RClone quickmanual: several populations

Diane Bailleul, Solenn Stoeckel and Sophie Arnaud-Haond

2021-05-15

"Eager Beginners" Manual for RClone package

RClone data format: several populations

A. Introduction to RClone

RClone is a R package version of *GenClone* program (Arnaud-Haond & Belkhir 2007): to analyse data (SSR, SNP, . . .), test for clonality and describe spatial clonal organisation.
Major improvements are multi-populations handling and definition of MLLs (Multilocus Lineages, i.e. slightly distinct Multi Locus Genotypes) through simulations.

RClone allows:

1. Description of data set

- discrimination of MLG (MultiLocus Genotypes);
- test for reliability of data (in terms of loci and sampling).

2. Determination of MLL (MultiLocus Lineages)

- psex/psex Fis with pvalue computation;
- genetic distance matrix computation and threshold definition.

3. Genotypic diversity and evenness indices calculation

- Simpson complement;
- Shannon-Wiener diversity and evenness indices;
- Hill's Simpson reciprocal;
- Pareto index.

4. Spatial organisation of MLG/MLL

- spatial autocorrelation methods;
- clonal subrange estimation;
- Aggregation and Edge Effect indices estimation.

Some of these analysis can be applied to dataset with no repeated MLG, regardless of the reproductive system (sexual, partially asexual or strictly asexual).

B. RClone data format: several population

RClone functions works on diploid/haploid, one or several populations dataset.

If you have only one population in your dataset, go to other vignette *RClone_quickmanual*.

C. General format

If you have haploid data, you can skip to *D. Description of data set*.

To use *RClone* functions, your data table must look like:

```
library(RClone)
data(posidonia)
```

Po15_1	Po15_2	Po4-3_1	Po4-3_2	Po5-10_1	Po5-10_2	Po5-39_1	Po5-39_2
137	161	182	188	212	216	234	234
139	171	182	182	222	226	234	242
161	161	182	182	210	216	234	234
161	161	182	182	210	216	234	234
161	161	182	182	210	216	234	234
161	161	182	182	210	216	234	234
161	161	182	182	210	216	234	234
161	161	182	182	210	216	234	234
137	157	182	188	208	210	234	234
137	157	174	180	208	210	234	234

There is only one allele per column and, per locus, alleles are sorted by increasing order.

This is **mandatory** for all *RClone* functions.

As formatting can be source of error, we included functions to help formatting your diploid data:

1, The classic infile you could have, one locus per column

```
data(zostera)
head(zostera)
```

population	x	y	GA35	GA2	GA17H	GA23	GA12	GA19	GA20	GA16	GA17D
SaintMalo	0.0	18.0	185187	102116	131135	167169	131131	148148	162162	168168	197197
SaintMalo	0.0	15.5	185187	102116	131135	167169	131131	148148	162162	168168	197197
SaintMalo	0.0	3.5	187187	102116	127133	161169	131131	148148	162162	168168	197197
SaintMalo	2.0	3.5	187187	102116	133135	159161	131131	148148	162162	168168	197197
SaintMalo	6.5	18.0	187187	102116	135141	169169	131137	148150	162162	168168	197197
SaintMalo	6.5	10.0	187187	116116	133133	167167	131131	148148	162162	168168	195197

Zostera data is composed of:
* a first column with population indication;
* a second and third columns with x/y coordinates;
* a genotypic dataset.

```
popvec <- zostera[,1] #futur vecpop
coord_zostera <- zostera[,2:3] #futur coordinates
zostera <- zostera[,4:ncol(zostera)] #dataset

zostera <- convert_GC(zostera, 3) #We used "3" because this is the length of each allele.
```

```
head(zostera)
```

GA35_1	GA35_2	GA2_1	GA2_2	GA17H_1	GA17H_2	GA23_1
185	187	102	116	131	135	167
185	187	102	116	131	135	167
187	187	102	116	127	133	161
187	187	102	116	133	135	159
187	187	102	116	135	141	169
187	187	116	116	133	133	167

2, The simple case: you already have a one-allele per column table

Just remove the pop/coords informations as above and sort your alleles:

```
sort_all(zostera)
```

3, You already work with Adegenet

Similar to case number 1, except you have to export your `genind` data into table first:

```
#library(adegenet)
#with data1, a genind object from Adegenet:

test <- genind2df(data1)
data2 <- convert_GC(test, 3, "/")
#only if yours alleles are of length "3"
```

D. Description of data set

D.1 Discrimination of MLG

List unique alleles per locus:

Basic commands:

```r
list_all_tab(zostera, vecpop = popvec)
```

or, for haploid data:

```r
list_all_tab(haplodata, haploid = TRUE, vecpop = haplovec)
```

Results:

```r
list_all_tab(zostera, vecpop = popvec)
```

```r
#SaintMalo
```

locus_1	locus_2	locus_3	locus_4	locus_5	locus_6	locus_7	locus_8	locus_9
185	102	131	167	131	148	162	168	197
187	116	127	161	137	150			195
189		133	159					
		135	169					
		137	163					
		119						
		141						

```r
#Arcouest
```

locus_1	locus_2	locus_3	locus_4	locus_5	locus_6	locus_7	locus_8	locus_9
187	102	131	161	131	150	162	168	197
189	116	129	169		148	156	166	
185	108	141	167			160		
	118	133				166		
	120	143				164		
		135						

List MLG:

Basic commands:

```r
MLG_tab(zostera, vecpop = popvec)
```

or, for haploid data:

```r
MLG_tab(haplodata, vecpop = haplovec)
```

Results:

```r
MLG_tab(zostera, vecpop = popvec)[[1]]
#SaintMalo
```

unit_1	unit_2	unit_3	unit_4	unit_5
1	2			
3				
4	11			
5	7	8	9	12
6				

Allelic frequencies:

Basic commands:

```
freq_RR(zostera, vecpop = popvec)
```

or, for haploid data:

```
freq_RR(haplodata, haploid = TRUE, vecpop = haplovec)
```

Options:

```
freq_RR(zostera, vecpop = popvec) #on ramets
freq_RR(zostera, vecpop = popvec, genet = TRUE) #on genets
freq_RR(zostera, vecpop = popvec, RR = TRUE) #Round-Robin methods
```

Results:

```
freq_RR(zostera, vecpop = popvec)[[1]]
#SaintMalo
```

locus	allele	freq_ramet	freq_genet	freq_RR
locus_1	185	0.0517241	0.0588235	0.0588235
locus_1	187	0.9137931	0.8823529	0.8823529
locus_1	189	0.0344828	0.0588235	0.0588235
locus_2	102	0.5000000	0.5000000	0.5000000
locus_2	116	0.5000000	0.5000000	0.5000000
locus_3	119	0.0172414	0.0294118	0.0294118
locus_3	127	0.0689655	0.1176471	0.1176471

D.2 Tests for reliability of loci and subsampling of individuals

On loci

Basic commands:

```
sample_loci(zostera, vecpop = popvec, nbrepeat = 1000)
```

or, for haploid data:

```
sample_loci(haplodata, haploid = TRUE, vecpop = haplovec, nbrepeat = 1000)
```

Options:

```
sample_loci(zostera, vecpop = popvec, nbrepeat = 1000, He = TRUE) #with He results
sample_loci(zostera, vecpop = popvec, nbrepeat = 1000, graph = TRUE) #graph displayed
sample_loci(zostera, vecpop = popvec, nbrepeat = 1000, bar = TRUE)
                                       #progression bar, could be time consuming
sample_loci(zostera, vecpop = popvec, nbrepeat = 1000, export = TRUE)
                                       #graph export in .eps
```

Results:

```
res <- sample_loci(zostera, vecpop = popvec, nbrepeat = 1000, He = TRUE)
names(res)
```

```
> [1] "SaintMalo" "Arcouest"
```

```r
names(res$SaintMalo)
```

```r
names(resvigncont2$res2_SU1$SaintMalo)
```

```r
#Results: MLG
res$Arcouest$res_MLG
```

nb_loci	min	max	mean_MLG	SE
1	1	8	3.861	0.0819780
2	1	12	6.997	0.1021539
3	2	14	9.800	0.0873248
4	4	16	11.757	0.0678864
5	7	16	13.157	0.0555647
6	11	16	14.299	0.0414405
7	13	16	15.061	0.0307933
8	14	16	15.576	0.0210396
9	16	16	16.000	0.0000000

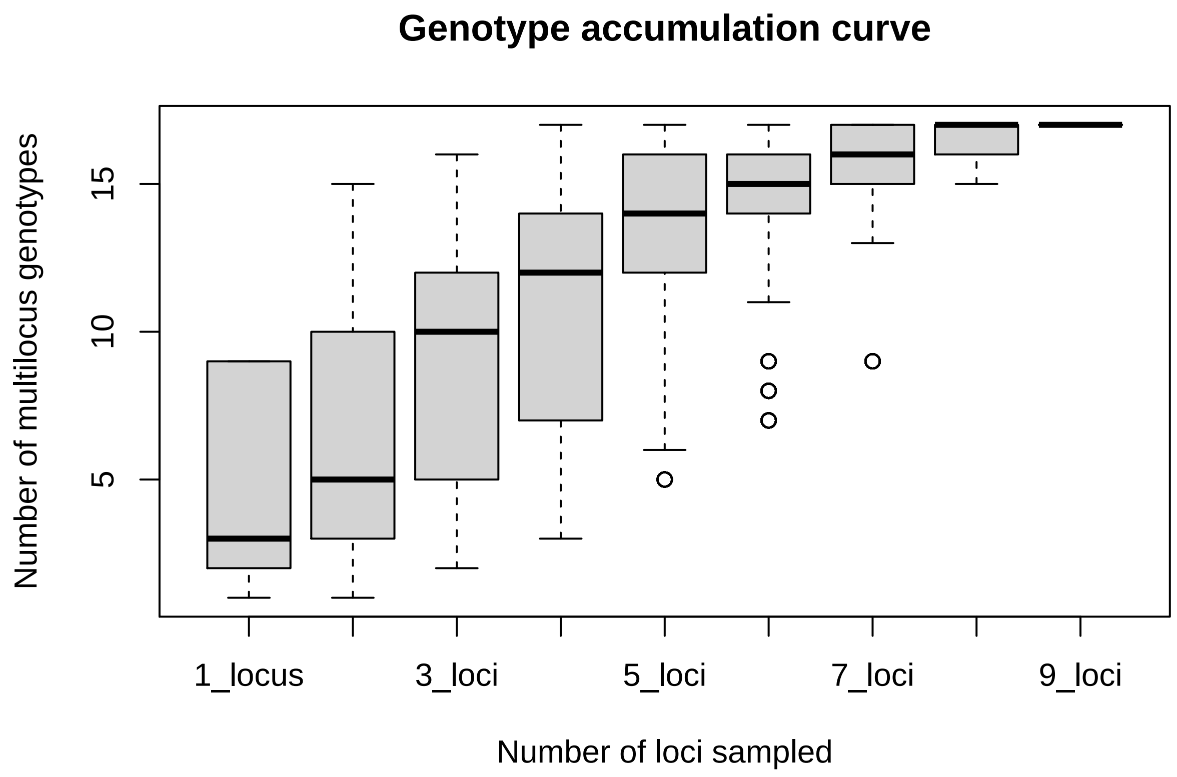
```r
#Results: alleles
res$Arcouest$res_alleles
```

```
> Warning in kable_pipe(x = structure(character(0), .Dim = c(0L, 0L), .Dimnames =
> list(: The table should have a header (column names)
```

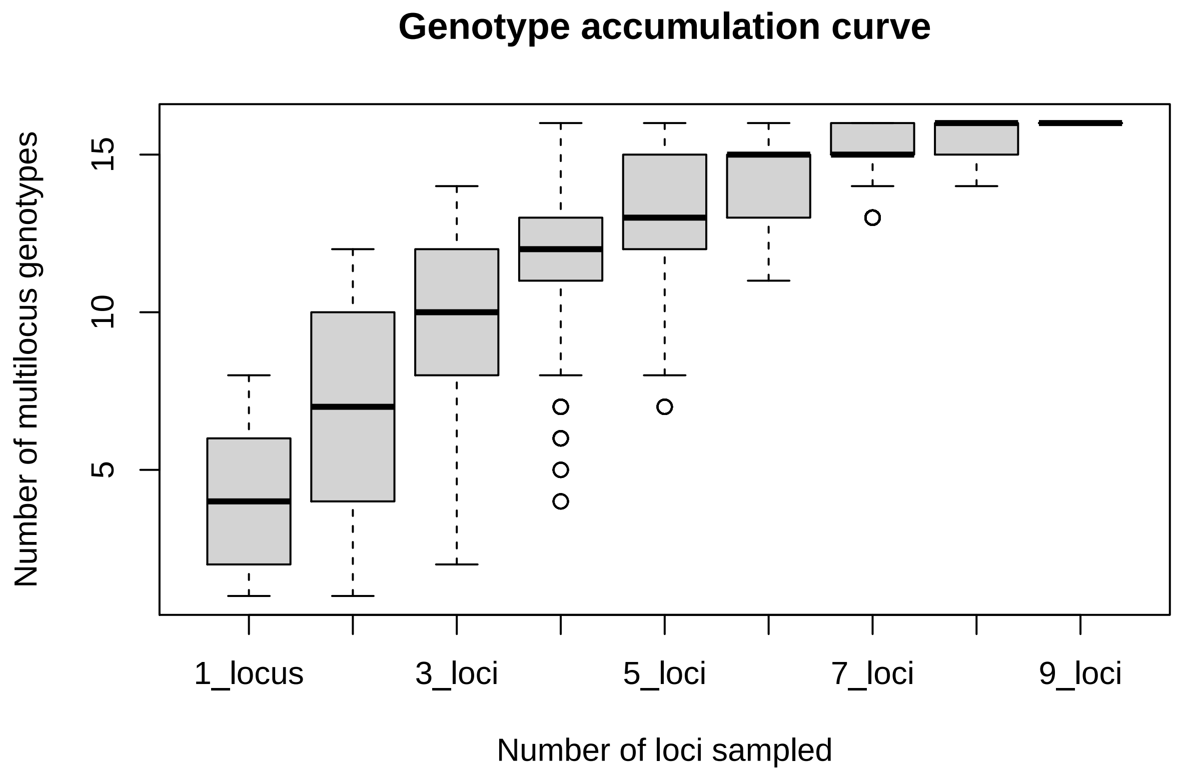
|| || || ||

```r
#Results: raw data
#res$Arcouest$raw_He
#res$Arcouest$raw_MLG
#res$Arcouest$raw_all
```

```r
boxplot(res$SaintMalo$raw_MLG, main = "Genotype accumulation curve",
    xlab = "Number of loci sampled", ylab = "Number of multilocus genotypes")
```



```r
boxplot(res$Arcouest$raw_MLG, main = "Genotype accumulation curve",
    xlab = "Number of loci sampled", ylab = "Number of multilocus genotypes")
```



Same on units

Basic commands:

```r
sample_units(zostera, vecpop = popvec, nbrepeat = 1000)
```

or, for haploid data:

```r
sample_units(haplodata, haploid = TRUE, vecpop = haplovec, nbrepeat = 1000)
```

This sub-sampling analysis deliver basic estimates of richness and diversity for an increasing number of sampling units.
They can be used to standardise estimates of populations with different sampling effort.

E. Discrimination of clonal lineages

E.1 psex/psex Fis with pvalue computation

pgen, psex and p-values

Basic commands:

```r
pgen(zostera, vecpop = popvec)
psex(zostera, vecpop = popvec)
```

or, for haploid data:

```r
pgen(haplodata, haploid = TRUE, vecpop = haplovec)
psex(haplodata, haploid = TRUE, vecpop = haplovec)
```

Options: *(idem on psex and pgen)*

```r
#allelic frequencies computation:
psex(zostera, vecpop = popvec) #psex on ramets
psex(zostera, vecpop = popvec, genet = TRUE) #psex on genets
psex(zostera, vecpop = popvec, RR = TRUE) #psex with Round-Robin method
#psex computation
psex(zostera, vecpop = popvec) #psex with one psex per replica
psex(zostera, vecpop = popvec, MLGsim = TRUE) #psex MLGsim method
#pvalues:
psex(zostera, vecpop = popvec, nbrepeat = 100) #with p-values
psex(zostera, vecpop = popvec, nbrepeat = 1000, bar = TRUE)
                                        #with p-values and a progression bar
```

Results:

```r
res <- psex(zostera, vecpop = popvec, RR = TRUE, nbrepeat = 1000)
res$Arcouest[[1]]
#if nbrepeat != 0, res contains a table of psex values and a vector of sim-psex values
```

pgen	genet	psex	pvalue
0.0001692			
0.0000416			
0.0000900			
0.0000416	2	0.00124641891362029	0.00571428571428571
0.0000000			
0.0001800			

```
res$Arcouest[[2]] #a part of sim-psex values
```

```
>  [1] 0.005507812 0.080799100 0.073342047 0.080799100 0.019908965 0.008798312
>  [7] 0.002194897 0.003586792 0.046359147 0.116134553
```

Fis, pgen Fis, psex Fis and p-values

Not for haploid data !

Fis

Basic commands:

```
Fis(zostera, vecpop = popvec)
```

Options:

```
Fis(zostera, vecpop = popvec) #Fis on ramets
Fis(zostera, vecpop = popvec, genet = TRUE) #Fis on genets
Fis(zostera, vecpop = popvec, RR = TRUE) #Fis with Round-Robin methods
#RR = TRUE contains two results : a table with allelic frequencies
                             #and a table with Fis results
```

Results:

```
Fis(zostera, vecpop = popvec, RR = TRUE)$Arcouest[[2]]
```

locus	Hobs	Hatt	Fis
locus_1	0.2666667	0.3300242	0.1919786
locus_2	0.7500000	0.6995968	-0.0720461
locus_3	0.6250000	0.7721774	0.1906005
locus_4	0.6250000	0.5383065	-0.1610487
locus_5	0.0000000	0.0000000	NaN
locus_6	0.2000000	0.1862069	-0.0740741
locus_7	0.2857143	0.3772941	0.2427280
locus_8	0.1875000	0.1754032	-0.0689655
locus_9	0.0000000	0.0000000	NaN

pgen Fis, psex Fis and p-values

Basic commands: *(idem for pgen_Fis and psex_Fis)*

```
pgen_Fis(zostera, vecpop = popvec)
```

Options:

```
#allelic frequencies:
psex_Fis(zostera, vecpop = popvec) #psex Fis on ramets
psex_Fis(zostera, vecpop = popvec, genet = TRUE) #psex Fis on genets
psex_Fis(zostera, vecpop = popvec, RR = TRUE) #psex Fis with Round-Robin method
#psex computation
psex_Fis(zostera, vecpop = popvec) #psex Fis, one for each replica
psex_Fis(zostera, vecpop = popvec, MLGsim = TRUE) #psex Fis with MLGsim method
#pvalues
psex_Fis(zostera, vecpop = popvec, nbrepeat = 100) #with p-values
psex_Fis(zostera, vecpop = popvec, nbrepeat = 1000, bar = TRUE)
                                        #with p-values and a progression bar
```

Results:

```
res <- psex_Fis(zostera, vecpop = popvec, RR = TRUE, nbrepeat = 1000)
res$Arcouest[[1]]
#if nbrepeat != 0, res contains a table of psex values and a vector of sim-psex Fis values
```

pgenFis	genet	psexFis	pvalue
0.0001459 0.0000587 0.0000708 0.0000587 0.0000003 0.0001417	2	0.00175772228659339	0.0154639175257732

```
res$Arcouest[[2]] #a part of sim psex Fis values
```

```
>  [1] 0.046691095 0.065845604 0.047964606 0.030472230 0.029045389 0.009682317
>  [7] 0.101570928 0.200020973 0.004458226 0.033269017
```

E.2 Tests for MLLs occurrence and assessment of their memberships

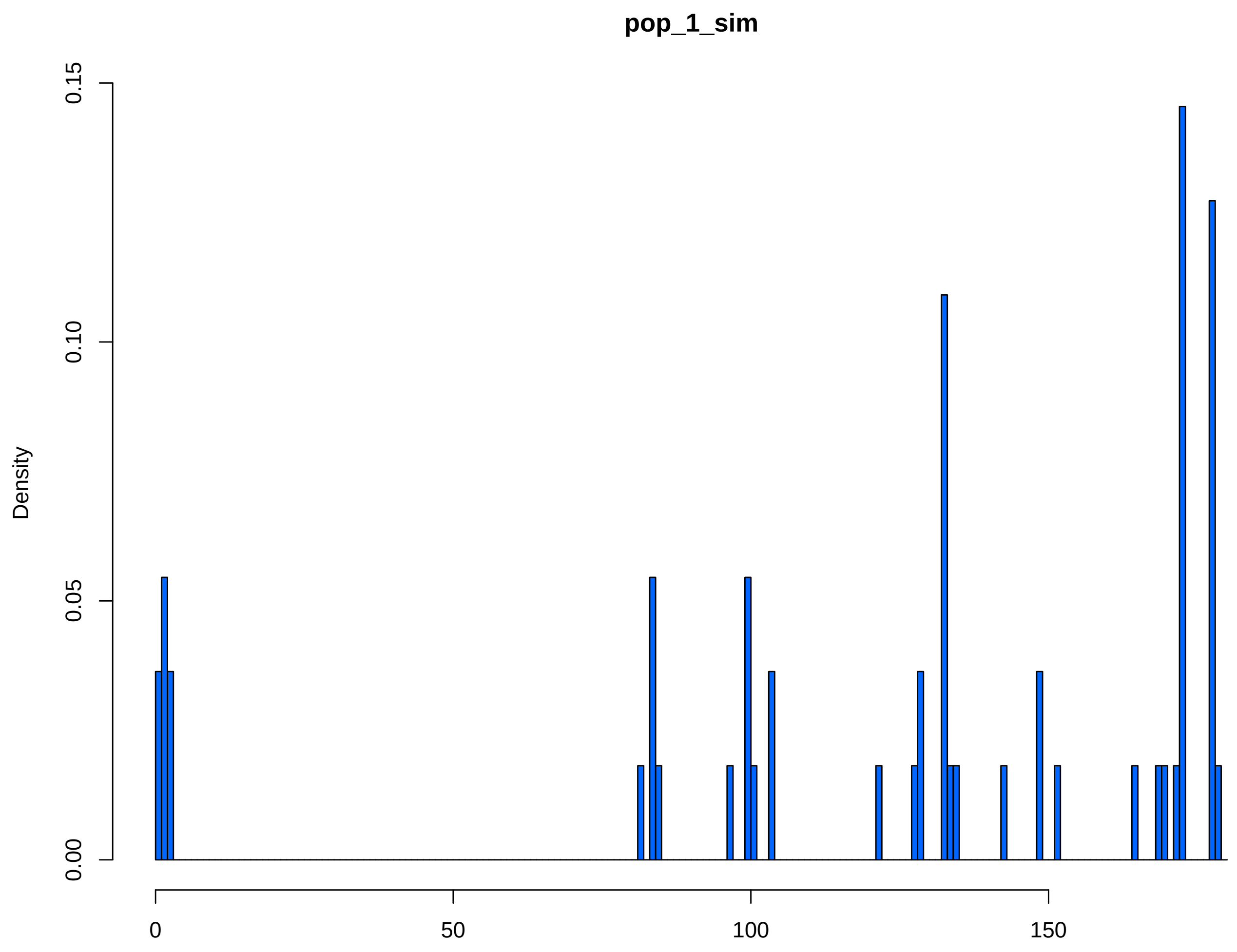
Genetic distance matrix computation and threshold definition

On a theoretical diploid population with c = 0.9999 (c, clonality rate):

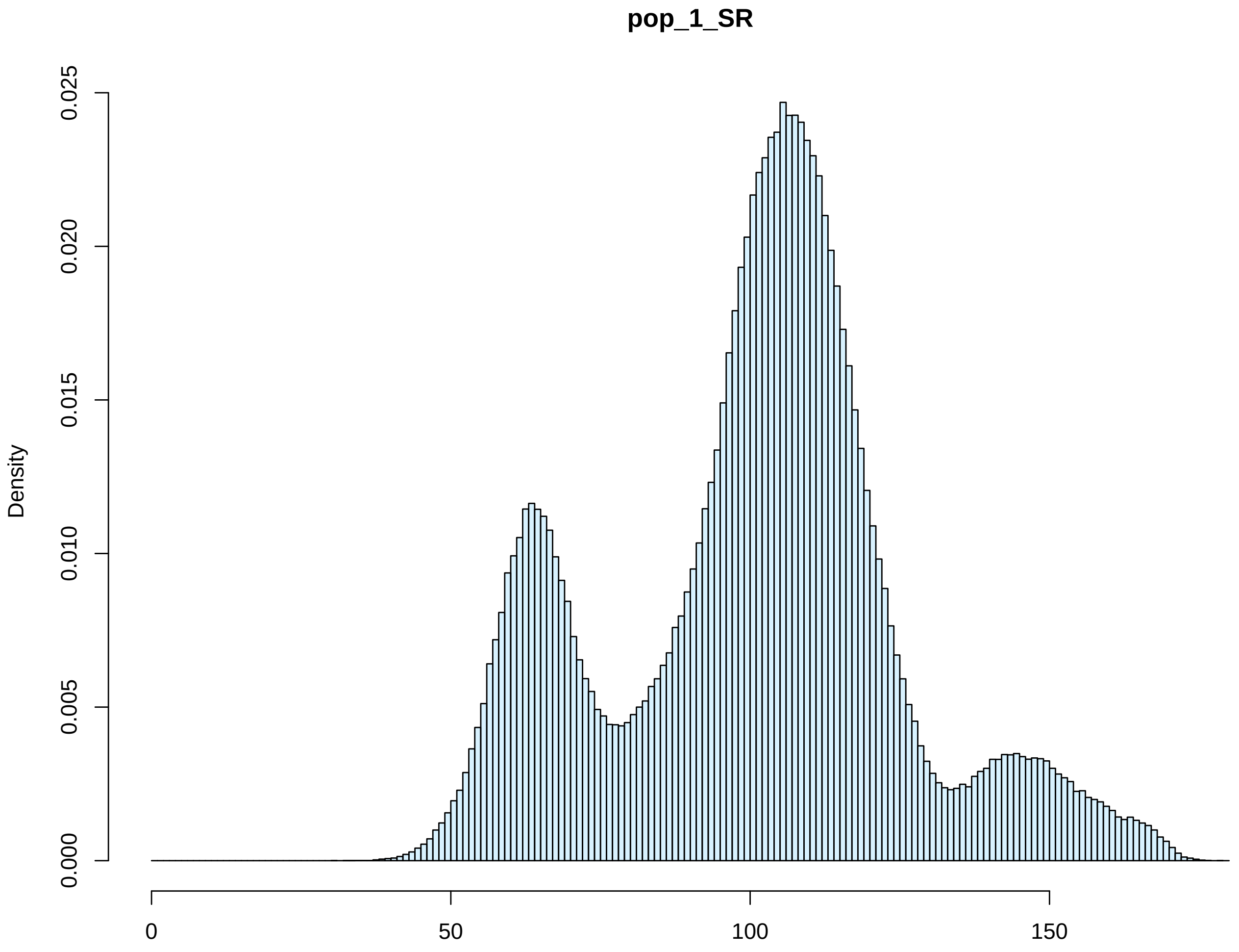
```
data(popsim)
vecsim <- c(rep(1,50), rep(2,50))
```

```
#genetic distances computation, distance on allele differences:
respop <- genet_dist(popsim, vecpop = vecsim)
ressim <- genet_dist_sim(popsim, vecpop = vecsim , nbrepeat = 1000)
                                #theoretical distribution: sexual reproduction
ressimWS <- genet_dist_sim(popsim, vecpop = vecsim , genet = TRUE, nbrepeat = 1000)
                                                            #idem, without selfing
```

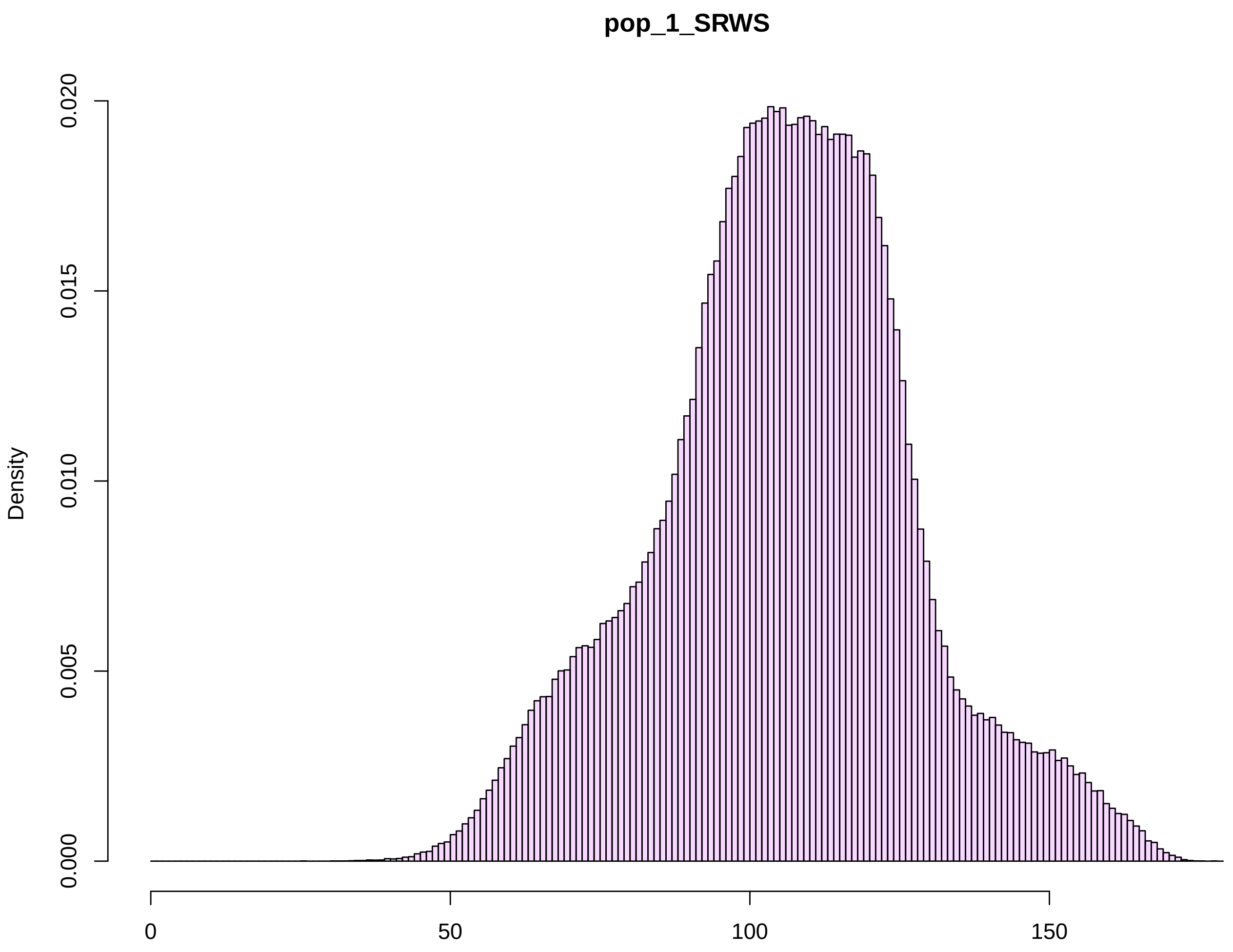
```
#graph prep.:
#first pop:
p1 <- hist(respop[[1]]$distance_matrix, freq = FALSE, col = rgb(0,0.4,1,1),
            breaks = seq(0, max(respop[[1]]$distance_matrix)+1, 1),
            main = "pop_1_sim", xlab = "")
```



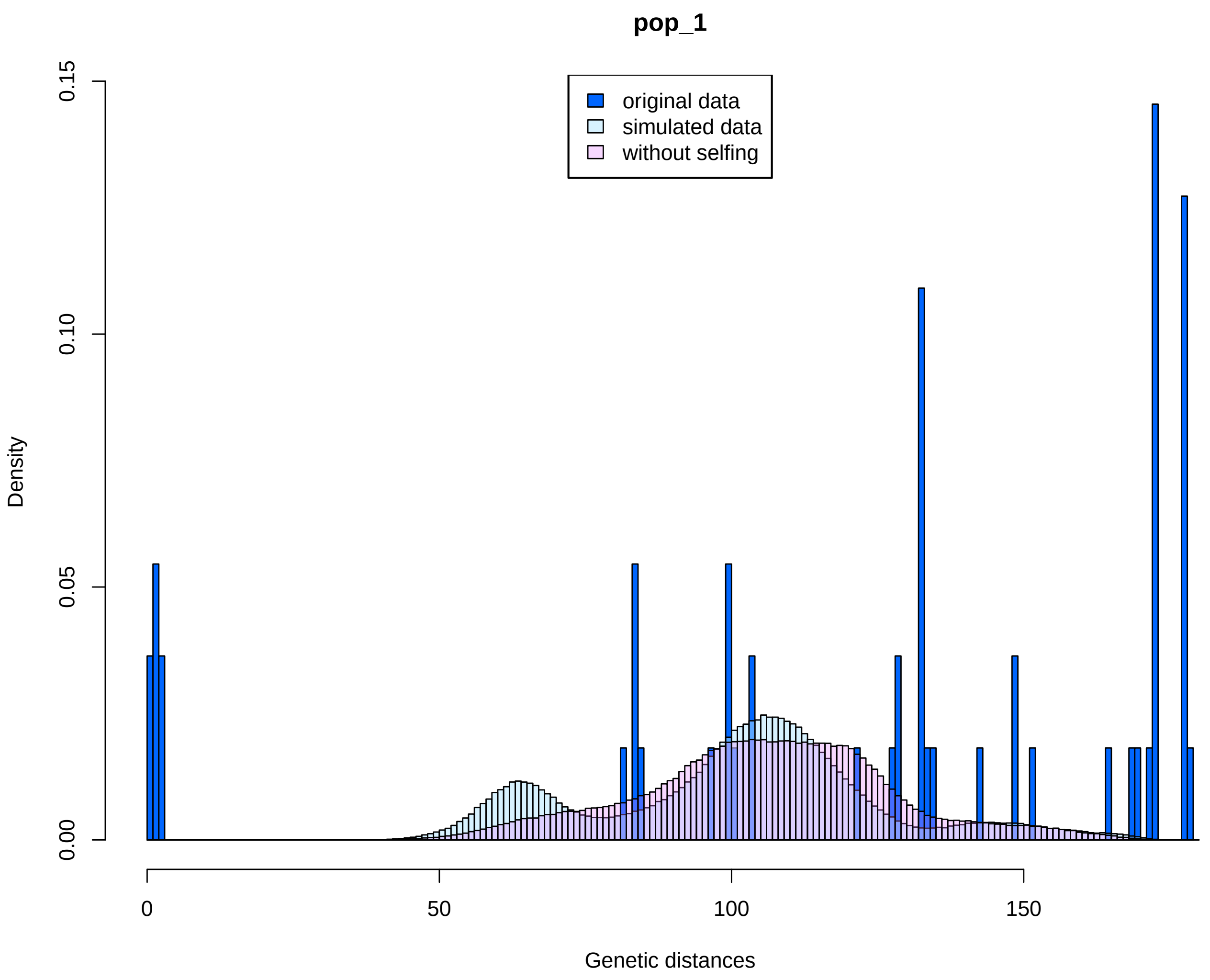
```
p2 <- hist(ressim[[1]]$distance_matrix, freq = FALSE, col = rgb(0.7,0.9,1,0.5),
           breaks = seq(0, max(ressim[[1]]$distance_matrix)+1, 1),
           main = "pop_1_SR", xlab = "")
```



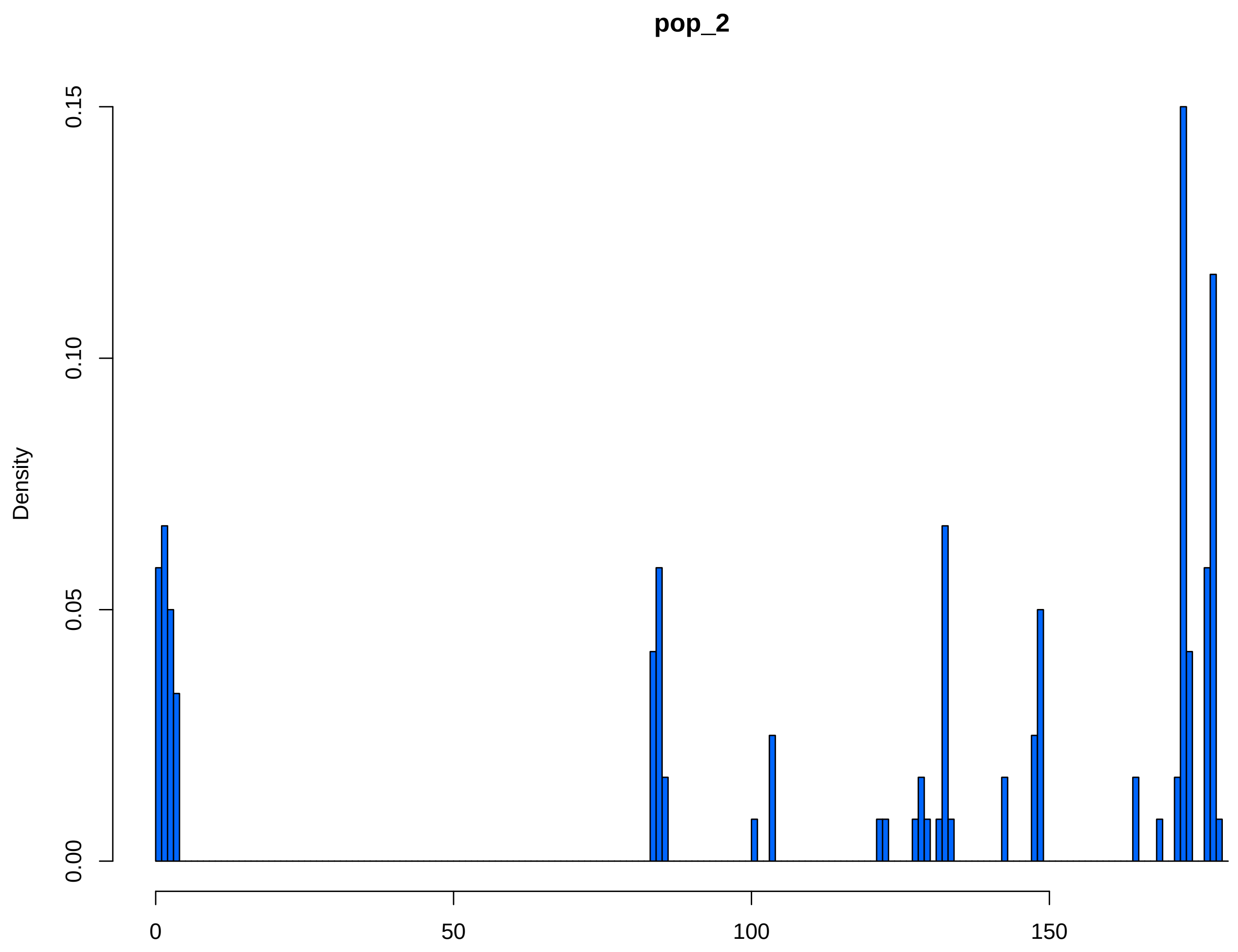
```
p3 <- hist(ressimWS[[1]]$distance_matrix, freq = FALSE, col = rgb(0.9,0.5,1,0.3),
            breaks = seq(0, max(ressimWS[[1]]$distance_matrix)+1, 1),
            main = "pop_1_SRWS", xlab = "")
```



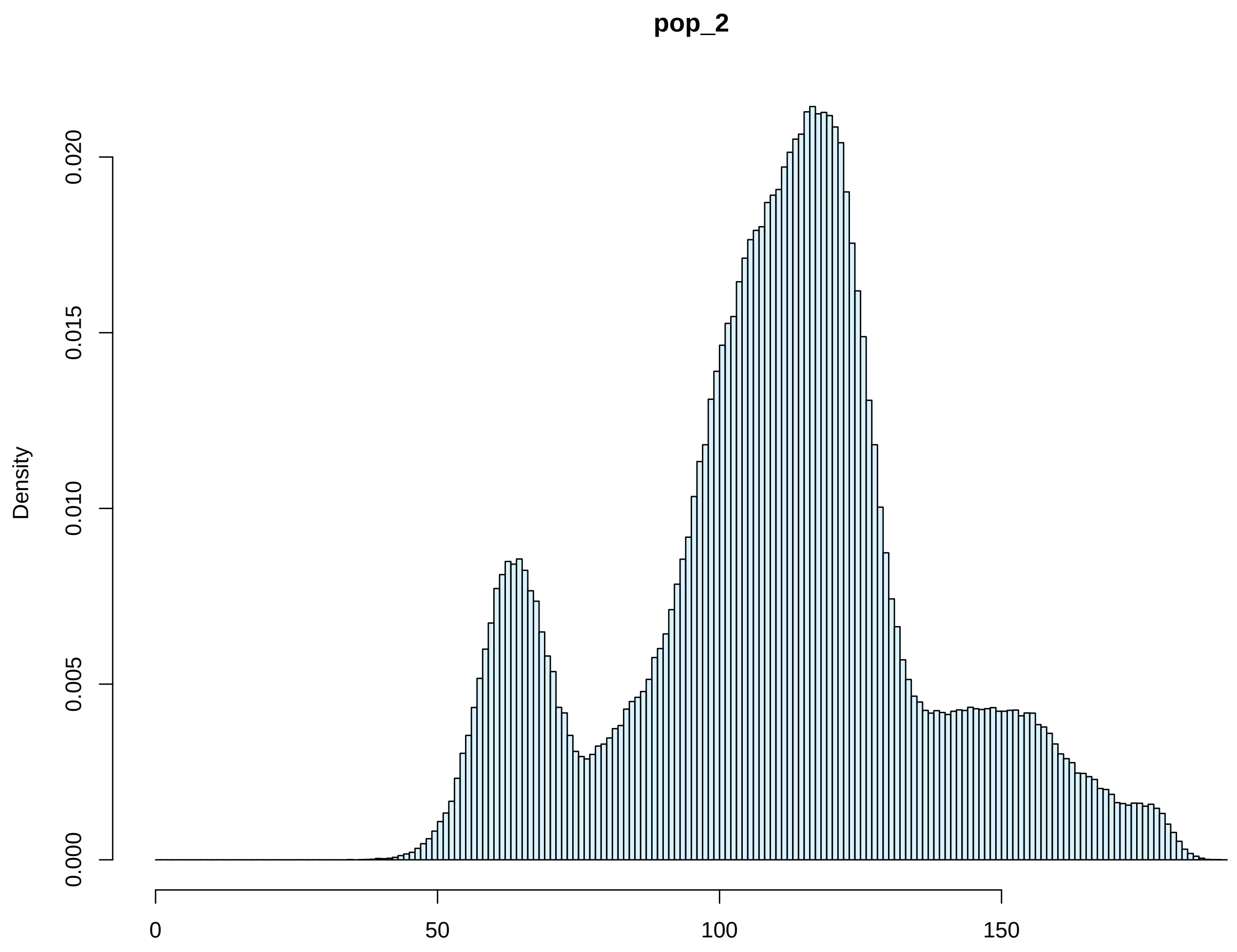
```
limx <- max(max(respop[[1]]$distance_matrix), max(ressim[[1]]$distance_matrix),
        max(ressimWS[[1]]$distance_matrix))

#graph superposition:
plot(p1, col = rgb(0,0.4,1,1), freq = FALSE, xlim = c(0,limx),
        main = paste("pop", unique(vecsim)[[1]], sep = "_"),
        xlab = "Genetic distances")
plot(p2, col = rgb(0.7,0.9,1,0.5), freq = FALSE, add = TRUE)
plot(p3, col = rgb(0.9,0.5,1,0.3), freq = FALSE, add = TRUE)

#adding a legend:
leg.txt <- c("original data","simulated data", "without selfing")
col <- c(rgb(0,0.4,1,1), rgb(0.7,0.9,1,0.5), rgb(0.9,0.5,1,0.3))
legend("top", fill = col, leg.txt, plot = TRUE, bty = "o", box.lwd = 1.5,
bg = "white")
```



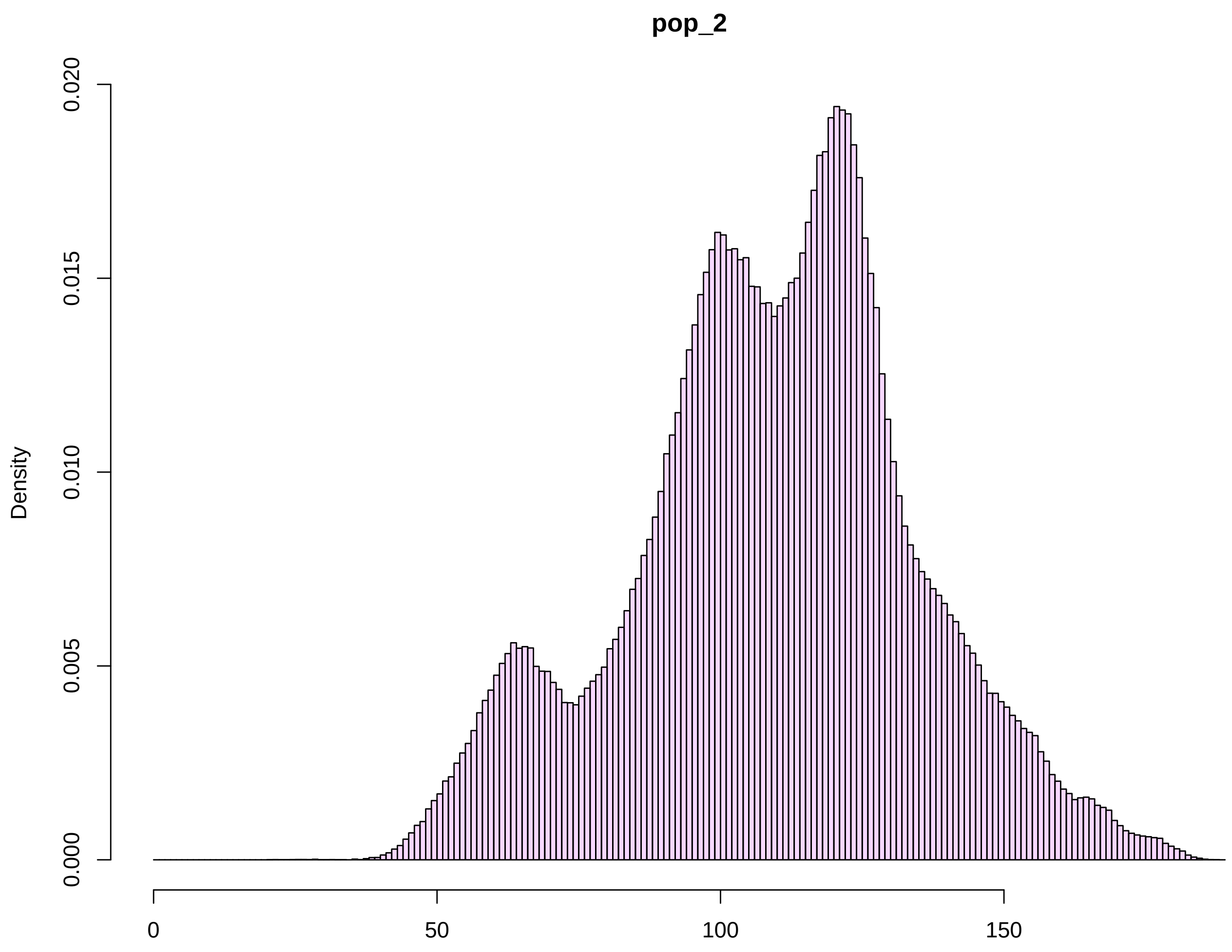
```
#second pop:
p <- 2 #useful if several populations: just change *p* and run lines

p1 <- hist(respop[[p]]$distance_matrix, freq = FALSE, col = rgb(0,0.4,1,1),
           breaks = seq(0, max(respop[[p]]$distance_matrix)+1, 1),
           main = paste("pop", p, sep = "_"), xlab = "")
```



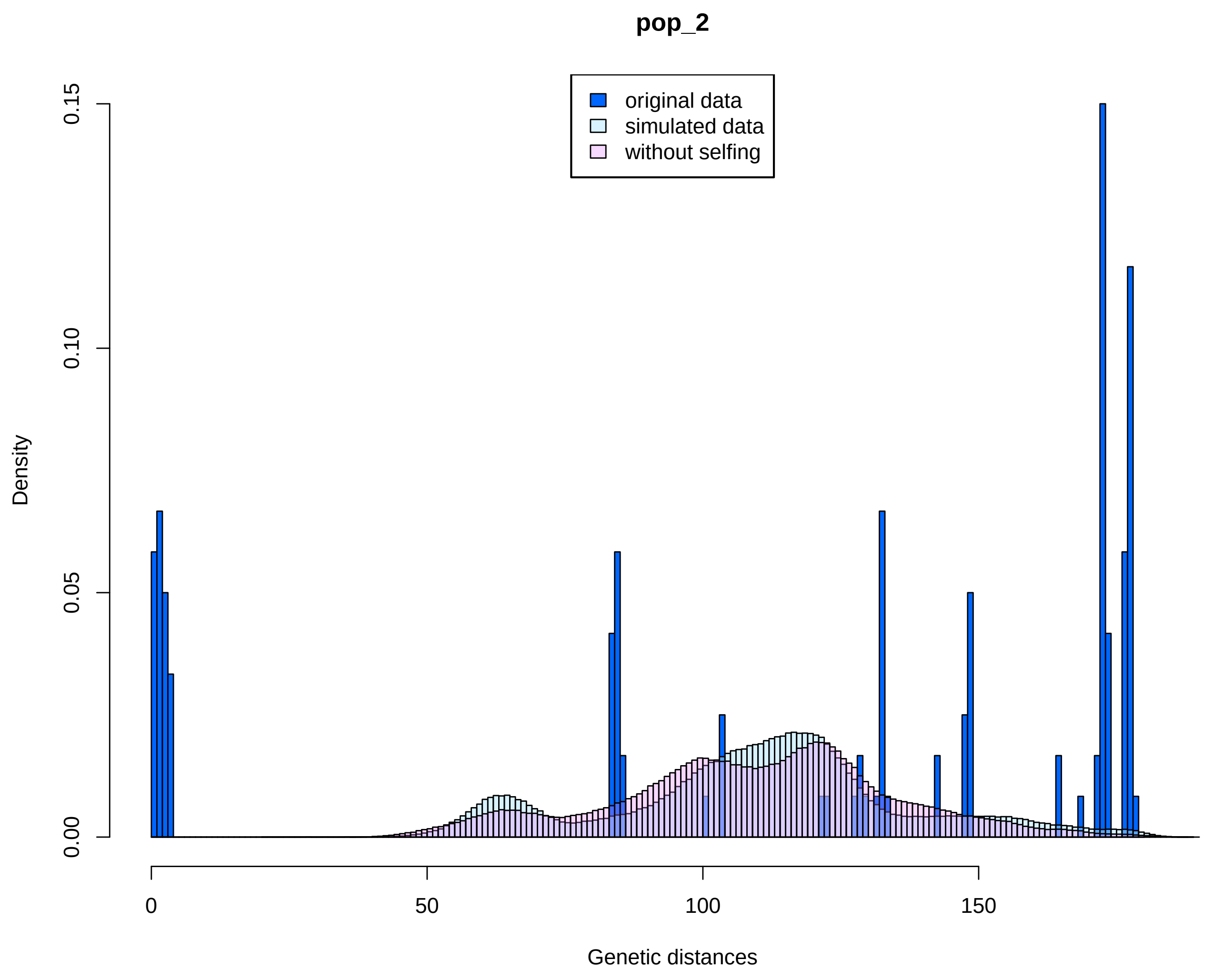
```
p2 <- hist(ressim[[p]]$distance_matrix, freq = FALSE, col = rgb(0.7,0.9,1,0.5),
           breaks = seq(0, max(ressim[[p]]$distance_matrix)+1, 1),
           main = paste("pop", p, sep = "_"), xlab = "")
```



```
p3 <- hist(ressimWS[[p]]$distance_matrix, freq = FALSE, col = rgb(0.9,0.5,1,0.3),
           breaks = seq(0, max(ressimWS[[p]]$distance_matrix)+1, 1),
           main = paste("pop", p, sep = "_"), xlab = "")
```



```
limx <- max(max(respop[[p]]$distance_matrix), max(ressim[[p]]$distance_matrix),
            max(ressimWS[[p]]$distance_matrix))

#graph superposition:
plot(p1, col = rgb(0,0.4,1,1), freq = FALSE, xlim = c(0,limx),
        main = paste("pop", unique(vecsim)[[p]], sep = "_"),
        xlab = "Genetic distances")
plot(p2, col = rgb(0.7,0.9,1,0.5), freq = FALSE, add = TRUE)
plot(p3, col = rgb(0.9,0.5,1,0.3), freq = FALSE, add = TRUE)

#adding a legend:
leg.txt <- c("original data","simulated data", "without selfing")
col <- c(rgb(0,0.4,1,1), rgb(0.7,0.9,1,0.5), rgb(0.9,0.5,1,0.3))
legend("top", fill = col, leg.txt, plot = TRUE, bty = "o", box.lwd = 1.5,
bg = "white")
```



```
#determining alpha2
table(respop[[1]]$distance_matrix)
>
>   1   2   3  82  84  85  97 100 101 104 122 128 129 133 134 135 143 149 152 165
>   2   3   2   1   3   1   1   3   1   2   1   1   2   6   1   1   1   2   1   1
> 169 170 172 173 178 179
>   1   1   1   8   7   1
#alpha2 = 3
```

```
#creating MLL list:
MLLlist <- MLL_generator(popsim, vecpop = vecsim, alpha2 = c(3,0))
##This will create a list of MLL (alpha2 = 3) and MLG (alpha2 = 0) !

#or
res <- genet_dist(popsim, vecpop = vecsim, alpha2 = c(3,0))
MLLlist <- MLL_generator2(list(res[[1]]$potential_clones,
    res[[2]]$potential_clones), MLG_list(popsim, vecpop = vecsim), vecpop = vecsim)
```

For haploid data, theoretical example:

```
respop <- genet_dist(haplodata, haploid = TRUE, vecpop = vechaplo)
ressim <- genet_dist_sim(haplodata, haploid = TRUE, vecpop = vechaplo,
                         nbrepeat = 1000)
```

```
MLLlist <- MLL_generator(haplodata, haploid = TRUE, vecpop = vechaplo,
                              alpha2 = c(3,0))
#or
res <- genet_dist(haplodata, haploid = TRUE, vecpop = vechaplo, alpha2 = c(3,0))
MLLlist <- MLL_generator2(list(res[[1]]$potential_clones, res[[2]]$potential_clones),
            haploid = TRUE, MLG_list(haplodata, vecpop = vechaplo), vecpop = vechaplo)
```

F. Genotypic diversity, richness and evenness indices calculation

F.1 Classic genotypic indices

Basic commands:

```
clonal_index(zostera, vecpop = popvec)
```

or, with MLL:

```
clonal_index(popsim, vecpop = vecsim, listMLL = MLLlist)
```

or, for haploid data:

```
clonal_index(haplodata, vecpop = vechaplo)
```

Results:

```
clonal_index(zostera, vecpop = popvec)
```

	N	G	R	H”	J’	D	V	Hill
SaintMalo	29	17	0.5714286	2.671294	0.9428497	0.9507389	0.8559028	20.300000
Arcouest	30	16	0.5172414	2.268605	0.8182264	0.8413793	0.3918367	6.304348

F.2 Pareto index

Basic commands:

```
Pareto_index(zostera, vecpop = popvec)
```

or, with MLL:

```
Pareto_index(popsim, vecpop = vecsim, listMLL = MLLlist)
```

or, for haploid data:

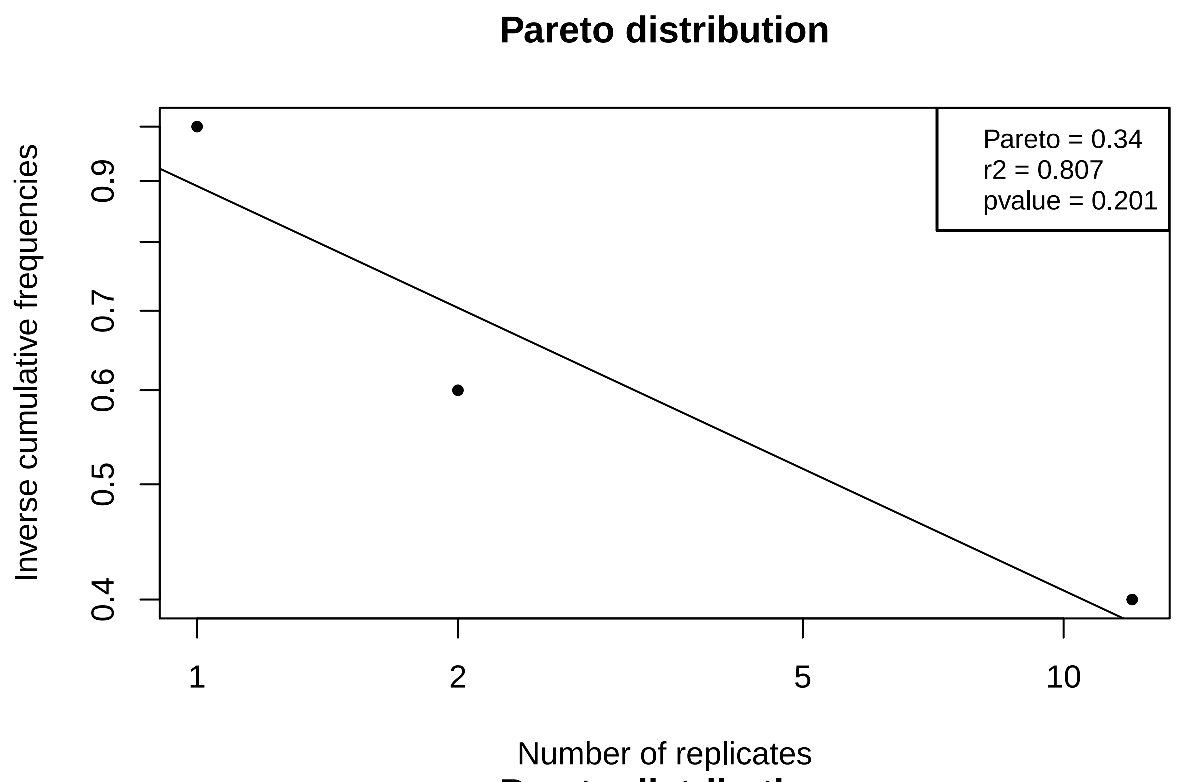
```
Pareto_index(haplodata, vecpop = vechaplo)
```

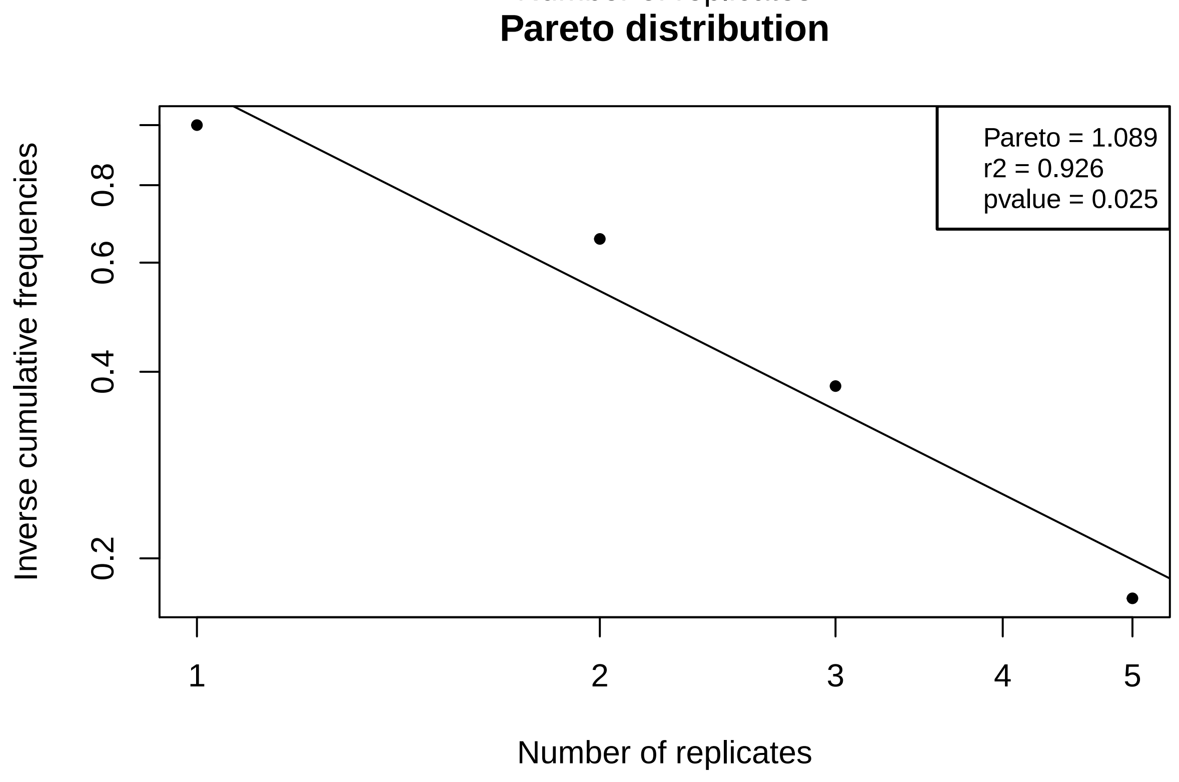
Options:

```
Pareto_index(zostera, vecpop = popvec, graph = TRUE) #classic graphic
Pareto_index(zostera, vecpop = popvec, legends = 2, export = TRUE)
                                                     #export option
Pareto_index(zostera, vecpop = popvec, full = TRUE) #all results
```

Results:

```r
res <- Pareto_index(zostera, vecpop = popvec, full = TRUE, graph = TRUE, legends = 2)
```





```r
names(res$SaintMalo)
```

```
> [1] "Pareto"              "c_Pareto"            "regression_results"
> [4] "coords_Pareto"
```

```
res$SaintMalo$Pareto
```

```
> [1] 0.3403204
```

```
res$SaintMalo$c_Pareto
```

```
> [1] 1.34032
```

```
#res$SaintMalo$regression_results
#res$SaintMalo$coords_Pareto #points coordinates
```

G. Spatial components of clonality

G.1 Spatial autocorrelation

Basic commands:

```
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE)
```

or, with MLL:

```
autocorrelation(popsim, coords = coord_sim, Loiselle = TRUE, listMLL = MLLlist)
```

or, for haploid data:

```
autocorrelation(haplodata, haploid = TRUE, coords = coord_haplo, Loiselle = TRUE)
```

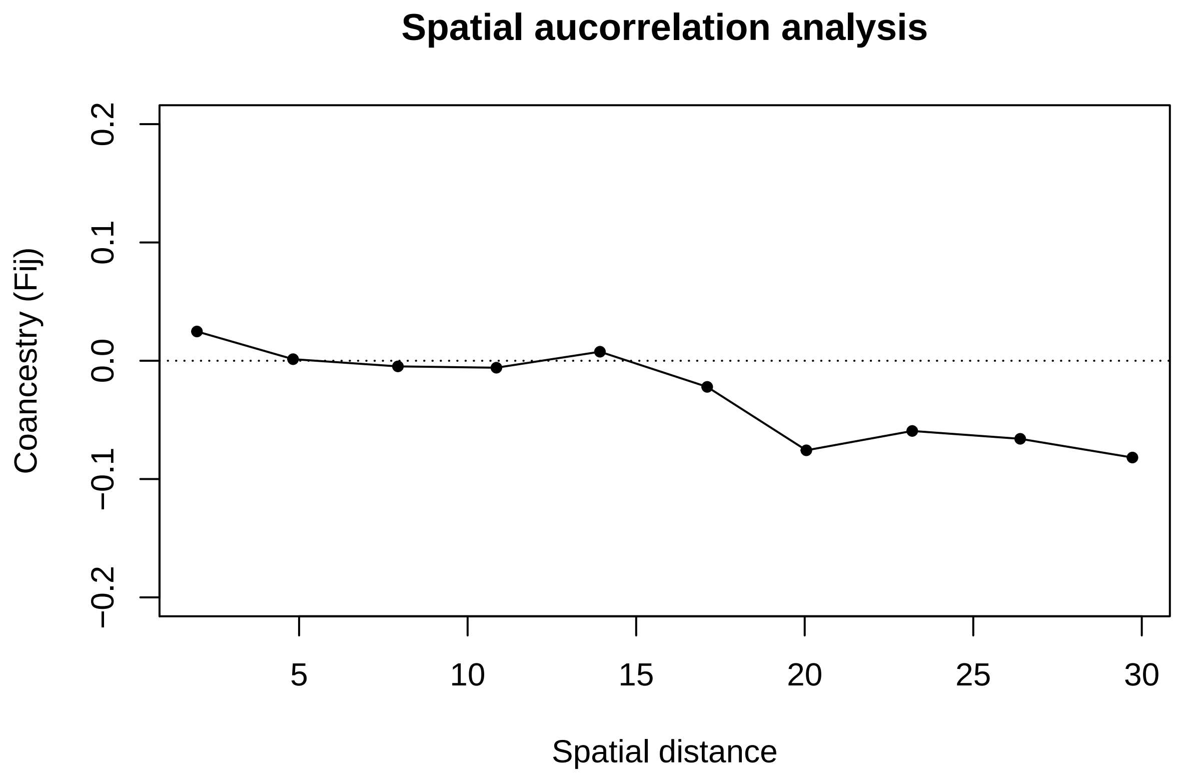
Lot's of options:

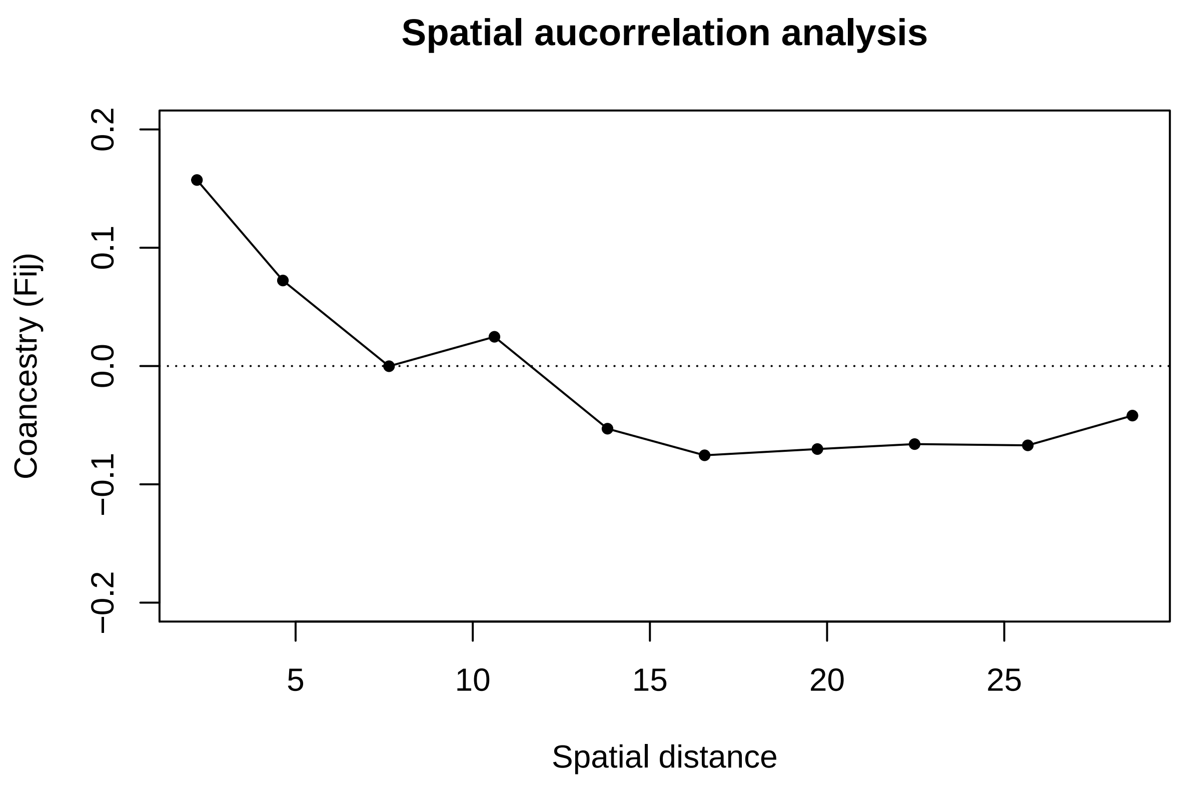
```
#kinship distances:
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE)
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Ritland = TRUE)

#ramets/genets methods:
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE)
                                                                        #ramets
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec,, Loiselle = TRUE,
                    genet = TRUE, central_coords = TRUE)
                                              #genets, central coordinates of each MLG
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE,
                    genet = TRUE, random_unit = TRUE)
                                                      #genets, one random unit per MLG
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE,
                    genet = TRUE, weighted = TRUE)
                                              #genets, with weighted matrix on kinships

#distance classes construction:
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE)
                                                              #10 equidistant classes
distvec <- c(0,10,15,20,30,50,70,76.0411074)
                                    #with 0, min distance and 76.0411074, max distance
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE,
                    vecdist = distvec) #custom distance vector
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE,
                    class1 = TRUE, d = 7) #7 equidistant classes
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Loiselle = TRUE,
                    class2 = TRUE, d = 7)
```

```
                                #7 distance classes with the same number of units in each

#graph options:
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Ritland = TRUE,
                                                graph = TRUE) #displays graph
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Ritland = TRUE,
                                                export = TRUE) #export graph

#pvalues computation
autocorrelation(zostera, coords = coord_zostera, vecpop = popvec, Ritland = TRUE,
                                                            nbrepeat = 1000)
```

Results:

```
res <- autocorrelation(zostera, coords = coord_zostera, vecpop = popvec,
                                        Ritland = TRUE, nbrepeat = 1000, graph = TRUE)
```





```r
names(res$Arcouest)
```

```
> [1] "Main_results"            "Slope_and_Sp_index"     "Slope_resample"
> [4] "Kinship_resample"        "Matrix_kinship_results" "Class_kinship_results"
> [7] "Class_distance_results"
```

```r
res$Arcouest$Main_results #enables graph reproduction
```

dist_min	dist_max	dist_mean	ln(dist_mean)	nb_pairs	mean_Ritland	pval_kin
1.11803	3.00000	2.215687	0.7955625	23	0.1572378	0.000
3.04138	6.02080	4.641553	1.5350490	35	0.0723040	0.000
6.10328	9.05539	7.636613	2.0329541	62	-0.0001518	0.292
9.12414	12.09339	10.612723	2.3620535	61	0.0247131	0.014
12.16553	15.13275	13.802695	2.6248639	58	-0.0529723	0.022
15.18223	18.06931	16.543032	2.8059650	50	-0.0754410	0.000
18.24829	21.10095	19.726507	2.9819633	54	-0.0701760	0.000
21.18962	24.08319	22.470382	3.1121981	28	-0.0659711	0.034
24.41311	27.22591	25.664587	3.2451121	20	-0.0670280	0.068
27.45906	30.26962	28.617915	3.3540329	15	-0.0418956	0.436

```r
apply(res$Arcouest$Main_results, 2, mean)[6] #mean Fij
```

```
> mean_Ritland
>   -0.0119381
```

```r
res$Arcouest$Slope_and_Sp_index #gives b and Sp indices
```

	b	b_log	Sp	Sp_log
obs_value	-0.0069054	-0.0877033	0.0081938	0.1040665
mean_sim	0.0000230	0.0002994	-0.0000088	-0.0000719
sd_sim	0.0009620	0.0107159	0.0009555	0.0106701
0.95_inf	-0.0021680	-0.0232025	-0.0015771	-0.0177145
0.95_sup	0.0016229	0.0187066	0.0021527	0.0233927
0.9_inf	-0.0017065	-0.0190059	-0.0014359	-0.0155080
0.9_sup	0.0014775	0.0162403	0.0017695	0.0190726
pval_upper	0.0000000	0.0000000	1.0000000	1.0000000
pval_lower	1.0000000	1.0000000	0.0000000	0.0000000
pval_2sides	0.0000000	0.0000000	0.0000000	0.0000000

```
#raw data:
#res$Arcouest$Slope_resample
#res$Arcouest$Kinship_resample
#res$Arcouest$Matrix_kinship_results
#res$Arcouest$Class_kinship_results
#res$Arcouest$Class_distance_results
```

G.2 Clonal subrange

Basic commands:

```
clonal_sub(zostera, coords = coord_zostera, vecpop = popvec)
```

or, with MLL:

```
clonal_sub(popsim, coords = coord_sim, listMLL = MLLlist)
```

or, for haploid data:

```
clonal_sub(haplodata, haploid = TRUE, coords = coord_haplo)
```

Options: same distance classes definition as *autocorrelation*:

```
clonal_sub(posidonia, coords = coord_posidonia) #basic, with 10 equidistant classes
distvec <- c(0,10,15,20,30,50,70,76.0411074)
                                #with 0, min distance and 76.0411074, max distance
clonal_sub(zostera, coords = coord_zostera, vecpop = popvec, vecdist = distvec)
                                                            #custom distance classes
clonal_sub(zostera, coords = coord_zostera, vecpop = popvec, class1 = TRUE, d = 7)
                                                              #7 equidistant classes
clonal_sub(zostera, coords = coord_zostera, vecpop = popvec, class1 = TRUE, d = 7)
                            #7 distance classes with the same number of units in each
```

Results:

```
res <- clonal_sub(zostera, coords = coord_zostera, vecpop = popvec)
res$Arcouest[[1]] #Global clonal subrange
> [1] 19.10497
```

```
res$Arcouest$clonal_sub_tab  #details per class
```

nb_pairs	dist_min	dist_max	dist_mean	Fr	log(Fr)
19	0.7071068	2.9154759	1.96918467368421	0.315789473684211	-0.500602350569185
49	3.2015621	6.1846584	4.82145653877551	0.326530612244898	-0.486076097372589
49	6.264982	9.2195445	7.93246008163265	0.183673469387755	-0.735953570589189
74	9.3407708	12.3693169	10.8538306148649	0.175675675675676	-0.755288367424139
77	12.5	15.435349	13.9262076792208	0.168831168831169	-0.772547372865645
70	15.5	18.5809042	17.10734185	0.128571428571429	-0.890855530574932
36	18.7216452	21.5928229	20.0492420277778	0.0833333333333333	-1.07918124604762
31	21.7082933	24.6981781	23.1899439677419	0	-Inf
25	24.8243832	27.8567766	26.392275048	0	-Inf
5	28.4121453	30.9919344	29.72321728	0	-Inf

G.3 Aggregation index

Basic commands:

```
agg_index(zostera, coords = coord_zostera, vecpop = popvec)
```

or, with MLL:

```
agg_index(popsim, coords = coord_sim, listMLL = MLLlist)
```

or, for haploid data:

```
agg_index(haplodata, coords = coord_haplo)
```

Options:

```
agg_index(zostera, coords = coord_zostera, vecpop = popvec, nbrepeat = 100)
                                                #pvalue computation
agg_index(zostera, coords = coord_zostera, vecpop = popvec, nbrepeat = 1000,
                                   bar = TRUE) #could be time consuming
```

Results:

```
res <- agg_index(zostera, coords = coord_zostera, vecpop = popvec, nbrepeat = 1000)
```

```
res$SaintMalo$results #Aggregation index
```

Ac	pval	nbrepeat
0.1965314	0.006	1000

```
#res$SaintMalo$simulation #vector of sim aggregation index
```

G.4 Edge Effect

Basic commands:

```
#for zostera, centers of quadra is at 15,10
edge_effect(zostera, coords = coord_zostera, vecpop = popvec,
                center = rep(c(15,10),2))
```

or, with MLL:

```
edge_effect(popsim, coords = coord_sim, center = rep(c(15,10),2), listMLL = MLLlist)
```

or, for haploid data:

```
edge_effect(haplodata, coords = coord_haplo, center = rep(c(15,10),2))
```

Options:

```
edge_effect(zostera, coords = coord_zostera, vecpop = popvec, center = rep(c(15,10),2),
                                                    nbrepeat = 100) #pvalue computation
edge_effect(zostera, coords = coord_zostera, vecpop = popvec, center = rep(c(15,10),2),
                                            nbrepeat = 1000, bar = TRUE) #could be time consuming
```

Results:

```
res <- edge_effect(zostera, coords = coord_zostera, vecpop = popvec,
        center = rep(c(15,10),2), nbrepeat = 100) #better put 1000 nbrepeat at least
```

```
res$SaintMalo$results #Aggregation index
```

Ee	pval_Ee	nbrepeat
0.0288465	0.798	1000

```
#res$SaintMalo$simulation #vector of sim aggregation index
```

H. BONUS: *"Ready to use"* Table

Summary function of main results:

Basic commands:

```
GenClone(zostera, coords = coord_zostera, vecpop = popvec)
```

or, with MLL:

```
GenClone(popsim, coords = coord_sim, listMLL = MLLlist)
```

or, for haploid data:

```
GenClone(haplodata, haploid = TRUE, coords = coord_haplo)
```

Options:

```
GenClone(zostera, coords = coord_zostera, vecpop = popvec, nbrepeat = 100) #pvalues
GenClone(zostera, coords = coord_zostera, vecpop = popvec, nbrepeat = 1000, bar = TRUE)
                                                            #could be time consuming
```

Results:

```r
GenClone(zostera, coords = coord_zostera, vecpop = popvec)
```

	N	Lineage	nb_L	nb_all	SE	Fis	pval_2sides	Fis_WR	pval_2sides.1
SaintMalo	29	MLG	17	2.777778	0.6620208	-0.1350152	NA	-0.03917716	NA
Arcouest	30	MLG	16	3.111111	0.6111111	-0.1017437	NA	-0.2108197	NA

	R	Pareto_index	Sp_Loiselle	pval_2sides	Sp_L_WR	pval_2sides.1	Sp_Ritland
SaintMalo	0.5714286	1.088812	0.008698691	NA	0.00542055	NA	0.008193788
Arcouest	0.5172414	0.3403204	0.007036951	NA	0.003786488	NA	0.003920472

	pval_2sides	Sp_R_WR	pval_2sides.1	H"	J'	D	V	Hill
SaintMalo	NA	0.004776727	NA	2.671294	0.9428497	0.9507389	0.8559028	20.3
Arcouest	NA	0.001466866	NA	2.268605	0.8182264	0.8413793	0.3918367	6.304348

When a locus is homozygous, it is ignored for Fis and Fis_WR values computation.