

  batch.cms <- rbind(batch.cms, cumsum(c(0, haldane(batch.map[[1]]$seq.rf))))
  batch.tot.size <- c(batch.tot.size, batch.cms[length(batch.cms)])

  int.time <- rbind(int.time, system.time(int.map <- map(seq1, tol=tol)))
  int.cms <- rbind(int.cms, cumsum(c(0,haldane(int.map$seq.rf))))
  int.tot.size <- c(int.tot.size, int.cms[length(int.cms)])

  par.time <- rbind(par.time,
                    system.time(map2 <- parmap(input.seq = seq1,
                                               cores = 6, overlap = 5, tol=tol)))

  diff2 <- rbind(diff2, map2[[1]])

  par.map <- map2[[2]]
  par.cms <- rbind(par.cms, cumsum(c(0,haldane(par.map$seq.rf))))
  par.tot.size <- c(par.tot.size, par.cms[length(par.cms)])
}
result <- list(diff2, int.tot.size, par.tot.size, batch.tot.size,
               int.time, par.time, batch.time, int.cms, par.cms, batch.cms)
names(result) <- c("diff", "int.tot.size", "par.tot.size",
                   "batch.tot.size", "int.time", "par.time",
                   "batch.time", "int.cms", "par.cms", "batch.cms")
  return(result)
}
```

Running each scenario

- scenario (1)

```
cen1 <- runcomp(n.marker = 504,
                mk.types = c("A1", "A2", "A3", "A4", "B1.5", "B2.6", "B3.7",
                             "C.8", "D1.9", "D1.10", "D1.11", "D1.12", "D1.13",
                             "D2.14", "D2.15", "D2.16", "D2.17", "D2.18"),
                n.types = rep(28,18),
                miss.perc = 0)
```

- scenario (2)

```
cen2 <- runcomp(n.marker = 504,
                mk.types = c("B3.7","D1.10", "D2.15"),
                n.types = rep(168,3),
                miss.perc = 0)
```

- scenario (3)

```
cen3 <- runcomp(n.marker = 504,
                mk.types = c("A1", "A2", "A3", "A4", "B1.5", "B2.6", "B3.7",
                             "C.8", "D1.9", "D1.10", "D1.11", "D1.12", "D1.13",
                             "D2.14", "D2.15", "D2.16", "D2.17", "D2.18"),
                n.types = rep(28,18),
                miss.perc = 25)
```

- scenario (4)

```
cen4 <- runcomp(n.marker = 504,
                mk.types = c("B3.7","D1.10", "D2.15"),
                n.types = rep(168,3),
                miss.perc = 25)
```

- scenario (1) tol

```
cen1.tol <- runcomp(n.marker = 504,
                    mk.types = c("A1", "A2", "A3", "A4", "B1.5", "B2.6", "B3.7",
                                 "C.8", "D1.9", "D1.10", "D1.11", "D1.12", "D1.13",
                                 "D2.14", "D2.15", "D2.16", "D2.17", "D2.18"),
                    n.types = rep(28,18),
                    miss.perc = 0, tol=10E-4)
```

- scenario (2) tol

```
cen2.tol <- runcomp(n.marker = 504,
                    mk.types = c("B3.7","D1.10", "D2.15"),
                    n.types = rep(168,3),
                    miss.perc = 0, tol=10E-4)
```

- Scenario (3) tol

```
cen3.tol <- runcomp(n.marker = 504,
                    mk.types = c("A1", "A2", "A3", "A4", "B1.5", "B2.6", "B3.7",
                                 "C.8", "D1.9", "D1.10", "D1.11", "D1.12", "D1.13",
                                 "D2.14", "D2.15", "D2.16", "D2.17", "D2.18"),
                    n.types = rep(28,18),
                    miss.perc = 25, tol=10E-4)
```

- Scenario (4) tol

```
cen4.tol <- runcomp(n.marker = 504,
                    mk.types = c("B3.7","D1.10", "D2.15"),
                    n.types = rep(168,3),
                    miss.perc = 25, tol=10E-4)

save.image("results.RData")
```

Results

Jointing information from all scenarios:

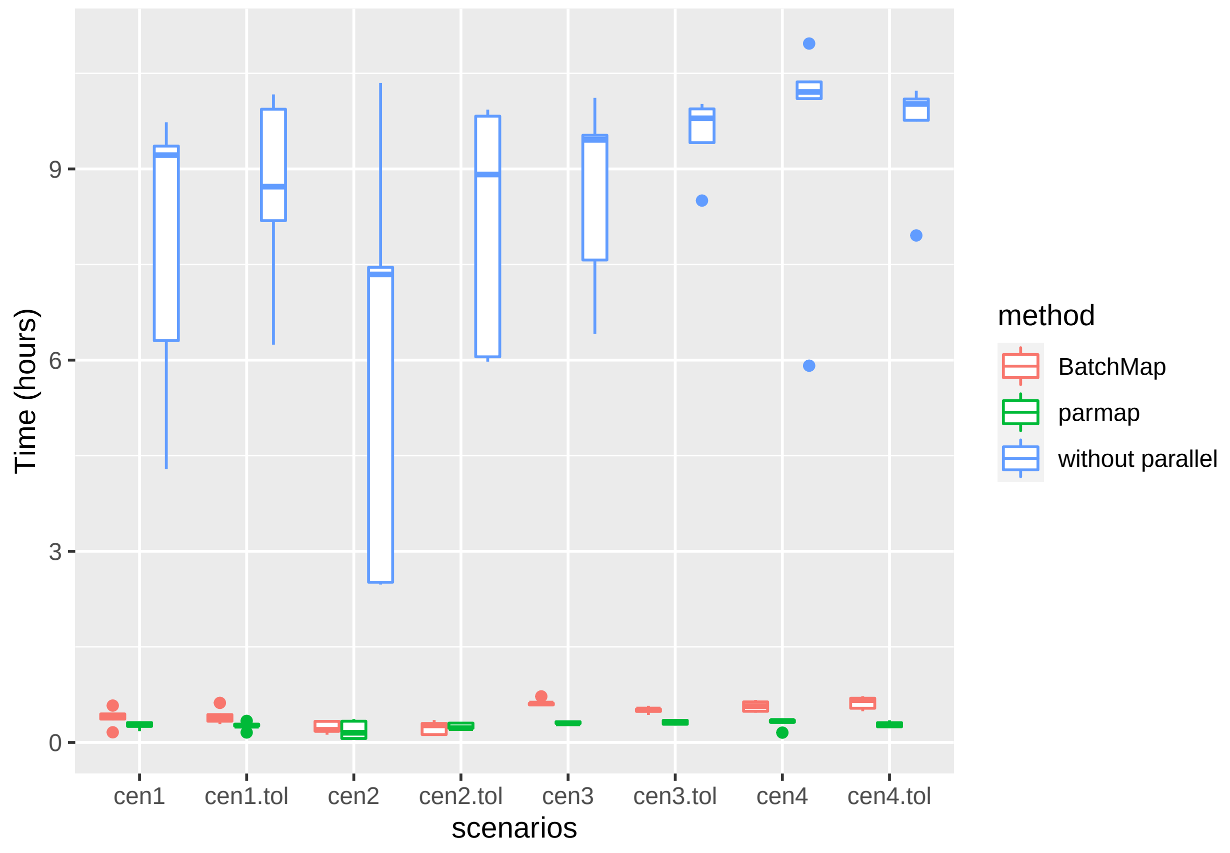
```
load("results.RData")

tot.size.cm <- 100
n.marker <- 504
int <- tot.size.cm/n.marker
pos <- seq(from=0, to=tot.size.cm, by=int)

tot.cen <- list(cen1, cen1.tol, cen2.tol, cen2, cen3, cen3.tol, cen4, cen4.tol)
names(tot.cen) <- c("cen1", "cen1.tol", "cen2", "cen2.tol",
                    "cen3", "cen3.tol", "cen4", "cen4.tol")

df_diff <- df_tot_size <- df_times <- df_sizes <-  vector()
for(i in 1:length(tot.cen)){
  # Diff
  temp.df <- cbind(paste0("simu",1:length(tot.cen[[i]][[2]])),
                   sqrt(tot.cen[[i]][[1]]^2))
  colnames(temp.df) <- c("simu", paste0("Overlap",
                                        1:dim(tot.cen[[i]][[1]])[2]))
  diff <- gather(data.frame(temp.df), key, value, -simu)
  df_diff <- rbind(df_diff, data.frame(scen= names(tot.cen)[i], diff))

  # tot.size
  temp.df <- data.frame(tot.cen[[i]][c(2:3,8)]) # change here
  temp.df <- cbind(simu = paste0("simu", 1:length(tot.cen[[i]][[2]])), temp.df)
  colnames(temp.df) <- c("simu", "all", "parmap", "batchmap")
  df_tot_size <- rbind(df_tot_size, data.frame(scen= names(tot.cen)[i],temp.df))

  # time
  temp.df <- data.frame(simu = paste0("simu", 1:length(tot.cen[[i]][[2]])),
                        int.time= tot.cen[[i]][[4]][,3],
                        par.time = tot.cen[[i]][[5]][,3],
                        batch.time = tot.cen[[i]][[9]][,3]) # change here
  colnames(temp.df) <- c("simu", "all", "parallel", "batchmap")
```

```r
  df_times <- rbind(df_times, data.frame(scen= names(tot.cen)[i],temp.df))


  # cMs

  temp.int <- t(apply(tot.cen[[i]][[6]],1, function(x) sqrt((x-pos[-1])^2)))
  temp.par <- t(apply(tot.cen[[i]][[7]],1, function(x) sqrt((x-pos[-1])^2)))
  temp.batch <- t(apply(tot.cen[[i]][[10]],1, function(x) sqrt((x-pos[-1])^2)))

  colnames(temp.int) <- paste0("MK",1:dim(temp.int)[2])
  colnames(temp.par) <- paste0("MK",1:dim(temp.par)[2])
  colnames(temp.batch) <- paste0("MK",1:dim(temp.batch)[2])

  temp.int <- data.frame(simu = paste0("simu", 1:length(tot.cen[[i]][[2]])),
                         temp.int)
  temp.int <- gather(temp.int, key, value,-simu)
  temp.par <- data.frame(simu = paste0("simu", 1:length(tot.cen[[i]][[2]])),
                         temp.par)
  temp.par <- gather(temp.par, key, value,-simu)
  temp.batch <- data.frame(simu = paste0("simu", 1:length(tot.cen[[i]][[2]])),
                         temp.batch)
  temp.batch <- gather(temp.batch, key, value,-simu)

  temp.df <- merge(temp.int, temp.par, by = c("simu", "key"))
  temp.df <- merge(temp.df, temp.batch, by=c("simu", "key"))
  colnames(temp.df) <- c("simu", "mk", "all", "parmap", "batchmap")
  df_sizes <- rbind(df_sizes, data.frame(scen = names(tot.cen)[i],temp.df))
}
```

Time

Comparison between time spent for each approach: BatchMap, parmap, and OneMap without parallelization (map function).

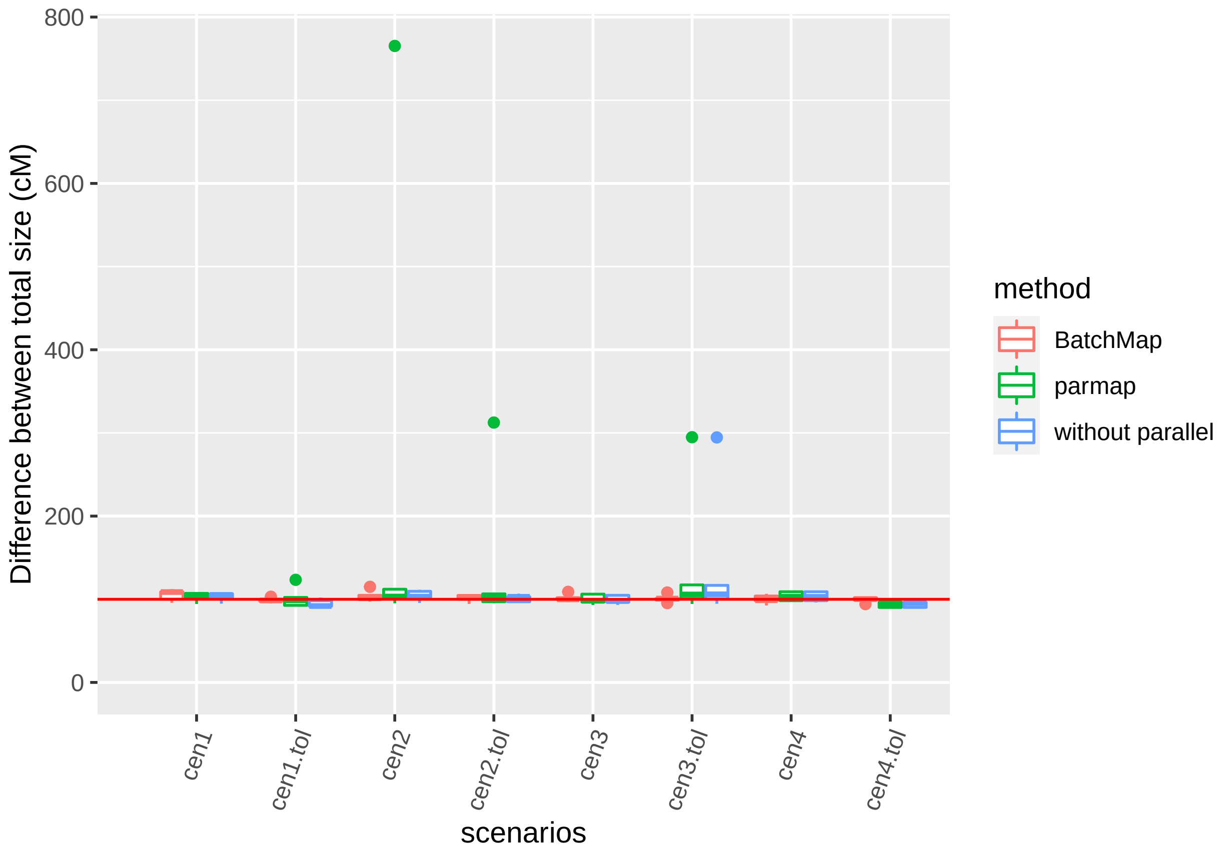
```r
# rename columns
colnames(df_times)[3:5] <- c("without parallel", "parmap", "BatchMap")

df_times %>% gather(key, value, -simu, -scen) %>%
  ggplot(aes(x=scen, y=value/3600, color=key)) +
  geom_boxplot() +
  xlab("scenarios") +
  ylab("Time (hours)") +
  scale_color_discrete(name="method")
```



Total sizes

Comparison between total map size built by each approach: BatchMap, parmap, and OneMap without parallelization (map function). Remember here that the simulated size was 100 cM (red line).

```r
# rename columns
colnames(df_tot_size)[3:5] <- c("without parallel", "parmap", "BatchMap")

df_tot_size %>% gather(key, value, -scen, -simu) %>%
  ggplot(aes(x=scen, y=value, color=key)) +
  geom_boxplot() +
  geom_hline(yintercept=100, color = "red") +
  theme(axis.text.x = element_text(angle = 70, hjust = 1)) +
  xlab("scenarios") +
  ylab("Difference between total size (cM)") +
  scale_color_discrete(name="method") +
  expand_limits(x = 0, y = 0)
```

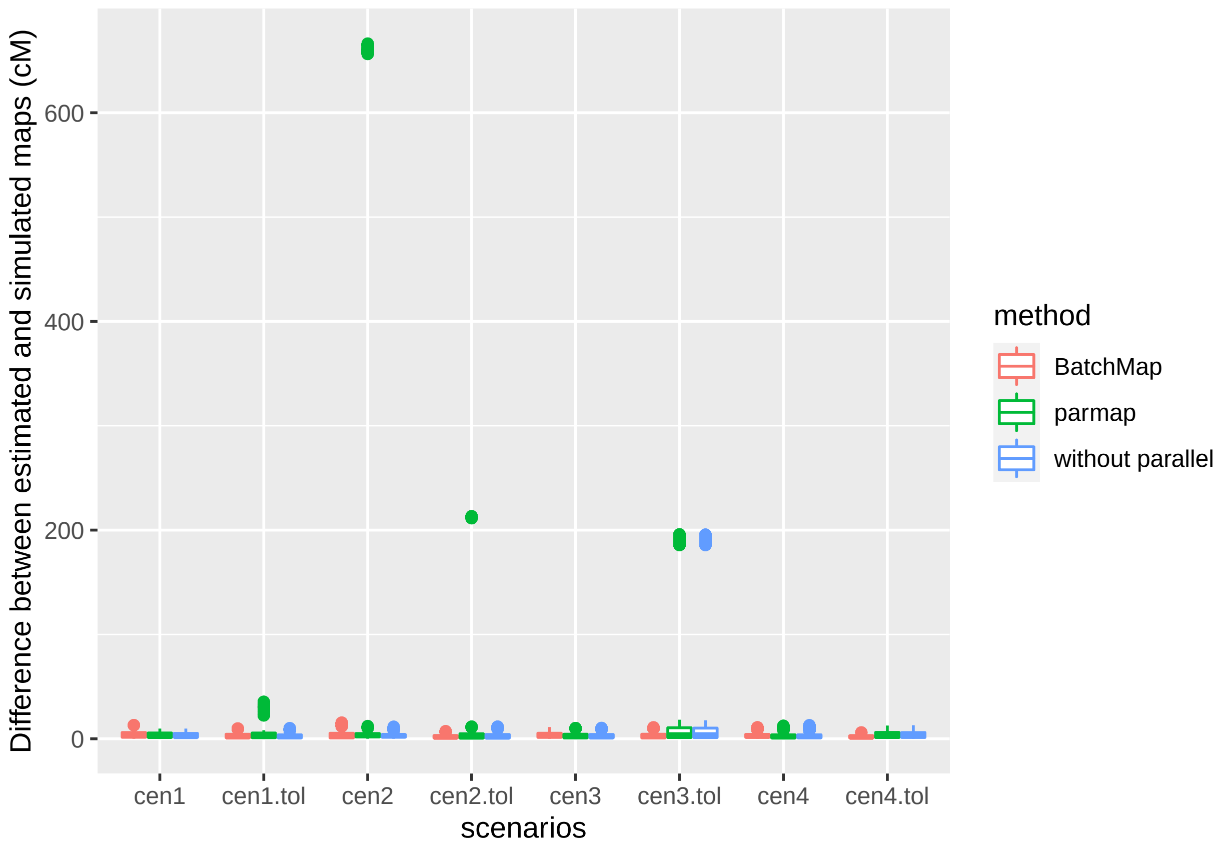


Intervals difference

Size differences in the intervals between the markers of the generated map and the simulated map.

PS: the best result would be if all the points are close to the x axis (difference 0)

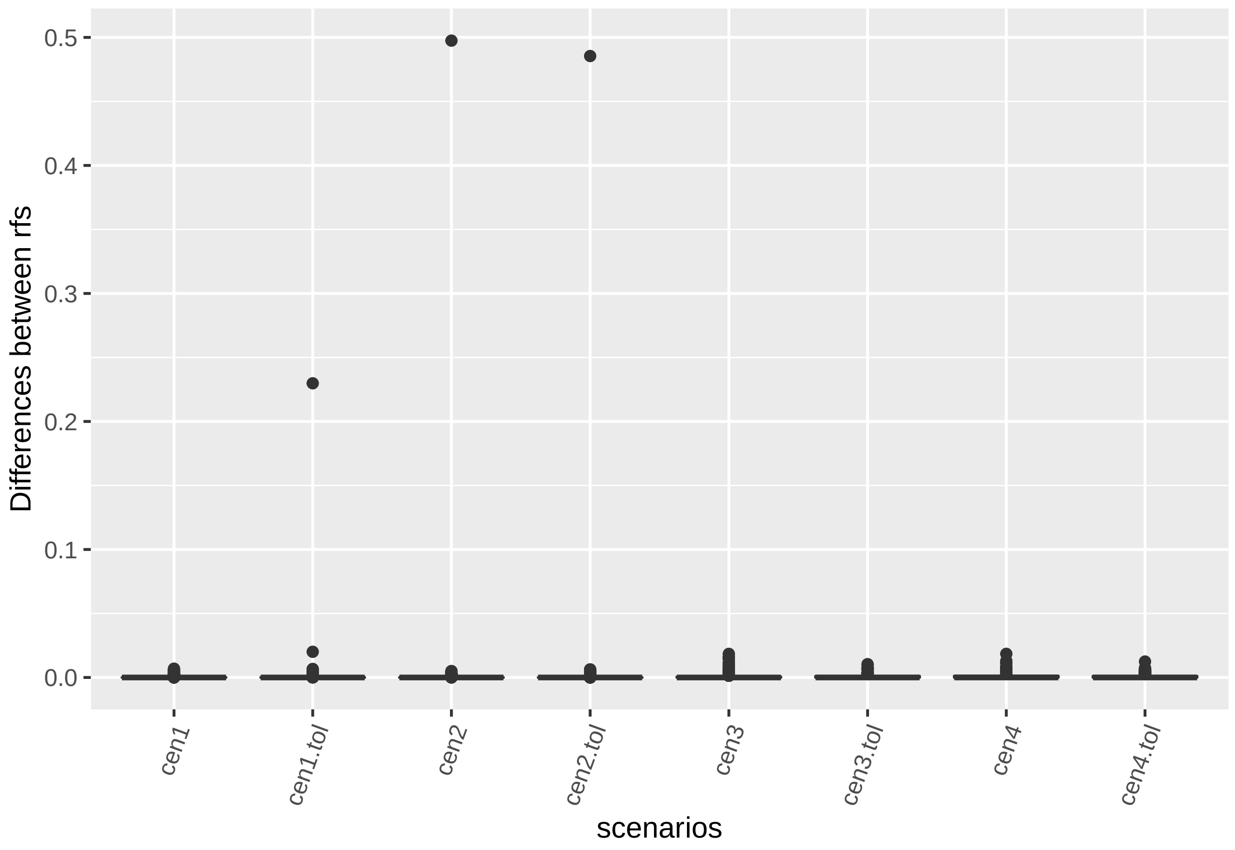
```r
# rename columns
colnames(df_sizes)[4:6] <- c("without parallel", "parmap", "BatchMap")

df_sizes %>% gather(key, value, -scen, -simu, -mk) %>%
  ggplot(aes(x=scen, y=value, color=key)) +
  geom_boxplot() +
  scale_color_discrete(name="method") +
  xlab("scenarios") +
  ylab("Difference between estimated and simulated maps (cM)")
```



Evaluation of overlap markers in parmap

Measure the difference of recombination fraction between overlap markers, in other words, the markers which are at the end of the last group and at the beginning of the next group.

```
df_diff %>% ggplot(aes(x = scen, y = as.numeric(value))) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 70, hjust = 1)) +
  xlab("scenarios") +
  ylab("Differences between rfs")
```



Conclusion

Both new approaches (parmap and BatchMap) implemented in OneMap are efficient in reducing the time spent to run the HMM and estimate the genetic distances. Parmap has the capacity to reduce even more the time spent because it can use more than four cores to do the parallelization, but smaller is the marker batches higher the chance of occurring errors in estimation. In general, parmap make more mistakes than BatchMap, therefore, if the priority is quality instead of fast analysis, BatchMap is ideal. BatchMap approach produces maps similar to approach without parallelization and seems to be even more stable (lower variation) between the simulations.

References

Margarido, G. R. A., Souza, A. P., & Garcia, A. A. F. (2007). OneMap: software for genetic mapping in outcrossing species. Hereditas, 144(3), 78–79. https://doi.org/10.1111/j.2007.0018-0661.02000.x

Schiffthaler, B., Bernhardsson, C., Ingvarsson, P. K., & Street, N. R. (2017). BatchMap: A parallel implementation of the OneMap R package for fast computation of F1 linkage maps in outcrossing species. PLoS ONE, 12(12), 1–12. https://doi.org/10.1371/journal.pone.0189256